A Comparison of Machine Learning Techniques for Software Effort Estimation

[1]Ashwni kumar, [2]Dr D.L.Gupta
[1]M.Tech student, [2]Associate Professor
[1]Department of Computer Science and Engineering,
[1]K.N.I.T, Sultanpur, U.P, India.

Abstract: In the software engineering the most difficult problem is the software effort estimation. Today's industry is facing the most common problem of developing error free software at low cost with less time. So to develop good quality of software it is very essential to know how to estimate the cost of the software. For good result of estimation and prediction it is very essential to take consideration of any previous projects. It is not very easy task to determine which techniques gives better result in software effort estimation. This paper focus on performance of M5 Rule, Decision Table, Conjection Rule, Zero Rule classifier is experimented for software effort estimation. The performance measures criteria are based on RMSE and MAE values. The result shows that the M 5 Rule technique gives best performance in software effort estimation model.

IndexTerms - Software Effort Estimation, Machine Learning Techniques, COCOMO 2 dataset, comparative Analysis.

I. INTRODUCTION

An accurate estimation is the key to the success of software development projects. As the cost of software development continues to grow due to inflation , wages and other factors, software application are expanding into multiple domains, architectures and platforms the demand for smarter and more innovative software is stronger than ever, which justifies the need for better, cheaper and faster software solutions. In software project management, estimation method are used to estimate the effort development process generally involves software estimation is particularly difficult due to intangible nature of the software and the margin of error inherited in the estimate.

Software effort estimation is one of the most important activities in software development. Software estimation is difficult task in project planning and management processes [1]. Estimating software effort is one of the most persistent problems in software engineering. Software is the most expensive component of many computer systems. A large amount of bugs creates a huge difference between gain and gain in effort estimation [2]. Evaluation the most practical use possible of the effort required to produce or maintain software based on inadequate and unreliable inputs. The effort is usually evaluated in person- months [3]. The effort estimation acts as an input to the project plans. Using the software effort estimation, it is easy to find resources that are used to complete project on time. Estimating the software effort plays a fundamental role in finalizing any project. An accurate calculation normally leads to the completion of the project at the right time [4]. A machine learning method plays an important role in this area because it can increase the efficiency of estimation by applying the training rule to estimate a correct effort required [5].

II. RELATED WORK

The estimation techniques required for software development has been discussed in this section. The work of various authors has been complied in this section.

The first technique was introduced in 1960s [6] to estimate software development effort, for this approach it required expert judgment for estimation. In these cases, a domain expert applies his or her prior experience to come up with an estimation of the needed effort. During the last 30 years, a number of formal models for software effort estimation have been proposed such as Cocomo [6], Cocomo II [7], SLIM [7], and Function Points Analysis [8]. These approaches have some advantages, which help in the formulaic underpinning of software effort estimation [9].

M. H. Halstead [10] has Proposed the model which predicts the rate of error and do not require the in-depth analysis of programming structure. It pro- posed the code length and volume metrics. Code length is used to measure the source code program and volume corresponds to the amount of required storage space. Numerous industry studies support the use of Halstead in predicting programming effort and mean number of programming bugs. However it depends on completed code and has little or no use as a predictive estimating model.

Doty Model [11] published in 1977, is used to estimate efforts for Kilo lines of code (KLOC). This model constitutes various aspects of the software develop- ment environment such as user participation, customer- oriented changes, memory constraints etc.

Heetika Duggal et al. [12] has proposed a model to obtain good results for effort estimation. In this study, they calculate the performance of M5-Rules Algorithm, single conjunctive rule learner and decision table majority classifier are experimented for modeling of Effort Estimation of Software Projects and performance of developed models is compared with the existing algorithms namely Halstead, Walston-Felix, Bailey-Basili, Doty in terms of MAE and RMSE.

A hybrid method has been suggested by **Bardsiri et al** [13] to raise the precision of development effort estimation based on the mixture of fuzzy clustering, Analogy-based estimation (ABE) and artificial neural networks (ANN) methods. In the suggested method, the result of unrelated and not consistent projects on estimates was reduced by planning a framework, in which all the projects were grouped. Two comparatively large datasets were used to assess the presentation of the suggested method and the attained results were compared to eight other estimation methods. These methods were chosen from the most common algorithmic and non-algorithmic methods employed widely in the field of software development effort estimation. Based on Mean Magnitude of Relative Error (MMRE) and prediction (PRED) (0.25) parameters, comparisons were executed by means of three-fold cross validation technique. According to the attained effects, the suggested method outperformed the other methods and considerably enhanced the precision of estimates in both datasets.

Lin et al. [14] have suggested a model which joins genetic algorithm (GA) with support vector machines (SVM). They could locate the best parameter of SVM regression by the suggested model, and makes more precise prediction. During the research, they check and authenticate their model by employing the historical data in COCOMO, Desharnais, Kemerer, and Albrecht. They demonstrate the effects by prediction level (PRED) and mean magnitude of relative error (MMRE).

Satapathy et al. [15] have suggested a model which is executed by means of Multi-Layer Perceptron (MLP) and Radial Basis Function Network (RBFN) and produced results has been compared. In addition, a relative study of software effort estimation by means of MLP and RBFN has been offered. The results reveal that MLP model provides less value of MMRE, NRMSE and higher values of prediction precision. Therefore it was concluded that the effort estimation by means of MLP model will offer more precise results than RBFN.

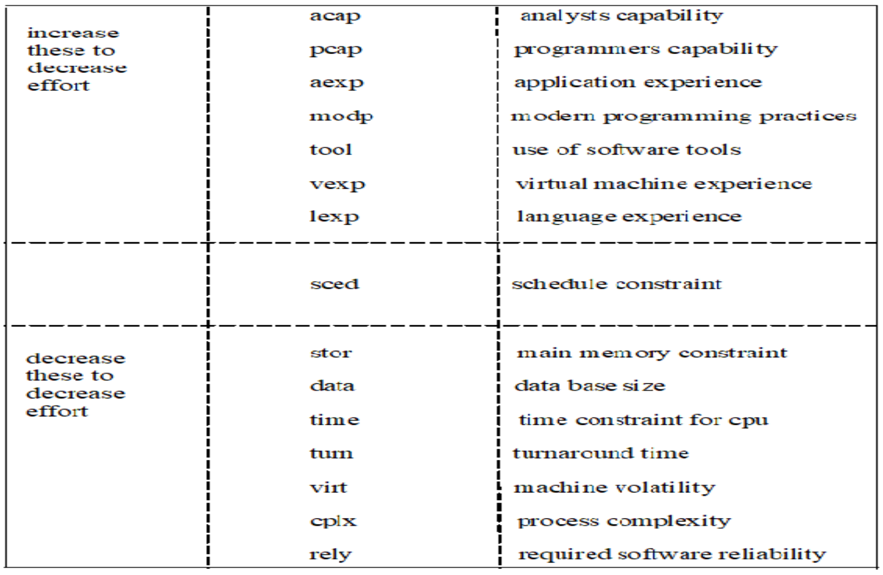
In 2011, **Ruchika Malhotra and Ankita Jain** [16] presented a paper on effort estimation. She has comparing the techniques of linear regression, ANN, SVM, Decision Tree and the inclusion in the project of the software dataset. She has used dataset of 499 projects. This dataset contains 19 attributes that must be abbreviated using the CFS method. The results show that decision tree approach has relative magnitude error of 17% compared to another approach. Therefore the estimation or results of the Decision Tree approach are good rather than with other methods.

III. RESEARCH BACKGROUND

There are many tools are available to estimate the software effort using machine learning techniques for example Jupyter Notebook, MATLAB, WEKA etc. In this research paper we are evaluating performance measures of the software effort estimation using machine learning techniques with the help of WEKA tool. Basically WEKA is used to solve a large number of problem such as classification, clustering and neural network. WEKA stands for Waikato Environment of Knowledge. It is a machine learning toolkit developed at the University of Waikato in New Zealand. The WEKA 3.2 version provides graphical user interface based on graphical tool which is basically used for preprocessing a calculation method and a comparison platform for various machine learning techniques.

3.1 Data Collection

We have used a series of data to evaluate the reliability of the software using machine learning method. In this proposed work we have used the publicly available PROMISE Repository dataset. COCOMO NASA 2 dataset is used for this proposed work. COCOMO NASA 2 dataset is used for analysis and validation of the model can be got from historic projects of NASA. It consists of 93 NASA project taken from different centers. It consists of 23 input attributes and 93 instances. The row of dataset represent the project, column represent the attributes. The datasets is of COCOMO II format. Some of the product metrics that are included in the dataset are: RELY, DATA, CPLX, AEXP, LEXP, MODP, SCED, KLOC, and ACT_EFFORT. They are effort multipliers for COCOMO dataset. Figure3.1 shows the effort multiplier of COCOMO II model.



increase these to decrease effort	acap	analysts capability
	pcap	programmers capability
	aexp	application experience
	modp	modern programming practices
	tool	use of software tools
	vexp	virtual machine experience
	lexp	language experience
	sced	schedule constraint
decrease these to decrease effort	stor	main memory constraint
	data	data base size
	time	time constraint for cpu
	turn	turnaround time
	virt	machine volatility
	cplx	process complexity
	rely	required software reliability

Figure3.1: Effort Multiplier of COCOMO II Model

3.2 Evaluation Measures

There are following evaluation measures for the accuracy of the software effort estimation.

- **Mean Absolute Error (MAE)**: - It measures how estimate are calculated from the actual value.
- **Root Mean Square Error (RMSE):-** It is used for the entity of the different between the value predicted by the estimator and the value actually observed by the modelled [17].

IV. RESEARCH METHODOLOGY

In this research paper we have use a 4 different machine learning approach, such as the M5 Rule, Decision Table, Conjection Rule, Zero Rule to estimate software effort. For the implementation WEKA tool is used to measure performance and the ARFF file for the dataset. From the data, an ARFF file was created and loaded into WEKA Explorer contain a preprocessing panel. Clicking on it, we can see the file open, clicking on it loads the file in WEKA. Now click on the classification panel that clicks on the techniques. There are several techniques, choose one by one M5 Rule, Decision Table, Conjection Rule, Zero Rule to perform an evaluation. Currently, the tested option is chosen for 80% for all training data, and performing this machine learning approach on it.

In figure 4.1 it shows the working model of proposed work for software effort estimation. We have collected the NASA COCOMO 2 dataset for effort estimation. We have preprocessed the dataset, and then we have divided the dataset in 80% of training dataset and 20% of test dataset. Apply 4 different machine learning algorithm on COCOMO 2 dataset. After that we have trained 4 different machine learning techniques on NASA COCOMO 2 training dataset and we test the trained model with the testing dataset to estimate the effort. We have also check the performance measure of 4 machine learning techniques by using performance parameters MAE and RMSE. Then after computing performance we have compares the result of 4 different machine learning and analysis of result provides the best performance of technique for effort estimation.

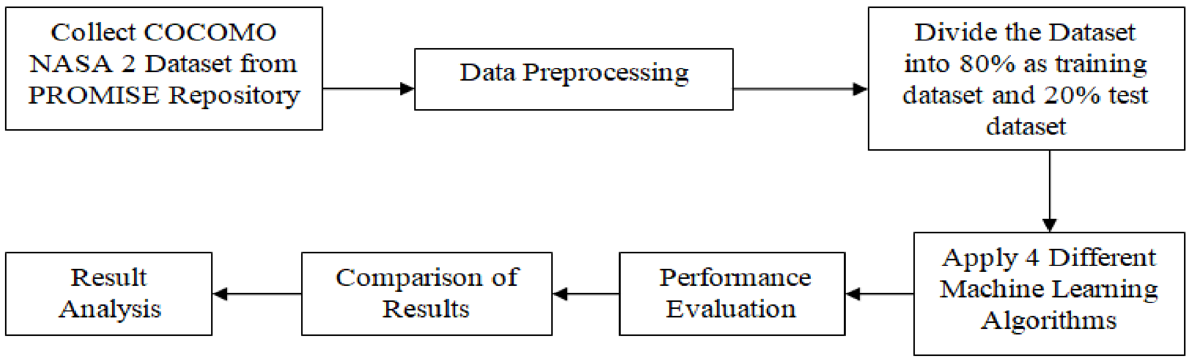

Collect COCOMO NASA 2 Dataset from PROMISE Repository
Data Preprocessing
Divide the Dataset into 80% as training dataset and 20% test dataset
Apply 4 Different Machine Learning Algorithms
Performance Evaluation
Comparison of Results
Result Analysis

Figure 4.1: Proposed Methodology

V. RESULTS AND DISCUSSION

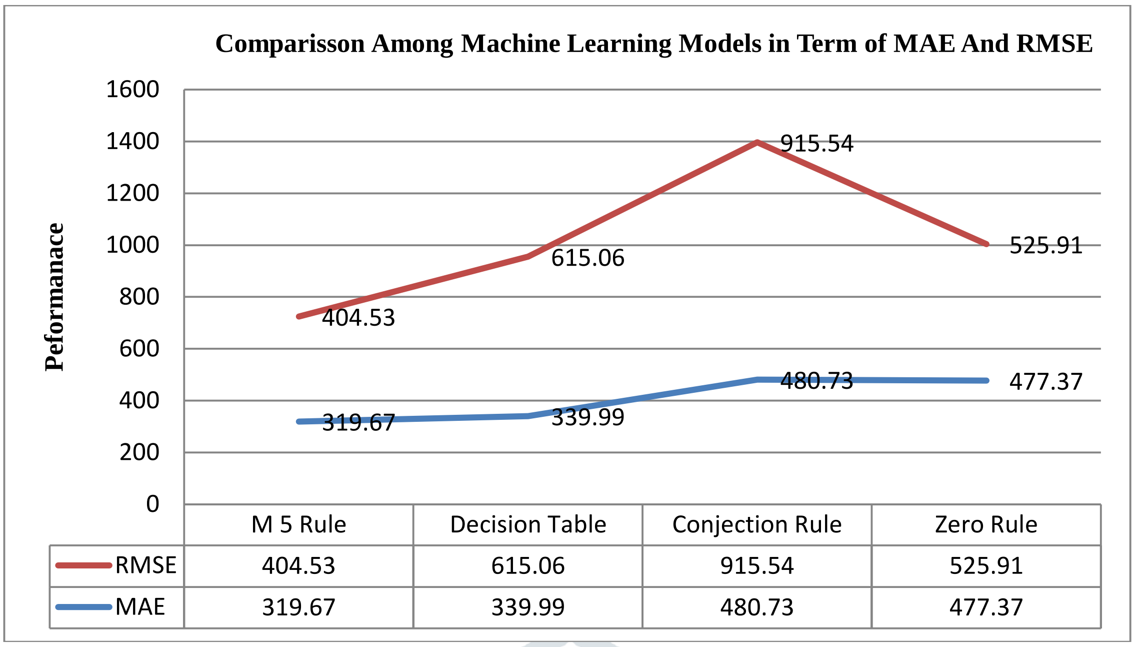
5.1 Comparison of Techniques

The implementation is performed using the WEKA tool. To perform the estimation, 80% of the training dataset and 20% of the test dataset method are used in this research. The comparison between the M5 Rule, Decision Table, Conjection Rule, Zero Rule is performed using the COCOMO 2 dataset. The result of this evaluation is shown in table 5.1.

Table 5.1: 4 Machine Learning Algorithms

Model	Performance Criteria on 80% training dataset and 20% test dataset	
	MAE(Mean Absolute error)	**RMSE(Root mean square error)**
M 5 Rule	319.67	404.53
Decision Table	339.99	615.06
Conjection Rule	480.73	915.54
Zero Rule	477.37	525.91

The value of mean Absolute Error, Root Mean Square Error are observed to be 319.67, 404.53. on this values we found that M5 Rule gives better outcomes and to be effective in estimating software effort.



	M 5 Rule	Decision Table	Conjection Rule	Zero Rule
RMSE	404.53	615.06	915.54	525.91
MAE	319.67	339.99	480.73	477.37

Figure 5.1: Line graph representing comparison of techniques for COCOMO 2 Dataset

The figure 5.1 shows the performance measures for COCOMO 2 dataset with 4 machine Learning techniques. From figure 5.1 we can observe that M 5 Rule provides better result as compared to other machine learning techniques.

V. CONCLUSION AND FUTURE SCOPE

In this research paper, different machine learning algorithms are tested to estimate the software effort for the project. The result obtained by applying an M5 Rule to the COCOMO 2datset it shows that the M 5 Rule calculated better estimation result. Therefore, it is suggested to use the technique of M 5 Rule to create an appropriate model for the software effort estimation. Future work involves the study of new methods and a software effort estimation model that can help us easily understand the software effort estimation process. The work can be continued by choosing a combination of machine learning techniques that gives better result.

REFERENCES

[1] Jonathan Lee,Wen-Tin Lee,Jong-Yih Kuo, "Fuzzy Logic as a Basic for Use Case Point Estimation" IEEE International Conference on Fuzzy Systems ,June 27-30, Taipei, Taiwan, 2011

[2] Khaled Hamdan, Mohamed Madi, "Software Project Effort: Different Methods of Estimation", International Conference on Communications and Information Technology (ICCIT), Aqaba., IEEE, 2011

[3] Alhad Vinayak Sapre, "Feasibility of Automated Estimation of Software Development Effort in Agile Environments", 2011

[4] Saleem Basha, Dhavachelvan, "Analysis of Empirical Software Effort Estimation Models" (IJCSIS) International Journal of Computer Science and Information Security, Vol. 7, No. 3, 2010

[5] Mamoona Humayun and Cui Gang, "Estimating Effort in Global Software Development Projects Using Machine Learning Techniques" , International Journal of Information and Education Technology, Vol. 2, No. 3, June 2012

[6] Boehm, B.W. "Software Engineering Economics", Prentice-Hall, Englewood Cliffs, N, 1981.

[7] B. Boehm, R. Madachy, and B. Steece, Software Cost Estimation with Cocomo II. Prentice Hall, 2000

[8] L.H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimation Problem," IEEE Trans. Software Eng., vol. 4, no. 4, pp. 345-361, July 1978.

[9] A.J. Albrecht and J.E. Gaffney, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," IEEE Trans. Software Eng., vol. 9, no. 6, pp. 639-648, Nov. 1983.

[10] M. H. Halstead, "Elements of Software Science," Elsevier, New York, 1977

[11] Doty Associates, Inc., "Software Cost Estimates Study," Vol. 1, 1977, pp. 77-220.

[12] Heetika Duggal, P. S,"Comparative Study of the Performance of M5-Rules Algorithm with Different Algorithms," *Journal of Software Engineering and Applications*, pp 270-276, 2012.

[13]Bardsiri, Vahid Khatibi et al.,"A PSO-based model to increase the accuracy of software development effort estimation Software Quality Journal", vol. 21.3, pp.501-526, 2013.

[14] Du, Wei Lin, et al. "A hybrid intelligent model for software cost estimation." arXiv preprint arXiv: 1512.00306, 2015.

[15] Panda, Aditi, Shashank Mouli Satapathy, and Santanu Kumar Rath, "Empirical validation of neural network models for agile software effort estimation based on story points," *Procedia Computer Science* vol. 57, pp.772-781, 2015.

[16] Ruchika Malhotra, "Software Effort Prediction using Statistical and Machine Learning Methods", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No.1, January 2011

[17]V. U. B. Challagulla, F. B. Bastani, I.-L. Yen and R. A. Paul, "Empirical Assessment of Machine Learning Based Software Defect Prediction Techniques," 10*th IEEE Inter-national Workshop on Object-Oriented Real-Time Depend- able Systems*, Sedona, 2-4 February 2005, pp. 263-270. doi:10.1109/WORDS.2005.32