International Core Journal of Engineering Volume 6 Issue 6, 2020
ISSN: 2414-1895 DOI: 10.6919/ICJE.202006_6(6).0028

Software Project Effort Estimation with Machine Learning Algorithm

Sonin-Erdene

College of Information Science and Engineering, Henan University of Technology, China.

soniosdek@yahoo.com

Abstract

The efforts are to be estimated before beginning with the development of software. This technique is known as effort estimation. Various models are proposed to measure the efforts accurately. The proposed work includes estimating the efforts of software using hybrid technique involved in previous models. COCOMO is constructive cost model and is considered as the most accurate model for effort estimation. Another method to measure the efforts is function point. It basically measures the size of the project that further helps to calculate efforts. In this work, a hybrid formula used that depends upon the values of cost drivers. The generated result will estimate closer efforts than COCOMO. The tool used for the implementation is MATLAB. IVR dataset is being used after taking authentication from the website. This report presented effort estimation models, COCOMO, COCOMO 2, Bailey-Baisly model, Halstead, Bee Colony Optimization Algorithm and the proposed one. In this there 47 projects with defined values of cost drivers. The efforts of each model are being calculated and their MMRE is calculated. The result shows that the hybrid model is derived by using SVM algorithm provides more accurate results than the other estimation models.

Keywords

Effort estimation, COCOMO, COCOMO2, Software effort estimation, Software development lifecycle, Software project estimation.

1. Introduction

1.1 Introduction to Software Project Effort Estimation

To ensure their competitiveness, it is important for the software developers to deliver a software product that is in the budget as well as time. So, for meeting the software requirements of users, it is important for such developers to manage their projects adequately. The important activity in software project management that estimates the amount of effort depending upon the person-months needed to design new software is known as Software Development Effort Estimation (SDEE). An important part of software development and software project management is the SDEE. Effort estimation is defined here as the approach through which the most realistic effort required to complete a software development is estimated [1]. Through a process for predicting the software effort needed to express in person-hours, the estimation is defined based on the incomplete inputs, uncertainty and the noisy inputs.

The effort is described using the ABC estimation strategy in the generic procedure through which the software estimation is performed. It can estimate any of the quantitative parameters of any domain. Following are the three basic steps applied in the generic estimation process:

Step A: Quantification: This step calculates the complexity or size of system model that is to be designed. For calculating this value, a software complexity metric like the System Meter is used that

ISSN: 2414-1895 DOI: 10.6919/ICJE.202006_6(6).0028

is proposed on the system model [2]. The expression called the "metric of size" is used such that the sizes of such kinds of software metrics can be defined. The expression is otherwise known as the "effort inducing property" since adapting the notion of size from the non-software domains is not necessarily relevant to the effort.

Step B: Estimation: The estimation value for which the approximation function is applied to the predictor value is calculated in this step. The function can be represented using this estimation function.

Step C: Adaptation: After achieving the statistically derived estimation "e", the process is considered as incomplete. The activities that are to be performed are involved in this non-standard process model. The last step of result model is to adapt and interpret the resultant value.

1.1.1 Software Effort Estimation Techniques

There are three important categories in which the various effort estimation techniques are divided which are expert judgment, algorithmic estimation [3], and analogy based estimation. Following is the brief explanation for all these techniques:

A. Algorithmic methods: On the basis of mathematical models which provide cost estimations on the basis of function of different variables that are known as major cost factors are known as the algorithmic methods.

1), Function Points Analysis: A method with high business significance that helps in measuring size is known as FPA. The estimation of lines of code for software is the major factor on which this method relies. The effort for a software project can be estimated at the early stage when the requirements are known but there is no specification about the details of implementation through FPA. Most of the software projects are built up in such a way.

2), COCOMO (Constructive Cost Model): A regression model that is based on Line of Code and is also known as a procedural cost estimation model for the software projects is known as COCOMO. The different parameters that are associated with the designing of a project are predicted reliably through this process. For predicting the cost estimation at various levels on the basis of accuracy and correctness, various COCOMO variants or models have been designed over the years. The value of constant to be used in subsequent calculations can be determined by the characteristics of the models that are applied to various projects.

B. Expertise Based Estimation: Since there is no substantial evidence in favor of the use of estimation models and there are scenarios in which expert estimates can be expected to be more accurate in comparison to formal estimation models, this is most commonly used estimation method for software projects. In cases where the data to be found and the gathering requirements are limited, this method is applied. The basic issue of this method is the consultation.

C. Learning Oriented Techniques: For developing a software estimation model, the prior and current knowledge are used by these methods. The most common examples of learning oriented techniques are neural network and analogy estimation.

1), Neural Networks: Based on the principle of learning, the neural networks are designed. The neurons, the interconnection structure and the learning algorithm are the three important components of the neural networks [4]. The back propagation trained feed forward networks are used by most of the software models that are designed using the neural networks.

2), Analogy Based Estimation (ABE): Due to its ability to derive solutions by imitating the problem solving approach of human, ABE is used commonly as an alternative for expert judgment and algorithmic cost estimation.

D. Price-to-win: The best price to win the project is known to be the estimated software cost. Instead of the functionality of software, the estimation relies on the budget of customer.

E. Bottom-up approach: Here, an estimate is produced for the overall system by estimating every component of software system separately. For producing an estimate of overall system, the results

ISSN: 2414-1895 DOI: 10.6919/ICJE.202006_6(6).0028

are then aggregated. An initial design must be in place which is the requirement for this approach. So, it indicates the way in which various components are generated from the decomposition of one system.

F. Top-down approach: Opposite to the bottom-up method this approach helps in estimating the overall cost of the system with the help of either algorithmic or non-algorithmic methods and the global properties. Further, the cost can be divided into different components. In case when cost estimation is needed at early stage, this approach provides the best outputs.

1.2 Machine Learning Algorithms

With the aim of enabling machines to perform tasks in a skillful manner using intelligent software, machine learning is evolved that is a branch of artificial intelligence [5]. The intelligent software that is used to design machine intelligence includes statistical learning methods. It is important to have a connection with the database since data is required by machine learning algorithms to perform learning. Based on the nature of learning the "signal" or the "feedback" required for a learning system, the machine learning tasks are broadly categorized among three types:

a. Supervised learning: The example inputs and their desired outputs as given by a "teacher" are included in the computer. A general rule that maps the inputs to outputs is learned as a major objective of this type of learning.

b. Unsupervised learning: The structure in its input is found on its own in this type of learning since no labels are given to the learning algorithm.

c. Reinforcement learning: A dynamic environment, in which a certain objective is performed without a teacher telling it explicitly if the goal to be achieved is closer or not, interacts with a computer program is known as reinforcement learning.

2. Research Background

Mustafa Hammad, et.al (2018) made use of several machine learning algorithms for constructing software effort estimation models using software features. Different machine learning algorithms with real-time software efforts were evaluated on an openly available dataset [6]. The evaluation results were used for the early prediction of real-time effort from the software features. The tested results revealed that it was possible to implement machine learning algorithm for making prediction about the software effort with less MAE value. The SVM (Support Vector Machine) classifier provided lowest MAE value of 2.6. Some other machine learning algorithms with more datasets would be evaluated in nearby future. Also, the level of prediction accuracy could be increased by implementing filters in the data of software features. For this purpose, it is required to understand the responsiveness of the prediction model for all datasets.

Suyash Shukla, et.al (2019) reviewed Multi Layer Perceptron (MLPNN) and its ensemble models for improving the performance of software effort estimation practice [7]. Initially, the designing of MLPNN (Multi Layer Perceptron), Ridge-MLPNN, Lasso-MLPNN, Bagging-MLPNN, and AdaBoost-MLPNN models had been carried out. After this, these models were compared in terms of their performances. The comparison was based on R^2 score for getting the best fitted model within the used dataset. The achieved review results achieved demonstrated that the AdaBoost-MLPNN approach provided R^2 score of 82.213%. This was the maximum score achieved among all reviewed models.

Suyash Shukla, et.al (2019) used different data preparation methods for improving the quality of the used data set. Initially, the most significant features in the Desharnais data set were detected. Afterward, MLPNN (Multi Layer Perceptron) approach was implemented on the compact data set. The main aim here was to get more accurate results of software effort estimation [8]. The recommended technique mainly considered four machine learning models. These models included LR (Linear Regression), SVM (Support Vector Machine), KNN (K Nearest Neighbor), and MLPNN (Multi Layer Perceptron). Therefore, the most feasible model in terms of software effort estimation

ISSN: 2414-1895 DOI: 10.6919/ICJE.202006_6(6).0028

had been detected by comparing the results of all used models. The tested results revealed that MLPNN model achieved maximum R^2 value of 0.79380.

Seiji Fukui, et.al (2018) made attempts to regulate the kurtosis and the skewness of project feature variables. This task was necessary for getting improved fitting of software estimation models. The recommended carried out transformed variables in logarithmic form. Afterward, the kurtosis and skewness conversion was carried out to fill the gap between variable distribution and the normal distribution [9]. Three industry data sets and linear regression models with threefold cross validation had been utilized in this work for evaluating the efficiency of the recommended technique. The evaluation outcomes demonstrated that the models using recommended technique provided better results in terms of the integrity of fitness and the estimation accuracy than log-log regression approach.

Ahmed BaniMustafa, (2018) suggested the use of three machine learning algorithms for carrying out prediction task. These algorithms were implemented on a preprocessed COCOMO NASA standard data set [10]. Five folds cross -validation were used for the testing of the generated models. These models were evaluated in terms of different performance metrics. These metrics include classification accuracy, precision, recall, and AUC. Afterward, the comparison of estimation results was carried out with the estimation results of COCOMO model. In contrast to COCOMO model, all machine learning algorithms achieved better results. Both Naïve Bayes as well as Random Forest algorithms provided optimal results among all three machine learning algorithms. Naïve Bayes classifier performed better than other algorithms in terms of ROC curve and Recall score. On the other hand, Random Forest classifier achieved better Confusion Matrix. This classifier performed better in terms of both Classification Accuracy as well as Precision measures than other classifiers.

3. Research Status

3.1 Problem Statement

The software project effort is the major issue of software development. The software project effort estimation depends upon the number of lines. Many software effort estimation models have been designed in the previous years. The most common software effort estimation models are COCOMO, Bailey, Halstead, COCOMO-II etc. The models like COCOMO, Bailey, Halstead etc are totally based on the number of lines. The projects which are designed in today era are very large in size and also number of functions is very high. In such type of projects, it is really very difficult to estimate exact number of lines. Due to inaccurate estimation of number of lines leads to inaccurate estimation of project efforts. The technique needs to be designed which can accurately estimate project efforts

4. Research Content and Methods

This research work is related to software project effort estimation. The COCOMO is the most common and popular model which is used for the software project effort estimation. The formula used for the effort estimation using COCOMO model is given below:-

1), Effort $E = a * (KDSI)^{b} * EAF$ Where KDSI is number of thousands of delivered source instructions a and b are constants, may vary depending on size of the project

2), Schedule $S= c * (E)^{d}$ where E is the Effort and c, d are constants.

3),EAF is called Effort Adjustment Factor which is 1 for basic cocomo , this value may vary from 1 to 15.

Table 1: Contact Values

	Constants			
Project modes	a	b	**c**	d
Organic project mode	2.4	1.05	2.5	0.38
Semi-detached project mode	3.0	1.12	2.5	0.35
Embedded project mode	3.6	1.20	2.5	0.32

ISSN: 2414-1895 DOI: 10.6919/ICJE.202006_6(6).0028

It is analyzed that the effort estimation is directly proportional to KLOC values. The KLOC value defines the number of lines in the project. To accurately estimate the KLOC value the machine learning algorithm will be applied in this research work. The machine learning algorithms are broadly classified into supervised and unsupervised learning algorithms. In this research work, the Support vector machine learning which supervised learning algorithm will be used for the KLOC value estimation. The SVM classifier will derive the training model and also use the test the model for the KLOC value estimation. The output of the test set will be given as input as input to COCOMO model for the effort estimation.

References

[1] Nancy Merlo Schett, "Seminar on software cost estimation", University of Zurich, Switzerland, 2003.

[2] Chetan Nagar, "Software efforts estimation using Use Case Point approach by increasing technical complexity and experience factors", IJCSE, ISSN:0975-3397, Vol.3 No.10 ,Pg No 3337- 3345,October 2011.

[3] YunsikAhn, Jungseok Suh, Seungryeol Kim, Hyunsoo Kim, "The software maintenance project effort estimation model based on function points", Journal of software maintenance and evolution, 2003.

[4] Ashish Sharma, Dharmender Singh Kushwaha, "A Metric Suite for Early Estimation of Software Testing Effort using Requirement Engineering Document and its validation", International Conference on Computer & Communication Technology (ICCCT), 2011

[5] Ricardo Britto, Emilia Mendes, Claes Wohlin, "A Specialized Global Software Engineering Taxonomy for Effort Estimation", 2016 IEEE 11th International Conference on Global Software Engineering, IEEE, 2016

[6] Mustafa Hammad, Abdulla Alqaddoumi, "Features-Level Software Effort Estimation Using Machine Learning Algorithms", 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)

[7] Suyash Shukla, Sandeep Kumar, Pravas Ranjan Bal, "Analyzing Effect of Ensemble Models on Multi-Layer Perceptron Network for Software Effort Estimation", 2019 IEEE World Congress on Services (SERVICES)

[8] Suyash Shukla, Sandeep Kumar, "Applicability of Neural Network Based Models for Software Effort Estimation", 2019 IEEE World Congress on Services (SERVICES)

[9] Seiji Fukui, Akito Monden, Zeynep Yücel, "Kurtosis and Skewness Adjustment for Software Effort Estimation", 2018 25th Asia-Pacific Software Engineering Conference (APSEC)

[10] Ahmed BaniMustafa, "Predicting Software Effort Estimation Using Machine Learning Techniques", 2018 8th International Conference on Computer Science and Information Technology (CSIT)