AI-Based Models for Software Effort Estimation

Ekrem Kocaguneli, Ayse Tosun, Ayse Bener
Department of Computer Engineering
Bogazici University
Istanbul, Turkey
ekrem.kocaguneli@boun.edu.tr, ayse.tosun@boun.edu.tr, bener@boun.edu.tr

Abstract

Decision making under uncertainty is a critical problem in the field of software engineering. Predicting the software quality or the cost/ effort requires high level expertise. AI based predictor models, on the other hand, are useful decision making tools that learn from past projects' data. In this study, we have built an effort estimation model for a multinational bank to predict the effort prior to projects' development lifecycle. We have collected process, product and resource metrics from past projects together with the effort values distributed among software life cycle phases, i.e. analysis & test, design & development. We have used Clustering approach to form consistent project groups and Support Vector Regression (SVR) to predict the effort. Our results validate the benefits of using AI methods in real life problems. We attain Pred(25) values as high as 78% in predicting future projects.

1. Introduction

One of the key challenges in software industry is the accurate estimation of the development effort, which is particularly important for risk evaluation, resource scheduling as well as progress monitoring [2]. Inaccuracies in estimations lead to problematic results; for instance, overestimation causes waste of resources, whereas underestimation results in approval of projects that will exceed their planned budgets [2].

Various effort estimation models have been proposed to make reliable predictions to finish projects on time and within the budget. These models can be examined based on methodologies used: Expert-based, analogy-based and regression-based. Expert based models depend on the expert knowledge to use past experience on software projects. Based on a comprehensive review [3], expert based estimation is one of the most frequently applied estimation strategy.

Alternatively, regression-based methods use statistical techniques such as least square regression, in the sense that a set of independent variables explain the dependent variable with minimum error rate [4]. Mathematical models like Barry Boehm's COCOMO [6] and COCOMO II [7] are widely investigated regression-based methods. Parameters of these models are calibrated according to the projects in a company. Thus, they have the drawback of requiring local calibration [2].

To address these issues in regression-based models, AI techniques have been proposed such as step-wise regression, decision tree, artificial neural networks [8, 9] and other rule-based methods [4]. When dealing with effort estimation, these methods also suffer from large deviations and low accuracies. Selecting a single algorithm would often be far from fulfilling the expectations of practitioners from the industry. Therefore, AI techniques should be selected carefully to build an effort estimation model that would best fit into the company needs.

In this study, we have proposed an effort estimation model for the software development division of the Turkish subsidiary of a multinational bank (Bank). Bank has been developing their in-house banking application since 2000. They work with tight schedules due to severe competition in the banking industry. Software managers look for effective strategies to plan their schedule and effectively allocate their resources to meet budget constraints. To address this problem we have built a learning-based effort estimation model and validated the performance of our model using Bank data of completed projects as well as public datasets gathered from various organizations. We have used various AI techniques for feature selection, clustering and estimation. The results of our empirical study reveal that we

successfully predicted 78% of projects' effort with less than 30% error.

2. Description of the model

We can define five important phases twhen building an effort estimation model: Data, Performance Measures, Feature Selection, Algorithm Selection and Model Construction.

2.1. Software Data

According to Fenton and Neil [10], traditional software metrics collection has not addressed the actual purpose of providing information to support quantitative decision making. Basically, each metric in effort estimation should correspond to process, product or resource to measure internal attributes of these categories. We have followed the same principle when collecting software metrics of Bank effort estimation data.

Table 1. Datasets used in model construction

Dataset	# Features	# Projects	Content
Bank	40	29	Banking applications
Albrecht	7	24	Projects from IBM
Deshernais	12	81	Canadian software projects
SDR	22	24	Turkish software projects
ISBSG	14	29	Banking projects only
Cocomo81	17	63	Nasa projects
Nasa93	17	93	Nasa projects

Based on Boehm's COCOMO survey [6], we have defined 22 questions most of which captures process (8) and resource (14) aspects of a software project. We have further used configuration management systems in Bank to derive 18 product metrics. Furthermore, we have used 6 public effort estimation datasets.

As an initial analysis, we have eliminated software metrics collected from Bank that are not predictable prior to the project startup. We also observed the R-Squared coefficient to analyze how well each feature approximates effort values with 95% significance. Figure 1 shows R-Squared coefficients for predictable project features. As it is seen, most of the features have values less than 0.3, which indicates that they are not explanatory to predict the project effort on their own. Therefore, we have adopted an algorithmic approach to find the best subset of features that would have a significant effect on the project effort.

2.2. Performance Measures

To assess the performance of our proposed model, we have used two popular performance measures in the field of effort estimation: Mean Magnitude of Relative Error (MMRE) and Pred(25) [12]. MMRE computes the average magnitude of relative error between the predicted and actual effort values of all projects. Despite criticism, it gives an overall view of the performance of a model using the formula [12]:

$$MMRE = \frac{1}{N}\sum_{i=1}^{N}\frac{|\ actual\ \ (i) - estimated\ \ (i)\ |}{actual\ \ (i)}$$

In order to report the performance of the model for a particular success criteria, Pred(k) is used (in our case k = 25). The formula for calculating Pred(k) is as follows [12]:

$$\Pr ed\ (k) = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1 \text{ if } MRE_{\ i} \le (k/100) \\ 0 \text{ otherwise}\end{cases}$$

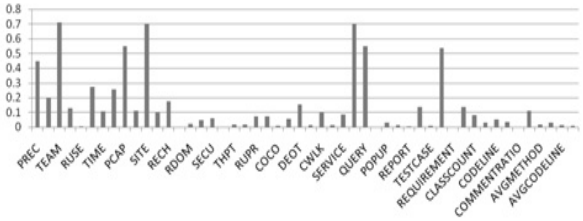

Figure 1. R-Squared coefficients of a regression line for 25 project features

2.3. Feature Selection

The feature selection techniques that we have used in this study can be summarized as Wrapper for feature selection [4], PCA-based weight assignment [14] and correlation based weight assignment [15].

Firstly, we used Wrapper algorithm. Wrapper considers all possible combinations of features and selects a set of features that would result in the minimum error rate [4].

Secondly, *PCA-based weighting heuristic* is inspired from the popular statistical technique, *Principal Components Analysis (PCA)*, which is used for dimensionality reduction. This heuristic, however, is proposed to assign weights to features by taking major principal components from PCA [14].

Thirdly, *correlation-based weighting heuristic* investigates Pearson's correlation coefficients to identify significant relations between project features and the effort. Initially all features have weights 1 [15]. Based on this heuristic, weights of any feature

whose correlation is significant with 95% confidence are doubled. Then, the same ML algorithm is applied to the modified dataset.

Table 2. Results of feature selection techniques on Bank dataset

Data Type	Pred(25)
All features	27.78%
Predictable features only	29.41%
Wrapper Selected Features	47.06%
PCA Weighted Features	52.94%
Correlation-Based Weighted Features	52.94%

Table 2 presents Pred(K) with k=25 for three feature selection approaches. Results show that feature selection techniques applied on predictable features would improve the prediction accuracy. To ensure the statistical validity of these results, Wilcoxon rank sum tests are conducted on all data types.

2.4. Algorithm Selection

For deciding the best algorithm in terms of performance measures, we have tried out various AI techniques. They are single learners such as Linear Regression (LR), Support Vector Regression (SVR), Decision Tree (DT), k-Nearest Neighbor (k-NN) and Multilayer Perceptrons (MLP) and a Mixture of Experts (MOE) with all algorithms. These algorithms are selected based on two criteria: a) they are widely used techniques whose performances are validated on various datasets in effort estimation [4, 8, 17], and b) from the machine learning perspective, each algorithm can achieve strengths on different datasets based on their regression methodologies [1].

We have executed these algorithms on all datasets to see their overall performance. Our methodology for initial model construction can be summarized as follows:

- Select an algorithm from the following set: {LR, SVR, DT, k-NN, MLP, MOE}
- Select a dataset from Table 2.
    - Apply Wrapper with Exhaustive Search using k-NN algorithm (k=3)
    - Use Leave-One-Out strategy:
        - Apply the algorithm on dataset
        - Report MMRE, Pred(25) values
- Apply non-parametric Friedman test to find statistical differences between algorithms [1]. This statistical test checks the null hypothesis:"H0: Performance differences among algorithms are random" against the alternative hypothesis: "H1: Performance differences among algorithms are not random."

If H0 is rejected, we apply Nemenyi's multiple comparisons based on mean ranks [11].

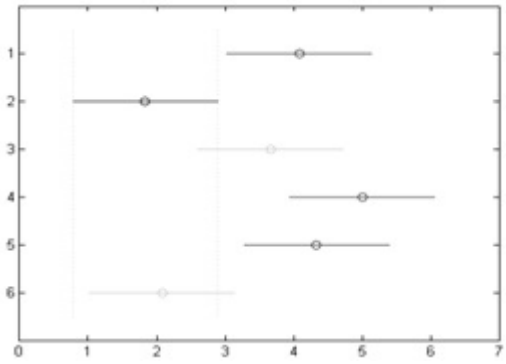

Figure 2. Mean ranks of multiple comparisons among algorithms. In y-axis, each number corresponds to an algorithm such that 1:LR, 2: SVR, 3: MLP,4: k-NN, 5: DT, 6: MOE. Circles show the mean ranks of algorithms. Right end of each line shows from which mean rank onward, another algorithm is outperformed significantly

The results of Nemenyi's multiple comparisons [11] are presented with a graphical interpretation in Figure 2. In Figure 2, when plotting the significance regions, each algorithm is represented with a line having a circle showing the mean rank. Right end of a line is an indicator for an algorithm A to count the number of algorithms, from the algorithm A onwards, that are significantly dominated by A. For instance, the prediction performance of the second line, which corresponds to SVR, is significantly different than other four algorithms, LR, k-NN, DT, MOE. Since it has the highest number of wins in terms of significant differences, we have decided to use SVR as the algorithm of our effort estimation model. The performance of SVR on all datasets can be seen in Table 3 for simplicity. Table 3 shows that we have still very high values for MMRE and values lower than 70% for Pred(25).

2.5. Model Construction

Although we have invested a lot of effort in data quality and collection, some projects in Bank dataset are outliers and they degrade the performance of SVR. To overcome that, we have decided to form project groups that provide consistency in terms of metrics and effort values within themselves. Therefore, we have applied clustering before estimation.

Table 3. Effort Estimation Accuracy with SVR

Dataset	MMRE	Pred(25)
Bank*	41%	47%
Albrecht	126%	25%
Deshernais	320%	18%
SDR	275%	0%
ISBSG	36%	42%
Cocomo81	479%	9.5%
Nasa93	154%	19%

We have used *Expectation Maximization* algorithm to find clusters with similar project features and effort values. In Bank data, after selecting 7 project features as the best subset from *Wrapper* algorithm, we have applied *SVR* on each cluster. Each cluster has been trained only with the projects inside the cluster and tested on these projects using leave-one-out validation. Results are summarized in Table 4.

Table 4. Effort estimation results with final model

		Best Cluster			
Dataset	**Clusters**	**Projects**	**MMRE**	**Pred(25)**	**Pred(30)**
Bank	3	9	41%	67%	78%
Cocomo81	3	16	20%	72%	83%
Nasa93	6	15	25%	66%	73%

The proposed model on all datasets is able to obtain Pred(30) values higher than 70% on the clusters with the best accuracy. Although there are still inconsistencies about the correctness of Bank data, we have managed to form a cluster, in which we can estimate 7 out of 9 projects with relative error less than 30%. Similarly, for NASA datasets, MMRE values are below 30% on the best cluster with 16 and 15 projects on Cocomo81 and Nasa93. Therefore, we can conclude that AI approaches would significantly improve the estimation accuracy when accurate and consistent data is given.

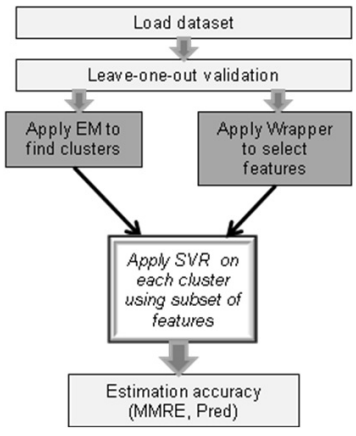

Figure 3. Experimental design of the model

3. Conclusion

In this study, we have built an effort estimation model to predict the project effort by using AI techniques. We have collected software metrics and actual effort values from 29 projects in Bank and used public datasets to externally validate our proposed model. Our results show that combinations of AI techniques provide significant improvements in estimation accuracy. Furthermore, we have done statistical tests to check the significance of these estimation results. Statistical tests show that SVR is the best among six algorithms. When used with clustering, the model performance increases significantly, from, on the average, 25% to 68% in terms of Pred(25) in three datasets.

4. Acknowledgements

This research is supported in part by Turkish Scientific Research Council (TUBITAK) under grant number EEEAG108E014 and IBTech Inc.

5. References

6. Alpaydın, E. 2004. *Introduction to machine learning*. Cambridge: MIT Press.
7. Boehm, B., Abts, C., Chulani, S. 2000. *Software development cost estimation approaches: A survey*. Annals of Software Engineering (10): 177–205.
8. Jorgensen M. 2004. A review of studies on expert estimation of software development effort. Journal of Systems and Software (70): 37-60.
9. Menzies, T., Chen, Z., Hihn, J., & Lum, K. 2006. *Selecting best practices for effort estimation*. IEEE Transactions on Software Engineering (32): 883–895.
10. Li, J., Ruhe, G. 2007. *Decision support analysis for software effort estimation by analogy*. In Proceedings of the Third International Workshop on Predictor Models in Software Engineering (PROMISE 2007). Minnesota, USA.
11. Boehm, B.W. 1981. *Software Engineering Economics*. Upper Saddle River, NJ, USA: Prentice Hall PTR.
12. Boehm, B. W., Abts, C. Brown, A.W., Chulani, S., Clark, B. K., Horowitz, E., Madachy, R., Reifer, J.D., and Steece, B. 2000. *Software Cost Estimation with Cocomo II*. Upper Saddle River, NJ, USA: Prentice Hall PTR.
13. Shepperd M., Kadoda, G. 2001. *Comparing software prediction models using simulation*. IEEE Transactions on Software Engineering, 1014–1022.
14. Wittig, G., Finnie, G. 1997. *Estimating Software Development Effort with Connectionists Models*. Information & Software Technology (39): 469-476.
15. Fenton, N., Neil, M. 2000. *Software Metrics: Roadmap*, In Proceedings of 22nd International Conference on Software Engineering, ACM Press ISBN 1-58113-253-0, 357-370.
16. Demsar, J. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. J. Machine Learning Research, (7): 1-30.
17. Port, D., Korte, M. 2008. Comparative Studies of the Model Evaluation Criterions MMRE and PRED in Software Cost Estimation Research. In ESEM 2008, 51-61.