AI Based Models for Estimating Software Projects' Efforts

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. Results validate the benefits of using AI in real life applications: 78% accuracy in predicting project efforts.

Keywords: Software effort estimation, regression, machine learning, model construction.

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 (Boehm, Abts, Chulani 2000). 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 (Boehm, Abts, Chulani 2000).

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 totally depend on the expert knowledge to use past experience on software projects. Based on a comprehensive review (Jorgenson 2004), expert based estimation is one of the most frequently applied estimation strategy. They provide a broad definition of estimation, such a "gut feelings" or "structured estimations" supported by historical data and processes of the companies. Selecting one of these "heuristics" while making expert-based estimations may sometimes perform better than formal methods. However, when the prior knowledge of experts is strongly biased, then their estimation accuracy significantly degrades.

Alternatively, regression-based methods use statistical techniques such as least square regression, such that a set of independent variables, *project features,* are converted to the dependent variable, *effort,* with minimum error (Menzies et al. 2006). Mathematical models like Barry Boehm's COCOMO (Boehm 1981) and COCOMO II (Boehm et al. 2000), are widely investigated as 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. Local calibration, on the other hand, is a very labor-intensive task as it requires regular fine-tunings of parameters with new data. Once the calibration is stopped, the model would easily depreciate. Therefore, the fast changes in software development make it very difficult to build a parametric model that fits for all software domains (Boehm, Abts and Chulani 2000).

To address these issues in regression-based models, AI techniques have been proposed such as step-wise regression, decision tree, artificial neural networks (Shepperd and Kadoda 2001, Wittig and Finnie 1997) and other rule-based methods (Menzies et al. 2006). AI-based (algorithmic) models learn from previous projects and based on the knowledge gathered from past projects, new project's effort is estimated. 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, IBTech. IBTech has been developing their in-house banking application since 2000. They work with tight schedules due to severe competition in the banking industry and changing and tighter banking regulations. 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 IBTech data from completed projects in terms of project features and actual efforts 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.

In this paper, we explain the phases of our project and we emphasize the benefits of AI in real life projects. The outline of this paper is as follows: First, we briefly explain the phases of this case study. We then provide details on construction of the model, its parameters, AI techniques and performance measures. Finally we present our empirical results to examine the importance of AI in software effort estimation domain.

Description of the Model

There are five fundamental issues to decide while building an effort estimation model: Data, Performance Measures, Feature Selection, Algorithm Selection and Model Construction.

Software Data

According to Fenton and Neil (2000), traditional software metrics collection has not addressed the actual purpose of providing information to support quantitative decision making. It often provides little support for managers that aim to use measurement for risk and what-if analysis. Therefore, before building intelligent oracles for estimation, we must define the actual metrics that would measure what is intended to measure. 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 IBTech effort estimation data.

Based on Boehm's COCOMO survey (1981), 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 IBTech to derive 18 product metrics. All metrics set is summarized in Appendix A. Initially, a filtering approach based on expert judgment and statistical analysis has been applied to reduce the number of metrics that have significant relation with software effort.

Furthermore, we have used 6 public effort estimation datasets whose metrics and actual effort values are collected from various organizations in United States, Canada and Turkey. They contain a variety of project features, some of which are collected via COCOMO questionnaire (Boehm 1981). Table 1 represents dataset features.

Table 1. Datasets used in model construction

Dataset	# Features	# Projects	Content
IBTech	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

Expert Judgment. As an initial analysis, we have eliminated software metrics collected from IBTech that are predictable prior to the project startup. There are certain features (such as "bug count") that can be easily collected from past projects, whereas it is very unlikely to estimate at the beginning of a new project. With this kind of a filtering, 15 project features are defined as unpredictable and removed from the project set.

Statistical Analysis. Selecting only predictable metrics based on expert judgment, we have fitted a regression line and 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.

Performance Measures

To assess the performance of our proposed model, we have used two popular performance measures: *Mean Magnitude of Relative Error* (*MMRE*) and *Pred(k)* in the field of effort estimation (Port and Korte 2008). *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 (Port and Korte 2008):

$$MMRE = \frac{1}{N}\sum_{i=1}^{N}\frac{Effort_{Estimated}(i) - Effort_{Actual}(i)}{Effort_{Actual}(i)}$$

Pred(25) defines the average fraction of the magnitude of relative error off by no more than 25%. In order to report the performance of the model for a particular success criteria, *Pred(k)* is used. The formula for calculating *Pred(k)* is as follows (Port and Korte 2008):

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

Inverse to *MMRE,* high *Pred* values are desirable. Although both measures provide meaningful and precise interpretation of "accuracy measure" (Port and Korte 2008), we would often observe *Pred* values to count the number of estimates that are below a pre-defined error value during our experiments

Feature Selection

In the literature, software metrics are used in the estimation model either as unweighted (Shepperd and Schofield 1997, Li et al. 2007) or with some weights assigned based on different heuristics. Some of these techniques that we have also used in this study can be summarized as *Wrapper* for feature selection (Menzies et al. 2006), *PCA-based weight assignment* (Tosun, Turhan, Bener 2009) and *correlation based weight assignment* (Mendes et al. 2003).

First, Wrapper algorithm considers all possible combinations of features, which is 2^n-1 where n is the number of features, and selects a set of features which would result in the minimum error rate (Menzies et al. 2006). It is better to use only a subset of features if they are able to represent the effort value better than using the whole set. The reasons for this are that all possible features would a) introduce noise into the model and b) increase the dimension of the space such that the estimation accuracy of the algorithm may degrade. Thus, we have applied Wrapper to our initial dataset by selecting a popular machine learning (ML) algorithm, *Support Vector Regression (SVR)* in effort estimation studies. Our objective here is to evaluate project features rather than selecting the best algorithm for effort estimation. Therefore, we have selected SVR to observe the impact of three feature selection techniques on model performance.

Second, *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 (Tosun, Turhan, Bener 2009). Weights close to zero indicates that those features have particular importance on the project effort. We have also utilized this heuristic before applying SVR.

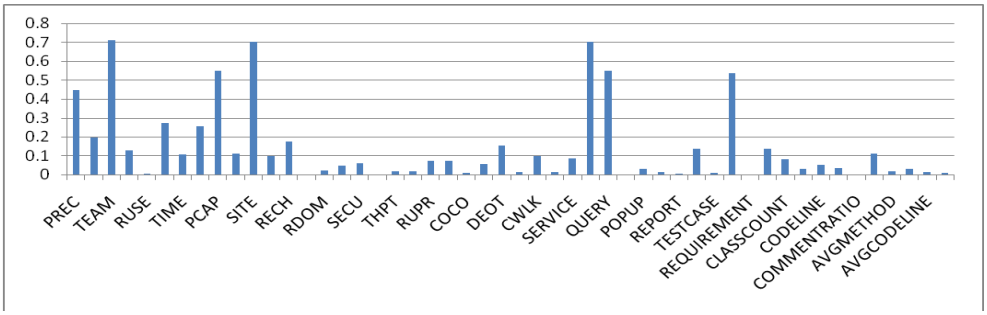

Figure 1. R-Squared coefficients of a regression line for 25 project features

Third, *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 (Mendes et al. 2003). Based on this heuristic, weights of any feature whose correlation is significant with 95% confidence are doubled. Then same ML algorithm is applied to the modified dataset.

Table 2 presents *Pred(K)* with *k=25* for three feature selection approaches on all. 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. This non-parametric test reveals that three feature selection algorithms are not significant from each other in terms of *Pred(25)* values (Wilcoxon 1945). Although they are proved to improve the prediction accuracy, selecting one out of the others would not change the prediction performance significantly. Due to this fact, we have selected *Wrapper feature selection algorithm,* since it exhaustively looks at all feature combinations and selects the best feature set.

Table 1. Results of feature selection techniques on IBTech 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%

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 (Shepperd and Kadoda 2001, Menzies et al. 2006, Corazza et al. 2009), and b) from the machine learning perspective, each algorithm can achieve strengths on different datasets based on their regression methodologies (Alpaydin 2004).

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)
  - Using *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 (Alpaydin 2004). This statistical test checks the null hypothesis:
  "H_0: Performance differences among algorithms are random"
  against the alternative hypothesis:
  "H1: Performance differences among algorithms are not random."
- If H0 is rejected, apply Nemenyi's multiple comparisons based on mean ranks (Demsar 2006). These comparisons would select the algorithm from which mean rank onward, most of the other algorithms are significantly outperformed.

In our methodology, we have used *leave-one-out validation* to select one project for test set and the rest is left as training set, since the sizes of datasets are relatively small (Alpaydin 2004). We have performed Friedman's test for classifier comparisons, since it is non-parametric alternative to ANOVA and relies on less restrictive assumptions (Demsar 2006). To accomplish Friedman's significance test, we have built our model using multiple algorithms on multiple datasets. Our aim is to assess whether the differences in terms of *Pred(25)* and *MMRE* values among algorithms are statistically different. Based on our analysis, this test rejects the null hypothesis with 95% confidence level. That is; one or more algorithms are statistically different, i.e. superior, from others, although we could not identify which of them. To select which

algorithm(s) significantly dominates others in terms of performance measures on seven datasets, we have done Nemenyi's multiple comparisons (Demsar 2006) whose results 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)*.

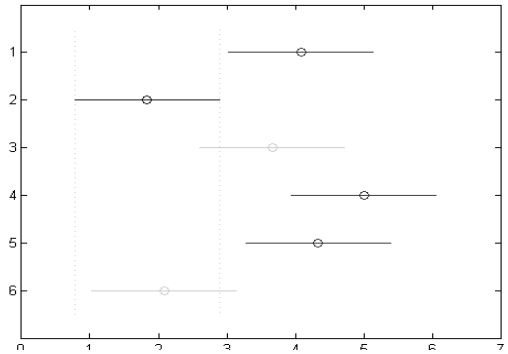

Figure 3. Mean ranks of multiple comparisons among algorithms. In y-axis, each number corresponds to an algorithm such that 1:LR, 2: SVR, 3: MLP,4: k-NN, 5: DT, 6: MOE. Circles show the mean ranks of algorithms. Right end of each line shows from which mean rank onward, another algorithm is outperformed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

Table 3. Effort estimation accuracy with SVR

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

Model Construction

Although we have put high effort on data quality and collection, some projects in IBTech dataset are outliers, i.e. their actual effort values could not be accurately collected, so they do not look like the others in the dataset. Such projects in all datasets degrade the performance of *SVR* during training phase. To overcome this issue, we have decided to form project groups that provide consistency in terms of metrics and effort values within groups. We have applied *Clustering* before estimation via *SVR*.

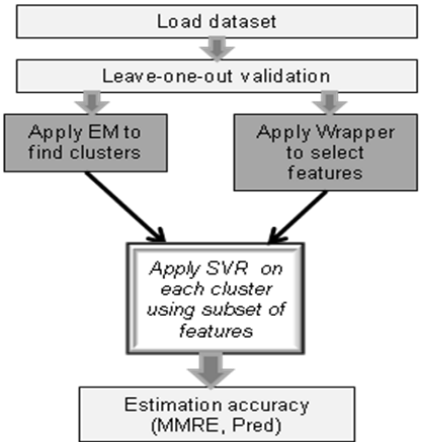

Figure 2. Experimental design for model construction

The main steps for building the effort estimation model are described in Figure 3. We have used *Expectation Maximization* algorithm to find clusters with similar project features and effort values. In IBTech 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.

We have built our final model on IBTech dataset as well as two NASA projects to assess the performance in multiple organizations. Results are summarized in Table 4. 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 IBTech 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. Our results would also be improved when actual efforts of the other projects in IBTech dataset are re-calculated to avoid large deviations in the model.

Table 4. Effort estimation results with the final model

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

Conclusion

In this study, we have built an effort estimation model to predict the project effort at the beginning by using AI

techniques. We have collected software metrics and actual effort values from 29 projects in IBTech and used public datasets to satisfy external validity of such models. Combinations of AI techniques provide significant improvements on estimation accuracy on datasets. 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.

As a future direction, we aim to divide project effort into phases and build a model that would predict phase-based efforts. We plan to select features for each phase separately and built an effort estimator that would help decision making in every phase of software development lifecycle.

Appendix A

Table A1. Software metrics collected in this study

Metric Category: Product	
Component count	Query Number
Module Difficulty	Screen Number
Data Exchange	Popup Number
Deploy count	Region Number
Service Number	Report Number
Batch Number	Table Number
Requirement Count	Method Count
Business Request Count	Lines of Code
Class Count	Lines of Comment
Metric Category: Process	
TIME: Time constraint on project	RUPR: Re-use percentage
RECH: Percentage of extra time due to requirements change.	NEWP: New code percentage
CWLK: Percentage of code walkthrough.	Test Case Count
SECU: Percentage of extra time due to security constraints.	Bug Count
Metric Category: Resource	
ACAP: Analyst capability	SCED: Schedule constraints
PCAP: Programmer capability	RCAP: Architect capability
RELY: How reliable the developed project shall be	DOCU: The percentage overhead due to documentation needs.
RDOM: Requirement team's domain knowledge	TOOL: Tool usage percentage within this project.
TMNO: Number of team's worked in this project	SITE: How distributed the teams worked in this project were
THPT: Percentage of project developed by third parties	DEOT: Development team's workload other than the project.
THIM: Whether there is third party involvement	TEAM: The cohesion of the teams worked in this project

References

Alpaydın, E. 2004. *Introduction to machine learning*. Cambridge: MIT Press.

Boehm, B., Abts, C., Chulani, S. 2000. *Software development cost estimation approaches: A survey*. Annals of Software Engineering (10): 177–205.

Jorgensen M. 2004. *A review of studies on expert estimation of software development effort*. Journal of Systems and Software (70): 37-60.

Menzies, T., Chen, Z., Hihn, J., & Lum, K. 2006. *Selecting best practices for effort estimation*. IEEE Transactions on Software Engineering (32): 883–895.

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.

Boehm, B.W. 1981. *Software Engineering Economics*. Upper Saddle River, NJ, USA: Prentice Hall PTR.

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.

Shepperd M., Kadoda, G. 2001. *Comparing software prediction models using simulation*. IEEE Transactions on Software Engineering, 1014–1022.

Wittig, G., Finnie, G. 1997. *Estimating Software Development Effort with Connectionists Models*. Information & Software Technology (39): 469-476.

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.

Demsar, J. 2006. *Statistical Comparisons of Classifiers over Multiple Data Sets*. J. Machine Learning Research, (7): 1-30.

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.

Shepperd, M, Schofield, C. 1997. *Estimating software project effort using analogies*. IEEE Transactions on Software Engineering (23): 736–743.

Tosun, A., Turhan, B., Bener, A. 2009. *Feature weighting heuristics for analogy-based effort estimation models*. Expert Systems with Applications (79), 1:

Mendes, E., Watson, I., Triggs, C., Mosley, N., Counsell, S. 2003. *A comparative study of cost estimation models on web hypermedia applications*. Empirical Software Engineering, (8): 193–196.

Wilcoxon, F. 1945. *Individual comparisons by ranking methods*. Biometrics Bulletin, (1): 80–83.

Corazza, A., Di Martino, S., Ferruci, F., Gravino, C., Mendes, E. 2009. *Applying Support Vector Regression for Web Effort Estimation using a Cross-Company Dataset*. In Proc. Third International Symposium on Empirical Software Engineering and Measurement, 191-203.