IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 1, 2021, accepted June 13, 2021, date of publication June 21, 2021, date of current version July 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3091313

Extreme Learning Machine Applied to Software Development Effort Estimation

HALCYON DAVYS PEREIRA DE CARVALHO, ROBERTA FAGUNDES, (Member, IEEE), AND WYLLIAMS SANTOS
Department of Computer Engineering, University of Pernambuco, Recife 50720-001, Brazil

Corresponding authors: Halcyon Davys Pereira De Carvalho (hdpc@ecomp.poli.br), Roberta Fagundes (roberta.fagundes@upe.br), and Wylliams Santos (wbs@upe.br)

This work was supported by the Improvement Coordination of Higher Education Personnel—Brazil (CAPES) under Grant 001.

ABSTRACT The project management process has been used in the area of Software Engineering to support project managers to keep projects under control. One of the essential processes in Software Engineering is to conduct an accurate and reliable estimation of the required effort to complete the project. This article's objectives are: i) to identify the variables that influence the estimation based on the correlation, and ii) to apply the Extreme Learning Machine - ELM model for effort estimation and compare it with the literature models. Thus, it was investigated which technique has better effort prediction accuracy. The models were compared with each other based on predictive precision in the criterion of absolute mean residue (MAR) and statistical tests. The main findings in this study were: i) important variables for effort estimation and; ii) the results indicated that the ELM model presents the best results compared to the models in the literature for estimating software design effort. In this way, the use of Machine Learning techniques in the effort estimation process can increase the chances of success in the accuracy of the time estimates and the project's costs.

INDEX TERMS Extreme learning machine, machine learning, effort estimation, software development, project management.

I. INTRODUCTION

Software development effort forecasting models have been evaluated for decades to meet the needs of the software industries [1]. Estimating the software development effort is one of the project management processes responsible for determining the effort required to complete the project [2]. Therefore, accurate effort estimation is one of the critical points in reducing risks increasing the chances of success in the projects [3].

The software project management process comprises conducting activities through methods to achieve the projects' objectives [4]. During the software development phase, quantitative and qualitative metrics are estimated to meet customer needs. According to Saraiva [5], having successful software measurement programs allows users to interpret data and support decision-making.

There are several techniques to estimate the effort. The organization needs to decide which technique is the most suitable to deliver the product within the estimated time and cost [6]. One of the techniques used by project managers to perform software development effort estimates is Expert Judgment [7]. Still, this type of technique has limitations since it is subject to human errors.

One of the main challenges faced by Developers and Project Managers in Software Engineering is the estimation of software effort, as different project development lifecycle models require a different amount of effort at each stage of the process [8]. Traditional estimates [9] require an effort to document activities, making the estimate more complex and time-consuming. In addition, the experience of software developers, the project history of the software team in the same business area, and a large number of parameters and the relationship between these parameters are not always accurately predicted [10].

Thereupon, proposing ML techniques for Software Development Effort Estimation (SDEE) is an efficient alternative due to its learning capacity, modifying its behavior autonomously. Furthermore, they assist in decision-making based on data analysis, using minimal human interference

The associate editor coordinating the review of this manuscript and approving it for publication was Davide Aloini.

(specialist) [3]. Thereby, specialists spend less time estimating the project and more time on other tasks of the system that satisfy the customer [1].

In addition, the project objectives are not always well defined in the initial phase of the project. The necessary estimates for the project's development are uncertain, impacting the effective management of the project. The allocation of resources for the project's development depends on a realistic estimate of the effort during the initial phase of the software development life cycle [11]. According to Fadhil *et al.* [12], estimating project costs in the initial stage is a process of high significance, as it is necessary for computing resources and the budget required to deliver the project.

In other words, uncertainties about system requirements and inadequate estimation of the effort, cost, and staff required to develop the project are the main reasons for software projects' failure. Therefore, to minimize uncertainties during the software development cycle, Machine Learning (ML) techniques have been used to predict software development efforts over the past few years [3], [13].

According to the Systematic Literature Review (SLR) of Ali and Gravino [1], in the last decade, several Machine Learning techniques have been used to predict the software development effort, such as Multilayer Perceptron (MLP), Artificial Neural Network (ANN), Support Vector Regression (SVR), Support Vector Machine (SVM), Bayesian Network (BN), K-Nearest Neighbors (kNNs) and Extreme Learning Machine (ELM), among others. Ali and Gravino [1] also identified about 20 different datasets.

In this article, five machine learning algorithms are applied to estimate the software effort: K-Nearest Neighbors (kNNs), Linear Regression (LR), Support Vector Machine (SVM), Multilayer Perceptron (MLP), and Extreme Learning Machine (ELM). KNN is a non-parametric technique used for classification and regression tasks, where the size of the population can result in slow execution of the algorithm requiring high memory consumption [14]. SVM is a technique with the ability to model linear and nonlinear problems, however, its training process can be time-consuming with large volumes of data [13]. MLP networks are also used for classification and regression, according to [15], some parameters must be analyzed carefully, such as numbers of hidden layers, numbers of neurons of each hidden layer, numbers of training periods, and learning rate.

Recently, many modified ELM algorithms have been proposed, Residual Compensation ELM (RC-ELM) [16], Robust ELM (R-ELM) [17], Multilayer Extreme Learning Machines (ML-ELM) [18], Integrated Multiple Kernel ELM (IMK-ELM) [19]. These mentioned algorithms have been widely applied in many fields, such as in the prediction of the rate of gas used in the blast furnace during the manufacture of iron, in the treatment of tasks with Gaussian and non-Gaussian noise, deep learning, free location of devices in environments disordered using spatiotemporal information, among others.

The Systematic Literature Review (SLR) [1] has selected 75 primary studies, but just one [20] of them applies the ELM technique for Software Development Effort Estimation (SDEE). The SLR also reveals that among the various Artificial Neural Networks (ANN) types, feedforward networks are the most popular. However, according to Huang *et al.* [21], training an ELM can be faster than training a neural network by backpropagation, producing better generalization performance. In this sense, the focus of our study is based on the advantages of using the ELM technique for Software Development Effort Estimation (SDEE).

This paper was also investigated which attributes should be selected to obtain the best results in the effort estimates and how accurate is the software effort estimate, which leads to the two research questions answered in the section referenced in parentheses.

RQ1: Which features are important to obtain better results in estimating software development effort? (Section III-C)

RQ2: What is the Machine Learning technique that excels in effort estimation accuracy? (Section V)

Pearson's correlation coefficient [22] between the variables and their significance was analyzed, in order, to answer the first question RQ1. Finally, the performance of the model ELM were compared with the models investigated in the literature for software effort estimation [11], [23]–[29]. The Magnitude of Relative Error (MRE), Mean Absolute Error (MAE), and Mean Square Error (MSE) criteria were used to obtain the answers for the second question RQ2.

In summary, this article's main contribution is to demonstrate that the ELM technique for building a machine learning model, using it in a software development effort estimation dataset. It achieves results equivalent to or superior to the models investigated in the literature. Also, to bring benefits to the Software Engineering area, such as reducing time and costs spent during the project life cycle phases.

This paper is organized as follows. Section II presents related works. Section III presents the methodology employed in this research. Section IV the measurements used for model accuracy. Section V presents the results of the experiments. Section VI presents threats to validity. Final considerations and future work are addressed in Section VII.

II. RELATED WORKS

The investigation conducted by [23] presented a comparative study of various machine learning techniques used to estimate the effort required to develop software, such as Linear Regression (LR), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Multi-Layer Perceptron Neural Network (MLPNN) using the determination coefficient (R2) to evaluate the models. While in the study proposed by [24] used a Neural Feedforward Network to improve the Software Development Effort Estimation accuracy. The study [28] evaluated the use of Bayesian Networks (BNs) based on data in predicting software effort through extensive validation procedures.

Oliveira *et al.* [25] used the Genetic Algorithm (GA) as a Machine Learning (ML) technique, and the study proposed by [30] used the GA-based feature model. Both are applied

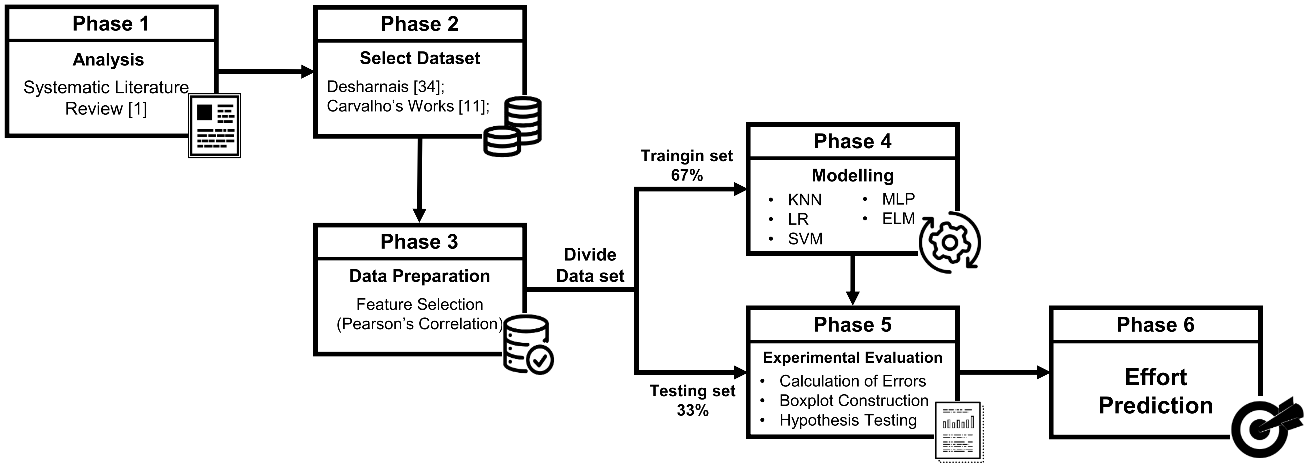


FIGURE 1. **Proposed Methodology for Accurate Effort Prediction.**

to selecting input resources and parameter optimization of ML techniques for estimating software development efforts. In this context, [27] investigated the use of an optimization algorithm, Bees Algorithm [31], to choose parameters depending on the data set used by the Model Tree (MT). The other work conducted by [26] proposed non-algorithmic techniques to estimate the software development effort. The study pointed out that the evolutionary algorithms showed better results. Experimental studies of automated machine learning ensembles were carried out in Minku's work [29]. The proposed approach performs well, being highly demanding in terms of performance across different datasets.

In Rahman's work [32], were implemented different machine learning algorithms, Radial Based Function Neural Network (RBFNN), Extreme Learning Machine (ELM), and Decision Tree (DT). The effort was measured for each category (i.e., small, medium, large) based on the size of the software. International Software Benchmarking Standards Group (ISBSG) Release 11 was the data set used to train and test the algorithms. RBFNN presented an excellent result for small software. However, DT presented better estimation results for small and medium software. ELM, on the other hand, demonstrated better efficiency for large software. The study [33] utilized a case-based reasoning model hybridized with the machine learning models: ABE-LS-SVM, ABE-ELM, ABE-ANN. The study shows that the ABE-LS-SVM outperforms the testing results ABE-ELM and ABE-ANN for Maxwell, Finnish and Kemmerer datasets.

Extreme Learning Machine (ELM) and Linear Least Squares Regression (LSR) have been applied in [20] and [11] to estimate effort under 231 small programs. In both studies, the only parameter to be decided for ELM is the number of hidden nodes. Mean Magnitude of Relative Error (MMRE) is implemented to analyze the performance of the proposed method. Our approach involves the use of the ELM algorithm by varying the parameters based on the prediction error. In addition, were used other MAE, MSE and RMSE metrics to assess the model's performance. Was added the Desahanais dataset to work to confirm the performance of the applied ELM model. A statistical test was used to compare the performance of the proposed model with other models in the literature through the normality test and hypothesis test. Furthermore, a temporal performance test was performed between the models. Were utilized Box-Plot graphs to present the results, through which we obtained significantly better results.

III. RESEARCH METHODOLOGY

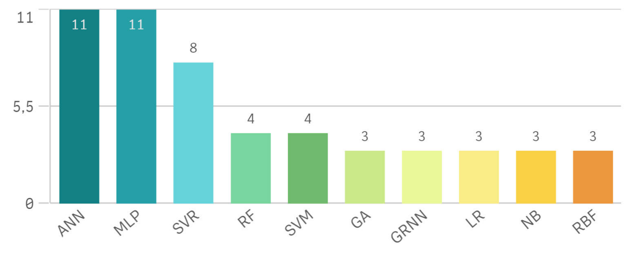
This section presents the methodology used to build the ELM model applied to software effort estimation. Figure 1 shows the proposed methodology for accurate effort prediction, which consists of six phases: Analysis (Section III-A), Select Data Set (Section III-B), Data Preparation (Section III-C), Modeling (Section III-D), Experimental Evaluation (Section IV), Effort Forecasting (Section V).

A. SYSTEMATIC LITERATURE REVIEW's ANALYSIS

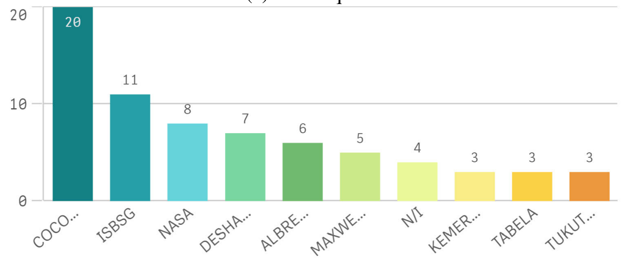
The proposal to build the effort estimation model for software development was based on the Systematic Literature Review (SLR) carried out by [1]. The SLR presents empirical studies published from January 1991 to December 2017, and after the selection criteria, 75 primary studies were selected.

The ML techniques used in the 75 primary studies were: Artificial Neural Network (ANN), which is the most referenced technique and used in 60%; the Support Vector Machine (SVM) method, which was used in 25%, whereas Case-Based Reasoning (CBR) was applied in 17%. The other techniques, Bayesian Network (BN), K-Nearest Neighbors (KNNs), Decision Tree (DT), Genetic Programming (GP), Classification and Regression Tree (CART), Random Forest (RF), were the least referenced machine learning techniques.

The databases used in the 75 primary studies were: NASA, the most referenced in RSL with 23%. Extensively investigated, COCOMO and ISBSG are registered equally at 21%, followed by Desharnais, used in 19%. Kemerer, Maxwell,



(a) Techniques



(b) DataSet

FIGURE 2. **Top 10 of the Techniques and Dataset used in the last 10 years.**

Albrecht, Tukutuku, Finnish, and Miyazaki were the least referenced databases.

Based on the systematic review, it was selected the primary studies from the last 10 years. Figure 2 (a) shows the Top 10 machine learning techniques, in which ANN, MLP are cited in 11 studies and SVR in 8 reviews, the 3 most mentioned ones, while Figure 2 (b) shows the Top 10 datasets used in this period, with emphasis on the COCOMO dataset cited in 20 studies, followed by ISBSG and NASA, mentioned in 11 and 8 reviews respectively.

From this analysis, it was observed that among the various Artificial Neural Networks (ANN) types, the ELM technique just emerged in one primary study [20]. According to Huang [21], training an ELM can be faster to meet the criteria to complete the training than training a neural network by backpropagation, producing better generalization performance in most cases. Given the advantages of using the ELM technique for Software Development Effort Estimation (SDEE), it was chosen to model the ELM technique for Software Development Effort Estimation in this work.

We analyzed the datasets with the technique's choice to be used in our model to estimate the software development effort. We decided to use the Desharnais dataset, Section III-B1, as it is a little explored dataset among effort estimation models.

The estimation of software developments is compared with other models in the literature. We will also use the dataset referenced in Carvalho's work [11] described in Section III-B2 in order to compare our model's performance with the porposed models in Carvalho's article [11].

B. SELECT DATASET

This subsection presents the descriptive analysis of the datasets for the construction of our model.

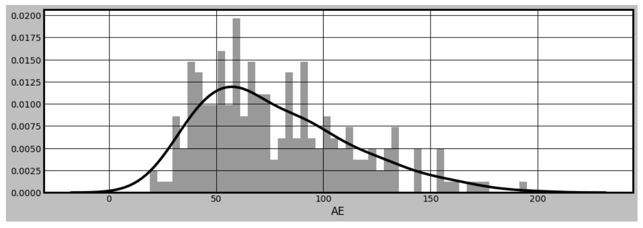


FIGURE 3. **Distribution of the effort value in the Desharnais database.**

1) DESHARNAIS DATASET

The Desharnais [34] dataset analyzed consists of information from 81 software projects from a Canadian company. Each project has 12 attributes: Project id, Team Experience, Manager Experience, Year End, Length, Effort, Transaction, Entities, Point Adj, Adjustment, Point Non-Adjust, and Language, which in turn attributes are classified into numeric and categorical as described and detailed in Table 1.

Based on Table 1, it was selected only the numerical data, and the "Project" attribute was also excluded because it is not correlated to the project effort estimate.

With the selected attributes, the next step was to identify the independent variables and the dependent variable. Based on the software development effort estimation models, the main variable is the effort required to complete the project, which in this case, the dependent attribute is "Effort", and the other correlated variables are the independent attributes: "TeamExp", "ManagerExp", "YearEnd", "Length", "Transactions", "Entities", "PointsNonAjust", "Envergure" and "PointsAdj".

Table 2 presents the statistical measures for all independent and dependent attributes. The elapsed time (Length) of the 81 measured projects varied between 1 and 39 months (*with an average of 11.7 months*). Two attributes that represent the software size, "PointsNonAdjust" and "PointsAdjust", in which the difference between the two is not great, since the average of PointsNonAdjust is 304, while the average of PointsAdjust is 289. Finally, the level of effort recorded was between 546 and 23,940 person-hours (*with an average of 5,046.31 person-hours*).

The histogram in Figure 3 illustrates the distribution of data on effort, the dependent variable. The measurement of effort is in person-hours. The variables are positively skewed: variables with the majority of records located towards lower values and some very high external values.

2) CARVALHO's WORK [11]

The second set of analyzed data will be available in [11], which consists of 231 software projects, and has 5 attributes classified into numeric and string: "P" (Program Number), "N&C" (New and Changed Code), composed of added and modified code, "R" (Reused Code) where both are recognized as physical lines of code (LOC) and "AE" (Actual Effort) which is measured in minutes, are the type numeric and "DP" (Developer Code) is of type string.

TABLE 1. Desharnais Dataset Variables.

Attributes	Classification	Description
Project	Numeric	Project ID which starts by 1 and ends by 81
TeamExp	Numeric	Team experience measured in years
ManagerExp	Numeric	Manager experience measured in years
YearEnd	Numeric	Year the project ended
Length	Numeric	Duration of the project in months
Effort	Numeric	Actual effort measured in person-hours
Transactions	Numeric	Number of the logical transactions in the system
Entities	Numeric	Number of the entities in the system
PointsNonAdjust	Numeric	Size of the project measured in unadjusted function points. This is calculated as Transactions plus Entities
Envergure	Numeric	Function point complexity adjustment factor. This is based on the General Systems Characteristics (GSC). The GSC has 14 attributes; each is rated on a six-point ordinal scale. $Envergure = \sum_{i=1}^{14} CGS_1$
PointsAdjust	Numeric	Size of the project measured in adjusted function points. This is calculated as: $PointsAdjust = PointsNonAdjust * (0,65 + 0,01 * Envergura)$
Language	Categorical	Type of language used in the project expressed as 1, 2 or 3. The value "1" corresponds to "Basic Cobol", where the value "2" corresponds to "Advanced Cobol" and the value "3" to 4GL language.

TABLE 2. Descriptive Statistics for the Desharnais Dataset.

Attributes	Mean	Median	Stdev	Min	Max
TeamExp	2.185	2.000	1.415	-1.000	4.000
ManagerExp	2.531	3.000	1.644	-1.000	7.000
YearEnd	85.741	86.000	1.222	82.000	88.000
Length	11.667	10.000	7.425	1.000	39.000
Transactions	182.123	140.000	144.035	9.000	886.000
Entities	122.333	99.000	84.882	7.000	387.000
PointsNonAdjust	304.457	266.000	180.210	73.000	1127.000
Adjustment	27.630	28.000	10.592	5.000	52.000
PointsAjust	289.235	255.000	185.761	62.000	1116.000
Effort	5046.309	3647.000	4418.767	546.000	23940.000

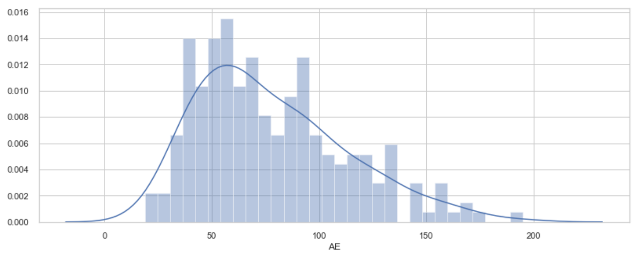


FIGURE 4. Effort Value Distribution.

TABLE 3. Descriptive Statistics for the Carvalho's Work Dataset.

Attributes	Mean	Median	Stdev	Min	Max
N&C	38.32	30.00	25.47	10.00	137.00
R	39.94	33.00	29.03	1.00	149.00
AE	78.12	71.00	34.60	19.00	195.00

This dataset selected only the numeric attributes and excluded the "P" attribute because it has no relation to the effort estimate. The descriptive statistics for attributes selected are shown in Table 3.

During the dataset analysis, it was identified two independent attributes, "N&C" (New and Changed Code) and "R" (Reused Code), and the target attribute, dependent, "AE" (Actual Effort).

The histogram in Figure 4 illustrates the distribution of the dependent variable (Ae-effort) in relation to the effort, which is measured in minutes. A normal distribution trend is noticed in the histogram since the largest concentration of data is around the average, and the frequency is close to the limits.

C. DATA PREPARATION

In addition to the descriptive and statistical analysis of the bases, this phase used Pearson's correlation coefficient [22], also called linear correlation, which measures the degree of relationship between two quantitative variables and expresses the degree of correlation by means of values between -1 and 1. A correlation coefficient close to zero indicates that there is no relationship between the two variables. The closer to 1 or -1, it suggests that it has a perfect positive or negative correlation. From the point of view of [35], results between 0.5 and 1.0 indicate a high correlation.

In this study, a high correlation from 0.5 was considered, according to [35]. As it can be seen in Figure 5, the independent attributes, such as: *"Transactions", "Length", "Entities", "PointsAdj"* and *"PointsNonAdjust"* have their correlation coefficients, *0.514296, 0.585761, 0.655355, 0.715757, 0.732605* respectively, above 0.50 in relation to the *Effort* in the Desharnais dataset.

In conclusion, the mentioned attributes are statistically the most significant for our model's construction, answering the Q1 research question:

RQ1: Which features are important to obtain better results in estimating software development effort?

The scatter plot Figure 6 represents the correlation coefficients of the attributes "Transactions", "Length", "Entities", "PointsAdj" and "PointsNonAdjust" and the attribute "Effort". A scatterplot demonstrates the magnitude of the correlation between two variables if any; Positive correlation when there is an agglomeration of points in an increasing trend and Negative correlation when points are concentrated in a decreasing line [36].

We noted that in Figure 6, the data correlation has an specific direct (or positive) linear association, showing a high correlation between the independent variables and the

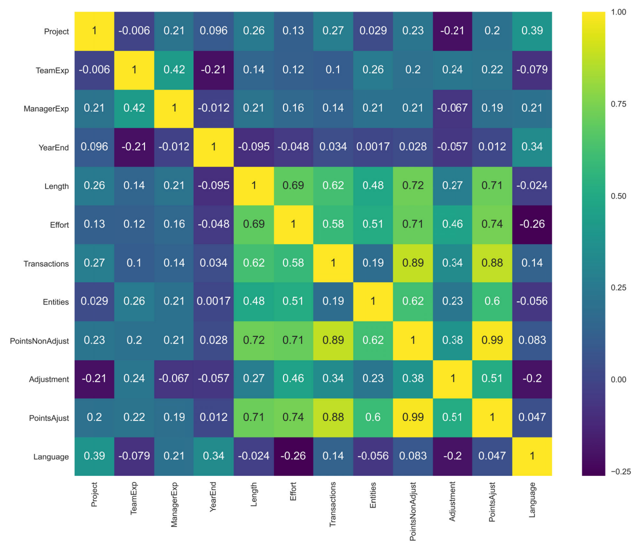


FIGURE 5. Pearson Correlation Desharnais Dataset.

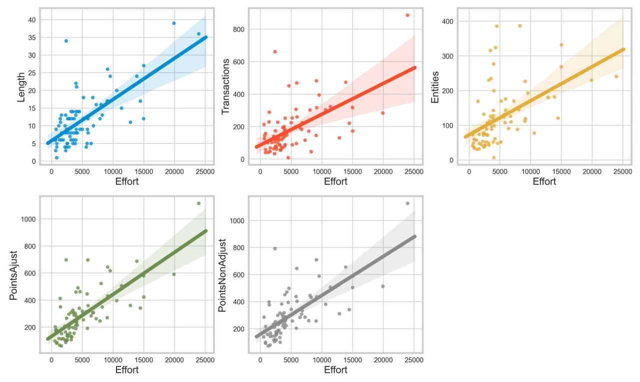


FIGURE 6. Correlation coefficients between variables in the Desharnais database.

dependent variable and that most points on the scatter plot approximate the straight line.

For the dataset available in Carvalho's Work [11], it was also performed the correlation between two independent attributes, "N&C" and "R" with the dependent attribute "AE".

Figure 7 (a) shows a high correlation between the attributes "N&C" and "AE" (0.69). For Figure 7 (b), it is observed a low correlation between the attributes "R" and "AE". Therefore, there seems to be no linear association between the two variables. As stated earlier, the correlation or correlation coefficient measures the tendency of two variables to change, depending on their relationship.

After analyzing and preparing the datasets, building the model for estimating the software development effort based on the ELM technique came next.

D. MODEL BUILDER

In this study, the software development effort estimation model was built with the Extreme Learning Machine technique. It was compared with the literature models and the same data set to estimate the effort was used [23]. The models in the literature are Linear Regression (LR), Support Vector Machine (SVM), Nearest K-Neighbor (KNN), and Multi-Layer Perceptron (MLP).

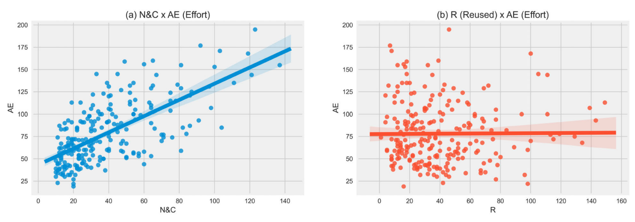


FIGURE 7. Correlation Carvalho's Work [11] Dataset.

1) EXTREME LEARNING MACHINE

The Extreme Learning Machine (ELM) algorithm was proposed by [37]. Its architecture is equivalent to a Single-layer Feedforward Networks (SLFN) or Feedforward Neural Network (FNN), with slight differences. The weights of the input layer neurons are generated randomly instead of being adjusted, and the weights of the neurons of the output layer are calculated analytically, without using iterative processes as in backpropagation. In this way, the output layer's activation function results in a linear model [38].

According to Huang *et al.* [37], and Huang *et al.* [21], the architecture of an ELM can be represented with a hidden layer with $\tilde{N}$ neurons. To learn **N** different arbitrary samples (x_i, t_i), where $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in R^n$ and $t_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in R^m$, input weights and hidden bias are randomly generated and the activation function is g(x). According to Huang *et al.*, [37] and Huang *et al.*, [21] the mathematical formula of ELM is represented by:

$$\sum_{i=1}^{\tilde{N}} \beta_i g(w_i \cdot x_j + b_i) = o_j, \quad j = 1, \ldots, N, \tag{1}$$

where $\tilde{N}$ is the number of neurons in the hidden layer and **N** is the number of training samples, $\beta_i = [b_{i1}, b_{i2}, \ldots, b_{im}]^T$ represents the weight vector that connects the *i-th* hidden layer neuron to neurons of the output layer, $g(\cdot)$ is the activation function, $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is the weight vector that connects the *j-th* hidden layer neuron and the input layer neurons, $\mathbf{x}_j$ represents each distinct sample and $\mathbf{b}_j$ denotes the bias of the *j-th* neuron of the hidden layer. To make the network outputs equal to the expected results, in other words, to perform error training equal to zero, there must be β_i, $\mathbf{w}_i$ and $\mathbf{b}_i$ so that the equation can be written as:

$$\sum_{i=1}^{\tilde{N}} \beta_i g(w_i \cdot x_j + b_i) = t_j, \quad j = 1, \ldots, N, \tag{2}$$

where $\mathbf{t}_j$ are the outputs expected by the network, referring to the $\mathbf{x}_j$ sample input.

The previous **N** equations are written compactly in the following equations:

$$H\beta = T, \tag{3}$$

where

$$H = \begin{bmatrix} g(w_i \cdot x_j + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_j + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(w_i \cdot x_N + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}, \tag{4}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \cdot \\ \cdot \\ \cdot \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m} \tag{5}$$

and

$$T = \begin{bmatrix} t_1^T \\ \cdot \\ \cdot \\ \cdot \\ t_N^T \end{bmatrix}_{N \times m} \tag{6}$$

H is called the hidden layer output matrix of the neural network the *i-th* column of **H** is the *i-th* neuron output vector of the hidden layer in with respect to the entries x_1, $x_2, \ldots, x_N$. In this study, we use the Extreme Learning Machine model available at [39]. Machine learning techniques have parameters that, most of the time, significantly affect the performance of these techniques. For our model we defined the following parameters: n_hidden = 5, alpha = 1.0, rbf_width = 0.1 and activation = 'sigmoid'.

2) OTHER MODELS INVESTIGATED

Linear Regression (LR) is a technique used to predict an unknown dependent variable, given the independent variables' values. [40].

Support Vector Machine (SVM) is a machine learning algorithm used for classification and regression. SVM used for regression analysis is called Support Vector Regression (SVR) [15].

K-Nearest Neighbor (KNN) technique can be used for classification or regression. In general, the algorithm uses Euclidean distance to calculate distances between its closest neighbors. The results are based on the average of the nearest neighbor k [15].

Multilayer Perceptron (MLP) neural networks are feedforward neural networks usually trained with a backpropagation algorithm. Traditional MLP networks contain at least three layers: an input layer, a hidden layer, and an output layer. The number of nodes in the input layer is defined according to the independent variables identified. In the hidden or intermediate layer, the number of nodes is defined through the configuration parameters. In contrast, in the output layer, the number of nodes depends on the solution, the number of dependent variables [41].

To implement the models, we use the Python language such as the libraries: Numpy [42], Pandas [43], Scikit-Learn [44], and Matplotlib [45].

IV. EXPERIMENTAL EVALUATION

This section describes the measurements used for model accuracy. A basic factor for any forecasting model is whether forecasts are accurate or not.

It is possible to find several metrics in the literature to assess the software development effort estimation models' accuracy. The frequently used assessment measures are MMRE and PRED (k). According to Shepperd and MacDonell [46], MMRE does not present a good precision in the forecast of software effort estimation because these criteria are biased. Although being applied in this work, the results are used only for comparison with other results in the literature.

In turn, the performance indices, Mean Absolute Error (MAE), Mean Square Error (MSE), and Root Mean Square Error (RMSE), are being used as metrics to assess the accuracy of the evaluated model.

A. MEAN MAGNITUDE OF RELATIVE ERROR

The Magnitude of Relative Error (MRE) is the difference between the actual effort (work done by the developer to complete the project) and the predicted effort (estimated using project management techniques), divided by the real effort. MRE is represented in Equation 7.

$$MRE = \frac{|y_i - \hat{y}_i|}{y_i} \tag{7}$$

where y_i is the actual effort, and $\hat{y}_i$ is the estimated effort, both of which are used in software project i.

The Mean Magnitude of Relative Error (MMRE) is the average of the MRE of the software project. MMRE is calculated for each project in the dataset following Equation 8

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i \tag{8}$$

where n the number of cases in dataset.

The Mean Absolute Error is a measure of how far the estimates are from actual values. MAE is defined in Equation 9

$$MAE = \frac{1}{n}\sum_{t=1}^{n} |y_i - \hat{y}_i| \tag{9}$$

y_i is the ith value of the variable being predicted, $\hat{y}_i$ its estimate, $y_i - \hat{y}_i$ the ith residual.

Mean Squared Error (MSE) is the mean quadratic difference between the estimated values and the current value as denoted in Equation 10.

$$MSE = \frac{1}{n}\sum_{t=1}^{n} (y_i - \hat{y}_i)^2 \tag{10}$$

Root Mean Squared Error (RMSE), as denoted in Equation 11.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(y_i - \hat{y}_i\right)^2} \tag{11}$$

The sample of 500 iterations was calculated as the Standard Deviation (SD) of the Error. Besides, we performed statistical tests, such as the Shapiro-Wilk [47] and Wilcoxon [48] tests. It was also used relative p-value to evaluate the performance of the models.

V. RESULTS AND DISCUSSION

This section discussed the results obtained from the experiments using the Desharnais [34] data set and the data set available in Carvalho's [11] work. The applied model is based on the ELM technique according to the configurations presented previously. The **Algorithm 1** presents the pseudocode for experimental evaluation.

Algorithm 1 Pseudo-Code of the Experiment Execution

```
Input: Use the dataset
 1: Set: Model = KNN, LR, SVM, MLP, ELM
    LOOP Process
 2: for each Model do
 3:     for $i = 1$ to 500 do
 4:         Shuffle dataset in Training (67%) and Testing (33%)
 5:         Apply: Model to Training set
 6:         Select: Best Model
 7:         Apply: Model to Testing set
 8:         Calculate: the Erros of the interaction
 9:     end for
10:     Calculate mean and standard deviation of error MMER (8)
11:     Calculate mean and standard deviation of error MAE (9)
12:     Calculate mean and standard deviation of error MSE (10)
13:     Calculate mean and standard deviation of error RMSE (11)
14: end for
```

A. DESHARNAIS DATASET

It was applied the techniques of KNN, LR, SRV, MLP, and ELM to the Desharnais [34] datasets. It was analyzed the mean and standard deviation (SD) for all metrics, considering the 500 simulations performed for this dataset. Table 4 shows the results Means and, in parentheses (SD), the Standard Deviation of errors in the dataset.

Figure 8 shows the MAE boxplot graph in each model and that there is an outlier in all regression models, except the LR model. Indeed, KNN, LR, SVM, and MLP models are overestimated concerning the applied ELM model. Hence, it is possible to observe that the applied ELM model

TABLE 4. Desharnais: Comparison of the results Means and Standard Deviation (SD) of the literature study in [23].

Technics	MMER Means (SD)	MAE Means (SD)	MSE Means (SD)	RMSE Means (SD)
KNN[1]	0.2413 (0.0398)	0.0727 (0.0130)	0.0127 (0.0048)	0.1107 (0.0221)
LR[1]	0.2287 (0.0457)	0.0646 (0.0113)	0.0088 (0.0032)	0.0924 (0.0173)
SVM[1]	0.4130 (0.0500)	0.0719 (0.0133)	0.0102 (0.0039)	0.0993 (0.0189)
MLP[1]	0.4235 (0.0646)	0.0677 (0.0133)	0.0104 (0.0050)	0.0994 (0.0224)
ELM[2]	**0.1808 (0.0121)**	**0.0562 (0.0046)**	**0.0078 (0.0017)**	**0.0880 (0.0086)**

[1] results of the literature study in [23].
[2] results of the applied model.

TABLE 5. Results *p-value*.

Erro	Technics	p-value
MRE	KNN X ELM	$5.287x10^{-79}$
MRE	LR X ELM	$1.461x10^{-62}$
MAE	KNN X ELM	$1.286x10^{-69}$
MAE	LR X ELM	$1.266x10^{-36}$
MAE	SVM X ELM	$3.097x10^{-67}$
MAE	MLP X ELM	$3.781x10^{-50}$
MSE	LR X ELM	$9.021x10^{-11}$
RMSE	KNN X ELM	$6.711x10^{-56}$
RMSE	LR X ELM	$1.687x10^{-08}$
RMSE	SVM X ELM	$2.794x10^{-27}$
RMSE	MLP X ELM	$7.779x10^{-23}$

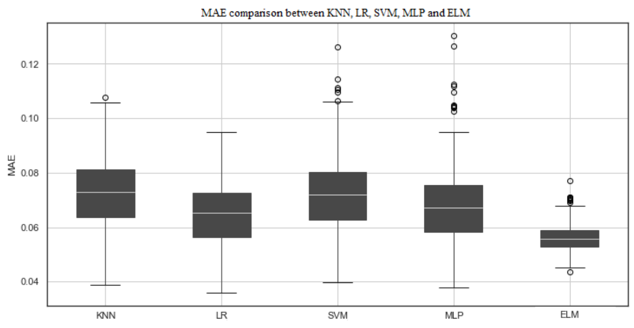


FIGURE 8. Boxplots of Mean Absolute Erro.

obtained the best result because of its simplicity of usage, higher speed of learning, greater generalization performance. Besides, the box and tail (boxplot) of the applied model ELM are less distorted than those of the other commonly used literature models.

Analyzing the results obtained in Table 4, it is possible to conclude that the ELM model obtained the smallest amount of errors shown in table 4 confirming better performance. However, in most cases, in the second decimal place, the gain was necessary to carry out hypotheses to validate the results.

Before performing the hypothesis test, it was verified the existence of normality between the values using the Shapiro-Wilk test [47]. Since the dataset does not have a normal distribution, the Wilcoxon hypothesis test [48] was used with a 5% significance. In the null hypothesis (H0), the model presents results equal to or less than the applied ELM model. Whereas, in the alternative hypothesis (H1), the applied ELM model had the smallest error shown in

TABLE 6. Critical Evaluation Table of Related Work.

Authors	Dataset	Techniques	MMRE
Jodpimai P, Sophatsathit P, Lursinsap C [24]	Albrecht, CF, Cocomo, Desharnais, Nasa	ANN	0.420
Oliveira et al. [25]	Albrecht, Cocomo, Desharnais, Kemerer, Nasa	M5P MLP SVR L SVR RBF	0.594 0.315 0.368 0.405
Gabrani, Goldie and Saini, Neha [26]	Desharnais, Maxwell, Miyazaki94	ANFIS MLP SVR	1.057 0.782 0.738
Mohammad Azzeh [27]	Albrecht, Cocomo, Desharnais, ISBSG, Kemerer, Maxwell, Nasa93, Telecom	OMT	0.323
Tierno et al. [28]	Cocomo, Desharnais, Maxwell	BN	0.689

The MMRE value is relative to the Desharnais dataset for each model compared

TABLE 7. Article: Results Means and Standard Deviation (SD) of dataset available in Carvalho's [11] work.

Technics (Ref)	Error (SD)
ELM with 2 n_hidden [11]	23.8934 (3.9770)
ELM with 5 n_hidden [11]	24.2228 (2.0757)
ELM with 2 n_hidden Model	21.5982 (0.2708)
ELM with 5 n_hidden Model	21.6207 (0.2918)

Equation 12.

$$\begin{cases} H_0 : \mu_1 <= \mu_2 \\ H_1 : \mu_1 > \mu_2 \end{cases} \tag{12}$$

Table 5 shows the p-value results for the Wilcoxon tests. With 95% confidence, the null hypothesis (H0) is rejected. It is possible to conclude that the difference between the population medians is statistically significant, according to the Wilcoxon test.

Table 6 shows the comparison with other studies in the literature that used Desharnais datasets to estimate software development effort. According to the results presented in Table 4, the applied ELM model obtained better results than studies in the literature due to its remarkable generalization performance and implementation efficiency.

B. CARVALHO's WORK DATASET

Likewise, the ELM technique applies to the other dataset used by [11] for two independent variables **N&C** and **R**. For this dataset, 1000 simulations were performed, the same amount of simulations used by the author. Table 7 shows the mean and standard deviation in parentheses (SD) of errors in the dataset. The applied ElM model with 2 and 5 hidden layers appears more stable and reliable generalization performance.

Figure 9 show the boxplot graph of the ELM Models with 2 and 5 hidden layers. The "**ELM with** 2 **and** 5 **n_hidden [11]**" presented a mean with greater variability in relation to the "**ELM with** 2 **and** 5 **_hidden applied**" models. It is also possible to note that data dispersion in the "**ELM with 2 and** 5 **n_hidden applied**" models are almost non-existent and remarkable generalization performance.

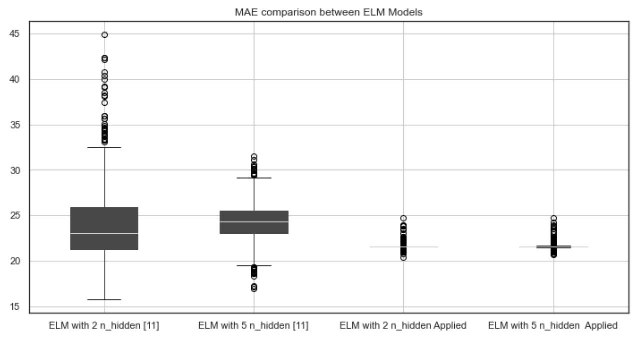


FIGURE 9. MAE comparison between ELM models.

C. PERFORMANCE EVALUATION

In this section, the performance of the applied ELM learning algorithm is compared with KNN, LR, SVM algorithms, and the conventional back-propagation (MLP) algorithm. All simulations of the algorithms are performed using packages developed in Python language and executed in the Jupyter Notebook environment, running on an Intel(R) Core(TM) i7 2.1 GHz CPU, 8 GB RAM. To evaluate the time consumption of the algorithms, all simulations were conducted with 500 iterations, 1,000 iterations, 5,000 iterations, and 10,000 iterations. Figure 10 shows the comparison of the time consumption speed of the evaluated models. The ELM algorithm is faster than the KNN and MLP models in this case and has practically the same performance compared to the LR and SVM algorithms. According to the values presented, the ELM model was 3 times faster than the backpropagation algorithm, MLP.

Finally, let's answer our last research question:

RQ2: What is the machine learning technique that stands out in the accuracy of effort estimation?

Throughout the work, we investigated five machine learning techniques used in datasets to estimate the software development effort, namely: (i) Linear Regression (LR), (ii) Support Vector Machine (SVM), (iii) K-Nearest Neighbor (KNN), (iv) Multilayer Perceptron (MLP), and (v) Extreme Learning Machine (ELM). In simulations, we evaluated the accuracy of the precision of each effort estimation technique

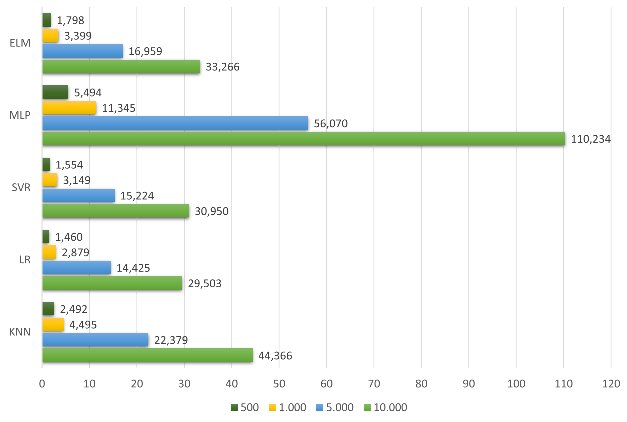


FIGURE 10. **Comparison of time consumption of the applied methods.**

where the ELM was the one that presented the best results, Table 4, compared to the other techniques used in the literature.

We can also highlight that the use of machine learning in the area of Software Engineering can have a great gain, since Software Engineering is related to obtaining quality results defined in the project management plan to meet the schedule (effort) and the budget (cost) of the project. Therefore, the use of machine learning techniques to predict software development efforts can help the project team deal with uncertainties in estimates during the project lifecycle, providing better quality project deliverables in terms of effort and cost.

VI. THREATS TO VALIDITY

This section will discuss our study's validity based on internal and external threats and construct validity [49]. Internal validity relates to the examination of causal relations [49]. A possible threat to internal validity is the choice of ELM algorithm parameters. In our study, we considered selecting parameters as an explicit step in dealing with internal validity. For each data set used in the research, it was necessary to adjust the parameters, as there are no studies in this area on how to determine these parameters for each data set.

We randomly divided the data into the training and test sets in the proportion of 67% and 33%, respectively. The random assignment of the data can have a considerable influence on the results of the model. However, considering that all models are executed on the same data sets, there will be no significant impact on the overall work since the objective is to compare the performance of the models in the applied data set.

External validity concerns the generalization of study results outside the study to other situations [49]. One of the problems of external validation in machine learning is related to the number of samples available in the data set. Due to the small size of the data set, the number of data remaining for testing is even more limited. Another limitation we encountered was the availability of data sets from free software projects. Data availability is not frequent and often paid for, causing forecasting difficulties with a reduced amount of data.

Construction validity refers to the extent to which the operational measures studied represent what the researcher has in mind and what is investigated according to the research questions [49]. One of the metrics used to assess the models' accuracy to estimate the software development effort was the Mean Magnitude of the Relative Error (MMRE). However, according to Shepperd and MacDonell [46], MMRE does not present a good precision in the forecast of software effort estimation. Although it is being used at work, this metric is used only for comparison with other results in the literature. In the study, we used the Mean Absolute Error (MAE), proposed by [46] as a good metric to measure the accuracy of the software effort estimation model.

VII. CONCLUSION AND FUTURE WORK

In Software Engineering, obtaining a reliable and accurate estimate of the software development effort has always been challenging. Estimating the effort and precise cost in the initial phase of the project would greatly benefit the area. With this in mind, to minimize the uncertainty of the estimates during the software development cycle, a technique based on machine learning was applied, the Extreme Learning Machine (ELM), as a model for estimating software development effort compared with other effort estimation models used in the literature. The applied model will support the Specialist and Project Managers in forecasting effort estimates, reducing the time and costs spent during the execution phase of the effort project and cost estimation.

In simulations, it was used two sets of data from software projects similar to our reality. Desharnais [34] containing 81 software projects from a Canadian company and the data set evaluated in Carvalho's work [11] having 231 software projects. For the two data sets, the data set in training and tests was divided in the proportion of 67% and 33%, respectively. In all simulations, the inputs were normalized in the interval [0.15; 0.85], and the data were randomly assigned to a data set in training and testing to obtain unbiased results.

Many projects use the Mean Magnitude of the Relative Error (MMRE) to assess the forecasting methods' accuracy in estimating the software project effort, as shown in the related works [24]–[28]. However, according to Shepperd and MacDonell's study [46], MMRE is not an ideal metric for evaluation, as it does not have a good precision in estimating the software effort. The MMRE metric results are being used only in comparison with other results in the literature and the MAE, MSE and RMSE metrics were adopted to evaluate the ELM model.

In the Desharnais [34] dataset, it was conducted experiments to compare the ELM technique with other techniques in the literature, (i) Linear Regression (LR), (ii) Support Vector Machine (SVM), (iii) K-Nearest Neighbor (KNN), (iv) Multilayer Perceptron (MLP). In the data set evaluated by Carvalho's work [11], experiments were carried out compared to the author's model, adjusting the parameters to obtain a better result as suggested in future works.

One of this article's main contributions is that the applied model, based on the ELM technique, was obtained by comparing it with the literature models used to predict software development effort estimates. Consequently, the error estimate rates tend to be reduced, helping project managers increase the forecast in the effort estimate during the project life cycle.

In addition, during the process of preparing the bases, Pearson's correlation coefficient was used to identify the variables with high correlation, which in turn identified the potential variables to be used in the model applied for estimating software effort, thus achieving better results compared to the literature.

It is concluded that the accurate and reliable estimation of the effort required to complete a project is an important process in Software Engineering. In this way, the use of the technique based on machine learning increases the project's chances of success, reducing the time and costs of the project.

For future work, we propose using an optimization method, namely, Particle Swarm Optimization (PSO), to optimize the model parameters for estimating the software development effort.

REFERENCES

[1] A. Ali and C. Gravino, "A systematic literature review of software effort prediction using machine learning methods," *J. Softw., Evol. Process*, vol. 31, no. 10, pp. 1–25, Oct. 2019.

[2] *Project Management Body of Knowledge (PMBOK)*, 6th ed., Project Manage. Inst., Newtown Square, PA, USA, 2017.

[3] N. Rankovic, D. Rankovic, M. Ivanovic, and L. Lazic, "A new approach to software effort estimation using different artificial neural network architectures and Taguchi orthogonal arrays," *IEEE Access*, vol. 9, pp. 26926–26936, 2021.

[4] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 8th ed. New York, NY, USA: McGraw-Hill, 2016.

[5] R. Saraiva, A. Medeiros, M. Perkusich, D. Valadares, K. C. Gorgonio, A. Perkusich, and H. Almeida, "A Bayesian networks-based method to analyze the validity of the data of software measurement programs," *IEEE Access*, vol. 8, pp. 198801–198821, 2020.

[6] S. Tariq, M. Usman, and A. C. M. Fong, "Selecting best predictors from large software repositories for highly accurate software effort estimation," *J. Softw., Evol. Process*, vol. 32, no. 10, pp. 1–19, Oct. 2020.

[7] C. Lopez-Martin, "A fuzzy logic model for predicting the development effort of short scale programs based upon two independent variables," *Appl. Soft Comput. J.*, vol. 11, no. 1, pp. 724–732, Jan. 2011.

[8] M. Hammad and A. Alqaddoumi, "Features-level software effort estimation using machine learning algorithms," in *Proc. Int. Conf. Innov. Intell. Informat., Comput., Technol. (ICT)*, Nov. 2018, pp. 1–3.

[9] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, "Neural network models for software development effort estimation: A comparative study," *Neural Comput. Appl.*, vol. 27, no. 8, pp. 2369–2381, Nov. 2016.

[10] V. Yurdakurban and N. Erdoğan, "Comparison of machine learning methods for software project effort estimation," in *Proc. 26th IEEE Signal Process. Commun. Appl. Conf.*, May 2018, pp. 1–4.

[11] H. D. P. Carvalho, M. N. C. A. Lima, W. B. Santos, and R. A. de A. Fagunde, "Ensemble regression models for software development effort estimation: A comparative study," *Int. J. Softw. Eng. Appl.*, vol. 11, no. 3, pp. 71–86, May 2020.

[12] A. A. Fadhil, R. G. H. Alsarraj, and A. M. Altaie, "Software cost estimation based on dolphin algorithm," *IEEE Access*, vol. 8, pp. 75279–75287, 2020.

[13] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms," *J. Syst. Softw.*, vol. 137, pp. 184–196, Mar. 2018.

[14] Y. Song, J. Liang, J. Lu, and X. Zhao, "An efficient instance selection algorithm for k nearest neighbor regression," *Neurocomputing*, vol. 251, pp. 26–34, Aug. 2017.

[15] M. Hosni, A. Idri, A. B. Nassif, and A. Abran, "Heterogeneous ensembles for software development effort estimation," in *Proc. 3rd Int. Conf. Soft Comput. Mach. Intell. (ISCMI)*, Nov. 2016, pp. 174–178.

[16] J. Zhang, W. Xiao, Y. Li, and S. Zhang, "Residual compensation extreme learning machine for regression," *Neurocomputing*, vol. 311, pp. 126–136, Oct. 2018.

[17] J. Zhang, Y. Li, W. Xiao, and Z. Zhang, "Robust extreme learning machine for modeling with unknown noise," *J. Franklin Inst.*, vol. 357, no. 14, pp. 9885–9908, Sep. 2020.

[18] J. Zhang, Y. Li, W. Xiao, and Z. Zhang, "Non-iterative and fast deep learning: Multilayer extreme learning machines," *J. Franklin Inst.*, vol. 357, no. 13, pp. 8925–8955, Sep. 2020.

[19] J. Zhang, Y. Li, and W. Xiao, "Integrated multiple kernel learning for device-free localization in cluttered environments using spatiotemporal information," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4749–4761, Mar. 2021.

[20] S. K. Pillai and M. K. Jeyakumar, "Extreme learning machine for software development effort estimation of small programs," in *Proc. Int. Conf. Circuits, Power Comput. Technol. (ICCPCT)*, Mar. 2014, pp. 1698–1703.

[21] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, Dec. 2006.

[22] D. Montgomery and G. Runger, *Estatística Aplicada e Probabilidade Para Engenheiros*, 5th ed. Rio de Janeiro, Brazil: LTC, 2012.

[23] S. Shukla and S. Kumar, "Applicability of neural network based models for software effort estimation," in *Proc. IEEE World Congr. Services (SERVICES)*, Jul. 2019, pp. 339–342.

[24] P. Jodpimai, P. Sophatsathit, and C. Lursinsap, "Estimating software effort with minimum features using neural functional approximation," in *Proc. 10th Int. Conf. Comput. Sci. Appl. (ICCSA)*, 2010, pp. 266–273.

[25] A. L. I. Oliveira, P. L. Braga, R. M. F. Lima, and M. L. Cornélio, "GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation," *Inf. Softw. Technol.*, vol. 52, no. 11, pp. 1155–1166, Nov. 2010.

[26] G. Gabrani and N. Saini, "Effort estimation models using evolutionary learning algorithms for software development," in *Proc. Symp. Colossal Data Anal. Netw. (CDAN)*, Mar. 2016, pp. 1–6.

[27] M. Azzeh, "Software effort estimation based on optimized model tree," in *Proc. 7th Int. Conf. Predictive Models Softw. Eng. (PROMISE)*, 2011, pp. 20–21.

[28] I. A. P. Tierno and D. J. Nunes, "An extended assessment of data-driven Bayesian networks in software effort prediction," in *Proc. 27th Brazilian Symp. Softw. Eng. (SBES)*, Oct. 2013, pp. 157–166.

[29] L. L. Minku and X. Yao, "Ensembles and locality: Insight on improving software effort estimation," *Inf. Softw. Technol.*, vol. 55, no. 8, pp. 1512–1528, Aug. 2013.

[30] C.-L. Huang and C.-J. Wang, "A GA-based feature selection and parameters optimizationfor support vector machines," *Expert Syst. Appl.*, vol. 31, no. 2, pp. 231–240, Aug. 2006.

[31] D. T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, and M. Zaidi, "The bees algorithm—A novel tool for complex optimisation problems," in *Intelligent Production Machines and Systems*. Amsterdam, The Netherlands: Elsevier, 2006, pp. 454–459.

[32] M. T. Rahman and M. M. Islam, "A comparison of machine learning algorithms to estimate effort in varying sized software," in *Proc. IEEE Region Symp. (TENSYMP)*, Jun. 2019, pp. 137–142.

[33] T. R. Benala and R. Bandarupalli, "Least square support vector machine in analogy-based software development effort estimation," in *Proc. Int. Conf. Recent Adv. Innov. Eng. (ICRAIE)*, Dec. 2016.

[34] J. S. Shirabad and T. J. Menzies. (2005). *The PROMISE Repository of Software Engineering Databases*. Accessed: Oct. 24, 2020. [Online]. Available: http://promise.site.uottawa.ca/SERepository

[35] S. Boslaugh, *Statistics in a Nutshell: A Desktop Quick Reference*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2012.

[36] W. O. Bussab and P. A. Morettin, *Estatística Básica*, 9th ed. Pinheiros, Brazil: Saraiva, 2017.

[37] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 2, Jul. 2004, pp. 985–990.

[38] Q.-Y. Zhu, A. K. Qin, P. N. Suganthan, and G.-B. Huang, ''Evolutionary extreme learning machine,'' *Pattern Recognit.*, vol. 38, no. 10, pp. 1759–1763, Oct. 2005.

[39] D. C. Lambert. (2013). Basic ELM algorithms. Simple BSD. Accessed: Oct. 24, 2020. [Online]. Available: https://personal.ntu.edu.sg/egbhuang/elm_codes.html

[40] S. Bhatia and V. K. Attri, ''Machine learning techniques in software effort estimation using COCOMO dataset,'' *J. Comput. Sci. Eng.*, vol. 2, no. 6, pp. 101–106, 2015.

[41] K. V. Kumar, V. Ravi, M. Carr, and N. R. Kiran, ''Software development cost estimation using wavelet neural networks,'' *J. Syst. Softw.*, vol. 81, no. 11, pp. 1853–1867, Nov. 2008.

[42] NumPy. (2021). *NumPy V1.20, NumPy Project and Community*. Accessed: Feb. 14, 2021. [Online]. Available: https://numpy.org/

[43] Pandas. (2021). *Pandas V.1.2.2*. Accessed: Feb. 14, 2021. [Online]. Available: https://pandas.pydata.org/

[44] Scikit. (2021). *Scikit-Learn V.0.24*. Accessed: Feb. 14, 2021. [Online]. Available: https://scikit-learn.org/stable/

[45] Matplotlib. (2021). *Matplotlib V.3.3.4*. Accessed: Feb. 14, 2021. [Online]. Available: https://matplotlib.org/stable/index.html

[46] M. Shepperd and S. MacDonell, ''Evaluating prediction systems in software project estimation,'' *Inf. Softw. Technol.*, vol. 54, no. 8, pp. 820–827, Aug. 2012.

[47] Z. Hanusz and J. Tarasińska, ''Normalization of the Kolmogorov–Smirnov and Shapiro–Wilk tests of normality,'' *Biometrical Lett.*, vol. 52, no. 2, pp. 85–93, Dec. 2015.

[48] F. Wilcoxon, ''Individual comparisons by ranking methods,'' *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, 1945.

[49] P. Runeson and M. Höst, ''Guidelines for conducting and reporting case study research in software engineering,'' *Empirical Softw. Eng.*, vol. 14, no. 2, pp. 131–164, Apr. 2009.

HALCYON DAVYS PEREIRA DE CARVALHO was born in Brasília, Distrito Federal, Brazil, in 1980. He received the bachelor's degree in information systems from the Integrated University of Recife, PE, Brazil, in 2008. He is currently pursuing the graduate degree in computer engineering with the University of Pernambuco, Recife. He is experienced in project management for ten years in the information technology (IT) area. Currently, he is a Project Manager with Tribunal Regional Federal da 5ª Região, Recife. He is responsible for implementing the Project Management Office and Portfolio Management of the Organization.

ROBERTA FAGUNDES (Member, IEEE) received the graduate degree in telematics technology from the Federal Center for Technological Education of Paraíba (CEFET-PB), in 2002, the master's and Ph.D. degrees in computer science, and the Ph.D. degree in statistics from the Federal University of Pernambuco (UFPE), Brazil, in 2006, 2013, and 2015, respectively. She is currently an Adjunct Professor in information systems and computer engineering with the University of Pernambuco (UPE), Brazil. She is also a Vice-Coordinator and a Professor of the Graduate Program in Computer Engineering, (PPGEC), where there are masters and doctorate courses. Her research interest includes computer science, with an emphasis on computational intelligence.

WYLLIAMS SANTOS received the M.Sc. and Ph.D. degrees in computer science from the Informatics Center (CIn), Federal University of Pernambuco (UFPE), Brazil, in 2011 and 2018, respectively. From 2015 to 2016, he undertook his joint Ph.D. research with the Department of Computer Science and Information Systems (CSIS), University of Limerick, Ireland, and in collaboration with Lero—the Irish Software Research Centre. He is currently an Adjunct Professor with the University of Pernambuco (UPE), Brazil, where he leads the REACT Research Laboratories. His research interests include the management of software projects, agile software development, technical debt, and industry–academia collaboration.