Software Productivity Estimation by Regression and Naïve-Bayes Classifier-An Empirical Research

Jun Wu [*] and *Sisi Gao*

School of Economics and Management, Beijing University of Posts and Telecommunications, Beijing, China

[*]Corresponding author: Jun Wu, wujun1127@139.com

Abstract

Software cost estimation is now a big concern in software engineering. Although many measurement-based analytical approaches have been proposed, some are focused on producing point estimates rather than interval predictions. The objective of this paper is to investigate the software productivity using linear regression and Naive–Bayes classifier methods. We conduct empirical experiments with 66 historical project data sets from China telecommunication operator and compare the fitting and predictive results of project delivery rate using two approaches respectively. The paper demonstrates that Naive–Bayes classifier is robust enough when predicting the software productivity during the early stage of development.

Keywords: *software productivity estimation, regression analysis, naive–bayes classifier*

1 Introduction

With the ever growing IT software expenses in large companies, accurate software cost estimation is crucial to the success of software development projects. Previous scholars have proposed numerous estimation models, based on expert judgment, estimation by analogy or algorithmic approaches [1]. In most cases, effort and duration estimation normally leads to the estimation of cost [2]. When predicting the software effort, it is important to calculate the project development productivity. Productivity is conceptualized as output produced per unit of input, and normally defined as the ratio between the size S and the effort E of a project. As the functional size increases, so does associated effort. According to the International Software Benchmarking Standards Group (ISBSG), Project Delivery Rate (PDR) can be a good indicator to measure the software productivity [1]. PDR here is defined as the ratio of size divided by effort and is expressed as effort hours per functional point. Clearly PDR is an inverse measure of productivity in that the larger PDR, the smaller is the productivity.

Previous researchers have performed quantified analysis of the project data repositories from ISBSG to identify the project attributes that influence PDR [1-5]. They found that PDR differed much between different industries or business sectors. However, since their works mainly aimed to identify the key factors that affect PDR, their conclusions are some kind of description of the average level of productivity, which is not good enough for companies to conduct the business specific estimation with an accurate PDR value or range. In addition, to model the relationship between productivity and diverse factors, different measurement-based analytical approaches have been proposed but some are not always successful, due to the high degree of uncertainty and the lack of information in the early stages of software development [6]. To overcome these difficulties, expert judgment should be used in parallel with other estimation models for better results.

In this paper, we try to make two main contributions: (1) using regression analysis and Naive–Bayes classifier to estimate the PDR based on project historical data from China telecommunication operator; (2) demonstrating that the Naive–Bayes classifier has the potential to be used as an additional technique for the prediction of new software development productivity. The approach described in the paper is general and could be used to estimate values of effort and any other variables of interest provided that sufficient historical data are available.

2 Estimation by Regression and Naive–bayes Classifier

2.1 Regression analysis approach

Previous studies have assumed an approximately linear relation between software size (measured by function point) and work effort (measured by man hours) [2]. On the other hand, J. E. Matson, etc. (1994) demonstrated that the log-linear relationship assumption between effort and function points was superior to the linear one based on the 104 project data sets analysis [8]. Studies of the projects in the ISBSG Repository have shown that Size and Maximum Team Size are the most important drivers of effort [1]. Since PDR=Effort/FP, similar relationships exist between software function point and PDR, see Equations (1) and (2) as the general forms.

$$\mathrm{PDR} = \mathrm{a} \times FP^{\mathrm{b}} \times Team^{\mathrm{c}} - - - (1)$$
$$\mathrm{PDR} = \mathrm{a} \times FP^{\mathrm{b}} - - - -(2)$$

2.2 Bayesian networks and Naive–Bayes classifier

A Bayesian network is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). Nodes show variables and the probability of the variables to take a specific value. Between the variables a cause-effect relationship exists. On the top a node for productivity is shown. The productivity can be defined in intervals of productivity values. Based on the probabilities assigned to the variables and the knowledge of cause-effect relationship the probability of achieving a specific productivity can be calculated. The theoretical foundation of Bayesian network is Bayesian analysis and inference which excels in dealing with imprecise data and expert judgment.

The Naive–Bayes classifier is a special case of Bayesian network classifier, derived by assuming that the variables are conditionally independent given the class variable (see Fig. 1). No other connections are allowed in a Naive-Bayes structure.

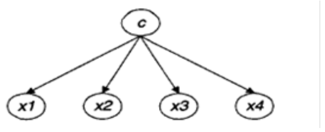


Fig. 1 –A simple Naïve-Bayes structure

.

The graph in Fig. 1 shows the structure of the Naive–Bayes classifier in which the variable C is the class variable. The posterior probability distribution P (c|x1, x2, x3, x4) can be computed efficiently from Bayes theorem:

$$P(c|x_1,x_2,x_3,x_4) = \frac{P(c,x_1,x_2,x_3,x_4)}{P(x_1,x_2,x_3,x_4)} - - - (7)$$

Where

$$P(c,x_1,x_2,x_3,x_4) = P(C)P(x_1|c)P(x_2|c)P(x_3|c)P(x_4|c)$$

And

$$P(x_1,x_2,x_3,x_4) = \sum_{C} P(c,x_1,x_2,x_3,x_4)$$

Naive-Bayes has been used as an effective classifier for many years due to its advantages over other classifiers. First, it is relatively simple to implement, efficient, robust with respect to noisy or missing data, and performs very well in many domains. Second, it frequently outperforms more sophisticated classifiers even for the small data sets. Previous study has demonstrated its validity in predicting the PDR [16].

In our case, the estimation of PDR value of a new project involves a prediction of the PDR on the basis of the observed values of the historical project characteristics (variables). We assume that PDR (the class variable) has discrete values, referred to as categories or classes. Then, Naive–Bayes can compute the probability that a given project's PDR falls within each of these intervals. Although it cannot predict the actual value of PDR directly, we can approximate the value of the class variable (PDR) by its expected value using the following formula:

$$E(PDR|observation) = \sum_{i=1}^{n} mid_i * P(PDR_i|observation)\ (8)$$

Where

$$mid_i = \frac{(lower\ interval_i + uper\ interval_i)}{2}$$

$P(PDR_i|observation)$ denotes the conditional probability of the i th interval given the observation and can be computed by Naive–Bayes algorithm.

3 Analysis and Findings

3.1 Regression analysis and Naive–Bayes classifier results

We collected data sets from 66 projects obtained from a large telecommunication operator in China. he set of project data represents a wide range of software applications, ranging in size from 76 function points to 2595 function points. Project data included the following information: software size in function points, development effort in man hours, system domain, team size, and primary development language.

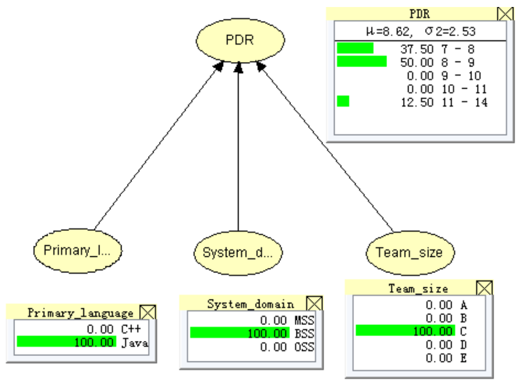
Based on equation (1) and (2), we applied linear and power regression to fit the model shown in Table I. It seems they both fits well, but the power regression model is superior to some extent, for the power estimation model has a much smaller Std. Error of the Estimate.

Table 1 –Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
Linear	0.963	0.927	0.925	113.278
Power	0.968	0.938	0.937	0.140

After having constructed the Naive–Bayes classifier, we first split the original data set into predefined training set and test set. Then the Naive–Bayes model is trained using a training data set. Finally, we use the model to predict PDR for the projects in the associated test set.

Calculated by Hugin 7.7, the output probability distribution of PDR for projects with system domain= BSS, primary development language= Java, and team size categorize to interval number 5 are shown in Fig. 2.



```
PDR
μ=8.62,  σ2=2.53
37.50 7 - 8
50.00 8 - 9
 0.00 9 - 10
 0.00 10 - 11
12.50 11 - 14

Primary_language
  0.00 C++
100.00 Java

System_domain
  0.00 MSS
100.00 BSS
  0.00 OSS

Team_size
  0.00 A
  0.00 B
100.00 C
  0.00 D
  0.00 E
```

Fig. 1 –The output probability distribution of PDR.

The fitting accuracy of the model was evaluated by the usual mean magnitude of relative error (MMRE) and PRED (25%) measures. Briefly, The MMRE measure is defined as the average of the magnitudes of relative errors (MRE) of the n samples selected from the data set. Based on the prediction models, the MMRE and PRED (25%) results of Naive–Bayes classifier method and regression method are presented in Table II. We can see that although the Naive–Bayes classifier model is used for categorization its predictive accuracy when used for point estimates is quite satisfactory.

Table 1 –Model Comparison Summary

			MMRE	PRED (25%)
Naive–Bayes classifier	Using the mean point as point estimate	Fitting accuracy of the model for all 66 projects	0.29	0.65
		Fitting accuracy for the training dataset	0.32	0.67
		Predictive accuracy for the test dataset	0.21	0.78
Regression approach		Fitting accuracy of the model for all 66 projects	0.41	0.45
		Fitting accuracy for the training dataset	0.42	0.43
		Predictive accuracy for the test dataset	0.39	0.42

4 Conclusions

In this paper we investigated the regression analysis and Naive–Bayes classifier for estimating project delivery rates. The results show that the fitting and predictive results using Naive–Bayes classifier are quite encouraging comparable to those of the previous works in this field [1, 2, 3, 8, and 12]. We believe that Naive–Bayes classifier will be a valuable tool for analysis of software productivity data and will play an important part in software cost estimation.

Future work will focus on the application of Naive–Bayes classifier to other sources of productivity data sets and the comparison of the ability of various alternative methods to produce categories estimates.

References

1. *Stamelos, L. Angelis, P. Dimou, and E. Sakellaris*, On the use of bayesian belief networks for the prediction of software productivity, Information and Software Technology, vol. 45 (1) , 2003, pp. 51–60.
2. *P. R. Hill, Ed.*, Practical Software Project Estimation: A Toolkit for Estimating Software Development Effort & Duration, International Software Benchmarking Standards Group, 2011
3. *M. Sadiq, F. Mariyam, A. Ali, S. Khan, and P. Tripathi*, Prediction of Software Project Effort Using Fuzzy Logic, Engineering and Technology, pp. 353-358, 2011.
4. *M. Ross*, Parametric Project Monitoring and Control, in Proc. Joint ISPA/SCEA 2005 Conference, 2005.
5. *M. Ross*, Managing Software Size, in ISPA/SCEA Conference, 2003.
6. *Z. Jiang and C. Comstock*, The Factors Significant to Software Development Productivity, Engineering and Technology, pp. 160-164, 2007.
7. *I. Stamelos, L. Angelis, P. Dimou, and E. Sakellaris*, On the use of bayesian belief networks for the prediction of software productivity, Information and Software Technology, vol. 45 (1), 2003, pp. 51–60.
8. *J.E. Matson, B. E. Barrett, and J. M. Mellichamp*, Software Development Cost Estimation Using Function Points, in IEEE Transactions on Software Engineering, 1994, vol. 20, no. 4, pp. 275--287.
9. *Y. Zheng and B. Wang*, Estimation of software projects effort based on function point, Journal of Management, pp. 3519-3521, 2009.
10. *K.D. Maxwell*, Benchmarking Software Development Productivity, Ieee Software, no. February, pp. 80-88, 2000.
11. *M. He, Y. Yang, Q. Wang, and M. Li*, An Investigation on Performance of Software Enhancement Projects in China, 2008 15th Asia-Pacific Software Engineering Conference, pp. 67-74, 2008.
12. *H. Wang, H. Wang, and H. Zhang*, Software Productivity Analysis with CSBSG Data Set, 2008 International Conference on Computer Science and Software Engineering, pp. 587-593, 2008.
13. *IFPUG*, Function Point Counting Practices Manual, 2010.
14. *A. J. Albrecht and J. E. Gaffney*, Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation, IEEE Transactions on Software Engineering, vol. 9, no. 6, pp. 639-648, Nov. 1983.
15. *Friedman N, Geiger D, Goldszmidt M*. Bayesian network classifiers. Machine Learning 1997, vol. 29(2–3), pp. 131–163.
16. *B. Stewart*, Predicting project delivery rates using the Naive–Bayes classifier, Journal of Software Maintenance and Evolution: Research and Practice. 2002, vol. 14, pp. 161–179
17. *S.L. Pfleeger*, Software Engineering Theory and Practice. Prentice-Hall: Upper Saddle River NJ,