COMPARATIVE ANALYSIS OF SOFTWARE EFFORT ESTIMATION USING DATA MINING TECHNIQUE AND FEATURE SELECTION

Abdul Latif 1,*, Lady Agustin Fitriana 2, Muhammad Rifqi Firdaus 3

Computer Science1,2,3
Sekolah Tinggi Manajemen Informatika dan Komputer Nusa Mandiri1,2,3
www.nusamandiri.ac.id
14002369@nusamandiri.ac.id 1, 14002386@nusamandiri.ac.id 2, 14002384@nusamandiri.ac.id 3

(*) Author Corresponding

Abstract—Software development involves several interrelated factors that influence development efforts and productivity. Improving the estimation techniques available to project managers will facilitate more effective time and budget control in software development. Software Effort Estimation or software cost/effort estimation can help a software development company to overcome difficulties experienced in estimating software development efforts. This study aims to compare the Machine Learning method of Linear Regression (LR), Multilayer Perceptron (MLP), Radial Basis Function (RBF), and Decision Tree Random Forest (DTRF) to calculate estimated cost/effort software. Then these five approaches will be tested on a dataset of software development projects as many as 10 dataset projects. So that it can produce new knowledge about what machine learning and non-machine learning methods are the most accurate for estimating software business. As well as knowing between the selection between using Particle Swarm Optimization (PSO) for attributes selection and without PSO, which one can increase the accuracy for software business estimation. The data mining algorithm used to calculate the most optimal software effort estimate is the Linear Regression algorithm with an average RMSE value of 1603.024 for the 10 datasets tested. Then using the PSO feature selection can increase the accuracy or reduce the RMSE average value to 1552.999. The result indicates that, compared with the original regression linear model, the accuracy or error rate of software effort estimation has increased by 3.12% by applying PSO feature selection.

Keywords: Software Effort Estimation, Machine Learning, Feature Selection, PSO, RMSE.

INTRODUCTION

Software Effort Estimation (SEE) is needed because software development is limited by predetermined costs and schedules. Estimation is the activity of estimating how many resources are needed to complete a project plan. Developer often faces a variety of difficult situations that make it fail.
such as software being delivered late, unreliable, using costs several times higher than originally estimated, and often exhibiting poor performance characteristics so that project managers have difficulty estimating projects which it runs. The project failure was caused by a management approach to developing software [1].

The success of a development project is influenced by many factors, including executive support, user involvement in the project, project management experience, clear business objectives, software infrastructure, and the use of formal development methodologies. Other factors are factors related to the timing and scope of the project, including the minimal scope and reliable estimation. A software project is considered a failure if the project exceeds 50% of the planning cost and passes the predetermined schedule. The accuracy of estimating and measuring a software project is very important in facilitating the resource manpower and estimation effort on an IT project [2]. Ideally, in estimating software effort/software effort estimation, machine learning techniques can be used to predict, control, or significantly reduce the effort associated with building software [3].

In this study, researchers will apply machine learning methods to predict software business estimates using the Linear Regression (LR) [4], Multilayer Perceptron (MLP) [5], Radial Basis Function (RBF) [6], and Decision Tree Random Forest (DTRF) [7]. It is hoped that this will generate new knowledge on what machine learning methods are most accurate for estimating software effort. As well as knowing how effective the use of Particle Swarm Optimization (PSO) is in increasing the accuracy of software effort estimation [7].

Several effort estimation techniques exist and they can be classified under three main categories [8]. These categories are:

1. Expert judgment: In this category, a project estimator tends to use his or her expertise which is based on historical data and similar projects to estimate software. This method is very subjective and it lacks standardizations and thus, cannot be reusable. Another drawback of this method is the lack of analytical argumentation because of the frequent use of phrases such as "I believe that..." or "I feel that...".

2. Algorithmic models: This is still the most popular category in the literature [9]. These models include COCOMO [10], SLIM [11], and SEER-SEM [12]. The main cost driver of these models is the software size, usually the Source Lines of Code (SLOC). Algorithmic models either use a linear regression equation, like the one used by Kok et al. (1990), or non-linear regression equations, those which are used by Boehm (1981).

3. Machine learning: Recently, machine learning techniques are being used in conjunction or as alternatives to algorithmic models. These techniques include neural networks, fuzzy logic, neuro-fuzzy, Genetic Algorithm, and regression trees. Machine learning models can incorporate historical data and can be trained to better predict software effort.

This study aims to provide a framework that enables managers to make reasonable estimates of resources, costs, and schedules. Then, these estimates are made within a limited time frame at the start of the project and must be updated regularly as the project progresses. So that we get the best scenario and the worst scenario and the project results can be limited. The benefits of this research are expected to minimize software project failures by providing a framework that enables project managers to make reasonable estimates of resources, costs, and schedules [4]. This paper is a continuation of previous research, which compares Linear Regression (LR), Multilayer Perceptron (MLP), Radial Basis Function (RBF), and Decision Tree Random Forest (DTRF) to get the best machine-learning algorithm to predict software effort estimation and improve the accuracy with feature selection Particle Swarm Optimizer (PSO).

Several previous studies by Nassif have discussed software effort estimation using a log-linear regression model based on the use case point (UCP) model to calculate software effort as well as fuzzy logic and multilayer perceptron neural network (MLP) models [8]. Then, Nassif continued his research using Regression Fuzzy Models [13]. In other research, BaniMustafa has predicted software effort estimation using three machine learning methods include Naïve Bayes, Logistic Regression, and DTRF [7]. Other research using kNN [14], neural networks [15], Artificial Neural Network (ANN) [16], Cascade-Correlation Neural Network (CNN) [17], Radial Basis and Generalized Regression [18]. In other cases, researchers used the particle swarm optimizer (PSO) feature selection to improve the accuracy of software effort estimation using the Artificial Neural Network algorithm [19]. Then a comparative study using tree/rule-based models (M5 and CART), linear models (ordinary least squares regression with and without various transformations, ridge regression (RR), and robust regression (RoR)), nonlinear models (MARS, least squares support vector machines, multilayered perceptron neural networks (NN), and radial basis function (RBF)) [5]. Previous literature in Software Effort Estimation can be seen in Table 1.
Table 1. Previous literature of Software Effort Estimation

| Author | Title                                                                 | Algorithm | Dataset       | Result             |
|--------|----------------------------------------------------------------------|-----------|---------------|--------------------|
| [8]    | Towards an early software estimation using log-linear regression and a multilayer perceptron model | log-LR    | Western       | MMER, RMS, MAE, SD |
|        |                                                                      | MLP       | CompuTop      |                    |
|        |                                                                      | UCP       | ISBSG         |                    |
|        |                                                                      | Schneider  |               |                    |
| [13]   | Software Development Effort Estimation Using Regression Fuzzy Models | Regression fuzzy logic | ISBSG | MAE, MBRE, MIBRE, SA, Scott-Knott |
| [7]    | Predicting Software Effort Estimation Using Machine Learning Techniques | -ANN      | COCOMO        | MAE                |
|        |                                                                      | -LR       |               | AUC                |
| [14]   | Kombinasi Median Weighted Information Gain Dengan K-Nearest Neighbor Pada Dataset Label Months Software Effort Estimation | -DTRF    | China         | RMSE               |
|        |                                                                      | -KNN+ Median WIG | Desharnais Kitchenham | |
| [18]   | Software Effort Estimation using Radial Basis and Generalized Regression Neural Networks | -Radial Basis NN | COCOMO81 | MMRE, MARE, VARE, Mean BRE |
|        |                                                                      | - GRNN    |               |                    |
| [15]   | Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points | GRNN      | agile projects | MSE, MMRE         |
|        |                                                                      | PNN       |               |                    |
|        |                                                                      | GMDH      |               |                    |
| [16]   | Proposing an Enhanced Artificial Neural Network Prediction Model to Improve the Accuracy in Software Effort Estimation | -ANN      | Cocomo81      | MMRE               |
|        |                                                                      | -COCOMO II |               |                    |
| [17]   | Software Effort Estimation in the Early Stages of the Software Life Cycle Using a Cascade Correlation Neural Network Model | -CNN      | industrial and educational projects | MMRE |
|        |                                                                      | -Multiple Linear Regression |               |                    |
| [19]   | Improving the Accuracy in Software Effort Estimation                | -ANN+PSO  | COCOMO I      | MMRE               |
|        |                                                                      |           | Nasa93        |                    |
| [5]    | Data Mining Techniques for Software Effort Estimation: A Comparative Study | OLS       | ISBSG         | MMRE               |
|        |                                                                      | RBF       | Nasa93        |                    |
|        |                                                                      | MLP       | Cocomo81      |                    |
|        |                                                                      |           | Desharnais Maxwell |               |

MATERIALS AND METHODS

This research had been done using several machine learning algorithms, namely LR, MLP, RBF, and DTRF. To improve accuracy, we use feature selection. The feature selection we use is PSO. The research process is shown in Figure 1.

Figure 1: Step of the research process.
A. Dataset Description

In this research, the dataset to be used is the Albercht dataset with 24 projects and 8 attributes, Kemerer with 15 projects and 8 attributes, China with 499 and 19 attributes, Cocomonasa2 with 101 and 24 attributes, Cocomonasa_v1 with 60 and 17 attributes of data, Desharnais with 81 and 12 attributes, Kitchenham with 145 and 10 attributes, Maxwell with 62 and 27 attributes, Miyazaki94 with 48 and 9 attributes, and cocomo81 with 63 projects and 17 attributes.

Data collection is using the dataset from many resources. The datasets used are Cocomo81 (1981), Desharnais (1989), Miyazaki (1994), Maxwell (2002), Kitchenham CSC (2002), Cocomo NASA v1 (2005), Cocomo NASA 2 (2006), China (2007), Albrecht (2009), and Kemerer.

The entire dataset is then calculated by comparing the performance results between the Linear Regression (LR) algorithm, Multilayer Perceptron (MLP), Radial Basis Function (RBF), and Decision Tree Random Forest (DTRF) using Waikato Environment for Knowledge Analysis (Weka) under the GNU General Public License.

Then the experiment was carried out using the addition of the PSO feature selection algorithm in all algorithms used. The resulting performance is based on the Root Mean Squared Error (RMSE). The algorithm with the lowest RMSE value is the best method for software effort estimation. The collected dataset has many variations of attributes and instances, as shown in Table 2.

| Dataset       | Attribute | Instance |
|---------------|-----------|----------|
| Albercht      | 8         | 24       |
| Desharnais    | 8         | 15       |
| Kemerer       | 19        | 499      |
| Cocomonasa1   | 17        | 60       |
| Cocomonasa2   | 24        | 101      |
| China         | 12        | 81       |
| Cocomo81      | 10        | 145      |
| Miyazaki94    | 27        | 62       |
| Kitchenham    | 9         | 48       |
| Maxwell       | 17        | 63       |

B. Linear Regression (LR) Model

This section presents the proposed linear regression model is presented, linear regression is the method most often used in effort estimation software and always gets high accuracy values [8]. According to Harlan [20], the dependent variable in linear regression is also called response or criterion, while the independent variable is also known as a predictor or regressor. The model used for simple linear regression can be described as follows:

\[ Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i \; ; \; i = 1, 2, \ldots, n \]  \hspace{1cm} (1)

Where:
- \( Y_i \) : Response for subject \( i \)
- \( X_i \) : Predictor for subject \( i \)
- \( \varepsilon_i \) : error for subject \( i \)

\( \beta_0 \) and \( \beta_1 \) are the parameters in the population to be estimated in the fitting model. Fitting the model with sample data will produce the equation:

\[ Y_i = b_0 + b_1 X_i \; ; \; i = 1, 2, \ldots, n \]  \hspace{1cm} (2)

C. Multi-Layer Perceptron (MLP) Model

Multilayer Perceptron (MLP) is a class of Artificial Neural Networks (ANN). This section presents the Multi-Layer Perceptron neural network model. The neural network structure is very suitable for calculating software effort estimates. The network will stop training when the number of epochs reaches 250 or when the Mean Squared Error (MSE) becomes zero or when the MU value exceeds 1e+10. The time is set to "infinity" which indicates that the training time has no control over when the exercise should be stopped. Two common activation functions have historically been both sigmoids, and are described by the equation:

\[ y (v_i) = \tanh (v_i) \text{ and } y (v_i) = (1 + e^{-v_i})^{-1} \]  \hspace{1cm} (3)

Backpropagation works through an iterative process using training data, comparing the predicted value of each network with each data contained in the training. In each process, the weight of the relation in the network is modified to minimize the Mean Squared Error (MSE) value between the predicted value of the network and the real value.

D. Radial Basis Function (RBF) Model

The radial basis Layer contains different types of neurons, which contains the Radial Basis Function (RBF) as an activation function. A single (same) radial basis layer may contain neurons with different radial basis functions. Radial Basis Function (RBF) artificial neural network is an artificial neural network model with one unit in the hidden layer, where the activation function is a basic function and a linear function in the output layer. This method is suitable for predicting software effort estimates. RBF is usually used to build a functional approach from an equation:

\[ y (x) = \sum_{i=1}^{N} w_i \varphi(||x - x_i||) \]  \hspace{1cm} (4)

Where the approximation function \( y(x) \) is represented as the sum of \( N \) radial basis functions,
each corresponding to a different center $x_i$, and weighted by the corresponding coefficient $w_i$.

D. Decision Tree Random Forest (DTRF) Model

The DTRF model consists of a collection of decision trees that grow in parallel. According to Nassif, the tree predictions are combined to make the overall tree prediction for the forest [4].

DTRF can be defined as an ensemble learning method for classification, regression, and other tasks that operate by building multiple decision trees at the time of training and issuing classes which are class (classification) mode or average/average (regression) prediction of each tree.

If the classifying ensemble is $h_1(x), h_2(x), \ldots, h_n(x)$, and with the training set randomly drawn from the random vector distribution $Y, X$, then to determine the margin function as follows:

$$mg(X,Y) = av_k(I(h_k(X) = Y)) - \max_{j \neq Y} av_k(I(h_k(X) = j))$$  

where $I(\cdot)$ is the indicator function. The margin measures the extent to which the average number of votes at $X, Y$ for the right class exceeds the average vote for any other class.

E. Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is a research-based on population, and this population includes lots of particles where each particle represents a solution of an optimization problem. During every iteration, each particle is updated by following two “best” values, pbest and gbest. After finding the two best values, the position and velocity of the particles are updated by the following two equations:

$$v_i^k = w v_i^{k-1} + c_1 r_1 (p_{\text{best}_i}^k - x_i^k) + c_2 r_2 (g_{\text{best}}^k - x_i^k)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1}$$ ...................................(6)

where $v_i^k$ is the velocity of the $i$th particle at the $k$th iteration, and $x_i^k$ is the current solution (or position) of the $i$th particle at the $k$th iteration. $c_1, c_2$ are positive constants, and $r_1, r_2$ are two random variables with a uniform distribution between 0 and 1.

In this equation, $w$ is the inertia weight which shows the effect of the previous velocity vector on the new vector. An upper bound is placed on the velocity in all dimensions $v_{\text{max}}$. This limitation prevents the particle from moving too rapidly from one region in the search space to another. This value is usually initialized as a function of the range of the problem.

RESULTS AND DISCUSSION

A. Evaluation Method

The dataset is calculated by comparing the performance results between LR, MLP, RBF, and DTRF. To evaluate the performance of the model or algorithm the data is separated into two subsets, namely learning process data and validation/evaluation data. The model or algorithm is trained by the learning subset and validated by the validation subset then 10-fold cross-validation is applied. Antoni Wibowo recommends 10-Fold Cross-validation is recommended as the best model selection that can provide a clearer estimate of accuracy than other cross-validations [21].

The researcher uses cross-validation to select the appropriate model by comparing the value of the Root Mean Squared Error Cross-Validation (RMSECV) with the following formula:

$$RMSECV = \sqrt{\frac{1}{10} \sum_{i=1}^{10} \left(\sum_{j=1}^{N_{cv}} r_{ij} - y_{ij}\right)^2}$$ ............................................(7)

In this section, the researcher will explain the use of machine learning methods to calculate estimated software effort estimation. The result from RMSE values of comparative analysis between machine learning algorithms namely LR, MLP, RBF, and DTRF are compared with the machine learning algorithm with the feature selection PSO shown in Table 3 and Table 4.

Table 3. The Results Obtained From LR, MLP, RBF, and DTRF

| Dataset   | LR  | MLP | RBF | DTRF |
|-----------|-----|-----|-----|------|
| Alberth   | 13.007 | 22.24 | 12.42 | 12.92 |
| Desharnais| 2988.93 | 5992.12 | 4052.81 | 3348.23 |
| Kemerer   | 281.06 | 303.82 | 256.10 | 238.54 |
| Cocomonas1| 431.76 | 310.36 | 516.99 | 403.44 |
| Cocomonas2| 1142.46 | 1025.20 | 1004.62 | 825.10 |
| China     | 968.62 | 1444.62 | 4816.94 | 2088.45 |
| Cocomo81  | 1480.80 | 1651.88 | 1616.80 | 1288.80 |
| Miyazaki94| 155.92 | 192.67 | 238.83 | 198.90 |
| Kitchenham| 2302.34 | 8407.07 | 9297.63 | 8528.53 |
| Maxwell   | 6306.54 | 6849.10 | 6507.27 | 7197.51 |

The table above shows the RMSE results through the calculation of all the selected methods. Of the 10 data used to have different RMSE values. This shows that the method used gives different results according to the function of the algorithm for what calculation. However, the LR method gives the best value from the average results obtained.

To try and find the maximum result, this research adds the PSO selection feature to the calculation. The results can be seen in Table 4.
Table 4. The Results Obtained From LR, MLP, RBF, and DTRF with Feature Selection PSO

| Dataset        | Method + PSO | Method Without PSO |
|----------------|--------------|--------------------|
|                | LR           | MLP                |
| Albercht       | 10.691       | 11.045             |
| Deshmaris      | 2920.137     | 5254.074           |
| Kemerer        | 261.052      | 389.714            |
| Coconomas1     | 659.657      | 358.027            |
| Coconomas2     | 976.800      | 1826.799           |
| China          | 1077.627     | 1468.317           |
| Cocone81       | 1442.859     | 2160.937           |
| Miyazaki94     | 157.504      | 249.214            |
| Kitchenham     | 2329.636     | 7871.083           |
| Maxwell        | 5762.562     | 7510.228           |
|                | RBF          | DTRF               |
|                | 8.946        | 10.413             |
|                | 3195.768     |                    |
|                | 257.341      |                    |
|                | 419.651      |                    |
|                | 809.119      |                    |
|                | 1907.217     |                    |
|                | 1273.597     |                    |
|                | 198.814      |                    |
|                | 8452.060     |                    |
|                | 6593.583     |                    |
|                | 10.691       |                    |
|                | 2329.636     |                    |
|                | 5762.562     |                    |
|                | 10.413       |                    |
|                | 1273.597     |                    |
|                | 198.814      |                    |
|                | 8452.060     |                    |
|                | 6593.583     |                    |

The table above shows some data that displays the results with a significant change in value after adding the feature selection. However, there are still data that give the same results. The addition of the PSO feature selection can increase the value for the difference from the RMSE value obtained.

C. Result Evaluation

In this section, the significance level is used to determine how influential the use of the PSO feature selection is in increasing accuracy or reducing the RMSE value.

The level of significance is the threshold used to determine significance. If the p-value is less than or equal to the level of significance, the data are considered statistically significant.

• As a general rule, the level of significance (alpha) is set at 0.05, meaning that the probability of the two data groups being equal is only 5%.

• Using a higher confidence level (lower p-value) means that the experimental results will be considered more significant.

• If you want to increase the confidence level of your data, decrease the p-value even more to 0.01. Lower p values are commonly used in manufacturing when detecting product defects. A high level of confidence is essential to ensure that every part produced works according to its function.

• For hypothesis testing experiments, a significance level of 0.05 is acceptable.

A test that uses F-distribution, named by Sir Ronald Fisher, is called an F-Test. F-distribution or the Fisher-Snedecor distribution is a continuous statistical distribution used to test whether two observed samples have the same variance.

F-Test compares two variances, $s_x$ and $s_y$, by dividing them. Since variances are positive, the result is always a positive number. The critical value for F-Test is determined by the equation.

$$F = \frac{(S_X^2)/n_i}{(S_Y^2)/(n_j)}$$

where $S_X^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$ and $S_Y^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2$.

If the variances are the same, then the variance ratio is 1. The larger sample variance must be in the F-ratio numerator and the smaller sample variance in the denominator. Thus, this ratio is always greater than 1 and makes hypothesis testing easier.

After being calculated, then compared, the most optimal machine learning method between LR, MLP, RBF, and DTRF with the PSO Selection Feature can be seen in table 5 below.

Table 5. Comparison Of The Results Obtained From LR, MLP, RBF, and DTRF with Feature Selection PSO

| Method      | Mean of RMSE Improvement |
|-------------|--------------------------|
| Without PSO |                         |
| With PSO    |                         |
| Percentage  |                         |
| Status      |                         |
| LR          | 1607.149                 |
| MLP         | 2619.912                 |
| RBF         | 2832.040                 |
| DTRF        | 2413.043                 |
| LR          | 1559.852                 |
| MLP         | 2709.941                 |
| RBF         | 2409.984                 |
| DTRF        | 2311.525                 |
| 2.94%       | Not enhanced             |
| 3.44%       | Not enhanced             |
| 14.90%      | enhanced                 |
| 4.21%       | enhanced                 |

The sample standard deviation (S) of population X (without PSO) is considered to be less than or equal to the sample standard deviation (S) of population Y (with PSO). The p-value equals 0.4425, (p(X<Y) = 0.5575). Standard deviation X is 535.32, and Y is 488.94. The test statistic F equals 1.1987, which is in the 95% region of acceptance: [1.09, 9.2766]. F=S_Y^2/(S_X^2) = 1.1987, which is in the 95% region of acceptance: [1.09, 9.2766]. F=S_Y^2/(S_X^2) = 1.1987, which is in the 95% region of acceptance: [1.09, 9.2766]. F=S_Y^2/(S_X^2) = 1.1987, which is in the 95% region of acceptance: [1.09, 9.2766]. F=S_Y^2/(S_X^2) = 1.1987, which is in the 95% region of acceptance: [1.09, 9.2766]. F=S_Y^2/(S_X^2) = 1.1987, which is in the 95% region of acceptance: [1.09, 9.2766].
the RMSE value with a significant level (alpha) = 0.05. The results show that the value of F = 1.094 exceeds the significant level (alpha) value.

CONCLUSION

Time, cost, and labor are important factors in the software development process, an effective software development process that can be achieved by evaluating these parameters at an early stage of the project. The estimated evaluation of software efforts will lead to an increase in the efficiency of software development and increase its success rate. Based on this research, the data mining algorithm used to calculate the most optimal software effort estimate is the Linear Regression algorithm with an average RMSE value of 1607.149 for the 10 datasets tested. Then using the PSO feature selection can increase the accuracy or reduce the RMSE average value to 1559.852. The result indicates that, compared with the original regression linear model, the accuracy or error rate of software effort estimation has increased by 2.94% by applying PSO feature selection. Some other computation technologies such as genetic algorithms with another method to increase the accuracy can be explored and applied on software effort estimation models in the future.

REFERENCE

[1] B. Jeng, D. Yeh, D. Wang, S. L. Lhu, and C. M. Chen, “A Specific Effort Estimation Method Using Function Point,” J. Inf. Sci. Eng., vol. 27, no. 4, pp. 1363–1376, 2011.

[2] D. Pratiwi, "Implementation of Function Point Analysis in Measuring the Volume Estimation of Software System in Object Oriented and Structural Model of Academic System," Int. J. Comput. Appl., 2013.

[3] G. R. Finnie, G. E. Wittig, and J.-M. Desharnais, “A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models,” J. Syst. Softw., vol. 39, no. 3, pp. 281–289, Dec. 1997.

[4] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, “A comparison between decision trees and decision tree forest models for software development effort estimation,” in 2013 Third International Conference on Communications and Information Technology (ICCIT), 2013, pp. 220–224.

[5] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens, “Data mining techniques for software effort estimation: A comparative study,” IEEE Trans. Softw. Eng., vol. 38, no. 2, pp. 375–397, 2012.

[6] M. O. Elish, “Improved estimation of software project effort using multiple additive regression trees,” Expert Syst. Appl., 2009.

[7] A. BaniMustafa, “Predicting Software Effort Estimation Using Machine Learning Techniques,” in 2018 8th International Conference on Computer Science and Information Technology (CSIT), 2018, pp. 249–256.

[8] A. B. Nassif, D. Ho, and L. F. Capretz, “Towards an early software estimation using log-linear regression and a multilayer perceptron model,” J. Syst. Softw., vol. 86, no. 1, pp. 144–160, Jan. 2013.

[9] I. Wieczorek, “Improved Software Cost Estimation – A Robust and Interpretable Modelling Method and a Comprehensive Empirical Investigation,” Empir. Softw. Eng., vol. 7, no. 2, pp. 177–180, 2002.

[10] B. W. Boehm, Software Engineering Economics. Prentice Hall, 1981.

[11] Putnam and L. H, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," IEEE Trans. Softw. Eng., vol. 4, no. 4, pp. 345–361, 1987.

[12] D. D. Galorath and M. W. Evans, Software Sizing, Estimation, and Risk Management. Auerbach Publication, 2006.

[13] A. B. Nassif, M. Azzeh, A. Idri, and A. Abran, “Software Development Effort Estimation Using Regression Fuzzy Models,” Comput. Intell. Neurosci., vol. 2019, pp. 1–17, Feb. 2019.

[14] I. Kurniawan, “Kombinasi Median Weighted Information Gain Dengan K-Nearest Neighbor Pada Dataset Label Months Software Effort Estimation," J. Teknoinfo, vol. 14, no. 2, p. 138, Jul. 2020.

[15] A. Panda, S. M. Satapathy, and S. K. Rath, “Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points,” Procedia - Procedia Comput. Sci., vol. 57, pp. 772–781, 2015.

[16] I. Attarzadeh, A. Mehranzadeh, and A. Barati, “Proposing an enhanced artificial neural network prediction model to improve the accuracy in software effort estimation,” Proc. - 2012 4th Int. Conf. Comput. Intell. Commun. Syst. Networks, CISCyN 2012, pp. 167–172, 2012.

[17] A. B. Nassif, L. F. Capretz, and D. Ho, “Software effort estimation in the early stages of the software life cycle using a cascade correlation neural network model,” Proc. - 13th ACIS Int. Conf. Softw. Eng. Artif.
[17] P. V. G. D. P. Reddy, K. R. Sudha, P. R. Sree, and S. N. S. V. S. C. Ramesh, “Software Effort Estimation using Radial Basis and Generalized Regression Neural Networks,” vol. 2, no. 5, pp. 87–92, 2010.

[18] Z. Dan, “Improving the Accuracy in Software Effort estimation using ann model based on PSO,” in Proceedings of 2013 IEEE International Conference on Service Operations and Logistics, and Informatics, 2013, pp. 180–185.

[19] J. Harlan, Analisis Regresi Linear. Yogyakarta: Penerbit Gunadarma, 2018.

[20] A. Wibowo, “10-Fold Cross Validation,” MTI Bina Nusantara, 2017. [Online]. Available: https://mti.binus.ac.id/2017/11/24/10-fold-cross-validation/. [Accessed: 02-Jan-2021].