Software Effort Prediction using Statistical and Machine Learning Methods

Ruchika Malhotra
Department of Software Engineering,
Delhi Technological University
Bawana, Delhi 110042
ruchikamalhotra2004@yahoo.com

Ankita Jain
Department of Computer Engineering,
Delhi Technological University
Bawana, Delhi 110042
ankita4813@yahoo.com

Abstract - **Accurate software effort estimation is an important part of software process. Effort is measured in terms of person months and duration. Both overestimation and underestimation of software effort may lead to risky consequences. Also, software project managers have to make estimates of how much a software development is going to cost. The dominant cost for any software is the cost of calculating effort. Thus, effort estimation is very crucial and there is always a need to improve its accuracy as much as possible. There are various effort estimation models, but it is difficult to determine which model gives more accurate estimation on which dataset. This paper empirically evaluates and compares the potential of Linear Regression, Artificial Neural Network, Decision Tree, Support Vector Machine and Bagging on software project dataset. The dataset is obtained from 499 projects. The results show that Mean Magnitude Relative error of decision tree method is only 17.06%. Thus, the performance of decision tree method is better than all the other compared methods.**

Keywords— **Software effort estimation, machine learning, decision tree, linear regression**

I. INTRODUCTION

For any software organization, accurate estimation of effort is crucial for successful management and control of software project. In other words, in any software effort estimation, making an estimate of the person-months and the duration required to complete the project, is very important. Software effort estimation also plays very important role in determining cost of the software. Thus, effort estimation is crucial for the quality of the software.

Software Effort estimation techniques fall under following categories: Expert judgment, Algorithmic estimation, Machine Learning, Empirical techniques, Regression techniques, and Theory-based techniques. It is difficult to determine which model gives more accurate result on which dataset. Thus, there is a need for predicting effort and making a comparative analysis of various machine learning methods.

In this paper, we have done empirical study and comparison of some of the models on well-known China dataset [21]. The models which we are dealing with are developed using statistical and machine learning methods in order to verify which model performs the best. Linear Regression, Artificial Neural Network, Support Vector machine, Decision Tree, and bagging are the methods which are used in this work. These methods have seen an explosion of interest over years and hence it is important to analyse the performance of these methods. We have analysed these methods on large datasets collected from 499 projects.

The paper is organized as follows: Section 2 summarizes the related work. Section 3 explains the research background, i.e. describes the dataset used for the prediction of effort and also explains various performance evaluation measures. Section 4 presents the research methodology followed in this paper. The results of the models predicted for software development effort estimation and the comparative analysis are given in section 5. Finally, the paper is concluded in section 6.

II. RELATED WORK

Software effort estimation is a key consideration to software cost estimation [5]. There are numerous Software Effort Estimation Methods such as Algorithmic effort estimation, machine learning, empirical techniques, regression techniques and theory based techniques. Various models have been discussed in previous researches. An important task in software project management is to understand and control critical variables that influence software effort [5]. The paper by K.Smith, et.al. [17] has discussed the influence of four task assignment factors, team size, concurrency, intensity, and fragmentation on the software effort. These four task assignment factors are not taken into consideration by COCOMO I and COCOMO II in predicting software development effort. The paper [17] has proposed the Augmented and Parsimonious models which consider the task assignment factors to calculate effort and thus has proved that estimates are improved significantly by adding these factors while determining effort. Besides these task assignment factors which influence the effort estimation, the paper by Girish H. Subramanian, et.al.[5] ,concluded that the adjustment variables i.e. software complexity, computer platform, and program type have a significant effect on software effort. COCOMO I, COCOMO II, Function Points [1] and its various extensions all use adjustment variables, such as software complexity and reliability among others, to arrive at an adjusted estimate of software effort and cost. Also there is significant interaction between the adjustment variables which indicate that these adjustment variables influence each other and their interactions also have a significant effect on effort.

Some recent study is also done in the field of “Analogy based Estimations”. Analogy based estimations compare the similarities between the projects whose effort is to be estimated with all the historical projects. In other words, it tries to identify that historical project which is most similar to the project being estimated. To measure the similarity between pairs of projects, distance metrics are used. Euclidean (Jeffery et al., 2000), Manhattan (Emam et al., 2001) and Minkowski distances (Stamelos et al., 2003) are the widely used distance metrics in analogy-based estimations, [14]. Various researches have been done to improve the estimation accuracy in analogy – based estimations. The author Chiu, et.al. [14], proposed an adjusted analogy-based software effort estimation model by adopting the GA method to adjust the effort based on the similarity distances. In other words, the effort of the closest project is not used directly, but it is adjusted to improve the accuracy. Another method of improving the estimation accuracy is proposed by Tosun, et.al [3]. In the traditional formula for Euclidean distance, the features are either unweighted or same weight is assigned to each of the features. The problem in the unweighted case is that importance of each feature is not taken into account. In the paper [3], the authors have proposed a novel method for assigning weights to features by taking their particular importance on cost into consideration. Two weight assignment heuristics are implemented which are inspired by a widely used statistical technique called PCA.

A lot of research has also been done in Machine learning techniques of estimation. The paper by Finnie and Wittig [7], has examined the potential of two artificial intelligence approaches i.e. artificial neural networks (ANN) and case-based reasoning (CBR) for creating development effort estimation models using the same dataset which is ASMA (Australian Software Metrics Association). Also, the potential of artificial neural networks (ANN) and case-based reasoning (CBR), for providing the basis for development effort estimation models in contrast to regression models is examined by the same authors in their paper [6]. The authors concluded that artificial intelligence models are capable of providing adequate estimation models. Their performance is to a large degree dependent on the data on which they are trained, and the extent to which suitable project data is available will determine the extent to which adequate effort estimation models can be developed. CBR allows the development of a dynamic case base with new project data being automatically incorporated into the case base as it becomes available while ANNs will require retraining to incorporate new data.

Besides ANN and CBR, other important machine learning techniques is CART (Classification and regression trees). Recently, MART (Multiple additive regression trees) has been proposed that extends and improves the CART model using stochastic gradient model. The paper by Elish [12] empirically evaluates the potential and accuracy of MART as a novel software effort estimation model when compared with recently published models, i.e. radial basis function (RBF) neural networks , linear regression , and support vector regression models with linear and RBF kernels. The comparison is based on a well-known and respected NASA software project dataset. The paper [2] has compared the results of Support vector regression with both linear regression and RBF kernels.

Genetic Algorithms are also widely used for accurate effort estimation. The paper by Burgess and Lefley [4], evaluates the potential of genetic programming (GP) in software effort estimation and comparison is made with the Linear LSR, ANN etc. The comparison is made on the Desharnais data set of 81 software projects. The results obtained depend on the fitness function used.

As we have seen, software repositories or datasets are widely used to obtain data on which effort estimation is done. But software repositories contain data from heterogeneous projects. Traditional application of regression equations to derive a single mathematical model results in poor performance [8]. The paper by Gallogo [8] has used Data clustering to solve this problem.

In this research, the models are predicted and validated using both statistical and machine learning methods. The comparative analysis with previous researches has also been done. The results showed that the Decision Tree was the best among all the other models used with MMRE of 17 %.

III. Research Background

A. Feature Sub Selection Method

The data we have used is obtained from Promise data repository. The dataset comprises of 19 features, one dependent and eighteen independent variables. But, some of the independent variables are removed as they are not much important to predict the effort, thus making the model much simpler and efficient. There are various techniques used for reducing data dimensionality. We have used Feature sub selection technique which is provided in the WEKA tool [21] to reduce the number of independent variables. After applying Correlation Based Feature Subselection (CFS), the 19 variables were reduced to 10 variables (one dependent and nine independent variables). Correlation based feature selection technique (CFS) is applied to select to select the best predictors out of independent variables in the datasets [11], [18]. The best combinations of independent variable were searched through all possible combinations of variables. CFS evaluates the best of a subset of variables by considering the individual predictive ability of each feature along with the degree of redundancy between them. The dependent variable is Effort. Software development effort is defined as the work carried out by the software supplier from specification until delivery measured in terms of hours [18].

The independent variables are Output, Enquiry, Interface, Added, PDR_AFP, PDR_UFP, NPDR_AFP, NPDU_UFP and Resource. All the independent variables correspond to function point method [1].

B. Empirical Data Collection

The dataset which we have used consists of 19 drivers of effort for predicting effort estimation model. The descriptive statistics of nine independent variables chosen by CFS method is shown in table 1. The mean value of the effort (dependent variable) is found to be 3921.

TABLE I. DATASET STATISTICS

Variables	Mean	Min.	Max.	Median	Standard Deviation
Output	114	0	2455	42	221
Enquiry	62	0	952	24	105
Interface	24	0	1572	0	85
Added	360	0	1358	135	829
PDR_AFP	12	0.3	83.8	8.1	12
PDR_UFP	12	0.3	96.6	8	13
NPDR_AFP	13	0.4	101	8.8	14
NPDR_UFP	14	0.4	108.3	8.9	15
Resource	1	1	4	1	1
Effort	**3921**	**26**	**50620**	**1829**	**6474**

C. *Performance Measures*

We have used the following evaluation criterion to evaluate the estimate capability. Among all the mentioned measures, the most commonly used are PRED(A) and MMRE. Hence, we will use these two measures to compare our results with the results of the previous researches.

1. Mean Magnitude of relative error (MMRE) (or mean absolute relative error) [9], [6]

$$\mathrm{MMRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|P_i - A_i|}{A_i} \qquad (1)$$

Where Pi is the predicted value for datapoint i;
Ai is the actual value for datapoint i;
n is the total number of datapoints

2. PRED(A)

It is calculated from the relative error. It is defined as the ratio of datapoints with error (MRE) less than equal to A to the total number of datapoints. Thus, higher the value of PRED(A), the better it is considered.

$$\mathrm{PRED}\ (A) = d/n \qquad (2)$$

Where d is the value of MRE where datapoints have less than or equal to A error .Commonly used value of A is 25% in the literature.

3. Root Mean Squared Error (RMSE)
The root mean squared error is defined as:

$$E = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(P_i - A_i)^2} \qquad (3)$$

Where P_i is the predicted value for datapoint i;
A_i is the actual value for datapoint i;
n is the total number of datapoints
If $P_i = A_i$, $\forall$ i = 1,2, …, n; then E=0 (ideal case)

Thus, range of E is from 0 to infinity. RMSE gives high importance to large errors because the errors are squared before they are averaged. Thus, RMSE is used the most when large errors are undesirable.

4. Relative absolute Error (RAE)
The relative absolute error of individual dataset j is defined as:

$$E_j = \frac{\sum_{i=1}^{n}|P_{ij} - A_i|}{\sum_{i=1}^{n}|A_i - A_m|} \qquad (4)$$

Where P_{ij} is the value predicted by the individual dataset j for datapoint i;
A_i is the actual value for datapoint ;
n is the total number of datapoints;
A_m is the mean of all A_i
For ideal case, the numerator is equal to 0 and E_j = 0. Thus, the E_j ranges from 0 to infinity.

5. Root Relative Squared Error

The root relative squared error of individual dataset j is defined as:

$$E_j = \sqrt{\frac{\sum_{i=1}^{n}(P_{ij} - A_i)^2}{\sum_{i=1}^{n}(A_i - A_m)^2}} \qquad (5)$$

Where P_{ij} is the value predicted by the individual dataset j for datapoint i;

A_i is the actual value for datapoint i;
n is the total number of datapoints;
A_m is the mean of all Ai
For ideal case, the numerator is equal to 0 and E_j = 0. Thus, the E_j ranges from 0 to infinity.

6. Mean Absolute error

The mean absolute error measures of how far the estimates are from actual values. It could be applied to any two pairs of numbers, where one set is "actual" and the other is an estimate, prediction.

7. Correlation Coefficient

Correlation measures of the strength of a relationship between two variables. The strength of the relationship is indicated by the correlation coefficient. The larger the value of correlation coefficient, the stronger the relationship.

D. *Validation Measures*

There are three validation techniques namely hold-out, leave-one-out and K-cross validation [13]. As our dataset is large consists of 499 data points, the hold out method is used where the dataset is divided into two parts, i.e. the training and validation set.

IV. Research Methodology

In this paper, we are using the machine learning techniques in order to predict effort. We have used one regression and four machine learning methods in order to predict effort. Support vector machines, Artificial Neural

Network, Decision tree and Bagging methods have seen an explosion of interest over the years, and have successfully been applied in various areas.

A. *Linear Regression*

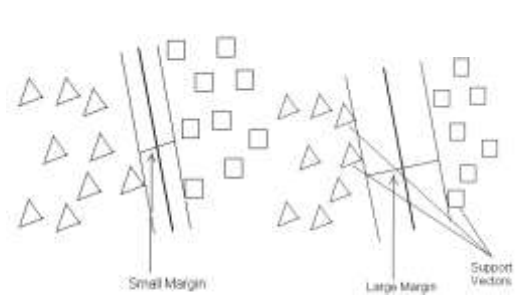
Linear regression analyses the relationship between two variables, X and Y. One variable is the dependent variable and the other is the independent variable. For doing this, it finds a line which minimizes the sum of the squares of the vertical distances of the points from the line. In other words, it is method of estimating the conditional expected value of one variable y given the values of some other variable or variables x.

B. *Support Vector Machine*

Support Vector Machine (SVM) is a learning technique which is used for classifying unseen data correctly. For doing this, SVM builds a hyperplane which separates the data into different categories. The dataset may or may not be linearly separable. By 'linearly separable' we mean that the cases can be completely separated i.e. the cases with one category are on the one side of the hyperplane and the cases with the other category are on the other side. For example Figure 1 shows the dataset where examples belong to two different categories – triangles and squares. Since these points are represented on a 2 – dimensional plane, they can be separated by a 1-dimensional line. To separate these points into 2 different categories, there is infinite number of lines possible. Two possible candidate lines are shown in the figure 1. However, only one of the lines gives maximum separation/margin and that line is selected.'Margin' is defined as distance between the dashed lines (as shown in figure) drawn parallel to the separating lines. These dashed lines give the distance between the separating line and closest vectors to the line. These vectors are called as support vectors. SVM can also be extended to the non-linear boundaries using kernel trick. The kernel function transforms the data into higher dimensional space to make the separation easy. We have used SVM for estimating continuous variable, i.e. effort. [19].

Figure 1.

S VM



C. *Artificial Neural network*

Artificial Neural Network (ANN) comprises a network of simple interconnected units called "neurons" or "processing units". The ANN has three layers, i.e. the input layer, hidden layer and the output layer. The first layer has input neurons which send data via connections called weights to the second layer of neurons and then again via more weight to the third layer of output neurons. More complex systems have more than one hidden layers. But it has been proved in literature that more than one hidden layer may not be acceptable [20]. *The most common algorithm for training or learning is known as error back –propagation algorithm.*

Error back-propagation learning consists of two passes: a forward pass and a backward pass. In the forward pass, an input is presented to the neural network, and its effect is propagated through the network layer by layer. During the forward pass the weights of the network are all fixed. During the backward pass the weights are all updated and adjusted according to the error computed.. An error is composed from the difference between the desired response and the system output. This error information is fed back to the system and adjusts the system parameters in a systematic fashion (the learning rule). The process is repeated until the performance is acceptable. The model predicted in this study consists of 9 input layers (independent variables chosen in our study) and one output layer.

D. *Decision Tree*

Decision tree is a methodology used for classification and regression. It provides a modelling technique that is easy for human to comprehend and simplifies the classification process. Its advantage lies in the fact that it is easy to understand; also, it can be used to predict patterns with missing values and categorical attributes. Decision tree algorithm is a data mining induction techniques that recursively partitions a data set of records using depth-first greedy approach or breadth-first approach until all the data items belong to a particular class.

A decision tree structure is made of (root, internal and leaf) nodes and the arcs. The tree structure is used in classifying unknown data records. At each internal node of the tree, a decision of best split is made using impurity measures. We have used M5P implemented in WEKA Tool [21]. This method generated M5 model rules and trees. The details of this method can be found in [16].

E. *Bagging*

Bagging which is also known as bootstrap aggregating is a technique that repeatedly samples (with replacement) from a data set according to a uniform probability distribution [10]. Each bootstrap sample has the same size as the original data. Because the sampling is done with replacement, some instances may appear several times in the same training set, while others may be omitted from the training set. On average, a bootstrap sample D_i contains approximately 63% of the original training data because each sample has a probability $1- (1- 1/N)^N$ of being selected in each $D_{i.}$ If N is sufficiently large, this probability converges to 1-1/e = 0.632. After training the k classifiers, a test instance is assigned to the class that receives the highest number of votes.

V. Analysis Results

A. *Model Prediction Results*

China Dataset [15] was used to carry out the prediction of effort estimation model. A holdout technique of cross validation was used to estimate the accuracy of effort estimation model. The dataset was divided into two parts i.e. training and validation set in ratio of 7:3. Thus 70% was used for training the model and 30% was used for validating accuracy of the model. Four machine learning methods and one regression method was used to analyse the results.

The model with the lower MMRE, RMSE, RAE, RRSE, MAE and the higher correlation coefficient and PRED(25) is considered to be the best among others. As shown in table, the decision tree results are found to be best with the MMRE value 17.06%,RAE value 32.02, RRSE value 38.40 ,correlation coefficient 0.93 , and PRED(25) value 52%. Hence decision tree method is found to be effective in predicting effort. Also, the results of decision tree are competent with the traditional linear regression model.

TABLE II. OUR RESULTS

Performance Measures	Linear regression	Support Vector Machine	Artificial Neural Network	Decision tree	Bagging
Mean Magnitude Relative Error (MMRE) %	17.97	25.63	143.79	**17.06**	74.23
Root mean squared error (RMSE) %	4008.11	3726.71	5501.18	**2390.47**	3656.75
Relative absolute error (RAE) %	54.16	48.49	71.50	**32.02**	45.79
Root relative squared error (RRSE) %	64.74	60.13	91.17	**38.40**	59.11
Correlation coefficient	0.79	0.81	0.75	**0.93**	0.83
Mean absolute error (MAE) %	1981.48	1774.36	2561.00	**1173.43**	1668.03
PRED(25) %	36	38.66	11.33	**52**	34.66

The graphs as shown in figures 2-6, for the actual and the values as predicted by the particular model are shown on Y-axis and they correspond to the 499 projects.. The 'black' curve presents the curve for the actual values, whereas the 'red' curve presents the curve for the predicted values. The closer the actual and predicted curves, the lesser are the error and better are the model. The graphs show that the actual and the predicted values are very close to each other. As shown in figure 5, the decision tree shows the best result.

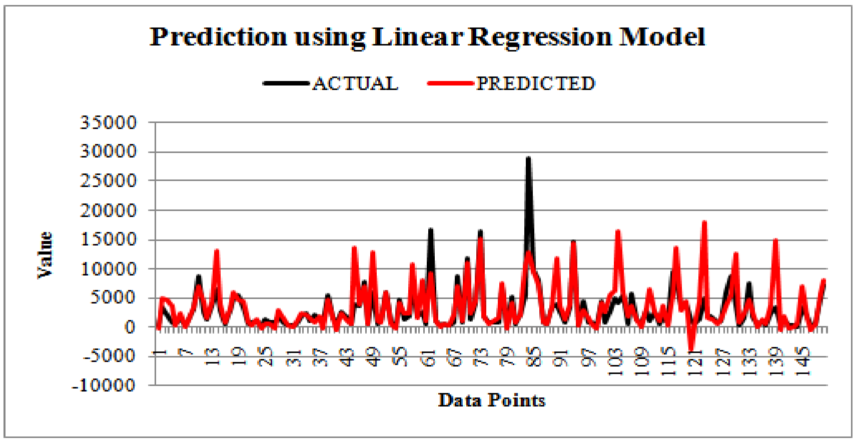


Figure 2. Results Using Linear Regression

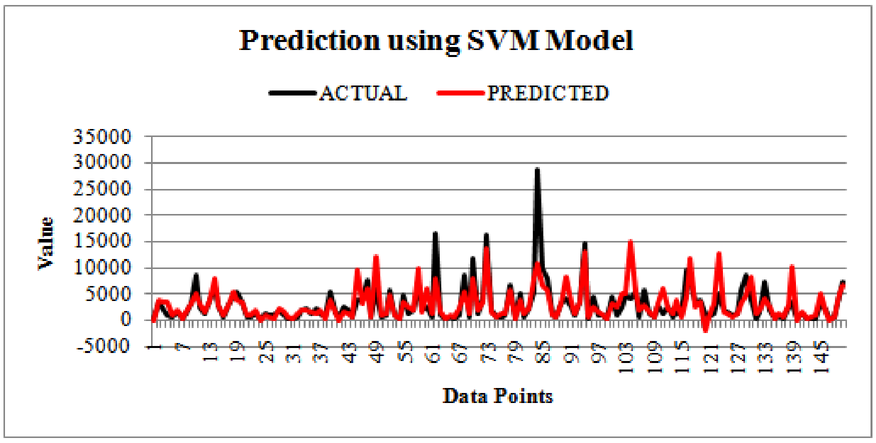


Figure 3. Results Using Support Vector Machine

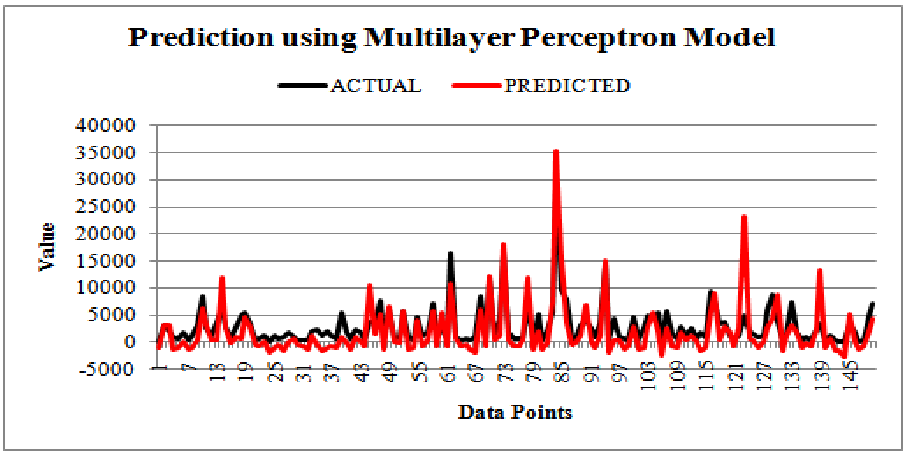


Figure 4. Results Using Artificial Neural Network

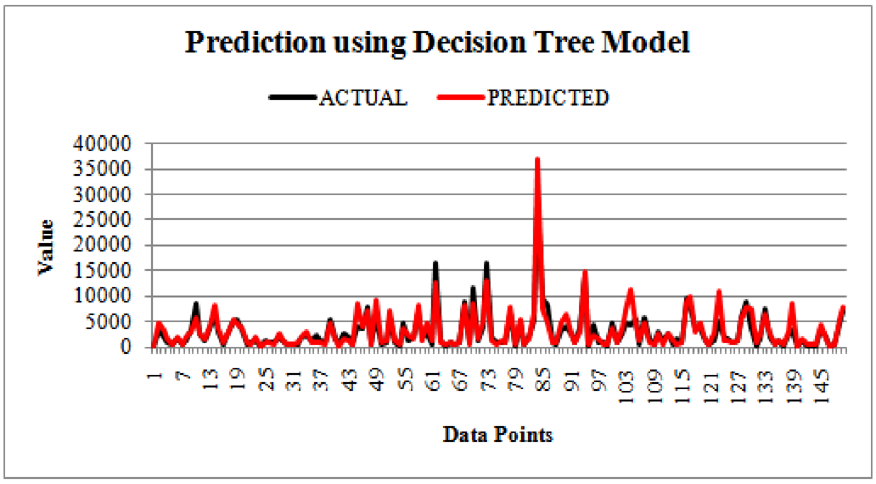


Figure 5. Results Using Decision Tree

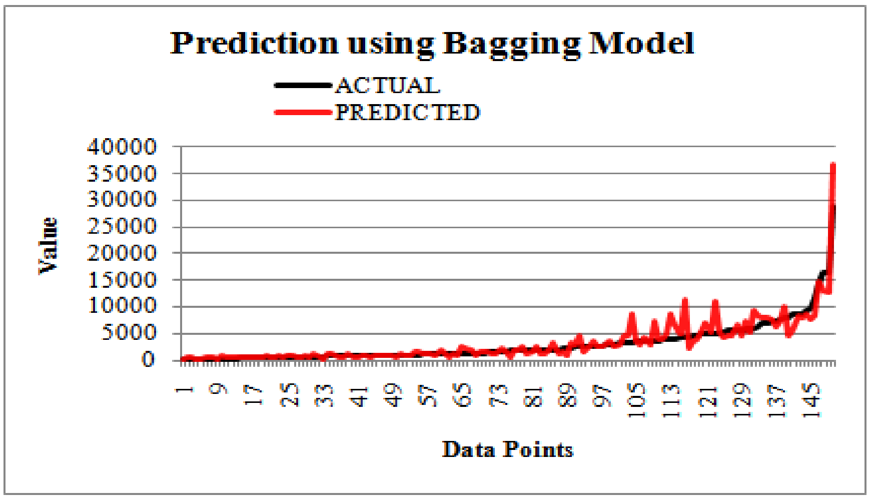


Figure 6. Results Using Bagging

B. *Comparative analysis with previous studies*

Our results of effort estimation model predicted using decision tree method were better than all the other four methods used in our study. We have compared our results with eight other previous studies. The previous studies under comparison have same dependent variable, although the independent variables vary for each study. As shown in table 3, out of 24 models of literature, our decision tree model has outperformed the MMRE values of 21 models. The results of PRED(25) are also very good and higher as compared to most of the models. Thus, our effort estimation model using decision tree method is widely acceptable.

TABLE III. COMPARISON ANALYSIS

Papers	Methods Used	Pred(25)	MMRE
Our Results	Decision Tree	52	17.06
	Bagging	34.66	74.23
	Linear Regression	36	17.97
	SVM	38.66	25.63
	ANN	11.33	143.79
[14]	ANN	22	90
	Classification And Regression Trees	26	77
	Ordinary Least Square Regression	33	72
	Adjusted analogy-based estimation using Euclidean distance	57	38
	Adjusted analogy-based estimation using Manhattan distance	52	36
	Adjusted analogy-based estimation using Minkowski distance	61	43
[17]	Augmented COCOMO	Pred(20)31.67	65
	Parsimonious COCOMO	30.4	64
[8]	Clustering	Pred(30) 35.6	1.03
[6]	Regressive	-	62.3
	ANN	-	35.2
	Case Based Reasoning	-	36.2
[12]	Multiple Additive Regression Trees	88.89	8.97
	Radial Basis Function	72.22	19.07
	SVR Linear	88.89	17.4
	SVR RBF	83.33	17.8
	Linear Regression	72.22	23.3
[4]	Genetic Programming	23.5	44.55
	ANN	56	60.63
[7]	ANN	-	17
	Case Based Reasoning	-	48.2
[21]	SVR	88.89	16.5
	Linear Regression	72.22	23.3
	Radial Basis Function	72.22	19.07

VI. CONCLUSION

In this research we have made a comparative analysis of one regression with four machine learning methods for predicting effort. We have obtained results using the data obtained from Promise data repository. The dataset consists of 19 features which we have reduced to 10 features using CFS method. The results show that the decision tree was the best method for predicting effort with MMRE value 17% and PRED(25) value 52%. The software practitioners and researchers may apply decision tree method for effort estimation. Hence, machine learning methods selected in this study have shown their ability to provide an adequate model for predicting maintenance effort.

The future work can further replicate this study for industrial software. We plan to replicate our study to predict effort prediction models based on other machine learning algorithms such as genetic algorithms. We may carry out cost benefit analysis of models that will help to determine whether a given effort prediction model would be economically viable.

REFERENCES

[1] A. Albert and J.E. Gaffney, "Software Function Source Lines of Code and Development Effort Prediction: A Software Science Validation," IEEE Trans. Software Engineering, vol. 9, pp. 639-648, 1983.

[2] A.L.I. Oliveira, " Estimation of software effort with support vector regression," Neurocomputing, vol. 69, pp. 1749-1753, Aug. 2006.

[3] A.Tosun, B.Turhan and A.B. Bener, "Feature weighting heuristics for analogy-based effort estimation models," Expert Systems with Applications, vol. 36, pp. 10325-10333, 2009.

[4] C.J. Burgess and M.Lefley, "Can genetic programming improve software effort estimation? A comparative evaluation," Information and Software Technology, vol. 43, pp. 863-873, 2001.

[5] G.H. Subramanian , P.C. Pendharkar and M.Wallace, "An empirical study of the effect of complexity, platform, and program type on software development effort of business applications," Empirical Software Engineering, vol. 11, pp. 541-553, Dec. 2006.

[6] G.R. Finnie and G.E. Wittig, "A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models," Journal of Systems and Software, vol. 39, pp. 281-289, 1997.

[7] G.R.Finnie and G.E. Wittig, "AI Tools for Software Development Effort Estimation," in Proc. SEEP '96 , 1996, International Conference on Software Engineering: Education and Practice (SE:EP '96).

[8] J.J.C Gallego, D.Rodriguez, M.A.Sicilia, M.G.Rubio and A.G. Crespo, "Software Project Effort Estimation Based on Multiple Parametric Models Generated Through Data Clustering," Journal of Computer Science and Technology, vol. 22, pp. 371-378, May 2007.

[9] K. Srinivasan and D. Fisher, " Machine Learning Approaches to Estimating Software Development Effort," IEEE Transactions on Software Engineering, vol. 21, Feb. 1995.

[10] L.Breiman, "Bagging predictors," Machine Learning, vol. 24, pp. 123-140, Aug. 1996.

[11] M.Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in Proceedings of the 17th International Conference on Machine Learning, pp.359-366.

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

[13] M.Stone, "Cross-validatory choice and assess-ment of statistical predictions," Journal Royal Stat. Soc., vol. 36, pp. 111-147, 1974.

[14] N.H. Chiu and S.J. Huang, " The adjusted analogy-based software effort estimation based on similarity distances," The Journal of Systems and Software, vol. 80, pp. 628-640, 2007.

[15] Promise. Available: http://promisedata.org/repository/.

[16] R.J. Quinlan, "Learning with Continuous Classes," in 5th Australian Joint Conference on Artificial Intelligence, Singapore, pp. 343-348, 1992.

[17] R.K. Smith, J.E.Hale and A.S.Parrish, "An Empirical Study Using Task Assignment Patterns to Improve the Accuracy of Software Effort Estimation," IEEE Transactions on Software Engineering, vol. 27, pp. 264-271, March 2001.

[18] R.Malhotra, A.Kaur and Y.singh, "Application of Machine Learning Methods for Software Effort Prediction," in Newsletter ACM SIGSOFT Software Engineering Notes , vol. 35, May 2010.

[19] S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, K.R.K. Murthy, "Improvements to the SMO Algorithm for SVM Regression," IEEE Transactions on Neural Networks, vol. 13, March 2001.

[20] T.M. Khoshgaftaar, E.D. Allen, J.P. Hudepohl, S.J. Aud, "Application of neural networks to software quality modeling of a very large telecommunications system," IEEE Transactions on Neural Networks, vol. 8, pp. 902-909, July 1997.

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

AUTHORS PROFILE

Ruchika Malhotra She is an assistant professor at the Department of Software Engineering, Delhi Technological University (formerly Delhi College of Engineering), Delhi, India, She was an assistant professor at the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. Prior to joining the school, she worked as full-time research scholar and received a doctoral research fellowship from the University School of Information Technology, Guru Gobind Singh Indraprastha Delhi, India. She received her master's and doctorate degree in software engineering from the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. Her research interests are in software testing, improving software quality, statistical and adaptive prediction models, software metrics, neural nets modeling, and the definition and validation of software metrics. She has published more for than 40 research papers in international journals and conferences. Malhotra can be contacted by e-mail at ruchikamalhotra2004@yahoo.com

Ankita Jain She is at the Department of Computer Engineering, Delhi Technological University (formerly Delhi College of Engineering), Delhi, India, She received her bachelor's degree in computer engineering from the Guru Gobind Singh Indraprastha University, Delhi, India. Her research interests are statistical and adaptive prediction models and improving software quality. She can be contacted by e-mail at ankita4813@yahoo.com