EFFORT ESTIMATION METHODS IN SOFTWARE DEVELOPMENT USING MACHINE LEARNING ALGORITHMS

Bhaskar Marapelli  
Research Scholar  
Shri JJT University  
Rajasthan  
bhaskarmarapelli@gmail.com

Dr. Prasadu Peddi  
Professor  
Shri JJT University  
Rajasthan

ABSTRACT

In software engineering, estimation plays a vital role in software development. Thus, affecting its cost and required effort and consequently influencing the overall success of software development. The error margin in Expert-Based, Analogy-Based and algorithmic based methods including: COCOMO, Function Point Analysis and Use-Case-Points is quite significant, which exposes software projects to the danger of delays and running over-budget. To obtain better estimation, we propose an alternative method through performing data mining on historical data. This paper suggests performing this prediction using three machine learning techniques that were applied to a preprocessed COCOMO NASA benchmark data which covered 93 projects: Naïve Bayes, Logistic Regression and Random Forests. The generated models were tested using five folds cross-validation and were evaluated using Classification Accuracy, Precision, Recall, and AUC. The estimation results were then compared to COCOMO estimation.

Keywords: Software Effort Estimation; COCOMO Data Mining; Machine Learning; Naïve Bayes; Logistic Regression; Random Forests.

INTRODUCTION

The use of software grows continuously in the society. Software factories need to produce software of high quality and in time to assure competitiveness. The high competitiveness forces the software industry to conclude software projects as planned, that is, in time and within budget. Therefore, the need for planning and project management has increasingly demanded more attention and control from software project managers. Estimation of the effort is one of the most important tasks in software planning and management. In a report from the Standish Group’s Chaos, 66% of the software projects analyzed were delivered with delay or above the foreseen budget, or worse, they were not finished. One of the major causes of such failures is inaccurate estimates of the effort of the projects.

Hence, it is very important to investigate novel methods for improving the accuracy of such estimates. The precision of the effort estimate is very important for software factories because both overestimates and underestimates of the software effort are harmful to software companies. Several methods have been investigated for software effort estimation, including traditional methods such as the COCOMO, and, more recently, machine learning techniques such as radial basis function (RBF) neural networks, bagging predictors and support vector regression (SVR). Machine learning techniques use data from past projects to build a regression model that is subsequently employed to predict the effort of novel projects. In most methods for software effort estimation, only the estimations of the efforts are given.
Yet, it would be very important to provide a confidence interval for the estimation along with the estimation. This would enable an estimation method to give an interval where the effort would fall. In many applications, the confidence intervals for predictions are computed by assuming that the errors follow some probability distribution. Often, it assumed that the errors follow a Gaussian distribution. The errors are then used to estimate the parameters and compute the confidence interval. Unfortunately, in many cases the distribution assumed for the errors may not be correct. Robust confidence intervals are more general because they do not make any assumptions about the errors. They were applied recently by Oliveira and Meira to detect novelty in time series.

**LITERATURE REVIEW**

**M. Shin, A.L. Goel (2000)** Estimation of effort for the proposed software is a standout amongst the most essential activities in project management. Proper estimation of effort is often desirable in order to avoid any sort of failures in a project and is the practice to adopted by developers at the very beginning stage of the software development life cycle. Estimating the effort and schedule with a higher accuracy is a challenge that attracts attention of researchers as well as practitioners. Predicting the effort required to develop a software to a certain level of accuracy is definitely a difficult assignment for a manager or system analyst, when the requirements are not very clearly identified. Effort estimation helps project managers to determine time and effort required for the successful completion of the project. In order to help the organization in developing qualitative products within a planned time frame, the job of appropriate software effort estimation is of primary requirement. For measuring the cost and effort of software development, traditional software estimation techniques like Constructive Cost Estimation (COCOMO) model and Function Point Analysis (FPA) have not been proved very much satisfactory, because of uncertainties associated with parameters such as Line Of Code (LOC) and Function Point (FP) respectively, used for procedural programming concept. The procedural oriented design splits the data and procedure, whereas accepted practice of present day i.e., the object-oriented design combines both of them since class and use case are the basic logical units of an object-oriented system, the use of Class Point (CP) and Use Case Point (UCP) approach to estimate the project effort helps to get more accurate result.

**A. J. Smola, B. Scholkopf (2004)** Similarly, in case of agile projects, Story Point Approach (SPA) is used to measure the effort required to implement a user story. By adding up the estimates of user stories which were nished during an iteration (story point iteration), the project velocity is obtained. The dataset related to CP, UCP and SPA are collected from previous projects mentioned in few research articles or from industries in order to assess the results. In order to create results of estimation with more accuracy, when managing issues of complex connections in the middle of inputs as well as yields, and where, there is a distortion in the inputs by high noise levels, the application of machine learning (ML) techniques helps to bring out results with more accuracy. A number of past research studies indicate that no single technique turns out to be the best for all cases. This is because of the dependency of system's execution altogether on the predicted function types, variations in properties of collected data, number of tests, noise ratio and so on. Hence the use of ML techniques in
order to cope with issues arises in real-life situation is considered to be worthwhile. The research work carried out here presents the use of various ML techniques for software effort estimation using CP, UCP, Web-based and SPA approaches. The ML techniques are implemented taking into consideration of related dataset to predict the required effort.

I. H. Witten, E. Frank (2005) In software engineering, estimation plays a vital role in software development. Thus, affecting its cost and required effort and consequently influencing the overall success of software development. The error margin in Expert-Based, Analogy-Based and algorithmic based methods including: COCOMO, Function Point Analysis and Use-Case-Points is quite significant, which exposes software projects to the danger of delays and running over-budget. To obtain better estimation, we propose an alternative method through performing data mining on historical data. This paper suggests performing this prediction using three machine learning techniques that were applied to a preprocessed COCOMO NASA benchmark data which covered 93 projects: Naïve Bayes, Logistic Regression and Random Forests. The generated models were tested using five folds cross-validation and were evaluated using Classification Accuracy, Precision, Recall, and AUC. The estimation results were then compared to COCOMO estimation.

METHODOLOGY

Dataset Used

The benchmark dataset in this research was collected by Jairus Highn. It covers information regarding 93 software development projects that was developed by NASA for the period between 1971 and 1987. The dataset include 24 attributes that cover projects information including project number, name, and category; development information including: development center and actual effort; in addition to a number of nominal values that describe COCOMO parameters regarding development mode and drivers. The target of the estimation was set to the actual effort attribute.

Pre-Processing Procedures

The values of driver multipliers in the dataset were converted from nominal into its corresponding numerical weight. The mode constants for project were also assigned according to COCOMO predefined values. The 15 multiplier values and the constants obtained were then used in addition to the given software size in order to calculate the estimated effort for each projects in man month. A new feature was constructed in the dataset in order to express the project parameters in the data mining prediction and classification. The value of this attribute was calculated by multiplying the numerical value of each project driver and then transformed it by applying the log function. The attributes that represent project multipliers were then deleted. The identifiers of projects development centers were converted from numerical into nominal values.

This was performed in order to avoid any potential bias when building the models which might be caused by giving numerical weights to these identifiers. The actual effort in each project was categorized into three categories. The ranges of the values of each class in these
categories were selected to provide natural splits for these values were possible and also to ensure a balanced distribution for the samples in each class. Three extreme effort projects were excluded from the dataset as the value of the standard deviation for their effort values was very high compared to the rest of the projects. This step was indeed necessary in order to avoid bias and to provide more realistic prediction, so the number of samples in the dataset was reduced to 90 instead 93.

RESULTS

The actual effort for each project was recorded as a continuous numerical value, which was then converted into categorical values. Three categories were defined. Each corresponds to the level of the actual effort involved in each projects. The data was then explored and found in need for even more preprocessing. The multipliers were converted into their relevant numerical values based on the mode of the projects and the values given by COCOMO Intermediate model. The values of the multipliers were found not correlated with the prediction of the actual effort and therefore, were multiplied to construct a new attribute that was called project multipliers. The original individual multipliers were then deleted.

![Graph Density Plot](image)

Since the projects parameters in reality affect code, the newly constructed multiplier attribute was multiplied by the equivalent physical code attribute and the result was transformed using the log function. Fig. shows the density plot for the three effort classes. The data exploration also involved projecting effort over years, projects and the size of code and its LN transformation multiplied by the COCOMOM parameters. The result of this projection shows some influence of both size and years on effort increase while showing little but un unexpected negative correlation between size and effort. Projection plot in Fig. shows the results of effort projection. The Euclidian distance of each sample from the center of each group was also calculated in order to find the consistency of samples in each effort category.
Graph Projection Plot: showing the projection of effort classes over projects, years, code size and the transformed code Size multiplied by OCOMO parameters

Three machine learning techniques were selected to act on the processed data: Naïve Bayes, Logistic Regression Forests. The selection was based on the define nature of the explored data which were matched to the machine learning techniques that may fulfill these objectives and suite the nature of data. The data was then formatted and acclimatized to suite the requirements of apply techniques. The selected techniques were then applied to the acclimatized data and evaluated iteratively validation. One technique was applied model in each iteration loop.

CONCLUSION

We have proposed the use of machine learning techniques combined with robust confidence intervals to improve the precision of estimates of software project effort. In our simulations, we used two datasets of software projects: Desharnais and NASA. The simulation results showed that bagging was able to improve performance of the following regression methods in the effort estimation task: (1) SVR, (2) MLP and (3) model trees (M5P). We compared our results to previous ones published in the literature. The comparisons were made considering results reported. The results showed that bagging with M5P/model trees achieved the best performance in terms of MMRE. Bagging with MLP outperformed previous methods reported in the literature in terms of PRED.

REFERENCES


