ISSN: 2321-2152

IJMECE

International Journal of modern electronics and communication engineering

E-Mail
editor.ijmece@gmail.com
editor@ijmece.com

www.ijmece.com

ISSN2321-2152

www.ijmece .com

Vol 12, Issue 2, 2024

SOFTWARE DEVELOPMENT EFFORT ESTIMATION USING EXTREME LEARNING MACHINE

[*1]Chenga Jhansi, [*2]Perati Mani Preetham, [*3]Kasireddy Srinivas Reddy, [*4]Abdul Razzaq, [*5]Mr. Lalam Ramu,[*6]Dr.K.Vasanth Kumar

Department of Computer Science and Engineering(IOT), Malla Reddy Engineering College, Maisammaguda, Secunderabad, Telangana, India 500100

ABSTRACT

In the field of software engineering, the project management approach has been utilized to help managers maintain project control. Estimating the amount of work needed to finish a project with precision and dependability is one of the key steps in software engineering. The article's goals are to: i) use the correlation to find the variables that affect the estimation; and ii) apply the Extreme Learning Machine (ELM) model for effort estimation and compare it with other models in the literature. Therefore, the method with the highest accuracy for effort prediction was studied. Based on statistical tests and absolute mean residue (MAR) criteria predictive precision, the models were compared to one another. The study's primary conclusions were that: i) there are significant effort estimation factors; and ii) the ELM model outperforms other models in the literature in terms of software design effort estimation. Thus, the application of machine learning methods in the endeavor the likelihood that the project's price and time estimates will be accurate can be increased by using an estimation procedure.

Key Words: Extreme Learning Machine, Machine Learning, Effort Estimation, Software Development, Project Management

1. INTRODUCTION

For many years, forecasting models for software development efforts have been assessed in order to satisfy the demands of the software sectors [1]. One of the project management procedures that determines how much work is needed to finish the project is estimating the software development effort [2]. As a result, one of the most important factors in lowering risks and raising project success rates is accurate effort estimation [3].The process of managing software projects involves carrying out tasks using strategies to meet the goals of the projects [4]. In order to satisfy customer needs, both quantitative and qualitative metrics are estimated during the software development phase. Successful software measurement programs, in accordance with Saraiva [5], enable users to comprehend data and facilitate decision-making.

There are various methods for calculating the effort. In order to deliver the product within the projected time and cost, the business must determine which technique is best [6]. Expert Judgment is one method that project managers employ to estimate the amount of work required to build software [7]. Still, because human error can occur, this kind of approach has its limitations. Since different project development lifecycle models need a varied amount of effort at each stage of the process, estimating software effort is one of the biggest issues faced by developers and project managers in the field of software engineering [8]. Conventional estimates [9] are more complicated and time-consuming since they necessitate an effort to document operations. Furthermore, it is not always possible to foresee a wide range of elements and their relationships, the experience of software

engineers, and the software team's project history in the same business field [10].

Therefore, because machine learning (ML) has the ability to learn and change its behavior on its own, it presents an effective alternative for Software Development Effort Estimation (SDEE). Additionally, they support decision-making based on data analysis with the least amount of professional human intervention [3]. As a result, experts focus more of their time on other system chores that delight customers and less time estimating the project [1].
Furthermore, the project's initial phase objectives are not always well stated. The project's effective management is impacted by the uncertainty around the essential estimates for the project's development. A accurate assessment of the work required during the first stage of the software development life cycle is necessary to determine how much funding will be allocated for the project's development [11]. Fadhil [12] asserts that the process of estimating project expenses from the outset is very important since it is required for calculating the resources and budget needed to complete the project.

Put another way, the primary causes of software project failure are ambiguities surrounding system requirements and a lackluster assessment of the time, money, and labor needed to complete the project. Consequently, during the past few years, software development efforts have been predicted using Machine Learning (ML) approaches in an effort to reduce uncertainty during the software development cycle [3] [13].Asad Ali and Carmine Gravino's Systematic Literature Review (SLR) [1] states that over the past ten years, a number of machine learning techniques, including 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), have been used to predict the software development effort.

In this article, the software work is estimated using five machine learning algorithms: Extreme Learning Machine (ELM), Support Vector Machine (SVM), K-Nearest Neighbors (kNNs), Multilayer Perceptron (MLP), and Linear Regression (LR). KNN is a non-parametric method used for regression and classification problems, where the population size may cause the algorithm to execute slowly and use a lot of memory [14]. Although SVM is a technique that may be used to model both linear and nonlinear problems, training it on huge amounts of data can be time-consuming [13]. According to [15], MLP networks are also employed for regression and classification. However, a number of factors, including the number of hidden layers, the number of neurons in each hidden layer, the number of training periods, and the learning rate, need to be carefully considered.

Numerous modified ELM algorithms, such as Integrated Multiple Kernel ELM (IMK-ELM) [19], Multilayer Extreme Learning Machines (ML-ELM) [18], Residual Compensation ELM (RC-ELM) [16], and Robust ELM (R-ELM) [17], have been proposed recently. These previously mentioned algorithms have been widely used in a variety of fields, including deep learning, the free location of devices in environments that are disordered using spatiotemporal information, the prediction of the rate of gas used in the blast furnace during the manufacture of iron, and the treatment of tasks with Gaussian and nonGaussian noise.
Out of the 75 primary papers that the Systematic Literature Review (SLR) [1] has identified, only one [20] of them use the ELM technique for Software Development Effort Estimation (SDEE). The SLR also shows that feedforward networks are the most often used kind of Artificial Neural Networks (ANN) among other ANN kinds.

However, Huang [21] claims that training an ELM can result in higher generalization performance since it can be faster than training a neural network by backpropagation. In this way, the benefits of applying the ELM technique for Software Development Effort Estimation (SDEE) form the basis of our study.

In order to address the two study issues addressed in the part cited in parenthesis, this paper additionally looked at which attributes should be chosen to get the best outcomes in the effort estimates and how accurate the software effort estimate is. RQ1: What characteristics are crucial to get better estimates of the work required to develop software? (Part III-C) RQ2: Which machine learning method has the best accuracy in effort estimation? Section V

To address the first question, an analysis of Pearson's correlation coefficient [22] between the variables and their significance was conducted. RQ1. Lastly, an analysis was conducted to compare the model ELM's performance with the models for software effort estimating that have been studied in the literature [11] [23] [24] [25] [26] [27] [28] [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, the primary contribution of this paper is to show how to use the ELM technique to build a machine learning model and apply it to a dataset used to estimate software development efforts. It produces outcomes that are on par with or better than those of the models studied in the literature. In addition, to improve the field of software engineering by saving money and time during the project life cycle stages. The structure of this document is as follows. Related works are included in Section II. The research approach is presented in Section III. The model accuracy measurements are presented in Section IV. The experiment findings are shown in Section V. There are challenges to validity in Section VI. Section VII deals with closing thoughts and upcoming projects.

2. RELATED WORK

The investigation carried out by [23] conducted a comparative analysis of different machine learning methods for predicting software development effort, including Linear Regression (LR), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Multi-Layer Perceptron Neural Network (MLPNN). They utilized the determination coefficient (R2) to evaluate these models. Meanwhile, [24] employed a Neural Feedforward Network to enhance the accuracy of Software Development Effort Estimation. Another study, [28], assessed the effectiveness of Bayesian Networks (BNs) in predicting software effort through rigorous validation procedures.

Oliveira [25] employed the Genetic Algorithm (GA) as a Machine Learning (ML) technique, while [30] proposed a GA-based feature model, both aimed at selecting input resources and optimizing parameters for ML techniques in estimating software development efforts. Additionally, [27] explored the use of the Bees Algorithm [31] to select parameters for the Model Tree (MT) depending on the dataset used. Conversely, [26] introduced non-algorithmic techniques for effort estimation, noting the superior performance of evolutionary algorithms.

Minku's work [29] involved experimental studies on automated machine learning ensembles, showing promising performance across various datasets. Rahman's research [32] implemented different machine learning algorithms, such as Radial Based Function Neural Network (RBFNN), Extreme Learning Machine (ELM), and Decision Tree (DT), for effort estimation based on software size categories. ISBSG

Release 11 dataset was used for training and testing these algorithms, revealing varying degrees of effectiveness across different software sizes.

In another study [33], a case-based reasoning model hybridized with machine learning models (ABE-LS-SVM, ABE-ELM, ABE-ANN) was utilized, with ABE-LS-SVM showing superior performance across multiple datasets.

Extreme Learning Machine (ELM) and Linear Least Squares Regression (LSR) were applied in [20] and [11] to estimate effort for a set of small programs. Mean Magnitude of Relative Error (MMRE) was used for performance analysis, with adjustments made to ELM parameters based on prediction errors. Other metrics such as MAE, MSE, and RMSE were also used to assess model performance. The Desahanais dataset was added to confirm the effectiveness of the applied ELM model. Statistical tests, including normality and hypothesis tests, were conducted to compare the proposed model's performance with others in the literature, along with temporal performance tests and the presentation of results using Box-Plot graphs, showing significantly improved outcomes.

3. RESEARCH METHODOLOGY

This section outlines the methodology employed to construct the ELM model for software effort estimation. Figure 1 illustrates the proposed methodology for precise effort prediction, comprising 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), and Effort Forecasting (Section V).

A. ANALYSIS OF SYSTEMATIC LITERATURE REVIEW

The development of the effort estimation model for software development was guided by the Systematic Literature Review (SLR) conducted by [1]. The SLR encompassed empirical studies published from January 1991 to December 2017, resulting in the selection of 75 primary studies based on specific criteria. Among the machine learning (ML) techniques used in these studies, Artificial Neural Network (ANN) was the most prevalent, employed in 60% of cases, followed by Support Vector Machine (SVM) at 25%, and Case-Based Reasoning (CBR) at 17%. Bayesian Network (BN), K-Nearest Neighbors (KNNs), Decision Tree (DT), Genetic Programming (GP), Classification and Regression Tree (CART), and Random Forest (RF) were less frequently utilized.

The databases predominantly utilized in the 75 primary studies were NASA (23%), COCOMO, and ISBSG, each accounting for 21% of references, followed by Desharnais (19%). Kemerer, Maxwell, Albrecht, Tukutuku, Finnish, and Miyazaki were the least cited databases.

Primary studies from the past decade were selected based on the systematic review. Figure 2(a) highlights the Top 10 machine learning techniques, with ANN and MLP cited in 11 studies each, and SVR in 8 reviews, as the most frequently mentioned. Figure 2(b) showcases the Top 10 datasets used during this period, with the COCOMO dataset mentioned in 20 studies, followed by ISBSG and NASA in 11 and 8 reviews, respectively.

Among various types of Artificial Neural Networks (ANN), the ELM technique was mentioned in only one primary study [20]. As per Huang [21], training an ELM can be faster compared to training a neural network via backpropagation, often resulting in better generalization performance. Given these advantages for Software Development Effort Estimation (SDEE), the ELM technique was chosen for modeling in this study.

Datasets were scrutinized for the selection of the appropriate technique to estimate

ISSN2321-2152
www.ijmece .com
Vol 12, Issue 2, 2024

software development effort. The Desharnais dataset [34] was chosen, as it is relatively underexplored among effort estimation models (Section III-B1).

The software development estimation generated by our model will be compared with other models in the literature. Additionally, Carvalho's dataset referenced in Section III-B2 will be utilized to compare our model's performance with the proposed models in Carvalho's article [11].

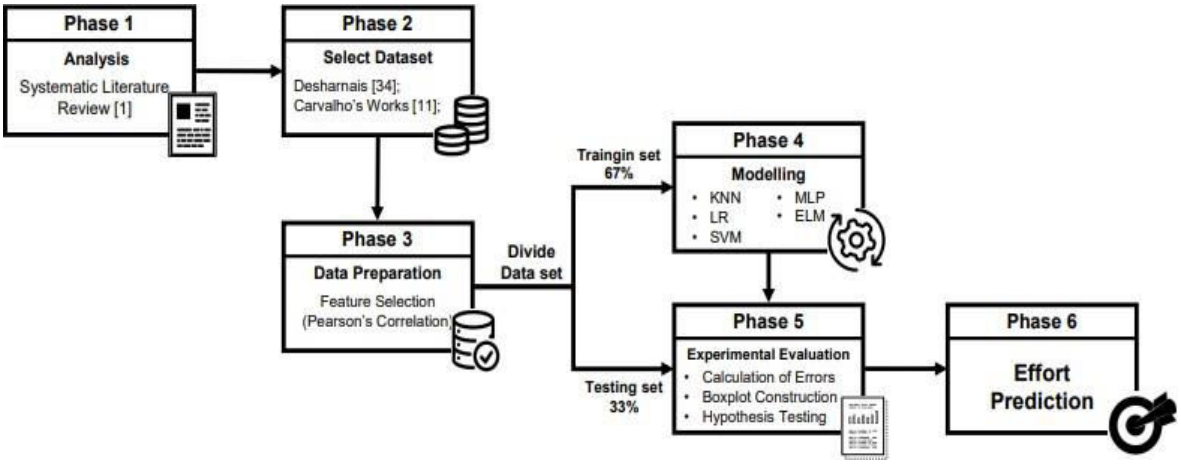

Figure 1: Proposed Methodology for Accurate Effort Prediction

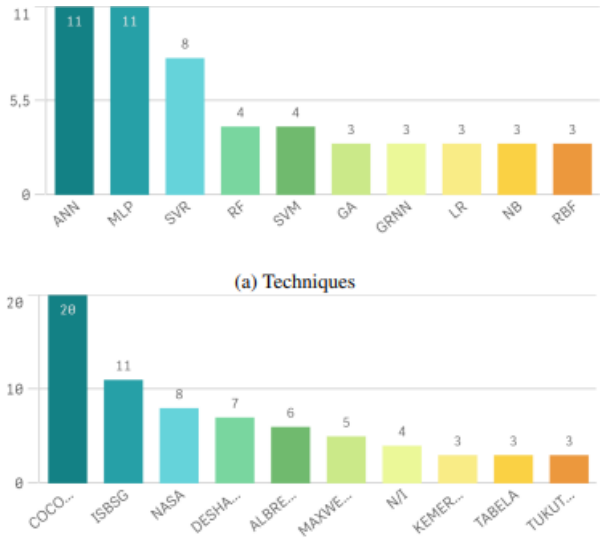

Figure 2: Top 10 of the Techniques and Dataset used in the last 10 years

B. CHOOSING THE DATASET

This subsection provides a descriptive analysis of the datasets utilized in constructing our model:

1) Desharnais Dataset

The Desharnais dataset [34] comprises information from 81 software projects from a Canadian company. Each project includes 12 attributes: Project id, Team Experience, Manager Experience, Year End, Length, Effort, Transaction, Entities, Point Adj, Adjustment, Point Non-Adjust, and Language, categorized into numeric and categorical attributes, as outlined in Table 1.

Based on Table 1, only the numerical data was selected, excluding the "Project" attribute as it is not correlated with the project effort estimate. With the selected attributes, the subsequent step was to identify the independent variables and the dependent variable. Following software development effort estimation models, the primary variable is the effort required to complete the project. In this case, the dependent attribute is "Effort," with the

ISSN2321-2152
www.ijmece .com
Vol 12, Issue 2, 2024

correlated independent attributes being "TeamExp," "ManagerExp," "YearEnd," "Length," "Transactions," "Entities," "PointsNonAjust," "Envergure," and "PointsAdj."

Table 2 presents statistical measures for all independent and dependent attributes. The elapsed time (Length) of the 81 measured projects ranged from 1 to 39 months, with an average of 11.7 months. Two attributes representing the software size, "PointsNonAdjust" and "PointsAdjust," have a minor difference, with an average of 304 for PointsNonAdjust and 289 for PointsAdjust. The recorded level of effort ranged from 546 to 23,940 person-hours, with an average of 5,046.31 person-hours.

Figure 3 depicts a histogram illustrating the distribution of data on effort, the dependent variable, measured in person-hours. The variables exhibit positive skewness, with the majority of records situated towards lower values and some outliers with very high values.

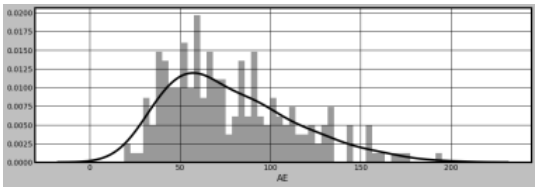

Figure 3: Distribution of the effort value in the Desharnais database

2) Carvalho's Work [11]

The second set of analyzed data is sourced from [11], encompassing 231 software projects with 5 attributes categorized into numeric and string types: "P" (Program Number), "N&C" (New and Changed Code), comprising added and modified code, "R" (Reused Code), recognized as physical lines of code (LOC), and "AE" (Actual Effort), measured in minutes, are construction of the model, effectively addressing the research question concerning numeric types, while "DP" (Developer Code) is a string type.In this dataset, only the numeric attributes were selected, and the "P" attribute was excluded as it is unrelated to the effort estimate. Descriptive statistics for the selected attributes are presented in Table 3. During dataset analysis, two independent attributes, "N&C" (New and Changed Code) and "R" (Reused Code), were identified, along with the target attribute, "AE" (Actual Effort), as the dependent variable.

Figure 4 illustrates a histogram depicting the distribution of the dependent variable (AE-effort) in relation to effort, measured in minutes. A normal distribution trend is observed in the histogram, with the highest concentration of data around the average, and the frequency close to the limits.

C. DATA PREPARATION

During the data preparation phase, the study employed Pearson's correlation coefficient, also known as linear correlation, to assess the degree of relationship between two quantitative variables. This coefficient ranges from -1 to 1, with values closer to zero indicating no relationship, those closer to 1 or -1 suggesting strong positive or negative correlations, respectively. In this analysis, correlations exceeding 0.5 were deemed significant, indicating a high correlation level.

Specifically, attributes such as "Transactions," "Length," "Entities," "PointsAdj," and "PointsNonAdjust" exhibited correlation coefficients of 0.514, 0.586, 0.655, 0.716, and 0.733, respectively, in relation to the Effort variable in the Desharnais dataset, all surpassing the threshold of 0.5. Consequently, these attributes were identified as statistically significant for the

the importance of features in estimating software development effort.

ISSN2321-2152
www.ijmece .com
Vol 12, Issue 2, 2024

To visually represent these correlations, a scatter plot was generated, illustrating the magnitude of correlation between the aforementioned attributes and the Effort variable. In such plots, positive correlation is indicated by a clustering of points in an upward trend, while negative correlation is depicted by points concentrated along a downward line.

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 1: Desharnais Dataset Variables

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	1127.00
Adjustment	27.630	28.000	10.592	5.000	52.000
PointsAjust	289.235	255.000	185.761	62.000	1116.00
Effort	5046.309	3647.000	4418.767	546.000	23940.0

Table 2: Descriptive Statistics for the Desharnais 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

Table 3: Descriptive Statistics for the Carvalho's Work Dataset

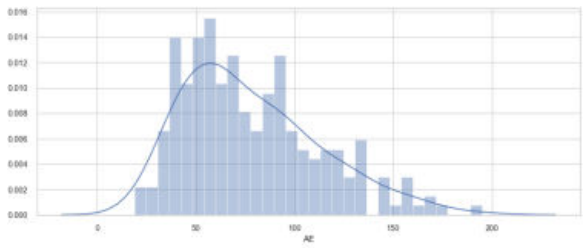

FIGURE 4. Effort Value Distribution

Figure 4: Effort Value Distribution

We noted 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 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.

ISSN2321-2152
www.ijmece .com
Vol 12, Issue 2, 2024

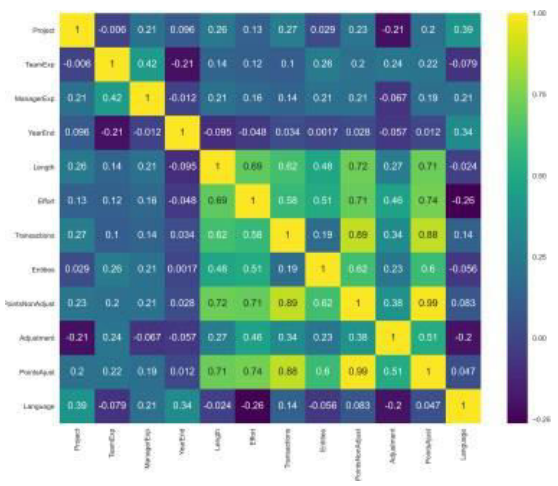

Figure 5: Pearson Correlation Desharnais Dataset

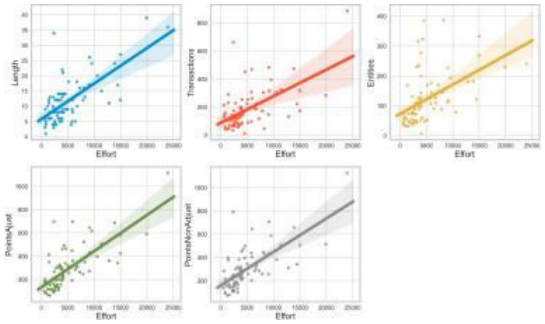

Figure 6: Correlation coefficients between variables in the Desharnais database

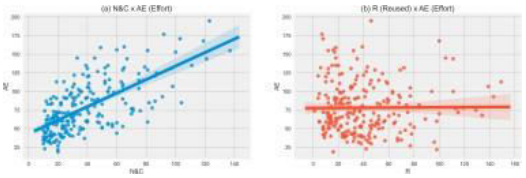

Figure 7: Correlation Carvalho's Work [11] Dataset

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 MultiLayer Perceptron (MLP). 1) Extreme Learning Machine The Extreme Learning Machine (ELM) algorithm was proposed by [37]. Its architecture is equivalent to a Singlelayer 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 N˜ neurons. To learn N different arbitrary samples (xi , ti), where xi = [xi1, xi2, ..., xin] T R n and ti = [ti1, ti2, ..., tim] T 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:

$$\sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + \mathbf{b}_i) = \mathbf{o}_j, j = 1, ..., N, \qquad (1)$$

where N˜ is the number of neurons in the hidden layer and N is the number of training samples, βi = [bi1, bi2, ..., bim] T represents the weight vector that connects the i-th hidden layer neuron to neurons of the output layer, g(·) is the activation function, wi = [wi1, wi2, ..., win] T is the weight vector that connects the j-th hidden layer neuron and the input layer neurons, xj represents each distinct sample and bj 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 βi , wi and bi so that the equation can be written as:

$$\sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + \mathbf{b}_i) = \mathbf{t}_j, j = 1, ..., N, \qquad (2)$$

where tj are the outputs expected by the network, referring to the xj sample input.

ISSN2321-2152
www.ijmece .com
Vol 12, Issue 2, 2024

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

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

where

$$\mathbf{H} = \begin{bmatrix} g(\mathbf{w}_i \cdot \mathbf{x}_j + \mathbf{b}_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_j + \mathbf{b}_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(\mathbf{w}_i \cdot \mathbf{x}_N + \mathbf{b}_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_N + \mathbf{b}_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}, \tag{4}$$

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

and

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \cdot \\ \cdot \\ \cdot \\ t_N^T \end{bmatrix}_{N \times m} \tag{6}$$

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 x1, x2,. . . , xN . 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) [41]. 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 [41]. 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 the output layer, the number of nodes depends on the solution, the number of dependent variables [42]. To implement the models, we use the Python language such as the libraries: Numpy [43], Pandas [44], Scikit-Learn [45], and Matplotlib [46].

4. 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 [47], 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

ISSN2321-2152
www.ijmece .com
Vol 12, Issue 2, 2024

(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.

$$MRE = \frac{|y_i - \hat{y}_i|}{y_i} \quad (7)$$

where yi is the actual effort, and yˆi 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

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i \quad (8)$$

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

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

yi is the ith value of the variable being predicted, yˆi its estimate, yi − yˆi 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.

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

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

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

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 [48] and Wilcoxon [49] tests. It was also used relative p-value to evaluate the performance of the models.

5. RESULTS AND DISCUSSION

This section delves into the results derived from experiments conducted using both the Desharnais [34] dataset and the dataset from Carvalho's [11] work. The model employed is based on the ELM technique as per the configurations outlined previously. Algorithm 1 provides the pseudocode for experimental evaluation.

A. DESHARNAIS DATASET

Various techniques including KNN, LR, SRV, MLP, and ELM were applied to the Desharnais [34] dataset. Mean and standard deviation (SD) were analyzed for all metrics, considering the 500 simulations conducted for this dataset. Table 4 presents the mean results, with the standard deviation (SD) shown in parentheses representing the standard deviation of errors in the dataset.

Figure 8 illustrates the MAE boxplot graph for each model, revealing outliers in all regression models except for the LR model. Notably, the KNN, LR, SVM, and MLP models are overestimated compared to the applied ELM model. Thus, it's evident that the applied ELM model yielded the best results owing to its simplicity of usage, faster learning speed, and superior generalization performance. Furthermore, the box and tail (boxplot) of the applied ELM model are less distorted compared to those of other commonly used models in the literature.

Analyzing the results presented in Table 4, it can be inferred that the ELM model achieved the lowest error rates, confirming its superior performance. However, in most cases, validation of the results required hypotheses to be conducted due to the minimal gains observed in the second decimal place.

ISSN2321-2152
www.ijmece .com
Vol 12, Issue 2, 2024

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**

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

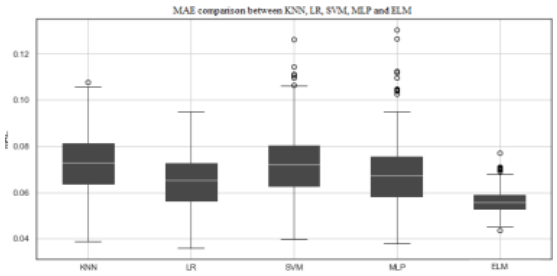

FIGURE 8: Boxplots of Mean Absolute Error.

Before performing the hypothesis test, it was verified the existence of normality between the values using the ShapiroWilk test [48]. Since the dataset does not have a normal distribution, the Wilcoxon hypothesis test [49] 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 Equation 12

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

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.

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	$7.779x10^{-23}$

Table 5: Results p-value

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

Similarly, the ELM technique was applied to the dataset used by [11], considering two independent variables, N&C and R. For this dataset, 1000 simulations were conducted, matching the number used by the author. Table 7 displays the mean error results, with standard deviation shown in parentheses (SD) for errors in the dataset. The applied ELM model with 2 and 5 hidden layers

ISSN2321-2152
www.ijmece .com
Vol 12, Issue 2, 2024

exhibited greater stability and reliable generalization performance.

Figure 9 depicts the boxplot graph of the ELM Models with 2 and 5 hidden layers. The "ELM with 2 and 5 n_hidden [11]" exhibited a mean with greater variability compared to the "ELM with 2 and 5 _hidden applied" models. Additionally, data dispersion in the "ELM with 2 and 5 _hidden applied" models is minimal, showcasing remarkable generalization performance.

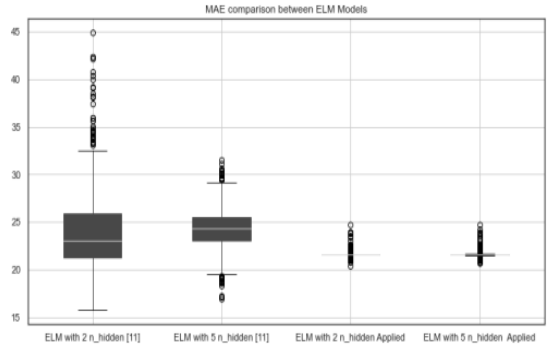

Figure 9: MAE comparison between ELM models.

C. PERFORMANCE EVALUATION

This section compares the performance of the applied ELM learning algorithm with KNN, LR, SVM algorithms, and the conventional back-propagation (MLP) algorithm. All algorithm simulations were executed using packages developed in Python language and run in the Jupyter Notebook environment, on an Intel(R) Core(TM) i7 2.1 GHz CPU with 8 GB RAM. To evaluate algorithm time consumption, simulations were conducted with 500 iterations, 1,000 iterations, 5,000 iterations, and 10,000 iterations. Figure 10 presents the comparison of time consumption speed of the evaluated models. The ELM algorithm outperformed the KNN and MLP models in terms of speed and demonstrated nearly identical performance to the LR and SVM algorithms. Based on the presented values, the ELM model was approximately three times faster than the backpropagation algorithm, MLP.

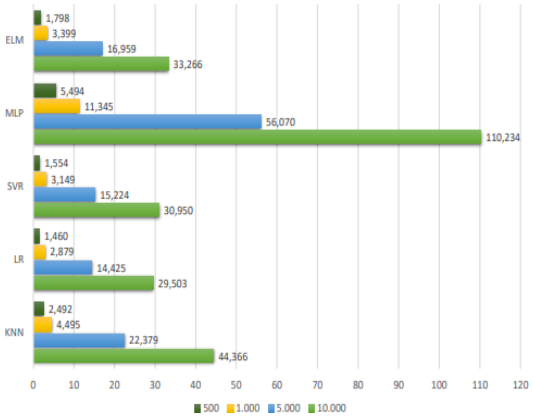

Figure 10: Comparison of time consumption of the applied methods .

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)

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

Finally, let's answer our last research question:

RQ2: Which machine learning technique excels in effort estimation accuracy?

Throughout our investigation, we explored five machine learning techniques utilized in datasets to estimate software development effort: (i) Linear Regression (LR), (ii) Support Vector Machine (SVM), (iii) K-Nearest Neighbor (KNN), (iv) Multilayer Perceptron (MLP), and (v) Extreme Learning Machine (ELM). Through simulations, we assessed the accuracy of effort estimation precision for each technique, with ELM emerging as the most promising, as depicted in Table 4, when compared to other techniques commonly employed in the literature.

Moreover, it's noteworthy that the integration of machine learning in Software Engineering holds significant potential benefits. Given that Software Engineering aims to achieve quality results outlined in the project management plan, including adhering to schedules (effort) and budgets (cost), the utilization of machine learning techniques for predicting software development efforts can aid project teams in addressing uncertainties in estimates throughout the project lifecycle. This, in turn, can contribute to delivering higher quality project outcomes in terms of effort and cost.

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 6: Critical Evaluation Table of Related Work

6. THREATS TO VALIDITY

In this section, we will address the validity of our study, considering internal and external threats as well as construct validity [50]. Internal validity pertains to the examination of causal relations [50]. A potential threat to internal validity is the selection of ELM algorithm parameters. In our study, we addressed this by explicitly considering parameter selection as a step in dealing with internal validity. For each dataset utilized in the research, parameter adjustments were necessary, given the absence of established guidelines on determining these parameters for each dataset.

We randomly divided the data into training and test sets in a 67% to 33% proportion, respectively. Random assignment of data can significantly influence model results. However, since all models were executed on the same datasets, this should not significantly impact the overall study, as the objective is to compare model performance on the applied dataset.

External validity concerns the generalization of study results beyond the study to other scenarios [50]. One challenge related to external validation in machine learning is the limited number of samples available in datasets. Due to the small size of some datasets, the number of data points available for testing is further constrained. Additionally, the availability of datasets from free software projects poses a limitation, as data availability is often scarce and sometimes requires payment, hindering forecasting efforts due to a limited amount of data.

Construction validity refers to the extent to which the operational measures studied represent the researcher's intentions and align with the research questions [50]. One metric used to evaluate model accuracy in estimating software development effort was the Mean Magnitude of Relative Error (MMRE). However, according to Shepperd and MacDonell [47], MMRE lacks precision in forecasting software effort estimation. Despite being used in this study, MMRE is solely employed for comparison with other results in the literature. We also utilized the Mean Absolute Error (MAE), as proposed by [47], as a more reliable metric for measuring the accuracy of the software effort estimation model.

7. CONCLUSION AND FUTURE WORK

Obtaining a reliable and accurate estimate of software development effort has long been a challenge in Software Engineering. Accurate estimation of effort and costs in the project's initial phases would greatly benefit the field. To address this, we applied a

machine learning technique, the Extreme Learning Machine (ELM), for estimating software development effort, comparing it with other effort estimation models found in the literature. The implemented model will aid Specialists and Project Managers in forecasting effort estimates, thus reducing time and costs during the project execution phase.

In our simulations, we utilized two datasets from software projects reflecting real-world scenarios: the Desharnais [34] dataset containing 81 software projects from a Canadian company, and the dataset evaluated in Carvalho's work [11], comprising 231 software projects. For both datasets, the data were divided into training and testing sets in a 67% to 33% proportion, respectively. Inputs were normalized to the interval [0.15, 0.85], and data were randomly assigned to training and testing sets to ensure unbiased results.

While many projects traditionally use the Mean Magnitude of Relative Error (MMRE) to assess forecasting method accuracy in estimating software project effort, we adopted alternative metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) to evaluate the ELM model, following recommendations from Shepperd and MacDonell [47].

In the Desharnais [34] dataset, we conducted experiments comparing the ELM technique with other techniques in the literature, including Linear Regression (LR), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Multilayer Perceptron (MLP). Similarly, in the dataset evaluated by Carvalho's work [11], experiments were conducted, comparing our model with the author's model and adjusting parameters to achieve better results.

A significant contribution of this study is the comparison of the applied ELM model with literature models used for predicting software development effort estimates, leading to reduced error estimate rates and aiding project managers in improving effort estimate forecasts during the project lifecycle. Additionally, we used Pearson's correlation coefficient during the data preparation process to identify variables with high correlation, resulting in the selection of potential variables for the applied effort estimation model and achieving better results compared to the literature.

In conclusion, accurate and reliable estimation of the effort required to complete a project is crucial in Software Engineering. Employing machine learning techniques increases the project's chances of success while reducing time and costs. For future work, we propose using an optimization method, such as Particle Swarm Optimization (PSO), to optimize model parameters for estimating 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, p. 1–25, 2019.

[2] PMI, "Project Management Body of Knowledge (PMBOK®)", 6th ed. Newtwn Square: Project Management Institute, Inc, 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. 1–1, 2021.

[4] R. S. Pressman, "Software Engineering: A Practitioner's Approach", 8th ed. McGraw-Hill, 2016.

[5] R. Saraiva et al., "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, p. 1–19, 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, p. 724–732, 2011.

[8] M. Hammad and A. Alqaddoumi, "Features-level software effort estimation using machine learning algorithms." 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies, 3ICT 2018, p. 1–3, 2018.

[9] A. B. Nassif, et al. "Neural network models for software development effort estimation: a comparative study." Neural Computing and Applications, v. 27, n. 8, p. 2369–2381, 2016. [10] V. Yurdakurban, N. Erdogan, "Comparison of machine learning methods ˇ for software project effort estimation | Yazilim projelerinde i¸s gücü tahmini için makine ögrenmesi yöntemlerinin kar¸sila¸stirilmasi." ˇ 26th IEEE Signal Processing and Communications Applications Conference, SIU 2018, p. 1–4, 2018.

[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, p. 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, p. 184–196, 2018.

[14] Y. Song, et al. "An efficient instance selection algorithm for k nearest neighbor regression." Neurocomputing, v. 251, p. 26–34, 2017.

[15] M. Hosni, et al. "Heterogeneous Ensembles for Software Development Effort Estimation." Proceedings - 2016 3rd International Conference on Soft Computing and Machine Intelligence, ISCMI 2016, p. 174–178, 2017.

[16] J. Zhang, et al. "Residual compensation extreme learning machine for regression." Neurocomputing, v. 311, p. 126–136, 2018.

[17] J. Zhang, et al. "Robust extreme learning machine for modeling with unknown noise.", Journal of the Franklin Institute, v. 357, n. 14, p. 9885–9908, 2020.

[18] J. Zhang, et al. "Non-iterative and Fast Deep Learning: Multilayer Extreme Learning Machines." Journal of the Franklin Institute, v. 357, n. 13, p. 8925–8955, 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 of Things Journal, v. 8, n. 6, p. 4749–4761, 2021. [20] S. K. Pillai, M. K. Jeyakumar, "Extreme Learning Machine for Software Development Effort Estimation of Small Programs", Int. Conf. Circuit, Power Comput. Technol., p. 1698–1703, 2014.

[21] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: Theory and applications”, Neurocomputing-Neural Networks Sel. Pap. from 7th Brazilian Symp. Neural Networks (SBRN ’04), vol. 70, no 1–3, p. 489–501, Dec. 2006.

[22] D. Montgomery and G. Runger, "Estatística Aplicada e Probabilidade para Engenheiros", LTC, 5o ed., 2012.

[23] S. Shukla and S. Kumar, “Applicability of Neural Network Based Models for Software Effort Estimation”, in 2019 IEEE World Congress on Services (SERVICES), 2019, vol. 2642–939X, p. 339–342.

[24] P. Jodpimai, P. Sophatsathit, and C. Lursinsap, “Estimating software effort with minimum features using neural functional approximation”, Proc. - 2010 10th Int. Conf. Comput. Sci. Its Appl. ICCSA 2010, p. 266–273, 2010.

[25] A. L. I. Oliveira, P. L. Braga, R. M. F. L. 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, p. 1155–1166, 2010.

[26] G. Gabrani and N. Saini, “Effort estimation models using evolutionary learning algorithms for software development”, 2016 Symp. Colossal Data Anal. Networking, CDAN 2016, 2016.

[27] M. Azzeh, “Software Effort Estimation Based on Optimized Model Tree Mohammad”, Proc. 7th Int. Conf. Predict. Model. Softw. Eng. PROMISE 2011, p. 20–21, 2011.

[28] I. A. P. Tierno and D. J. Nunes, “An extended assessment of data-driven Bayesian Networks in software effort prediction”, Proc. - 2013 27th Brazilian Symp. Softw. Eng. SBES 2013, p. 157–166, 2013.

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

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

[31] D. T. Pham et al., “The Bees Algorithm-A Novel Tool for Complex Optimisation Problems”, Intell. Prod. Mach. Syst., p. 454–459, 2006.

[32] M. T. Rahman and M. M. Islam, "A Comparison of Machine Learning Algorithms to Estimate Effort in Varying Sized Software." Proceedings of 2019 IEEE Region 10 Symposium, TENSYMP 2019, v. 7, p. 137–142, 2019.

[33] T. R. Benala and R. Bandarupalli, "Least Square Support Vector Machine in Analogy-Based software development effort estimation.", 2016 International Conference on Recent Advances and Innovations in Engineering, ICRAIE 2016, 2016.

[34] J. Sayyad Shirabad and T. J. Menzies, “The PROMISE Repository of Software Engineering Databases”, Promise, 2005. [Online]. Available at: http://promise.site.uottawa.ca/SERepository. Accessed on: Oct. 24, 2020.