






Automatic Evolutionary Settings of Machine Learning Methods for Buildings' Thermal Loads Prediction

Gisele Goulart Tavares^(✉), Priscila V. Z. Capriles, and Leonardo Goliatt

Federal University of Juiz de Fora, Juiz de Fora, Brazil
{giselegoulart,capriles,goliatt}@ice.ufjf.br

Abstract. Due to climate change, buildings can consume 30% more energy by 2040, with energy performance being the critical element for achieving sustainable development in the civil construction sector. One way to solve this evaluation problem is by applying Machine Learning Methods that can assist specialists in civil construction in analyzing scenarios even in the initial phase of the project. The present work evaluates the application of the Elastic Net, Extreme Learning Machine, and Extreme Gradient Boosting models for the prediction of heating and cooling loads in residential buildings. The database used has 768 samples, with eight geometric input variables and two thermal output variables. Differential Evolution optimization algorithm was applied to select method parameters to find the sets of hyperparameters that reinforce the predictive capabilities of the models. The comparisons of the results occurred using the metrics MAE, MAPE, RMSE, and R^2 . The results showed that the Extreme Gradient Boosting method obtained a better performance among the tested methods than the literature, presenting the lowest values for the error metrics and significant differences in the statistical tests. Thus, combining Differential Evolution and Extreme Gradient Boosting methods, thermal loads can be predicted, assisting projects that aim at energy savings and sustainability

Keywords: Energy efficiency · Heating and cooling loads · Extreme Gradient Boosting · Load forecast

1 Introduction

The increase in population and the growing use of new technologies have resulted in the emergence of greater energy demands, leading to a rise in consumption of around 30% by 2040 [1]. In Brazil, only in the residential sector, it is estimated that the possession of air conditioning by families has more than doubled between 2005 and 2017, arousing interest in strategies aimed at reducing energy consumption coupled with the maintenance of environmental comfort [2]. In this context, energy savings are increasingly necessary to reduce their generation's environmental and social impacts.

The energy performance of buildings is highlighted as a key element for achieving sustainable development since it can reduce about 20% of greenhouse gas emissions and 20% of primary energy savings [3]. In commercial buildings, due to the negative influence that an uncomfortable environment causes on users' performance, the search for thermal comfort associated with low energy consumption has resulted in an increase in research in this field [4]. The continued use of electronic equipment and the high density of people in offices increase the challenge of maintaining these thermally comfortable environments.

To obtain the best performance of construction, three factors must be considered: architectural design, heating, cooling systems, and occupation. The architectural project is developed iteratively, with a team that reviews all aspects of the building and rethinks about decisions related to architecture. With a highly optimized project in hand, specialists in civil construction can reduce cooling and heating systems' capacity and minimize the need for this set of services.

However, despite the architectural design being an essential aspect of the building's performance, determining materials and configurations that optimize consumption and comfort in the structure is not easy. Considerations about the location, ventilation strategies, lighting, and materials to be used, increase the complexity of designing an energy-efficient project.

In this scenario, studies that consider computer simulations of models that deal with buildings' consumption are becoming more and more present. Several approaches are proposed for computational models of buildings, such as:

- The use of neural networks and support vector machines to predict the use of electricity in home heating systems [5];
- The generation of a database through residence parametrization to use machine learning algorithms to forecast heating and cooling loads [6];
- The application of genetic algorithms to minimize energy consumption and discomfort hours in a typical Italian residence simulated in different climatic zones [7].

The literature presents a diversity of works that carried out the modeling and simulation of different scenarios with various architectural types. However, the alternatives are often tested one by one, separately, and the results refer to comparisons between generated outputs. This process requires numerous tests and considerable execution time, turning to analyze many variables simultaneously, unviable. Thus, the combination of optimization methods and intelligent algorithms has shown promise [8]. The integration of these methods can improve the predictions related to the energy market, helping service providers understand different consumers' and users' demands to save energy by knowing their usage habits. Predicting the building's behavior based on design parameters can assist in decision-making by specialists, so manual and operationally costly analysis is unnecessary.

The present work aims to propose a combined method of machine learning and evolutionary algorithms to predict thermal loads (heating and cooling loads) in civil construction. The maximization of the regression methods' predictive performance is sought by optimizing hyperparameters of the Elastic-Net

Regression (NET), Extreme Learning Machine (ELM), and Extreme Gradient Boosting (XGB) techniques.

2 Methods

2.1 Dataset

The dataset used in this paper can be found in [9]. The dataset is composed by eight input variables and two output variables. The input variables are: relative compactness (RC), surface area, wall area, roof area, overall height, orientation, glazing area and glazing area distribution. The output variables are the heating loadings (HL) and cooling loadings (CL). Heating/cooling loads refer to the amount of thermal energy that would need to be added/removed from a space to keep the temperature in an acceptable range. To generate different building shapes, eighteen such elements were used according to Fig. 1(A). A subset of twelve shapes with distinct RC values was selected for the simulations as can be seen in Fig. 1(B).

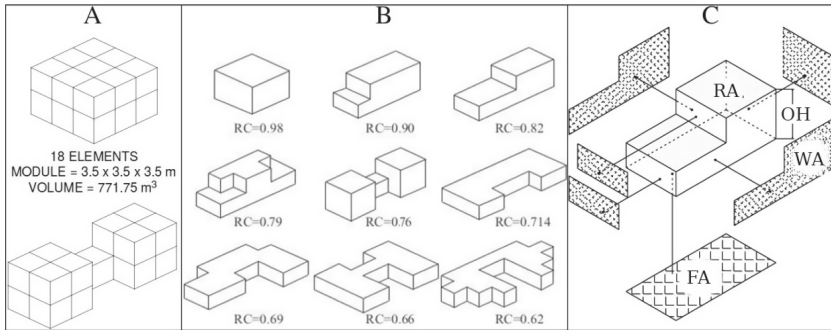


Fig. 1. A: Generation of shapes based on eighteen cubical elements [9]. B: Examples of building shapes [10]. C: Generic definition of building areas, where OH is the Overall Height, RA is the Roof Area, WA is the Wall Area and FA is the Floor Area. Adapted from [11].

2.2 Machine Learning Methods

Elastic Net Regression. The Elastic Net technique is an extension of the LASSO method, robust to correlations between the predictors [12]. Elastic Net uses a mix of $L1$ (LASSO) and $L2$ (Ridge) penalties and can be formulated as:

$$\hat{\beta}(enet) = \left(1 + \frac{\lambda_2}{n}\right) \left\{ \arg \min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda_2 \|\beta\|_2^2 + \lambda_1 \|\beta\|_1 \right\} \quad (1)$$

By setting $\alpha = \lambda_2/(\lambda_1 + \lambda_2)$, the estimator is equivalent to minimizing:

$$\hat{\beta}(enet2) = \arg \min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2, \text{ subject to } P_{\alpha}(\beta) = (1-\alpha) \|\beta\|_1 + \alpha \|\beta\|_2^2 \leq s \text{ for some } s \quad (2)$$

where $P_{\alpha}(\beta)$ is the penalty Elastic Net [13]. The method is simplified to a Ridge regression when $\alpha = 1$ and to a LASSO regression when $\alpha = 0$. The $L1$ penalty, part of the NET method, makes the automatic selection of variables. In contrast, the $L2$ part encourages the grouped selection and stabilizes the solution paths about random sampling, thus improving the forecast. By inducing a grouping effect during the selection of variables, the method can select groups of correlated characteristics when the groups are not known in advance.

Extreme Learning Machine. Extreme Learning Machine (ELM) [14] is a feedforward artificial neural network, which has a single hidden layer. Compared with the Artificial Neural Network, the Support Vector Machine and other traditional prediction models, the ELM model retains the advantages of fast learning, good ability to generalize and convenience in terms of modeling. In ELMs there are three levels of randomness [15]: (i) fully connected, hidden node parameters are randomly generated; (ii) the connection can be randomly generated, not all input nodes are connected to a particular hidden node; (iii) a hidden node itself can be a subnetwork formed by several nodes resulting in learning local features; The output function of ELM used in this paper is given by

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^L \beta_i G(\alpha, \gamma, \mathbf{w}_i, b_i, \mathbf{c}, \mathbf{x}) = \sum_{i=1}^L \beta_i G(\alpha \text{MLP}(\mathbf{w}_i, b_i, \mathbf{x}) + \gamma(1 - \alpha) \text{RBF}(\mathbf{x}, \mathbf{c})) \quad (3)$$

where \hat{y} is the ELM prediction associated to the input vector \mathbf{x} , \mathbf{w}_i is the weight vector of the i -th hidden node, b_i are the biases of the neurons in the hidden layer, β_i are output weights, \mathbf{c} is the vector of centers. MLP and RBF are the input activation functions, respectively, while α is a user-defined that multiplies MLP(\cdot) and RBF(\cdot) terms. $G(\cdot)$ is the nonlinear output activation function and L is the number of neurons in the hidden layer. The output activation functions $G(\alpha, \mathbf{w}_i, b_i, \mathbf{c}, \mathbf{x})$ with the hidden nodes weights (\mathbf{w}, b) are presented in Table 1.

Table 1. ELM activation functions.

#	Name	Activation function G
1	Identity	$G(x) = x$
2	ReLU	$G(x) = \max(0, x_i; i = 1, \dots, D)$
3	Swish	$G(x) = x/(1 + \exp(-x))$
4	Gaussian	$G(x) = \exp(-x^2)$
5	Multiquadric	$G(x) = \sqrt{x^2 + b^2}$
6	Inverse multiquadric	$G(x) = 1/(x^2 + b^2)^{1/2}$

The parameters (\mathbf{w}, b) are randomly generated (normally distributed with zero mean and standard deviation equals to one), and weights β_i of the output layer are determined analytically, while MLP and RBF are written as

$$\text{MLP}(\mathbf{w}_i, b_i, \mathbf{x}) = \sum_{k=1}^D w_{ik} x_k + b_i \quad \text{and} \quad \text{RBF}(\mathbf{x}, \mathbf{c}) = \sum_{j=1}^D \frac{x_j - c_{ij}}{r_i} \quad (4)$$

where D is the number of input features, the centers c_{ij} are taken uniformly from the bounding hyperrectangle of the input variables and $r = \max(\|\mathbf{x} - \mathbf{c}\|)/\sqrt{2D}$.

The output weight vector $[\beta_1, \dots, \beta_L]$ can be determined by minimizing the approximation error [15]

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^L} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{y}\| \quad (5)$$

where \mathbf{y} is the output data vector, \mathbf{H} is the hidden layer output matrix

$$\mathbf{H} = \begin{bmatrix} G_1(\alpha, \gamma, \mathbf{w}_1, b_1, \mathbf{c}, \mathbf{x}_1) & \cdots & G_L(\alpha, \gamma, \mathbf{w}_L, b_L, \mathbf{c}, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G_1(\alpha, \gamma, \mathbf{w}_1, b_1, \mathbf{c}, \mathbf{x}_N) & \cdots & G_L(\alpha, \gamma, \mathbf{w}_L, b_L, \mathbf{c}, \mathbf{x}_N) \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \quad (6)$$

is the output data vector with N the number of data points. The optimal solution is given by

$$\boldsymbol{\beta} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{y} = \mathbf{H}^\dagger \mathbf{y} \quad (7)$$

where \mathbf{H}^\dagger is the pseudoinverse of \mathbf{H} .

Gradient Boosting Machines. In several problems the goal is, using a training set $\{(x_i, y_i)\}_{i=1}^N$ with N samples, to find an approximation $\hat{f}(x)$ to a function $f(x)$ that minimizes the expected value of the loss function

$$L(y, \hat{f}(x)) = \sum_i^N [y_i - \hat{f}(x_i)]^2. \quad (8)$$

GB approximates f by an additive expansion of the form $\hat{f} = \sum_{m=1}^M \beta_m h(x, a_m)$ where the functions $h(x, a)$ are $h(x, a_m)$ is an K -node regression tree and the parameters $\{\beta, a\}$ are jointly fit to the training data in a forward stage wise manner [16]. At each iteration m , a regression tree partitions the variable space into disjoint regions $\{R_{km}\}_{k=1}^K$ at the m th iteration. A constant γ_{jm} is assigned to each such region and the predictive rule is $x \in R_{jm} \Rightarrow f(x) = \gamma_{jm}$. Using the indicator notation, the output of h for input x can be written as

$$h(x, \{R_{km}\}_{k=1}^K) = \sum_{k=1}^K \gamma_{km} I(x \in R_{km}), \quad I(\cdot) = 1 \text{ if } x \in R_{km} \text{ else } 0 \quad (9)$$

with parameters $\{R_{km}, \gamma_{km}\}$, $k = 1, 2, \dots, J$, $m = 1, \dots, M$, where γ_{km} is the value predicted in the region R_{km} .

As the model (9) predicts a constant value in each region R_{km} , the solution reduces to

$$\gamma_{km} = \arg \min_{\gamma} \sum_{x_i \in R_{km}} L(y_i, f_{m-1}(x_i) + \gamma), \quad \gamma \text{ constant.} \tag{10}$$

The current approximation $f_{m-1}(x)$ is then updated following the rule

$$\hat{f}_m(x) = \hat{f}_{m-1}(x) + \lambda \sum_{k=1}^K \gamma_{km} I(x \in R_{km}) \tag{11}$$

where parameter $0 < \lambda \leq 1$ is called the learning rate.

A substantial improvement in Gradient Boosting’s accuracy can be achieved when at each iteration of the algorithm the base learner is fitted on a subsample of the training set drawn at random without replacement. Subsample size is some constant fraction of the size of the training set. Smaller values of subsample introduce randomness into the algorithm and help prevent overfitting [17]. The algorithm also becomes faster, because regression trees have to be fit to smaller datasets at each iteration.

The XGB method follows the same principles as the GB, with some differences in details of the modeling that perform more accurate approximations using the second-order derivative of the loss function (in the case of the logistic function), L1, and L2 regularization and parallel computing. XGB is the most regularized form of GB, using regularization similar to those of the Elastic Net method, which improve the generalization capabilities of the model. It presents better computational performance due to being able to perform faster training that can be distributed through different cores [18]. Uses improved data structures for better utilization of the processor’s cache memory, which makes it faster.

2.3 Model Selection Based on Differential Evolution

Setting the parameters of an estimator is usually a difficult task. Often, these parameters are defined empirically, by testing different settings by hand. An alternative is the use of population-based evolutionary algorithms, such as Differential Evolution (DE) [19]. DE is one of the most efficient evolutionary algorithms (EAs) [20]. The basic strategy of DE consists in applying weighted and stochastic vector operations between the candidate solutions set [21]. Given a population of NP vectors $\{\theta_i | i = 1, 2, \dots, NP\}$, at each iteration J ($J = 1, 2, \dots, J_{max}$) of the DE, the following operations will be performed on such vectors:

1. Mutation: For each vector θ_i , a mutant vector \mathbf{v}_i is generated according to

$$\mathbf{v}_i = \theta_{r_1} + \mathbf{F}(\theta_{r_2} - \theta_{r_3}) \tag{12}$$

where r_1, r_2 and $r_3 \in \{1, 2, \dots, NP\}$ are randomly chosen indexes, mutually different and different from i , and $F \in (0, 2)$ is a user-defined parameter.

2. Crossover: In this step, the D -dimensional trial vector μ_i will be generated by a stochastic operation given by

$$\mu_{i,j} = \begin{cases} v_{i,j}, & \text{if } \text{rand}(j) \leq CR \text{ or } j = \text{rand}(i) \\ \theta_{i,j}, & \text{if } \text{rand}(j) > CR \text{ and } j \neq \text{rand}(i) \end{cases} \quad (13)$$

where $j = 1, 2, \dots, D$, $v_{i,j}$ is the value of j -th variable of vector v_i produced by Eq. (12), $\text{rand}(j)$ is the j -th random value in the range $[0, 1]$, $\text{rand}(i) \in \{1, 2, \dots, D\}$ is a random integer value produced for each solution and CR is a user-defined parameter in the range $[0, 1]$.

3. Selection: If vector μ_i is better than θ_i , then θ_i will be replaced by μ_i in the set (population). Otherwise, the old value θ_i will be maintained.

DE was applied here to find the best hyperparameters for NET, ELM, and XGB models for predict thermal loads in buildings. Each candidate solution θ_i encodes an estimator. Each vector θ_i is composed by 2 variables to NET ($D = 2$), 4 variables to ELM ($D = 4$), and 6 variables to XGB ($D = 6$), that correspond to the total of parameters to be adjusted (Table 2). Considering the DE approach, the goal is to find a candidate solution so that the method generates computed outputs that match the outputs of the training data.

Table 2. Hyperparameters sets used in model selection step.

NET		ELM		XGB	
Parameters	Sets	Parameters	Sets	Parameters	Sets
<code>lr_ratio</code>	[0,1]	<code>n_hidden</code>	1, 2, 3, ... , 500	<code>learning_rate</code>	[0,1]
<code>alpha</code>	[0.1,1]	<code>rbf_width</code>	[0.01, 10]	<code>n_estimators</code>	10, 11, 12, ... , 900
<code>max_iter</code>	1000	<code>activation_func</code>	identity, relu, swish, gaussian, multiquadric, inv_multiquadric	<code>colsample_bytree</code>	[0,1]
<code>tol</code>	0.0001	<code>alpha</code>	[0, 1]	<code>min_child_weight</code>	1, 2, 3, ... , 10
<code>normalize</code>	false			<code>subsample</code>	[0,1]
				<code>max_depth</code>	1, 2, 3, ... , 30
				<code>objective</code>	squared_error

3 Computational Experiments

In this section, we present the results obtained for the regressions models described in Sect. 2. We ran each experiment 30 times using 10-fold cross-validation with shuffled data generated by different random seeds. The K -fold validation reduces the variation in estimating the model's performance for different data samples. Because of this, the performance becomes less sensitive to the partitioning of the data.

The experiments were conducted in Python language (3.5 version) and using scikit-learn framework [22] and XGBoost library [18]. The experiments were

conducted on computers with the following specifications: Intel (R) Xeon (R) E5620 CPU (8 cores of 2.40 GHz and 2 MB cache memory), 8 GB RAM, and Linux Ubuntu 14.04 LTS operating system. In order to evaluate the predictive performance of each model we have used the evaluation metrics shown in Table 3.

Table 3. Performance metrics: \hat{y}_i is the estimated target output, y_i is the corresponding target output, N is the number of samples, p is the number of model parameters, and \bar{y} is the mean of the vector $[y_1, \dots, y_N]$.

Metric	Expression
R^2	$1 - \frac{\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{N-1} (y_i - \bar{y})^2}$
RMSE	$\sqrt{\frac{1}{N} \sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}$
MAPE	$100 \times \frac{1}{N} \sum_{i=0}^{N-1} \frac{ y_i - \hat{y}_i }{ y_i }$
MAE	$\frac{1}{N} \sum_{i=0}^{N-1} y_i - \hat{y}_i $

Figure 2 illustrates the values of the four statistical measures averaged in 30 runs for the predicted heating loads and cooling loads. In each bar, the vertical black line indicates the standard deviation. In the MAE, RMSE, and MAPE metrics, the lower values indicate better performance, and for the coefficient R^2 , the best models should present their value closer to 1. By observing the metrics, it is possible to verify that the XGB method achieved better results, both for heating and cooling loads. The ELM method's heating loads obtained values slightly close to those achieved by the XGB in the metrics MAE, RMSE, and R^2 . For all the metrics presented, it is possible to notice a common behavior among the three tested methods, which is the best average performance in predicting heating loads.

Table 4. Parameter distribution - heating loads.

Model	Hyperparameters	Min.	Median	Max.	Average	DP
NET	<code>alpha</code>	0.114	0.189	0.193	0.186	0.014
ELM	<code>n_hidden</code>	190.000	259.500	314.000	254.867	38.684
	<code>rbf_width</code>	0.010	0.021	0.067	0.028	0.018
	<code>alpha</code>	0	0.001	0.534	0.129	0.163
XGB	<code>learning_rate</code>	0.043	0.159	0.422	0.183	0.100
	<code>n_estimators</code>	259.000	720.500	879.000	654.633	204.629
	<code>colsample_bytree</code>	0.630	0.719	0.867	0.728	0.065
	<code>subsample</code>	0.915	0.976	0.999	0.928	0.022

Table 4 shows the distributions of some hyperparameters of the methods tested over the 30 runs to predict heating loads. For NET, `alpha` values were

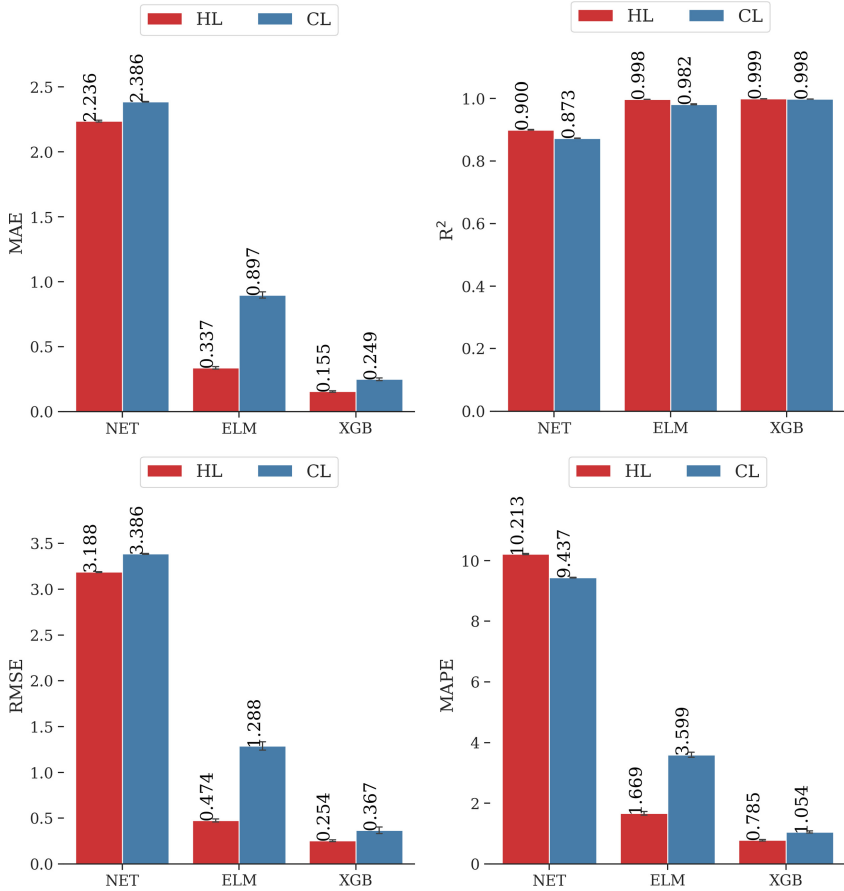


Fig. 2. Barplots for the statistical measures (averaged over 30 runs) for HL and CL. The performance metrics are Mean Absolute Error (MAE), Coefficient of Determination (R^2), Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE).

distributed in the interval $[0.114, 0.193]$ with mean 0.186, standard deviation 0.014, and median 0.189, and `l1_ratio=1` was selected in 29 out of 30 runs.

For the ELM method, it is possible to notice that the parameter `alpha` varied in the range $[0; 0.53]$ while the parameter `rbf_width`, despite having a larger range of possibilities $([0.01, 10])$ varied only between the values $[0.010, 0.067]$, indicating that the interval used in the optimization algorithm can be reduced and thus consequently decrease the search space. The number of neurons in the hidden layer of ELM varied between 190 and 314, with an average of 254.87 and a median of 259.5. The most frequent activation function was `gaussian`, being selected in 20 out of 30 executions.

In the case of the XGB method, the hyperparameter `colsample_bytree` presented values in the range $[0.630, 0.867]$, with a mean of 0.728 and a median of

Table 5. Parameter distribution - cooling loads.

Model	Hyperparameters	Min.	Median	Max.	Mean	DP
NET	<code>alpha</code>	0.124	0.128	0.134	0.128	0.002
ELM	<code>n_hidden</code>	304.00	365.500	410.00	360.967	25.026
	<code>rbf_width</code>	0.064	0.182	9.996	1.378	2.884
	<code>alpha</code>	0.001	0.122	0.941	0.293	0.337
XGB	<code>learning_rate</code>	0.150	0.439	0.978	0.463	0.261
	<code>n_estimators</code>	189.000	829.500	894.000	769.933	161.142
	<code>colsample_bytree</code>	0.382	0.452	0.604	0.475	0.069
	<code>subsample</code>	0.812	0.956	0.995	0.945	0.046

0.719 for heating loads. The `learning_rate` had its values distributed between [0.043, 0.422], mean 0.183 and median 0.159, while the `subsample` had values very close to 1, with a distribution between [0.915, 0.999], mean 0.968 and median 0.976. The `n_estimators`, which has a range of variation [10] obtained an average of 654.633 and a median of 720.500. For the parameter `min_child_weight` the value 1 was selected in 29 out of 30 executions and for the `max_depth` the value 7 was more frequent, being selected in 16 out of 30 executions.

Table 5 shows the distribution of some hyperparameters of the methods tested over the 30 runs to predict cooling loads. For the NET method, `l1_ratio` = 1 it was chosen by the optimization algorithm in the 30 executions, while for the parameter `alpha` the assigned values were distributed in the interval [0.124, 0.134] with a mean of 0.128 and a median of 0.128, indicating that values close to 0.128 improve the predictive performance of the NET method for cooling loads.

In the case of ELM, the values for `rbf_width` were distributed in the range [0.064, 9.996], with an average of 1.378 and a median of 0.182. The values of `alpha` were distributed in [0.001, 0.941], with an average of 0.293 and a median of 0.122, covering most of the range of possibilities [0, 1]. In the XGB method, the parameter `colsample_bytree` presented distribution in the interval [0.382, 0.604], with a mean of 0.475 and a median of 0.452. The `subsample` had distribution in the range [0.812, 0.995], with a mean of 0.475 and a median of 0.452, while the `learning_rate` was distributed in the range [0.150, 0.978], with a mean of 0.463 and a median of 0.439. The `n_estimators` showed a distribution concentrated in values close to the upper limit used by the evolutionary algorithm, with values in the range [189.000, 894.000], with an average of 769.933 and a median of 829.500. For `min_child_weight` the value 2 presented a higher frequency, being returned in 17 out of 30 executions, while for `max_depth` the frequencies were distributed in the interval [6], with the highest frequency occurring at value 7 in 6 out of 30 runs.

Table 6. Comparison between the results obtained from the best model of this study.

(a) Heating Loads					(b) Cooling Loads				
Reference	MAE (kW)	RMSE (kW)	MAPE (%)	R ²	Reference	MAE (kW)	RMSE (kW)	MAPE (%)	R ²
[9]	0.510	–	2.180	–	[9]	1.420	–	4.620	–
[23]	0.340	0.460	–	1.000	[23]	0.680	0.970	–	0.990
[10]	0.236	0.346	–	0.999	[10]	0.890	1.566	–	0.986
[24]	0.380	–	0.430	–	[24]	0.970	–	3.400	–
[11]	0.315	0.223	1.350	0.998	[11]	0.565	0.837	2.342	0.991
[25]	0.262	0.404	1.395	0.998	[25]	0.486	0.763	1.924	0.994
[26]	0.224	0.341	1.114	0.999	[26]	0.491	0.722	1.973	0.994
[27]	0.175	0.265	0.913	0.999	[27]	0.307	0.461	1.197	0.998
DE+NET	2.202	3.178	10.157	0.901	DE+NET	2.384	3.387	9.416	0.873
DE+ELM	0.329	0.329	1.573	0.998	DE+ELM	0.861	1.222	3.434	0.986
DE+XGB	0.150	0.243	0.753	0.999	DE+XGB	0.231	0.327	0.983	0.999

Table 6 present the statistical measures for the best models (along 30 runs) found in this paper. To provide a comparison with other models in the literature, we also show the results collected from other studies that used the same dataset employed in this paper. Reference [9] implemented random forests, while [23] developed multivariate adaptive regression splines and gaussian processes were used in [25]. Reference [10] implemented a linear combination of two or more machine learning models. The results presented [24] were obtained by genetic programming, an automated learning of computer programs using a process inspired by biological evolution. The results in [11] were obtained using Random Forests and Multilayer Perceptron Neural Networks. Reference [26] implemented Gradient Boosting Machines and [27] used Extreme Gradient Boosting. As can be seen in Table 6 for the heating loads, DE+XGB obtained competitive results. For cooling loads, DE+XGB model reaches the best average performance for all statistical measures reflecting its ability to learn highly nonlinear relationships from data.

4 Conclusion

This paper evaluated the prediction of heating and cooling loads in buildings. For it, NET, ELM, and XGB models were used, coupled to the Differential Evolution algorithm to optimize the hyperparameters of the models. The use of the evolutionary algorithm in conjunction with the machine learning methods showed satisfactory results compared to the data in the literature. The XGB model achieved the best results for all the metrics tested, considering the three models tested in HL and CL. In addition, it obtained competitive results with recent works in the literature for heating loads and better performance in all metrics when approaching cooling loads. As future work, for the XGB model, which presented a better overall performance, it is proposed to apply dimensionality reduction methods to analyze the importance of each of the input variables, making it possible to improve the computational performance. Also, we purpose to test the DE + ELM method with a greater number of iterations for optimization to analyze the convergence of the method and check if it is possible to find lower error values.

References

1. Pérez-Lombard, L., Ortiz, J., Pout, C.: A review on buildings energy consumption information. *Energy Buildings* **40**(3), 394–398 (2008)
2. Ministério de Minas e Energia do Brasil: Uso de ar condicionado no setor residencial brasileiro: Perspectivas e contribuições para o avanço em eficiência energética. Technical report (2018)
3. Boermans, T., Grözinger, J.: Economic effects of investing in EE in buildings - the beam2 model. Background paper for EC Workshop on Cohesion policy (2011)
4. Touzani, S., Granderson, J., Fernandes, S.: Gradient boosting machine for modeling the energy consumption of commercial buildings. *Energy Buildings* **158**, 1533–1543 (2018)
5. Wang, Z., Srinivasan, R.S., Shi, J.: Artificial intelligent models for improved prediction of residential space heating. *J. Energy Eng.* **142**(4), 04016006 (2016)
6. Jihad, A.S., Tahiri, M.: Forecasting the heating and cooling load of residential buildings by using a learning algorithm “gradient descent”, morocco. *Case Stud. Thermal Eng.* **12**, 85–93 (2018)
7. Ascione, F., Bianco, N., Mauro, G.M., Napolitano, D.F.: Building envelope design: Multi-objective optimization to minimize energy consumption, global cost and thermal discomfort. application to different italian climatic zones. *Energy* **174**, 359–374 (2019)
8. AlFaris, F., Juaidi, A., Manzano-Agugliaro, F.: Intelligent homes’ technologies to optimize the energy performance for the net zero energy home. *Energy Buildings* **153**, 262–274 (2017)
9. Tsanas, A., Xifara, A.: Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings* **49**, 560–567 (2012)
10. Chou, J.S., Bui, D.K.: Modeling heating and cooling loads by artificial intelligence for energy-efficient building design. *Energy Buildings* **82**, 437–446 (2014)
11. Duarte, G.R., Fonseca, L., Goliatt, P., Lemonge, A.: Comparison of machine learning techniques for predicting energy loads in buildings. *Ambiente Construído* **17**(3), 103–115 (2017)
12. Friedman, J., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. *J. Stat. Softw.* **33**(1), 1 (2010)
13. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. *J. Roy. Stat. Soc. Ser. B (Statistical Methodology)* **67**(2), 301–320 (2005)
14. bin Huang, G., yu Zhu, Q., kheong Siew, C.: Extreme learning machine: a new learning scheme of feedforward neural networks, pp. 985–990 (2006)
15. Huang, G., Huang, G.B., Song, S., You, K.: Trends in extreme learning machines: A review. *Neural Netw.* **61**, 32–48 (2015)
16. Hastie, T., Tibshirani, R., Friedman, J.: *The elements of statistical learning: data mining, inference, and prediction*, springer series in statistics (2009)
17. Friedman, J.H.: Stochastic gradient boosting. *Comput. Stat. Data Anal.* **38**(4), 367–378 (2002)
18. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. In: *Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. ACM (2016)
19. Storn, R., Price, K.: Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.* **11**(4), 341–359 (1997)

20. Kachitvichyanukul, V.: Comparison of three evolutionary algorithms: GA, PSO, and DE. *Ind. Eng. Manage. Syst.* **11**(3), 215–223 (2012)
21. Zhu, Q.Y., Qin, A., Suganthan, P., Huang, G.B.: Evolutionary extreme learning machine. *Pattern Recogn.* **38**(10), 1759–1763 (2005)
22. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
23. Cheng, M.Y., Cao, M.T.: Accurately predicting building energy performance using evolutionary multivariate adaptive regression splines. *Appl. Soft Comput.* **22**, 178–188 (2014)
24. Castelli, M., Trujillo, L., Vanneschi, L., Popovič, A.: Prediction of energy performance of residential buildings: a genetic programming approach. *Energy Buildings* **102**, 67–74 (2015)
25. Goliatt, L., Capriles, P., Duarte, G.R.: Modeling heating and cooling loads in buildings using gaussian processes. In: 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1–6. IEEE (2018)
26. Goliatt, L., Capriles, P.V.Z., Goulart Tavares, G.: Gradient boosting ensembles for predicting heating and cooling loads in building design. In: Moura Oliveira, P., Novais, P., Reis, L.P. (eds.) EPIA 2019. LNCS (LNAI), vol. 11804, pp. 495–506. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30241-2_42
27. Al-Rakhami, M., Gumaei, A., Alsanad, A., Alamri, A., Hassan, M.M.: An ensemble learning approach for accurate energy load prediction in residential buildings. *IEEE Access* **7**, 48328–48338 (2019)