2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)

Features-Level Software Effort Estimation Using Machine Learning Algorithms

Mustafa Hammad
Department of Computer Science
Mutah University,
Al Karak, Jordan
University of Bahrain
Sakheer, Bahrain
hammad@mutah.edu.jo

Abdulla Alqaddoumi
Department of Computer Science
University of Bahrain
Sakheer, Bahrain
aqaddumi@uob.edu.bh

***Abstract*— Software effort estimation is a paramount mission in the software development process, which covered by project managers and software engineers. In the early stages, software system features are the only available measures. Therefore, cost estimation is a mission that comes under the planning stage of software venture management. In this paper, various machine learning algorithms are used to build software effort estimation models from software features. Artificial Neural Network (ANN), Support Vector Machines (SVM), K-star, and Linear Regression machine learning algorithms are evaluated on a public dataset with actual software efforts. Results showed that machine learning approach can be dependable on predicting the future effort of a software system.**

Keywords—machine learning, software development, effort estimation, cost estimation.

I. INTRODUCTION

Due to the rapid development of software in our time, the software development process has become a basic activity of modern society. However, software effort estimation is a critical part of the software development lifecycle. Software development task aims to deliver the product within the budget and schedule. Therefore, early software effort estimation is an essential part of the planning process for any software project.

Software effort estimation includes predicting the effort needed to build a software system. It is expressed in terms of working hours or hours required to build the software. The estimation of software effort can be considered as a broad area of many software activities, which include predicting software testing, maintenance, requirements engineering, etc.

Different software development lifecycle models require a different amount of effort at each stage to deliver software. Software effort estimation is considered also as one of the most important challenges faced in software engineering, which many developers and project managers suffer with and thus affect the project's cost. As a result, efficiency is determined by the management, which uses effort estimation to evaluate projects and manage the development processes more clearly [1]. The accuracy of software effort estimation is an important challenge that may cause damage to software companies.

Therefore, several models have been proposed by many researchers to estimate the effort. Many studies have been conducted to estimate the software effort in the early stages to clarify its importance [2][3]. Software organizations need to know how they can develop their projects, as well as, the amount of effort to develop.

In this paper, four machine learning algorithms are applied to estimate the software effort based on project features. The main goal of this study is to evaluate software effort estimation models that are built based on machine learning approaches.

The remainder of this paper is structured as follows; in the next section, literature concerning software effort estimation models is presented. Section 3 presents the proposed prediction models and the used dataset. Experimental results are presented in Section 4 followed by paper's conclusion in section 5.

II. RELATED WORK

Reliable and accurate software effort estimation is a success key that is needed at each phase of the software life cycle. It is an essential factor in determining the feasibility of software projects. Many methods have been suggested to predict software effort estimation. A number of researches were presented in the field of the software effort estimation using various models. For instance, Dorado [4] used the genetic software method to explore a cost estimation function. The results were benchmarked with earlier data sets. In the same year, Burgess and Lesley [5] attempted to use genetic programming and compared results with previous methods using known standard data. Lesley and Shepperd [6][7] used the method of genetic software to enhance software effort estimation accuracy based on general datasets.

Machine learning algorithms have been used in different works to estimate the software effort. For example, the work in [8] proposed a study on the comparison of a set of machine learning techniques in effort estimation for software development process. Albert [9] calculated the effort difference for software projects based on function points and neural networks with regression and case-based reasoning approaches. Other research studies [10] [11] [12] presented studies about using neural network methods and found that it was the least mature method compared to other methods, and eventually, he concluded that no method was the best at all the time. Our work differs from the work of others in the estimation of software efforts is done in the early stage, which is only concerned with the project features.

III. METHODS AND DATASET

This paper presents various software effort estimation models using four machine learning algorithms. All models

978-1-5386-9207-3/18/$31.00 ©2018 IEEE

are evaluated and compared using a public dataset. The following subsections detail the used machine learning algorithms and dataset.

A. Dataset

Collecting public software effort data is not an easy task because of the lack of software cost data. This is because most of the effort data are kept private due to the software companies' aspect. In this paper, we used a public dataset called Usp05-tf, which is available online thru promise repository, which is a repository for empirical software engineering data (http://tunedit.org/repo/PROMISE/EffortPrediction). This dataset contains data about 76 university students' software projects. Each project consists of 14 features. Table 1 presents a summary of these features.

TABLE I. PROJECT FEATURES IN THE USP05-TF DATASET

Project Features	Description	Value
IntComplx	The complexity of the internal project calculations	1 to 5, where 5 indicate the highest complexity
DataFile	The number of accessed files	Positive integer
DataOut	The number of output data items	Positive integer
DataEn	The number of entry data items	Positive integer
UFP	Unadjusted Function Point Count	Positive integer
Lang	The used programming language	C++, Java, HTML, etc.
Tools	The used platforms and tools	VJ++, Delphi, Junit, etc.
ToolExpr	The experience level of the developer team	Range of number of months, e.g. [3, 7]
AppExpr	The applications experience level	1 to 5, where 5 indicate the highest experience
TeamSize	The size of the developer team	Range of min-max number of developers, e.g. [3, 6]
DBMS	The used database system	SQLServer, Oracle, MySQL, etc.
Method	The used implementation methodology	OO, JAD, SD, etc.
AppType	The used system architecture	B/S, C/S, Centered, etc.
Effort	The actual effort (in hours) expended on implementation tasks by all participating developers	Positive float

As shown in Table 1, not all features are numeric. This is very important to validate the ability of learning in the machine learning approach. The historical data is used as input to learn from previous experiences to predict the effort of new software systems.

B. Prediction Approach

In this paper, four machine learning algorithms are used to build four prediction models. These algorithms are:

1) Artificial Neural Network (ANN)

ANN is one of the most commonly used learning-oriented approaches. Multilayer perception is one of the most used supervised model function, which can be seen as a multi-layers network of neurons, connected by a feed-forward mode. In the first layer, neurons act as input points and represent input variables. Other layers are responsible of processing and output the data. Therefore, a neural network can be seen as a complex computation function that passes data through the network to the output layer to expand the solution [13].

2) Support Vector Machines (SVM)

SVM depends on the declaration of a decision plan, which defines decision margins. A decision plane is used to separate between a set of entities with different classes. SVM acts as a supervised classifier model that implements the classification task by building hyper-plane in a multidimensional space. SVM is widely used as a predictive machine-learning model for different datasets. [14]

3) K-Star

The K-star algorithm uses similarity measurements to classify the data based on the classes' likelihood. It acts as instance classifier based on an entropy-based distance function to calculate the distance between instances. [15]

4) Linear Regression

This Algorithm is used to express the data and find the correlation between the dependent variable and one or more independent nominal, comma or level variables.

In order to compare the four prediction models, the predictive performance is evaluated based on Mean Absolute Error (MAE). MAE is calculated as:

$$MAE = \sum_{i=1}^{n} |Ei - \text{Ei}| / n$$

where n is the number of test projects, E_i is the actual effort and $\acute{E}_i$ is the estimated effort. MAE is considered a good choice for measuring symmetric unbiased under or overestimates [16] [17]. A higher (better) predictive performance can be achieved with lower MAE.

IV. EXPERIMENTAL RESULTS

The evaluation process is implemented using the Weka tool. (https://www.cs.waikato.ac.nz/ml/weka/) Weka is a framework of machine learning algorithms written in Java. The evaluation procedure is set to be through 10-fold cross-validation. The cross-fold validation technique can measure the impact of dataset in predicting the software effort. The 10-fold cross-validation technique has ten different randomizations of the dataset. The cross-validation procedure has been used efficiently with machine learning algorithms for different datasets. [18][19][20]

Table 2 shows the MAE values for the four prediction models. As shown in Table 2, SVM has the best prediction accuracy with the lowest MAE value, which is about 2.6. Moreover, results show that the linear regression has the highest MAE value. All other algorithms have close MAE value with less than 3. Therefore, we consider linear regression result as an odd performance. We believe that this is caused by fact that the distribution of the effort values is hard to be expressed as a linear equation.

TABLE II. MAE VALUES FOR THE FOUR PREDICTION MODELS

Prediction Model	MAE Value
ANN	2.7826
SVM	2.5958
K-star	2.8312
Regression	5.9965

The second experiment is set to evaluate the ability of the used machine learning algorithms to learn from the historical data. In this experiment, we did not use the 10-fold cross-validation scheme. Instead, the dataset is divided into two sets, one for training and the other one for testing. This evolution is repeated by using 90%, 50%, and 10% of the dataset as training data. The remaining data are used as testing sets in the three evaluation processes. Figure 1 shows the collected results.

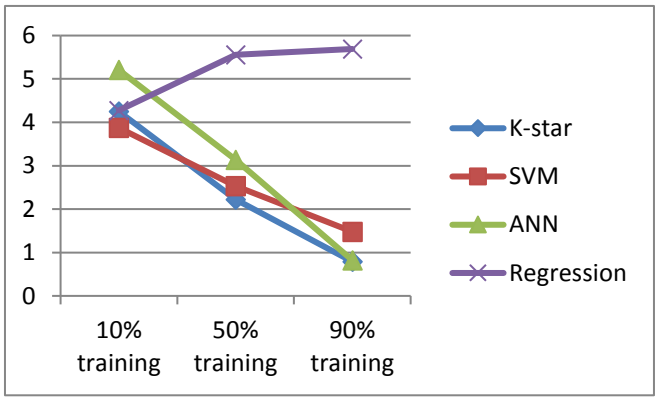

Fig. 1. MAE values for the four prediction models in the three evaluations.

As shown in Figure 1, the linear regression algorithm, also, fails to learn properly. All other algorithms learned perfectly as the size of the training set increased.

V. CONCLUSION

In this paper, four machine learning algorithms are used to build software effort estimation models. These algorithms are Artificial Neural Networks (ANN), Support Vector Machines (SVM), k-star, and linear regression. The estimation is set to predict the actual effort from the software features at early stages. A public dataset has been used to evaluate the prediction models. Results showed that machine learning approach can be applied to predict the software effort with low MAE value. The lowest MAE value was 2.6 for the SVM model. In the future work, we will evaluate other machine learning algorithms and use more datasets. In addition, applying filters in the software features data may increase the prediction accuracy due. This can be done by understanding the sensitivity of the prediction model for each dataset.

REFERENCES

[1] Ziauddin, Tipu S. K. and S. Zia, (2012), "An Effort Estimation Model for Agile Software Development," Advances in Computer Science and Its Applications, Vol. 2

[2] Krupka, E., Tishby, N., "Generalization from Observed to Unoberserved Features by Clustering", Journal of Machine Learning Research, Vol. 83, pp. 339–370 (2008).

[3] M. Jørgensen, M. Shepperd, "A systematic review of software development effort estimation studies", IEEE Transactions on Software Engineering.

[4] Dorado J.J., "on the problem of the software cost function", Information and Software Technology, 2001 Elsevier Science B.V.

[5] Burgess, C., Lesley, M., "Can Genetic Programming Improve Software Effort Estimation: A Comparative Evaluation". Inform. and Soft. Technology, Vol. 43 No.14, pp: 863–873.

[6] Lesley, M., Shepperd, M., "Using genetic programming to improve software effort estimation based on general data sets". In Procs. Of Genetic and Evolutionary Computation Conference, 2003, pp. 2477–2487.

[7] Wittig, G., Finnie, G., 1997. "Estimating software development abort with connectionist models". Inform. Software Technol.

[8] Ohsugi, N., Tsunoda, M., Monden, A. and Matsumoto, K. (2004) , "Effort Estimation Based on Collaborative Filtering", In the 5th International Conference on Product Focused Software Process Improvement (PROFES2004), pp. 274-286.

[9] Albert and J.E. Gaffney, "Software Function Source Lines of Code and Development Effort Prediction: A Software Science Validation"

[10] Saleem Basha, Dhavachelvan "Analysis of Empirical Software Effort Estimation Models" (IJCSIS) International Journal of Computer Science and Information Security, Vol. 7, No. 3, 2010.

[11] Huang, S., Chiu, N. (2006) "Optimization of Analogy Weights by Genetic Algorithm for Software Effort Estimation", Journal of Systems and Software, Vol. 48 No.11, pp: 1034-1045.

[12] Mendes, E., Mosley, "Bayesian Network Models for Web Effort Prediction: A Comparative Study". IEEE Trans. Software Eng., 34

[13] S. Haykin, Neural Networks: a Comprehensive Foundation. Prentice Hall, New Jersey, USA, 1999.

[14] A. J. Smola, B. Scholkopf, "A tutorial on support vector regression". *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[15] Painuli, Sanidhya, M. Elangovan, and V. Sugumaran. "Tool condition monitoring using K-star algorithm." *Expert Systems with Applications* 41.6 (2014): 2638-2643.

[16] Lokan, C., Mendes, E.: Investigating the use of chronological split for software effort estimation. IET Softw. **Vol. 3** No. 5, pp. 422–434 (2009)

[17] Shepperd, M., McDonell, S., "Evaluating prediction systems in software project estimation", IST vol. **54 no.** 8, pp. 820–827 (2012)

[18] Y. Bengio and Y. Grandvalet, "No unbiased estimator of the variance of k-fold cross-validation," The Journal of Machine Learning Research, vol. 5, pp. 1089-1105, 2004.

[19] M. Stone, "Cross-Validatory Choice and Assessment of Statistical Predictions," Journal of the Royal Statistical Society,Ser B, vol. 36, no. 2, pp. 111-147, 1974.

[20] P. Cohen and D. Jensen, "Overfitting Explained", Preliminary Papers of the Sixth International Workshop on Artificial Intelligence and Statistics, 1997, pp. 115-122.