Implementation of Random Forest Regression for COCOMO II Effort Estimation

Ilham Cahya Suherman[1]
Department of Technology Management
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
cahya.19092@mhs.its.ac.id

Riyanarto Sarno[2]
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
riyanarto@if.its.ac.id

Sholiq[3]
Department of Information Systems
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
sholiq@is.its.ac.id

***Abstract*—One of Project Manager early activity is to estimate time, and cost based on given scope, which can help project manager to plan schedule and used resources. Estimation is very important in project management because a bad result of estimation will result in bad management of project and may cause failure. There are methods that can be used to estimate software development effort; COCOMO II is one method that commonly used. Many researcher before have been used algorithm, such as Bat, Bee Colony, or MOPSO to increase COCOMO II estimation accuracy. However, as the technology advanced, there are a lot more options that can be used to predict software effort estimation based on COCOMO, such as machine learning. In this paper, we compare machine learning algorithm with tuning parameter method to know whether tuning parameter estimation is better than machine learning estimation or vice versa. In this paper, we use Random Forest Regression as machine learning algorithm to estimate the effort. We also compare it with another machine learning algorithm, Support Vector Regression, and Bee Colony Method as parameter tuning method. The results of experiment is evaluated by their error rate. The results show that Random Forest Regression is better than Support Vector Regression and Bee Colony Method.**

Keywords—effort estimation, COCOMO II, Random Forest Regression

I. INTRODUCTION

Project Manager (PM) is a role that has responsibilities to maintain project management process groups such as initiating, planning, executing, monitoring, controlling, and closing the project according to PMBOK [1]. One of PM early activity is to estimate time, and cost based on given scope, which can help project manager to plan timeline and used resources. In software development industry, the challenge of project manager is to make sure that the high quality software can be achieved with resources as few as possible. Estimation is very important in project management because a bad result of estimation will result in bad management of project and may cause failure. Based on [2], about 65% failed projects are caused by Management factors; one of it is poor estimation method.

One of methods that commonly used to estimate software development effort is Constructive Cost Model (COCOMO) II. COCOMO II [3] rely on Kilo Line of Code (KLOC) multiplied by constants, Effort Multipliers, and Scale Factors [3].

Many previous researches used algorithm, such as Fuzzy Logic Model [4], Bat Algorithm [5], MOPSO [6], and Bee Colony [7], to optimize COCOMO parameters in order to increase estimation accuracy. However, as the technology advanced, there are a lot more options that can be used to predict software effort estimation based on COCOMO, such as machine learning.

Therefore, this research aims to study whether tuning the constant parameter of COCOMO II is a better method than using machine learning algorithm to predict the estimation of software development effort or vice versa. Support Vector Regression and Random Forest Regression are machine learning algorithm that used in this research with the help of programming language, Python.

In this paper, we provide the study of the implementation of machine learning algorithms, Support Vector Regression and Random Forest Regression, as the comparison to parameter tuning method for COCOMO II effort estimation. In Section 2, related work about software development estimation method and steps of the working methodology of the research will be discussed. In Section 3, the result and the analysis of the research will be presented. In the last section, the research is concluded, whether tuning constant parameter of COCOMO II is a better method than using machine learning algorithm or vice versa.

II. RELATED WORK

Various studies have been conducted to optimize COCOMO II estimation results. Some studies used statistical models to predict time and effort. Various studies also used statistical models, but those models are used to change parameter values in the COCOMO formula, and some studies use machine learning algorithm to improve the COCOMO model in estimating time and effort in software development. There are also studies that comparing several methods or algorithms in order to find out, which method or algorithm that has the best accuracy and small error when used for COCOMO effort and time estimation.

Langsari et al. [6], [8], [9] have done several studies about optimizing COCOMO II parameter in order to increase its estimation accuracy. In [6] they applied Fuzzy Multi-Objective Particle Swarm Optimization (Fuzzy MOPSO) to optimize time and effort parameters of COCOMO II on NASA93 dataset. They optimized COCOMO II time and effort by using MATLAB to calibrate COCOMO II coefficients, A, B, C, and D. In the end, they found new parameters (A = 4.852, B = 0.2830, C = 2.802, D = 0.3615) and able to reduce MRE and MMRE significantly, and make the estimation close to its actual time and effort.

Sarno et al. [10] applied different neural network architectures to adjust the parameters of COCOMO effort estimation with the help of Backpropagation algorithm to

improve its architecture. The conclusion of the research shows that the modified neural network has better MMRE than basic neural network.

Kumari and Pushkar [11] compared MOPSO and Support Vector Regression (SVR). The researcher wants to proof, whether SVR can outperform MOPSO estimation result's accuracy. The researcher used Weka to implement SVR and it also used Sequential Minimal Optimization (SMO) for the SVR. The results shown that SVR gives better accuracy compared to MOPSO result.

Meanwhile, Abdelali et al. [12] used machine learning algorithm, Random Forest Regression, in multiple datasets such as ISBSG R8, Tukutuku. And COCOMO. The researcher wants to increase its accuracy and to understand its process to generate the estimates. The results show that "the optimized random forest outperforms the regression trees model on all evaluation criteria".

III. Literature Review

A. Cost Constructive Model

Published by Boehm in 1981, Constructive Cost Model (COCOMO) was a regression-based model that allowed users to "reason about the cost and schedule implications of their development decisions, investment decisions established project budget and schedules, client negotiations and requested changes, cost or schedule or performance or functionality tradeoffs, risk management decisions, and process improvement decisions" as stated in [13]. In short, it is a formula to help users to estimate cost by estimating efforts and development time. Later in 2000, COCOMO II was introduced to address some issues from COCOMO 81.

The differences between COCOMO II and COCOMO 81 are COCOMO II has three calculation models (Application Composition Model, Early Design Model, and Post-Architecture Model), 17 Effort Multipliers and 5 Scale Factors. While COCOMO 81 has three system models (Organic, Semi-detached, and Embedded) and 15 Cost Drivers for Intermediate COCOMO. Below is the COCOMO II equation for effort estimation:

$$PM = A(KLOC)^{E} \prod_{1}^{17} EM \tag{1}$$

where,

$$E = B + 0.01 \, \Sigma_{1}^{5} SF \tag{2}$$

PM stands for person months means the effort of the software development, *a* is calibration factor, KLOC or Kilo Line of Code is a size of software, *b* is a scale factor, EM or Effort Multipliers is sum of effort multipliers that influence effort and time of development, and SF or Scale Factors for scaling the KLOC. EM and SF are determined by expert judgement [13]. Meanwhile, in order to obtain software development time, we can use (3) as stated below:

$$\text{TDEV} = c(\text{PM})^{F} \tag{3}$$

where,

$$F = D + 0.2(E - B) \tag{4}$$

TDEV is development time, PM is person months, F is a scale factor. The constants themselves have their own values. A is equal to 2.94, B is equal to 0.91, C is equal to 3.67, and D is equal to 0.28.

B. Support Vector Regression

Support Vector Regression (SVR) was first introduced by Drucker et al. [14] in 1997. It is one of the Support Vector Machine (SVM) supervised models [15] that can be used for regression. The model of SVR only depends on a subset of the training to predict data.

As explained in [16], SVR has set of parameters called *hyperparameters*; kernel and C, where kernel parameters control the decision boundary. There are kernels that can be used for SVR, linear, RBF, polynomial, and sigmoid. The usage of kernel depends on the dataset data distribution and shape. C parameters control errors penalty. The larger its value, the larger its penalty to error. SVR has objective function [17] in equation (3) that has to be minimized, where $||w||$ is the magnitude of the normal vector.

$$min_{w} \frac{1}{2} |w|^{2} \tag{3}$$

C. Random Forest Regression

Introduced by Tin Kam Ho [18] in 1995, Random Forests are method that can be used for constructing decision trees for classification or regression tasks. Later, Breiman and Cutler [19] developed its extension by combining bagging method and random selection features so that controlled variance can be used to construct a collection of decision trees.

Compared to other classifiers, like SVM and Neural Networks, it is believed that Random Forest can give better result and it is robust against overfitting as stated in [20]. Random Forest Regression itself predict new data by growing an *unpruned* regression for each n-tree bootstrap samples collected from the original data and then aggregating the predictions of the n-tree.

IV. Research Method

Below is the following diagram of the method of this research. The diagram shown as Fig. 1.

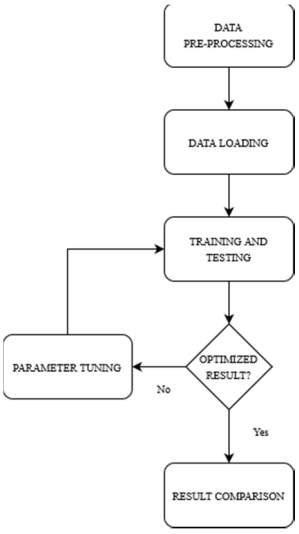

Fig. 1. Research Methodology

In this research we used Python as programming language and Google Colab as IDE with the help of sklearn [21] library for machine learning computation. For the dataset itself, we used COCOMO NASA93 [22] that available publicly on the internet .

A. Data Preprocessing

In the beginning, we conduct data preprocessing phase so that the program can read the dataset. The dataset itself has 93 rows, 26 columns, and formatted as ARFF (Attribute Relation File Format). We convert it into Excel format with the help of Weka. After that, we try to convert nominal values (low, normal, high, very high, and extremely high) of EM to its corresponding value. Then, we calculate the COCOMO II effort by using its equation (1) so that the result can be used later as a comparison to other methods or algorithms.

After that, we remove all irrelevant columns and left some usable column for the prediction program. Those columns are project ID, KSLOC, EM, SF, and Actual Effort. At last we convert the dataset format into comma separated values format so that the program can read the dataset. Table below is the part of preprocessed dataset.

TABLE I. PART OF DATASET

Project ID	KLOC	EM	Actual Effort
11	6	0.9609	24
12	100	0.8299	360
13	11.3	1.092	36
14	100	0.7089	215
15	20	0.9609	48
16	100	0.8951	360
17	150	1.2116	324
18	31.5	0.7089	60
19	15	0.9609	48
20	32.5	1.1761	60

B. Data Loading

After the data preprocessing phase, we load the dataset and import all Python machine learning libraries that needed for this research to Google Colab. The program read the dataset using pandas library. After that, we map the dataset into two separate variables, independent variables, KSLOC and EAF, as *X*, and dependent variable, Actual Effort, as *y*.

After that, we normalize the dataset using MinMaxScaler function from sklearn pre-processing library in order to minimize error and make the data range smaller so that the computation will be more precise, both in fitting and regression.

At last of this phase, we split the data into two different types, training and testing, using train_test_split function from sklearn model_selection library. We split the data into 80:20 ratio, 80 for training and 20 for testing. The function will also randomize the order of the data. Each program execution will result differently.

C. Training and Testing

In the beginning of this phase, we call all machine learning classes that we need to train and test the dataset. Those classes are sklearn.svm for SVR algorithm and sklearn.ensemble for RandomForestRegressor algorithm. Each algorithm has its own parameters. In SVR, we include parameters like epsilon, kernel, and C. While RandomForestRegressor has parameters like, n_estimators and max_depth. We also call sklearn.metrics that can help us to gain information about each machine learning's performance, including Mean Absolute Error, Mean Squared Error, and Max Error.

D. Parameter Tuning

In the first iteration, we are still using default parameters from the library. If the differences are still large, we modify the parameters' value manually in order to gain the most ideal value. After several iterations of trial and error, we find the best parameter that can used for testing the dataset. Best parameter means the results error is small.

At the end of this phase, we de-normalize the dataset using inverse_transform function so that we can retrieve the original value of the prediction.

E. Result Comparison

After we gain the result of each algorithm, we compare their result and we display it in a table. Things that we compare are MRE of data test and its MMRE. At the end, this phase will conclude whether tuning constant parameter of COCOMO II is a better method than using machine learning algorithm or vice versa.

We also calculate their Magnitude Relative Error (MRE) and Mean Magnitude Relative Error (MMRE), in order to find the difference between the estimated or predicted effort and the actual effort. The difference should be as small as possible, which means the accuracy of prediction is high. Below are equations of MRE and MMRE that used for calculating the differences:

$$\mathrm{MRE_i} = \frac{|Actual\ Effort_i - Estimated\ Effort_i|}{Actual\ Effort_i} * 100 \quad (2)$$

$$\mathrm{MMRE_i} = \frac{1}{N}\sum_{i=1}^{N} MRE_i \quad (3)$$

V. RESULT AND DISCUSSION

In this section, we want to discuss and display the result of the experiment. In the Table II, we show the Random Forest Regression parameters that we used in this experiment. According to sklearn documentation, the main parameters of Random Forest method is n_estimators and max_features. The larger the values, the better the results, but the trade-off is that the computation will take longer. But because in this research we only use two features, we replace max_features with max_depth to reduce the size of the model.

TABLE II. RANDOM FOREST REGRESSION PARAMETERS

Parameters	Value
n_estimatros	1000
max_depth	4

In Table III, it is the SVR parameters that we used in this research. The best Kernel used for the dataset is Linear, means that the dataset shape is linear.

TABLE III. SVR PARAMETERS

Parameters	Value
Kernel	Linear
C	100
Epsilon	0.1

We also estimate the effort using COCOMO II Bee Colony Method constant parameters (A and B) from [7] to know whether tuning parameter method estimation is better than machine learning estimation or vice versa.

We use these parameter values; A = 2.609 and B = 1.042 instead of using these parameter values A = 2.94 and B = 0.91 from original COCOMO II parameter. In the end, we calculate its MRE and compare it with another MRE. The comparison of each MRE shown in Figure 2.

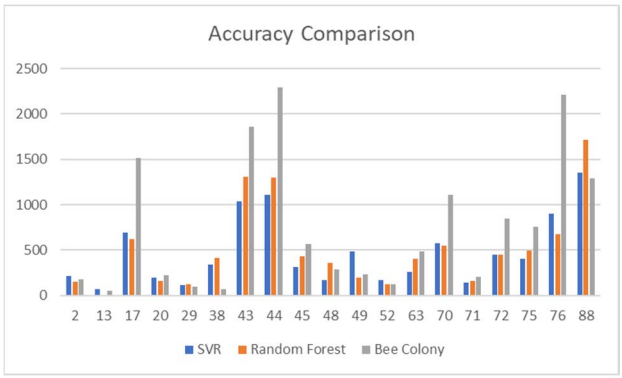

Fig. 2. Effort Estimation Accuracy Comparison

TABLE IV. EFFORT ESTIMATION COMPARISON

Project ID	Bee Colony	SVR	Random Forest
2	163.0	217	153
13	59.9	74	4
17	983.7	696	624
20	194.0	194	161
29	92.9	119	123
38	441.8	337	416
43	1070.6	1041	1305
44	1321.1	1110	1299
45	413.4	317	428
48	230.2	166	363
49	216.6	489	200
52	121.0	173	123
63	349.8	263	403
70	720.1	573	552
71	178.2	145	161
72	596.6	454	447
75	522.5	407	499
76	1418.3	899	678
88	1034.7	1351	1711
MMRE(%)	**115%**	**73%**	**54%**

The MMRE is calculated using (2). The MMRE value of Bee Colony, SVR, and Random Forest as 69%, 73%, and 54%. The result shows that Random Forest has the best result because its small MMRE compared to others algorithm/method. The detail is shown in Table IV.

VI. CONCLUSION

In software development, Project Manager has responsibility to estimate the effort of development so that cost, human resources, and timeline for the software development can be determined. Wrong estimation leads to project failure. Many studies have been conducted studies in order to answer the challenge. In this study, we propose Random Forest Regression to solve estimation problem. The proposed method is implemented using NASA93 dataset. The result shows that Random Forest Regression has the best result (54%) to estimate software effort using COCOMO II because of its small MMRE compared to SVR MMRE (73%) and Bee Colony Method MMRE (115%).

ACKNOWLEDGMENT

Authors give a thank to Department of Technology Management and Institut Teknologi Sepuluh Nopember, for supporting this research.

REFERENCES

[1] Project Management Institute, *A guide to the project management body of knowledge (PMBOK® guide)*. 2008.

[2] J. McManus and T. Wood-Harper, "A study in project failure," *British Computer Society - The Chartered Institute for IT*, 2008. .

[3] B. S. Barry Boehm, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald J. Reifer, "Software Cost Estimation with Cocomo {II}," *Englewood Cliffs, NJ:Prentice-Hall*, 2000.

[4] R. Sarno, J. Sidabutar, and Sarwosri, "Improving the accuracy of COCOMO's effort estimation based on neural networks and fuzzy logic model," in *Proceedings of 2015 International Conference on Information and Communication Technology and Systems, ICTS 2015*, 2016.

[5] Y. Amelia Effendi, R. Sarno, and J. Prasetyo, "Implementation of Bat Algorithm for COCOMO II Optimization," in *Proceedings - 2018 International Seminar on Application for Technology of Information and Communication: Creative Technology for Human Life, iSemantic 2018*, 2018.

[6] K. Langsari, R. Sarno, and Sholiq, "Optimizing time and effort parameters of COCOMO II using fuzzy Multi-objective Particle Swarm Optimization," *Telkomnika (Telecommunication Comput. Electron. Control.*, 2018.

[7] R. Y. Pratama, R. Sarno, and Sholiq, "Optimizing COCOMO II parameters using artificial bee colony method," in *Proceedings of the 11th International Conference on Information and Communication Technology and System, ICTS 2017*, 2018.

[8] K. Langsari, R. Sarno, and Sholiq, "Optimizing effort parameter of COCOMO II using Particle Swarm Optimization method," *Telkomnika (Telecommunication Comput. Electron. Control.*, 2018.

[9] K. Langsari and R. Sarno, "Optimizing COCOMO II parameters using particle swarm method," in *Proceeding - 2017 3rd International Conference on Science in Information Technology: Theory and Application of IT for Education, Industry and Society in Big Data Era, ICSITech 2017*, 2017.

[10] R. Sarno, J. Sidabutar, and Sarwosri, "Comparison of different Neural Network architectures for software cost estimation," in *Proceeding - 2015 International Conference on Computer, Control, Informatics and Its Applications: Emerging Trends in the Era of Internet of Things, IC3INA 2015*, 2016.

[11] S. Kumari and S. Pushkar, "Comparison and Analysis of Different Software Cost Estimation Methods," *Int. J. Adv. Comput. Sci. Appl.*, 2013.

[12] Z. Abdelali, H. Mustapha, and N. Abdelwahed, "Investigating the use of random forest in software effort estimation," in *Procedia Computer Science*, 2019.

[13] B. Boehm, R. Valerdi, J. A. Lane, and A. W. Brown, "COCOMO suite methodology and evolution," *CrossTalk*. 2005.

[14] H. Drucker, C. J. C. Surges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in Neural Information Processing Systems*, 1997.

[15] C. Cortes and V. Vapnik, "Support-Vector Networks," *Mach. Learn.*, 1995.

[16] A. Ben-Hur and J. Weston, "A user's guide to support vector machines.," *Methods Mol. Biol.*, 2010.

[17] M. Awad and R. Khanna, "Support Vector Regression," in *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, Berkeley, CA: Apress, 2015, pp. 67–80.

[18] T. K. Ho, "Random decision forests," in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 1995.

[19] L. Breiman, "Bagging predictors," *Mach. Learn.*, 1996.

[20] A. Liaw and M. Wiener, "Classification and Regression by randomForest," *R News*, 2002.

[21] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, 2011.

[22] T. J. Sayyad Shirabad, J. Menzies, "The PROMISE Repository of Software Engineering Databases," *School of Information Technology and Engineering, University of Ottawa, Canada*, 2005. [Online]. Available: http://promise.site.uottawa.ca/SERepository/. [Accessed: 25-Apr-2020].