Adaptive Machine Learning Models for Software Effort Estimation: Evidence from Turkish Software Industry Data

Abstract—— The main aim of software engineering is the development of projects that achieve the desired outcome within a budget and a specified timeframe. This project's most important budgetary factor is the effort. This is because, for software development, hiring a greater number of people than necessary means a loss of revenue, and hiring a smaller number of people than necessarily means the extension of the project's timeframe. This study's main aim is the analysis of software effort estimation, and the problems associated with it, like the budget and extension of the project's timeframe, are solved through the development of a model that makes use of machine learning techniques. Experiments are carried out using data from Turkish software companies and statistics, and the experiments prove that the best approach for a given data set might differ, indicating that the use of a single model does not guarantee the best results.

Keywords—component, formatting, style, styling, insert

I. INTRODUCTION

Since the 1970s, the focus has been on software cost estimation, which inherently includes software work estimation. Various software cost estimation models, like SLIM, Checkpoint, Price-S, Seer, COCOMO, and others, have been proposed to address software cost estimation. However, there is a common factor in all these models: as software size and importance increase, complexity increases, making software cost estimation a more difficult task. In order to define cost as a function of various cost parameters, several approaches have been proposed. These approaches vary from regression analysis and judgment to parametric approaches like SLIM and COCOMO. The biggest challenge that we face while doing software cost estimation is that there is a lack of data. Organizations may not be willing to provide information or may not want to collect information. We also used data from previous studies, such as those from USC and NASA. Once we had our data, we used machine learning to perform our estimation, including backpropagation networks, regression trees, radial basis functions, and support vector regression, and then compared our results. These are all techniques that use previous data to estimate new data. The data that we used is not from a single company, which is a positive because it could open the door for a universal model for effort estimation.

In order to learn from past experiences, these techniques employ historical data as input. They can then be applied to estimation. For the sake of a generic estimation, the data has been multi-organizational rather than business particular. Additionally, this opens up the potential of developing a universal effort estimation model. The creation of a hybrid effort estimation model that combines machine learning techniques with model-related data is the main contribution of this study. Next, we evaluated the performance of various machine learning techniques using our suggested model.

The second part of this research presents an overview of the past cost estimation models that are associated with our model. In the third part of this research, we will outline our problem. In the fourth part of this research, we will outline our methodology and the machine learning techniques that we will use in our model. In the fifth part of this research, we will present our experimental results and a comparison of our results. In the last part of this research, we will outline our conclusions and future work.

Literature Review

Throughout the software development cycle, precise and trustworthy cost estimates are necessary to assess the viability of software projects and ensure that resources are allocated appropriately. Numerous techniques have been put out in the literature to forecast software costs [3]. Expert knowledge estimate is one of the most widely used software estimation techniques. According to this approach, the degree to which a new project aligns with the expert's knowledge and recall of data from previous projects determines how reliable estimates based on expert judgement are [2, 3]. The fact that it is challenging for someone else to replicate and apply an expert's knowledge and experience is a significant issue with this approach [2, 3]. The fact that it is challenging for someone else to replicate and apply an expert's knowledge and experience is a significant issue with this

approach [2, 3]. Additional approaches in the literature include learning-oriented strategies and parametric models (COCOMO, COCOMO II) [4, 5, 6]. The estimation of development time and effort in parametric models depends on several variables [3]. Several algorithms and parameters (variables) form the core of such an estimation model. The content of a database of finished projects and research serves as the basis for the parameter values and algorithm type. The fact that it is challenging for someone else to replicate and apply an expert's knowledge and experience is a significant issue with this approach [2, 3]. Additional approaches in the literature include learning-oriented strategies and parametric models (COCOMO, COCOMO II) [4, 5, 6]. However, the estimation of the development period and the amount of development effort in parametric models depends on a number of factors. At the core of the estimation process in these models are a set of algorithms and parameters (variables). The parameters are based on a database of completed projects and studies. When you compare the accuracy of analogy-based estimates with the accuracy of estimates from the stepwise regression-based prediction models, you find that analogy provides higher accuracy. Neural networks may be used as an alternative to the mean least squares regression. The mean least square error is the foundation of these regression techniques. This method's basic premise is to train the network using historical data, then modify the network's parameters to provide improved estimates. In software engineering, it has been applied to schedule estimates and fault prediction [9].

Problem Statement

Software quality is dependent on a number of aspects of software development. In the case of software development companies, the main aim is to start a project and finish it within a reasonable budget and timeframe. The budget largely correlates with the amount of time taken and the staff involved. In other words, the budget largely correlates with the concept of 'effort.' In reality, the cost of a software development project is largely determined by the 'effort.' The estimation of 'effort' is a vital step in the development of software. At the very outset of a software development project, an estimate of the 'effort' is necessary to determine the budget and the timeframe. Software development processes are constantly changing with the introduction of new technologies and applications. A developing system is necessary to address the 'effort estimation problem,' especially in the present-day software world. By developing a learning system that combines various machine learning techniques with datasets on software effort estimates, this research aims to improve software effort estimation. The key motivation for using this learning system to improve effort estimation is to incorporate new project information into the process. The model is updated whenever there are changes to the estimation process due to the addition of new projects. In this way, the major limitation of parametric estimation models has been addressed. As new projects are developed, the learning system adjusts to improve the accuracy of effort estimates while capitalizing on the accuracy of previously developed metrics, such as COCOMO and COCOMO II.

Data and Model

We infer that software engineering research serves as the link between theory and practice for the specific effort estimation problem [12]. In addition to gathering information from Turkish software companies, we also used data from open data repositories. We attempt to develop models for effort estimation through machine learning techniques.

We make use of publicly accessible data that has been gathered by other software researchers worldwide. In this manner, two distinct datasets—NASA and USC—have been acquired [11]. These datasets' primary feature is that numerous researchers have already used them, demonstrating their reliability for use in any kind of research [8, 11]. Data from 63 projects and 17 measures, such as effort value per person-month, make up USC data. The NASA dataset shares 17 indicators with the USC dataset and includes data from 60 studies.

The SoftLab Data Repository (SDR), a dataset gathered from software companies in Turkey, includes 23 projects ranging from online banking apps to different automation projects. We created a thorough tutorial on data collection and used a questionnaire to be completed [13].

The SDR dataset is both new and represents the characteristics of the Turkish software sector, whereas the other datasets are somewhat old.

Method used in the Implementation

In summary, machine learning methods are used for classification, learning, and estimation problems, with a focus on the use of historical data. In practice, the scope of machine learning extends beyond speech and pattern recognition, and includes financial forecasting and analysis, etc. In the present study, several machine learning methods are used as tools for estimating the effort using historical data, which has been built up through experience, following the COCOMO guidelines, based on the COCOMO and COCOMO II models, as depicted in Figure 1. In the present study, the machine learning methods used for effort estimation are backpropagation multilayer perceptron, regression trees, radial basis function networks, and support vector regression. In addition, principal components analysis is used.

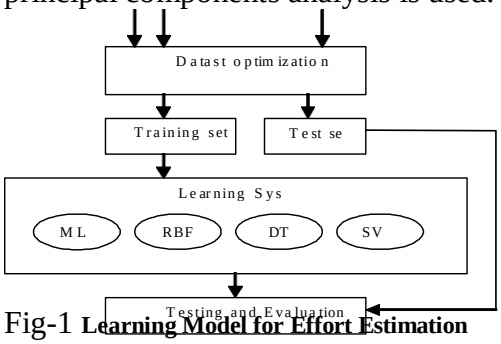

Fig-1 **Learning Model for Effort Estimation**

Evaluation of Experimental Results

The performance of effort estimation datasets and models may also be measured using different methods, such as PRED(L), MMRE, and correlation. These methods are commonly used and have been adopted for the evaluation of effort estimation models [2]. PRED, mdMRE, and MMRE are the standard measures for evaluating the accuracy of software project prediction models [15, 16]. PRED(L) is obtained from relative error, which is the difference between actual and estimated effort. The formula for PRED(L) is given below:

Equation

$$PRED\ (L) = \frac{k}{N} \qquad (1)$$

In this case, the forecast accuracy is at the level of L, where L is defined in accordance with the accuracy expectations and k is the number of observations where MRE is less than or equal to L. Typically, values of L are 25 or 30 [2]. L is assumed to be 30 in the experiments. If the PRED value exceeds or equals 75% of the anticipated values, it is proposed that the model is acceptable [14]. Having a measure that is independent of the output value is the underlying justification for adopting relative error measures, like MRE, as opposed to absolute ones. The rationale for the use of the relative error notion can be easily understood by software researchers and practitioners, since it is logical to predict the effort for small and large systems using relative error, rather than absolute error [15, 16]. This rationale for the use of relative error is further supported by the use of PRED, since it is logically linked with the selection of models that are highly accurate, i.e., a high percentage of accuracy. On the other hand, the rationale for the use of MMRE stems from the need for discrimination between various models, with a preference for models with a smaller MMRE. In addition, the use of MMRE is motivated by the need for a quantitative evaluation of the uncertainty of the prediction, where a smaller MMRE implies a smaller uncertainty or inaccuracy.

Experimental Results

1.1 Results for the NASA Dataset

There are two configurations for the experiments based on the size of the training set and the test set. In the first case, the training set size is 50 projects and the test set size is 10 projects. In the second case, the training set size is 40 projects and the test set size is 20 projects. In all cases, cross-validation is used.

Table 1 presents the results for the 16 different cases. In all cases in the table, the performance of the RT method is higher than the other methods. It is interesting to note that in the case of the other methods, a trend is observed such that the lower the size of the

data set, the higher the error. Another interesting observation is the decrease in the error for the MLP, RBF, and SVR methods when the PCA method is applied initially. This may be due to the fact that some of the attributes are particularly relevant to the COCOMO model, as indicated by the values for the MMRE.

In the case of the application of the parametric model, the value for the mdMRE is 18.63 and the value for the MMRE is 29.50. In this case, the performance of the RBF, RT, and SVR methods is at least as good as the performance of the conventional method. This is because the PRED(30) for the three methods is more than 75. In the case of the parametric model, the PRED(30) is less than 75.

NASA (# of training)	Without PCA		With PCA	
	MMRE (%) mdMRE(%)	PRED (%)	MMRE (%) mdMRE(%)	PRED (%)
MLP (50)	110.94 68.01	33	109.96 65.75	34
MLP (40)	166.57 76.89	23	130.62 74.58	24.50
RBF (50)	22.11 14.44	73.25	20.19 11.78	75.25
RBF (40)	22.39 15.47	73	20.27 11.99	75
DT (50)	12.85 5.31	83.75	22.51 11.03	72
DT (40)	13.10 6.29	83.50	23.22 12.58	67.75
SVM (50)	25.72 9.39	75	23.11 7.53	77
SVM (40)	47.56 14.95	69.50	23.43 8.71	75.75

Experimental Results for USC dataset

In the present experiments, two different configurations are analyzed, differentiated by the quantity of projects used for the training and the test sets. Firstly, the machine learning methods are used for a configuration consisting of 53 projects for the training set and 10 projects for the test set, and then for a configuration consisting of 42 projects for the training set and 21 projects for the test set, using cross-validation for the experiments.

As for the results of the experiments, the results of the 16 experiments are presented in Table 2. In the experiments, the following observations are noted: the MMRE values increase with the decrease of the training set size, and the increase in the values is small for the RBF and DT methods. In addition, the initial use of the PCA method decreases the MMRE values for the MLP, RBF, and SVR methods, and it increases the errors for the RT method. In contrast with the NASA data set, the PCA method has a significant effect in the present data set, and the optimal combination of methods used is the "RBF + PCA" method, indicating the presence of superfluous attributes in the data set, most probably due to the low deviation value.

If a parametric model, such as COCOMO, is used, then the mdMRE is 28.47, the MMRE is 49 50.79. This, it is understood, the performance of RBF, RT, and SVR is superior to or comparable with the performance of the COCOMO parametric model. This proves that the learning system is highly compatible with the COCOMO

model.

Table 2. Summary of experimental results for USC dataset

USC (# of training)	Without PCA		With PCA	
	MMRE (%) mdMRE(%)	PRED (%)	MMRE (%) mdMRE(%)	PRED (%)
MLP (53)	116.56 82.31	11.49	91.73 75.37	14.62
MLP (42)	126.18 82.80	8.12	114.99 78.13	8.50
RBF (53)	13.38 8.42	84.76	12.15 7.44	88.33
RBF (42)	14.56 9.61	84.50	12.46 7.46	86
DT (53)	13.57 3.98	85.24	33.12 9.15	72.14
DT (42)	14.25 5.18	85	37.97 17.19	66
SVM (53)	30.94 14.83	75.5	33.28 13.77	82
SVM (42)	31.02 16.45	70	33.39 15.66	73.81

Experimental Results for SDR dataset

These experiments include two types of experiments, depending on the size of both training and test sets. First, machine learning methods are applied with a training set including 20 projects and a test set including 3 projects, then with a training set including 16 projects and a test set including 7 projects. All the experiments are performed by cross validation.

Results of the 16 different experiments are shown in Table 3. The general

The observed trend is nearly the same as that for the NASA and USC data sets. Here, inturn, MMRE increases with a decrease in the size of the training set.

Table 3. Summary of experimental results for SDR dataset

SDR (# of training)	Without PCA		With PCA	
	MMRE (%) mdMRE(%)	PRED (%)	MMRE (%) mdMRE(%)	PRED (%)
MLP (20)	103.59 64.89	31.50	82.82 66.60	32.00
MLP (16)	199.41 69.64	25.00	141.88 89.51	29.50
RBF (20)	19.38 9.57	79.00	18.40 8.31	79.50
RBF (16)	22.81 15.46	74.00	20.69 10.10	75.00
DT (20)	13.14 4.36	87.50	18.83 7.55	77.50
DT (16)	12.50 5.02	87.00	24.35 11.54	73.00
SVM (20)	25.96 7.72	74.50	19.63 5.17	80.50
SVM (16)	49.41 8.45	70.50	22.59 7.02	79.25

For this data set, in MLP, RBF, and SVR, when PCA is first applied, it results in a decrease in MMRE. But it results in an increase in errors when applied to the RT method. This case is somewhat similar to that of NASA data set, as seen earlier, where RT is better than others. RT method is acceptable as an estimator since PRED is greater than 75.

In using the COCOMO parametric model, an mdMRE of 1860.32 and an MMRE of 3368.15 is 8.7. With this, it is noted that the output of all methods is better than the output of the COCOMO parametric model. This, in short, indicates that the learning system outperforms the parametric model. This may be because contemporary software methods have tools that can automatically generate most of the code themselves. This makes the program size larger, where the effort is minor with reference to the program size. Though there is a metric for TOOL of the COCOMO model; however, it fails to adhere to the same and does not reflect such a consideration on the impact on the estimates. Thus, the strength of our model can be more apparent here.

Conclusions and Future

In this research, we have conducted experiments for software effort estimation using different machine learning techniques on three different data sets, one of which is a native data set. We have achieved acceptable results for the purpose of software effort estimation, especially when we consider the results achieved by the parametric models, which have been previously used. As a conclusion of this study, it has been understood that parametric models themselves cannot solve the problem of software effort estimation, and the problem should be solved with an evolving system, not a static one. Moreover, the metrics, which are based on the models, may be adjusted or different metrics may be added.

Models developed based on limited data sets, like COCOMO and COCOMO II, normally reflect the characteristics of the data used to develop the models. A high degree of accuracy is normally reflected, but this has the disadvantage of limiting the application of the model. As we can see from the results of the experiment, the COCOMO model has a good ability to estimate efforts based on the two data sets given, i.e., USC and NASA. It is therefore not surprising to see that the result of the model for these two data sets is good. However, the COCOMO II model fails to perform sufficiently for the new SDR data set, which we have collected. The reason for this effect, which we perceive, is that a greater percentage of the total code is normally developed with the help of the various software tools like Visual Studio or Java plug-ins. These tools automatically generate code for the developers. Even though there is a corresponding metric called TOOL for both models developed, i.e., COCOMO and COCOMO II, this effect is not sufficient. This is why, with huge values of size, the results for effort is also very high, which is leading to huge inaccuracy in the SDR dataset, as the majority of projects developed in the dataset take around 2-3 years and heavy usage of tools is involved. This shows, in a way, that the preference to use the learning system instead of the parametric model is much higher, as the results of the parametric model might not be good, considering the limited amount of data available.

As a future direction, the learning system may be calibrated or different methods can be incorporated to test the system further. As another future work, the dataset may be improved such that it represents the software development characteristics of the software development organizations in Turkey and come up A better model is needed to enable the industry to cater to its needs in this context. This is required since there is an obvious lack of work and knowledge related to the particular problem within the industry itself.

1. References

[1] Boehm B., Abts C., "Software *Development Cost Estimation Approaches – A Survey*", University of Southern California, 1998.

[2] Briand L. C., Emam K. E., Surmann D., Wieczorek I., Maxwell K. D., "*An Assessment and Comparison of Common Software Estimation Modeling Techniques*", International Software Engineering Research Network Technical Report, 1998.

[3] Heemstra, F. J.,"*Software Cost Estimation Models*", IEEE Software, 1990.

[4] Boehm, B. W., "*Software Engineering Economics*", Prentice Hall, 1981.

[5] Rubin, H. , "*ESTIMACS*", IEEE, 1983.

[6] University of Southern California, *USC COCOMO Reference Manual*, 1994.

[7] Shepperd, M., Schofield, M., *"Estimating Software Project Effort Using Analogies"*, IEEE Transactions on Software Engineering, Nov 1997.

[8] Cowderoy, A.J.C. and J.O Jenkins, *"Cost estimation by analogy as a good management practice"*, *Proceeding.Software Engineering*, 1988.

[9] Evren Ceylan, F. Onur Kutlubay, Ayşe B. Bener, "*Defect Identification Using Machine Learning Techniques*", in Proceedings of 32nd Euromicro Conference on Software Engineering and Advanced Applications, 2006

[10] Mitchell, T. M., *Machine Learning*, McGraw-Hill and MIT Press, 1997.

[11] Basili, V., McGarry, F., Pajerski, R., Zelkowitz, M., "Lessons learned from 25 years of process improvement
: The rise and fall of the nasa software engineering laboratory", Proceedings of the 24th International Conference on Software Engineering (ICSE) 2002, Orlando, Florida, 2002.

Colin Potts, *Software Engineering Research Revisited*, IEEE Software, September 1993. COCOMO II Cost Estimation Questionnaire, http://sunset.usc.edu/research/COCOMOII/Docs/cform22.pdf, 2006.

[12] Software Cost Estimation : Metrics and Models, *http://sern.ucalgary.ca/courses/seng/621/W98/johnsonk/ cost.htm#Original%20COCOMO, 2006.*

[13] Fenton, N. E., Pfleeger, S.L., *Software Metrics: A Rigorous and Practical Approach*, International Thomson Computer Press, 1997.

[14] Conte, S. D., Dunsmore, H. E., Shen, V. Y., *"Software Engineering Metrics and Models"*, Benjamin- Cummings, Menlo Park CA, 1986.

[15] University of South California Software Engineering Research, *http://sunset.usc.edu/research/, 2006.*