ISSN 1980-5411 (On-line version)
Production, 35, e20240092, 2025
DOI: 10.1590/0103-6513.20240092

Research Article

Effort estimation for software products targeted at the manufacturing sector using machine learning algorithms

Diane Lenhart[a]*, Matheus Henrique Ribeiro[a], Flavio Trojan[a]

[a]Universidade Tecnológica Federal do Paraná, Pato Branco, PR, Brasil

*dianelenhartcontato@gmail.com

Abstract

Paper aims: This study seeks to investigate the accuracy of machine learning algorithms for estimation of the effort required for software development in the manufacturing sector to identify the most effective algorithms according to the nature and complexity of the data and the number of available attributes.

Originality: This work distinguishes itself from other studies in the field of effort prediction by utilizing a data repository that consists exclusively of projects from the manufacturing sector. This approach ensures that the specific characteristics of manufacturing projects are reflected in the predictions, addressing a gap in the existing literature. Another notable contribution of this study is the comparative analysis of various machine learning algorithms assessed under different dimensionality scenarios (three and five variables). Although this factor is crucial for enhancing effort estimation accuracy, it has received limited attention in the literature.

Research method: The investigated techniques in this work were (i) Support Vector Regression, (ii) Gradient Boosting Machines (GBM), (iii) eXtreme Gradient Boosting (XGBoost), (iv) Random Forest (RF), (v) Extreme Learning Machine (ELM); and (vi) Linear Regression (LR). Performance measures such as Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Coefficient of Determination (R^2) were used to compare the results achieved by each model, considering a dataset of 230 records originating from various countries.

Main findings: The comparison among machine learning models revealed significant performance variations depending on the number of variables and the evaluation metrics adopted. GBM stood out for its robustness in complex scenarios, while SVR achieved the lowest mean absolute error. ELM, in turn, proved effective with fewer variables but showed sensitivity to outliers and less stability in more complex contexts. Among all the techniques evaluated, XGB yielded the worst performance across all parameters.

Implications for theory and practice: This study contributes by applying these models to the manufacturing sector and comparing scenarios with three and five variables. The results support a more informed selection of models based on project complexity and data dimensionality. The more research conducted in this area, the stronger the theoretical and practical conclusions can be drawn.

Keywords

Software effort estimation. Software in the manufacturing sector. Software project management. Machine learning.

How to cite this article: Lenhart, D., Ribeiro, M. H., Trojan, F. (2025). Effort estimation for software products targeted at the manufacturing sector using machine learning algorithms. Production, 35, e20240092. DOI: https://doi.org/10.1590/0103-6513.20240092

Received: Sep. 5, 2024; Accepted: Sep. 22, 2025.

Financial Support
Author Ribeiro thanks the Araucária Foundation (Grant number: PRD2023361000550) for its financial support of this work.

Conflict of Interest
The authors have no conflict of interest to declare.

Ethical Statement
This research did not involve experiments with human participants, nor the collection of personal or sensitive data. Therefore, ethical approval and informed consent were not required.

Editor(s)
Adriana Leiras

1. Introduction

In software development, accurately estimating the effort required early in a project is essential for its success (Kassaymeh et al., 2024; Lavingia et al., 2024; Van Hai et al., 2022a). Software Development Effort Estimation (SDEE) involves forecasting the work and time needed to deliver a system within set constraints (López-Martín, 2022). Inaccurate estimates can disrupt later planning stages, reduce product quality, and lead to financial losses, especially when organizational needs are not met (Kaushik et al., 2020; Kumar et al., 2020).

The increasing availability of historical project data and the growing need for accurate forecasting in various scientific domains have driven the development of robust and efficient techniques capable of modeling stochastic dependencies between past and future observations (Bontempi et al., 2013).

Although similar techniques are frequently used in the literature, results often vary significantly. In the Artificial Intelligence field, for instance, Jiang et al. (2019) highlighted the effectiveness of Artificial Neural Networks. However, other studies have noted drawbacks such as slow convergence, low accuracy, and high sensitivity to initial parameters, often requiring metaheuristic algorithms to improve their performance (Kassaymeh et al., 2024). These issues contribute to increased complexity and longer implementation times.

In the context of Machine Learning (ML), various authors (Kassaymeh et al., 2024; Lavingia et al., 2024; Varshini & Kumari, 2024; Al-Betar et al., 2023; Rahman et al., 2023; Rao et al., 2024; Sharma & Vijayvargiya, 2020) have reported promising results in software effort estimation. Rankovic et al. (2021) emphasize that ML techniques are effective for SDEE due to their ability to learn from data with minimal human intervention. Nevertheless, there is still no consensus on the best ML method, as no single approach has proven to be universally optimal.

A review of the related literature also reveals a noticeable lack of studies specifically focused on SDEE in the manufacturing sector, which constitutes the central theme of this research. Moreover, many authors do not specify the types of variables used in model construction, hindering comparative analysis and limiting the applicability of results to other studies. It was further observed that most research efforts are conducted within the scope of specific systems or organizations, thereby restricting the generalizability of their findings.

This study aims to estimate software development effort in the manufacturing sector using international data from the ISBSG repository. A comparative analysis is performed on the performance of various machine learning models namely GBM, SVR, ELM, RF, LR, and XGBoost, using two variable sets (with three and five attributes) to evaluate the effect of dimensionality. Model performance is assessed based on R^2, RMSE, and MAE. The goal is to identify the most effective models in relation to data complexity and variable availability, supporting more accurate and context-sensitive effort estimations.

The objective of this study is to estimate the effort required for software projects in the manufacturing sector, using data from various countries available in the ISBSG repository. In this respect, a comparative analysis is conducted on the performance of different machine learning models in predicting software effort, considering two sets of variables (with three and five attributes) to assess the impact of dimensionality. The models analyzed include GBM, SVR, ELM, RF, LR, and XGBoost, and their performances are compared using R^2, RMSE, and MAE criteria. The study aims to identify the most effective models according to data complexity and the number of available variables, contributing to more accurate and context-aware effort estimations.

This research is justified by its focus on the manufacturing sector, which allows for integrating domain-specific characteristics into effort estimation. Using an international database enhances the representativeness of the results, making the predictions more accurate and generalizable. Additionally, the study compares different machine learning models, analyzing the impact of dimensionality (three and five variables) on predictive performance — an approach that has been underexplored in the literature. The results provide valuable insights for selecting algorithms best suited to the data complexity and application context.

The scientific contributions are as follows:

- Software effort estimation tailored explicitly for the manufacturing sector, integrating the unique characteristics of this domain and contributing to more realistic and context-aligned estimates for industrial applications.
- Utilization of an international database (ISBSG) comprising projects from various countries, enabling a comprehensive and representative analysis of different software development practices and realities.
- Comparative evaluation of multiple machine learning models, including SVR, GBM, XGBoost, RF, ELM, and LR, based on performance metrics such as R^2, RMSE, and MAE.
- Investigation of the impact of data dimensionality on model performance, with tests conducted in scenarios with three and five variables, offering insights into the trade-off between model simplicity and accuracy.

- Practical contribution to selecting predictive models in effort estimation, providing guidelines based on data nature and organizational objectives.

The remainder of this article is structured as follows: Section 2 reviews related studies in the context of software effort estimation using machine learning. Section 3 presents the dataset containing statistical information, the algorithms used, the performance metrics, and the proposed framework. Section 4 discusses the results, while Section 5 offers the conclusions.

2. Methodology

2.1. Information about the data repository

This study primarily relied on a database provided by the International Software Benchmarking Standards Group (ISBSG). The July 2022 version of the repository was used in this research. The repository includes contributions from multiple countries, with the largest being Spain (19.7%), Switzerland (19.5%), the United States (19.1%), Australia (7.6%), and Japan (7.5%). Brazil ranks twelfth, contributing 1.4% of the total data. Other contributing countries include Finland, China, France, Canada, India and Denmark. Considering all selected characteristics and after cleaning the dataset, which involved removing non-existent values and outliers, the resulting data sample consisted of 230 projects.

2.2. Dependent and independent variables predictor variables

The variable 'normalized effort,' measured in hours, was defined as the dependent (output) of the model. This variable encompasses all phases of the software development lifecycle, including planning, specification, design, construction, testing, and implementation.

The first predictor defined was 'relative size,' which corresponds to the functional size of the software, measured using the Functional Size Measurement (FSM) Method. This method includes approaches such as COSMIC, FiSMA, IFPUG 4+, IFPUG old, LOC, Mark II, and NESMA.

The second predictor defined was 'team size.' This variable represents the number of individuals who worked at any point during the project development.

The third predictor variable defined was 'development platform', which specifies the main development platform (as determined by the operating system used). Each project is classified as: PC, Mid Range (MR), Main Frame (MF), or Multiplatform (Multi). The fourth predictor variable is 'language type'. This variable defines the type of programming language used in the project, namely: 3GL, 4GL, and Application Generator (ApG).

2.3. Regression methods for effort prediction

2.3.1. Support Vector Regression

Support Vector Regression (SVR) identifies support vectors near a hyperplane to maximize the margin based on a threshold from the target value. It uses kernel functions to handle non-linear problems, with a linear kernel selected in this study (Box et al., 2015).

The forecasting is obtained through the linear regression stated as,

$$y = \sum_{i=1}^{n} \varpi K\left(x_i, x_j\right) + b \tag{1}$$

in which, y is the vector of outputs, ϖ_i is a weight, $K(x_i, x_j)$ is a kernel function, equivalent to an inner product between observations (x_i, x_j) in some feature space, b is the bias parameter, and n is the length of the time series. To determine the parameters estimation, an optimization problem is formulated as follows:

$$\min_{\omega,b,\xi,\xi^*} \frac{1}{2} \|\omega\|^2 + C\sum_{i=1}^{n}\left(\xi_i + \xi_i^*\right) \tag{2}$$

subject to: $y_i - \omega x_i - b \le \varepsilon + \xi_i^*,$ (3)

$$\omega x_i + b - y_i \leq \varepsilon + \xi_i, \tag{4}$$

$$\xi_i, \xi_i^* \geq 0, \quad i = 1, n. \tag{5}$$

where *C* is a penalty factor, is a loss function, ξ and ξ^* are two lack variables. There are several kernel functions which can be employed, and are described as,

Gaussian: $k(x_i, x_j) = \exp\left\{-\frac{\backslash x_i - x_j \backslash}{2\delta^2}\right\}$ (6)

Linear: $k(x_i, x_j) = x'_i, x_j$ (7)

Polynomial: $k(x_i, x_j) = \left(1 + x'_i, x_j\right)^p$ (8)

where δ is the width of the kernel function.

2.3.2. Gradient Boosting Machines

Gradient Boosting Machines (GBM) is an ML technique renowned for modeling complex regression and classification problems, including software effort estimation.The step by step of GBM is described as follows:

Let $D = \{x_i, y_i\}_{i=1}^{n}$ be the adopted dataset, and a loss function computed as follows: (for regression problems)

$$L[y_i, F(x)] = \frac{1}{2}[y_i - F(x)]^2 \tag{9}$$

where y_i is the vector of inputs, F(*x*) is the function related to the model used to obtain the predicted values. Initialize a model with a constant value

$$F_0(x) = \arg\min \gamma \sum_{i=1}^{n} L(y_i, \gamma) x \tag{10}$$

where the γ is the initial predicted value. In the context of regression problems, the argmin over γ means that it is need to find γ values which minimizes the loss function *L*, such as F_0 (*x*) = 0. In this context, mathematically,

$$F_0(x) = 0 \Leftrightarrow \sum_{i=1}^{n} \frac{\partial L(y_i, \gamma)}{\partial \gamma} = 0 \tag{11}$$

$$\Leftrightarrow \gamma = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{12}$$

Since m=1 to *M*,

A) Compute the pseudo-residual for i-th output value in the regression tree *m*, that is,

$$r_{i,m} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x) = F_{M-1}(x)}, \quad i = 1, \ldots, n, \tag{13}$$

B) Fit a regression tree for the $r_{i,m}$ residuals and creates a terminal region $R_{j,m}$ for all j = 1, . . ., J_M . In other words, $R_{j,m}$ represents the j-th leaves of the m-th tree;

C) For the j = 1, . . ., J_M compute the

$$\gamma_{j,m} = \arg\min_{\gamma} \sum_{x_i \in R_{i,j}}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma\right) \tag{14}$$

where in this step, for each j-th leaves, the predicted value is given by Equation 14. In fact, for each leaves, the predicted value is the average, as observed in Equation 12;

D) Update the F_m (x) value (new prediction for the output) for the m-th regression tree, that is,

$$F_m(x) = F_{m-1} + \eta \sum_{j=1}^{j_m} \gamma_{j,m} \mathrm{I}\left(\mathrm{x} \in R_j, \mathrm{m}\right), \tag{15}$$

where $F_{m-1}(x)$ are the previous prediction, η is the learning rate used to reduces the sensitivity of predictions regarding the individual outputs as well as reduces the effect of each tree on the new prediction. Finally, the summation represents the addition of new predicted values to the previous γ value.

The next step is compute the final prediction given by $\hat{y} = F_M(x)$ (Ribeiro, 2021).

2.3.3. eXtreme Gradient Boosting

eXtreme Gradient Boosting (XGBoost) was developed by Chen & Guestrin (2016), incorporating the boosting model proposed by Friedman (2001). XGBoost is a tree-based ML algorithm known for its efficiency, speed, and performance (Jabeur et al., 2024).

The predicted output is given by a sum of individual predictions and computed as following:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F, \tag{16}$$

where K is the number of trees, $\hat{y}_i$ is the i-*th* forecast output and f is a function in the space *F*. The objective function to be optimized (minimization problem) in the XGBoost approach is represented by:

$$obj = l(\hat{y}_i, y_i) + \Omega(f_k) = l(\hat{y}_i, y_i) + \left(\gamma T + \frac{1}{2}\lambda \|w\|^2\right), \tag{17}$$

in which, l is the a differentiable convex loss function that computes the difference between the forecast output $\hat{y}_i$ and real value usually represented by mean squared error. The $\Omega(f_k)$ is the regularization term, γ is a threshold for the gain, and λ is the regularization on leaf weights. Considering the predicted value in the s-th step denoted by $\hat{y}_i^{(s)}$, the objective function described in Equation 17 can be rewritten as follows:

$$obs^{(s)} = \sum_{i}^{n} \left[g_i f_k(x_i) + \frac{1}{2} h_i f_k^2(x_i) \right] + \Omega(f_k), \tag{18}$$

where taking into account the Taylor expansion of the mean squared error up to the second order,

$$g_i = \partial_{y(s-1)} l\left(y_i, \hat{y}_i^{(s-1)}\right) \text{ and } h_i = \partial^2_{y(s-1)} l\left(y_i, \hat{y}_i^{(s-1)}\right) \tag{19}$$

By reformulating the Equation 18, the objective function at step s is computed as follows:

$$obj^{(s)} = \sum_{j=1}^{T} \left[\left(\sum_{i \in i_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in i_j} h_i + \lambda \right) w_j^2 \right] + \gamma T, \tag{20}$$

where I_j is the set of indices of observations assigned to the j-th leaf of the tree. In the aforementioned equation, the first term of Equation 20 is quadratic, and the most suitable w_j for a give function q(x) is computed as follows:

$$w_j^* = \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{21}$$

Finally, the objective function can be rewritten as following:

$$obj^* = \frac{1}{2} \sum_{j=1}^{T} \left[\frac{\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} \right] + \gamma T, \tag{22}$$

where the objective function depends on g_i and h_i (Ribeiro, 2021).

2.3.4. Random Forest (RF)

Random Forest (RF) is a machine learning method that uses an ensemble of decision trees to improve prediction accuracy and reduce overfitting. Each tree is built using a random sample of the data and a random selection of variables, promoting diversity among the trees. The final prediction is made by averaging individual tree predictions for regression or using majority voting for classification. RF is popular for its robustness, accuracy, and capacity to handle large datasets and noisy features (Breiman, 2001).

Mathematically, the final prediction achieved by RF can be computed as follows, where $\hat{y}$ is the predicted output, m is the number of trees, and $f_k(x)$ represents the predictions of k-th according the input vector x (Ribeiro, 2021).

$$\hat{y} = \frac{1}{m} \sum_{k=1}^{m} f_k(x) \tag{23}$$

2.3.5. Extreme Learning Machine (ELM)

The Extreme Learning Machine (ELM) is a supervised learning method for single-hidden-layer feedforward neural networks (SLFN), introduced by Huang et al. (2006). This eliminates the need for iterative optimization, significantly reducing training time (Huang et al., 2011).

Mathematically, given a time series of n observations and m inputs, $D = \{(x_i, x_i) | x_i \in R_m, y_i \in R\}$, where x is the vector of inputs y is the vector of outputs, an ensemble model uses an aggregation function G that aggregates the predictions of K base model $f_1(x), \ldots, f_k(x)$ towards predicting a single forecasting model as follows:

$$\hat{y} = G(f1(x) f_k(x)) \tag{24}$$

where $\hat{y}_i$ is the forecasting value of i-th observation of time series in specific time window. Overall, the success of ensemble models is related to the diversity of their base models (Ribeiro, 2021)

2.3.6. Linear Regression (LR)

Linear regression is one of the simplest and most widely used statistical models for understanding the relationship between a dependent variable and one or more independent variables (Freedman, 2009).

Mathematically, the linear regression model can be expressed as:

$$y = \beta_0 + \beta_1 x1 + \beta_2 x_2 + \beta_p x_p + \varepsilon \tag{25}$$

where y is the dependent variable, $x_1, x_2, \ldots, x_p$ are the independent variables, β_0 is the intercept, $\beta_1, \ldots, \beta_p$ are the coefficients, and ε is the error term. The coefficients are estimated using the least squares criterion, which minimizes the residual sum of squares:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \quad (26)$$

2.3.7. Mean Absolute Percentage Error (MAPE)

The Mean Absolute Percentage Error (MAPE) is a widely used metric for evaluating the accuracy of forecasting and regression models. It measures the average absolute difference between predicted and actual values, expressed as a percentage of the actual values (Hyndman & Koehler, 2006). Its formula is given by Equation 27

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \frac{y_t - \hat{y}_t}{y_t} \quad (27)$$

where n is the number of observations, y_t is the actual value at time t and $\hat{y}_t$ is the predicted value at time t.

2.4. Performance measures

No single performance measure can fully evaluate the performance of an algorithm; therefore, this study utilized three evaluation parameters: RMSE, MAE, and R^2.

By definition, RMSE is the average distance of a data point from the fitted line, measured along a vertical line, and can be computed using Equation 28. (Chou et al., 2012).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (A_i - \tilde{A}_i)^2}{n}} \quad (28)$$

According to Al Betar et al. (2023), MAE is a good choice when outliers are not a major concern, as it is a less sensitive measure of accuracy. MAE is given by Equation 29.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |A_i - \tilde{A}_i| \quad (29)$$

R^2, or the coefficient of determination, is a statistical measure indicating the proportion of the variance in the dependent variable that is explained by the regression model. In simpler terms, R^2 measures how well the data fit the regression model (Myers et al., 2012). Its formula is given by Equation 30.

$$R^2 = 1 \frac{\sum_{i=1}^{n} (A_i - \tilde{A}_i)^2}{\sum_{i=1}^{n} (A_i - \tilde{A}_i)^2} \quad (30)$$

In Equations 27, 28 and 29, Ã is the actual value, A is the predicted value, and n is the number of data samples.

2.5. K-fold cross-validation

To evaluate the performance of a model more robustly and reliably, k-fold cross-validation with k = 5 was used. In this process, the dataset is initially randomly divided into approximately equal-sized subsets, or folds, each containing one-fifth of the data. Subsequently, the iteration among the folds begins, with one fold used for testing (validation) and the remaining four folds used for training the model. The model is trained on the four training folds and evaluated on the test fold, generating a performance measure for that specific iteration of cross-validation. This process is repeated five times, with each fold used once as the test set and the others as the training set.

2.6. Proposed framework

The methodology starts with acquiring data from the International ISBSG repository, followed by a data cleaning process to ensure dataset quality and consistency. Then, multiple predictive models, including SVR, ELM, GBM, RF, XGBoost, and LR, are applied, with each model undergoing a 5-fold cross-validation for robust performance evaluation.

After training and testing the models, performance metrics like RMSE, MAE, and R^2 are calculated. The observed versus predicted values are compared to assess accuracy both visually and statistically. A T-test is performed to

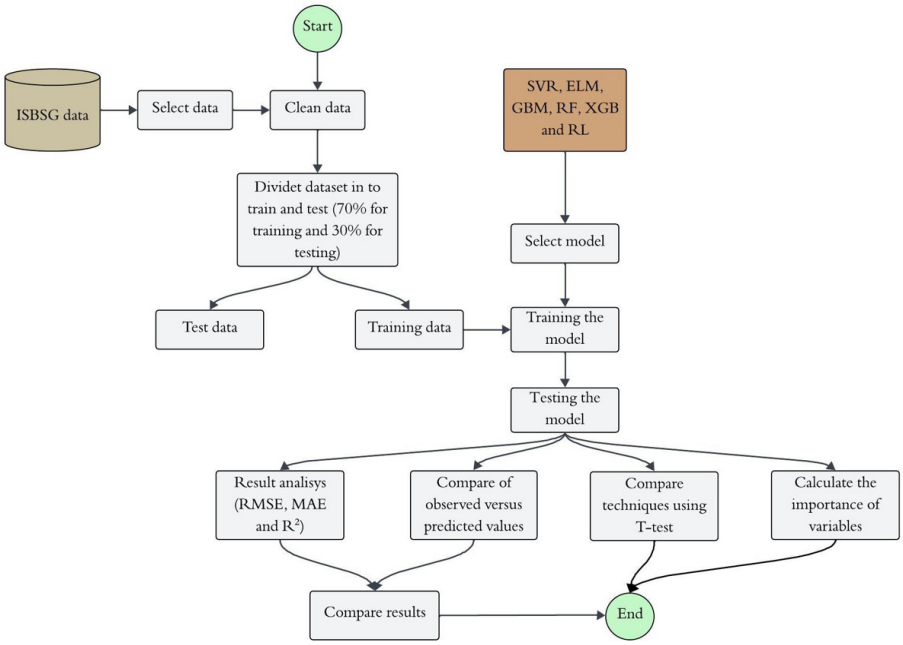

Figure 1. Proposed framework.

statistically compare the techniques and identify significant differences. Additionally, the importance of each input variable is calculated to assess its influence on predictions.

Finally, the results are compared to identify the most effective model and draw insights from the findings. This process is visually presented in Figure 1.

3. Results and discussions

In this section, the results of the SVR, ELM, RF, XGB, GBM, and LR models are presented for the performance metrics RMSE, MAE, and R^2, along with the observed versus predicted values, a comparison of the techniques using the T-test, and the importance of the variables.

3.1. Statistical analysis of the data

Table 1 presents a statistical summary for the numerical variables: effort, relative size, and team size. The effort variable, with a mean of 6525.54 and a median of 3926, shows high variability, suggesting that while most projects require modest effort, a few large-scale projects significantly raise the average. Similarly, the relative size variable, with a standard deviation of 910.66, indicates substantial dispersion, with most projects falling between the 1st quartile (170.75) and 3rd quartile (1041.25), but larger projects also exist. The team size variable has a median of 6 members,

Table 1. Descriptive statistics for numerical variables.

	Effort	Relative size	Team size
Mean	6525.5445	760.4739	8.1260
Median	3926	456	6
Standard Deviation	10410.6598	910.6642	11.4256
Variance	108855121.1953	832930.7744	131.115
Minimum	41	2	1
Maximum	109271	7599	100
Q1 (1st quartile)	1316	170.75	4
Q3 (3rd quartile)	7567.375	1041.25	6

with most projects involving small teams of 4 to 6 members. The presence of extreme values in effort and relative size highlights the need for robust modeling techniques to manage variability without being overly affected by outliers.

Figure 2 shows box-plots for effort, team size, and functional size, revealing positively skewed distributions with most data concentrated at lower values. Outliers indicate exceptional cases of high effort, large teams, and large functional sizes. The medians are near the lower limits of the boxes, suggesting that most projects have modest effort, team size, and functional size. While the overall dispersion is moderate, the extreme values highlight the need for careful consideration of outliers in the analysis.

The categorical variables used in this study are development platform and language type. Figure 3 presents the analysis of the development platform variable, showing that most projects are developed in MF environments, which

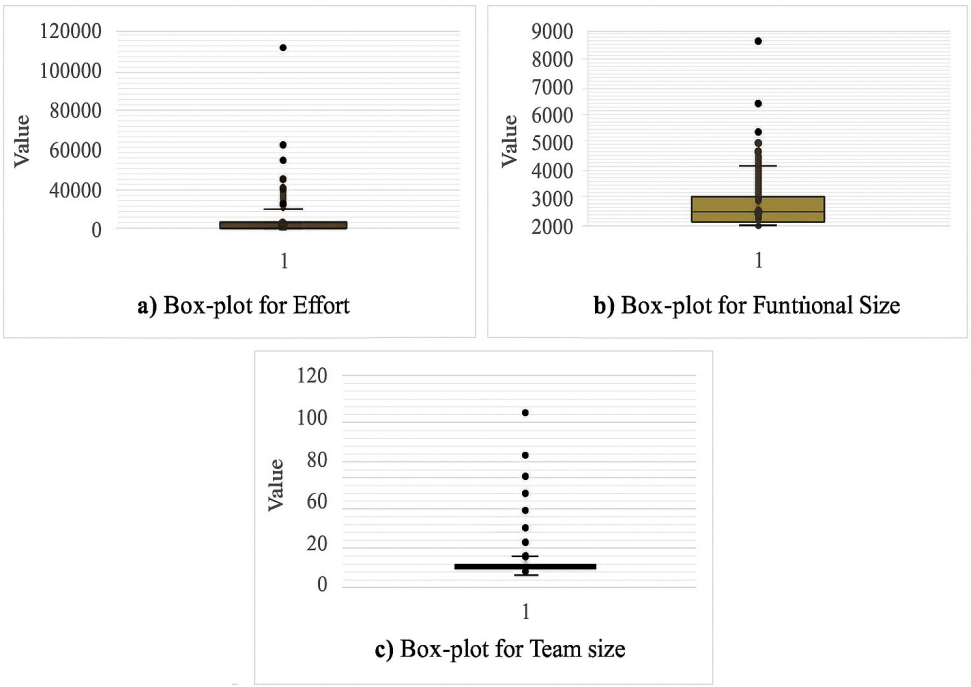

Figure 2. Graphical analysis of numerical variables.

account for 62% of the total. PC and Multi environments represent 17% and 16% of projects, respectively, while MR platforms make up only 5%. This highlights the dominance of large-scale environments in the analyzed projects.

Regarding language type, there is a balance between third-generation languages (3GL), which account for 49% of cases, and fourth-generation languages (4GL), which represent 44%. Languages categorized as ApG are present in 7% of projects. These results suggest that while traditional languages like Java and C dominate, higher-level abstraction languages like SQL also play a significant role.

In conclusion, the data reveals that most projects are concentrated in MF environments and predominantly use third-generation languages.

3.2. Predictive performance of the models using RMSE, MAE, and R^2

3.2.1. RMSE

The RMSE measures the dispersion of prediction errors and provides an idea of the magnitude of errors that the model is making in units of the dependent variable. In other words, the lower the RMSE, the better the model's performance in fitting the data.

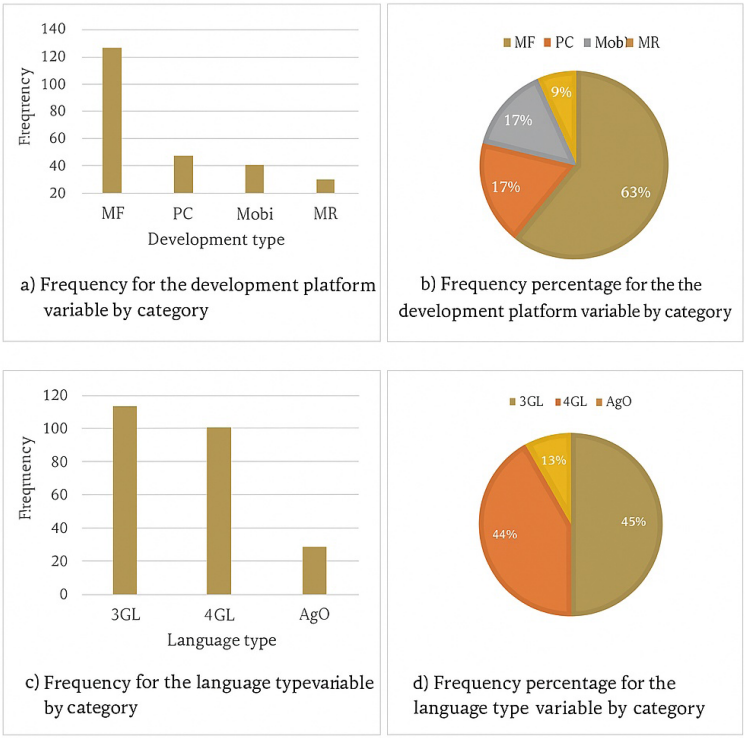

Figure 3. Frequency of qualitative variables.

The analysis of the results, presented in Tables 2 and 3 and Figure 4, showed that the performance of effort prediction models for software projects varied according to the number of variables used. With three variables, the ELM model achieved the lowest root mean square error (RMSE = 5648.579), indicating higher accuracy in this simpler scenario. However, when the number of variables increased to five, ELM was outperformed, with the GBM model delivering the best result (RMSE = 5824.245), closely followed by SVR (RMSE = 5848.167). These findings suggest that GBM has greater robustness and adaptability to increasing data complexity, while ELM performs better in lower-dimensional settings. SVR, in turn, showed sensitivity to the amount of available

Table 2. Statistical indicators for each of the models throughout the cross-validation task based on RMSE for three variables.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
SVR	2637.915	2878.900	6439.316	6228.620	7207.870	11979.10
ELM	2720.080	3185.886	5250.637	5648.579	6149.776	10936.51
RF	4482.945	4702.924	6468.849	7229.131	9326.421	11164.52
XGB	6438.246	6640.801	10960.936	9477.823	11566.585	11782.55
GBM	2753.988	3876.674	4796.566	5880.556	6529.052	11446.50
RL	2560.465	3990.975	5483.606	6088.965	5841.733	12568.04

Table 3. Statistical indicators for each of the models throughout the cross-validation task based on RMSE for five variables.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
SVR	2855.23	3639.305	4776.081	5848.167	5290.373	12679.85
ELM	3861.556	5092.513	5947.36	6449.51	6225.651	11120.47
RF	2951.21	4284.957	4518.134	6000.256	6130.342	12116.64
XGB	2970.957	6608.922	11141.68	9935.107	11504.18	17449.8
GBM	3480.321	3579.7	3680.717	5824.245	5548.991	12831.5
RL	3421.44	5728.558	6651.503	7012.444	6882.397	12378.32

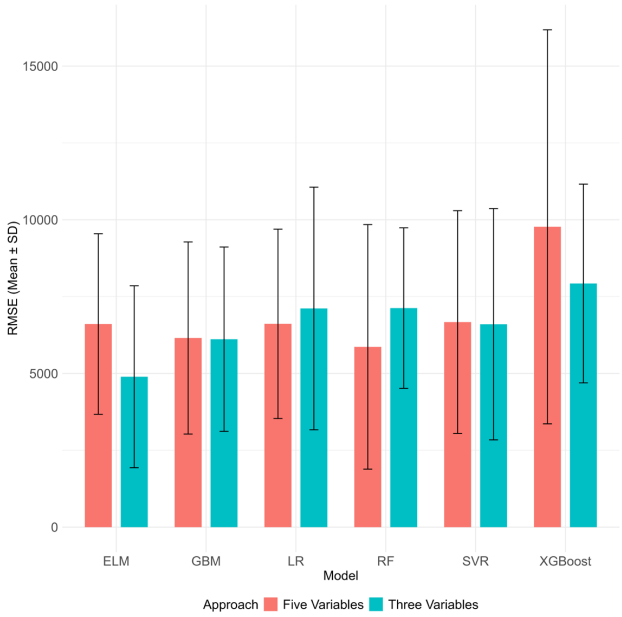

Figure 4. Performance graph of models according to RMSE for three and five variables.

information, with a significant improvement in performance as more variables were included. Therefore, selecting the most appropriate model should consider not only error metrics but also the quantity and quality of variables in the dataset.

3.2.2. MAE

Similar to RMSE, lower MAE values indicate better model fit. Based on MAE analysis, SVR achieved the best performance in both scenarios, three and five input variables, with the lowest mean absolute errors (2837.984 and 2727.108, respectively). This suggests SVR's consistency and accuracy, particularly as MAE is less sensitive to outliers than RMSE. GBM ranked second with three variables (MAE = 2901.099), while RF held that position with five. The modest reduction in MAE with more input variables indicates a slight benefit from additional information. Overall, SVR demonstrated strong generalization and low average deviation, whereas GBM showed robust performance across metrics. These findings, summarized in Tables 4 and 5 and Figure 5, highlight the complementary value of using both RMSE and MAE for a well-rounded evaluation.

Table 4. Statistical indicators for each of the models throughout the cross-validation task based on MAE for three variables.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
SVR	1708.006	1800.635	3269.473	2837.984	3560.475	3851.329
ELM	2019.973	2286.110	2827.652	2918.060	3122.904	4333.661
RF	2593.227	2594.689	2926.842	3194.971	3425.342	4434.755
XGB	3022.544	3407.728	3532.120	3764.246	4219.320	4639.516
GBM	1821.312	2331.341	2524.024	2901.099	3254.219	4574.600
RL	18840.66	2447.539	2609.389	2914.21	2945.598	4684.457

Table 5. Statistical indicators for each of the models throughout the cross-validation task based on MAE for five variables.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
SVR	1839.172	1977.642	2644.129	2727.108	2654.224	4520.372
ELM	2921.944	3897.94	3899.43	3762.313	4038.162	4054.09
RF	1934.975	2151.273	2618.03	2770.69	2995.199	4153.972
XGB	1805.435	3657.887	3967.571	4029.136	4200.891	6513.898
GBM	2212.77	2282.432	2482.883	2926.039	2987.985	4664.126
RL	2362.44	2712.319	3224.077	3199.084	3721.514	3975.069

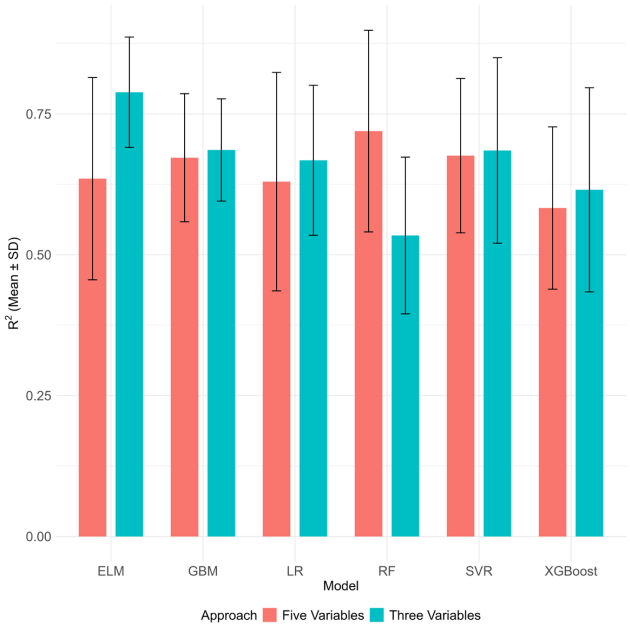

Figure 5. Performance graph of models according to MAE for three and five variables.

3.2.3. R^2

R^2 values range from 0 to 1 and indicate how well predicted values align with observed data, with values closer to 1 reflecting better model accuracy. When using three input variables, ELM achieved the highest R^2 (0.7059), explaining approximately 70.6% of the variance, followed by SVR (0.6670) and LR (0.6557). In contrast, with five variables, GBM performed best (R^2 = 0.7483), followed by RF (0.7220) and SVR (0.6422). These results suggest that ELM excels in lower-dimensional scenarios, GBM adapts well to higher-dimensional data, and SVR, while not achieving the highest R^2, shows consistent generalization and strong performance in MAE. Together, the metrics highlight the trade-offs between overall fit and error minimization. The complete results are shown in Tables 6 and 7 and Figure 6.

Table 6. Statistical indicators for each of the models throughout the cross-validation task based on R^2 for three variables.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
SVR	0.5272	0.5816	0.6375	0.6670	0.7389	0.8499
ELM	0.5626	0.5877	0.6462	0.7059	0.8381	0.8950
RF	0.3650	0.4829	0.6416	0.5978	0.7010	0.7985
XGB	0.3810	0.4214	0.5551	0.5278	0.5577	0.7237
GBM	0.4420	0.5379	0.7170	0.6410	0.7580	0.7699
RL	0.4253	0.4835	0.6205	0.6557	0.8664	0.8829

Table 7. Statistical indicators for each of the models throughout the cross-validation task based on R^2 for five variables.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
SVR	0.3085	0.5649	0.7060	0.6422	0.7857	0.8459
ELM	0.4234	0.4675	0.6813	0.6276	0.7452	0.8158
RF	0.5767	0.6712	0.7103	0.7228	0.7593	0.8962
XGB	0.3702	0.3706	0.5054	0.5294	0.6947	0.7059
GBM	0.5939	0.7024	0.7581	0.7483	0.8197	0.8383
RL	0.4090	0.4847	0.6350	0.6351	0.7293	0.8716

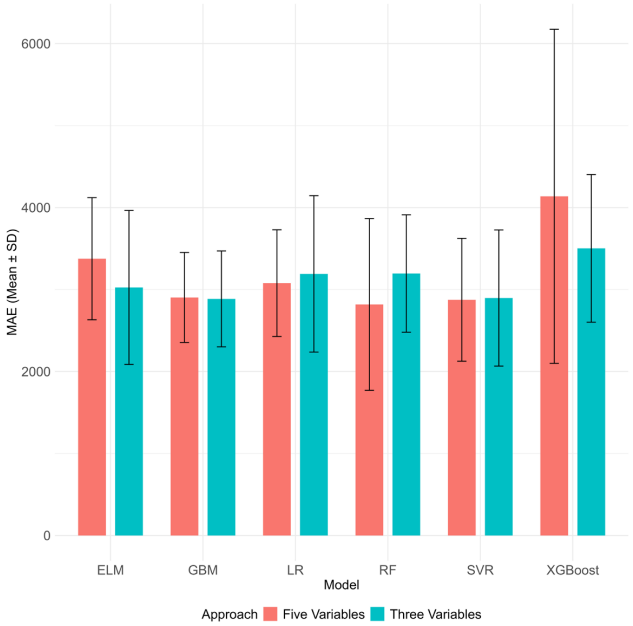

Figure 6. Performance graph of models according to R^2 for three and six variables.

3.2.4. MAPE

MAPE assesses model performance by expressing prediction errors as percentages. With three input variables, ELM recorded the highest MAPE, indicating poor accuracy, while tree-based models, GBM, RF, and XGBoost, performed significantly better, with RF and XGBoost achieving the lowest and most stable errors. With five variables, most models improved, and XGBoost slightly outperformed the others, while ELM continued to exhibit high MAPE, reinforcing its limited generalization with increased dimensionality. Overall, tree-based ensemble models (especially RF and XGBoost) proved more effective in minimizing relative error. SVR and LR showed moderate results but were consistently outperformed by ensemble approaches. These outcomes are detailed in Tables 8 and 9 and illustrated in Figure 7.

Table 8. Statistical indicators for each of the models throughout the cross-validation task based on R^2 for three variables.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
SVR	6.08	9.62	14.78	13.71	18.04	20.05
ELM	15.55	20.89	39.42	35.28	49.57	50.96
RF	4.82	6.76	8.61	9.80	9.96	18.83
XGB	6.78	7.59	7.73	9.61	9.63	16.34
GBM	8.81	9.22	9.91	10.77	11.97	13.95
RL	10.37	15.79	16.35	19.27	23.27	30.56

Table 9. Statistical indicators for each of the models throughout the cross-validation task based on R^2 for five variables.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
SVR	8.14	8.86	9.23	13.68	10.84	31.33
ELM	16.92	17.90	26.79	30.05	27.16	61.44
RF	5.31	6.61	7.65	10.01	11.40	19.06
XGB	5.61	6.04	9.78	9.16	11.70	12.65
GBM	6.64	6.64	7.17	9.96	10.24	19.09
RL	5.31	6.61	7.65	10.01	11.40	19.06

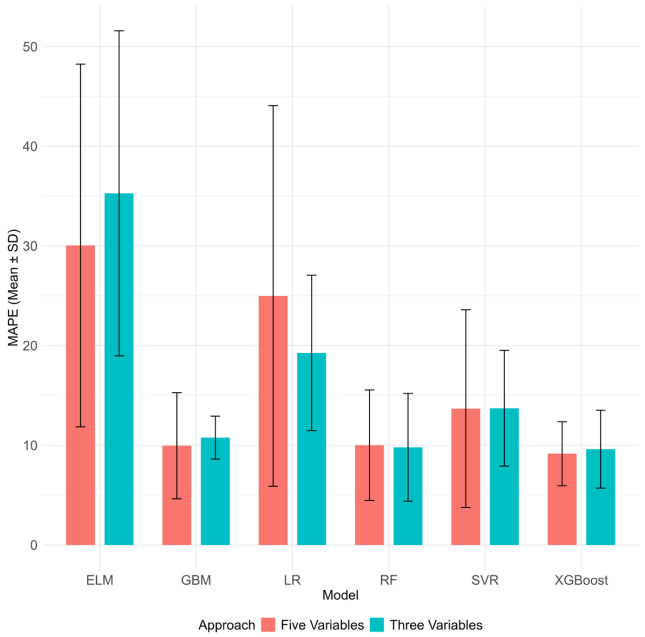

Figure 7. Performance graph of models according to MAPE for three and six variables.

The radar chart (Figure 8) summarizes model performance across RMSE, MAE, R^2, and MAPE. ELM excelled in RMSE and R^2, indicating good absolute fit and variance explanation, but performed poorly in MAPE, showing high relative error. GBM showed balanced performance, ranking well in RMSE and R^2, and moderately in MAE and MAPE, confirming its robustness. LR had weaker results across all metrics, especially in R^2, indicating limited explanatory power. SVR led in MAE and did well in MAPE, but underperformed in RMSE and R^2, suggesting good average error control but less variance capture. RF delivered strong, consistent performance, particularly in

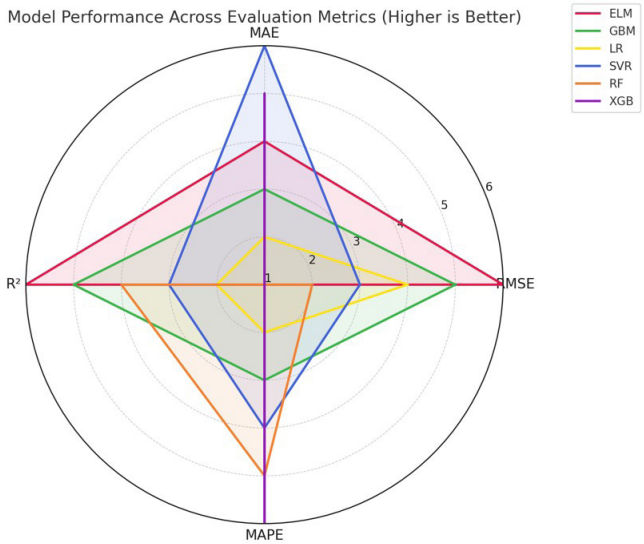

Figure 8. Radar chart indicating the position of best and worst results for the models.

MAPE, making it a reliable and well-rounded option. XGBoost topped MAPE, with strong MAE results, though it ranked lower in RMSE and R^2, highlighting its strength in relative accuracy over absolute fit.

To complement RMSE, MAE, and R^2, scatter plots of observed vs. predicted effort values were used to assess error distribution and generalization for the top models (ELM, GBM, SVR), based on five-fold cross-validation. In Figure 9, ELM showed high variability across folds, with greater dispersion, particularly for higher effort values,

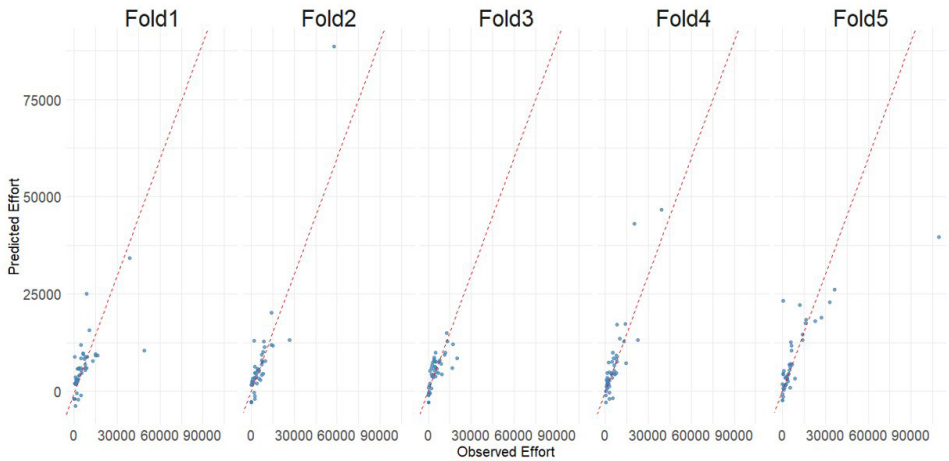

Figure 9. Observed versus predicted values for the ELM model.

indicating difficulties in capturing complex patterns. This inconsistency, including some large prediction errors, may stem from ELM's random weight initialization and sensitivity to outliers, which undermine its reliability.

Visually, the GBM model showed the most consistent performance among the three models analyzed. The predictions are well aligned with the ideal line across all folds, especially for low and medium effort values. Despite some deviations at extreme values, the model demonstrated low dispersion and good generalization ability, suggesting that it is robust and reliable for the task of effort estimation, as shown in Figure 10.

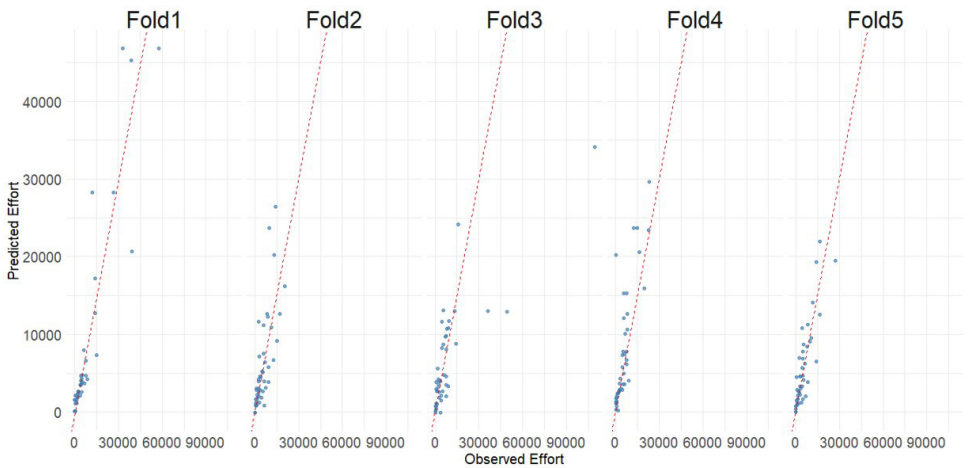

Figure 10. Observed versus predicted values for the GBM model.

The SVR model showed intermediate performance (Figure 11). Compared to the ELM, the predictions are closer to the ideal line, although with slightly more dispersion than the GBM. The model performed reasonably well across the folds and showed stable performance, with moderate errors in the more extreme cases. Overall, the SVR appears to be a viable alternative, although not as precise as the GBM.

Based solely on the visual analysis of the graphs, it can be concluded that the GBM model is the most suitable for effort estimation, followed by the SVR. The ELM, although showing potential, exhibited greater variability and imprecision in its predictions, which limits its practical applicability in this context.

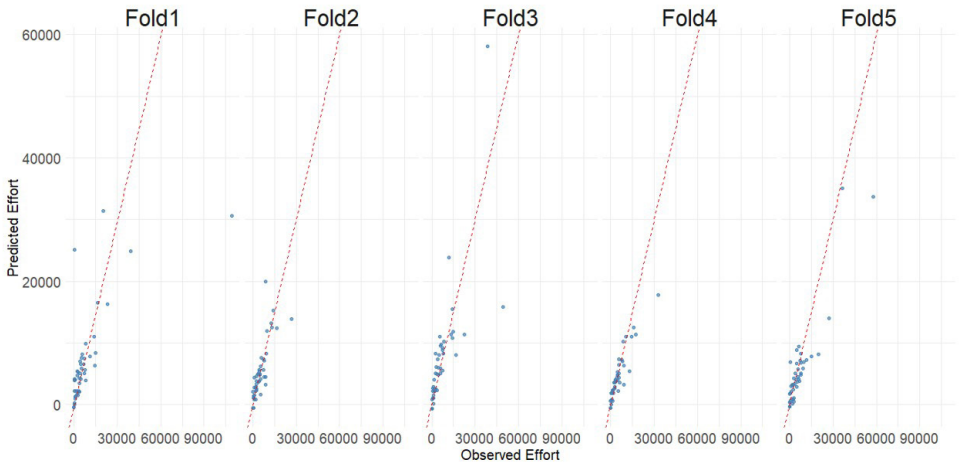

Figure 11. Observed versus predicted values for the SVR model.

Figure 12 graphically compares the confidence intervals of MAEs for different techniques using an independent samples t-test.

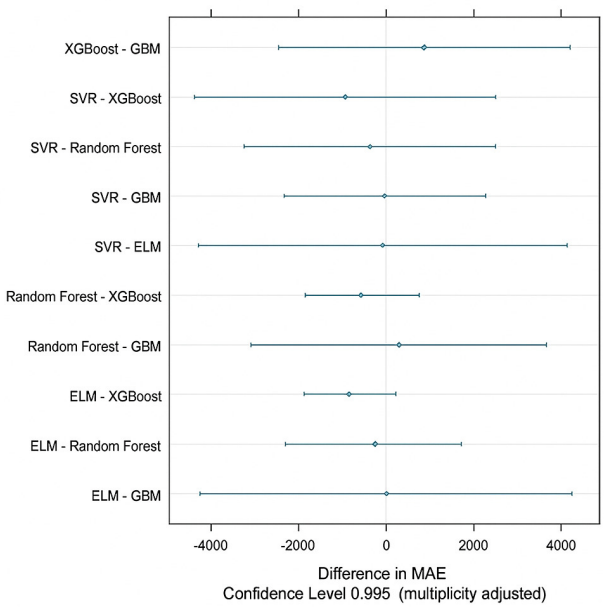

Figure 12. Comparison of techniques using confidence interval.

Through a confidence interval, an inference can be made as follows: if the CI does not contain zero, then H_0 is rejected, indicating a statistical difference between the groups. Since all intervals cross zero, it means that none are statistically more significant than the others.

The variable team size showed the highest importance score (0.4109), indicating its strong influence on model predictions, followed by functional size (0.3312), which also contributed significantly. In contrast, language type (0.0123) and development type (0.007) had minimal impact, as illustrated in Figure 13. These findings suggest that project complexity and scale, reflected in team and functional siz, are key drivers of effort estimation. The limited influence of language and development type may stem from low variability in the dataset or indirect effects. The marked difference in variable importance highlights the potential to streamline future models by excluding low-impact features, improving both efficiency and focus. Moreover, this insight can inform data collection priorities by emphasizing the most predictive attributes.

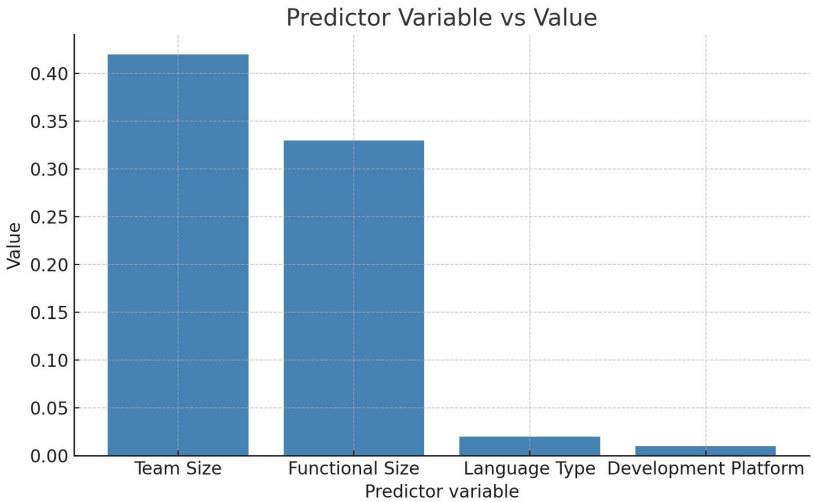

Figure 13. Importance of predictor variables.

4. Conclusion

The comparative analysis between different machine learning models applied to effort prediction in software projects showed that the performance of the algorithms varies significantly depending on the number of variables used and the evaluation metric considered. Among the models analyzed, the GBM stood out for its robustness, consistency, and ability to generalize, achieving the best results in terms of the coefficient of determination (R^2) and maintaining competitive performance in both RMSE and MAE. This highlights its suitability for handling more complex scenarios with multiple variables.

The SVR, in turn, showed the lowest MAE values with both three and five variables, indicating excellent performance in terms of mean absolute error. Its stability across different scenarios, despite having lower explanatory power (R^2), reinforces its value as an accurate and reliable model, especially when the goal is to minimize average deviations in the estimates.

On the other hand, the ELM demonstrated good performance in contexts with lower dimensionality, especially in scenarios with three variables, but exhibited greater variability and a drop in performance as complexity increased. This suggests that ELM can be useful in situations with fewer attributes and constrained execution time, although its sensitivity to outliers limits its robustness.

Other models, such as RF and LR, showed intermediate performance, while XGBoost achieved the worst results across all metrics, indicating difficulty in adapting to the analyzed dataset.

The graphical analysis of observed vs. predicted values reinforced these conclusions, with GBM showing the best adherence to the ideal prediction line, followed by SVR. Although promising, the ELM exhibited greater dispersion and inconsistency across the folds. Additionally, the statistical analysis through the T-test indicated

that the differences between the models, although evident in the metrics, were not statistically significant within the adopted confidence intervals.

Finally, the variable importance analysis highlighted that factors related to Team Size (0.4109) and Functional Size (0.3312) were the most influential predictors in the model, whereas Language Type (0.0123) and Development Type (0.007) had minimal impact. This indicates that team-related and functional aspects are key determinants of model performance. The findings suggest that less relevant variables may be excluded in future analyses to enhance model efficiency and guide data collection priorities.

Therefore, it is concluded that the choice of the ideal model should consider not only performance metrics but also the number and nature of the available variables. GBM stands out as the best overall alternative, especially in more complex contexts, while SVR offers an effective solution for minimizing mean error, and ELM can be advantageous in scenarios with fewer variables and computational constraints.

Although the results obtained are promising, this study has some limitations that should be considered. The main limitation relates to the quantity and variety of data available, which may have restricted the models' ability to capture more complex nuances of development effort. The inclusion of additional predictive variables, such as team experience, software quality, number of reported defects, user satisfaction level, and especially the total project cost, could enhance the robustness and explanatory power of the models. Therefore, we suggest that future studies explore the incorporation of these attributes, if available, to enrich the analysis.

Another limitation is related to the difficulty of making direct comparisons with other studies in the literature, as many use different data subsets or specific approaches for distinct sectors. To mitigate this issue, we recommend that future research test the proposed model using databases with similar characteristics, which could enable more consistent comparisons between techniques and improve the generalization of the findings presented here.

Acknowledgements

The author wishes to thank the Federal University of Technology – Paraná, Pato Branco Campus, for its support throughout this research, as well as the professors whose valuable guidance and contributions were essential to the progress of this study.

Data availability

Research data is only available upon request.

References

Al-Betar, M. A., Kassaymeh, S., Makhadmeh, S. N., Fraihat, S., & Abdullah, S. (2023). Feedforward neural network-based augmented salp swarm optimizer for accurate software development cost forecasting. *Applied Soft Computing*, *149*, 111008. http://doi.org/10.1016/j.asoc.2023.111008.

Bontempi, G., Ben Taieb, S., & Le Borgne, Y. A. (2013). Machine learning strategies for time series forecasting. In: M.-A. Aufaure, E. Zimányi (Eds.), *Business intelligence* (pp. 62-77). Berlin: Springer. http://doi.org/10.1007/978-3-642-36318-4_3

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. Hoboken: John Wiley & Sons.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5-32. http://doi.org/10.1023/A:1010933404324.

Chen, T., & Guestrin, C. (2016). Xgboost: a scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794). New York: Association for Computing Machinery. http://doi.org/10.1145/2939672.2939785.

Chou, J. S., Cheng, M. Y., Wu, Y. W., & Wu, C. C. (2012). Forecasting enterprise resource planning software effort using evolutionary support vector machine inference model. *International Journal of Project Management*, *30*(8), 967-977. http://doi.org/10.1016/j.ijproman.2012.02.003.

Freedman, D. A. (2009). *Statistical models: theory and practice*. Cambridge: Cambridge University Press. http://doi.org/10.1017/CBO9780511815867.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, *29*(5), 1189-1232. http://doi.org/10.1214/aos/1013203451.

Huang, G.-B., Wang, D. H., & Lan, Y. (2011). Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, *2*(2), 107-122. http://doi.org/10.1007/s13042-011-0019-y.

Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, *70*(1–3), 489-501. http://doi.org/10.1016/j.neucom.2005.12.126.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*(4), 679-688. http://doi.org/10.1016/j.ijforecast.2006.03.001.

Jabeur, S. B., Mefteh-Wali, S., & Viviani, J. L. (2024). Forecasting gold price with the XGBoost algorithm and SHAP interaction values. *Annals of Operations Research*, *334*(1), 679-699. http://doi.org/10.1007/s10479-021-04187-w.

Jiang, J., Chen, Z., Wang, Y., Peng, T., Zhu, S., & Shi, L. (2019). Parameter estimation for PMSM based on a back propagation neural network optimized by chaotic artificial fish swarm algorithm. *International Journal of Computers, Communications & Control*, *14*(6), 615-632. http://doi.org/10.15837/ijccc.2019.6.3705.

Kassaymeh, S., Alweshah, M., Al-Betar, M. A., Hammouri, A. I., & Al-Ma'aitah, M. A. (2024). Software effort estimation modeling and fully connected artificial neural network optimization using soft computing techniques. *Cluster Computing*, *27*(1), 737-760. http://doi.org/10.1007/s10586-023-03979-y.

Kaushik, A., Tayal, D. K., & Yadav, K. (2020). A comparative analysis on effort estimation for agile and non-agile software projects using DBN-ALO. *Arabian Journal for Science and Engineering*, *45*(4), 2605-2618. http://doi.org/10.1007/s13369-019-04250-6.

Kumar, P. S., Behera, H. S., Kumari, A., Nayak, J., & Naik, B. (2020). Advancement from neural networks to deep learning in software effort estimation: perspective of two decades. *Computer Science Review*, *38*, 100288. http://doi.org/10.1016/j.cosrev.2020.100288.

Lavingia, K., Patel, R., Patel, V., & Lavingia, A. (2024). Software effort estimation using machine learning algorithms. *Scalable Computing: Practice and Experience*, *25*(2), 1276-1285. http://doi.org/10.12694/scpe.v25i2.2213.

López-Martín, C. (2022). Machine learning techniques for software testing effort prediction. *Software Quality Journal*, *30*(1), 65-100. http://doi.org/10.1007/s11219-020-09545-8.

Myers, R. H., Montgomery, D. C., Vining, G. G., & Robinson, T. J. (2012). *Generalized linear models: with applications in engineering and the sciences.* Hoboken: John Wiley & Sons.

Rahman, M., Roy, P. P., Ali, M., Gonçalves, T., & Sarwar, H. (2023). Software effort estimation using machine learning technique. *International Journal of Advanced Computer Science and Applications*, *14*(4), 822-827. http://doi.org/10.14569/IJACSA.2023.0140491.

Rankovic, N., Rankovic, D., Ivanovic, M., & Lazic, L. (2021). A new approach to software effort estimation using different artificial neural network architectures and Taguchi orthogonal arrays. *IEEE Access: Practical Innovations, Open Solutions*, *9*, 26926-26936. http://doi.org/10.1109/ACCESS.2021.3057807.

Rao, K. E., Terlapu, P. R. V., Naidu, P. A., Kumar, T. R., & Pydi, B. M. (2024). Feature importance for software development effort estimation using multi level ensemble approaches. *Bulletin of Electrical Engineering and Informatics*, *13*(2), 1090-1102. http://doi.org/10.11591/eei.v13i2.5531.

Ribeiro, M. H. D. M. (2021). *Time series forecasting based on ensemble learning methods applied to agribusiness, epidemiology, energy demand, and renewable energy* (Doctoral dissertation). Pontifícia Universidade Católica do Paraná, Curitiba.

Sharma, S., & Vijayvargiya, S. (2020). Applying soft computing techniques for software project effort estimation modelling. In *Nanoelectronics, Circuits and Communication Systems: Proceeding of NCCS 2019* (pp. 211-227). Singapore: Springer.

Van Hai, V., Nhung, H. L. T. K., Prokopova, Z., Silhavy, R., & Silhavy, P. (2022a). Toward improving the efficiency of software development effort estimation via clustering analysis. *IEEE Access: Practical Innovations, Open Solutions*, *10*, 83249-83264. http://doi.org/10.1109/ACCESS.2022.3185393.

Varshini, A. G. P., & Kumari, K. A. (2024). Software effort estimation using stacked ensemble technique and hybrid principal component regression and multivariate adaptive regression splines. *Wireless Personal Communications*, *134*(4), 2259-2278. http://doi.org/10.1007/s11277-024-11010-9.

Author contributions

DL: Conceptualization, Data Curation, Methodology, Writing – Original Draft And Writing – Review & Editing.
MHR: Conceptualization, Data Curation, Formal Analysis, Writing – Review & Editing And Supervision.
FT: Formal Analysis, Writing – Review & Editing And Supervision.