Journal of Software Engineering and Applications, 2020, 13, 143-160
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Software Effort Prediction Using Ensemble Learning Methods

Omar H. Alhazmi, Mohammed Zubair Khan

Department of Computer Science, College of Computer Science and Engineering Taibah University, Madinah, KSA
Email: ohhazmi@taibahu.edu.sa, zubair.762001@gmail.com

How to cite this paper: Alhazmi, O.H. and Khan, M.Z. (2020) Software Effort Prediction Using Ensemble Learning Methods. *Journal of Software Engineering and Applications*, **13**, 143-160.
https://doi.org/10.4236/jsea.2020.137010

Received: June 4, 2020
Accepted: July 14, 2020
Published: July 17, 2020

Abstract

Software Cost Estimation (SCE) is an essential requirement in producing software these days. Genuine accurate estimation requires cost-and-efforts factors in delivering software by utilizing algorithmic or Ensemble Learning Methods (ELMs). Effort is estimated in terms of individual months and length. Overestimation as well as underestimation of efforts can adversely affect software development. Hence, it is the responsibility of software development managers to estimate the cost using the best possible techniques. The predominant cost for any product is the expense of figuring effort. Subsequently, effort estimation is exceptionally pivotal and there is a constant need to improve its accuracy. Fortunately, several efforts estimation models are available; however, it is difficult to determine which model is more accurate on what dataset. Hence, we use ensemble learning bagging with base learner Linear regression, SMOReg, MLP, random forest, REPTree, and M5Rule. We also implemented the feature selection algorithm to examine the effect of feature selection algorithm BestFit and Genetic Algorithm. The dataset is based on 499 projects known as China. The results show that the Mean Magnitude Relative error of Bagging M5 rule with Genetic Algorithm as Feature Selection is 10%, which makes it better than other algorithms.

Keywords

Software Cost Estimation (SCE), Ensemble Learning, Bagging, Linear Regression, SMOReg, REPTree, M5 Rule

1. Introduction

For software developers the quality of a software product is vital, and software cost estimation efforts help developers to maintain good quality. Software cost estimation in terms of the persons-months and time to complete the project is crucial. Though software cost estimation plays a vital role in the field of software development, there have been minor developments in this area in the last few decades. The most important reason for the failure of a project is poor cost estimation. Even though there are many efforts models available, novel methods for improving the accuracy of projects are still needed. So, the development of a software efforts prediction model is motivation to estimate software efforts as accurately as possible. Software cost estimation predictions are used to forecast the cost of software. Machine Learning methods use the historical dataset for predicting the actual cost for future software. The fundamental purpose for using Machine Learning systems is to become familiar with the inalienable examples of feature value and their relations with venture endeavours (project efforts) and anticipate the efforts for new software projects.

The ML approaches have been utilized as a commendation for both master judgment and algorithmic models in the past decade. These methodologies incorporate Artificial Neural Networks (ANN), Fuzzy rationale, bagging, boosting, decision trees, Support Vector Machine (SVM) and so on. The upside of these methodologies is that they show the mind-boggling connection between efforts and free factor. It is utilized for those troublesome issues where an outcome must be gained from authentic historical information. In the literature many machine learning approaches have been found, though it is very difficult to say which approach is better.

Software efforts estimation plays a very vital job in calculating the cost for developing the software project. The understanding and controlling of basic factors that influence programming cost is an exceptionally fundamental job in software project management. Software measurements are the software product measures and qualities. Since software estimations are basic in software engineering, there have been numerous investigations over the most recent four decades to give a thorough view of software's complex nature and to utilize it in software cost estimation and software examination. Despite the fact that the principal software measurements (metrics) book were published in 1976 [1], the historical backdrop of software measurements explorations dates to the 1960s, when the lines of code (LOC) metric was utilized to quantify the profitability of the developer and software complexity and quality. LOC was utilized as a principle key in efforts prediction for some forecast models, for example, [2] [3].

In the mid-1970s, the enthusiasm for software design complexity expanded when diagram hypothetical unpredictability was discussed by McCabe in [4]. He built up a scientific strategy for program modularization. A few meanings of the graph hypothesis were utilized so as to measure and control the quantity of ways through a software program known as the Cyclomatic Complexity metric. At that point this metric had been utilized for complexity estimations rather than size metrics. In 1984, Basili and Perricone [5] found a connection between McCabe's Cyclomatic Complexity and module sizes. They found that enormous modules have high intricacy.

Fei, Zhi and Chao [6] proposed an enhancement for the Halstead complex nature measurements. They added weights to the Halstead metrics. They gave various operators and operands various weights. Six object-oriented design metrics items were created and assessed by Chidamber and Kemerer in 1994 [7]. These items are called CK measurements. The CK measurements that came about because of Chidamber and Kemerer are weighted methods per class (WMC), depth of inheritance tree (DIT), number of children (NOC), coupling between object classes (CBO), response for a class (RFC), and lack of cohesion in methods (LCOM).

As per Smith, Hale and Parish [8], 4 task factors—force, concurrency, fragmentation and team size—have been considered for their effect on software efforts developments. Every one of these elements improved the estimations of the middle of the road COCOMO I model. These elements alongside un-balanced capacity focuses help in developing a superior effort estimation model which brings about improved predictive capacity when contrasted with COCOMO model.

Tosun, Turhan and Bener [9] proposed another novel methodology for improving the estimation precision with the assistance of another element weight assignment algorithm which gives better outcomes when contrasted with past research. Here a factual procedure called Principal Component Analysis (PCA) was used to actualize the two weighted task heuristics. Pahariya, Ravi, Carr and Vasu [10] proposed new computational knowledge sequential hybrid architectures including programming and Group Method of Data Handling (GMDH). This incorporates information mining strategies, for example, Multi-Layer Regression (MLR), Radial Basis Function (RBF), etc. [10]. Different investigations of ANN models for anticipating SCE are [10]-[17]. Andreou and Papatheocharous [18] utilized Fuzzy Decision Trees (FDTs) for foreseeing required efforts and code size in cost estimation as though solid proof about those fluffy changes of cost drivers added to improving the forecast procedure. More researches on this topic can be found in [19]-[25]. Reddy and Raju [26] improved fuzzy methodology for software efforts of the COCOMO utilizing the Gaussian membership function, which performs superior to the trapezoidal capacity to display cost drivers. In this paper, we describe COCOMO models, then explain the roles and application of machine learning techniques like Linear Regression, Support Vector Machine (SMOReg), Neural networks, MRules 5, REPTree and Random Forest. Then we apply ensemble Learning based on these classifiers. However, we compare the outcomes of these methods with results in given actual and estimated efforts.

As we have seen, software vaults or datasets are generally used to acquire information on which efforts estimation is finished. Yet software stores contain data from heterogeneous ventures. Customary utilization of regression equations to derive a single mathematical model results in poor performance [27]. Gallogo [27] utilized Data clustering to tackle this issue. In this study, the models are predicted and validated using statistics and ensemble Learning methods. Comparison with previous research is also done. The result shows that Bagging M5 rule with Genetic algorithm for feature selection shows MMRE 10%. Here we try to answer the following research questions.

RQ1. What is the impact of BESTFIT and GENETIC Algorithms for Feature Selection for software efforts Prediction on two datasets when the performance is measured using four metrics, MMRE (mean magnitude of relative error) and prediction at levels 0.25, 0.50 and 0.75 respectively?

RQ2. What is the performance of ensemble Learning Techniques? We determine which ML systems give the best and worst outcomes relating to each dataset explored in the investigation.

2. Background

2.1. Dataset

In this study we have taken the China dataset for software cost estimation from the Promise Data repository [28]. The China Dataset is comprised of 19 features: 18 autonomous variables and 1 ward variable. It has 499 instances corresponding to 499 projects. The clear insights of Chinese informational collection are in the appendix in Table 1. A set of autonomous variables chooses the estimation of the needy variable. The needy variable is efforts right now and the independent factors might be removed, as they may have little impact on predicting the

Table 1. China data set statistics.

Serial Number	Variable	Min	Max	Mean	Standard Deviation
1.	ID	1	499	250	144
2.	AFP	9	17518	487	1059
3.	Input	0	9404	167	486
4.	Output	0	2455	114	221
5.	Enquiry	0	952	62	105
6.	File	0	2955	91	210
7.	Interface	0	1572	24	85
8.	Added	0	13,580	360	830
9.	Changed	0	5193	85	291
10.	Deleted	0	2657	12	124
11.	PDR_AFP	0.3	83.8	12	12
12.	NPDR_AFP	0.4	101	13	14
13.	NPDU_UFP	0.4	108	14	15
14.	Resource	1	4	1	1
15	PDR_UFP	0.3	83.8	12	12
16.	Dev. Type	0	0	0	0
17.	Duration	1	84	9	7
18.	N_effort	31	54620	4278	7071
19.	Efforts	26	54260	3921	6481

efforts, consequently making the model much less difficult and productive. It has been seen from the China informational index that independent variables ID and Dev.Type do not play any role in deciding the value of effort. Consequently, variables ID and Dev are autonomous. Here we perform Cross-validation, a standard evaluation method that is an orderly method for running repeated percentage splits. It consists of partitioning a dataset into 10 pieces ("folds"); at that point hold out each piece for testing and train on the 9 staying together. This gives 10 assessment results, which are the average.

2.2. Feature Selection Method

There are different strategies utilized for diminishing information dimensionality. We have utilized the Feature sub-selection procedure given in the WEKA tool [29] to diminish the quantity of the independent variable. Applying CfsSubsetEval with BestFit feature selection method reduces 19 features to 7 features. When Genetic Algorithm is used for feature selection, 19 features are reduced to 9. The best combination of independent variables was searching through all possible combinations of variables. The dependent variable is Efforts. Software development efforts are defined as the work done by the product provider from detail until delivery estimated as far as hours.

2.3. Performance Measures

Mean Magnitude of relative error (MMRE) (or mean absolute relative error) currently utilizes the most effective and standard measures for estimation exactness, for example, MMRE and PRED at power levels 0.25, 0.50 and 0.75, respectively.

In order to assess capability, we use a common criterion called Mean Magnitude of Relative Error (MMRE) [30] [31] [32] [33].

$$\text{MMRE} = \frac{1}{k}\sum\nolimits_{i=1}^{k}\frac{\left|E_i - A_i\right|}{A_i} \tag{1}$$

here, E_i represents the estimated value for a data point, A_i represents the actual value of each data point, and k is the total number of data points. Here, Predict(A) is calculated as follows:

$$\text{Predict}\left(A\right) = \frac{m}{k} \tag{2}$$

Mean Relative Error (MRE) is denoted by (m) and includes the values when data points have less than or equal to A error. It is common to consider (25%) as the reference value [32].

2.3.1. Relative Absolute Error

Relative Absolute Error (RAE) calculates the accuracy of a predictive model. RAE can be used in machine learning. Furthermore, RAE is expressed as the ratio; it computes the mean error (residual) of errors produced by a trivial or naive model. The model is considered non-trivial if the result is less than 1. This is the model for a data set (k):

$$R_k = \frac{\sum_{i=1}^{n} \left| E_{ki} - D_i \right|}{\sum_{i=1}^{n} \left| D_i - \bar{D} \right|} \quad (3)$$

where E_i's is prediction, D_i's is actual values, and Rae is the measure of forecast accuracy. $\bar{D}$ is the mean of D_i's; n is the size of the dataset (in data points) [32] [33] [34].

2.3.2. Root Relative Squared Error

Root relative squared error (RRSE) takes the average errors, squares them and normalizes the average. Then, in order to maintain the error to the same dimension, the square root is calculated. In the following equation, E_i of an individual model i is shown:

$$R_k = \sqrt{\frac{\sum_{j=1}^{n} \left(E_{ki} - D_i \right)^2}{\sum_{j=1}^{n} \left(D_j - \bar{D} \right)^2}} \quad (4)$$

where E_i's is prediction, D_i's is actual values, Rae is the measure of forecast accuracy, $\bar{D}$ is the mean of D_i's, and n is the size of the dataset (in data points) [32] [33] [34].

2.3.3. Relative Absolute Error

Relative Absolute Error (RAE) calculates the accuracy of a predictive model. RAE can be used in machine learning. Furthermore, RAE is expressed as a ratio that computes the mean error (residual) of errors produced by a trivial or naive model. The model is considered non-trivial if the result is less than 1. This is the model for a data set (k):

$$R_k = \frac{\sum_{i=1}^{n} \left| E_{ki} - D_i \right|}{\sum_{i=1}^{n} \left| D_i - \bar{D} \right|} \quad (5)$$

where E_i's is prediction, D_i's is actual values, Rae is the measure of forecast accuracy. $\bar{D}$ is the mean of D_i's, and n is the size of the dataset in data points.

2.3.4. Root Relative Squared Error

Root relative squared error (RRSE) [32] [33] [34] takes the average errors, squares them and normalizes the average. Then, in order to maintain the error to the same dimension, the square root is calculated. In the following equation, the E_i of an individual model i is shown:

$$R_k = \sqrt{\frac{\sum_{j=1}^{n} \left(E_{ki} - D_i \right)^2}{\sum_{j=1}^{n} \left(D_j - \bar{D} \right)^2}} \quad (6)$$

where E_i's is prediction, D_i's is actual values, and Rae is the measure of forecast accuracy. $\bar{D}$ is the mean of D_i's; n is the size of dataset in data points.

3. Machine Learning Techniques

Right now, we are utilizing machine learning methods to predict effort. We have used ensemble learning method bagging with base Learner Linear Regression, Support Vector Machine, Neural Network (MLP), MRules 5, REPTree, and Random Forest.

3.1. Linear Regression

Linear regression (LR) is widely used for predictive analysis. Basically, LR measures the degree to which variables are linearly related. The formula for linear regression is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{7}$$

Furthermore, multiple linear regression (MLR) is an empirical model that utilizes data from past results. According to Liung and Fan MLR is an empirical model that utilizes data from past results in order to measure current results [33].

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_n x_{i.n} + \epsilon \tag{8}$$

where y_i is the dependent variable, x_i is the explanatory variable, β_0 is the y-intercept (constant), β_p is slope coefficient for each explanatory variable, and ϵ is the model's residuals (errors).

3.2. Multilayer Perception (MLP)

A multilayer perceptron (MLP) is used for regression; however, in MLP an intermediate layer (hidden layer) is used instead of using input as feed. MLP has several layers of nodes; it should have at least three layers of nodes: an input, a hidden layer and an output layer. MLP is a logistic model that can be in various depth level based on the number of intermediate layers, as shown in these equations: $y(v_i) = \tanh(v_i)$ and $y(v_i) = 1 + e^{-v_i}$. Here, y(v_i) is the output of the i^{th} node (neuron), and v_i is the weighted sum of the input connections

3.3. Sequential Minimal Optimization Regression

Sequential minimal optimization (SMO) is useful when applied to solve quadratic programming (QP) problem that comes out as a result of applying the training of support-vector machines. When looking at a binary classification problem with a dataset pairs (x_1, y_1), (x_2, y_2), ..., (x_n, y_n), where x_i is an input vector and y_i can be a label of either (−1) or (+1). A soft-margin support vector machine is then trained by solving a quadratic programming problem, which is expressed in form:

$$max_\alpha = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j K(x_i, x_j) \alpha_i \alpha_j \tag{9}$$

where $K(x_i, x_j)$ is the kernel function; α's are Lagrange multipliers, (Platt 1998). However, it must satisfy the two conditions $0 \le \alpha_i \le C$ (for $i = 1, 2, \cdots, n$) and $\sum_{i=1}^{n} y_i \alpha_i = 0$ (here C is a support-vector machine hyperparameter).

3.4. REPTree

REPTree is based on regression tree logic. It basically creates a number of trees;

the process is done on different iterations. Then, REPTree selects the best one from all generated trees. The selection will be considered representative. Then the mean square error is applied on the trees' prediction [35].

3.5. Decision Tree

Decision tree is a regression methodology. Basically, it provides an easily understandable modelling technique. Moreover, even if there are some imperfections in the data, such as missing values, it can predict patterns to overcome such issues. Decision tree is an approach that uses continuous recursive partitioning until it reaches classification of a dataset data. It can be of two types, breadth first (BF) or depth first (DF). In both a greedy algorithm is typically applied. One drawback of decision trees is overfitting of data samples [35].

3.6. Bagging

Bagging is a method to stabilize the accuracy of machine learning. Bagging helps reduce the issue of overfitting found in some regression methodologies such as decision tree. Hence, it is a method that iteratively samples from a certain data set according to a rectangular probability distribution with substitution [32] [35] [36]. In Bagging, each sample has the exact same size as the original data. Here, we should indicate that in sampling with replacement there is a chance that some dataset instances may never get a chance to be selected while others can be selected multiple times.

3.7. Random Decision Forest (RDF)

Random decision forest (RDF) was first introduced by Ho [35]. Random decision forest (RDF) corrects the problem of overfitting found in decision trees. Moreover, RDF is a learning scheme for regression; in RDF multiple decision trees are built. During the training of a model decision trees are built, and eventually an average prediction is reached. Moreover, the RDF training process for random forests uses the bagging technique.

4. Experiment Design

In the literature we have found some limitations and from our experiment, we discovered that most researchers ignore the steps in their Pre-processing. Before pre-processing remove the missing and noisy data from the dataset. Besides these limitations, attribute selection is another important limitation that directly affects the memory use and also affects the results. So, to overcome these limitations we follow these basic steps (see **Figure 1**).

- ○ Input as a dataset
- ○ Preprocess the data:
- ▪ by filtering out noisy and missing data,
- ▪ by conversion,
- ▪ by removing outlier.

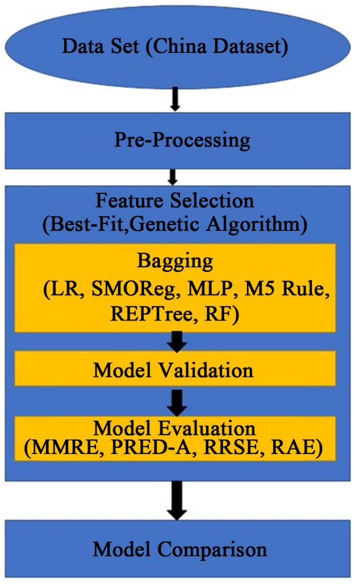


Figure 1. Experiment setup for software cost efforts prediction.

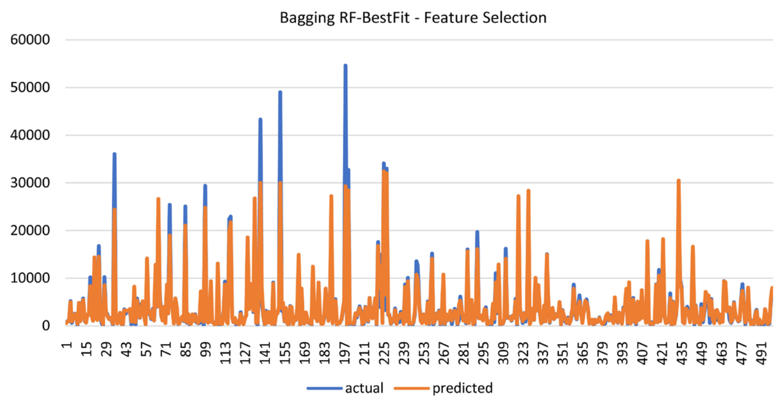
- Apply Feature Selection Method (CfsSubsetEval: 1) Genetic Algorithm, 2) BestFit)
- Ensemble Learning Methods
- Computing the results

5. Results

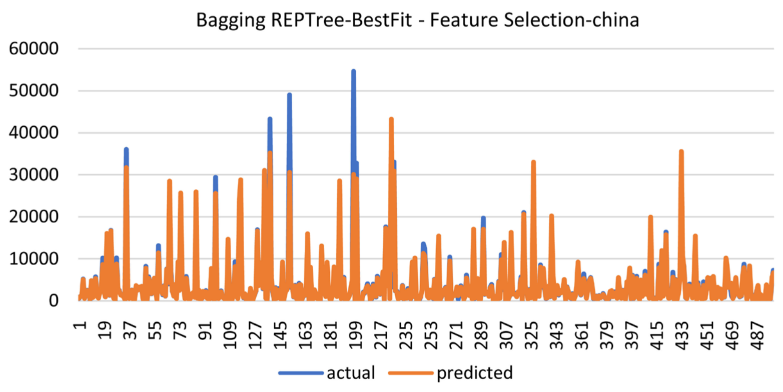
Figure 2(a) shows the predicted effort compared to the actual effort using the China dataset using best fit algorithm and RF bagging. Results show a strong correlation between predicted and actual. Figure 2(b) also shows the same but by applying REPTree. Figure 2(c) shows the results when MRule 5 is applied. Next, Figure 2(d) shows the result when using LR Bagging, Figure 2(e) when using SMOReg and Figure 2(f) when applying MLP.

Figures 3(a)-(f) show the same prediction experiment when the same six techniques are applied when genetic algorithms are applied.

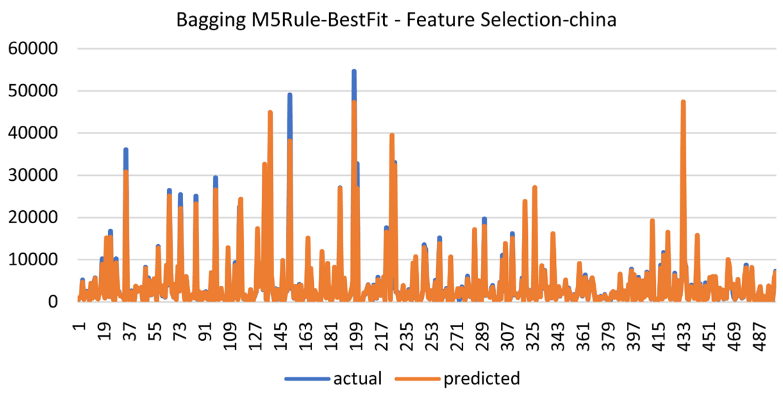
The results are shown in Figure 4, and in Table 2 and Table 3. Our results of effort estimation model predicted using bagging M5 rule with genetic Algorithm Feature selection method was better than all the other 12 methods examined in our study. This shows that bagging M5 rule has very good results for software efforts. The MMRE is 10% while Pred (25), Pred (50) and Pred (75) have 97%, 98% and 99%, respectively. This shows that performance of Bagging M5 rule is excellent, even when we compare our results in existing research. Table 2 also shows that the performance of Ensemble learning is best among all. Table 3 shows that the correlation of actual and predicted values is also very high. The charts in Figure 5 and Figure 6, for the real qualities and the qualities as predicted by the specific model, appear on the Y-axis and compare to the 499 Projects. The "“blue" band displays the bend for the real values, though the "red" band introduces the band for the predicted qualities. The closer the real and


Bagging RF-BestFit - Feature Selection
60000
50000
40000
30000
20000
10000
0
actual
predicted

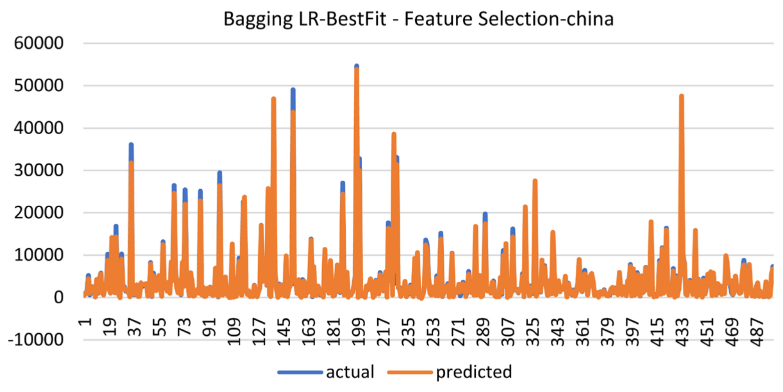
(a)


Bagging REPTree-BestFit - Feature Selection-china
60000
50000
40000
30000
20000
10000
0
actual
predicted

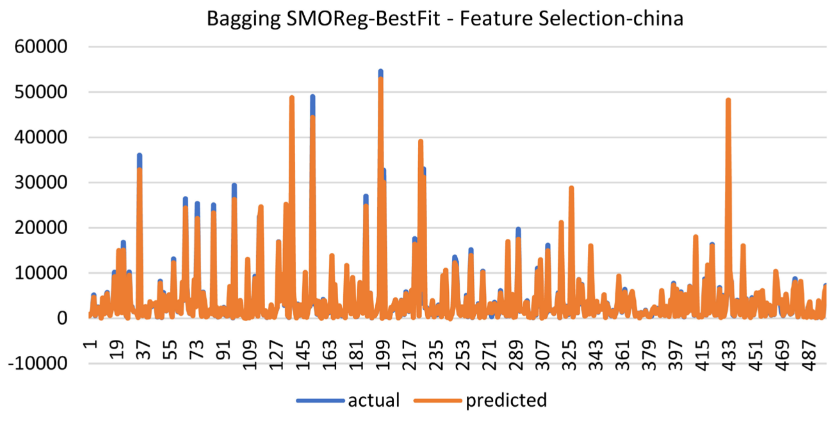
(b)


Bagging M5Rule-BestFit - Feature Selection-china
60000
50000
40000
30000
20000
10000
0
actual
predicted

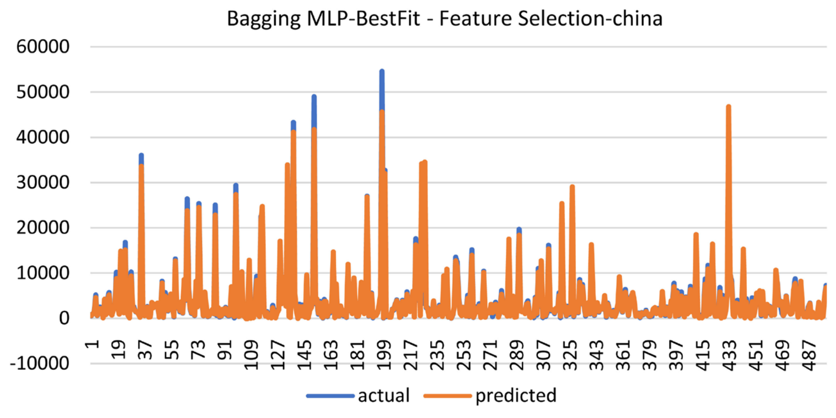
(c)


Bagging LR-BestFit - Feature Selection-china
60000
50000
40000
30000
20000
10000
0
-10000
actual
predicted

(d)



(e)

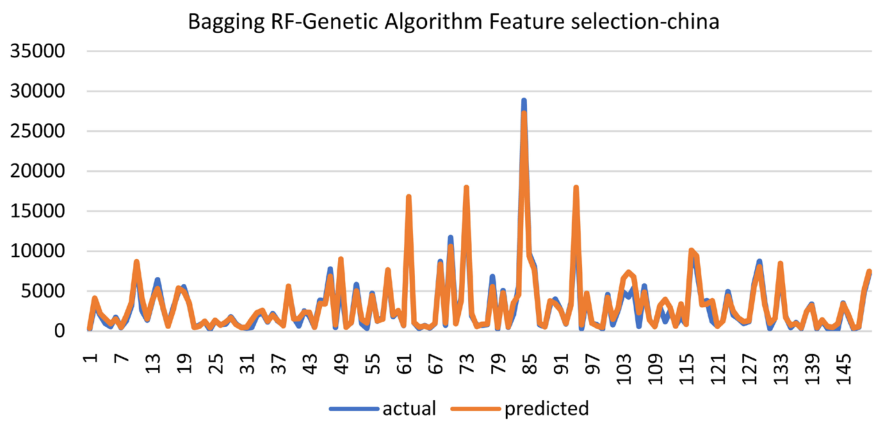


(f)

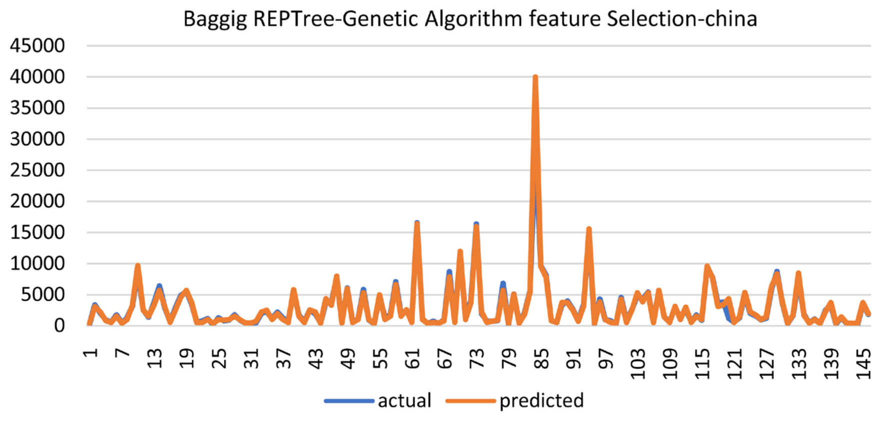
Figure 2. Bestfit feature selection using (a) Bagging RF; (b) Bagging REPTree; (c) M5Rule; (d) Bagging LR; (e) Bagging SMOReg; (f) Bagging MLP.

Table 2. Performance measures by bagging.

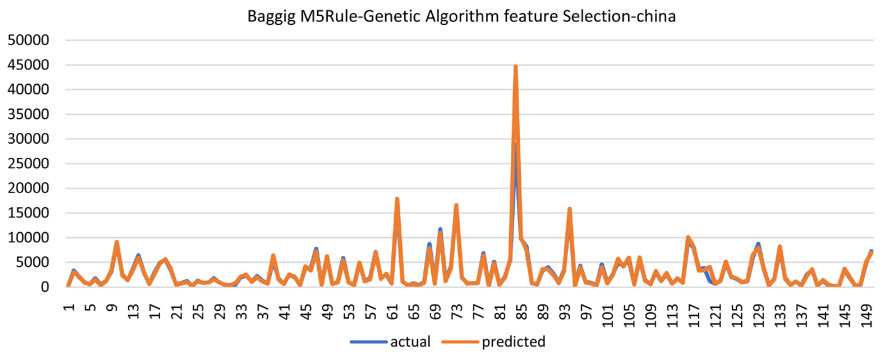
Feature Selection Algorithm	ML Algorithm	MMRE	PRED 25	PRED 50	PRED 75
BEST FIT	Bagging LR	0.147558	0.88	0.946667	0.98
	Bagging SMOReg	0.126655	0.911824	0.963928	0.983968
	Bagging MLP	0.176172	0.8	0.906667	0.96
	Bagging MRules5	0.10263	0.97333	0.98	0.99333
	Bagging REPTree	0.153349	0.88	0.95333	0.98
	Bagging RF	0.251015	0.78	0.88	0.93333
Genetic Algorithm	Bagging RF	0.318897	0.74	0.82	0.9
	Bagging REPTree	0.152987	0.88	0.95333	0.98
	Bagging M5 Rule	0.10006	0.97333	0.98	0.99333
	Bagging LR	0.193831	0.8133	0.9133	0.946667
	Bagging MLP	0.157302	0.8733	0.966667	0.966667
	Bagging SMOReg	0.128885	0.906667	0.966667	0.973333


Bagging RF-Genetic Algorithm Feature selection-china
actual
predicted

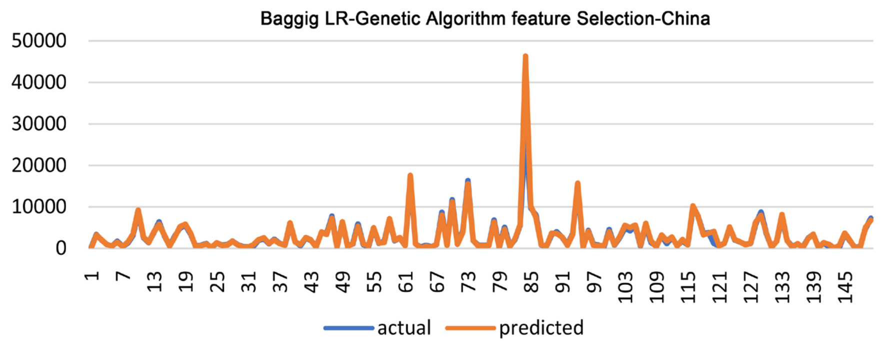
(a)


Baggig REPTree-Genetic Algorithm feature Selection-china
actual
predicted

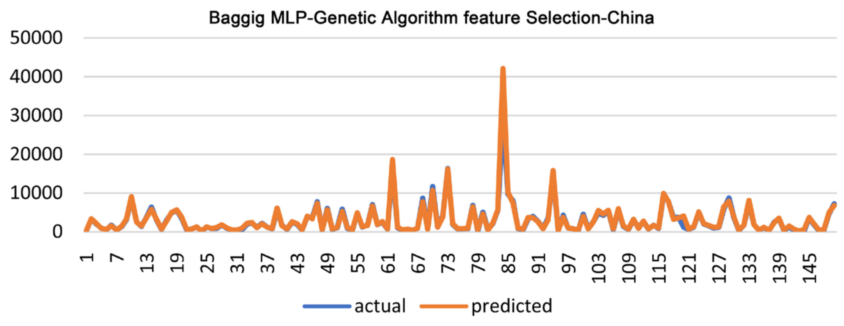
(b)


Baggig M5Rule-Genetic Algorithm feature Selection-china
actual
predicted

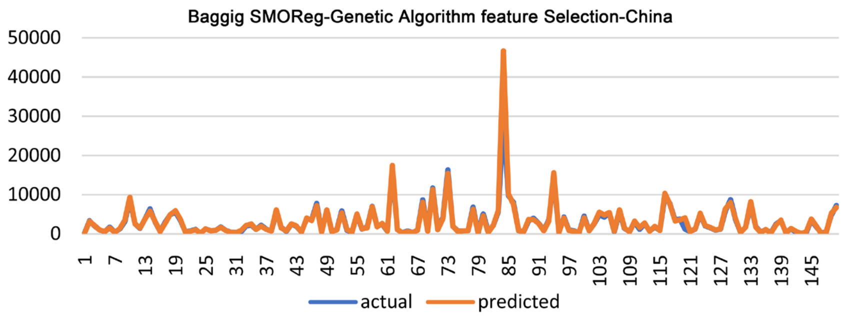
(c)


Baggig LR-Genetic Algorithm feature Selection-China
actual
predicted

(d)



(e)



(f)

Figure 3. Genetic feature selection using (a) Bagging RF; (b) Bagging REPTree; (c) M5Rule; (d) Bagging LR; (e) Bagging MLP; (f) SMOReg.

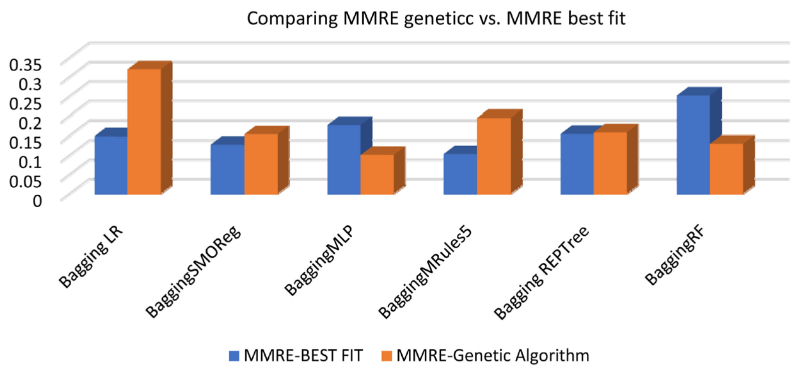


Figure 4. Comparing MMRE genetic vs. MMRE best fit.

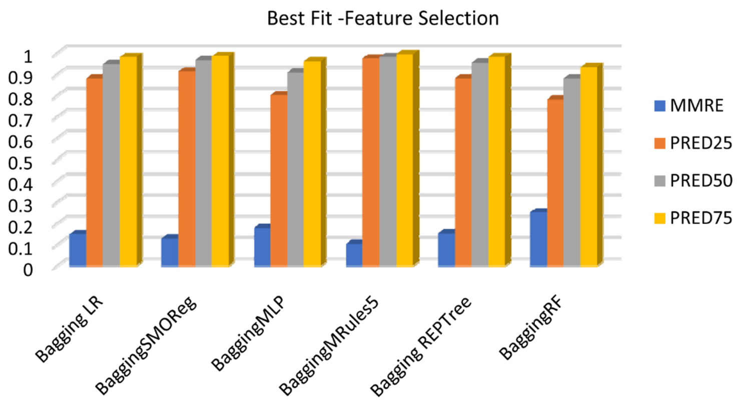


Figure 5. Best fit feature selection comparison.

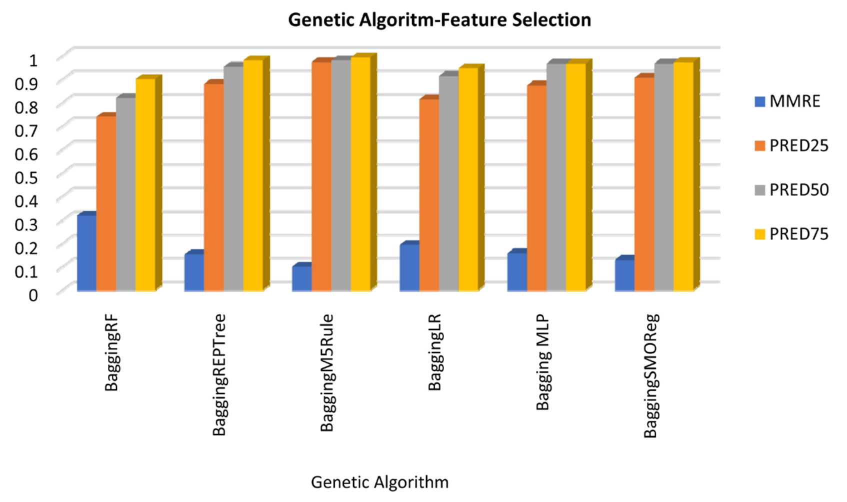


Figure 6. Genetic algorithm feature selection.

Table 3. Comparative MMRE and PRED analysis.

Results	Algorithms	MMRE-genetic algorithm	MMRE-bestfit	PRED (25) GA/ Pred (25) BF
Our	Bagging RF	0.251015	0.147558	0.8133/0.78
	Bagging REPTree	0.153349	0.126655	0.88/0.88
	Bagging M5 Rule	0.10263	0.176172	0.9733/0.9733
	Bagging LR	0.14755777	0.10263	0.8133/0.88
	Bagging MLP	0.176172	0.153349	0.87/0.80
	Bagging SMOReg	0.126655	0.251015	0.9066/0.9118
[8]	Augmented COCOMO	0.65	0.65	Pred (20) 0.3167
	Parsimonious COCOMO		0.64	0.304
[27]	Clustering	0.0103	0.0103	Pred (30) 0.356
	Regressive	0.623	0.623	-
[31]	ANN	0.352	0.352	-
	Case Based Reasoning	0.362	0.362	-
[20]	SVR	0.165	0.165	0.8889
	RBF	0.1907	0.1906	0.7222
	Linear Regression	0.233	0.233	0.7222
[33]	ANN	0.900	0.900	0.22
	Classification and Regression Tree	0.770	0.770	0.26
	Ordinary Least Square Regression	0.720	0.720	0.33
	Adjusted analogy-based estimation using Euclidean distance	0.3800	0.3800	0.57
	Adjusted analogy-based estimation using Manhattan distance	0.360	0.360	0.52
	Adjusted analogy-based estimation using Minkowski distance	0.430	0.430	0.61

predicted bands, the lower the error and the better the model. The charts show that the real and the predicted qualities are exceptionally near one another.

Answer to Research Questions

RQ1: Impact of feature selection algorithm on different ensemble learning algorithms.

If we look at Table 2 and Table 3 when we use BestFit feature selection the MMRE is minimum in all cases. If we check the PRED 0.25, 0.50 and 0.75 the value is high; this shows that the performance of all algorithms is best, while by using genetic algorithm for feature selection the value of MMRE is increased and the values of PRED 0.25, 0.50 and 0.75 are decreased so the impact of feature selection is very high in this study.

RQ2. What is the performance of ensemble Learning Techniques?

The performance of bagging M5Rule is best among all the algorithms, even for feature selection cases; at the same time, they are highly correlated in the comparative analysis available in Table 3.

6. Conclusion

In this study we perform comparative analysis among twelve ensemble methods for predicting the efforts. We use the Promise data set repository for predictions. The data set contains nineteen (19) features so we use two feature selection methods named BestFit and Genetic Algorithm on six ensemble learning algorithms. The results show that the Genetic Algorithm feature selection for the bagging M5 rule is the best method for predicting efforts with MMRE value 10%, and PRED (25), PRED (50) and PRED (75) have values 97%, 98% and 99%, respectively. In the future researchers can use Ensemble Learning with different feature selection methods for predicting efforts estimation. Hence, the ensemble learning method shows ability for predicting efforts.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Gilb, T. (1976) Software Metrics. Chartwell-Bratt, Learning.

[2] Boehm, B.W. (1981) Software Engineering Economics. Prentice-Hall, Englewood Cliffs.

[3] Putnam, L.H. (1978) A General Empirical Solution to the Macro Software Sizing and Estimating Problem. *IEEE Transactions on Software Engineering*, **4**, 345-361. https://doi.org/10.1109/TSE.1978.231521

[4] McCabe, T.J. (1976) A Complexity Measure. *IEEE Transactions on Software Engineering*, **SE-2**, 308-320. https://doi.org/10.1109/TSE.1976.233837

[5] Basili, V.R. and Perricone, B.T. (1984) Software Errors and Complexity: An Empirical Investigation. *Communications of the ACM*, **27**, 42-52. https://doi.org/10.1145/69605.2085

[6] Fei, Y.Y., Zhi, Z. and Chao, Z.S. (2004) Improvements about Halstead Model in Software Science. *Journal of Computer Applications*, 130-132.

[7] Chidamber, S.R. and Kemerer, C.F. (1994) Metrics Suite for Object-Oriented Design. *IEEE Transactions on Software Engineering*, **20**, 476-493. https://doi.org/10.1109/32.295895

[8] Smith, R.K., Hale, J.E. and Parrish, A.S. (2001) An Empirical Study Using Task Assignment Patterns to Improve the Accuracy of Software Effort Estimation. *IEEE Transactions on Software Engineering*, **27**, 264-271. https://doi.org/10.1109/32.910861

[9] Tosun, A., Turhan, B. and Bener, A.B. (2009) Feature Weighting Heuristics for Analogy-Based Effort Estimation Models. *Expert Systems with Applications*, **36**, 10325-10333. https://doi.org/10.1016/j.eswa.2009.01.079

[10] Pahariya, J.S., Ravi, V., Carr, M. and Vasu, M. (2010) Computational Intelligence Hybrids Applied to Software Cost Estimation. *International Journal of Computer Information Systems and Industrial Management Applications*, **2**, 104-112.

[11] Idri, A., Abran, A. and Mbarki, S. (2004) Validating and Understanding Software Cost Estimation Models Based on Neural Networks, Software Process and Product Measurement. *International Conference on Information and Communication Technologies: From Theory to Applications*, Damascus, Syria, 23 April 2004, 1-6.

[12] Gharehchopogh, F.S. (2011) Neural Network Application in Software Cost Estimation. *International Symposium on Innovations in Intelligent Systems and Applications* (*INISTA* 2011), Istanbul, 15-18 June 2011, 69-73. https://doi.org/10.1109/INISTA.2011.5946160

[13] Idri, A., Zakrani, A. and Zahi, A. (2010) Design of Radial Basis Function Neural Networks for Software Effort Estimation. *IJCSI International Journal of Computer Science Issues*, **7**, 11-17.

[14] Attarzadeh, I. and Ow, I.S.H. (2010) Proposing a New Software Cost Estimation Model Based on Artificial Neural Networks. 2*nd International Conference on Computer Engineering and Technology*, Chengdu, 16-18 April 2010, 487-491. https://doi.org/10.1109/ICCET.2010.5485840

[15] Weckman, G.R., Paschold, H.W., Dowler, J.D., Whiting, H.S. and Young, W.A. (2010) Using Neural Networks with Limited Data to Estimate Manufacturing Cost. *Journal of Industrial and Systems Engineering*, **3**, 257-274.

[16] Gunaydin, H.M. and Dogan, S.Z. (2004) A Neural Network Approach for Early Cost Estimation of Structural Systems of Buildings. *International Journal of Project Management*, **22**, 595-602. https://doi.org/10.1016/j.ijproman.2004.04.002

[17] Idri, A., Khoshgoftaar, T.M. and Abran, A. (2002) Can Neural Networks be Easily Interpreted in Software Cost Estimation. 2002 *Word Congress on Computational Intelligence*, Honolulu, 12-17 May 2002, 1-8.

[18] Andreou, A.S. and Papatheocharous, E. (2008) Software Cost Estimation Using Fuzzy Decision Trees. 23*rd IEEE*/*ACM International Conference on Automated Software Engineering*, L'Aquila, 15-16 September 2008, 371-374. https://doi.org/10.1109/ASE.2008.51

[19] Tadayon, N. (2005) Neural Network Approach for Software Cost Estimation. *Proceedings of the International Conference on Information Technology*: *Coding and Computing*, Las Vegas, 4-6 April 2005, 1-4. https://doi.org/10.1109/ITCC.2005.210

[20] Yu, W. and Lee, Y. (2004) Mining of Conceptual Cost Estimation Knowledge with a Neuro Fuzzy System. *Proceedings of ISARC* 2004, Session SA 03-05, Jeju, 21-25

September 2004, 118-124. https://doi.org/10.22260/ISARC2004/0023

[21] Khan, I.R., Alam, A. and Anwar, H. (2009) Efficient Software Cost Estimation Using Neuro-Fuzzy Technique. *National Conference on Recent Developments in Computing and Its Applications*, Delhi, 376-381.

[22] Cheng, M.Y., Tsai, H.C. and Sudjono, E. (2009) Evolutionary Fuzzy Hybrid Neural Network for Conceptual Cost Estimates in Construction Projects. *Information and Computational Technology*, 26*th International Symposium on Automation and Robotics in Construction* (*ISARC* 2009), Austin, 24-27 June 2009, 512-519. https://doi.org/10.22260/ISARC2009/0040

[23] Huang, S.J., Lin, C.Y. and Chiu, N.H. (2006) Fuzzy Decision Tree Approach for Embedding Risk Assessment Information in to Software Cost Estimation Model. *Journal of Information Science and Engineering*, No. 22, 297-313.

[24] Attarzadeh, I. and Hockow, S. (2010) Improving the Accuracy of Software Cost Estimation Model Based on a New Fuzzy Logic Model. *World Applied Sciences Journal*, **8**, 177-184.

[25] Huang, X., Ho, D., Ren, J. and Capretz, L.F. (2007) Improving the COCOMO Model using a Neuro-Fuzzy Approach. *Applied Soft Computing*, No. 7, 29-40. https://doi.org/10.1016/j.asoc.2005.06.007

[26] Reddy, C.S. and Raju, K. (2009) An Improved Fuzzy Approach for COCOMO's Effort Estimation Using Gaussian Membership Function. *Journal of Software*, **4**, 452-459. https://doi.org/10.4304/jsw.4.5.452-459

[27] Gallego, J.J.C., Rodriguez, D., Sicilia, M.A., Rubio, M.G. and Crespo, A.G. (2007) Software Project Effort Estimation Based on Multiple Parametric Models Generated through Data Clustering. *Journal of Computer Science and Technology*, **22**, 371-378. https://doi.org/10.1007/s11390-007-9043-5

[28] Boetticher, G., Menzies, T. and Ostrand, T. (2007) PROMISE Repository of Empirical Software Engineering Data. West Virginia University, Department of Computer Science, Morgantown. http://promise.site.uottawa.ca/SERepository/

[29] Weka. http://www.cs.waikato.ac.nz/ml/weka

[30] Kultur, Y., Turhan, B. and Bener, A.B. (2008) ENNA: Software Effort Estimation Using Ensemble of Neural Networks with Associative Memory. *Proceedings of the* 16*th ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, November 2008, 330-338. https://doi.org/10.1145/1453101.1453148

[31] Shepperd, M. and MacDonell, S. (2012) Evaluating Prediction Systems in Software Project Estimation. *Information and Software Technology*, **54**, 820-827. https://doi.org/10.1016/j.infsof.2011.12.008

[32] Malhotra, R. and Jain, A. (2011) Software Effort Prediction Using Statistical and Machine Learning Methods. *International Journal of Advanced Computer Science and Applications*, **2**, 145-152. https://doi.org/10.14569/IJACSA.2011.020122

[33] Chiu, N.H. and Huang, S.J. (2007) The Adjusted Analogy-Based Software Effort Estimation Based on Similarity Distances. *The Journal of Systems and Software*, **80**, 628-640. https://doi.org/10.1016/j.jss.2006.06.006

[34] Ahmed, B.M. (2018) Predicting Software Effort Estimation Using Machine Learning Techniques. 2018 8*th International Conference on Computer Science and Information Technology*, Amman, 11-12 July 2018, 249-256. https://doi.org/10.1109/CSIT.2018.8486222

[35] Zubair, K.M. (2020) Particle Swarm Optimisation Based Feature Selection for Software Effort Prediction Using Supervised Machine Learning and Ensemble Methods:

A Comparative Study. *Invertis Journal of Science & Technology*, **13**, 33-50. https://doi.org/10.5958/2454-762X.2020.00004.9

[36] Leung, H. and Fan, Z. (2002) Software Cost Estimation. In: *Handbook of Software Engineering and Knowledge Engineering*, World Scientific Publishing, Singapore, 307-324.