Received: July 25, 2018

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Software Development Effort Estimation Using Random Forests: An Empirical Study and Evaluation

Abdelali Zakrani [1]* **Mustapha Hain [1]** **Abdelwahed Namir [2]**

[1]*ENSAM, Hassan II University, Casablanca, Morocco*
[2]*Faculte des Sciences Ben M'sik, Casablanca, Morocco*
* Corresponding author's Email: abdelali.zakrani@univh2c.ma

Abstract: There is evidence that Software Development Effort Estimation (SDEE) plays a crucial role in managing the software project and controlling its whole lifecycle; an accurate effort estimate allows an effective monitoring and efficient scheduling of tasks and resources. Although extensive research has been carried out on SDEE techniques, no single technique has been shown to be superior to other in all situation. Recently, there has been an increasing amount of literature on predicting software effort using Machine Learning (ML) methods. Among these ML techniques, regression tree-based models have gained a considerable attention due to their generalization ability and understandability. So far, very few studies have investigated the potential of Random Forests (RF) in software effort estimation. In this paper, a RF model is designed and adjusted empirically by varying the values of its key parameters. Prior to the parameters adjustment, we analysed their impact on RF model accuracy which allows an efficient tuning of the model during the training stage. The performance of the RF is then evaluated and compared with that of classical Regression Trees (RT). The evaluation was performed through the 30% hold-out validation method using five datasets: ISBSG R8, Tukutuku, COCOMO, Desharnais and Albrecht. To identify the most accurate technique, we employed three widely known accuracy measures: Pred(0.25), MMRE and MdMRE. The results obtained show that the adjusted random forest outperforms the regression trees model on all evaluation criteria. Moreover, the proposed model performs better than some recent techniques reported in the literature for software effort estimation.

Keywords: Software development effort estimation, Random forest, Regression trees, Accuracy evaluation.

1. Introduction

The issue of software development effort estimation has received considerable critical attention so that numerous estimation methods have been proposed by the researchers in order to help software project managers to make informed and rational decisions about the project under development [1]. In fact, an accurate and reliable estimate can play a vital role for the project managers insofar it allows them to plan, monitor and staff more effectively and efficiently the software project and hence they will be able to carry out projects on time and within budget. Whereas, managing the software project without accurate estimate makes their mission more challenging and stressful. As a result, the software project may run several risks such as: delayed deliveries, financial losses, poor quality of the deliverables, dissatisfied customers, and frustrated developers [2].

In order to provide a basis for the improvement of software estimation research. Jorgensen and Shepperd conducted a comprehensive and systematic literature review (SLR) in which they identified up to 11 estimation approaches proposed in 304 selected journal papers [3]. These approaches are based on different techniques varying from expert judgment [4, 5] and statistical analysis of historical project data [6-9] to artificial intelligence tools [10-13].

Recently, there has been growing interest in using machine learning (ML) techniques to model the complex relationship between effort and software attributes, especially when this relationship is not linear and doesn't seem to have any predetermined form. Within this context, Wen et al. carried out an extensive literature search for relevant studies published in the period 1991–2010 and selected 84 primary empirical studies [14]. They found that eight types of ML techniques have been employed in SDEE: Case-Based Reasoning (CBR), Artificial Neural Networks (ANN), Decision Trees (DT), Bayesian Networks (BN), Support Vector Regression (SVR), Genetic Algorithms (GA), Genetic Programming (GP), and Association Rules (AR). Among them, CBR, ANN, and DT are most frequently used. Their review also showed that the overall estimation accuracy of most ML models is close to the acceptable level in terms of MMRE and Pred(0.25) and better than that of non-ML models. Nevertheless, each ML technique has its own strength and weakness and the performance of any model depends mainly on the characteristics of the dataset used to construct the model (dataset size, outliers, categorical features and missing values).

MacDonell and Shepperd [15] claimed that combining two or more ML techniques can improve estimation accuracy if no dominant technique can be found. This point of view was approved later by the review done in [16] where it has been revealed that combined model usually generates better estimate than individual model does. These findings were also confirmed by Idri et al in their recent review of ensemble effort estimation in which they analysed 25 studies [17]. However, it has been noted that the number of comparative studies is still insufficient and recommend that researchers should conduct more experiments on ensemble effort estimation techniques and should develop a uniform experimental design [17].

With respect to model combination, regression trees are revealed to be the most used technique to build an ensemble effort estimation model [17]. Actually, regression trees have become very popular, thanks to their ease of use and interpretability [18] as well as their ability to deal with both numerical and categorical variables. However, traditional decision trees techniques also have their drawbacks. For instance, they are prone to overfitting on small training dataset and suboptimal performance. Fortunately, many of these disadvantages have been dealt with by some researchers who optimized the DT technique [19-21].

In response to the limitations of conventional decision tree, the current work presents a further investigation of random forest in SDEE. To the best of our knowledge, only a few implementations of random forests in a software effort estimation have been published [22]. This empirical study contributes to the existing literature not only by investigating the effectiveness of the random forests approach in predicting software effort but also by illustrating the impact of the parameters of RF model on the accuracy of the estimates and comparing the performance of the proposed model with traditional regression trees using 30% hold-out method (70% training, 30% testing). The empirical study uses historical projects from five datasets namely ISBSG, Tukutuku, COCOMO, Desharnais and Albrecht.

The present paper starts with a review of the related work on the regression tree-based software effort estimation models. It is followed in Section 3 by a description of historical projects datasets and evaluation criteria employed to evaluate the accuracy of the proposed models. In Section 4, we present the experimental design including the validation method used. Section 5 presents and discusses the results. Finally, the conclusions and future work are presented in Section 6.

2. Related work

Since its introduction, decision trees have been enjoying increased popularity. The number of applications in the fields of empirical software engineering is growing. In software effort estimation, Selby and Porter generated automatically a large number of decision trees using ID3 algorithm to classify the software modules that had high development effort or faults [23, 24]. The decision trees correctly identified 79.3% of the software modules on the average of across all 9600 trees generated. In [25], the authors compared CARTX, a partial implementation of CART, and backpropagation learning methods to traditional regression approaches. They found that the CARTX was competitive with SLIM, COCOMO and Function Points. However, their experiments showed the sensitivity of learning to various aspects of data selection and representation.

In [26], the authors applied fuzzy decision tree on 1000 selected projects data from ISBSG repository R8. Then they extracted a set of association rules to produce mean effort value ranges. The authors claimed that the proposed approach provides accurate effort estimation and there is strong evidence that the fuzzy transformation of cost drivers contribute to enhancing the estimation process. The authors in [27] applied a fuzzy version of ID3 on two datasets: COCOMO and Tukutuku. The results obtained indicate the performance of fuzzy ID3 over the crisp version of ID3 in terms of MMRE and Pred(0.25).

In another study [28], Azzeh developed an optimized model tree based on M5P with optimal parameters identified by the Bees algorithm to construct software effort estimation model. The optimized model tree has been validated over eight datasets and evaluated by 3-fold cross-validation method. The results have shown the performance of the optimized model tree over Stepwise Regression, Case-Based Reasoning and Multi-Layer Perceptron techniques. In [29], the authors employed an evolutionary algorithm to generate a decision tree tailored to a private software effort dataset. The evolutionarily-induced DT statistically outperform greedily-induced ones (J48, CART and BFTree), as well as traditional logistic regression.

To improve the accuracy of single tree-based model, Elish investigated the use Multiple Additive Regression Trees (MART) in software effort estimation and compared their performance with that of Linear Regression, Radial Basis Function and SVR models on NASA dataset [30]. The MART model outperforms the others in terms of MMRE and Pred(0.25). In other work [31], the authors designed a Treeboost, also called Stochastic Gradient Boosting, model to predict software effort based on the Use Case Point method. It is to note that the main difference between the Treeboost model and a single decision tree is that the Treeboost model consists of a series of trees. Results showed that the Treeboost model outperformed the Multiple Linear Regression (MLR) model as well as the Use Case Point model in all evaluation criteria adopted in the empirical study.

Despite these promising results, very few studies have assessed the performance of random forest technique in the field of software effort estimation [17] and the only works found in the literature are those of [22, 32]. In [22], a comparative study is performed between Multiple Linear Regression (MLR), Decision Trees (DT) and Decision Tree Forest (DTF). The authors used ISBSG R10 and Desharnais datasets and 10-fold cross-validation method to develop the DTF. The results demonstrate that DTF performs better than MLR and DT in terms of MMRE, MdMRE and Pred(0.25) and the robustness of DTF was confirmed by the non-parametric Mann-Whitney U Test.

3. Data description and evaluation criteria

This section describes the datasets used to perform the empirical study and presents the evaluation criteria adopted to compare the estimating capability of the SDEE models.

3.1 Data description

The data used in the present study come from five datasets namely Tukutuku, ISBSG R8 and COCOMO, Desharnais and Albrecht. Table 1 displays the summary statistics for these datasets.

The Tukutuku dataset contains 53 Web projects [33]. Each Web application is described using 9 numerical attributes such as: the number of html or shtml files used, the number of media files and team experience (for more details see Table 1). However, each project volunteered to the Tukutuku database was initially characterized using more than 9 software attributes, but some of them were grouped together. For example, we grouped together the following three attributes: number of new Web pages developed by the team, number of Web pages provided by the customer and the number of Web pages developed by a third party (outsourced) in one attribute reflecting the total number of Web pages in the application (Webpages).

The ISBSG Release 8 repository is a multi-organizational dataset containing more than 2,000 projects gathered from different organizations in different countries [34]. Major contributors are in Australia (21%), Japan (20%), and the United States (18%). To decide on the number of software projects, and their descriptions, a data pre-processing study was already conducted by [11], the objective of which was to select data (projects and attributes), in order to retain projects with high quality. The first step of this study was to select only the new development projects with high quality data and using IFPUG counting approach. The second step was concerned by selecting an optimal subset of numerical attributes that are relevant to effort estimation and most appropriate to use as effort drivers in empirical studies.

The original COCOMO' 81 dataset contains 63 software projects [35]. Each project is described by 14 attributes: the software size measured in KDSI (Kilo Delivered Source Instructions) and the remaining 12 attributes are measured on a scale composed of six linguistic values: 'very low', 'low', 'nominal', 'high', 'very high' and 'extra high'. These 13 attributes are related to the software development environment such as the experience of the personnel involved in the software project, the method used in the development and the time and storage constraints imposed on the software. Because the original COCOMO'81 dataset contains only 63 historical software projects and in order to have a robust empirical study, we have artificially generated, from the original COCOMO'81 dataset, three other datasets each one contains 63 software projects (see [36] for more details). The union of the four datasets constitutes the artificial COCOMO'81 dataset that is used in this study.

The Albrecht dataset [37] is a popular dataset used by many recent studies[38-40]. This dataset

Table 1. Description statistics of the selected datasets

Dataset	# of software project	# of attributes	Effort					
			Min	Max	Mean	Median	Skewness	Kurtosis
ISBSG (Release 8)	151	6	24	60 270	5 039	2 449	4.17	21.10
COCOMO	252	13	6	11 400	683.4	98	4.39	20.50
TUKUTUKU	53	9	6	5 000	414.85	105	4.21	20.17
DESHARNAIS	77	8	546	23 940	4 834	3 542	2.04	5.30
ALBRECHT	24	7	0.5	105.20	21.88	11.45	2.30	4.67

includes 24 projects developed by third generation languages. Eighteen out of 24 projects were written in COBOL, four were written in PL1, and two were written in DMS languages. There are five independent features: 'Inpcout', 'Outcount', 'Quecount', 'Filcount', and 'SLOC'. The two dependent features are 'Fp' and 'Effort' which is recorded in 1,000-person hours.

The Desharnais dataset was collected by [41]. Despite the fact that Desharnais dataset is relatively old, it is one of the large and publicly available datasets. Therefore, it still has been employed by many recent empirical studies, such as [22, 39, 40]. This dataset includes 81 projects (with nine features) from one Canadian software company. Four of 81 projects contain missing values, so they have been excluded from further investigation. The eight independent features are '*TeamExp*', 'ManagerExp', 'Length', 'Language', 'Transactions', 'Entities', 'Envergure', and 'PointsAdjust'. The dependent feature 'Effort' is recorded in 1,000 h.

The descriptive statistics of these datasets are presented in Table 1. Among these statistics, the 'Skewness' and 'Kurtosis' measures. As it can be seen, the effort exhibits a fairly high skewness and kurtosis. When the absolute value of skewness is lower than 2, and the absolute value of kurtosis is lower than 7, the distribution is assumed to be normal enough not to distort statistical estimation [42]. However, most values, in Table 1, exceeded these criteria. Positive, high skewness indicates high outliers, which explains the disparity between the mean value and the median value. Positive kurtosis suggests that most effort values were concentrated around the mean value, creating a sharp, high curve. In sum, the skewness and kurtosis of effort distribution implies that most software development efforts are relatively similar each other, except for several high outliers. Thus, this poses a challenge for challenge for developing accurate estimation methods [43].

3.2 Evaluation criteria

We employ the following criteria to assess and compare the accuracy of the effort estimation models. A common criterion for the evaluation of effort estimation models is Magnitude of Relative Error (MRE), which is defined as

$$MRE = \left|\left(\frac{Effort_{actual} - Effort_{estimated}}{Effort_{actual}}\right)\right| \quad (1)$$

The MRE values are calculated for each project in the dataset, while Mean Magnitude of Relative Error (MMRE) computes the average over N projects as follows:

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

Generally, the acceptable target value for MMRE is 25%. This indicates that on the average, the accuracy of the established estimation models would be less than 25%.

Another widely used criterion is the Pred(l) which represents the percentage of MRE that is less than or equal to the value *l* among all projects. This measure is often used in the literature and is the proportion of the projects for a given level of accuracy. The definition of Pred(l) is given as follows:

$$Pred(l) = \frac{k}{N} \quad (3)$$

Where N is the total number of observations and k is the number of observations whose MRE is less or equal to l.

A common value for l is 0.25, which is also used in the present study. The Pred(0.25) represents the percentage of projects whose MRE is less or equal to 0.25. The Pred(0.25) value identifies the effort estimates that are generally accurate whereas the MMRE is fairly conservative with a bias against overestimates [44, 45]. For this reason, MdMRE has been also used as another criterion since it is less sensitive to outliers (Eq. (4)).

$$MdMRE = median(MRE_i) \quad i \in \{1 \dots N\} \quad (4)$$

3.3 Statistical testing

Although the performance measures can show if any SDEE techniques are better than others in a descriptive and graphical manner, the remaining question is whether the observed differences are statistically significant [46]. We used the Mann-Whitney test at the significance level of 0.05 to check the significance difference between absolute errors of the SDEE techniques. This statistical test has been used because the absolute errors are not normally distributed.

4. Experimental design

In this section, we configure random forests models to predict software development effort and we examine the impact of the key parameters of RF model on the estimates accuracy. Therefore, the developed RF model is compared to decision tree model. In the first subsection, we present the methodological underpinnings of the random forests techniques and the tuning strategy used to find empirically an optimal model. In the second subsection, we present the configuration of Regression Tree used to generate the effort estimates.

4.1 Configuration of random forests model

The random forests method, introduced by Breiman [47] adds an additional layer of randomness to bootstrap aggregating (‘‘bagging”) and is found to perform very well compared to many other classifiers. In addition, it is robust against overfitting and very user-friendly [48].

The strategy of random forests is to select randomly subsets of m_{try} features to grow trees, each tree being grown on a bootstrap sample of the training set. This number, m_{try}, is used to split the nodes and is much smaller than the total number of features available for analysis [49] because each tree depends on the values of an independently sampled random vector and standard random forests are a combination of single tree predictors with the same distribution for all trees in the forest [47].

The use of random forest in software development effort estimation needs the determination of a set of parameters like: the number of trees constituting the forest (ntree), the number of features chosen randomly at the level of each node (m_{try}), the size of the sample 'in bag' (sampsize) and the maximum number of nodes of each tree (maxnodes). In this paper, a hyperparameter tuning approach is implemented. This latter relies more on experimental results than theory, and thus the best method to determine the optimal settings is to try many different combinations and evaluate the performance of each model.

Table 2. Experimental design of RF models

Datasets	**Random Forest parameterization**	
	Empirical study 1 varying m_{try}	**Empirical study 2 varying ntree**
ISBSG (R8)	From 1 to 5 ntree=300	From 100 to 2000 m_{try}=5
COCOMO	From 1 to 10 ntree=100	From 100 to 2000 m_{try}=7
TUKUTUKU	From 1 to 9 ntree=500	From 100 to 1000 m_{try}=5
DESHARNAIS	From 1 to 8 ntree=500	From 100 to 1000 m_{try}=5
ALBRECHT	From 1 to 7 ntree=500	From 100 to 1000 m_{try}=5

In order to perform a robust empirical study, we focus on two important hyperparameters for the induction of these forests namely the number of decision trees, ntree, and the number of features, m_{try}, considered by each tree when splitting a node. It should be noted that the software, R, used to perform this experimental study, implements a set of default hyperparameters for all models, but these are not guaranteed to be optimal for all cases, whereas the best hyperparameters are usually data-dependent. Hence, we conducted a series of experiments each time varying the parameter m_{try} and ntree to get the best estimates. Table 2 provides an overview of the followed experiment design.

4.2 Configuration of regression tree models

The parameters of the RT model were chosen in such a way that the error is minimal with one exception which is tree pruning. To avoid overfitting, the regression tree was built using 10-fold cross-validation and it was pruned to minimum cross-validation error. The minimum size of the node to split was set at 20 and the maximum depth of tree was set at 30. The splitting function chosen is ANOVA since it is a regression tree. The Minimum rows allowed in a node was set at 7 as recommended by Breiman [50]. It must be noted that the complexity parameter was set so that any split that does not decrease the overall lack of fit by a factor of 0.01 is not attempted. We used this configuration to generate the five regression tree models from the five datasets.

5. Results of the experimental Studies

The following section presents and discusses the results obtained when performing the aforementioned experimental design. The dataset was divided into two subsets: 70% of the historical projects as training set and 30% of the projects as test set. The training set is used to obtain the model without the participation of the test set, which is employed to assess the accuracy of the estimation model. Thus, RT and RF models were trained using 70% of the historical projects and 30% of remaining projects were maintained in order to perform a final validation/test of these models (see Table 3)

We performed our experiments with a 10-fold cross-validation approach in order to train our models with the training data and to ensure that our findings will generalize adequately with the independent test set and to avoid overfitting problem. It shall be noted that the internal 10-fold cross-validation process was performed using the abovementioned 70% of the training set.

5.1 Impact of the m_{try} and ntree on the accuracy of Random Forest Estimates

The objective of this subsection is to illustrate the impact of the ntree and m_{try} on the accuracy of estimates produced by RF models. The empirical studies were performed employing ISBSG, COCOMO and Tukutuku datasets and following the experimental design outlined in Table 3.

- COCOMO dataset:

In the first empirical study, the value of ntree was kept equal to 100 and 10 random forest models were generated by varying the value of m_{try} from 1 to 10. After that we have compared the performance of these models using COCOMO testing set. As it can be seen from the results obtained in Fig. 1-a, the accuracy of RF model is increasing, in general, with value of m_{try} in terms of Pred(0.25) until a certain value. The RF model with m_{try}=8 yields to better accuracy estimates (MMRE=1.07 and Pred(0.25)=33.33%).

In the second empirical study, we generated a series of Random forest models with m_{try}=7 and a number of trees varying from 100 to 2000. After that, the power of generalization of these models was compared using COCOMO testing set. Looking at the results showed in the Fig. 1-b, it is apparent that the accuracy is not monotonically increasing as the number of trees, ntree, increases. The best estimates were obtained when the value ntree is not too large (ntree=100 and when ntree=700).

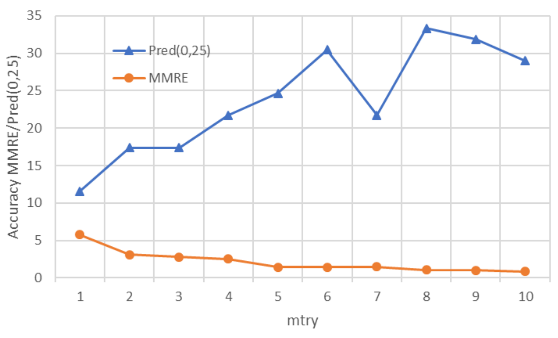
- ISBSG dataset:

Similarly, to the first empirical study for COCOMO, we can see from the Fig. 2-a that the accuracy of RF model is increasing, in general, with value of m_{try} in terms of Pred(0.25) until a certain value. The RF model with m_{try}=5 leads to better accuracy estimates (MMRE=1.29 and Pred(0.25)=40%).

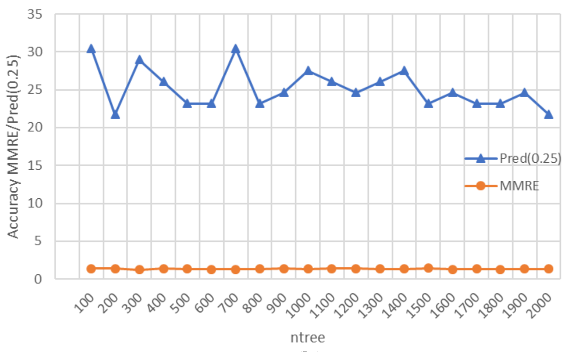
Looking at results of the second study shown in the Fig. 1-b, it is apparent that the accuracy of the estimates is not monotonically increasing as the number of trees, ntree, increases. For example, the best estimates were achieved when the value ntree is not too large (ntree=100 and when ntree=1100).

Table 3. Training and testing datasets

Datasets	# of projects in training dataset	# of projects in testing dataset
ISBSG (R8)	106	45
COCOMO	176	76
TUKUTUKU	37	16
DESHARNAIS	77	24
ALBRECHT	24	8

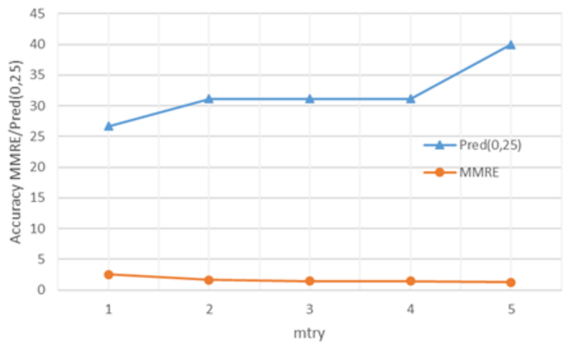


(a)

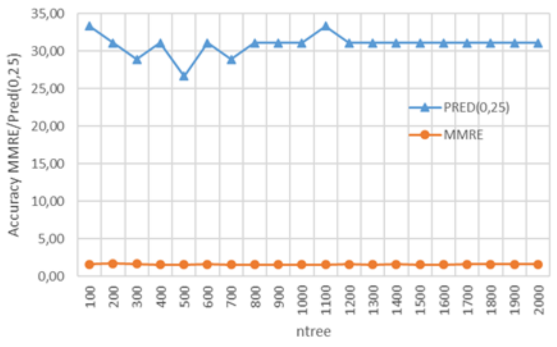


(b)

Figure. 1 Variation of the accuracy measures, MMRE and Pred(0.25), according to mtry and ntree values using COCOMO dataset: (a) Variation of MMRE and Pred(0.25) with respect to mtry and (b) Variation of MMRE and Pred(0.25) with respect to ntree

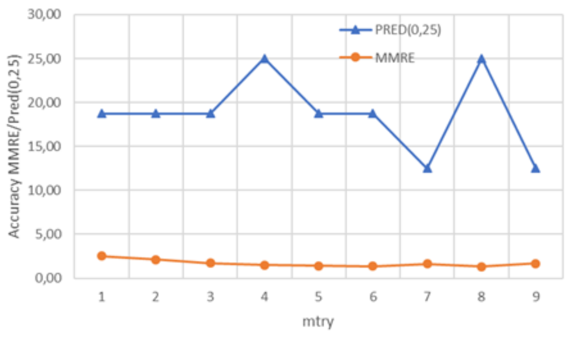


(a)

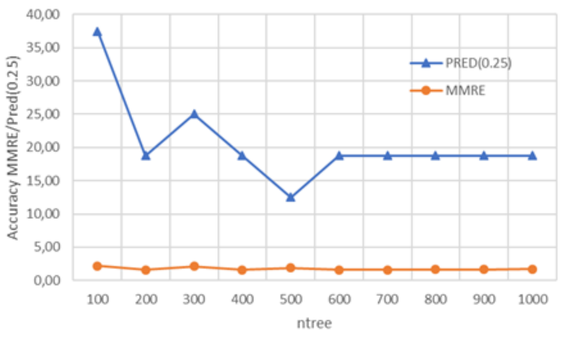


(b)

Figure. 2 Variation of the accuracy measures, MMRE and Pred(0.25), according to m_{try} and ntree values using ISBSG dataset: (a) Variation of MMRE and Pred(0.25) with respect to mtry and (b) Variation of MMRE and Pred(0.25) with respect to ntree



(a)



(b)

Figure. 3 Variation of the accuracy measures, MMRE and Pred(0.25), according to m_{try} and ntree values using Tukutuku dataset: (a) Variation of MMRE and Pred(0.25) with respect to mtry and (b) Variation of MMRE and Pred(0.25) with respect to ntree

- Tukutuku dataset:

As it can be seen from the results obtained in Fig. 2-a, the accuracy of RF model is increasing, in general, with value of m_{try} in terms of Pred(0.25) until a certain value. The RF model with m_{try}=5 yields to better accuracy estimates (MMRE=1.5 and Pred(0.25)=25%).

Regarding the second study, the results in Fig. 1-b show obviously that the accuracy of the estimates is not monotonically increasing as the number of trees, ntree, increases. As case in point, the best estimates were obtained when the value ntree is relatively small (ntree=100).

5.2 Comparison between random forest model and regression tree

Once the RF models were trained using the best values found for m_{try} and ntree. We compared the generalization capability of these developed models with regression trees using the testing sets. The evaluation was based on the MMRE, MdMRE and Pred(0.25) criteria. The results obtained are shown in Table 4.

It can be seen from the data in Table 4 that the random forest model outperforms the regression tree-based model in terms of all evaluation criteria when using ISBSG R8, COCOMO, Tukutuku and Albrecht datasets. For Desharnais dataset, the results obtained are similar in terms of Pred(0.25). Nevertheless, the RF model made a lower MMRE (0.42) and MdMRE(0.32) than RT model which generated 0.52 and 0.34 respectively.

From the chart in Fig. 5, we observed that both models yielded high values of MMREs especially for ISBSG, COCOMO and Tukutuku datasets. This is due to the fact that MMRE measure is extremely sensitive to individual predictions with excessively large MREs [44], which is, in turn, a result of the presence of outliers in these datasets (kurtosis >20 and MdMREs are much lower than MMRE as shown in Fig. 6).

The results, reported in Fig. 4, show that the best Pred(0.25) obtained is 51.31% when using RF model COCOMO dataset whereas the MMRE obtained is large 0.97. Therefore, it confirms again completely our assumptions about values of MMRE obtained.

To statistically check the results obtained, we

Table 4. Evaluation of the Regression Tree and Random Forest models in terms of MMRE, MdMRE and Pred(0.25) over three datasets using testing sets 30%

Datasets	**Regression Trees**			**Random Forests**		
	MMRE	MdMRE	Pred(0.25) (%)	MMRE	MdMRE	Pred(0.25) (%)
ISBSG (R8)	3.71	0,56	26.67	1,17	0,51	33.33
COCOMO	2.74	0.74	15.79	0.97	0.24	51.31
TUKUTUKU	1,81	0,89	18.75	0.98	0.60	31.25
DESHARNAIS	0.52	0.34	43.48	0.42	0.32	43.48
ALBRECHT	0.97	0.85	28.57	0.73	0.60	42.86

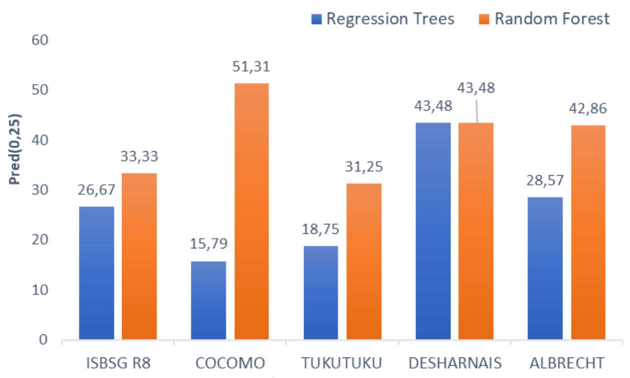


Figure. 4 Comparison of Pred(0.25) values expressed in (%) for the two SDEE models

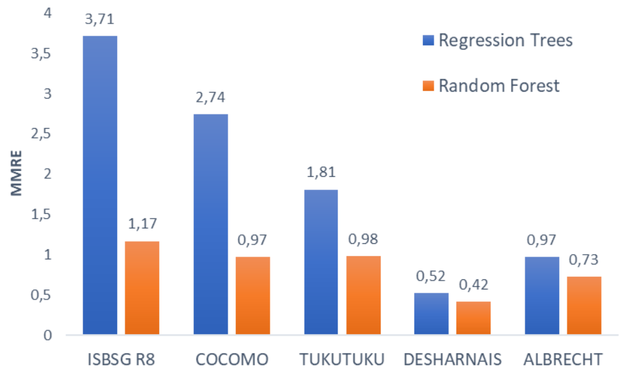


Figure. 5 Comparison of MMRE values for the two SDEE models

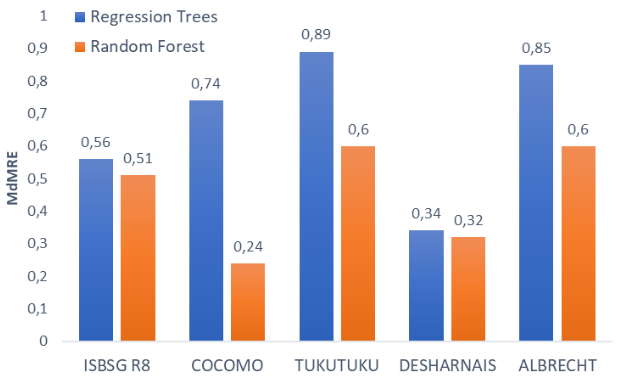


Figure. 6 Comparison of MdMRE values for the two SDEE models

Table 5. Statistical significance (Mann-Whitney U Test) over all datasets

SDEE Models	**Datasets**	**Mann-Whitney U Test**
Random Forest vs Regression Tree	ISBSG (R8)	**0.486**
	COCOMO	0.000
	TUKUTUKU	0.012
	DESHARNAIS	0.026
	ALBRECHT	0.039

used the Mann-Whitney statistical test based on absolute residuals, at the significance level of 0.05. The results of the statistical test are shown in Table 5.

As it can be seen from Table 5:

- For COCOMO, Tukutuku, Desharnais and Albrecht datasets: Random forest statistically outperformed regression tree.
- For ISBSG dataset: the p-value indicate that the difference Random Forest performance compared with the Regression Tree is not significant (p-value larger than 0.05).

5.3 Comparison between random forest models and other SDEE techniques

To further investigate the efficiency of random forest models in SDEE, we want to compare them against results reported in scientific articles: [11, 51-53]. Unfortunately, these papers' results used different validation methods and datasets version. In addition, they didn't describe results in terms of MdMRE. Nevertheless, we performed several empirical experiment designs employing the same validation methods and datasets, used in aforementioned papers, in order to make a fair comparison. We also noted that we used Kemerer dataset, which is a company-specific data from Kemerer's empirical work. This dataset contains data from 15 large completed business data-processing

Table 6. Comparison of random forest method with other methods on different datasets

Dataset	Technique	Validation Method	Pred (0.25)	MMRE
Desharnais	PSO based CBR [52]	LOOCV	37,2	0,69
	GA based SVR-RBF [51]	22%holdout (18 out of 81)	72,22	0,4
	GA based MLP [51]	22%holdout (18 out of 81)	72,22	0,31
	GA based M5P [51]	22%holdout (18 out of 81)	61,11	0,59
	Random Forest	22%holdout (17 out of 77)	***76,47***	***0,237***
ISBSG R8	Fuzzy Analogy [11]	LOOCV	24,32 (Pred(0.20)	1,77
	GA based Analogy [53]	3-folds CV	30	0,74
	Random Forest	LOOCV	***30.46 / 25.16*** *(Pred(0.20))*	***1.14***
Kemerer	GA based SVR-RBF [51]	LOOCV	66,67	0,37
	GA based MLP [51]	LOOCV	64,00	0,33
	GA based SVR-Linear [51]	LOOCV	60	0,44
	GA based M5P [51]	LOOCV	46,67	0,52
	Random Forest	LOOCV	***66,67***	***0,57***
Tukutuku	Fuzzy Analogy [11]	LOOCV	23,15 (Pred(0.20)	2,22
	Random Forest	LOOCV	***24.53*** (Pred(0.20))	***1.99***

projects of the same company. Each project is described by six input features: (i) programming language, (ii) hardware, (iii) duration, (iv) KSLOC, (v) AdjFP (adjusted function points), and (vi) RAWFP (raw function points).

Table 6 shows the results of the comparison of the RF models with other techniques on different employed datasets. As it can be seen, the proposed random forest model performed better than the four other techniques on Desharnais dataset in terms of Pred(0.25) and MMRE. Also, it outperformed Fuzzy Analogy on ISBSG R8 and Tukutuku datasets while it generated comparable results as Genetic Algorithm based Analogy on ISBSG dataset. Regarding the results obtained when using Kemerer dataset, the proposed random forest method generated the same results obtained by GA based SVR-RBF (support vector regression with RBF kernel optimized by genetic algorithm). Whereas, it outperformed the genetically optimized SVR-linear, MLP and M5P methods in terms of Pred(0.25). It worth noting that, using Kemerer dataset, RF generated a higher MMRE with respect to other techniques which is due mostly to the fact that the MMRE is sensitive to individual predictions with excessively large MREs (outliers). Finally, according to these results, we can conclude that random forest is very competitive method to the existing SDEE techniques and it can be used successfully to estimate software development effort.

6. Conclusion and perspectives

In this paper, we have empirically studied the use of random forest technique for software effort estimation. We first have investigated the impact of the number of trees and the number of attributes chosen to grow the tree on the estimation model. The results showed that the accuracy of RF model is sensitive to these parameters. In addition, this investigation has allowed us to optimize the RF model by choosing the best values for these two parameters. Next, the designed RF model was compared to the regression tree model using 30% hold-out validation method and via five datasets: COCOMO, ISBSG, Tukutuku, Desharnais and Albrecht. The evaluation criteria used were MMRE, MdMRE and Pred(0.25).

The results showed that the random forest model surpasses the regression tree-based model on all evaluation criteria. The robustness of the RF model was confirmed using the non-parametric Mann-Whitney U Test. In addition, the proposed model outperformed the genetically optimized version of MLP, M5P, Analogy, and SVR based methods. In the light of these results, we conclude that the random forest is a promising technique for software development effort estimation.

For future work, it would be interesting to improve further the accuracy of RF model using a powerful optimization method such as particle swarm optimization or genetic algorithm. Besides, further research might investigate the use of random forest as feature selection method in SDEE models.

References

[1] S. K. Sehra, Y. S. Brar, N. Kaur, and S. S. Sehra, “Research patterns and trends in software effort estimation”, *Information & Software Technology*, Vol. 91, pp. 1-21, 2017.

[2] M. Jorgensen, “The influence of selection bias on effort overruns in software development projects”, *Information & Software Technology*, Vol. 55, No. 9, pp. 1640-1650, 2013.

[3] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies", *IEEE Transactions on Software Engineering*, Vol. 33, No. 1, pp. 33-53, 2007.

[4] M. Jorgensen and T. Halkjelsvik, "The effects of request formats on judgment-based effort estimation", *Journal of Systems and Software*, Vol. 83, No. 1, pp. 29-36, 2010.

[5] M. Jorgensen, "Practical guidelines for expert-judgment-based software effort estimation", *IEEE Software*, Vol.22, No.3, pp.57-63, 2005.

[6] B. W. Boehm Clark, Horowitz, Brown, Reifer, Chulani, R. Madachy, and B. Steece, *Software Cost Estimation with Cocomo II with Cdrom*. Prentice Hall PTR, NJ, 2000.

[7] C. F. Kemerer, "An Empirical Validation of Software Cost Estimation Models", *Communications of the ACM*, Vol.30, No.5, pp.416-429, 1987.

[8] S. Basri, N. Kama, H. M. Sarkan, S. Adli, and F. Haneem, "An Algorithmic-Based Change Effort Estimation Model for Software Development", In: *Proc. of the 23rd Asia-Pacific Software Engineering Conf.*, Hamilton, New Zealand, pp.177-184, 2016.

[9] D. Nandal and O. P. Sangwan, "Software cost estimation by optimizing COCOMO model using hybrid BATGSA algorithm", *International Journal of Intelligent Engineering and Systems*, Vol.11, No.4, pp.250-263, 2018.

[10] A. Zakrani and A. Idri, "Applying radial basis function neural networks based on fuzzy clustering to estimate web applications effort", *International Review on Computers and Software*, Vol.5, No.5, pp.516-524, 2010.

[11] F. A. Amazal, A. Idri, and A. Abran, "Software development effort estimation using classical and fuzzy analogy: A cross-validation comparative study", *International Journal of Computational Intelligence and Applications*, Vol.13, No.3, 2014.

[12] A. Idri, A. Zakrani, M. Elkoutbi, and A. Abran, "Fuzzy radial basis function neural networks for web applications cost estimation", In: *Proc. of the 4th International Conf. on Innovations in Information Technology*, Dubai, UAE, pp.576-580, 2008.

[13] T. R. Benala and R. Mall, "DABE: Differential evolution in analogy-based software development effort estimation", *Swarm and Evolutionary Computation*, Vol.38, pp.158-172, 2018.

[14] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models", *Information and Software Technology*, Vol.54, No.1, pp.41-59, 2012.

[15] S. G. MacDonell and M. J. Shepperd, "Combining techniques to optimize effort predictions in software project management", *Journal of Systems and Software*, Vol.66, No.2, pp.91-98, 2003.

[16] M. Jorgensen and M. J. Shepperd, "A Systematic Review of Software Development Cost Estimation Studies", *IEEE Trans. Software Eng.*, Vol.33, No.1, pp.33-53, 2007.

[17] A. Idri, M. Hosni, and A. Abran, "Systematic literature review of ensemble effort estimation", *Journal of Systems and Software*, Vol.118, pp.151-175, 2016.

[18] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, 2nd Edition. Wiley, 2001.

[19] W. Pedrycz and Z. A. Sosnowski, "Genetically optimized fuzzy decision trees", *IEEE Trans. Systems, Man, and Cybernetics, Part B*, Vol. 35, No.3, pp.633-641, 2005.

[20] F. E. B. Otero, A. A. Freitas, and C. G. Johnson, "Inducing decision trees with an ant colony optimization algorithm", *Appl. Soft Comput.*, Vol.12, No.11, pp.3615-3626, 2012.

[21] S. K. Shukla and M. K. Tiwari, "Soft decision trees: A genetically optimized cluster oriented approach", *Expert Syst. Appl.*, Vol.36, No.1, pp.551-563, 2009.

[22] 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: *Proc. of the 3rd Int. Conf. on Communications and Information Technology*, Beirut, pp.220-224, 2013.

[23] R. W. Selby and A. A. Porter, "Learning from Examples: Generation and Evaluation of Decision Trees for Software Resource Analysis", *IEEE Trans. Software Eng.*, Vol.14, No.12, pp.1743-1757, 1988.

[24] A. A. Porter and R. W. Selby, "Evaluating techniques for generating metric-based classification trees", J*ournal of Systems and Software*, Vol.12, No.3, pp.209-218, 1990.

[25] K. Srinivasan and D. Fisher, "Machine Learning Approaches to Estimating Software Development Effort", *IEEE Trans. Softw. Eng.*, Vol.21, No.2, pp.126-137, 1995.

[26] A. S. Andreou and E. Papatheocharous, "Software cost estimation using fuzzy decision trees", In: *Proc. of the 23rd IEEE/ACM International Conf. on Automated Software Engineering*, pp.371-374, 2008.

[27] S. Elyassami and A. Idri, "Applying Fuzzy ID3

Decision Tree for Software Effort Estimation", *International Journal of Computer Science Issues*, Vol.8, No.1, pp.131-138, 2011.

[28] M. Azzeh, "Software effort estimation based on optimized model tree", In: *Proc. of the 7th International Conf. on Predictive Models in Software Engineering*, pp.1-8, 2011.

[29] M. P. Basgalupp, R. C. Barros, T. S. Da Silva, and A. C. P. L. F. De Carvalho, "Software effort prediction: A hyper-heuristic decision-tree based approach", In: *Proc. of the 28th Annual ACM Symposium on Applied Computing*, pp.1109-1116, 2013.

[30] M. O. Elish, "Improved estimation of software project effort using multiple additive regression trees", *Expert Systems with Applications*, Vol.36, No.7, pp.10774-10778, 2009.

[31]A. B. Nassif, L. F. Capretz, D. Ho, and M. Azzeh, "A treeboost model for software effort estimation based on use case points", In: *Proc. of the 11th IEEE International Conf. on Machine Learning and Applications*, Vol. 2, pp.314-319, 2012.

[32] S. M. Satapathy, B. P. Acharya, and S. K. Rath, "Early stage software effort estimation using random forest technique based on use case points", *IET Software*, Vol.10, No.1, pp.10-17, 2016.

[33] E. Mendes and B. A. Kitchenham, "Further Comparison of Cross-Company and Within-Company Effort Estimation Models for Web Applications", In: *Proc. of the 10th International Symposium on Software Metrics*, pp.348-357, 2004.

[34] ISBSG, *International Software Benchmarking Standards Group, Data Release 8 Repositor*y, http://www.isbsg.org. , 2003.

[35]B. W. Boehm, *Software Engineering Economics*. Prentice Hall PTR, pp.768, 1981.

[36] A. Idri, A. Abran, and T. M. Khoshgoftaar, "Estimating software project effort by analogy based on linguistic values", In: *Proc. of the 8th IEEE Symposium on Software Metrics*, pp.21-30, 2002.

[37] A. J. Albrecht and J. E. Gaffney, Jr., "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation", *IEEE Transactions on Softw. Eng.*, Vol. SE-9, No.6, pp.639-648, 1983.

[38] Y. F. Li, M. Xie, and T. N. Goh, "A study of the non-linear adjustment for analogy based software cost estimation", *Empirical Software Engineering*, Vol.14, No.6, pp.603-643, 2009.

[39] A. Idri and I. Abnane, "Fuzzy Analogy Based Effort Estimation: An Empirical Comparative Study", In: *Proc. of the 17th IEEE International Conf. on Computer and Information Technology*, pp.114-121, 2017.

[40] E. Kocaguneli and T. Menzies, "Software effort models should be assessed via leave-one-out validation", *Journal of Systems and Software*, Vol.86, Vo.7, pp.1879-1890, 2013.

[41] J.-M. Desharnais, *Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction*, Master thesis, University of Montreal, 1989.

[42] P. J. Curran, S. G. West, and J. F. Finch, "The robustness of test statistics to non-normality and specification error in confirmatory factor analysis", *Psychological Methods*, pp.16-29, 1996.

[43] M. Azzeh, A. B. Nassif, and L. L. Minku, "An empirical evaluation of ensemble adjustment methods for analogy-based effort estimation", *Journal of Systems and Software*, Vol.103, pp.36-52, 2015.

[44] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A Simulation Study of the Model Evaluation Criterion MMRE", *IEEE Trans. Softw. Eng.*, Vol.29, No.11, pp.985-995, 2003.

[45] A. Idri, I. Abnane, and A. Abran, "Evaluating Pred(p) and standardized accuracy criteria in software development effort estimation", *Journal of Software: Evolution and Process*, Vol. 30, No. 4, 2018.

[46] J. Demsar, "Statistical Comparisons of Classifiers over Multiple Data Sets", *Journal of Machine Learning Research*, Vol.7, pp.1-30, 2006.

[47] L. Breiman, "Random Forests", *Machine Learning*, Vol.45, No.1, pp.5-32, 2001.

[48] A. Liaw and M. Wiener, "Classification and Regression by randomForest", *R News*, Vol. 2, No.3, pp.18-22, 2002.

[49] B. Larivière and D. V. d. Poel, "Predicting customer retention and profitability by using random forests and regression forests techniques", *Expert Syst. Appl.*, Vol.29, No.2, pp.472-484, 2005.

[50] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Wadsworth, 1984.

[51] A. L. I. Oliveira, P. L. Braga, R. M. F. Lima, and M. L. Cornélio, "GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation", *Information & Software Technol.*, Vol.52, No.11, pp.1155-1166, 2010.

[52] D. Wu, J. Li, and C. Bao, "Case-based reasoning with optimized weight derived by particle

swarm optimization for software effort estimation”, *Soft Computing*, pp.1-12, 2017.

[53] S. J. Huang and N. H. Chiu, "Optimization of analogy weights by genetic algorithm for software effort estimation”, *Inf. and Softw. Technol.*, Vol. 48, No. 11, pp. 1034-1045, 2006.