Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 56 (2013) 13677-13682

ISSN: 2229-712X

Application of machine learning techniques for predicting software effort

Prabhakar and Maitreyee Dutta

Department of Computer Science, National Institute of Technical Teachers Training and Research, Chandigarh, India.

ARTICLE INFO

Article history:

Received: 28 February 2013;

Received in revised form:

21 March 2013;

Accepted: 23 March 2013;

Keywords

Software effort estimation,
Machine Learning Techniques,
Artificial Neural Network,
Adaptive Neuro Fuzzy Inference System, and
Decision Tree.

ABSTRACT

Software effort estimation is an important area in the field of software engineering. If the software effort is over estimated it may lead to tight time schedules and thus quality and testing of software may be compromised. In contrast, if the software development effort is underestimated it may lead to over allocation of man power and resource. There are many models for estimating software effort. The aim of the work is to estimate software effort using various machine learning techniques like Artificial Neural Network (ANN), Adaptive Neuro Fuzzy Inference System (ANFIS), and Decision Tree (DT). China dataset of software projects has been used in order to compare the performance results obtained from these models. The indices are Sum-Square-Error (SSE), Mean-Square-Error (MSE), Root-Mean-Square-Error (RMSE), Mean-Magnitude-Relative-Error (MMRE), Relative-Absolute-Error (RAE), Relative-Root-Square-Error (RRSE), Mean-Absolute-Error (MAE), Correlation Coefficient (CC), and PRED (25).

1. Introduction

One of the most important research areas of the software measurement is software effort estimation. Software effort estimation models are mainly categorized in to algorithmic and non-algorithmic. The algorithmic models are mainly COCOMO, Function Points and SLIM. Theses models are also known as parametric models because they predict software development effort using a formula of fixed form that is parameterized from historical data.

The algorithmic model require as input attributes such as experience of the development team, the required reliability of the software, the programming language in which the software is to be written, an estimate of the final number of delivered source line of code (SLOC), complexity and so on which are difficult to obtain during the early stage of a software development life cycle (SDLC). They have also difficulty in modelling the inherent complex relationships [9].

The limitations of algorithmic models compel us to the exploitation of non-algorithmic techniques which are soft computing based. These techniques have advantage of

1. Ability to learn from previous data.
2. Able to model complex relationship between the dependent (effort) and independent variables (cost).
3. Ability to generalize from the training dataset thus enabling it to produce acceptable result from previous unseen data.

2. Related Work

A lot of research has been done using machine learning techniques like Artificial Neural Networks, Decision Tree, Linear Regression, Support Vector Machine, Fuzzy Logic, Genetic Algorithm, Empirical Techniques, and Theory based techniques for predicting the software effort estimation.

The paper by **FINNIE** and **WITTIG [4]**, has examined the potential of two artificial intelligence approaches i.e. Artificial Neural Networks (ANNs) and Case Based Reasoning (CBR), for creating development effort estimation models using the dataset Australian Software Metrics Association (ASMA). Also, the potential of Artificial Neural Networks (ANNs) and Case Based Reasoning (CBR), for providing the basis for development effort estimation models in contrast to regression models is examined by the same author **[3]**. 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 which they have 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.

The paper proposed by **TOSUN, et.al. [1],** a novel method for assigning weights to features by taking their particular importance on cost in to consideration. Two weight assignment heuristics are implemented which are inspired by a widely used statistical technique called Principal Component Analysis (PCA).

The paper by **ELISH [6]**, 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 NASA software project dataset.

The paper based on machine learning by **Braga**, **et.al. [11]**, states the estimation of the effort together with a confidence interval. The authors have proposed to employ robust confidence intervals, which do not depend on the form of probability distribution of the errors in the training set. A number of experiments using two datasets aimed to compare machine learning techniques for software effort estimation and to show that robust confidence intervals for the effort estimation can be successfully built.

Tele:

E-mail addresses: er_prabhakar@rediffmail.com

The paper by **Martin, et. al., [10],** describes an enhanced Fuzzy Logic model for the estimation of software development effort and proposed a new approach by applying Fuzzy Logic for software effort estimates.

Genetic Algorithms (GA) are also widely used for accurate effort estimation. The paper by **BURGESS** and **LEFLEY [2],** evaluates the potential of Genetic Programming (GP) in software effort estimation and comparison is made with the Linear LSR, ANNs etc. The comparison is made on the Desharnais dataset of 81 software projects.

In **Ref. [13],** comparative research has been done by using three machine learning methods such as Artificial Neural Networks (ANNs), Case-Based Reasoning (CBR) and Rule Induction (RI) to build software effort prediction systems. The paper compares the software effort prediction systems in terms of three factors: accuracy, explanatory value and configurability.

The paper by **Bibi Stamatia, et. al., [14],** suggests the several estimation guidelines for the choice of a suitable machine learning techniques for software development effort estimation.

In **Ref. [9],** the author's presents that the one of the greatest challenges for software developers is predicting the development effort for a software system based on developer abilities, size, complexity and other metrics for the last decades. The ability to give a good estimation on software development efforts is required by the project managers. Most of the traditional techniques such as function points, regression models, COCOMO, etc, require a long term estimation process. New paradigms as Fuzzy Logic may offer an alternative for this challenge.

The paper by **Regolin, et. al. [12],** explores the use of machine learning techniques such as Genetic Programming (GP) and Neural Networks (NNs) for software size estimation (LOC).

The paper by **Parag C. Pendharkar [15],** propose a Probabilistic Neural Networks (PNN) approach for simultaneously estimating values of software development parameters (either software size or software effort) and probability that the actual value of the parameter will be less than its estimated value.

The paper by **L. Radlinki** and **W. Hoffmann [16],** analyses the accuracy of predictions for software development effort using various machine learning techniques. The main aim is to investigate the stability of these predictions by analyzing if particular techniques achieve a similar level of accuracy for different datasets. The results show that the accuracy of predictions for each technique varies depending on the dataset used.

3. Research Methodology

The Artificial Neural Network (ANN), Adaptive Neuro Fuzzy Inference System (ANFIS), and Decision Tree (DT) machine learning techniques have been used for predicting the software effort using China dataset of software projects in order to compare the performance results obtained from these models.

(A) Empirical Data Collection

The data we have used is China Dataset. This data is obtained from PROMISE (PROMISE = PRedictOr Models In Software Engineering) Data Repository [7]. The mostly used software data sets for software Effort Predictions are China, Maxwell, NASA, Finnish, Telecom, Kemerer and Desharnais.

The China Dataset consists of 19 features, 18 independent variable and 1 dependent variables. It has 499 instances correspond to 499 projects. The independent variables are ID, AFP, Input, Output, Enquiry, File, Interface, Added, Changed, Deleted, PDR_AFP, PDR_UFP, NPDR_AFP, NPDR_UFP, Resource, Dev. Type, Duration, and N_effort. The dependent variable is Effort. The descriptive statistics of China data set is appended at **Table 1**.

Set of independent variables decides the value of the dependent variable. The dependent variable is effort in this work. Some of the independent variables may be removed, if they are not much important to predict the effort, thus making the model much simpler and efficient. It has been observed from the China data set that independent variables ID and Dev. Type does not play any role in deciding the value of effort.

Hence, independent variables ID and Dev. Type have been removed.

The China data set was divided into two parts, i.e. training and testing set in a ratio of 4:1. Thus, 80% of the data was used for the purpose of training the model and remaining 20% was used for testing purpose.

MATLAB programs were developed for training and testing of various models with 16 independent variables- AFP, Input, Output, Enquiry, File, Interface, Added, Changed, Deleted, PDR_AFP, PDR_UFP, NPDR_AFP, NPDR_UFP, Resource, Duration, and N_effort for computation of dependent variable effort.

(B) Machine Learning Techniques

Machine Learning is considered as a subfield of Artificial Intelligence and it is concerned with the development of techniques and methods which enable the computer to learn. In simple terms development of algorithms which enable the machine to learn and perform tasks and activities. Over the period of time many techniques and methodologies were developed for machine learning tasks.

(1) Artificial Neural Networks (ANN)

A neural network is a simplified model of the biological neuron system. It is a massively parallel distributed processing system made up of highly interconnected neural computing elements that have ability to learn and thereby acquire knowledge and make it available for use.

McCulloch and Pitts (1943) proposed the first computational model of a neuron, namely the binary threshold unit, whose output was either 0 or 1 depending on whether its net input exceeded a given threshold.

An Artificial Neural Network is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process the information. In common with biological neural networks, ANN can accommodate many inputs in parallel and encode the information in a distributed fashion.

The most common algorithm for training or learning is known as error back-propagation algorithm. The error back-propagation learning consists of two phases: a forward pass and a backward pass, an input is presented to the neural network, and its effect is propagated through the network layer by layer. This is also called Testing Phase. During the forward pass the weights of the network are all fixed. During the backward pass or "Training Phase", 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 feedback to the system and adjusts the system parameters in a learning rule. The process is repeated until the performance is acceptable [8].

(2) Adaptive Neuro Fuzzy Inference System (ANFIS)

Adaptive Neuro Fuzzy Inference System (ANFIS) is a kind of neural network that is based on takagi–sugeno fuzzy inference system. ANFIS is an adaptive network that is functionally equivalent to fuzzy inference system. Since, it integrates both neural networks and fuzzy logic principles; it has potential to capture the benefits of both in a single framework. Its inference system corresponds to a set of fuzzy if–then rules that have learning capability to approximate nonlinear functions. Hence, ANFIS is considered to be universal approximator [17]. ANFIS is a model that maps input characteristics to input membership functions, input membership function to rules, rules to a set of output characteristics, output characteristics to output membership functions, and the output membership function to a single-valued output, or a decision associated with the output. The parameters can be automatically adjusted depending on the data that we try to model.

(i) Grid Partitioning

Grid partitioning generates rules by enumerating all possible combinations of membership functions of all inputs; this leads to an exponential explosion even when the number of inputs is moderately large. For instance, for a fuzzy inference system with 10 inputs, each with two membership functions, the grid partitioning leads to 1024 (=210) rules, which is inhibitive large for any practical learning methods. This leads to curse of dimensionality which refers to such situation where the number of fuzzy rules, when the grid partitioning is used, increases exponentially with the number of input variables.

(ii) Subtractive Clustering

Subtractive clustering is a fast one-pass algorithm for estimating the number of clusters and the cluster centers in a set of data if we don't have a clear idea how many clusters there should be for a given set of data. The cluster estimates obtained from the Subtractive clustering can be used in model identification methods (like ANFIS). It is a one-pass method to take input-output training data and generate a Sugeno-type fuzzy inference system that models the data behaviour.

(3) Decision Tree (DT)

Decision trees are powerful and popular tools for classification and prediction. The attractiveness of decision trees is due to the fact that, in contrast to neural networks, decision trees represent rules. Rules can readily be expressed so that humans can understand them or even directly used in a database access language like SQL so that records falling into a particular category may be retrieved. Decision tree is a classifier in the form of a tree structure, where each node is either:

(a) Leaf Node - Indicates the value of the target attribute.

(b) Decision Node - Specifies some test to be carried out on a single attribute-value, with one branch and sub-tree for each possible outcome of the test.

A decision tree can be used to classify an example by starting at the root of the tree and moving through it until a leaf node, which provides the classification of the instance.

(C) Evaluating the Performance of the Models

The main measures used for evaluating the performance of machine learning techniques for predicting the software effort are as follows:-

1. Sum Squared Error (SSE)

The sum squared error is defined as

$$E = \left(\sum_{i=1}^{n} (Pi - Ai)^2 \right)$$

Where Pi = Predicted value for data point i;
Ai =Actual value for the data point i;

n = Total number of data points.

If Pi = Ai, $\forall$ i = 1, 2… n; then E=0 (ideal case).

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

2. Mean Squared Error (MSE)

The mean squared error is defined as

$$E = \left(\frac{1}{n} \sum_{i=1}^{n} (Pi - Ai)^2 \right)$$

Where Pi = Predicted value for data point i;
Ai =Actual value for the data point i;

n = Total number of data points.

If Pi = Ai, $\forall$ i = 1, 2… n; then E=0 (ideal case).

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

3. Root Mean Squared Error (RMSE)

The root mean squared error is defined as

$$E = \left(\sqrt{\frac{1}{n} \sum_{i=1}^{n} (Pi - Ai)^2} \right)$$

Where Pi = Predicted value for data point i;
Ai =Actual value for the data point i;

n = Total number of data points.

If Pi = Ai, $\forall$ 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. Mean Magnitude of Relative Error (MMRE) [3, 5]

$$MMRE = \frac{1}{n} \sum_{i=1}^{n} \frac{|(Pi - Ai)|}{Ai}$$

Where Pi = Predicted value for data point i
Ai = Actual value for data point i
n = Total number of data points.

5. Relative Absolute Error (RAE)

The Relative absolute error is defined as the summation of the difference between predictive value and given value for the sample case j to that divide it by the summation of the difference between the given value and average of the given value.

The relative absolute error of individual data set j is defined as

$$Ej = \frac{\sum_{i=1}^{n} |Pij - Ai|}{\sum_{i=1}^{n} |Ai - Am|}$$

Where Pij = Predicted value by the individual data set j for data point i.
Ai = Actual value for data point;
n = Total number of data points;
Am = Mean of all Ai

For ideal case, the numerator is equal to zero and Ej = 0. Thus Ej ranges from 0 to infinity.

6. Root Relative Squared Error (RRSE)

The root relative squared error of individual data set j is defined as

$$\text{Root Relative Squared Error} = \left(\sqrt{\frac{\sum_{i=1}^{n} (Pij - Ai)^2}{\sum_{i=1}^{n} (Ai - Am)^2}} \right)$$

Where Pij = Predicted value by the individual dataset j for data point in i;

Ai = Actual value for the data point i ;

n = Total number of data points;

Am =Mean of all Ai;

For ideal case, the numerator is equal to 0 and Ej =0. Thus, the Ej ranges from 0 to infinity.

7. Mean Absolute Error (MAE)

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.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} | Pi - Ai |$$

Where Pi = Predicted value for data point i

Ai = Actual value for data point i

n = Total number of data points.

8. Correlation Coefficient

Correlation measures of the strength of the 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.

9. PRED (A)

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

$$\text{PRED (A)} = \frac{d}{n}$$

d = value of MRE where data points have less than or equal to A error.

The commonly used value of A is 25% in the literature.

4. Result Analysis

China Dataset was used to carry out the prediction of software effort estimation model. The data set was divided into two parts, i.e. training and testing set in a ratio of 4:1. Thus, 80% of the data was used for the purpose of training the model and remaining 20% was used for testing purpose.

The Artificial Neural Network (ANN), Adaptive Neuro Fuzzy Inference System (ANFIS), and Decision Tree (DT) machine learning techniques have been used for predicting the software efforts. Nine performance indices have been used in order to compare the results obtained from these models. These indices are Sum-Square-Error (SSE), Mean-Square-Error (MSE), Root-Mean-Square-Error (RMSE), Mean-Magnitude-Relative-Error (MMRE), Relative-Absolute-Error (RAE), Relative-Root-Square-Error (RRSE), Mean-Absolute-Error (MAE), Correlation Coefficient (CC), and PRED(25). The model possessing the lower values of SSE, MSE, MMRE, RMSE, RAE, MAE, and RRSE and the higher values of correlation coefficient and PRED (25) is considered to be the best among others.

MATLAB programs were developed for training and testing of various models and also for computation of performance indices. The results are tabulated in Table 4 and plotted in Figures 4.1-4.4. In these plots, the blue curve represents the curve for the actual value and red curve represents the curve for the predicted values. The more the closeness between the curves for actual and predicted output values, the lesser is the error and hence better is the model.

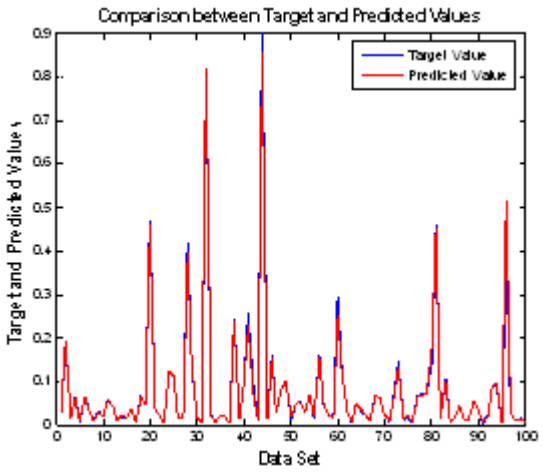

Figure 4.1(A1): Target and Predicted values using Artificial Neural Networks with One hidden layers

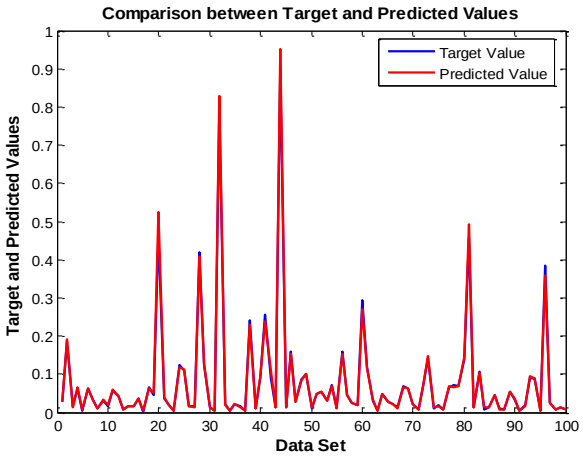

Figure 4.1(B1): Target and Predicted values using Artificial Neural Networks with Two Hidden layers

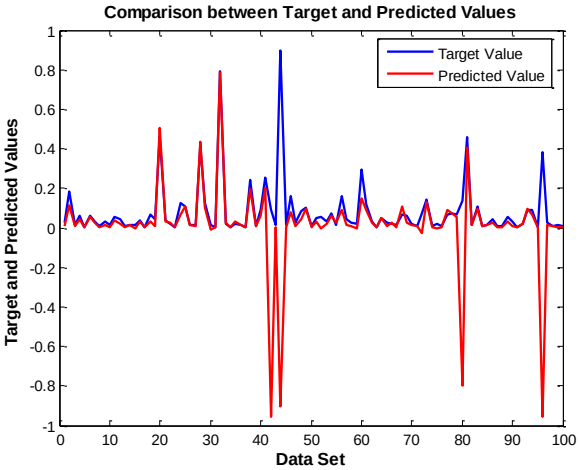

Figure 4.2(A1): Target and Predicted values using Adaptive Neuro Fuzzy Inference System (ANFIS) with Grid Partitioning

Table 1China Data Set Statistics

S N	Variables	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	13580	360	830
9	Changed	0	5193	85	291
10	Deleted	0	2657	12	124
11	PDR_AFP	0.3	83.8	12	12
12	PDR_UFP	0.3	96.6	12	13
13	NPDR_AFP	0.4	101	13	14
14	NPDU_UFP	0.4	108	14	15
15	Resource	1	4	1	1
16	Dev. Type	0	0	0	0
17	Duration	1	84	9	7
18	N_effort	31	54620	4278	7071
19	Effort	26	54620	3921	6481

Table 2 Comparison of Performance indices with various Machine Learning Techniques

S N	Performance Measures	Artificial Neural Network (ANN)		Adaptive Neuro Fuzzy Inference System (ANFIS)		Decision Tree (DT)
		One Hidden Layer	Two Hidden Layer	Grid Partitioning (with PCA)	Subtractive Clustering	
1	Sum Square Error (SSE)	0.04490	0.06440	7.09360	0.01460	0.14190
2	Mean Square Error (MSE)	0.00045	0.00064	0.07090	0.00015	0.00140
3	Root Mean Square Error (RMSE)	0.02120	0.02540	0.26630	0.01210	0.03770
4	Mean Magnitude Relative Error (MMRE)	0.07630	0.11120	0.64090	0.14500	0.13440
5	Relative Absolute Error (RAE)	0.05650	0.08710	0.84200	0.07480	0.14510
6	Root Relative Squared Error (RRSE)	0.02180	0.03120	3.43950	0.00710	0.06880
7	Mean Absolute Error (MAE)	0.00460	0.00710	0.06890	0.00610	0.01190
8	Pred(25)	0.92000	0.90000	0.37000	0.84000	0.90000
9	Correlation Coefficient	0.99350	0.99370	0.02450	0.99680	0.97100

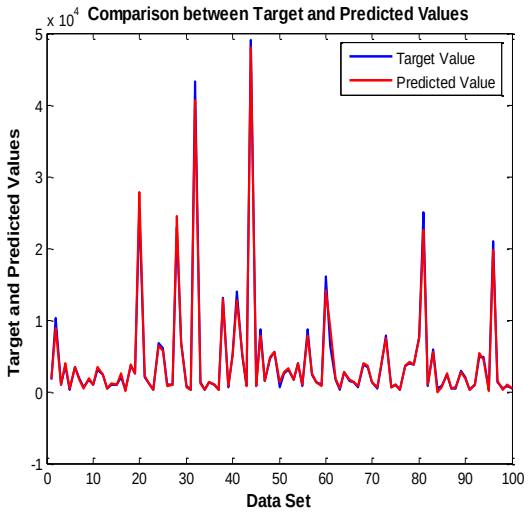

Figure 4.2(B1): Target and Predicted values using Adaptive Neuro Fuzzy Inference System (ANFIS) with Subtractive Clustering

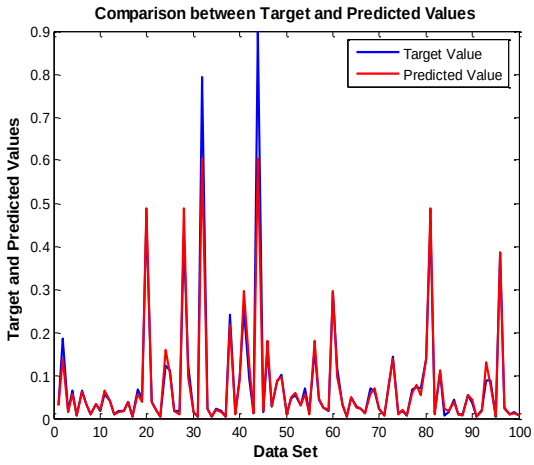

Figure 4.3: Target and Predicted values using Decision Tree

In case of Adaptive Neuro Fuzzy Inference System (ANFIS) with Grid Partitioning, the original data is too large to be handled. The dimensionality reduction has been performed based on Principal Component Analysis (PCA) in order to apply the ANFIS effectively. Because of the error involved during pre-processing, the overall error in this method is high. However, the above method of dimensionality reduction is not required in case of ANFIS with Subtractive Clustering, as a result of which the

results in this case are comparable with other methods. As shown in Table 2, the artificial neural network with one hidden layer shows the best results.

5. Conclusion

The Artificial Neural Network (ANN), Adaptive Neuro Fuzzy Inference System (ANFIS), and Decision Tree (DT) learning techniques have been used to analyze the results using China dataset for predicting software development effort.

The model with the lower Sum-Square-Error (SSE), Mean-Square-Error (MSE), Root-Mean-Square-Error (RMSE), Mean-Magnitude-Relative-Error (MMRE), Relative-Absolute-Error(RAE), Relative-Root-Square-Error(RRSE), Mean-Absolute-Error (MAE), and the higher Correlation Coefficient and PRED (25) has been considered to be the best among others. Our work will benefit to Software Testers and Managers in selecting best models for predicting software efforts in early phases of Software Development Life Cycle (SDLC).

A similar study can be carried out to predict software effort using prediction models based on other machine learning algorithms such as Genetic Algorithms (GA) and Random Forest (RF) techniques. Cost benefit analysis of models may be carried out to determine whether a given effort prediction model would be economically viable.

References:-

[1] 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.

[2] C.J. Burgess and M.Lefley, "Can Genetics Programming improves Software Effort Estimation? A Comparative Evaluation," Information and Software Technology, vol.43, pp.863-873, 2001.

[3] 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.

[4] G. R. Finnie and G.E. Wittig, "AI Tools for Software Development Effort Estimation," Proceedings of the International Conference on Software Engineering: Education and Practice (SEEP' 96).

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

[6] M. O. Elish, "Improved Estimation of Software Project Effort using Multiple Additive Regression Tree," Expert Systems with Applications, vol.36, pp. 10774-10778, 2009.

[7] *G*. Boetticher, T. Menzies and T. Ostrand , PROMISE Repository of Empirical Software Engineering data http://promisedata.org/repository, West Virginia University, Department of Computer Science, 2007.

[8] R. Malhotra, A. Jain, "Software Effort Prediction using Statistical and Machine Learning Methods," International Journal of Advanced Computer Science and Applications, vol.2, No.1, January 2011.

[9] I. Attarzadeh and Siew Hock Ow, "Software Development Effort Estimation Based on a New Fuzzy Logic Model," International Journal of Computer Theory and Engineering, Vol. 1, No. 4, pp.1793-8201, October 2009.

[10] C. L. Martin, J. L. Pasquier and Cornelio Y M and Agustin G. T., "Software Development Effort Estimation using Fuzzy Logic: A Case Study," Proceedings of the Sixth Mexican International Conference on Computer Science (ENC'05), IEEE Software, 2005.

[11] P. L. Braga , A. L. I. Oliveira and S. R. L. Meira, "Software Effort Estimation using Machine Learning Techniques with Robust Confidence Intervals," 19th IEEE International Conference on Tools with Artificial Intelligence, pp.181-185,2007.

[12] E. N. Regolin, G. A. de Souza, A. R. T. Poza, S. R. Vergilio, "Exploring Machine Learning Techniques for Software Size Estimation," Proceedings of IEEE Conference (SCCC'03), 2003.

[13] C. Mair, G.Kadoda, M. Lefley, K.P.C.Schofield, M. Shepperd and Steve Webster, "An Investigation of Machine Learning Based Prediction Systems," Empirical Software Engineering Research Group, Bournemouth University, U.K. 09 July, 1999.

[14] Bibi Stamatia and Stamelos Ioannis, "Selecting the Appropriate Machine Learning Techniques for Predicting of Software Development Costs," Artificial Intelligence Applications and Innovations, vol. 204, pp.533-540, 2006.

[15] Parag C. Pendharkar, "Probabilistic estimation of software size and effort," An International Journal of Expert Systems with Applications, vol. 37, pp.4435-4440, 2010.

[16] L. Radlinki and W. Hoffmann, "On Predicting Software Development Effort Using Machine Learning Techniques and Local Data," International Journal of Software Engineering and Computing, vol. 2, pp.123-136, 2010.

[17] Jang, Sun, Mizutani (1997) – Neuro-Fuzzy and Soft Computing – Prentice Hall, pp 335–368, ISBN0-13-261066-3.