International Journal of IT & Knowledge Management (ISSN: 0973-4414)
Volume 11 • Issue 2 pp. 34-38 Jan 2018 - June 2018 www.csjournals.com

Software Effort Estimation Analysis Using Data Mining Techniques

RavneetPreet Singh Bedi and Amardeep Singh
Department of Computer Engineering, Punjabi University, Patiala – 147002, Punjab, India
bedirps2000@yahoo.com, amardeep_dhiman@yahoo.com

Abstract: Programming exertion to the level of precise estimation is imperative for programming engineers. In the field of programming building, it is likewise an exceptionally difficult theme. Misjudged programming exertion in the early stage may cause in-genuine outcome. It impacts the calendar, as well as expands the cost. It may cause an immense shortfall. Since the greater part of the distinctive programming improvement group has approach to ascertain the product exertion, the variables influencing venture advancement are likewise fluctuate.In this paper, two data mining techniques named Bagging and Decision tree have been usedto analyse the software estimation. Results are analysed using weka tools and various performance parameters are analysed named correlation coefficient, mean absolute error, root mean squared error, relative absolute error and root relative absolute error.

Keywords: Soft Computing, Bagging classifier, Decision tree classifier, Software effort estimation.

I. INTRODUCTION

Soft computing, proposed by L.A. Zadeh in the mid 1990s for developing another age of computational savvy framework, is a strategy portrayed by the utilization of inaccurate answer for issues that has no known technique to figure the correct arrangement. Delicate registering systems depend on the human sort data preparing techniques, which includes both intelligent and natural data handling. Regular PC frameworks are useful for the former , however their capacity for the later is a long way behind that of people. Delicate figuring yields rich information portrayal (image and example), adaptable learning securing (by machine gaining from information and by meeting specialists), and adaptable learning preparation (induction by interfacing amongst representative and example learning), which empower frameworks to be built with ease. Delicate figuring has been utilized as a part of numerous applications, for example, time arrangement gauging, production network administration, movement control and most extreme power point following , and so on [2].

Soft computing incorporates neural systems,fuzzy logic, genetic calculations, mimicked strengthening, ordinal enhancement, and confusion hypothesis strategies. Some certifiable complex issues require the coordination of a few of these procedures to truly accomplish the effectiveness and precision required. In soft computing technique, the individual devices, act artificially, instead of intensely, to upgrade each other's application space. Presently, it has got the enthusiasm of numerous analysts. [1].

Software effort estimation is an essential portion of software progress. As the software increases in size and complexity the software effort approximation job gets complex, in order to deal with the complexity occurring subsequently from the last few ages. Many researchers, all over the world, try to advance new demonstrating techniques, which could deal with the varying complexity and increased size of software. The skilled approximation is the overwhelming technique while assessing programming advancement exertion. Making of endorsed programming exertion estimation models has been the fundamental accentuation of research. In this paper, we have presented few of new methods, which are for software effort estimation.

II. BACKGROUND

Chen et al. [1] presented the soft computing and three fundamental sorts of coordination innovations of soft computing and their applications. Likewise, the paper demonstrated a few uses of soft computing in multi-operator and versatile specialist individually. At last, it reasons that SCA has splendid future in applications and can unravel the vast majority of loose and unverifiable issues viably and proficiently.

Zhang et al. [2] introduced an audit of profound learning based delicate processing procedures in a few applications. The preparation of profound neural system, can be enhanced with delicate figuring strategies, for example, hereditary calculation and fluffy rationale and the execution of delicate processing techniques in past applications can be upgraded by utilizing profound neural system for its element extraction capacity.

Ganwani et al. [3] proposed a structurethat works in two phases. In first phase, papers are secured in a dataset. Initially the data profit is figured for finding features of each chronicle in dataset. In second stage, significant reports are recovered in view of inquiry question.

Jorgensen et al. [4] aimed to give a point of difference in programming estimation asked, through a considerable review of past work. The review perceived 304 programming cost estimation papers in 76 journals and describes the papers as demonstrated by investigated point, estimation approach, asked about technique, examine setting and informational collection.

Lin et al. [5] proposed a model which joins hereditary calculation (GA) with support vector machines (SVM). This paper likewise tried and confirmed our model by utilizing the chronicled information in COCOMO, Desharnais, Kemerer, and Albrecht.

Popli et al. [6] concentrated on the examination work in Agile Software change and estimation in Agile. It in like manner connected with the issues in current Agile practices thusly proposed a technique for correct cost and effort estimation.

III. PROPOSED TECHNIQUE

Bagging:A Bagging classifier is a collection meta-estimator. Bagging hysterics,similar base classifiers for every arbitrary subsets of the first dataset and afterwards total individual expectations of every sporadic subset either by voting or by averaging to shape a last expectation. A meta-estimator fundamentally can be utilized as a part of a strategy for the lessening of change of an estimator, for example, choice tree, by including randomization into the procedure and then creating an ensemble from it. If samples are drawn/created with substitution, then the method used is named as Bagging.

The classifier efficiently provides more accurate results as compared to a single classifier on the training dataset. It will provide worst results, if the dataset contains noisy data. The increased accuracy occurs because the fact that the composite model lessens the change of the individual classifiers.

As contrast with the single classifier, the enhanced classifier regularly has more prominent exactness that has been obtained from the first information. This model declines the difference of the single classifier as the outcome execution and exactness increments.

Algorithm: Bagging Technique

Input:

T, a set of t training tuples;

m, the no. of models in the ensemble;

a learning algorithm REP Tree

Output: A classification model, M∗.

(1) for i = 1 to m do // create m models:

(2) generate bootstrap sample, Ti, by sampling T with replacement;

(3) use Ti to derive a model, Mi;

(4) end for

To utilize the composite model on a tuple, X:

(1) if classification then

(2) suppose all m models classify X and return the majority vote;

(3) if prediction then

(4) suppose all m models forecast a value for X and return the average predicted value;

Given a set, D, of d tuples.For emphasis (I = 1, 2, ..., k), a preparation set, Di, of d tuples is tested with substitution from the first arrangement of tuples, D. Since testing with substitution is utilized, a portion of the first tuples of D may not be incorporated into Di, though others may happen more than once. A classifier show, Mi, is found out for each preparation set, Di. To arrange an obscure tuple, X, every classifier, Mi, restores its class expectation, which considers one vote. The classifier, M∗, checks the votes and appoints the class with the most votes to X. This calculation can be connected to the forecast of continuous values by taking the average value of each prediction for a given test tuple.

Decision Tree: A DT classifier uses a tree model to predict the class of an example. The tree consists of one root node, which is where the classifier starts. The other nodes are either leaf nodes, when they have no branches or internal nodes. The internal nodes and the root node represent a feature and a test that has to be performed on that feature. For each possible outcome of the test, the node has a branch that leads to the next node. The leaf nodes eventually indicate a class.

A DT classifier predicts the class of an example by following a path from the root node of the tree until it encounters a leaf node. At every node (except leaf nodes) a test is performed to choose which branch to follow to the next node. When a leaf node is encountered, the classifier predicts the class that the leaf nodes indicates.

Decreases Error Pruning Tree Classifier is a brisk decision tree learning figuring and relies upon the rule of calculating the data get with entropy and constraining the oversight rising up out of difference. REP Tree applies backslide tree method of reasoning and creates relapse trees in balanced emphasess. A while later it picks finest one from all delivered trees. This computation assembles the relapse/decsion tree using fluctuation and data pick up. In like manner, this count prunes the tree utilizing minimized blunder pruning with back fitting technique. At the begin of the model arranging, it sorts the estimations of numeric qualities once. RepTree uses the relapse tree method of reasoning and makes different trees in variousemphasess. After that it picks finest one from all conveyed trees. That will be considered as the authority. In pruning the tree, the measure utilized is the mean square bungle on the figures made by the tree. Fundamentally, Reduced Error Pruning Tree ("REPT") is lively decision tree learning and it accumulates a decision tree in context of the data get or diminishing the difference. REP Tree is a snappy choice tree student which gathers a choice/relapse tree utilizing data pick up upas the part model, and prunes it using decreased blunder pruning. It just sorts esteems for numeric qualities once. Missing values are managed to utilize C4.5's strategy for utilizing fragmentary occurrences.

Algorithm: Decision Tree Procedure
Input: T//Training data
Output T_d //Decision tree
DTBUILD (*T)
{
T_d=φ;
T_d= Generate root node and label with splitting attribute;
T_d= Add arc to root node for all split predicate and label;
For every arc perform
T= Database generated by performing splitting predicate to T;
If terminate point achieved for this path, then
T_d'= generate leaf node and label with appropriate class;
Else
T_d'= T_dBUILD(T);
T_d= add T_d' to arc;
}
While building a choice tree, J48 overlooks the missing esteems i.e. the incentive for that thing can be anticipated in view of what is thought about the property estimations for alternate records. The key thought is to part the information into run in view of the quality esteems for that thing that are recognized in the preparation test.

IV. EXPERIMENTAL RESULTS

Performance Evaluation Criteria

- **Correlation Coefficient:** Two factors correlation coefficient in an informational index equivalents to their covariance independent by the outcome of their individual

- **Mean Absolute Error:**MAE calculates the regular noteworthiness of the errors in an arrangement of forecasts, without contemplating their bearing. It's the regular over the check example of the supreme contrasts among forecast and genuine analysis where every individual distinction have approach weight.
$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$..... Eq (13)

- **Root mean squared error (RMSE)**: RMSE is a quadratic scoring that calculates the normal extent of the blunder. It's the square foundation of the normal of squared contrasts amongst forecast and genuine perception.
$MAE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$ ..... Eq (4)

- **Relative Absolute Error**: The relative absolute error takes the aggregate total mistake and standardizes it by separating the aggregate total mistake of the straightforward indicator.

Scientifically, the relative total blunder $E_i$of an individual program i is assessed by the condition:
$E_i = \frac{\sum_{j=1}^{n}|P_{ij}-T|}{\sum_{j=1}^{n}|T_j-\bar{T}|}$..... Eq (14)

Where

$P_{(ij)}$: Value predicted by the individual program *i* for sample case *j*

T_j: Target value for sample case *j*; and $\bar{T}$ is given by the formula:

$\bar{T} = \frac{1}{n}\sum_{j=1}^{n} T_j$..... Eq (15)

- **Root Relative Squared Error:** The relative squared blunders takes the aggregate squared mistake and standardizes it by setting apart by the aggregate squared blunders of the straightforward indicator. Mathematically, the root relative squared errors E_i of an man or woman program i is evaluated with the aid of the equation

$E_i = \sqrt{\frac{\sum_{j=1}^{n}(P_{ij}-T)^2}{\sum_{j=1}^{n}(T_j-\bar{T})^2}}$..... Eq (16)

Table 1: Classification Results Using Cross Validation Model Taking 10 Folds (Prediction Parameter - Months)

Name	Correlation Coefficient	Mean Absolute Error	Root Mean Squared Error	Relative absolute Error	Root Relative Square Error
Bagging Classifier	0.8712	3.5045	6.7541	36.14	51.93
Decision Tree Classifier	0.6787	6.1516	9.5567	63.44	73.48

From table 1, it is clearly shown that the results have been compared with the various classification method such as linear regression classifier, multilayer perceptron neural network classifier.

International Journal of IT & Knowledge Management (ISSN: 0973-4414)
Volume 11 • Issue 2 pp. 34-38 Jan 2018 - June 2018 www.csjournals.com

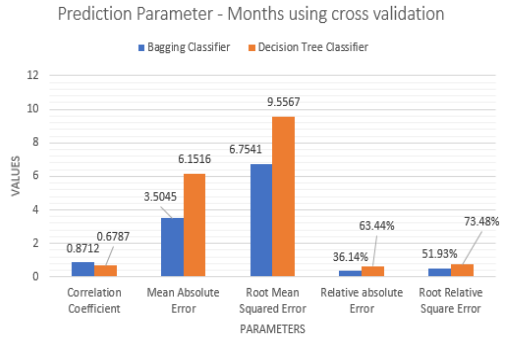

Figure 1: Graphical analysis of cross validation model (parameter – months).

The figure 1, shows the graphical comparison of various classification method such as bagging classifier, decision tree classifier in term of mean absolute error, root mean squared error, correlation coefficient, relative absolute error and root relative square error.

Table 2: Classification Results Using Percentage Split Model Taking 70% Data as Training And 30% as Testing (Prediction Parameter - Months)

Name	Correlation Coefficient	Mean Absolute Error	Root Mean Squared Error	Relative absolute Error	Root Relative Square Error
Bagging	0.9624	2.5734	3.1805	31.11	33.54
Decision Tree	0.8332	4.4196	5.6371	53.42	59.44

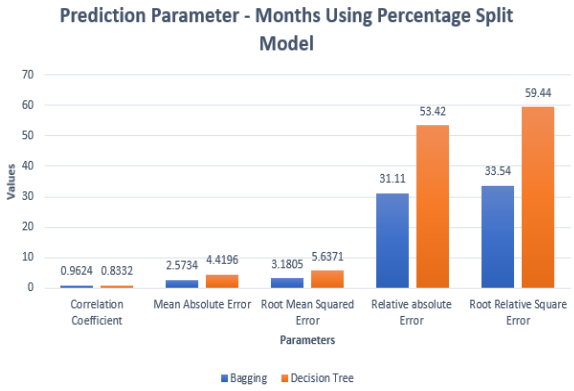

Figure 2: Graphical analysis of percentage split model (parameter – months).

The figure 2, shows the graphical analysis of percentage split with various classification method such as bagging classifier, decision tree classifier in term of mean absolute error, root mean squared error, correlation coefficient, relative absolute error and root relative square error.

Table 3: Classification Results Using Cross Validation Model Taking 10 Folds (Prediction Parameter Efforts x Months)

Name	Correlation Coefficient	Mean Absolute Error	Root Mean Squared Error	Relative absolute Error	Root Relative Square Error
Bagging	0.6427	16455.0813	55628.8066	50.28	80.83
Decision Tree	0.8857	19577.2567	32354.7616	59.82	47.01

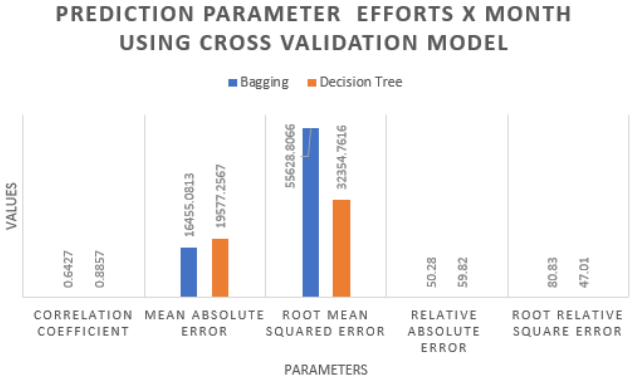

Figure 3: Graphical analysis of cross validation model (Prediction Parameter Efforts x Month)

The figure 3 shows the graphical analysis of cross validation model with various classification method such as bagging classifier, decision tree classifier in term of mean absolute error, correlation coefficient, relative absolute error, root mean squared error, and root relative square error. The red colour illustrates multilayer perceptron and green colour illustrates the bagging classifier

Table 4: Classification Results Using Percentage Split Model Taking 70% Data as Training And 30% as Testing (Prediction Parameter Efforts x Month)

Name	Correlation Coefficient	Mean Absolute Error	Root Mean Squared Error	Relative absolute Error	Root Relative Square Error
Bagging	0.878	9236.2968	14516.2754	38.80	57.84
Decision Tree	0	15717.555	18294.838	66.03	72.90

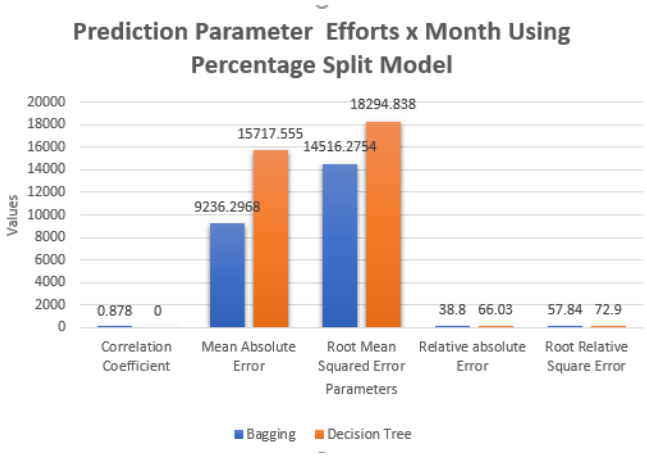

Figure 4: Graphical analysis of percentage split model (Prediction Parameter Efforts x Month).

Figure 4, shows the graphical analysis of percentage split model with various classification methods such as bagging classifier, decision tree classifier in term of correlation coefficient, root mean squared error, mean absolute error, and root relative square error relative absolute error.

V. CONCLUSION

Finding the most imperative purpose behind the product accomplishment failures has been the subject of numerous expertsabout a decade ago. As per the findings of the experts, , the main driver for programming venture disappointments is erroneous estimation in beginning times of the task. So, presenting and concentrating on the estimation techniques appear to be fundamental for accomplishing the exact and dependable estimations. Error and accuracy computation in Software development projects is a thoroughly researched area in software engineering. Researchers have used a number of techniques to estimate software efforts errors and accuracy. Some of the existing techniques from the well-known data mining tool weka have been studied to analyse the error. In this paper, bagging classifier and Decision tree classifier is analysed and compared on the basis of Months and Efforts * Months for the PROMISE Dataset.

REFERENCES

[1] Peiyou Chen, Jing Zhang, “Research on Applications of Soft Computing Agents”, IEEE International Symposium on Intelligent Information Technology Application Workshops, 2008, pp. 259-262.

[2] Jian Zhang, Chongyuan Tao, Pan Wang, “A Review of Soft Computing Based on Deep Learning”, IEEE International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration, 2016, pp. 136-144.

[3] JuhiGanwani, JayantGadge, “Framework for searching research papers in dataset using soft computing approach”, IEEE, 2017.

[4] MagneJørgensen, Martin Shepperd, “A Systematic Review of Software Development Cost Estimation Studies”, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 33, NO. 1, JANUARY 2007, pp. 33-53.

[5] Jin-Cherng Lin, Chu-Ting Chang and Sheng-Yu Huang, “Research on Software Effort Estimation Combined with Genetic Algorithm and Support Vector Regression”, IEEE International Symposium on Computer Science and Society, 2011, pp. 349-352.

[6] RashmiPopli, NareshChauhan, “Cost and Effort Estimation in Agile Software Development”, International Conference on Reliability, Optimization and Information Technology - ICROIT 2014, India, Feb 6-8 2014, pp. 57-61.

[7] Bishop, C. M. (2006).Pattern recognition. Machine Learning, 128, 1-58.

[8] Heiat, A. (2002). Comparison of artificial neural network and regression models for estimating software development effort. Information and software Technology, 44(15), 911-922.

[9] Braga, P. L., Oliveira, A. L., Ribeiro, G. H., &Meira, S. R. (2007, August). Bagging predictors for estimation of software project effort.In Neural Networks, 2007.IJCNN 2007. International Joint Conference on (pp. 1595-1600). IEEE.

[10] Srinivasan, K., & Fisher, D. (1995). Machine learning approaches to estimating software development effort. IEEE Transactions on Software Engineering, 21(2), 126-137.