Australian Journal of Basic and Applied Sciences, 6(7): 312-317, 2012
ISSN 1991-8178

A Comparative evaluation of Software Effort Estimation using REPTree and K* in Handling with Missing Values

[1]K. Suresh Joseph, [2]T. Ravichandran

[1]Department of Computer Science, Pondicherry University
[2]Hindhusthan Institute of Technology, Coimbatore

Abstract: One of the most important tasks of a project manager in software development is to produce accurate schedule and effort estimations. Missing data are frustration to a certain extent universally but especially in effort estimation it is a greater hindrance. The popular algorithmic estimation models like Boehm's COCOMO and other models require accurate inputs for estimation of effort. It is very difficult to obtain such information at the early development stage of a software project. Imputation is one of the key strategies that researchers employ to fill in missing data in such a dataset. Imputed data is used in order to find a suitable replacement for better accurate analyses. These issues led to introduction of non-algorithmic techniques. With the Computational Intelligence (CI) appropriate information is imputed in the position of vague and imprecise data. This information if it is neglected may yield blurred and opaque data which is a hindrance in effort estimation. The objective of this paper is to focus on missing values in effort estimation with soft computing techniques.

Key words: Computational Intelligence, Imputation, Software development

INTRODUCTION

IN the past decade we find many machine learning algorithms were developed and found to be dominating in comparison with other statistical methods. Statistical techniques involved in finding effort fails or under estimating hence it will have adverse effects on the product. This will lead to failure in project planning. Machine learning models are suitable when vague and incomplete information is to be accounted for. The machine learning algorithms that were used are probabilistic. In majority cases we find that the effort estimation data sets have approximately 50% of missing values i.e is incomplete hence processing with such data will yield a biased result. With exposure to different types of computations most likely all studies show that researchers are more similar in showing responses with missing values. Hence imputation maintains lesser variance i.e. is it has more accurate quantification of variance.

Imputation is the replacement of missing values in data (Little, R.J.A., D.B. Rubin, 1987) with assessment techniques and assumptions. The major problem that hampers useful application of imputation methods is bias, when an estimator's expectation gets differed from the quantity being estimated. The deviation is dangerous. Basically there are two flavors in imputation namely Single imputation and multiple imputation. The volume of data missing decides the kind of employability whether it is single or multiple imputation. Single Imputation involves somewhat less computation and multiple imputation is crucial in majority cases. Hence imputation maintains lesser variance i.e. is it has more accurate quantification of variance.

A good imputation method should be plausible and consistent and has to reduce bias while preserving the relationship between items within the data, and has to be evaluated for bias and precision. The ability to estimate accurately about the time taken for successful completion of the software project, cost taken for a project to come in to its successful existence is a serious problem for software engineers.

Software Engineering has been one among the hot areas in which impact of imputation is of greater demand. More accurate knowledge of software effort which comprises of time and cost estimates enables project managers exploit their resource ingeniously.

Why data has to be imputed?

If significant amount of missing or incomplete data are often found in such software engineering data sets it is tedious to work with it and it is prone to error. They cannot neglect such missing data and cannot work with partial information which is at hand.

What are the associated problems to be addressed with Missingness?

Three problems that are linked with missing values namely Loss of efficiency, Barriers in analyzing and handling the data and biased results because of incomplete data set.

Corresponding Author: K. Suresh Joseph, Department of Computer Science, Pondicherry University
E-mail: Ksjoseph.csc@gmail.com

Whether imputation is a better model?

Definitely imputation model is a compatible one and is affluent enough which confirms based on the associations and relationships among the other variables in the data set. Imputation imposes models like probability model, Decision tree models, neural models on the complete data set and also other models.

This paper provides a performance comparison of decision tree methods and neural network methods for handling missing values in effort prediction. The paper is organized as the follows. Section II, describes the background knowledge on effort prediction and related work; Section III. Explores the Evaluation and performance analysis used in this work and Result discussions there on; Section IV concludes the outcome of this work.

Background:

Review of COCOMO 81:

COCOMO 81 (Constructive Cost Model) is an empirical estimation scheme proposed in 1981 (Parag C. Pendharkar, 2005). It is a model for estimating effort, cost, and schedule for development of software projects. This parametric model certainly provides better results. The other approaches are mathematical and analogous. If the regression norm is produced with the help of past data it is referred as analogy method. Data mining models in recent years also carry competitive results to the people in the managerial capacity (Rodriguez *et al*., 2006). These data were analyzed to discover a set of formulae that were the best fit to the observations. These formulae link the size of the system and Effort Multipliers (EM) to find the effort to develop a software system. In COCOMO 81, effort is expressed as Person Months (PM) and this can be computed as

$$PM = a * Size^{b} * \prod_{i=1}^{15} EM_{i} \quad (1)$$

where "a" and "b" are the domain constants in the COCOMO model. It has 15 effort multipliers. This estimation scheme accounts the experience and data of the past projects, which is extremely complex to understand and apply the same.

Though this cognitive approach has its own uncertainty, but its influence will be comparatively increase the accuracy of the estimation schemes because of incomplete information. In this perspective, the work presented in this paper proposed a new approach for handling missing information which improves the performance of the COCOMO effort estimation scheme.

Table I: Effort Estimation Approaches.

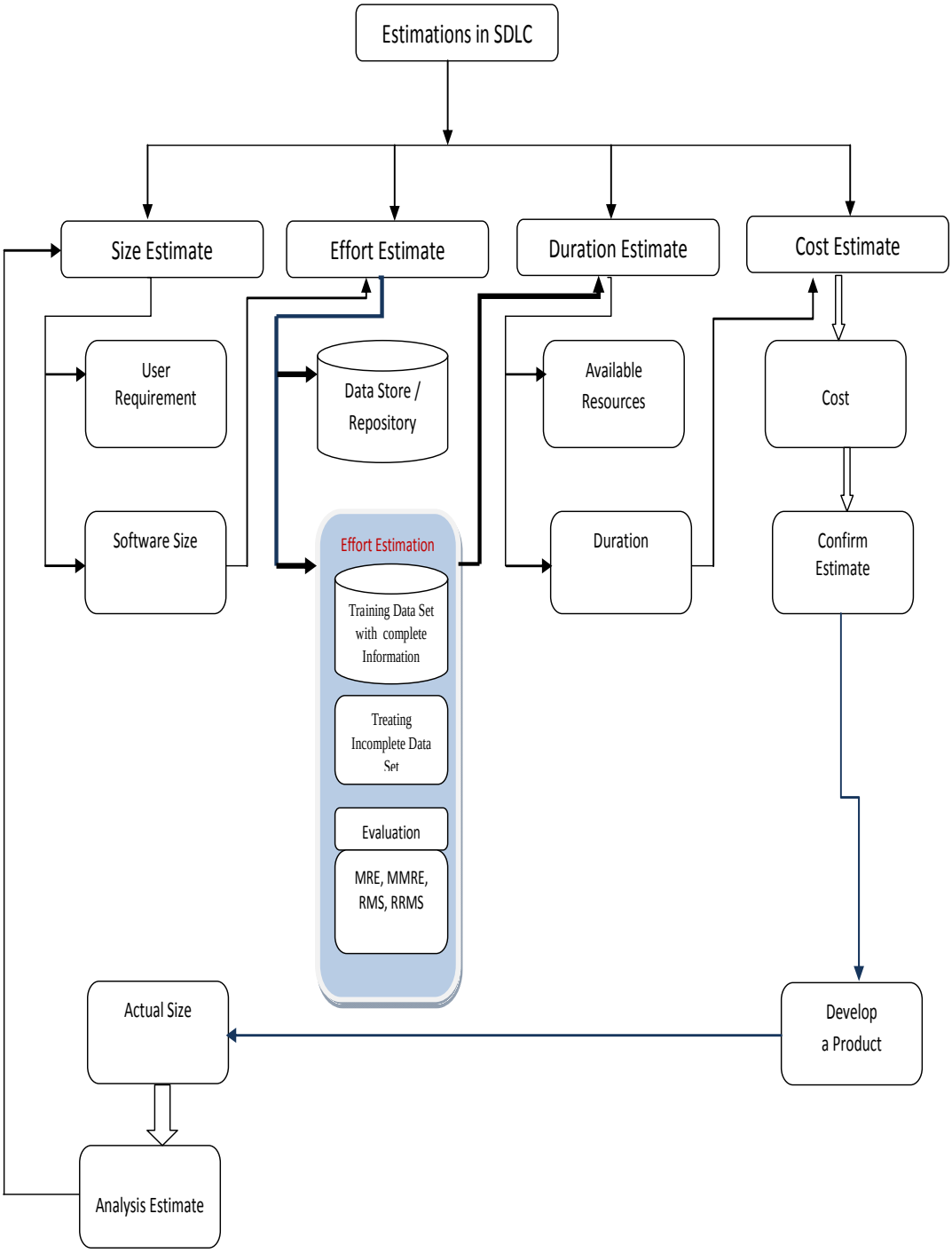
Approach	Category of Estimation	Models / Tools
Analogy – Based Estimation	Formal Estimation	ANGEL – Shepperd and Schofield
WBS – Based (Bottom Up) Estimation	Expert Estimation	MS Project, Company Specific Activity Templates
Parametric Models	Formal Estimation	COCOMO, SLIM
Size – Based Estimation	Formal Estimation	FP Analysis, Use case Analysis, Story Points-Based Estimation in Agile Software Development
Group Estimation	Expert Estimation	Planning Poker, Wideband Delphi
Mechanical Combination	Combination – Based	Average of an Analogy – Based and a Work Breakdown Structure – Based
Judgmental Combination	Combination – Based	Expert Judgment Based on Estimation From a Parametric Model and Group Estimation

K*:

K* is an instance based learner (Witten, I.H. and E. Frank, 2005) which uses such a measure and examine the performance on a range of problems. Instance based learners classify an instances by comparing it to a database of pre-classified examples. The implementation of an instance – based classifier which uses the entropic distance measure which provides a unified approach to dealing with handle missing value problems. It treat missing values as s separate value, treat them as maximally different replace them with average value.

Rep Tree:

It is fast decision tree learner it is also based on c4.5 algorithm and can produce classification of regression trees. It builds a decision tree using information gain/variance and prunes it using reduced-error pruning. It constructs a decision tree by evaluating the forecaster attributes against a quantitative goal attribute. This constructing process is accomplished by using variance reduction to derived balanced tree splits and minimizes error corrections. This process is recursively repeated to simplify and prune the tree by comparing it against already established trees. This process is repeated until a stopping criterion is satisfied.



Evaluations and Performance Analysis

Data Description:

The data used in the analysis of the present work is from COCOMO'81 published by Boehm in "Software Engineering economics" (Boehm, B.W., 1981). WEKA (http://www.cs.waikato.ac.nz/ml/weka.) is a collection of many open source machine learning algorithms and data mining includes pre-processing on data, Classification, clustering, rule extraction and association. It was created by the researchers at the University of Waikato in New Zealand in JAVA.

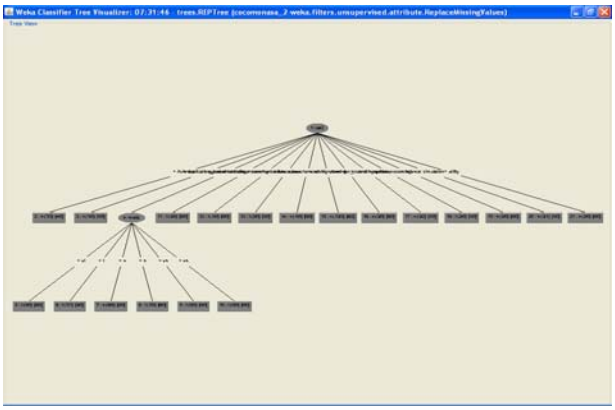
Table II: Performance Evaluation Criteria Table

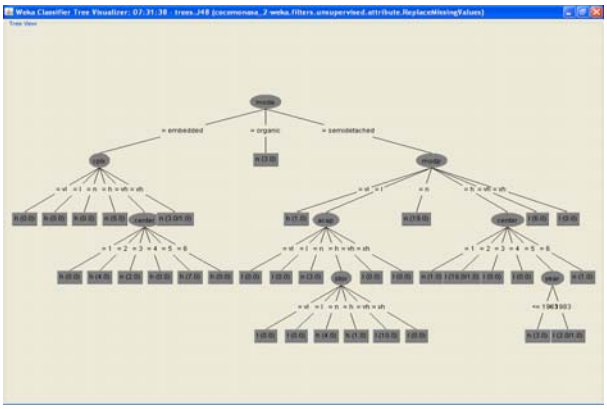
Criteria	Formula	Description
MAE	$MAE = \frac{\sum \lvert E_{ACT} - E_{PRED} \rvert}{n}$	MAE measures of how far the estimates are from actual values. It could be applied between any two pairs , where one set is actual and the other is predicted or estimate
MMRE	$MMRE = \frac{100}{N}\sum_{1}^{N}\frac{\lvert predicted_i - actual_i \rvert}{actual_i}$	The Mean Magnitude of the Relative Error is the average percentage of the absolute values of the relative errors over an entire data set.
RMS	$RMS = \left[\frac{1}{N}\sum_{1}^{N}\frac{(predicted_i - actual_i)^2}{actual_i}\right]^{0.5}$	Relative Mean Square is the square root of the arithmetic mean of the squares of the original values
RRMS	$RRMS = \frac{RMS * N}{\sum_{i}^{N} actual}$	Relative Root Mean square is the ratio between RMS and the average of the actual effort

Results:

REPTree (With missing Values)		
Correctly Classified Instances	78	83.871 %
Incorrectly Classified Instances	15	16.129 %
Kappa statistic		0.7445
Mean absolute error		0.0808
Root mean squared error		0.216
Relative absolute error		37.1458 %
Root relative squared error		66.0763 %
Coverage of cases (0.95 level)		93.5484 %
Mean rel. region size (0.95 level)		28.853 %
Total Number of Instances		93

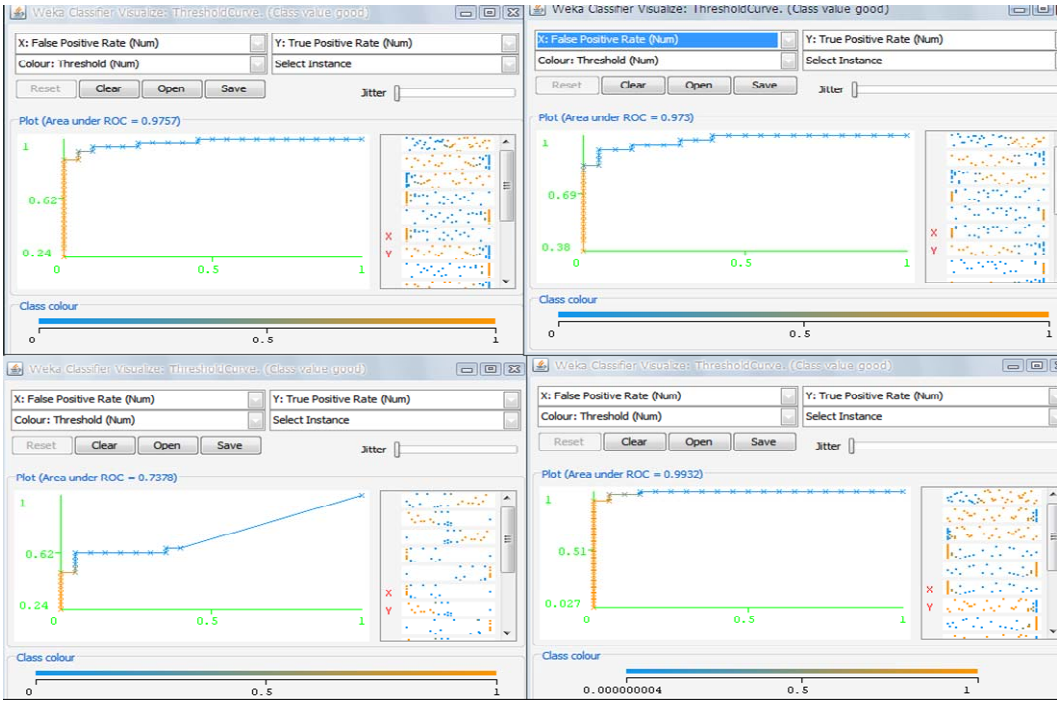
REPTree (After treating with missing values)		
Correctly Classified Instances	76	81.7204 %
Incorrectly Classified Instances	17	18.2796 %
Kappa statistic		0.7111
Mean absolute error		0.0811
Root mean squared error		0.2139
Relative absolute error		37.2845 %
Root relative squared error		65.4299 %
Coverage of cases (0.95 level)		94.6237 %
Mean rel. region size (0.95 level)		29.5699 %
Total Number of Instances		93





K* (After treating missing values)		
Correctly Classified Instances	86	96.4731 %
Incorrectly Classified Instances	7	7.5269 %
Kappa statistic	0.8828	
Mean absolute error	0.0266	
Root mean squared error	0.1497	
Relative absolute error	12.2241 %	
Root relative squared error	45.7865 %	
Coverage of cases (0.95 level)	93.5484 %	
Mean rel. region size (0.95 level)	17.9211 %	
Total Number of Instances	93	

Models	Accuracy
K-Star (Ignoring missing values)	66.6%
K-Star (Handling missing values)	96.4%



Performance Measures:

The following evaluation criterion is used for validation of estimation capability. Experiments were done by taking original data from the COCOMO dataset. The software development effort is obtained from COCOMO as well as from REP tree and K* using WEKA tool. After analyzing the results attained by means of applying COCOMO and other models taken in this study, it is observed that the effort estimation of the proposed model is giving more precise results than the other models.

Conclusion:

Evaluation of these imputational methods fits appropriate data in effort estimation of software engineering data sets and empirical analysis of the same reveals that K * is well suited for such estimation problems. The focused research in this paper is to sharpen up the COCOMO estimation scheme and to provide an improved estimation scheme even with partial or incomplete data. The results taken from WEKA reveals that treating or imputing proper data in the position of missing values gives an enhanced accuracy of 96.4%, if the missing values is ignored it yields accuracy of only 66.6%, if missingness is ignored it shows detrimental effects. The obtained experimental results conclude K* enhances the effort precision.

REFERENCES

Ali Idri, Abdelali Zakrani and Azeddine Zahi, 2010. "Design of Radial Basis Function Neural Network for software effort estimation", IJCSI.

Boehm, B.W., 1981. "Software Engineering Economics", Prentice-Hall.

http://www.cs.waikato.ac.nz/ml/weka.

Little, R.J.A., D.B. Rubin, 1987. "Statistical analysis with missing data", Wiley series in probability and statistics, 1st edn. Wiley, New York.

Parag C. Pendharkar, Girish H. Subramanian and James A. Rodger, 2005. "A Probabilistic Model for Predicting Software Development Effort." IEEE Transactions on Software Engineering, 31(7).

Witten, I.H. and E. Frank, 2005. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco, CA, 2nd edition.