Factor Analysis and Random Forest Based Model of Software Cost Estimation

Wei Zhang[1], Haixin Cheng[2*], Siyu Zhan[2], Ming Luo[1], Feng Wang[1], Zhan Huang[1]

[1]PetroChina Southwest Oil and Gasfield Company, Chengdu, 610041, China
[2]University of Electronic Science and Technology of China, Chengdu, 611731, China

*Corresponding author's e-mail:xhcmail@std.uestc.edu.cn

Abstract. Software Cost Estimation is one of the challenges in software engineering. Accurate estimates can increase the speed of the effort for developing software projects, and prevent probabilistic failure consequently. Based on factor analysis and random forest, this article proposed a new SCE model. The model recombines factors that affect software workload into six factors, measuring the size of workload from aspects such as software performance requirements, developer capabilities, and data size. The random forest model using the XGBoost framework is built to complete the software workload prediction task. Then, we evaluated the performance of the model on three datasets, including COCOMO81, and the results showed that the model has high prediction accuracy and strong robustness, and can achieve high precision with fewer data samples.

Keywords: Software Cost Estimation; Factor Analysis; Random Forest

1 Introduction

Accurate estimation of cost and time is crucial for the success of software development projects. Therefore, it is a potential factor in estimating the cost, time, and effort required for software development projects.

In order to estimate the cost and effort required for software development projects, different algorithmic software models, such as COCOMO I[1], COCOMO II[2], SLIM[3,4], and FP[5] have been used. Most of the coefficients in these methods are based on empirical values, and many weight factors have significant errors. In practical work, the evaluation results are often inaccurate.

Using machine learning methods to establish evaluation models can greatly compensate for the deficiencies of traditional models that rely solely on empirical judgments. However, there is a lack of research on using machine learning techniques to solve workload evaluation problems, and the research that has been conducted is limited to comparing several machine learning (ML) models. The ML methods that have been studied include artificial neural networks (ANN), decision trees (DT), regression trees (RT), Bayesian networks (BN), and support vector regression (SVR). Among these, only the models based on regression trees have received attention from researchers due to their strong generalization ability and interpretability.

Selby and Porter[6] used the ID3 algorithm to generate a large number of decision trees to classify software modules with high development intensity. In order to deal with the uncertainty and inaccuracy of data in software development projects, the authors in [7] studied the application of fuzzy ID3 decision trees. This method was designed by combining the concepts of the ID3 algorithm and fuzzy set theory, and used MMRE and Pred as standards to measure prediction accuracy. After comparing the results with other decision tree-based models such as ID3, CART, and C4.5, it was found that the fuzzy ID3 method was more accurate than the other methods mentioned. In paper [8], decision tree forests (DTF) were further used to create predictive models for workload estimation. Nassif compared three methods: multiple linear regression (MLR), DT, and DTF. They used the ISBSG R10 and Desharnais datasets and implemented DTF using ten-fold cross-validation, with MMRE, MdMRE, and Pred(.25) as the metrics for accuracy measurement. The results showed that the performance of DTF was better than that of DT and MLR.

It is not difficult to find from existing research that using decision tree-based models to improve the accuracy of project management workload estimation is quite common in these few studies. However, decision trees, especially regression trees, as a relatively simple machine learning algorithm, have inherent flaws such as easy overfitting, sensitivity to noise, and unstable prediction results. Given the various shortcomings of regression tree models, we should expand our thinking to use other methods to optimize decision-trees-based models.

Factor analysis and random forest are two effective methods for improving the robustness and accuracy of models. By recombining weighted subsets of attributes, various factors that affect software workload can be comprehensively reflected from different perspectives. This makes the new attributes more robust with strong interpretability, reducing the possibility of large prediction errors caused by biased attribute values. At the same time, using a random forest model for workload prediction avoids the problem of low model accuracy due to insufficient data sets. The resulting model has strong generalization and robustness. Innovatively introducing these two methods into software workload assessment will further enhance the performance of the prediction model.

The structure of the paper is as follows: In Section 2, we'll deal with factors extraction problem in Software Cost Estimation (SCE); in Section 3, we will review XGBoost model; in Section 4, we will discuss the building process and the performance of our model for the SCE, and finally in Section 5, we'll deal with the conclusions and future works.

2 Factors extraction

In software workload evaluation, the same task is often characterized by many variables to depict the actual workload size. Multiple variable samples can provide a lot of information for predicting results, but to some extent, it increases the workload of data collection. More importantly, in most cases, there may be correlations between many variables, which means that variables that seem different on the surface may not reflect different attributes of things from various perspectives, but rather different expressions of the same attribute. To enhance the robustness and interpretability of the model, we used factor analysis to combine closely related variables, so that the recombined factors reflect the size of the software workload from different dimensions, and avoid the problem of inaccurate model caused by large estimation deviation of individual attribute.

To establish a software workload assessment model, we used the COCOMO81 dataset as the data for training and testing the model. It includes 15 independent variables (such as analyst ability, programmer ability, application experience, etc.) and two dependent variables (actual workload and source code lines), each of which is a continuous variable. To perform factor analysis on this dataset, the correlation between variables should be determined first, and the covariance matrix is shown in the Fig. 1. The KMO value obtained through the KMO test is 0.70, indicating a high correlation between the attributes, which allows for factor analysis to be conducted.

Next, we need to determine the number of factors. Having too many factors can make the model more susceptible to noise and reduce its robustness, while having too few factors can result in a low expression rate for the factors, which cannot effectively extract the underlying information in the original data. Therefore, we combine the practical significance of scree plot and software workload evaluation to comprehensively determine the optimal number of factors. When the number of factors is 6, the eigenvalue of the matrix reaches the inflection point, and the expression rate of these factors reaches 81%. Therefore, the number of factors after dimensionality reduction is determined to be 6.

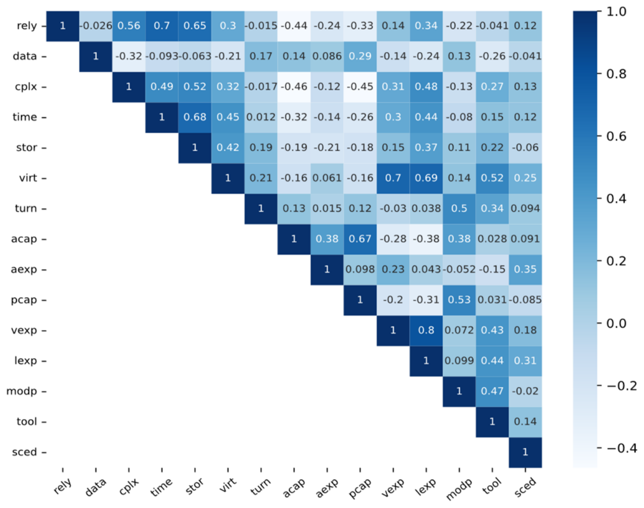

Fig. 1. Heatmap of the covariance matrix between attributes

After determining the number of factors, it is necessary to name the obtained factors based on the actual background of software workload evaluation and the actual significance of the original attributes. The contribution rates of the original attributes to each factor are shown in the Fig. 2. Based on this, we named the six factors as shown in the Table 1. These six factors evaluate the workload of software development from aspects such as software complexity, development environment, and personnel technical level, which have strong practical guidance significance and make the model highly interpretable. Subsequently, all data will undergo factor rotation to obtain the dimensionality reduced data.

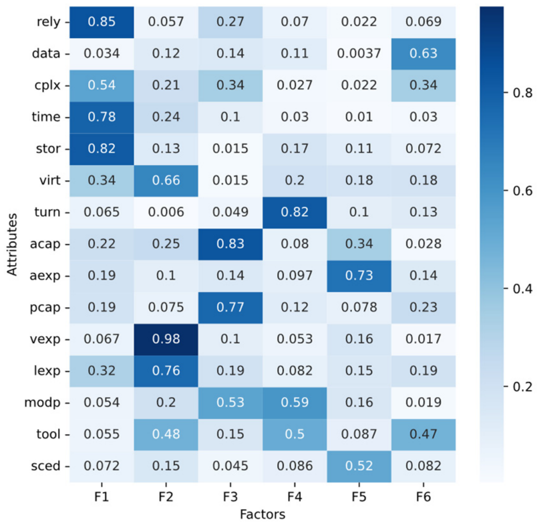

Fig. 2. Factor loading coefficient heatmap

Table 1. Factor Naming Result Table

Factor index	Factor name
F1	Software performance requirements
F2	Difficulty in platform development
F3	Personnel development capability
F4	Teamwork ability
F5	Difficulty in software operation and maintenance
F6	Software data scale

3 Model building strategy based on XGBoost

In order to overcome the shortcomings of low prediction accuracy and poor model interpretability in the software workload evaluation process of existing technologies, we propose a prediction model that combines factor analysis and random forest. The model mainly includes four steps: data cleaning, feature extraction, model training, and result prediction. The model building process is shown in the figure.

In practical applications, there may be some attributes in the sample data used for model training and prediction whose values cannot be estimated or whose data errors are large due to the strong subjectivity during estimation, leading to inaccurate prediction results. To address this issue, we use the box plot method to remove outlier data from the data to avoid the influence of extreme values on prediction results. For the missing values that arise from the removal of outliers, we fill them using a linear regression tree model. Specifically, we select a factor with missing values as the dependent variable and other factors as independent variables to construct a regression tree model. We use the constructed regression tree model to predict the numerical value of the missing value, and repeat this step until all missing values are filled.

After completing the data preprocessing, the factor analysis step will be performed, and its specific method has been detailed in section three. The reduced data will be shuffled and divided into a training set and a test set. To ensure that the model has sufficient training data, the training set accounts for 90% of the total data, with a total of 53 data samples, and the test set accounts for 10%, with a total of 7 data samples. Next, for the training set data, the XGBoost[9] distributed gradient boosting library is used to train the model. The prediction of random forests depends on various factors, such as the number of decision trees used, the number of features used for splitting at each node, the method used to calculate information gain (such as Gini coefficient, entropy), the number of samples, and the maximum depth of the tree. In order to determine the parameter values that can generate the best random forest, this experiment adopts the method of adjusting hyperparameters instead of theoretical analysis. During the hyperparameter tuning process, a random forest model is established by using different combinations of parameter values. Then, the parameter values that generate the best predictive model are considered to be the most suitable values for that model. The final model contains 10 regression trees, with a maximum depth of no more than 8 layers, and a training rate set at 0.1.

So far, our software workload estimation model has been fully constructed. We input the data from the testing set into the model to obtain the model's predicted results. In order to facilitate horizontal comparison between models, we will use some common performance evaluation standards to evaluate the predicted results of the testing set. There are many traditional methods and ML models available for workload estimation. To determine the better model or method, it is necessary to calculate the accuracy of the model. Several criteria can be used to evaluate the model. This experiment considers the three most common evaluation criteria shown in the Table 2, all based on the calculation of the Magnitude of Relative Error [10] (MRE):

$$MRE = \frac{|E_{actual} - E_{estimated}|}{E_{actual}}$$

E_{actual} represents the actual workload value, while $E_{estimated}$ represents the predicted value of the model.

Table 2. Some common evaluation criteria used in the experience

Evaluation Criteria	**Description**
$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i$	Mean MRE (MMRE). MMRE is a common method used for evaluation prediction models.
$MdMRE = Median(MRE)$	Median MRE (MdMRE). MdMRE is criterion for mean MRE error. MdMRE has been used as another criterion because it is less sensitive to outliers.
$PRED(x) = \frac{1}{n} \times \sum_{i=1}^{n} \begin{cases} 1, & if\ MRE_i \leq x \\ 0, & otherwise \end{cases}$	Where n denotes the total number of projects and k denotes the number of projects whose MRE is less than or equal to x. normally, x is set to be 0.25.

In order to evaluate the effectiveness of the model on different datasets, we built the model using the above method on three datasets: COCOMO 81, Albrecht, and Desharnais. The evaluation results of the model are shown in the Table 3.

From the results, it can be seen that the difference between MMRE and MdMRE indicators of different datasets are small, indicating that the predicted results are relatively stable and there are no individual predicted results that differ significantly from the true values. Although the Albrecht training set contains only 21 training samples, the obtained model still has a high prediction accuracy, which confirms the efficient convergence of the random forest model. Our models possess the ability to discern the connections between input and output within system data. The training process modulates the activity levels of these connections, enabling the model to uncover the associated rules between input and output, even in cases where these rules are non-linear and intricate. Essentially, the models are capable of learning and possess the knowledge to address issues through learning methods. The learning capacity of the model is achieved by adjusting the parameters. The objective is to ensure that the model remains efficient in adapting to new conditions with minimal training, even when slight changes transpire in the model's environmental circumstances.

Table 3. Evaluation of Criteria MMRE, MdMRE and PRED

Dataset	**MMRE**	**MdMRE**	**PRED(25)**
COCOMO81	0.44	0.50	0.29
Albrecht	0.17	0.38	0.33
Desharnais	0.56	0.64	0.22

4 Conclusions and Future Works

Estimating the cost of software development is a particularly difficult task for managers since a multitude of cost factors are subject to change and can be difficult to accurately predict, especially in the initial stages of development. After conducting investigations and carefully considering the findings presented in this paper, it can be concluded that both factor analysis and random forest are valuable tools for Software Cost Estimation. These methods can be applied to analyze and estimate software projects with large and small dataset. The model built through these methods have the advantages of fast training speed, good convergence effect, and strong robustness to noise. It has great practical value in software development applications. We hope that in future software workload assessment tasks, combining factor analysis and random forest to build models will lead to faster prediction speeds and more accurate forecasting results.

Acknowledgments

This study was supported by Sichuan Science and Technology Program (No. 2022YFG0176).

References

1. Boehm, B. W. (1984). Software engineering economics. IEEE transactions on Software Engineering, (1), 4-21. DOI: 10.1109/TSE.1984.5010193
2. Musílek, P., Pedrycz, W., Sun, N., & Succi, G. (2002, June). On the sensitivity of COCOMO II software cost estimation model. In Proceedings Eighth IEEE Symposium on Software Metrics (pp. 13-20). IEEE. DOI: 10.1109/METRIC.2002.1011321
3. Côté, V., Bourque, P., Oligny, S., & Rivard, N. (1988). Software metrics: An overview of recent results. Journal of Systems and Software, 8(2), 121-131. https://doi.org/10.1016/0164-1212(88)90005-2
4. Conte, S. D., Dunsmore, H. E., & Shen, Y. E. (1986). Software engineering metrics and models. Benjamin-Cummings Publishing Co., Inc.. https://dl.acm.org/doi/abs/10.5555/42168
5. Albrecht, A. J., & Gaffney, J. E. (1983). Software function, source lines of code, and development effort prediction: a software science validation. IEEE transactions on software engineering, (6), 639-648. DOI: 10.1109/TSE.1983.235271
6. Porter, A. A., & Selby, R. W. (1990). Evaluating techniques for generating metric-based classification trees. Journal of Systems and Software, 12(3), 209-218. https://doi.org/10.1016/0164-1212(90)90041-J
7. Idri, A., & Elyassami, S. (2011). A fuzzy decision tree to estimate development effort for web applications. International Journal of Advanced Computer Science and Applications, 1(3). DOI : 10.14569/SpecialIssue.2011.010314
8. Nassif, A. B., Azzeh, M., Capretz, L. F., & Ho, D. (2013, June). A comparison between decision trees and decision tree forest models for software development effort estimation. In 2013 Third International Conference on Communications and Information Technology (ICCIT) (pp. 220-224). IEEE. DOI: 10.1109/ICCITechnology.2013.6579553
9. Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794). https://doi.org/10.1145/2939672.2939785
10. Capretz L F, Marza V. Improving effort estimation by voting software estimation models[J]. Advances in Software Engineering, 2009, 2009. https://doi.org/10.1155/2009/829725