Multinomial Logistic Regression Applied on Software Productivity Prediction

Panagiotis Sentas, Lefteris Angelis, Ioannis Stamelos

Department of Informatics, Aristotle University
54124 Thessaloniki, Greece
Email: {sentas,lef,stamelos}@csd.auth.gr

Abstract. In software cost estimation various methods have been proposed to yield a prediction of the productivity of a software project. Most of the methods produce point estimates. However, in practice it is more realistic and useful to have a method providing interval predictions. Although some methods accompany a point estimate with a prediction interval, it is also reasonable to use a method predicting the interval in which the cost will fall. In this paper, we consider a method called Multinomial Logistic Regression using as dependent variable the predefined cost intervals and as predictor variables the attributes, similar to the ones characterizing completed projects of the available data set. The method builds a model, which classifies any new software project, according to estimated probabilities, in one of the predefined intervals. The proposed method was applied to a well-known data set and was validated with respect to its fitting and predictive accuracy.

1 Introduction

Software Cost Estimation (SCE) is a crucial activity in the initial project phases. Usually, by SCE we mean the prediction of the necessary project development effort, time or productivity. A lot of methods for SCE have been proposed in the literature, and they can be classified in three classes: *Expert Judgment*, which is the most commonly used approach and is based on the experience of a team of experts [1, 2], *Estimation by Analogy*, where the manager uses the notion of distance to find neighbors of the new project among completed ones and then produces a prediction based on their efforts [3], and finally *Algorithmic Models,* where the estimation involves the generation and application of a cost model, which is derived through statistical data analysis [1, 4, 5]. This last approach needs a historical cost database in order to estimate the model's parameters, e.g. coefficients of a regression model. In this paper we propose a method, which belongs to the class of algorithmic models.

In SCE, it is safer [6] to produce interval estimates, i.e. to predict that the actual cost of a project will fall within an interval consisting of a lower value (optimistic estimate), and a higher one (pessimistic estimate) along with a probability value. Some methods, for example the Ordinary Least Squares (OLS) regression models produce, along with the point estimate, a prediction interval with a certain confidence level.

In this paper, we explored an alternative approach, namely the definition of a certain number of cost intervals with certain bounds in advance and then the building of a model predicting in which of the intervals the cost of the new project will fall. Such a method is based on a set of historical data with completed projects, which will be used to predefine the cost intervals and build the model. Specifically, we investigated the use of a regression method, called Multinomial Logistic Regression (MLR), which uses as dependent the categorical variable with the predefined cost intervals and as predictor (independent) variables the numerical or categorical attributes characterizing completed projects of the available data set. The proposed method builds a model, which is able to classify any new software project, according to estimated probabilities, in one of the predefined intervals.

In Section 2, we review some methods related to the one presented here. In Section 3, we describe the data set used for building the model. In Section 4, we give the principles of MLR and we describe in detail the process of building and validating the model. Finally, in Section 5, we provide conclusions and directions of future research.

2 Related Work

Most of the methods proposed in the area of software cost estimation, produce point estimates. One of these methods is the Ordinary Least Square regression [7]. Maxwell [8] builds regression models based on a data set from a bank, while Kitchenham [9] proposes a method for analyzing unbalanced data sets, applied on the COCOMO81 dataset [1]. Her method is based on the successive application of Analysis of variance (ANOVA) or OLS, depending on the scale of the independent variable examined each time. The ultimate purpose is to build a model using the factors that mostly affect productivity. One drawback of the method is that it is very human-intensive, i.e. the analyst must work iteratively and build successive cost models. Another method that uses point estimates is Categorical regression or regression with Optimal Scaling [10], where a particular model was built based on the International Software Benchmarking Standards Group (ISBSG) [11] database.

Methods that propose interval estimates are less frequent in the literature. One method that proposes interval estimates based on expert judgement opinion is illustrated in [12]. In [13], a simulation tool for efficient analogy based cost estimation was suggested producing interval estimates. Another approach that produces interval estimates for project portfolios is reported in [6]. Also, in [14], predefined interval estimates have been used for a Bayesian Belief Network (BBN) in order to predict software productivity.

Comparative studies of software development cost modelling techniques can be found in [15, 16, 17]. In [15], the comparison of two cost modelling techniques based on multi-organizational and company-specific datasets is reported, while in [16] and [17] the comparison of ANOVA, OLS, Classification and Regression Trees (CART) and estimation by analogy based on multi-organizational databases is performed.

3 Description of the data

The database we used in our analysis is the COCOMO81 database. We selected COCOMO81 because it is a public domain database and various methods have been already applied on it [1]. The dataset we used is illustrated in Table 1 and is taken from [9].

Variable	Full Name	Levels
TYPE	Type of project	1-BUS 2-CTL 3-HMI 4-SCI 5-SUP 6-SYS
RELY	Reliability	1-Extra Low 2-Very Low 3-Low 4-Nominal 5-High 6-Very High
DATA	Data base size	1-Low 2-Nominal 3-High 4-Very High 5-Extra High
CPLX	Complexity	1-Very Low 2-Low 3-Nominal 4-High 5-Very High 6-Extra High 7-Super High
TIME	Execution time constraints	1-Nominal 2-High 3-Very High 4-Extra High 5-Super High
STOR	Main storage constraint	1-Nominal 2-High 3-Very High 4-Extra High 5-Super High
VIRT	Virtual machine volatility	1-Low 2-Nominal 3-High 4-Very High

TURN	Turnaround Time	1-Very High 2-High 3-Nominal 4-low 5-Very Low
CTYPE	Type of computer	1-MAX 2-MID 3-MIN 4-MIC
ACAP	Analyst Capability	1-SuperHigh 2-Very High 3-High 4-Nominal 5-Low 6-Very Low 7-Super Low
AEXP	Applications Experience	1-Very High 2-High 3-Nominal 4-Low 5-Very Low
PCAP	Programmer Capability	1-Super High 2-Very High 3-High 4-Nominal 5-Low 6-Very Low 7-Super Low
VEXP	Virtual Machine Experi-ence	1-High 2-Nominal 3-Low 4-Very Low
LEXP	Programming Language Experience	1-High 2-Nominal 3-Low 4-Very Low
CONT	Personnel Continuity	1-Low 2-Nominal 3-High
MODP	Modern Programming Practices	1-Extra High 2-Very High 3-High 4-Nominal 5-Low 6-Very Low
TOOL	Use of Software Tools	1-Extra High 2-Very High

		3-High 4-Nominal 5-Low 6-Very Low
SCHED	Required development schedule	2-Lax 3-Nominal 4-Compressed 5-Very Compressed
RVOL	Requirements volatility	1-Low 2-Nominal 3-High 4-Very High 5-Extra High 6-Super High
RMODE	Software development mode	Organic Semidetached Embedded
DUR	Duration of completion of each project	Numerical variable
YEAR	The year of completion of each project	Numerical variable
PRODUCT	Productivity of each project	Numerical variable

Table 1. Variable Definition in COCOMO81 dataset

We calculated productivity by dividing the adjusted delivered source instructions (ADJ DSI) by the effort in man months (MM). Duration is a ratio scale measure and is measured in months. Year is interval scale measure and is the year of delivery of each project.

4 Data Analysis and Generation of the Model

The purpose of the present study is to make use of the data set described in Section 3 in order (a) to predefine cost intervals for productivity and (b) to build a regression model with dependent the variable having as values the different categories of productivity and independent variables the various cost factors. As the scope of any method is to be utilized in a wide area of applications, we will present in detail the process starting from the basic principles of MLR and continue with the preparation of the data for the building of the model, the choice of the appropriate variables, the estimated model parameters and finally the validation of the model fitting and predictive accuracy. The statistical tool we used in our analysis is the SPSS package.

4.1 The Multinomial Logistic Regression

The Logistic Regression (LR) method is used to model the relationship between a dichotomous (binary) dependent variable and a set of k predictor variables $\{x_1, x_2, ..., x_k\}$, which are either categorical (factors) or numerical (covariates). As the binary dependent variable can be always interpreted as the occurrence or not of an event E, the logistic regression model is an expression of the form

$$\log\left(\frac{prob(E)}{1-prob(E)}\right) = b_0 + \sum_{i=1}^{k} b_i x_i \tag{1}$$

where the b_i's denote the unknown logistic regression coefficients (b_0 is the intercept) while $prob(E)$ denotes the probability that event E will occur. The quantity on the left side of equation (1) is called a *logit*. So, the simple LR model can be used for predicting the probability of an event occurrence.

The model can be generalized in the case where the dependent variable is polytomous, i.e. its values are more than two categories. In such a case, if we assume that the possible categories are q, we need to model $q-1$ logits,

$$\log\left(\frac{prob(category_j)}{prob(category_q)}\right) = b_0^{(j)} + \sum_{i=1}^{k} b_i^{(j)} x_i \ , \quad j = 1, ..., q-1 \, . \tag{2}$$

We can see in the expression above that one of the categories is used as reference and is called *baseline* category. Since in our applications the different categories are successive cost intervals, we consider the last interval as the baseline category for our models. After estimating the coefficients of the model (2) by the method of maximum likelihood, we can readily calculate the logits and therefore the probabilities of each one of the categories. The final prediction is the category with the maximum probability.

4.2 Defining intervals of the dependent variable

In our paper, we used as dependent variable the predefined productivity intervals. We have tried a number of divisions of the distribution of productivity (*product*) into intervals. Since the number of projects is relatively small (63) with respect to the number of variables, a large number of intervals cannot give us valid models. So, eventually we divided productivity into 4 intervals using the quartiles of its empirical distribution. The 4 intervals in which we have divided productivity can be seen in table 2:

First interval	[0, 92.5]
Second interval	(92.5, 190]
Third interval	(190, 350]
Fourth interval	(350, highest]

Table 2. The four predefined productivity intervals

4.3 Preparation of the independent variables

The COCOMO81 dataset contains 63 completed projects. We considered all of them in our analysis. Most of the predictor variables are categorical and in cases where categorical data had to be included in the model, we chose to recode the corresponding variables into two or more homogeneous groups for each factor. The same approach was taken in a similar study [10]. For each one of these categorical variables we performed one-way Analysis of Variance (ANOVA) in order to check the impact of every factor on the dependent variable. We used also Post-Hoc tests (Tukey, Tukey's-b, Bonferroni, LSD, Scheffe and Duncan) in order to identify the various categories that have to be concatenated in every factor. Next, we present these new categorical variables and the results of the ANOVA and Post-Hoc tests.

1) Type of project (*type*): Resulted in factor *type_2* which has two levels: 1=CTL+HMI and 2=SYS+BUS+SUP+SCI
2) Reliability (*rely*): Level "low" has only one value. Tests resulted in factor *rely_2* with two levels: 1=Extra low+Very low+low+ Nominal, 2=High+Very High
3) Data base size (*data*): Resulted in factor *data_2* with two levels: 1=Low+Nominal+ High, 2=Very High+Extra High
4) Complexity (*cplx*): One level (super high) has only one value. Tests resulted in factor *cplx_2* with 2 levels: 1=very low+low+nominal+high+very high, 2=Extra high+super high
5) Execution time constraints (*time*): Tests resulted in factor *time_2* with 2 levels: 1=Nominal, 2=High, Very High, Extra High, Super High
6) Main storage constraint (*stor*): One level (Very High) has only 1 value. Tests resulted in factor *stor_3* with 3 levels: 1=Nominal, 2=High +Very High +Extra High, 3=Super High
7) Virtual machine volatility (*virt*): Tests resulted in factor *virt_2* with 2 levels: 1=low, 2=Nominal+High+Very High
8) Turnaround Time (*turn*): One level (very low) has only one value. Although we tried concatenation obtaining 2 levels (*turn_2*), there is no evidence of significant difference.
9) Type of computer (*ctype*): There is no evidence that there is significant difference between any two levels of this factor, so we left it out.
10) Analyst Capability (*acap*): There are 3 levels (Very High, Low, Super Low) with only one value. Although we tried concatenation and obtained 2 levels (*acap_2*), there is no evidence of significant difference.
11) Applications Experience (*aexp*): Although we tried concatenation and obtained in 2 levels (*aexp_2*), there is no evidence of significant difference.
12) Programmer Capability (*pcap*): Tests resulted in factor pcap_2 with 2 levels: 1= Super High+Very High+High, 2=Nominal+Low+Very Low+Super Low.
13) Virtual Machine Experience (*vexp*): Tests resulted in factor *vexp_2* with 2 levels: 1=High+Nominal, 2=Low +Very Low
14) Programming Language Experience (*lexp*): Tests resulted in factor *lexp_3* with 3 levels: 1=High, 2=Nominal, 3=Low+Very Low
15) Personnel Continuity (*cont*): Tests resulted in factor *cont_2* with 2 levels: 1=Low+Nominal, 2=High

16) Modern Programming Practices (*modp*): Tests resulted in factor *modp_2* with 2 levels: 1=Extra High+Very High+High, 2=Nominal+Low+Very Low
17) Use of Software Tools (*tool*): Tests resulted in factor *tool_2* with 2 levels: 1=Extra High+Very High+High+Nominal, 2=Low+Very Low
18) Required development schedule (*sched*): Tests resulted in factor *sched_2* with 2 levels: 1=Lax+Nominal, 2=Compressed+Very Compressed
19) Requirements volatility (rvol): Tests resulted in factor *rvol_2* with 2 levels: 1=Low+Nominal, 2=High+Very High+Extra High+Super High
20) Software development mode (rmode): Tests resulted in factor *rmod_2* with 2 levels: 1=E, 2=ORG+SD

Two of the predictor variables are numerical, year and duration. We decided to use variable year as it is, but to use the logarithm of duration because we had better fitting and prediction results.

4.4 Generation of the Logistic model

We have tried a lot of combinations of the predictor variables before constructing a valid model. A model can be characterized as "valid" if it fulfills some predefined accuracy measures. These accuracy measures is (a) the *significance of the model* (we defined a valid model to have significance less than 0.05), (b) the *significance of each variable of the model* (every variable of the model should have significance less than 0.05), and (c) the *classification table* which compares the observed and the predicted groups (the highest the overall percentage of the classification table, the better the model is).

The final model for all 63 projects in our analysis has the following predictor variables: LNDUR, RMODE_2, PCAP_2, TOOL_2, CPLX_2 and STOR_3.The coefficients of the predictor variables in each predefined productivity interval can be seen in Table 3.

PI_1	b	PI_2	b	PI_3	b
LNDUR	13.397	LNDUR	4.756	LNDUR	-1.199
RMODE_2=1	1.088	RMODE_2=1	-2.793	RMODE_2=1	21.929
RMODE_2=2	-13.455	RMODE_2=2	-10.119	RMODE_2=2	21.492
PCAP_2=1	-1.149	PCAP_2=1	-3.009	PCAP_2=1	-3.859
PCAP_2=2	0	PCAP_2=2	0	PCAP_2=2	0
TOOL_2=1	-9.018	TOOL_2=1	-4.077	TOOL_2=1	2.358
TOOL_2=2	0	TOOL_2=2	0	TOOL_2=2	0
CPLX_2=1	-10.365	CPLX_2=1	-1.347	CPLX_2=1	-3.733
CPLX_2=2	0	CPLX_2=2	0	CPLX_2=2	0
STOR_3=1	-20.628	STOR_3=1	-3.175	STOR_3=1	-16.471
STOR_3=2	-22.635	STOR_3=2	1.690	STOR_3=2	-13.917
STOR_3=3	0	STOR_3=3	0	STOR_3=3	0

Table 3. Variables and Coefficients of the model for all 63 COCOMO81 projects (*b* is the regression coefficient for a specific level of an independent variable, PI_k is the k^{th} Productivity Interval)

Since the dependent variable has 4 categories, we know from the theory of our method that the fourth category is redundant. If we want to compute point estimates for our model, we have to represent each interval by a single representative value. In order to represent each interval with a single point, we used the mean point or the median point for each interval.

In order to validate the model with respect to its fitting accuracy we used the Mean Magnitude of Relative Error (MMRE) and PRED(20%) or PRED(25%). The relative error (RE) is ((actual effort-estimated effort) / actual effort) *100. The magnitude of relative error (MRE) is the absolute value of the relative error (MRE = |RE|). The mean magnitude of relative error (MMRE) is the average of all magnitudes of relative errors. Pred(20%) is the percentage of projects with an MRE of 20% or less. Respectively, Pred(25%) is the percentage of projects with an MRE of 25% or less. The classification of the model is 77.8%. The fitting accuracy of the model can be seen in Table 4.

	Using the mean point as point estimate	Using the median point as point estimate
MMRE	43.37%	41.77%
PRED(20%)	36%	41.3%
PRED(25%)	47.6%	46%

Table 4.Fitting Accuracy of the model for all projects

We can see that using the median point of its interval we have slightly improved MMRE and PRED(20%) but PRED(25%) is little worse.

4.5 Model Predictive Accuracy

In order to investigate the predictive accuracy of our model we followed Kitchenham's method [9]. Kitchenham developed a simple method to investigate the predictive accuracy of her model on the COCOMO81 database. Based on her method, we omitted a subset of projects (which she called a test dataset), developed a model with the remaining projects (which she called the learning data set), and finally assessed the predictive accuracy of the model on the test dataset. So, we created six different learning and test dataset pairs. We constructed each learning dataset by removing every sixth project starting from the first project the first time. Thus, learning dataset 1 was constructed by removing the projects 1,7,13,19,25,31,37,43,49,55 and 61, learning dataset 2 was constructed by removing the projects 2,8,14,20,26,32,38,44,50,56, 62 and so on for the next learning datasets. Since we used all 63 projects of the COCOMO81 database in order to build our model, the first, second and third learning datasets contained 52 projects and the remaining learning datasets contained 53 projects each. The predictor variables for the six learning datasets are presented in Table 5.

Our method builds a model, which classifies any new software project, according to estimated probabilities, in one of the predefined intervals. As a consequence, the method hitrate [12, 13] (i.e. the number of software projects that are classified into the correct predefined interval for a test dataset, divided by the total number of software projects that had to be classified for the same test dataset) for every test dataset is the

most important accuracy measure. It corresponds to the classification rate of the method and can be seen in Table 6 for the six learning datasets and Table 7 for the six test datasets. Notice that the overall Hitrate for the six test datasets is 74.6%, which we consider to be satisfactory (3 out of 4 projects are estimated correctly). For comparison purposes with the results in [9], we provide also tentative point estimate results. The overall MMRE, PRED(20%) and PRED(25%) for the six learning and test datasets are presented in Tables 8 and 9 respectively.

Variables	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5	Subset 6
LNDUR	√	√	√	√	√	√
RMODE_2	√	√	-	√	√	-
PCAP_2	-	-	√	√	√	√
TOOL_2	√	√	√	√	√	√
CPLX_2	√	-	√	√	-	√
STOR_3	√	√	√	-	-	√

Table 5. Predictor variables for the six learning datasets (a hyphen denotes that the variable is not present in the model)

	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5	Subset 6
Hitrate	78.8%	65.4%	67.3%	69.8%	66%	66%

Table 6. Classification for the six learning datasets

	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5	Subset 6
Hitrate	82%	64%	73%	70%	80%	80%

Table 7. Hitrate for the six test datasets

	Using the mean point as point estimate	Using the median point as point estimate
MMRE	54.12%	51.77%
PRED(20%)	32.4%	33.96%
PRED(25%)	42.86%	43.183%

Table 8. Overall MMRE, PRED(20%) and PRED(25%) for the six learning datasets.

	Using the mean point as point estimate	Using the median point as point estimate
MMRE	44.685%	44.64%
PRED(20%)	32.8%	34.695%
PRED(25%)	42.2%	38.028%

Table 9. Overall MMRE, PRED(20%) and PRED(25%) for the six test datasets.

In comparison, in [9] the accuracy measures in Table 10 were produced. The results are quite close with those obtained through MLR.

	The model with all the projects	The six learning datasets	The six test datasets
MMRE	36%	56%	42%
PRED(20%)	49%	30%	35%

Table 10. Fitting Accuracy of Kitchenham's model for all the 63 projects and overall MMRE and PRED(20%) for the six learning and test datasets.

5 Conclusions and Future Work

In this paper, we have proposed a novel method for Software Cost Estimation, namely Multinomial Logistic Regression, in order to produce interval estimates according to estimated probabilities. We have described the process starting from the basic theory of the method and continued with the definition of the intervals for the dependent variable, the preparation of the independent variables, the generation of the model and finally the validation of the model with respect to its fitting and predictive accuracy. We have applied MLR to a well-known public domain database and obtained various fitting accuracy measures. We concluded that the ability of the model to classify any future software project in one of the predefined intervals is high. In order to compute point estimates for our model we represented each interval with the mean or the median point of each interval. We found out that MLR estimation results are comparable to those of one of the most significant works in this field [9]. However, we claim that the MLR approach is somewhat easier to apply, and therefore may be more appealing to the average software manager. Kitchenham's method is a quite complicated method as the analysis procedure is extensive with repeatable ANOVA's and regression models. In MLR, the analysis is simple and the remarkable of the method is the implementation of only one regression model.

Future work includes the application of MLR to other cost datasets (e.g. [8]) and multi-organizational cost datasets (i.e. ISBSG [11]), and the comparison of the ability of various alternative methods to produce interval estimates.

References

1. Boehm B.W., *Software Engineering Economics*, Prentice-Hall, (1981)
2. H.C. Wohlin, "An Experimental Study of Individual Subjective Effort Estimation and Combinations of Estimates", *Proceedings International Conference on Software Engineering*, (1998), 332-339
3. M.J. Shepperd, C. Schofield, "Estimating Software Project Effort Using Analogies", *IEEE Transactions on Software Engineering,* 23(12):736-743 (1997)
4. B. Clark, S. Chulani, B. Boehm, "Calibrating the COCOMO II Post-Architecture Model", *Proceedings International Conference on Software Engineering*, (1998) 477-480.
5. S. Chulani, B. Boehm, and B. Steece, "Bayesian Analysis of Empirical Software Engineering Cost Models", *IEEE Transactions on Software Engineering*, 25(4):573-583, (1999)
6. I.Stamelos, L.Angelis, Managing Uncertainty in Project Portfol Estimation, *Information and Software Technology* 43(13):759-768, (2001)

7. N. Draper, H. Smith, *Applied Regression Analysis*, Wiley, (1981)
8. K. Maxwell, *Applied Statistics for Software Managers*, Prentice-Hall, PTR (2002)
9. B. Kitchenham, A procedure for analyzing unbalanced datasets, *IEEE Transactions on Software Engineering*, 24(4), (1998)
10. L. Angelis, I. Stamelos, M. Morisio, Building a Software Cost Estimation Model Based on Categorical Data.
11. ISBSG Data Disk, Rel.6, (1999)
12. M. Jorgensen, An Effort Prediction Interval Approach Based on the Empirical Distribution of Previous Estimation Accuracy, *Information and Software Technology* 45:123-126 (2003)
13. L. Angelis, I. Stamelos, "A Simulation Tool for Efficient Analogy Based Cost Estimation", *Empirical Software Engineering*, 5(1):35-68 (2000)
14. I. Stamelos, L. Angelis, P. Dimou, E. Sakellaris, On the Use of Bayesian Belief Networks for the Prediction of Software Productivity, *Information and Software Technology,* 45:51-60 (2003)
15. R. Jeffery, M. Ruhe and I. Wieczorek,"A Comparative Study of Two Software Development Cost Modelling Techniques Using Multi-organizational and Company-Specific Data", *Information and Software Technology*, 42:1009-1016 (2000)
16. R. Jeffery, M. Ruhe, I. Wieczorek, Using Public Domain Metrics to Estimate Software Development Effort, *Proceedings 7th International Software Metrics Symposium*, London UK, (2001) 16-27.
17. K. Maxwell, L. Briand, K. Emam, D. Surmann, I. Wieczorek, An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques, *Proceedings 22nd International Conference on Software Engineering (ICSE),* Limerick, (2000), 377-386.