Machine Learning Approaches for Software Effort Estimation Using Public Project Datasets

Ritu
Department of Computer Science & Engineering
Guru Nanak Dev Engineering College, Ludhiana, Punjab, India
ratheeritu@yahoo.in

Pankaj Bhambri
Department of Information & Technology
Guru Nanak Dev Engineering College, Ludhiana, Punjab, India
pkbhambri@gmail.com

.

***Abstract*—Estimating software effort is a critical task that project managers and software engineers handle during the software development process. The only available measures in the early stages are those of the software system. As a result, estimating costs is a task that falls under the software venture management planning stage. In this paper, software effort estimation models are constructed from software features using a variety of machine learning algorithms. Using real software efforts, machine learning algorithms such as Artificial Neural Networks (ANN), Support Vector Machines (SVM), K-star, and Linear Regression are assessed on a publicly available dataset. The outcomes demonstrated that a machine learning approach can be relied upon to forecast a software system's future effort.**

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

I. INTRODUCTION

Due to the fast evolution of software in our current era, the software development process has become a fundamental activity of modern society. However, software effort estimation is an integral part of the software development life cycle. Software development task is to provide the product within the budget and time. Thus, initial software effort estimation is an important part of the planning phase of any software development project.

Software effort estimation involves estimating the effort required to develop a software system. It is measured in terms of working hours or hours required to develop the software. The estimation of software effort can be viewed as a wide area of various software activities, which involve estimating software testing, maintenance, requirements engineering, and so on.

Various software development life cycle models involve a different amount of effort in each phase to develop the software. Software effort estimation is also viewed as one of the most significant challenges involved in software engineering, which many software developers and managers are suffering from, thus affecting the cost of the project. Thus, efficiency is measured by the management, which uses effort estimation to assess projects and manage the development process more clearly [1].

Thus, a lot of researchers proposed several models to estimate the effort. Several studies have been carried out for the estimation of software effort in the early stages to make its importance clear to them [2][3]. Software organizations must be aware of how they can develop their projects and the effort required to develop them.

In this paper, four different algorithms of machine learning will be implemented to determine the software effort using the features of the project. In general, the major objective of the study is to assess software effort estimation models that can be developed using the machine learning approach.

The rest of this paper is organized as follows: the next section identifies the software effort estimation models through the literature. Section 3 discusses the proposed prediction models, and the used dataset. Experimental results are described in Section 4. Finally, the conclusion of this paper is introduced in Section 5.

II. RELATED WORK

Reliable and precise estimation of software effort is one of the key factors for software development success, which is required at all stages of the software development cycle. It is one of the important factors for judging the feasibility of software development. A number of methods have been proposed to accurately estimate software effort estimation. A number of research works have been proposed in the area of software estimation method. For example, Dorado [4] has used the genetic software method to investigate the cost estimation function. The result was compared with the earlier results with respect to the datasets. Burgess and Lesley proposed genetic programming to improve the results of software effort estimation with respect to the earlier methods, and results were compared with the earlier methods using standard known data [5]. Lesley and Shepperd proposed the method of genetic software for improving the accuracy of software estimation based on general datasets [6,7].

Some other machine learning algorithms have been used to estimate the efforts of software development. For example, a study regarding the investigation of the comparative analysis of various machine learning methods and their application to software development process effort estimation has been proposed in [8]. Albert proposed a neural network for calculating the difference of the efforts of software projects based on function points, regression, and case-based reasoning. Some other research studies [10], [11], [12] have been done regarding the application of neural network methods, where it has been indicated that this method was less mature compared with other methods, and eventually, he concluded that no method was the best at all times. Our work is different from those of other authors, where it has been indicated that the estimation of efforts of software development is performed at the early stage of software development, which is concerned with the features of the project.

III. METHODS AND DATASET

The paper introduces various software effort estimation models using four machine learning methods. The models were evaluated and compared using a data set that is available to the public. The following sections will discuss various machine learning algorithms and the data used in the paper. In recent years, machine learning and

artificial intelligence have become essential terms used to describe the latest development and progress in the field of computer science and technology. To study and explore these concepts.

A. Dataset

It has not been an easy task to collect public software effort data due to the absence of software cost data. This is because most of the effort data is held as private data as indicated by the software companies' aspect. In this paper, we have relied on a public software effort dataset labeled "Usp05-tf," which is available online via the promise repository, which is the "repository for empirical software engineering data" (http://tunedit.org/repo/PROMISE/EffortPrediction). This dataset offers details on software projects completed by 76 university students. There are 14 features in each software project. Table 1 offers an overview of the 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

Four machine learning algorithms will be used in this paper to build four prediction models. These algorithms are:

1) Artificial Neural Network (ANN)

One of the most popular learning-oriented methods is the ANN. Multilayer perception is one of the most popular supervised model functions and can also be considered as a multilayers network of neurons connected in a feed-forward manner. In the first layer, the neurons act as input points and correspond to the input variables. The remaining layers are responsible for processing and delivering the information. Thus, a neural network can be described as a complex computation function that transmits the information through the network to the output layer to enlarge the solution. [13]

2) Support Vector Machines (SVM)

SVM is dependent on the definition of a decision plan, which defines the decision margins. A decision plane is applied to distinguish between a set of entities with different classes. SVM is a supervised classifier model that applies the definition of the classification task through the construction of a hyper-plane in multidimensional space. SVM can be applied as a predictive machine learning model for various datasets. [14]

3) K-Star

The K-star algorithm employs similarity measurements to classify the data based on the classes' likelihood. The algorithm serves as an instance classifier, using an entropy-based distance function to measure 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 variables. independent nominal, comma or level variables.
In order to compare these four prediction models, the The performance of the predictor is usually assessed according to Mean Absolute Error. MAE is calculated as:

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

where n is the number of test projects, Ei is the actual effort and Ei is the estimated effort. MAE is regarded as a good option for estimating symmetric unbiased under or overestimates [16] [17]. Greater (better) performance can be attained with smaller MAE.
IV. EXPERIMENTAL RESULTS This evaluation procedure is designed using Weka to form a powerful tool. (https://www.cs framework of machine learning algorithms developed in Java. The method of evaluation is fixed to be performed by 10-fold cross-validation. The cross-fold validation method can estimate the effect of the dataset on the prediction of software effort. The 10- The fold cross-validation method provides ten different randomizations of the dataset. The cross-validation method has been used effectively with machine learning algorithms for various datasets. [18][19][20].

IV. EXPERIMENTAL RESULTS

Through the Weka tool, the evaluation process is carried out. (https://www.cs.waikato.ac.nz/ml/weka/) Weka is a Java-based machine learning framework implemented as a collection of machine learning algorithms. Here, the evaluation procedure has been set to perform 10-fold cross-validation. Cross-fold validation helps to assess the effect of the dataset in the prediction of software effort. In the cross-fold validation, there are ten different randomizations in the cross-validation technique. Cross-validation has been successfully implemented with machine learning techniques for different datasets. [18][19][20]

As shown in Table 2, SVM gets the best prediction accuracy when it has the minimum MAE value, which is around 2.6. In addition, it can be noted that the linear regression model gets the maximum MAE value, while all other algorithms have similar MAE value, less than 3. Hence, we consider the

result of linear regression to be odd. The reasons for such results are that the values of effort cannot be expressed in a linear.

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 designed with the aim of assessing the capacity of the applied machine learning algorithms to learn from the historical data. During this experiment, unlike in the first one, the proposed 10-fold cross-validation technique is not applied. The data are divided into two, one for training and the other for testing purposes. This process of evolution is repeated using 90%, 50%, and 10% of the dataset as the training sets. The remaining part of the dataset represents the testing sets used in the three evaluation processes. Figure 1 below represents the results obtained.

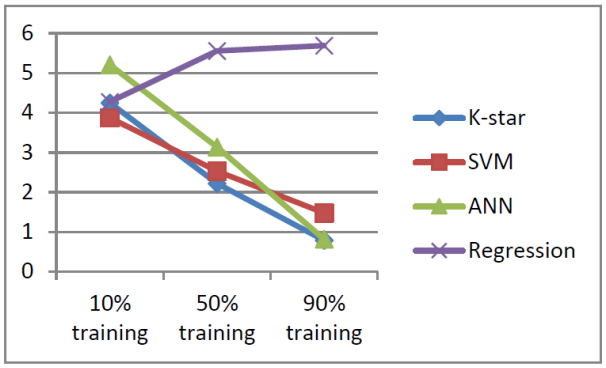

Fig. 1. MAE values for the four prediction models in the three evaluations.

As seen in Figure 1, the linear regression algorithm, also, fails to learn properly. All other algorithms learned perfectly with an increased training set.

V. CONCLUSION

In this research, four different machine learning techniques are used to build software effort estimation models. The machine learning techniques used are Artificial Neural Networks (ANN), Support Vector Machines (SVM), k-star, and linear regression. This estimation is required to forecast the actual effort using software features during the early stages. To work and test the effectiveness of the prediction models, a public dataset is taken into consideration. From the findings, the machine learning technique is effective in forecasting software effort with a lower MAE value. The lowest value of MAE with the machine learning technique is 2.6 for the SVM technique. In the future study, the effectiveness of various machine learning techniques and the use of various datasets are taken into consideration. Filters can also be used on the software features dataset to obtain better results.

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.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published.

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi TIFF or EPS file, with all fonts embedded) because, in an MSW document, this method is somewhat more stable than directly inserting a picture.

To have non-visible rules on your frame, use the MSWord “Format” pull-down menu, select Text Box > Colors and Lines to choose No Fill and No Line.