A predictive analysis approach using linear regression to estimate software effort

Antogio G. L. Esteves[1], **Leonardo M. Medeiros**[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia de Alagoas (IFAL) – Campus Maceió – Maceió - AL – Brasil

toni.esteves@gmail.com, leonardomelomedeiros@gmail.com

Abstract. *Making decisions with a highly uncertain level is a critical problem in the area of software engineering. Predicting software quality requires high accurate tools and high-level experience. Otherwise, AI-based predictive models could be a useful tool with an accurate degree that helps on the prediction of software effort based on historical data from software development metrics. In this study, we built a software effort estimation model to predict this effort using a linear regression model. This statistical model was developed using a non-parametric linear regression algorithm based on the K-Nearest Neighbours (KNN). So, our results show the possibility of using AI methods to predict the software engineering effort prediction problem with an coefficient of determination of 76%.*

1. INTRODUCTION

Software development involves a number of correlated factors which affects development effort and productivity. But, sometimes these correlations are not so strong. So, an accurate software development estimation effort is a toward task[Fowler et al. 1999]. Considering the challenge of predict the software development time estimation within the confined timeframe and budget this problem gains economical and scientific relevancy. The accuracy has a vital role in software development, effort prediction estimation is one of the critical tasks required for developing software. Thus, in this research, we focus on analyzing the importance weights of attributes in the estimating of software cost and time and his correlation.

In this paper, we set out to answer two research questions related to the dataset:

1. Which the correlation of each metrics in the estimation of software effort ?
2. How accurate is the model of software effort ?

We have shown that some software metrics are more necessary than other attributes. The results of our empirical study reveal that a predictive model of software effort presented by both models could successfully predict more than 70% with less than 3% difference between them.

This study is splitted in five sections. On second section effort estimation and related works are described. Section tree describes the procedures applied and details about how statistical models were build. This section also describes reproducible search conditions. On section four results are detailed and explained. Finally on section five final considerations are made about this work, as well as future studies proposals.

2. EFFORT ESTIMATION

When measurements embrace structure system they become more meaningful indicators called metrics. The structure could be a simple algebraic formulation model. The structure behind metrics could be built on software engineer problems or business situations. Moving from measures to metrics is like moving from observation to understanding. Metrics are conceived by the user and designed to reveal chosen characteristics in a reliable meaningful manner. Then these metrics are mapped to ongoing measurements, to arrive at the best fit [Pandian 2003]

Another way to classify software metrics is in terms of the criteria used to determine them. In this respect, metrics are differentiated into objective metrics and subjective metrics. Objectives are obtained through well-defined rules and are the best way to enable consistent subsequent comparisons. The subjective ones depend on the subject who is carrying out the measurement and, therefore, discriminate the comparisons and the reproducibility of the measures. In order to enable software evaluation in a systematic, efficient, efficient and inexpensive way, the values obtained through the metrics should always be the same regardless of the moment, conditions or individual carrying out the measurements.

One of the fundamental issues in a software project is to know, before executing it, how much effort, in working hours, it will be necessary to bring it to term. This area called effort estimation counts on some techniques that have presented interesting results over the last few years[Wazlawick 2013].

One of reasons for failed estimates is an insufficient background of estimators in the area of software estimation. Unfortunately, human experts are not always as good at estimating as one could hope: estimates of cost and effort in software projects are often imprecise, with an average overrun of about 30% [Halkjelsvik and Jørgensen 2011]. Deliberate decisions regarding the particular estimation method and knowledgeable use require insight into the principles of effort estimation[Trendowicz and Jeffery 2014].

Most of the software estimation models that are available are based on some form of regression technique [Matson et al. 1994]. Regression models have a statistical foundation and are constructed by collecting data on completed projects and developing regression equations that characterise the relationships among the different variables [Fairley 1992].

The *Learning-oriented models* attempts to automate the estimation process by building computerised models that can learn from previous estimations experiences [Boehm et al. 2000]. These models do not rely on assumptions and are capable of learning incrementally as new data are provided over time [Lee-Post et al. 1998]. Learning-oriented models cover a wide area and include techniques such as artificial intelligence approaches, artificial neural networks, case-based reasoning [Mukhopadhyay and Kekre 1992]. Otherwise, cost estimation tools such as COCOMO, learning oriented and regression-based models; expert knowledge; and composite-Bayesian methods [Ruhe et al. 2003]

Naturally there wrong tendency associated with estimation tools primarily because “an estimate is a probabilistic assessment of a future condition” and accuracy can therefore rarely be expected in the estimation process [Stamelos and Angelis 2001].

2.1. RELATED WORKS

The research developed by Ayyıldız makes use of Desharnais dataset to finding the necessary attributes that affects the software effort estimation and analyzing the necessity of these attributes [Erçelebi Ayyıldız and Can Terzi 2017]. In this study Ayyıldız analysis Pearson's Correlation between Desharnais dataset metrics and software effort as well as regression analysis applicability. To show the differences between the actual and estimated values of the dependent variable scatterplots and residual plots are shown. Finally to evaluate prediction performance Ayyıldız calculates Magnitude of Relative Error (MRE), Mean Magnitude of Relative Error (MMRE), Median Magnitude of Relative Error (MdMRE), MSE (Mean Square Error) and Prediction Quality (pred(e)).

Kocaguneli proposed an effort estimation model to the software development division of the Turkish subsidiary of a multinational bank (Bank) [Kocaguneli et al. 2010]. In this study Kocaguneli built a learning-based effort estimation model and validates the performance using Bank data of completed projects as well as public datasets gathered from various organizations. Kocaguneli also perform various AI techniques for metrics selection, clustering and estimation.

One of the most referenced article was written by Kitchenham [Kitchenham et al. 2002]. In this study, Kitchenham presents a dataset that enables to investigate the actual accuracy of industrial estimates and to compare those estimates with estimates produced from various function point estimation models. However, the study make it clear that any models derived from the current data set are context-specific. They are based on data from a single company with a specific estimation process (that depends on the particular skills of the estimation staff), and a specific client set. The full corporate data set, implemented in Microsoft Access with additional tools in Word and Excel, comprises more than 145 projects. However, the analysis to those projects for which actual effort, estimated effort and function point counts were all available. The projects were undertaken between 1994 and 1998. The conclusions drawed from this study are somewhat limited, because the projects studied were undertaken by a single company. Thus, it was not expected any of the models presented in this paper to generalize automatically to other maintenance or development situations.

3. MATERIALS AND METHODS

To perform this study we used Desharnais dataset from the PROMISE software engineer repository. This dataset consists of 81 software projects from a Canadian software house collected by J. M. Desharnais [1]. Firstly we analyze the correlation between each attributes of Desharnais dataset and effort attribute. We apply linear regression technique to investigate relation between these attributes. After that we apply a regression based on k-nearest neighbors regressor. Lastly we evaluate our prediction performance comparing the squared error value of both algorithms.

3.1. DATASET

The Desharnais dataset [Desharnais 1989] has the record of a total of 81 projects developed by a Canadian software house in 1989. Each project has twelve features which are described in Table I.

[1]The promise repository of empirical software engineering data

Feature	Description
Project	Project ID which starts by 1 and ends by 81
TeamExp	Team experience measured in years
Manager Exp	Manager experience measured in years
YearEnd	Year the project ended
Length	Duration of the project in months
Effort	ActualEffort is measured in person-hours
Transactions	Transactions is a count of basic logical transactions in the system
Entities	Entities is the number of entities in the systems data model
PointsAdj	Size of the project measured in unadjusted PointsAdj function points.
PointsNonAdjust	Size of the project measured in adjusted function points.
Language	Type of language used in the project expressed as 1, 2 or 3.

Table 1: Features definition to Desharnais dataset

This dataset has become very popular as many developers use it in addition to other datasets to train and evaluate software estimation models, in all its includes nine numerical attributes. The eight independent attribute of this data set, namely ”TeamExp”, ”ManagerExp”, ”YearEnd”, ”Length”, ”Transactions”, ”Entities”, ”PointsAdj”, and ”PointsNonAjust” are all considered for constructing the models. The dependent attribute ”Effort” is measured in person hours.

3.2. REPRODUCIBLE RESEARCH

A research work is reproducible when all research artifacts such as as text, data, figure and code are available for independent researchers reproduce the results. Data shared through reproducible research also provides improved continuity for related research and represents an act of better stewardship towards public resources (research funding) as well as expanding access to the generation of knowledge [Donoho 2010]. Reproducibility of experiments is essential during the validation process so all steps in the analytical workflow are coded in Python and available for download from a Github repository – https://github.com/toniesteves/assessment-software-effort. A full description of the steps necessary to download and execute code to reproduce this analysis in full are available in the ReadMe.md file on the home page to the repository.

Raw data for the example analysis come from PROMISE Software Engineering Repository. The permanent link to the data can be found here – http://promise.site.uottawa.ca/SERepository.

Additionally, a jupyter notebook has been developed to allow readers and interested researchers to assess the validity of the results under different data and model assumptions. In this manner, the results of this work are available not only to seasoned researchers with a moderate level of knowledge using the Python computing language but to any potentially interest reader, be it a graduate student, politi-

cian or home owner in the area. The interactive application can be found at https://github.com/toniesteves/assessment-software-effort.

3.3. FEATURE SELECTION

To address Desharnais dataset the correlations between attributes and software effort are analyzed. The correlation between two variables is a measure of how well the variables are related. As previous mentioned on [Kocaguneli et al. 2010] and [Erçelebi Ayyıldız and Can Terzi 2017], we can use a set of features or metrics to perform correlation. A feature is an individual measurable property of the process being observed.

The most common measure of correlation in statistics is the Pearson Correlation (or the Pearson Product Moment Correlation - PPMC) which shows the linear Relationship between two variables [REFERENCIA]. Pearson correlation coefficient (PCC) is a statistical metric that measures the strength and direction of a linear relationship between two random variables [Rodgers and Nicewander 1988]. It has been applied to various indices in statistics, such as data analysis and classification [Ye 2014]. Pearson correlation coefficient analysis produces a result between -1 and 1. A result of -1 means that there is a perfect negative correlation between the two values at all, while a result of 1 means that there is a perfect positive correlation between the two variables. Results between 0.5 and 1.0 indicate high correlation [Mehedi Hassan Onik et al. 2018]. The Pearson correlation coefficients between attributes and software efforts are given in Figure 1 for Desharnais dataset.

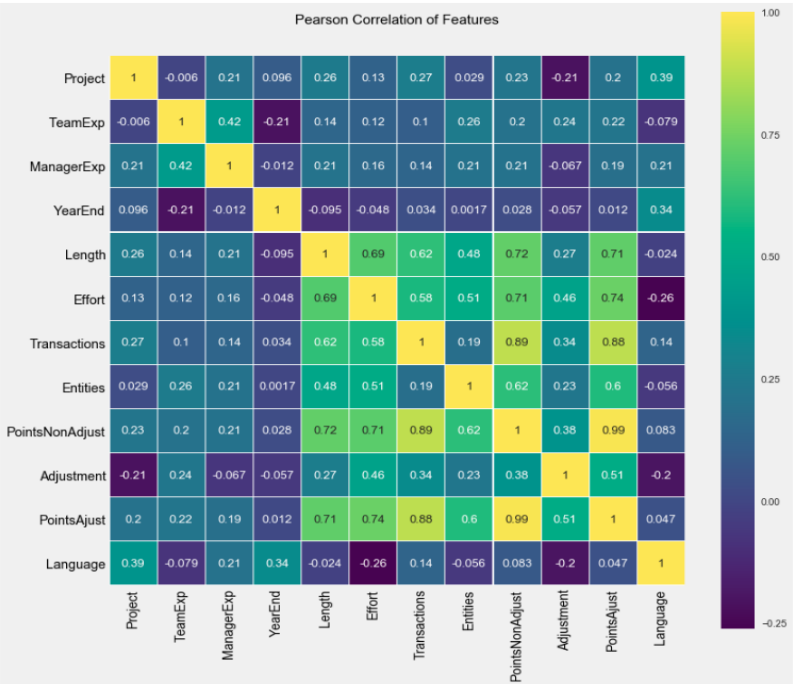


Figure 1. Pearsons Correlation to Desharnais dataset

3.4. MODELS CONSTRUCTION

In this study the following algorithms were used: *Linear Regression* and *K-Nearest Neighbors Regression*. The training of the regressors models were performed on 67% of the

instances of the base, chosen at random, the tests were performed on the remaining instances. The training of the models was carried out in Python language, along with the following libraries: Numpy, Pandas, Scikit-learn, Seaborn and Matplotlib. During the training it was necessary to estimate the values of the random state parameter, since they are not previously known.

The regression analysis aims to verify the existence of a functional relationship between a variable with one or more variables, obtaining an equation that explains the variation of the dependent variable *Y*, by the variation of the levels of the independent variables. The training of the *Linear Regression* model consists of generating a regression for the target variable *Y*.

Likewise the *K-Nearest Neighbor Regression* is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure and it's been used in a statistical estimation and pattern recognition as non-parametric technique classifying correctly unknown cases calculating euclidean distance between data points.

In fact our choice by *K-Nearest Neighbor Regression* was motivated by the absence of a detailed explanation about how effort attribute value is calculated on Desharnais dataset. In the *K-Nearest Neighbor Regression* we choose to specify only 3 neighbors for k-neighbors queries and uniform weights, that means all points in each neighborhood are weighted equally.

4. RESULTS

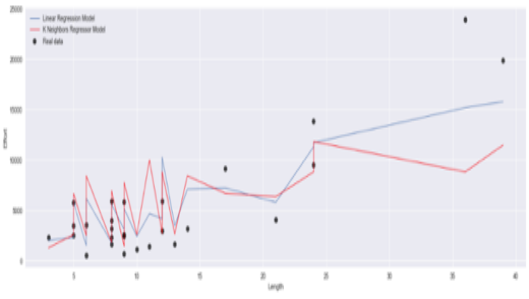
This section presents the results obtained through the methods applied in dataset. Both models generated from the training with data from the previous section will be applied to the remaining 33% of the base, previously isolated, and their performances will be evaluated in order to demonstrate how accurate the linear regression model can predict software effort estimation.

With the trained models, we run the tests on the test instances. Thus, we calculate respective R^2 values. Table 3 shows the coefficients reached.

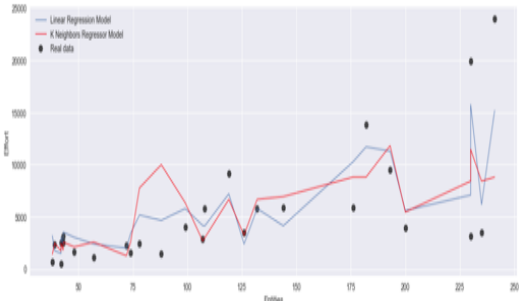
Algorithm	**R^2 Score**
Linear Model Regression	0.7680074954440712
K-Nearest Neighbor Regressor	0.7379861869550943

Table 2: Algorithms model results

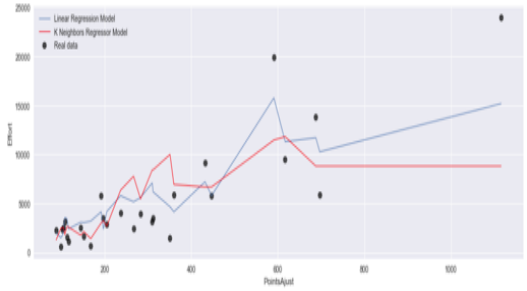
In Figure 2 plots of the best correlated variables applied to both models are displayed.



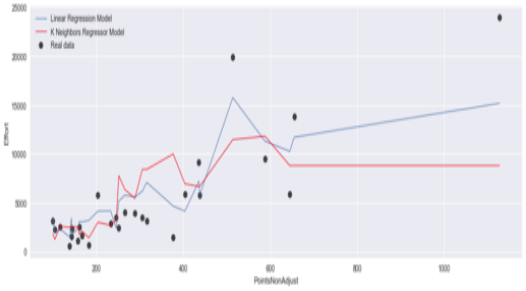
(a) *Knn* x *LR* on Length feature



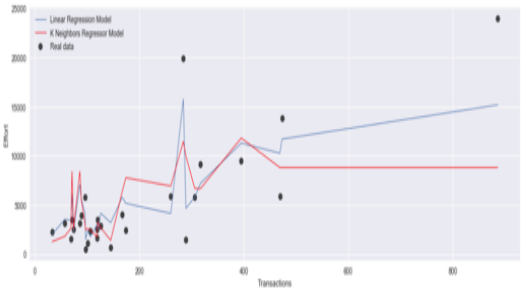
(b) *Knn* x *LR* on Entities feature



(c) *Knn* x *LR* on PointsAdjust feature



(d) *Knn* x *LR* on PointsNonAdjust feature



(e) *Knn* x *LR* on Transactions feature

Figure 2. Comparative R^2 scores from K-neighbors Regression and Linear Regression

Each feature from more correlated features is illustrated in Figure 2. The figure shows the linear model (blue line) prediction is fairly close to Knn model effort prediction (red line), predicting the numerical target based on a similarity measure.

5. CONCLUSION AND FUTURE WORKS

This work presents the feasibility of using regression and machine learning techniques to predict software effort estimation, providing estimates based on data from existing projects. The contributions of this work are based on the use of two output models that seek to take advantage of the relationships between the target values of the project. These methods, together with linear regression and K-neighbors regression algorithms , resulted in predictive models capable of estimating values for the software effort estimation operations.

The proposed methodology consisted in assembling the plans through the two models presented in this work. The results obtained with each method were compared in

order to evaluate the feasibility of using a linear regression model to estimate the software effort. In addition to these models, we also present the correlation between the variables and how each influence the target variable.

Our results obtained obtained a R^2 value of more than 70% and a difference of only 3% among them, indicating the feasibility of using linear regressors to predict software effort. However, to have a more concise and fair result we need to reproduce the same approach with other available algorithms.

Finally, we propose as future works the use of a larger project base in order to diversify and give greater reliability to the method. Another point to consider is apply these model comparing with function points.

References

Boehm, B., Abts, C., and Chulani, S. (2000). Software development cost estimation approaches – a survey. *Ann. Softw. Eng.*, 10(1-4):177–205.

Desharnais, J.-M. (1989). Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction.

Donoho, D. L. (2010). An invitation to reproducible computational research. *Biostatistics*, 11(3):385–388.

Erçelebi Ayyıldız, T. and Can Terzi, H. (2017). Case study on software effort estimation. 7:103–107.

Fairley, R. E. (1992). Recent advances in software estimation techniques. In *Proceedings of the 14th International Conference on Software Engineering*, ICSE '92, pages 382–391, New York, NY, USA. ACM.

Fowler, M., Beck, K., Brant, J., Opdyke, W., and Roberts, D. (1999). *Refactoring: improving the design of existing code*. Addison-Wesley Professional.

Halkjelsvik, T. and Jørgensen, M. (2011). From origami to software development: A review of studies on judgment-based predictions of performance time. 138:238–71.

Kitchenham, B., Pfleeger, S. L., McColl, B., and Eagan, S. (2002). An empirical study of maintenance and development estimation accuracy. *Journal of Systems and Software*, 64(1):57 – 77.

Kocaguneli, E., Tosun, A., and Bener, A. (2010). Ai-based models for software effort estimation. pages 323–326.

Lee-Post, A., Cheng, C. H., and Balakrishnan, J. (1998). Software development cost estimation: Integrating neural network with cluster analysis. 34:1–9.

Matson, J. E., Barrett, B. E., and Mellichamp, J. M. (1994). Software development cost estimation using function points. *IEEE Trans. Softw. Eng.*, 20(4):275–287.

Mehedi Hassan Onik, M., Ahmmed Nobin, S., Ferdous Ashrafi, A., and Mohmud Chowdhury, T. (2018). Prediction of a Gene Regulatory Network from Gene Expression Profiles With Linear Regression and Pearson Correlation Coefficient. *ArXiv e-prints*.

Mukhopadhyay, T. and Kekre, S. (1992). Software effort models for early estimation of process control applications. *IEEE Transactions on Software Engineering*, 18(10):915–924.

Pandian, C. R. (2003). Software metrics: A guide to planning, analysis, and application.

Rodgers, J. and Nicewander, W. (1988). Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66.

Ruhe, M., Jeffery, R., and Wieczorek, I. (2003). Cost estimation for web applications. pages 285–294.

Stamelos, I. and Angelis, L. (2001). Managing uncertainty in project portfolio cost estimation. 43:759–768.

Trendowicz, A. and Jeffery, R. (2014). *Software Project Effort Estimation: Foundations and Best Practice Guidelines for Success*. Springer Publishing Company, Incorporated.

Wazlawick, R. (2013). *ENGENHARIA DE SOFTWARE: CONCEITOS E PRÁTICAS*. Elsevier Editora Ltda.

Ye, J. (2014). Correlation coefficient of dual hesitant fuzzy sets and its application to multiple attribute decision making. 38:659–666.