mathematics

Article

Software Estimation in the Design Stage with Statistical Models and Machine Learning: An Empirical Study

Ángel J. Sánchez-García *, María Saarayim González-Hernández, Karen Cortés-Verdín and Juan Carlos Pérez-Arriaga

Facultad de Estadística e Informática, Universidad Veracruzana, Xalapa 91020, Veracruz, Mexico; saara.glex@gmail.com (M.S.G.-H.); kcortes@uv.mx (K.C.-V.); juaperez@uv.mx (J.C.P.-A.)
* Correspondence: angesanchez@uv.mx

Abstract: Accurate estimation of software effort and time in the software development process is a key activity to achieve the necessary product quality. However, underestimation or overestimation of effort has become a key challenge for software development. One of the main problems is the estimation with metrics from late stages, because the product must already be finished to make estimates. In this paper, the use of statistical models and machine learning approaches for software estimation are used in early stages such as software design, and a data set is presented with metric values of design artifacts with 37 software projects. As results, models for the estimation of development time and effort are proposed and validated through leave-one-out cross-validation. Further, machine learning techniques were employed in order to compare software projects estimations. Through the statistical tests, it was proven that the errors were not statistically different with the regression models for effort estimation. However, with Random Forest the best statistical results were obtained for estimating development time.

Keywords: software estimation; software design; regression; effort; software development time

MSC: 68T09; 68U07; 62P30; 62J05

check for updates

Citation: Sánchez-García, Á.J.; González-Hernández, M.S.; Cortés-Verdín, K.; Pérez-Arriaga, J.C. Software Estimation in the Design Stage with Statistical Models and Machine Learning: An Empirical Study. *Mathematics* **2024**, *12*, 1058. https://doi.org/10.3390/math12071058

Academic Editor: Florin Leon

Received: 22 February 2024
Revised: 20 March 2024
Accepted: 27 March 2024
Published: 1 April 2024

1. Introduction

One of the objectives of Software Engineering is to achieve high quality of the product and process. Quality is the "degree to which a set of inherent characteristics of an object meets the requirements" [1], and to achieve this, it is necessary to adhere to standards and follow methodologies for good resources management.

One of the aspects to keep in mind in the quality of a project is to make precise estimates of software attributes. If the estimations are not made, or if they are incorrectly calculated, the requirements of the project may not be met, and therefore, the required quality may not be achieved.

When a software project is planned, software effort prediction is performed to assign suitable resources in order to deliver a quality product in accordance with the established time and requirements. There are studies that, through systematic literature reviews, demonstrate that the most widely used variable regarding software development life cycle effort (SDLC) is person-hours [2–8]. Software project size is often used in the SDLC effort prediction as an explanatory variable [6]. Therefore, we selected person-hours to represent the effort, and size and complexity (little reported in literature but extremely important) of the software project as the explanatory factor for software effort prediction (SEP). In the SEP field, the models commonly used have been based on statistical regression equations and machine learning models [6].

Although the precision in software effort estimation has been become a challenge, the main problem that is addressed lies in the fact that not all the estimates are precise, which leads to a low reliability of these models. In addition, these estimates are usually based on late-stage metrics, such as the construction stage, which implies having a practically finished product. Therefore, estimates in early stages such as the design phase, could help to manage a project in a better way.

The No Free Lunch (NFL) theorem [9] states that there is no machine learning algorithm that works best on all problems. In other words, this theorem highlights the importance of choosing and adapting approaches to specific problems instead of looking for one that solves all problems. Thus, we chose to explore both statistical regression models and different machine learning approaches (such as decision trees and ensemble algorithms) in our proposal to estimate effort with software design metrics.

To make estimates, statistical regression models are often used. Regression analysis is a "statistical technique to determine the relationship between two or more variables" [10]. However, there are several approaches to building these models, so a comparison of statistical models against machine learning algorithms with data of the software design stage is the main objective of this research.

On the other hand, the characteristics of the public projects with which estimates are made are different, and surely do not conform to the characteristics of all types of projects; for example, for engineers who are beginning to develop software in small projects with low complexity. The literature reports estimations based on design metrics to make software estimates. These metrics can be defined in "terms of the probability that a design will meet its specifications" [11]. In 1990, Kitchenham [12] determined that design metrics can be used to rework a design and avoid anticipated problems, and they are also useful in giving an early judgment on the success of the design process.

Therefore, we summarize our contribution in the following points.

1. We contributed to software development practitioners in providing models that allow them to estimate development effort.
2. We generated a data set based on software design artifact metrics, since it is an early stage of development (unlike the main data sets in the literature).
3. Our data set projects are categorized by complexity and size.
4. We made an empirical comparison of the prediction results between statistical models and the most common machine learning approaches.
5. We used a leave-one-out cross-validation mechanism to validate the models obtained.
6. The results were compared statistically to validate significant differences between the results.

This paper is structured as follows: in Section 2 related work is explored; in Section 3 our proposal to generate the prediction models is shown. Section 4 is dedicated to the data acquisition process and Section 5 shows the experiments and results. Section 6 establishes the validity threats addressed in this work. Finally, Section 7 covers the conclusions and future work.

2. Related Work

In 2013, in the work carried out by Pomorova and Horushchenko [13], the need to evaluate the quality of software using Artificial Neural Networks is described. In this research, the authors used metrics from the design stage to carry out a comparative analysis of four different software projects, where the result was that the software projects showed similarity in cost and development time; however, they obtained different estimates in complexity and quality of the project.

In [14], the authors explain that the low success rate in the conclusion of software projects is a consequence of an overestimation or underestimation of software attributes, which leads to failures. On the other hand, in [15] the authors state that "the most challenging task in software effort estimation is managing complexity in software development". According to this research, an estimation of software attributes, cost and effort especially, influence product quality. In addition, the authors focused on creating a software effort estimation prediction model using the Desharnais data set.

In [16], the authors used three different prediction algorithms (Linear Regression, Support Vector Machine, and Multi-Layer Perceptron), to compare them and determine the one with the best performance for estimating software effort. In this research, the SEERA data set was used, and the values of Mean Square Error (MSE), Mean Absolute Error (MAE), and R-Squared evaluation metrics were analyzed. It was determined that the Linear Regression model calculated the best effort prediction.

In [17], the authors applied the Long Short-Term Memory (LSTM) algorithm to discover its accuracy in software effort estimation, with the purpose of comparing it to models that they previously had. The authors used two data sets: China, which contains 499 projects; and the Kitchenham data set, with 145 projects. In addition, to measure the accuracy of the models, they used the values of Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-Squared.

Table 1 gives a summary of works where some data sets have been used; however, none of them establish design metrics that allow for estimating the development effort for a project based on early-stage measurements.

Table 1. Data from studies related to software effort prediction models.

Data Source	Size	Study
Albrecht	24	[18–22]
NASA 60	60	[23,24]
NASA 93	93	[25,26]
Desharnais	77	[19,21,27,28]
Kemerer	15	[22,29–31]
Maxwell	62	[21,25,26,31]
COCOMO 81	63	[21,27,32,33]
China	499	[21,26,34]

Nevertheless, in the previously mentioned works, an estimation of software effort and time using design metrics is not observed, which is very important due to making estimates based on early stages of software development metrics and not with metrics of a finished product. Therefore, this research paper presents the comparison of the efficiency of software development time and effort estimates, using design metrics and regression models, such as Simple Linear Regression, LARS Regression, LASSO Regression, and machine learning techniques such as Decision Tree and Random Forest.

3. Proposal

The innovation of this work falls on two important aspects of software effort estimation. The first one is that it is commonly estimated by calculating metric values (or variables in a model) after the implementation stage. In this work, we propose to consider metrics from early stages such as design, since the characteristics of the system have already been identified. Regarding the second aspect, we propose effort estimates with statistical models, unlike classic machine learning algorithms. Statistical models and machine learning algorithms (such as those mentioned in the previous section) have different advantages and disadvantages, and the choice between them will depend on the specific problem and the available data. We chose to experiment with statistical prediction models (little investigated in the area of software estimation) due to the following four reasons:

1. Statistical models are often more interpretable than machine learning algorithms. This means that it is easier to understand how the variables are related and how the predictions are arrived at.
2. For smaller-scale problems, statistical models can be more powerful than machine learning algorithms. Statistical models typically require fewer data and computational resources than machine learning algorithms.
3. Statistical models are good for generalizing results to new populations, since they are designed to capture the essence of the problem and not just fit the training data.

4. Statistical models allow for more control of model complexity, which can be useful for avoiding overfitting in problems with a limited number of observations.

The aspects taken for each of the proposed innovations are specified below. These two elements are the independent variables (taken from metrics applied to software design artifacts) and the statistical models to be compared.

3.1. Design Metrics

In [35], the design metrics found after performing a Systematic Literature Review (SLR) are presented. The authors mention that the metrics mainly measure the classes modeled in Unified Modeling Language (UML), with their respective attributes and methods, plus other aspects such as encapsulation, inheritance, polymorphism, cohesion, or coupling between classes were measured. The selected metrics used in this work are those identified in [35], which are a selection of the metrics of Chidamber and Kemerer, Lorenz and Kidd, Abreu and Melo, and unknown authors, as shown in Table 2.

Table 2. Design metrics by author.

Author(s)	Design Metric
Chidamber and Kemerer	WMC: weighted method per class
Chidamber and Kemerer	CBO: coupling between objects
Chidamber and Kemerer	DIT: depth of inheritance tree
Chidamber and Kemerer	NOC: number of children
Lorenz and Kidd	PM: number of public methods
Lorenz and Kidd	NM: number of methods
Lorenz and Kidd	NPV: number of public variables per class
Lorenz and Kidd	NV: number of variables per class
Lorenz and Kidd	NMI: number of methods inherited by a subclass
Lorenz and Kidd	NMO: number of methods overridden by a subclass
Lorenz and Kidd	NMA: number of methods added by a subclass
Abreu and Melo	PF: polymorphism factor
Abreu and Melo	CF: coupling factor
Abreu and Melo	MHF: method hiding factor
Abreu and Melo	AHF: attribute hiding factor
Abreu and Melo	MIF: method inheritance factor
Abreu and Melo	AIF: attribute inheritance factor
Unknown authors	ExCoupling: total number of incoming methods calls to an object
Unknown authors	ImpCoupling: total number of outgoing method invocations of an object

3.2. Statistical Regression Models

In this research paper, three different regressions were used to estimate software effort and software development time. These three models were selected based on the results of [36], where various statistical models in Software Engineering were analyzed. Polynomial regression was not selected because it only uses one independent variable and for this work a multivariate analysis is needed. The regressions used are described below.

Simple Linear Regression: this regression is an extension of the Simple Linear Regression. It has multiple predictor variables and one outcome [37]. This regression shows a relationship between a dependent variable and p number of independent variables [38]. This regression is defined as (1).

$$Y = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p + \epsilon \tag{1}$$

LASSO Regression: this regression was proposed in 1996 by Robert Tibshirani. This statistical method "minimizes the residual sum of frames subject to the absolute value sum of the coefficients being less than a constant" [39]. LASSO regression aims to "identify the variables and the corresponding regression coefficients that conform to a model that minimizes the prediction error" [40].

Lasso regression has the capability to automatically select the most important variables for the regression model and assign a value of zero to the coefficients for less important variables. This means that you can handle large data sets with many explanatory variables and avoid overfitting. Furthermore, this regression uses a regularization technique that helps to avoid overfitting and improve the generalizability of the model. This technique penalizes large coefficients and reduces them to zero.

Least Angle Regression (LARS): this regression is a model selection algorithm, and it is relatively new. Defined as "a useful and less greedy version of traditional direct selection methods" [41], it is a kind of democratic version of stepwise regression, and it is closely connected to LASSO regression, providing an algorithm to compute the entire LASSO path [42].

The LARS regression formula is similar to that of Linear Regression. The differences lie in the method used to calculate the regression coefficients and the technique used to the variable selection. In LARS regression, the coefficients are fitted iteratively as explanatory variables are added, rather than fitting simultaneously for all variables as in Linear Regression. LARS regression is a robust method that can handle data containing outliers or errors in the explanatory variables.

4. Data Adquisition Process

A total of 37 software development projects belonging to students from the B.Sc. program in Software Engineering were collected. The projects' authors were interviewed in order to obtain the effort value (man-hours) for each project. Some of the interviewed authors had their PSP (Personal Software Process) [43] record. However, most of them did not have it, thus an approximation of effort was used. Because of this, it must be admitted that such an estimation affects the results.

Figure 1 shows the process that was followed to obtain the final data set after data preprocessing. Each stage of the process is detailed in the following subsections.

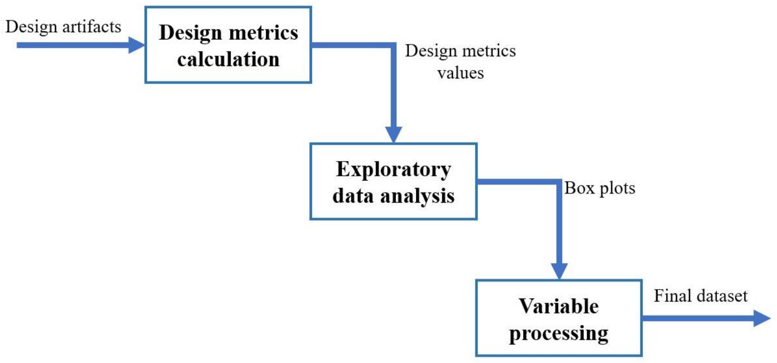

Figure 1. Data processing stages.

4.1. Project Characteristics

When building prediction models, it is necessary to explain the characteristics of the projects used, since the more similar the projects used to make estimates, the more accurate the estimates will be. These characteristics were divided into project size and project complexity.

4.1.1. Project Size

To calculate the size of each project, Use Case Points (UCPs) were used. UCP is a method developed in 1993 by Gustav Karner to estimate software [44]. This method "analyzes the use case actors, scenarios, and various technical and environmental factors and abstracts them into an equation" [45]. The UCP equation is shown in (2), where UUPC is the Unadjusted Use Case Points, TCF refers to the Technical Complexity Factor and it is calculated by (3), and the Environment Complexity Factor is denoted by ECF and it is computed by (4).

$$UCP = TCF * ECF * UUCP \tag{2}$$

$$TCF = 0.6 + 0.01 \sum_{i=1}^{n} w_i * F_i \tag{3}$$

$$ECF = 1.4 + (-0.03) \sum_{i=1}^{n} w_i * F_i \tag{4}$$

TCF is determined by assigning a subjective score between 0 (the factor is irrelevant) and 5 (the factor is essential) to each of the 14 technical factors described in [46]. The final value of the TCF is calculated by dividing the total sum obtained in the previous step by 100 and adding a constant equal to 0.6.

ECF is calculated in order to have environmental considerations of the system. Similarly, ECF is determined by assigning a score between 0 (no experience) and 5 (expert) to each of the 8 environmental factors whose are presented in [46]. This information should be collected from the development team that participated in the project. The final value of the ECF is calculated multiplying the sum of the multiplications of the factors and their weights, by a constant of −0.03 and adding a constant equal to 1.4.

According to [45], the UUPC is composed of computations: Unadjusted Use Case Weight (UUCW) and the Unadjusted Actor Weight (UAW) as (5).

$$UUCP = UUCW + UAW \tag{5}$$

To calculate the value of UUCW, use cases are divided into three categories: simple, average, and complex, according to the number of transactions for each use case identified, which are assigned a factor of 5, 10, and 15, respectively. So, UUCW can be defined as (6).

$$UUCW = \sum_{i=1}^{n} \text{Total type of use case * factor} \tag{6}$$

UAW is calculated by adding the number of actors in each category and multiplying each total by its specified weighting factor. Actors are divided into three categories: Simple, where the actor represents another system with a defined application programming interface (factor = 1); Medium, where the actor represents another system that interacts through a protocol (factor = 2); and Complex, where the actor is a person who interacts through a graphical user interface (factor = 3). Therefore, the UAW equation is represented as (7).

$$UAW = \sum_{i=1}^{n} \text{Total type of actor * factor} \tag{7}$$

After calculating the complexity of each project, none of the projects exceeded a UUCW value of 300, so it can be concluded that the complexity of the projects varies between simple and medium. Once the UCP values for each project have been calculated, they were analyzed, and their values had a minimum value of 73.92 and a maximum value of 183.551. Therefore, the estimates made in this study will be more precise for projects that have a similar complexity.

4.1.2. Project Complexity

As shown in Table 2, the Coupling Factor (CF) metric measures the coupling of a system. According to the authors of the metric, the coupling between classes can have a negative impact on the system, since as the coupling between classes increases, the density of defects and normalized rework are also expected to increase [47].

Given the above, it is recommended that the Coupling Factor value not be too high, to avoid increasing complexity, and reducing encapsulation and reuse [47]. The CF values for the projects oscillate between 0 and 1. Of the total projects, only 3 have a CF above 0.5, so these projects are more complex. The remaining set of projects has a low CF, so it is concluded that the majority of the data set is made up of projects whose complexity is low, which is because they are expected to have greater encapsulation and reuse.

4.2. Design Metrics Calculation

The selected metrics were calculated for each one of the 37 projects. The values obtained can be consulted in [48]. It was observed that from Chidamber and Kemerer metrics, and the Abreu and Melo metrics, several values are obtained for each project, and therefore it was decided to work with a representative value. The chosen value was the median, since that the median is not as sensitive to outliers as the mean is.

The Lorenz and Kidd metrics values are discrete for each project. On the other hand, the metrics of unknown authors could not be used with the 37 projects, because the needed artifacts to calculate them were not available.

4.3. Exploratory Data Analysis

Once the value of the metrics was calculated, an exploratory analysis of the data was carried out. Table 3 show descriptive statistics of the data set with 37 and it is available in [48].

Table 3. Descriptive statistics of the 37 project data set.

Variable	Type	Missing Values	Min	Max	Mean	Median	SD
WMC	numeric	11	0	17.5	6.521	5	5.544
CBO	numeric	1	1	1	1.549	2	0.404
DIT	numeric	22	1	1	1	1	0
NOC	numeric	20	0	4	1.735	2	0.868
PM	numeric	15	0	17.5	6.818	6.5	5.622
NM	numeric	15	0	17.5	6.818	6.5	5.622
NPV	numeric	0	0	8	0.722	0	2.132
NV	numeric	0	0.5	8	3.902	3.75	1.831
NMI	numeric	22	0	16	2.866	0	5.692
NMO	numeric	22	0	14	1	0	3.605
NMA	numeric	22	0	8	1.633	0.5	2.341
PF	numeric	15	0	0	0.168	0	0.496
CF	numeric	0	0	1	0.210	0.136	0.280
MHF	numeric	25	0.003	0.024	0.008	0.00765	0.005
AHF	numeric	4	0	1	0.481	0.161	0.454
MIF	numeric	26	0	0.2576	0.085	0	0.110
AIF	numeric	25	0	0.2871	0.080	0	0.112
ExCoup	numeric	26	0.05	0.72	0.343	0.31	0.221
ImpCoup	numeric	27	0.05	0.67	0.342	0.33	0.221
EFFORT	numeric	0	18.333	864.333	200.7783874	147.25	183.315
TIME	numeric	0	38.5	2593	533.408	375	578.798

Next, box plots were obtained. Box plots are used to understand the way in which the data of the independent variables behave, being defined as “a plot used when you want to explain how a series of quantitative data is distributed” [49]. Therefore, this kind of plot lets us see the variability of the values obtained for each calculated metric, as well as the way in which each metric behaved.

As can be seen in Figure 2, where the values of the WMC metric are plotted, the boxplot indicates that WMC is a metric that has great variability, because the size of the box is large and covers almost the entire range. Furthermore, the orange line indicates the average of the values obtained in the WMC metric, where it can be observed that although the range of values is large, on average the projects have a WMC value of four. So, it is concluded that the class diagrams have a varied number of methods, having at least a value of zero methods and at most a value of about 17.

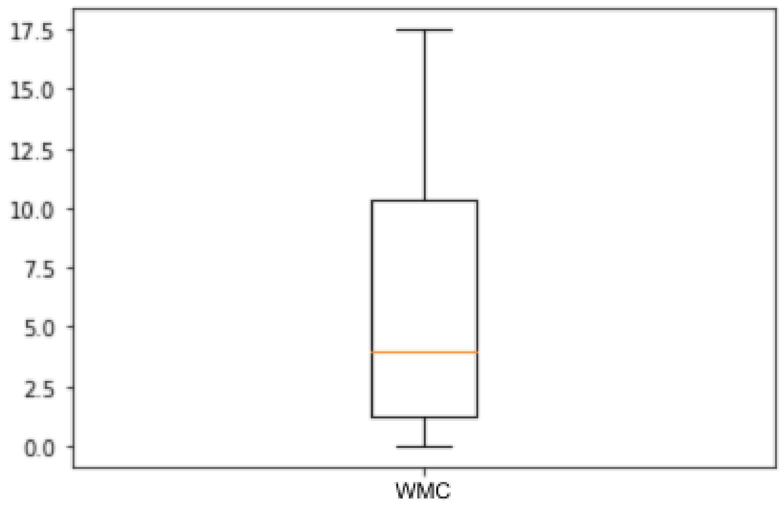

Figure 2. WMC variable boxplots.

However, not all variables behaved in the same way; the boxplots also showed variables with not great variability in the data. For example, in Figure 3, the boxplots of the PF, CF, and MHF variables (in this order) are shown, where it is observed that the PF variable has some outliers (indicated by the circles), but its mean of values is at 0. So, it is concluded that PF does not provide much information to the prediction model, in addition to disclosing that there are almost no projects that involve polymorphism. The CF variable has greater variability, while the MHF metric does not have variability, its mean being zero and having only some outliers (circles), which indicates that there is not a strong relationship between hidden methods; like the PF metric, it does not provide much information to the models.

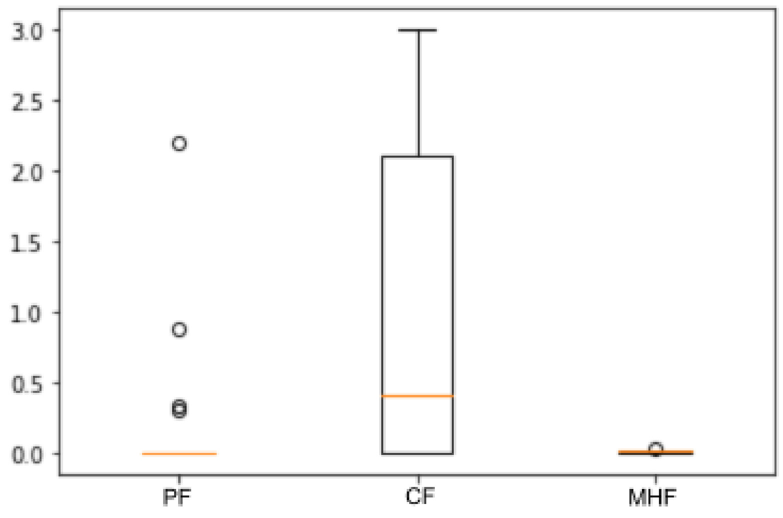

Figure 3. PF, CF, and MHF variable boxplot.

4.4. Variable Processing

After obtaining the values of the explanatory variables and performing an exploratory analysis of the data, it was observed that the metrics could not be calculated for all the projects, therefore, the projects had a null value. In addition, there were projects that did not have enough artifacts in order to calculate the variables. For this reason, considering the behavior of some variables, metrics, and projects began to be eliminated, in order to avoid empty or null spaces in the data set.

First, those metrics with little variability (those variables that only had a maximum of two different values) or null were discarded, since they were variables that did not provide information to the models. These variables were those corresponding to the DIT, NPV, NMI, NMO, NMA, PF, MHF, and NOC metrics.

Subsequently, those metrics that could not be applied in the projects, or that in most of them were not possible to calculate, were eliminated. These dropped metrics were MIF, AIF, ExCoupling, and ImpCoupling. In addition, those projects for which more than 5 metrics could not be calculated were discarded, so that each project had at least 5 variables to contribute to the model.

Finally, an analysis of the relationship between variables was performed. For this, a correlation matrix was generated, since it is an "indicator of the strength of the statistical relationship between two random variables or two signals" [50]. A correlation matrix indicates a possible dependency between the variables, in order to identify if there were variables that measured the same characteristic. It was found that three variables were related to each other: NM, PM, and WMC. In order to avoid redundancy and have a simpler model, the WMC and PM variables were discarded.

Once this procedure was completed, a data set was built consisting of 6 variables corresponding to the design metrics, in addition to the effort and time variables as shown in Table 4, and 21 project records.

Table 4. Selected variables for experimentation.

Explanatory Variable	Response Variable
CBO: Coupling Between Objects	Effort (human-hours)
NM: Number of Methods	Time (hours)
NV: Number of Variables per class	
CF: Coupling Factor	
MHF: Method Hiding Factor	
AHF: Attribute Hiding Factor	

Table 5 show descriptive statistics of the 21 project data set used, which it is available in [51].

Table 5. Descriptive statistics of the 21 project data set.

Variable	Missing Values	Min	Max	Mean	Median	SD
CBO	0	1	2	1.833	2	0.365
NM	0	0	17.5	7.142	8	5.545
NV	0	1	8	3.785	3.5	1.529
CF	0	0	1	0.2604	0.214	0.267
MHF	0	0	0.024	0.004	0.003	0.005
AHF	0	0.009	1	0.461	0.161	0.482
ESFUERZO	0	18.333	575	169.572	109.25	143.492
TIEMPO	0	55	2300	511.809	320	556.629

5. Experiments and Results

The design of the experiments, the method of validation of the models and the main results obtained are presented below.

5.1. Models Validation

To verify that the model values were valid and representative for the model, it was necessary to perform a validation of the models. To do this, leave-one-out cross-validation

(LOOCV) was used. Cross-validation is a "technique used to assess the quality of machine learning models and select reliable and stable models" [52].

In LOOCV, each value works as a test, and the other n-1 elements as a training set. Thus, "the first validation set contains only the first case, x_1, the second validation set contains only the second case, x_2, and so on" [53].

We used LOOCV because it is ideal for a small data set. In addition, LOOCV can be useful to detect overfitting in a model. If the model performs high on the training data but poorly on the test data, this may be an indicator that the model is overfitting the data. Since the data set contains 21 projects, the LOOCV was performed with 21 iterations, one for each method.

5.2. Prediction Models Building

To generate the prediction models, the Python programming language was used, using the scikit-learn library. Statistical models were created for the previously mentioned regressions, where the value of the coefficients belonging to the selected metrics was obtained, in addition to the value of the independent variable.

To compare these statistical models of our proposal, predictions were also generated with two machine learning approaches mentioned in the literature, such as Decision Trees and an ensemble algorithm such as Random Forest. Figure 4 shows the flow for the experimentation proposed in this study.

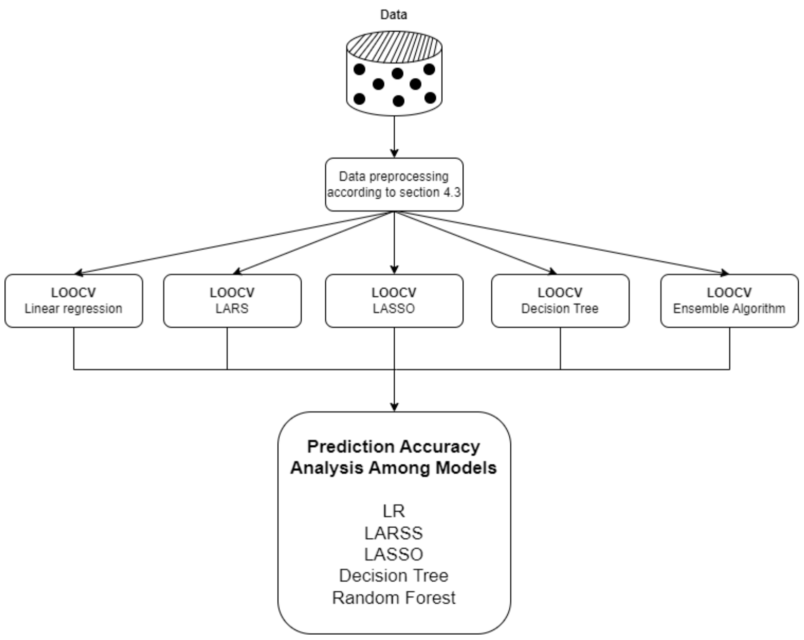

Figure 4. Description of the process followed for experimentation.

Because Random Forest has the disadvantage that a model cannot be generated, but instead takes an average of n trees from random samples, the numbers of 1, 5, 10, and 100 trees were used. In the same way, the error of the estimation of effort and time was generated, using from 1 to 500 random trees.

For the Decision Tree algorithm, the prediction of the attributes for time and effort was performed. For the depth of the tree, a value of 5 was used, since the number of variables is 6, and we wanted to avoid overfitting. Finally, the conclusions of the tree were obtained in sentences constructed in the if–then manner.

5.3. Results Analysis

After building the models for each of the estimation techniques, the prediction of effort and software development time of each of the projects belonging to the data set was carried out, and their results were compared. For the proposed regressions, it was possible to obtain the value of the coefficients of each explanatory variable (unlike machine learning algorithms used), and with them, build a model. Tables 6 and 7 show the coefficients of each statistical model built, where it can be observed that the coefficients obtained with a LASSO model are not similar to LARS and Linear Regression (the latter two are similar).

It can also be observed that to estimate the effort from design artifacts, Attribute Hiding Factor (AHF) and Coupling Between Objects (CBO) are the most influential, since they have coefficients farthest from zero, while Method Hiding Factor (MHF) is the metric that least influences the model. In addition, it can be seen with this type of model that AHF influences positively and CBO negatively. To estimate the development time, Coupling Factor (CF) is the most influential.

Using the data in Tables 6 and 7, software engineers and project managers can make estimates from their design artifacts. For example, for the Linear Regression Model, the weighting of the metrics and the intercept were calculated in order to substitute the values and obtain a model as shown in (8).

$$effort = -7.435CBO - 1.994NM - 5.405NV + 1.995CF - 1.275MHF + 9.179AHF + 354.1187 \tag{8}$$

Similarly, the value of the coefficients for the LARS regression and the LASSO regression were obtained. In addition to this, the Mean Square Error (MSE) value and Mean Absolute Error (MAE) also were calculated.

Table 6. Coefficients of statistical models for effort estimation.

Coefficient	**Linear**	**LARS**	**LASSO**
CBO	−7.435	−7.436	−8.581
NM	1.994	1.995	2.426
NV	−5.405	−5.405	−34.235
CF	1.995	1.996	127.598
MHF	−1.275	−1.275	0
AHF	9.179	9.197	220.671
Independent Term	354.118	354.1187	162.543

Table 7. Coefficients of statistical models for time estimation.

Coefficient	**Linear**	**LARS**	**LASSO**
CBO	3.614	1.173	−37.902
NM	−1.780	−1.734	2.019
NV	−1.682	−1.627	−126.944
CF	7.707	7.891	674.039
MHF	−4.417	−1.068	0
AHF	6.960	−2.214	634.994
Independent Term	1246.867	3106.842	578.956

Table 8 shows the comparison of the MSE and MAE of each model in effort estimation. As can be seen in Table 8, Linear Regression has the lowest MSE and MAE values. As can be seen in Figures 5 and 6, for Random Forest, as the number of trees increases, the error value tends to decrease, becoming constant from approximately 5 trees for effort, and 10 trees for time. Therefore, it is concluded that if Random Forest is used for estimation, it is not necessary to include more than 5 or 10 trees. On the other hand, the highest error value corresponds to the Decision Tree.

In Table 9, the MSE and MAE values of each model to predict software time are shown. As can be seen, the technique with the lowest MSE and MAE corresponds to Random Forest, followed by Decision Tree. The technique with the highest MSE and MAE corresponds to LASSO Regression.

Table 8. Accuracy metrics of the different models in effort estimation.

Prediction Model	MSE	MAE
Multiple Linear Regression	18,236.402	92.588
LARS Regression	23,063.221	102.550
LASSO Regression	25,644.562	109.492
Random Forest (Estimators = 1)	25,191.2457	113.216
Random Forest (Estimators = 5)	22,979.584	105.562
Random Forest (Estimators = 10)	20,910.574	100.521
Random Forest (Estimators = 100)	18,751.494	99.103
Random Forest (Estimators = 500)	18,535.951	97.334
Decision Tree	30,593.138	119.466

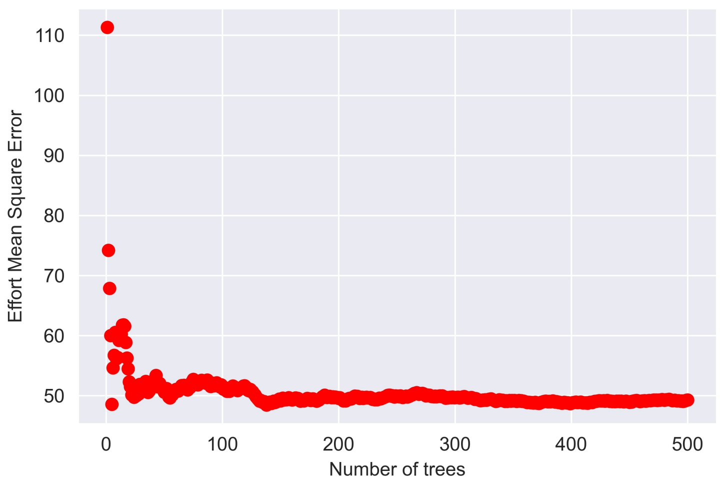

Figure 5. Effort estimation error increasing the number of estimators (trees) in Random Forest.

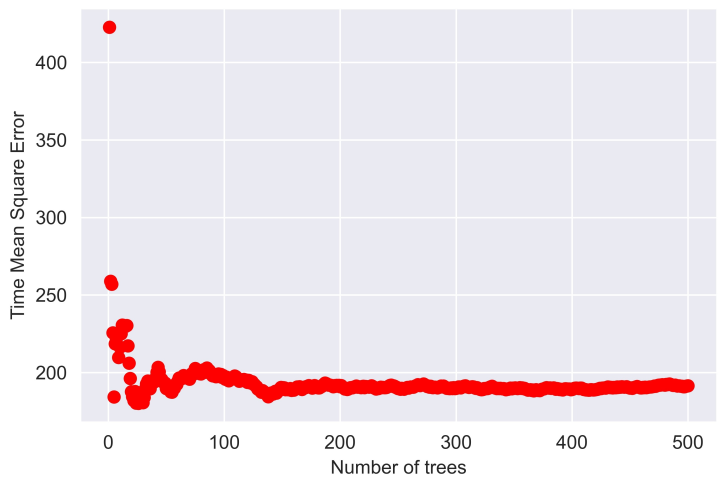

Figure 6. Time estimation error increasing the number of estimators (trees) in Random Forest.

Table 9. Accuracy metrics of the different models in time estimation.

Prediction Model	MSE	MAE
Multiple Linear Regression	499,625.814	485.094
LARS Regression	499,650.702	485.116
LASSO Regression	534,830.176	503.375
Random Forest (Estimators = 1)	413,663.855	398.633
Random Forest (Estimators = 5)	349,224.904	353.035
Random Forest (Estimators = 10)	309,150.506	343.638
Random Forest (Estimators = 100)	305,221.572	345.514
Random Forest (Estimators = 500)	288,001.178	338.384
Decision Tree	382,961.926	385.577

When the prediction results using the different prediction models are analyzed, it can be noted that the Random Forest and Decision Tree techniques are the ones that, in the prediction results, have the time values closest to the actual time values. In contrast, the LARS and LASSO regressions were the least effective, due to the fact that the results of the time prediction are further away from the actual values of time.

Tables 10 and 11 show four normality tests (χ^2, Shapiro–Wilk, Skewness, and Kurtosis) for each set of errors (MSE and MAE) for effort prediction. Since not all prediction errors met the assumption of normality, it was decided to apply the Friedman test to establish if it had at least one set of prediction errors different from the others. Since we did not obtain significant results between the groups, tests were no longer carried out between each pair of groups.

Table 10. Normal statistical test of MSE values for effort predictions.

Model	χ^2	SW	Skewness	Kurtosis	Friedman
Linear	0.0000	0.0000	0.0000	0.0001	
LARS	0.0000	0.0000	0.0001	0.0045	
LASSO	0.0000	0.0000	0.0001	0.0045	0.3568
Random Forest (10 estimators)	0.0000	0.0000	0.0000	0.0007	
Decision Tree	0.0000	0.0000	0.0000	0.0024	

Table 11. Normal statistical test of MAE values for effort predictions.

Model	χ^2	SW	Skewness	Kurtosis	Friedman
Linear	0.0001	0.0001	0.0008	0.0111	
LARS	0.0036	0.0001	0.0038	0.1049	
LASSO	0.0001	0.0001	0.0006	0.0089	0.3568
Random Forest (10 estimators)	0.0000	0.0000	0.0003	0.0075	
Decision Tree	0.0003	0.0000	0.0008	0.0247	

Similarly, in Tables 12 and 13 the same normality tests were run for the MSE and MAE errors for time prediction.

In Tables 12 and 13, it is observed that there is at least one set of errors statistically different from 95%. For this reason, it was compared for each set of errors if there was a statistically significant difference between each pair of differences. If all normality tests are met, the t-paired test was executed, otherwise the non-parametric Wilcoxon test was executed.

Table 12. Normal statistical test of MSE values for time predictions.

Model	χ^2	SW	Skewness	Kurtosis	Friedman
Linear	0.0000	0.0000	0.0000	0.0000	
LARS	0.0000	0.0000	0.0000	0.0000	
LASSO	0.0000	0.0000	0.0000	0.0000	0.0011
Random Forest (10 estimators)	0.0000	0.0000	0.0000	0.0025	
Decision Tree	0.0000	0.0000	0.0000	0.0003	

Table 13. Normal statistical test of MAE values for time predictions.

Model	χ^2	SW	Skewness	Kurtosis	Friedman
Linear	0.0001	0.0000	0.0001	0.0019	
LARS	0.0000	0.0000	0.0001	0.0019	
LASSO	0.0001	0.0001	0.0006	0.0089	0.0011
Random Forest (10 estimators)	0.0009	0.0000	0.0014	0.0505	
Decision Tree	0.0003	0.0000	0.0009	0.0231	

In Table 14, it can be seen that the only significant differences at 95% confidence are the following:

- Linear Regression and Random Forest;
- LARS and Random Forest;
- LASSO and Random Forest.

On the other hand, the results of Linear Regression and LARS are practically identical. Finally, there is no difference between a Decision Tree and any type of statistical regression for time prediction in this data set.

Table 14. Statistical tests for data distribution between models by data set for time prediction.

Model	χ^2	SW	Skewness	Kurtosis	T-Paired or Wilcoxon Test
Linear-LARS	0.0000	0.0000	0.0000	0.0000	0.9999
Linear-LASSO	0.0000	0.0000	0.0000	0.0008	0.4733
Linear-Random Forest	0.0020	0.0000	0.4460	0.0005	0.0126
Linear-Decision Tree	0.0048	0.0000	0.4239	0.0015	0.0759
LARS-LASSO	0.0000	0.0000	0.0000	0.0008	0.4733
LARS-Random Forest	0.0020	0.0000	0.4450	0.0005	0.0126
LARS-Decision Tree	0.0048	0.0000	0.4246	0.0015	0.0759
LASSO-Random Forest	0.0004	0.0000	0.0968	0.0004	0.0080
LASSO-Decision Tree	0.0063	0.0000	0.9884	0.0014	0.0759
Random Forest-Decision Tree	0.0173	0.0002	0.1359	0.0152	0.3737

6. Threats to Validity

According to [54], there are four categories of valid threats in search-based predictive modeling for Software Engineering: conclusion, internal, external, and construction. In Table 15 we show the threats to validity, indicating how we addressed each one if it applied to our study.

Table 15. Validity threats addressed.

Categorty	Description	How It Was Addressed
C	Is there any statistical verification of the results?	All our conclusions were based on the results of statistical tests.
C	Does the study verify all the assumptions of the chosen statistical test?	We selected the statistical tests based on the data distribution.
C	Do the results of the study take into account validation bias if any?	To reduce any bias in the validation, a leave-one-out cross-validation method was used.
I	Does the study ensure that there is no confounding effect on the relationship between the explanatory and the dependent variables due to extraneous attributes?	We propose to estimate effort and time with design artifacts.
CT	Does the study use effective representatives of the concepts which represent explanatory variables?	Metrics at the design stage are the core of our proposal.
CT	Does the study use performance measures that are effective representatives of the capability of the developed models?	We used the MSE (a most-used metric in the literature) and MAE.
ET	Are the data sets used by the study real industrial data sets?	In the industry, the data set with metrics in the design stage is novel in the area of software engineering.
ET	Does the study provide proper details for replicability?	The process used for prepossessing and processing are detailed for replicability.

7. Conclusions and Future Work

The objective of this work was to compare the estimates of software projects based on effort and time, between machine learning techniques and different statistical regression techniques, based on software design artifacts.

The projects selected to carry out the experiments were works belonging to university students of the Software Engineering career path. These projects have the characteristic of being mainly small and do not have high complexities.

Furthermore, according to the analysis of the results of the models, it can be observed that the model that has a lower error in the prediction of effort is Multiple Linear Regression. For time estimation, Random Forest has lowest error.

However, for effort estimation using design artifacts, there is no 95% significant difference when using any model. For time estimation, the assembled machine learning method is statistically different from the other models.

Furthermore, since the statistical models can be better interpreted by seeing the weights of each design metric value, it is concluded that using these regressions for software estimation is feasible.

It is important to mention that to carry out this research, there were limitations. Mainly student projects were evaluated because the public data sets belonging to the industry did not have design metrics, being above all techniques of the construction stage. In addition to the fact that those data sets that did have metrics from the design stage were not public.

Finally, as future work, it is expected to be able to test other Machine Learning algorithms as assembled algorithms, to make estimates in the design stage. Likewise, there is a need to have strategies to systematize the collection of historical data belonging to student projects, but they were not public.

Author Contributions: Conceptualization, Á.J.S.-G. and K.C.-V.; methodoloy, Á.J.S.-G. and K.C.-V.; software, M.S.G.-H. and J.C.P.-A.; formal analysis, Á.J.S.-G.; investigation, J.C.P.-A.; data curation, M.S.G.-H., Á.J.S.-G. and K.C.-V.; Validation, J.C.P.-A.; writing– original draft preparation, Á.J.S.-G. and K.C.-V.; writing—review and editing, J.C.P.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was financed with an institutional grant from the Consolidation Fund for Academic Groups 2023, of the General Office of Academic Development and Educational Innovation of the Universidad Veracruzana.

Data Availability Statement: We made this new data set with metrics based on the software design stage available to other researchers at https://docs.google.com/spreadsheets/d/18lm9AEwW0VmuzkT5de8PR8ldDIVpc-4L/edit?usp=drive_link&ouid=111159099755278392012&rtpof=true&sd=true [48]. Any questions can be written to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. *ISO 9000:2015*; Quality Management Systems—Fundamentals and Vocabulary (Norm 9000). International Organization for Standardization: Geneva, Switzerland, 2015.
2. Jorgensen, M.; Shepperd, M. A systematic review of software development cost estimation studies. *IEEE Trans. Softw. Eng.* **2007**, *33*, 33–53. [CrossRef]
3. Wen, J.; Li, S.; Lin, Z.; Hu, Y.; Huang, C. Systematic literature review of machine learning based software development effort estimation models. *Inf. Softw. Technol.* **2012**, *54*, 41–59. [CrossRef]
4. Bardsiri, V.K.; Abang Jawawi, D.N.; Khatibi, E. Towards improvement of analogy-based software development effort estimation: A review. *Int. J. Softw. Eng. Knowl. Eng.* **2014**, *24*, 1065–1089. [CrossRef]
5. Idri, A.; Azzahra Amazal, F.; Abran, A. Analogy-based software development effort estimation: A systematic mapping and review. *Inf. Softw. Technol.* **2015**, *58*, 206–230. [CrossRef]
6. Gautam, S.S.; Singh, V. The state-of-the-art in software development effort estimation. *J. Softw. Evol. Process.* **2018**, *30*, e1983. [CrossRef]
7. Ali, A.; Gravino, C. A systematic literature review of software effort prediction using machine learning methods. *J. Softw. Evol. Process.* **2019**, *31*, e2211. [CrossRef]
8. Mahmood, Y.; Kama, N.; Azmi, A. A systematic review of studies on use case points and expertbased estimation of software development effort. *J. Softw. Evol. Process.* **2020**, *32*, e2245. [CrossRef]
9. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 68–82. [CrossRef]
10. Anandhi, V.; Chezian, R.M. Regression Techniques in Software Effort Estimation Using COCOMO Dataset. In Proceedings of the 2014 International Conference on Intelligent Computing Applications, Coimbatore, India, 6–7 March 2014; pp. 353–357.
11. Aas, E.J. Design quality and design efficiency; definitions, metrics and relevant design experiences. In Proceedings of the IEEE 2000 First International Symposium on Quality Electronic Design, San Jose, CA, USA, 20–22 March 2000; pp. 389–394.
12. Kitchenham, B.A.; Linkman, S.J. Design metrics in practice. *Inf. Softw. Technol.* **1990**, *32*, 304–310. [CrossRef]
13. Pomorova, O.; Hovorushchenko, T. Artificial neural network for software quality evaluation based on the metric analysis. In Proceedings of the East-West Design & Test Symposium (EWDTS 2013), Rostov-on-Don, Russia, 27–30 September 2013; pp. 1–4.
14. Goyal, S.; Bhatia, P.K. A Non-Linear Technique for Effective Software Effort Estimation using Multi-Layer Perceptrons. In Proceedings of the 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 14–16 February 2019; pp. 14–16.
15. Shukla, S.; Kumar, S. Applicability of Neural Network Based Models for Software Effort Estimation. In Proceedings of the 2019 IEEE World Congress on Services (SERVICES), Milan, Italy, 8–13 July 2019; pp. 339–342.
16. Assefa, Y.; Berhanu, F.; Tilahun, A.; Alemneh, E. Software Effort Estimation using Machine learning Algorithm. In Proceedings of the 2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA), Bahir Dar, Ethiopia, 28–30 November 2022; pp. 163–168.
17. Ahmad, F.B.; Ibrahim, L.M. Software Development Effort Estimation Techniques Using Long Short Term Memory. In Proceedings of the 2022 International Conference on Computer Science and Software Engineering (CSASE), Duhok, Iraq, 15–17 March 2022; pp. 182–187.

18. Ilango, T.; Murugavalli, S. Advantage of using Evolutionary Computation Algorithm in Software Effort Estimation. *Int. J. Appl. Eng. Res.* **2014**, *9*, 30167–30178.
19. Bisi, M.; Goyal, N.K. Software development efforts prediction using artificial neural network. *Iet Softw.* **2016**, *10*, 63–71. [CrossRef]
20. Moosavi, S.H.S.; Bardsiri, V.K. Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation. *Eng. Appl. Artif. Intell.* **2017**, *60*, 1–15. [CrossRef]
21. Benala, T.R.; Mall, R. DABE: Differential evolution in analogy-based software development effort estimation. *Swarm Evol. Comput.* **2018**, *38*, 158–172. [CrossRef]
22. Karimi, A.; Gandomani, T.J. Software development effort estimation modeling using a combination of fuzzy-neural network and differential evolution algorithm. *Int. J. Electr. Comput. Eng.* **2021**, *11*, 707. [CrossRef]
23. Azath, H.; Mohanapriya, M.; Rajalakshmi, S. Software effort estimation using modified fuzzy C means clustering and hybrid ABC-MCS optimization in neural network. *J. Intell. Syst.* **2018**, *29*, 251–263. [CrossRef]
24. ul Hassan, C.A.; Khan, M.S.; Irfan, R.; Iqbal, J.; Hussain, S.; Ullah, S.S.; Umar, F. Optimizing deep learning model for software cost estimation using hybrid meta-heuristic algorithmic approach. *Comput. Intell. Neurosci.* **2022**, *2022*, 3145956. [CrossRef] [PubMed]
25. Khan, M.S.; Jabeen, F.; Ghouzali, S.; Rehman, Z.; Naz, S.; Abdul, W. Metaheuristic algorithms in optimizing deep neural network model for software effort estimation. *IEEE Access* **2021**, *9*, 60309–60327. [CrossRef]
26. Kaushik, A.; Singal, N. A hybrid model of wavelet neural network and metaheuristic algorithm for software development effort estimation. *Int. J. Inf. Technol.* **2022**, *14*, 1689–1698. [CrossRef]
27. Thamarai, I.; Murugavalli, S. An evolutionary computation approach for project selection in analogy based software effort estimation. *Indian J. Sci. Technol.* **2016**, *9*. [CrossRef]
28. Sharma, S.; Vijayvargiya, S. An optimized neuro-fuzzy network for software project effort estimation. *Iete J. Res.* **2023**, *69*, 6855–6866. [CrossRef]
29. Shukla, K.K. Neuro-genetic prediction of software development effort. *Inf. Softw. Technol.* **2000**, *42*, 701–713. [CrossRef]
30. Thamarai, I.; Murugavalli, S. A study to improve the software estimation using differential evolution algorithm with analogy. *J. Theor. Appl. Inf. Technol.* **2017**, *95*, 5587–5597.
31. Kassaymeh, S.; Abdullah, S.; Al-Betar, M.A.; Alweshah, M.; Salem, A.A.; Makhadmeh, S.N.; Al-Maitah, M.A. An enhanced salp swarm optimizer boosted by local search algorithm for modelling prediction problems in software engineering. *Artif. Intell. Rev.* **2023**, *56* (Suppl. 3), 3877–3925. [CrossRef]
32. Singh, S.P.; Singh, V.P.; Mehta, A.K. Differential evolution using homeostasis adaption based mutation operator and its application for software cost estimation. *J. King Saud-Univ.-Comput. Inf. Sci.* **2021**, *33*, 740–752. [CrossRef]
33. Gouda, S.K.; Mehta, A.K. Software cost estimation model based on fuzzy C-means and improved self adaptive differential evolution algorithm. *Int. J. Inf. Technol.* **2022**, *14*, 2171–2182. [CrossRef]
34. Wani, Z.H.; Bhat, J.I.; Giri, K.J. A generic analogy-centered software cost estimation based on differential evolution exploration process. *Comput. J.* **2021**, *64*, 462–472. [CrossRef]
35. Hernandez-Gonzalez, E.Y.; Sanchez-Garcia, A.J.; Cortes-Verdin, M.K.; Perez-Arriaga, J.C. Quality Metrics in Software Design: A Systematic Review. In Proceedings of the 2019 7th International Conference in Software Engineering Research and Innovation (CONISOFT), Mexico City, Mexico, 23–25 October 2019; pp. 80–86.
36. González-Hemández, S.; Sánchez-García, A.J.; Cortés-Verdín, K.; Pérez-Arriaga, J.C. Regression in Estimation of Software Attributes: A Systematic Literature Review. In Proceedings of the 2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT), San Diego, CA, USA, 25–29 October 2021; pp. 54–56.
37. Eberly, L.E. Multiple linear regression. in Topics in Biostatistics. *Methods Mol. Biol.* **2007**, *404*, 165–187.
38. Grégoire, T. Multiple Linear Regression. In *European Astronomical Society Publications Series*; Cambridge University Press: Cambridge, UK, 2005; Volume 66, pp. 45–72.
39. Tibshirani, R. Regression Shrinkage and Selection Via the Lasso. *J. R. Stat. Soc. Ser. B* **1996**, *58*, 267–288. [CrossRef]
40. Ranstam, J.; Cook, J.A. LASSO regression. *Br. J. Surg.* **2018**, *58*, 1348. [CrossRef]
41. Efron, B.; Hastie, T.; Johnstone, I.; Tibshirani, R. Least angle regression. *Ann. Stat.* **2005**, *32*, 407–499. [CrossRef]
42. Hastie, T.; Tibshirani, R.; Friedman, J.H. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*; Springer: New York, NY, USA, 2009.
43. Humphrey, W.S. *The Personal Software Process (Psp)*; Carnegie-Mellon University Pittsburgh, Software Engineering Institute: Pittsburgh, PA, USA, 2000.
44. Karner, G. Resource Estimation for Objectory Projects. *Object. Syst.* **1993**, *17*, 9.
45. Clemmons, R.K. Project Estimation With Use Case Points. *J. Def. Softw. Eng.* **2006**, *19*, 18–22.
46. Kusumoto, S.; Matukawa, F.; Inoue, K.; Hanabusa, S.; Maegawa, Y. *Effort Estimation Tool Based on Use Case Points Method*; Osaka University: Suita, Japan, 2005.
47. Abreu, F.B.; Melo, W. Evaluating the impact of object-oriented design on software quality. In Proceedings of the 3rd International Software Metrics Symposium, Berlin, Germany, 25–26 March 1996; pp. 90–99.
48. Appendix A: Dataset 37 Projects. Available online: https://docs.google.com/spreadsheets/d/18lm9AEwW0VmuzkT5de8PR8ldDIVpc-4L/edit?usp=drive_link&ouid=111159099755278392012&rtpof=true&sd=true (accessed on 27 March 2024).
49. Tukey, J.K. *Exploratory Data Analysis*; Addison-Wesley Publishing Company: Boston, MA, USA, 1977; Volume 2, pp. 131–160.

50. He, Z.; Jiao, S.M. Delay Estimation of Dynamic System Based on Correlation Coefficient. In Proceedings of the 2018 IEEE 4th International Conference on Control Science and Systems Engineering (ICCSSE), Wuhan, China, 24–26 August 2018; pp. 53–57.
51. Appendix B: Dataset 21 Processed Projects. Available online: https://drive.google.com/file/d/1G3niNiL0XPG7ZGi1sagBK3mmBgtFEsYj/view?usp=drive_link (accessed on 27 March 2024).
52. Li, L.; Yang, H.; He, Q.; Zhao, J.; Guo, T. Design and Realization of the Parallel Computing Framework of Cross-Validation. In Proceedings of the 2012 International Conference on Industrial Control and Electronics Engineering, Xi'an, China, 23–25 August 2012; pp. 1957–1960.
53. Berrar, D. *Cross-Validation*; Tokyo Institute of Technology: Tokyo, Japan, 2019.
54. Malhotra R.; Khanna, M. Threats to validity in search-based predictive modelling for software engineering. *Iet Softw.* **2018**, *12*, 293–305. [CrossRef]