12-2012

A Treeboost Model for Software Effort Estimation Based on Use Case Points

Luiz Fernando Capretz
University of Western Ontario, lcapretz@uwo.ca

Ali Bou Nassif
University of Western Ontario, abounas@uwo.ca

Citation of this paper:
@inproceedings{DBLP:conf/icmla/NassifCHA12, author = {Ali Bou Nassif and Luiz Fernando Capretz and Danny Ho and Mohammad Azzeh}, title = {A Treeboost Model for Software Effort Estimation Based on Use Case Points}, booktitle = {ICMLA (2)}, year = {2012}, pages = {314-319}, ee = {http://dx.doi.org/10.1109/ICMLA.2012.155}, crossref = {DBLP:conf/icmla/2012-2}, bibsource = {DBLP, http://dblp.uni-trier.de} } @proceedings{DBLP:conf/icmla/2012-2, title = {11th International Conference on Machine Learning and Applications, ICMLA, Boca Raton, FL, USA, December 12-15, 2012. Volume 2}, booktitle = {ICMLA (2)}, publisher = {IEEE}, year = {2012}, isbn = {978-1-4673-4651-1}, ee = {http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6403616}, bibsource = {DBLP, http://dblp.uni-trier.de} }

A Treeboost Model for Software Effort Estimation Based on Use Case Points

Ali Bou Nassif and Luiz Fernando Capretz
Department of ECE, Western University
{abounas, lcapretz}@uwo.ca

Danny Ho[1] and Mohammad Azzeh[2]
[1]NFA Estimation Inc.
[2]Applied Science University
[1]danny@nfa-estimation.com
[2]m.y.azzeh@asu.edu.jo

Abstract--Software effort prediction is an important task in the software development life cycle. Many models including regression models, machine learning models, algorithmic models, expert judgment and estimation by analogy have been widely used to estimate software effort and cost. In this work, a Treeboost (Stochastic Gradient Boosting) model is put forward to predict software effort based on the Use Case Point method. The inputs of the model include software size in use case points, productivity and complexity. A multiple linear regression model was created and the Treeboost model was evaluated against the multiple linear regression model, as well as the use case point model by using four performance criteria: MMRE, PRED, MdMRE and MSE. Experiments show that the Treeboost model can be used with promising results to estimate software effort.

Keywords-- Software effort estimation, use case points, project management, Treeboost Model, Stochastic Gradient Boosting.

I. Introduction

Predicting software cost and effort with good accuracy has been a challenge for many project managers and researchers. The Standish Chaos Report [1] states that 2 out of 3 projects fail to be delivered on time and within budget. Several cost estimation techniques have been used for software effort and cost prediction. These tools include algorithmic models, expert judgment, estimation by analogy and machine learning techniques. Software effort is a function of many factors such as software size and quality attributes; however, software size is the most important factor. The Source Lines of Code (SLOC) is one of the oldest size metrics and has been widely used by models such as COCOMO [2] and SLIM [3]. The SLOC metric has been criticized because it cannot be used early and it depends on the programming language and technology used to develop the project. Albrecht [4] introduced the function points metric to tackle the limitations of the SLOC metric; however, counting function points is sometimes subjective and complicated. Another available size metric is the use case points (UCP) which was proposed by G. Karner [5]. The UCP metric is computed based on the number and complexity of the use cases as well as actors in a use case diagram. Use case diagrams are developed in the requirements stage and they are usually included in the Software Requirements Specification (SRS).

In this research, we put forward a novel Treeboost (aka Stochastic Gradient Boosting) model to predict software effort from use case diagrams. The Treeboost algorithm was introduced by J. Friedman [6] [7]. This algorithm was developed to improve the accuracy of decision trees models. The inputs of our Treeboost model include software size, team productivity and project complexity. Software size is estimated based on the use case point (UCP) model as described in Section II, A. Team productivity is computed based on the environmental factors listed in Table I. Project complexity is estimated based on the rules proposed in Section IV, B. The Treeboost model was trained and tested using 59 and 25 data points respectively. To evaluate the Treeboost model, a multiple linear regression model was developed from the same 59 data points used to train the Treeboost model. The proposed Treeboost model is then evaluated against the multiple linear regression model as well as the UCP model using four different criteria. The evaluation experiments showed that the Treeboost model outperforms the multiple linear regression and UCP models and thus, can be used to predict software effort with promising results.

The remainder of this paper is organized as follows: Section II presents a background of terms used in this paper. Section III introduces related work whereas Section IV proposes the Treeboost and multiple linear regression models. In Section V, the Treeboost model is evaluated. Section VI lists threats to validity whereas Section VII concludes the paper and suggests future work.

II. Background

This section defines the main terms used in this paper which includes the UCP model, evaluation criteria, Treeboost algorithm.

A. Use Case Point Model

The use case point (UCP) model was first described by Gustav Karner in 1993 [5]. This model is used for software cost estimation based on the use case diagrams. First, the software size is calculated according to the number of actors and use cases in a use case diagram multiplied by their complexity weights. The software size is calculated through two stages. These include the Unadjusted Use Case Points (UUCP) and the Adjusted Use Case Points (UCP). UUCP is achieved through the summation of the Unadjusted Use Case Weight (UUCW) and Unadjusted Actor Weight (UAW). After calculating the UUCP, the Adjusted Use Case Points (UCP) is calculated. UCP is achieved by multiplying UUCP by the Technical Factors (TF) and the Environmental Factors (EF).

For effort estimation, Karner proposed 20 person-hours to develop each UCP. This approach is not always reasonable for all software houses so we propose to use the Treeboost algorithm.

B. *Evaluation Criteria*

To assess the accuracy of the proposed model, we have used the most common evaluation criteria used in software estimation.

- *MMRE:* This is a very common criterion used to evaluate software cost estimation models [8]. The Magnitude of Relative Error (MRE) for each observation *i* can be obtained as:

$$MRE_i = \frac{|Actual\ Effort_i - Predicted\ Effort_i\ |}{Actual\ Effort_i} \quad (1)$$

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i \quad (2)$$

MMRE is the most common method used for evaluating prediction models; however, this method has been criticized by others such as [9], [10] and [11]. For this reason, we used a statistical significant test to compare between the median of two samples based on the residuals. Since the residuals were not normally distributed, the non-parametric statistical test Mann-Whitney U has been used to assess the statistical significance between different prediction models.

- *MdMRE*: One of the disadvantages of the MMRE is that it is sensitive to outliers. MdMRE has been used as another criterion because it is less sensitive to outliers.

$$MdMRE = median(MRE_i) \quad (3)$$

- *PRED(x)*: The prediction level (PRED) is used as a complimentary criterion to MMRE. PRED calculates the ratio of a project's MMRE that falls into the selected range (x) out of the total projects.

$$PRED(x) = \frac{k}{n}. \quad (4)$$

where k is the number of projects where $MRE_i \leq x$ and n is the total number of observations. In this work, PRED(0.25) and PRED(0.5) have been used.

- *MSE*: The Mean Squared Error (MSE) is the mean of the square of the differences between the actual and the predicted efforts.

$$MSE = \frac{\sum_{i=1}^{N}(Actual\_Effort_i - Estimated\_Effort_i)^2}{N}. \quad (5)$$

The estimation accuracy is directly proportional to PRED (x) and inversely proportional to MMRE, MdMRE and MSE.

C. *Treeboost Model*

The Treeboost model is also called Stochastic Gradient Boosting (SGB) [7]. Boosting is a method to increase the accuracy of a predictive function by applying the function frequently in a series and combining the output of each function. In other words, as Kearns once asked [12], "can a set of weak learners create a single strong learner?" The main difference between the Treeboost model and a single decision tree is that the Treeboost model consists of a series of trees. The main limitation of the Treeboost is that it acts like a black box (similar to some neural network models) and cannot represent a big picture of the problem as a single decision tree does. The Treeboost model has the following characteristics:

- The Treeboost uses Huber-M loss function [13] for regression. This function is a hybrid of ordinary least squares (OLS) and Least Absolute Deviation (LAD). For residuals which are less than a cutoff point (Huber's Quantile Cutoff), the square of the residuals is used. Otherwise, absolute values are used. This method is used to alleviate the influence of outliers. For outliers, where residuals have high values, squaring the residuals will lead to huge values, so outliers will be treated with the "absolute values" method instead. The Huber's Quantile Cutoff value is recommended to be between 0.9 and 0.95. If it is 0.9, the residuals will first be sorted from small to high. Then, the smallest 90% of the residuals will be squared (OLS) and the other residuals (largest 10%) will be treated with the LAD method.
- In the Stochastic Gradient Boosting algorithm, "Stochastic" means that instead of using all data for training, a random percentage of training data points (50% is recommended) will be used for each iteration instead. This has yielded an improvement in the results.
- The Stochastic Gradient Boosting (SGB) algorithm has a factor called *Shrinkage* factor. Experiments show that multiplying each tree in the series by this factor (between 0 and 1) will delay the learning process and consequently, the length of the series will be longer to compensate for the shrinkage. This also leads to better prediction values.
- To improve the optimization of the process, an Influence Trimming Factor is applied. In the Treeboost model, the residual errors of a tree are used as inputs to the next consecutive iteration. The Influence Trimming Factor allows the rows with small residuals to be excluded. If this factor is 0.10, rows with residuals that represent less than 10% of the total residual weight will be ignored.

The Treeboost algorithm is described as:

$$F(x) = F_0 + A1 * T1(x) + A2 * T2(x) + ... + AM * TM(x). \quad (6)$$

Where F(x) is the predicted target, F_0 is the starting value, x is a vector which represents the pseudo-residuals, T1(x) is the first tree of the series that fits the pseudo-residuals (as defined below) and A1, A2, etc. are coefficients of the tree nodes. The Treeboost algorithm is applied based on the following rules:

1. Find the coefficient of F_0. This is the mean of the target variable (Software Effort).
2. Select the rows that will feed the next tree. If the stochastic factor is set to 0.5, 50% of the rows will be randomly chosen.
3. Sort the residuals of the rows being used and transform the residuals using Huber's Quantile Cutoff factor. The transformed residual values are called pseudo-residuals.
4. Fit the first tree (T1) to pseudo-residuals.
5. Calculate the mean of the pseudo-residuals in each of the terminal nodes. This mean becomes the predicted variable of the node.
6. Calculate the residuals between the predicted variable and the pseudo-residuals that fed the tree, and apply Huber's Quantile Cutoff factor again. Then, compute the mean of these residuals.
7. Calculate the boost coefficient (A1) of the node which is the difference between the mean residual value and the mean of the predicted values of the tree.
8. Multiply the boost coefficient by the shrink value to retard the learning process.

III. Related Work

This section presents related work regarding the Treeboost model and software estimation.

While Treeboost (Stochastic Gradient Boosting) models have been applied in many areas, to the best of our knowledge, there is only one modest work [14] where a Treeboost model was applied on software effort estimation.

M. Elish [14] compares a Stochastic Gradient Boosting model with other neural and regression models. The main limitation of Elish's work is that the Stochastic factor was set to 1. This means that all data points were used for training. However, the main goal of the SGB algorithm (the stochastic part) is that a random portion of the training data should be used for training as opposed to using all data. By setting the Stochastic factor to "1", the Stochastic Boosting Algorithm will no longer be "stochastic". Moreover, some important parameters such as the number of trees and shrinkage factor are missing. Furthermore, the model and other neural and regression models were only trained using 18 projects which is insufficient.

Decision trees and fuzzy decision trees algorithms such as [15], [16], [17] and [18] have been used in software effort prediction models.

None of the related work proposes a Treeboost model for software effort estimation from use case diagrams. Moreover, besides the modest work of Elish [14], we are among the first who proposed a Treeboost model for software effort estimation. Another contribution in this work is that we are simplifying the Use Case Point model by introducing a new approach to measure the project complexity.

IV. Regression and Treeboost Models

This section introduces the multiple linear regression and Treeboost models. Our dataset contains 84 projects of which 70% (59 projects) were randomly chosen to train the models and 30% (25 projects) were used to test the model. Each of the proposed models takes 3 inputs which include software size, productivity and project complexity.

A. Multiple Linear Regression Model

The multiple linear regression model was constructed using 59 data points. A normality test was applied and we found that "Effort" and "Size" were not normally distributed, so "ln(effort)" and "ln(size)" were used instead of "Effort" and "Size". The equation of the regression model is:

$$\ln(Effort) = 1.8 + 1.24 \times \ln(Size) + 0.007 \times Productivity + 0.12 \times Complexity. \quad (7)$$

Where Effort is measured in person-hours and Size in UCP. Productivity is measured based on Equation (8) and Complexity is measured as proposed in Section IV, B.

To measure the accuracy of the regression model, we measured the value of the coefficient of determination R^2 which is 0.8. This indicates that 80 % of the variation in Effort can be explained by the independent variables Size, Complexity and Productivity. Moreover, we measured the Analysis of Variance (ANOVA) of Equation (7) and the model parameters. The "p" value of the model is 0.000 which indicates that the relationship among the variables is significant. The "p" values of the independent variables ln(size), productivity and complexity are 0.000, 0.282 and 0.012, respectively. This shows that all independent variables are statistically significant at the 95% confidence level except "productivity" (p value > 0.05). Removing the variable "productivity" will not worsen the accuracy of the model; however, we decided to keep the "productivity" variable in the Treeboost model because this variable might be statistically significant in other data sets. We also measured the Variance Inflation Factor (VIF) of each independent variable to see if the multicollinearity issue (when one independent variable has a relationship with other independent variables) exists. We found that the highest VIF factor is for the variable "ln(Size)" which is 1.03. This indicates that the multicollinearity issue does not exit [19] (VIF is less than 4).

B. Model's Inputs

The inputs of the model are software size, productivity and complexity. Software size was estimated based on the UCP model as described in Section II, A.

The productivity factor was calculated based on Table I according to this equation:

$$Productivity = \sum_{i=1}^{8} E_i \times W_i. \qquad (8)$$

Where E_i and W_i are the Environmental factors of the UCP model and their corresponding weights as depicted in Table I.

TABLE I. ENVIRONMENTAL FACTORS [5]

E_i	Efficiency and Productivity Factors	W_i
E_1	Familiar with Objectory	1.5
E_2	Object oriented experience	1
E_3	Analyst capability	0.5
E_4	Stable requirements	2
E_5	Application experience	0.5
E_6	Motivation	1
E_7	Part-time workers	-1
E_8	Difficult programming language	-1

The complexity of the project is an important factor in software effort prediction. Complexity can be interpreted as an item having two or more elements [20] [21]. There are two dimensions of complexity. These include business scope such as schedule, cost, risk and technical aspect which is the degree of difficulty in building the product [21]. Technical complexity deals with the number of components of the product, number of technologies involved, number of interfaces and types of interfaces [21]. The project complexity can be classified as low complexity, medium complexity or high complexity [21]. Project complexity should be distinguished from other project characteristics such as size and uncertainty [20]. Complex projects require more effort to develop than simple projects that have the same size. In our research, we identify the project complexity based on five levels (from Level1 to Level5). The reason behind defining five levels is to be compatible with other cost estimation models such as COCOMO where cost drivers are classified into five or six levels (such as Very Low, Low, Nominal, High, Extra High). Additionally, this classification is compatible to the project complexity classification in [21]. Each level has its corresponding weight. The five complexity levels are defined as follows:

- Level1: The complexity of a project is classified as Level1 if the project team is familiar with this type of project and the team has developed similar projects in the past. The number and type of interfaces are simple. The project will be installed in normal conditions where high security or safety factors are not required. Moreover, Level1 projects are those of which around 20% of their design or implementation parts are reused (came from old similar projects). The weight of the Level1 complexity is 1.
- Level2: This is similar to level1 category with a difference that only about 10% of these projects are reused. The weight of the Level2 complexity is 2.
- Level3: This is the normal complexity level where projects are not said to be simple, nor complex. In this level, the technology, interface, installation conditions are normal. Furthermore, no parts of the projects had been previously designed or implemented. The weight of the Level3 complexity is 3.
- Level4: In this level, the project is required to be installed on a complicated topology/architecture such as distributed systems. Moreover, in this level, the number of variables and interface is large. The weight of the Level4 complexity is 4.
- Level5: This is similar to Level4 but with additional constraints such as a special type of security or high safety factors. The weight of the Level5 complexity is 5.

The Treeboost model proposed in our research work was trained using 59 data points based on the parameters listed in Table II. To avoid overfitting during the training process, 20% of the training rows were used for validation. The initial number of trees of the model was set to 1. The number of the trees is incremented by 1 up to a maximum number of 1,000. The optimal number of the trees is determined when the value of the pseudo-residuals is minimal based on the influence trimming factor. As shown in Figure 1, best validation results (the blue lower curve represents the training process and the red upper curve represents the validation process) were obtained when the number of trees was 1,000.

The analysis of variance (ANOVA) shows that the coefficient of determination (R^2) and the Root Mean Squared Error (RMSE) are 0.99 and 2,398, respectively in the training process. However, the R^2 and RMS values in the validation process are 0.93 and 15,250, respectively.

V. MODEL EVALUATION AND DISCUSSION

This section presents the evaluation of the Treeboost model against the regression as well as the UCP model based on the MMRE, PRED, MSE and MAE criteria.

TABLE II. MODEL'S PARAMETERS

# of trees	Huber Quantile Cutoff	Shrinkage Factor	Stochastic Factor	Influence Trimming Factor
1000	0.95	0.1	0.5	0.01

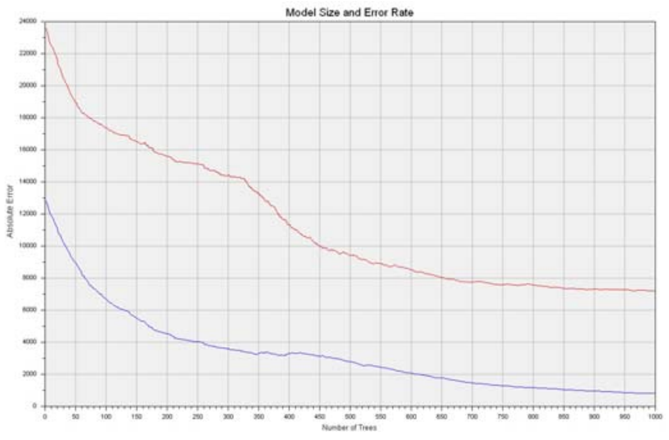

Figure 1. Number of trees

A. *Project Dataset*

This research is based on software effort prediction from use case diagrams. We have encountered many difficulties in acquiring industrial projects because revealing UML diagrams of projects is considered confidential. For this reason, we prepared a questionnaire that could help us obtain industrial data without actually having UML diagrams. In this questionnaire, we asked for the number of use cases in each project, the number of transactions of each use case, actual software effort as well as the project complexity, and factors contributing to productivity. Eighty four projects were collected from 3 main sources such that 58 are industrial projects and 26 are educational ones. Table III shows the characteristics of these datasets.

B. *Model Evaluation*

The Treeboost model was evaluated using 59 data points that were not included in the training stage. The criteria uses are MMRE, MdMRE, PRED(0.25), PRED(0.50) and MSE. Table IV shows the evaluation values of the Treeboost, multiple linear regression and UCP models.

TABLE III. DATASETS CHARACTERISTICS

Source	Ind1	Ind2	Edu
Min Effort (PH)	4,648	570	850
Max Effort (PH)	129,35 3	224,89 0	2380
Mean Effort	36,849	20,573	1,68 9
Standard Deviation (Effort)	39,350	47,327	496
Skewness (Effort)	1.37	3.26	-0.24

TABLE IV. MODEL EVALUATION

Criteria	Treeboost	Regression	UCP
MMRE	0.29	0.44	0.38
PRED(25)	64	8	40
PRED(50)	88	60	64
MdMRE	0.14	0.44	0.40
MSE	3.2+e7	5+e8	10+e8

C. *Discussion*

Table IV shows that the proposed Treeboost model surpasses the Regression and UCP models by 15% and 9%, respectively based on the MMRE criterion. Based on the MdMRE criterion, the Treeboost model surpasses the Regression and UCP models by 30% and 36%. Additionally, the Treeboost model gives better results based on PRED(0.25) and PRED(0.5), and this shows that the Treeboost model outperforms the other two models. To confirm the robustness of the Treeboost model, we measured the non-parametric Mann-Whitney U test between the Treeboost model and the other two models based on the MRE as shown in Table V. The Mann-Whitney U test was chosen because the values of the MRE were not normally distributed. Results show that the Treeboost model is statistically significant at the 95% confidence level.

TABLE V. MANN-WHITNEY U TEST

Models	Mann-Whitney (p-value)
Treeboost vs Regression	0.0003
Treeboost vs UCP	0.0361

VI. THREATS TO VALIDITY

1- The Treeboost model is a series of many small trees. The proposed model consists of 1,000 trees. The model was trained using 59 projects with efforts ranging between 507 and 224,890 person-hours. This shows that there is a significant difference in size between the smallest and the largest data point. Despite the good results obtained from the evaluation of the Treeboost model, this model would perform better if more training data points would have been used.

2- The neural network and linear/non-linear regression models have the capability to extrapolate the relationship between input and output vectors during the training process and thus, can map outputs to inputs even if these inputs are beyond (to a certain degree) the inputs of the training data points. However, this is not true with Treeboost models. Based on the decision tree models, the node with the largest number handles the last decision. The Treeboost model works in a similar way, but it is more complicated than the single decision tree. Nonetheless, the proposed Treeboost model also has limitations determined by the values of the three independent variables (size, productivity, complexity). To demonstrate this limitation, the Treeboost model was tested using an artificial dataset composed of 121 data points with sizes ranging between 1,000 and 4,000 UCP each incremented by 25. Since software size is the most important predictor in the model, productivity and complexity values were set to normal values (30 for productivity and 3 for complexity) for all projects. Figure (2) shows the Scatterplot graph between software size and predicted effort. The graph shows that the predicted effort of any project with a size greater than 2,475 UCP (productivity = 30 and complexity =3) is 185,004 person-hours. Although the size limitation varies based on the values of other predictors (productivity and complexity), it is not recommended to use the proposed Treeboost model to test projects of size more than 2,500 UCP.

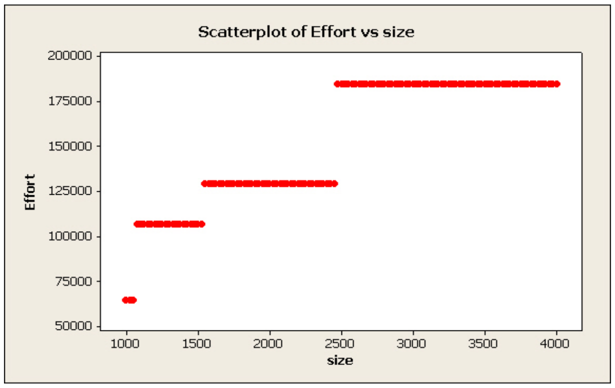

Figure 2. Scatterplot of size and predicted effort

VII. Conclusions

This paper proposed a Treeboost model to predict software effort based on three independent variables which include software size, productivity and complexity. The Treeboost model was developed through a series of 1,000 trees and was trained using 59 data points. The model was evaluated using 25 data points against the UCP, as well as a multiple linear regression model. The evaluation criteria used were MMRE, PRED, MSE and MdMRE. The proposed model is limited to projects of size around 2,475 UCP (around 200,000 person-hours). Results showed that the Treeboost model outperformed the multiple linear regression model as well as the UCP model in all evaluation criteria. Based on these results, we conclude that the Treeboost model can be used for software effort estimation and can compete with other regression models.

Future work will focus on calibrating the Treeboost model when new datasets are available.

References

[1] J. Lynch. Chaos manifesto. The Standish Group. Boston. 2009[Online]. Available: http://www.standishgroup.com/newsroom/chaos_2009.php.

[2] B. W. Boehm, *Software Engineering Economics.* Prentice-Hall, 1981.

[3] L. H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IEEE Transactions on Software Engineering,* vol. 4, pp. 345-361, 1978.

[4] A. Albrecht, "Measuring application development productivity," in *IBM Application Development Symp.* 1979, pp. 83-92.

[5] G. Karner, "Resource Estimation for Objectory Projects," *Objective Systems,* 1993.

[6] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *Annals of Mathematical Statistics,* vol. 29, pp. 1189-1232, 2001.

[7] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis,* vol. 38, pp. 367-378, 2002.

[8] L. C. Briand, K. E. Emam, D. Surmann, I. Wieczorek and K. D. Maxwell, "An assessment and comparison of common software cost estimation modeling techniques," in *International Conference on Software Engineering,* 1999, pp. 313-322.

[9] T. Foss, E. Stensrud, B. Kitchenham and I. Myrtveit, "A Simulation Study of the Model Evaluation Criterion MMRE," *IEEE Transactions on Software Engineering,* vol. 29, no. 11, pp. 985-995, 2003.

[10] M. Shepperd and C. Schofield, "Estimating software project effort using analogies,"*IEEE Transactions on Software Engineering,* vol. 23, pp. 736-743, 1997.

[11] I. Myrtveit and E. Stensrud, "Validity and reliability of evaluation procedures in comparative studies of effort prediction models," *Empirical Software Engineering,* vol. 17, pp. 23-33, 2012.

[12] M. Kearns, "Thoughts on Hypothesis Boosting," *Machine Learning Class Project,* 1988.

[13] P. J. Huber, "Robust Estimation of a Location Parameter," *Annals of Mathematical Statistics,* vol. 35, pp. 73-101, 1964.

[14] M. O. Elish, "Improved estimation of software project effort using multiple additive regression trees," *Expert Systems with Applications,* vol. 36, pp. 10774-10778, 9, 2009.

[15] K. Srinivasan and D. Fisher, *IEEE Transactions on Software Engineering,* vol. 21, pp. 126, 1995.

[16] A. S. Andreou and E. Papatheocharous, "Software cost estimation using fuzzy decision trees," in *23rd IEEE/ACM International Conference on Automated Software Engineering (ASE2008),* 2008, pp. 371-374.

[17] S. Huang, C. Lin and N. Chiu, "Fuzzy Decision Tree Approach for Embedding Risk Assessment Information into Software Cost Estimation Model," *Journal of Information Science and Engineering,* vol. 22, pp. 297-313, 2006.

[18] B. Baskeles, B. Turhan and A. Bener, "Software effort estimation using machine learning methods," in *22nd International Symposium on Computer and Information Sciences,* 2007, pp. 1-6.

[19] D. C. Montgomery, E. A. Peck and G. G. Vining, *Introduction to Linear Regression Analysis.* Wiley, 2012.

[20] D. Baccarini, "The concept of project complexity—a review," *Int. J. Project Manage.,* vol. 14, pp. 201-204, 8, 1996.

[21] L. Ireland, "Project complexity: A brief exposure to difficult situations," Asapm, Tech. Rep. PrezSez 10-2007, 2007.