Software Development Effort Estimation Using Ensemble Machine Learning

Omar Hidmi[1], and Betul Erdogdu Sakar [2]

Abstract—In software engineering, the main aim is to develop a high quality project that fall within scheduled time and budget, this procedure is called effort estimation. Effort estimation is crucial and important for a company to do because hiring more people than needed will lead to loss of income, and hiring less people than needed will lead to delay of project delivery. The aim of this study is to estimate software effort objectively by using machine learning techniques instead of subjective and time consuming estimation methods. Models using two machine learning techniques which are Support Vector Machine (SVM) and *K*-Nearest Neighbor (*k*-NN) separately and combining those together using ensemble learning were tried on two public datasets namely Desharnais and Maxwell. Results show that svm technique outperform *k*-nn technique, also ensemble learning improves the results.

Keywords—AdaBoost, Classification, Effort Estimation, *K*-Nearest Neighbor, Machine Learning, Support Vector Machine.

I. Introduction

A project manager usually faces with the problem of estimating the effort needed to develop a project. This task is obviously dependent on the software engineers in the team but it can be measured with different techniques. Some of these techniques are easy to use but require lots of additional data while the others are time consuming and difficult to follow [1]. So it can be said that there is no simple way to make an accurate estimate of the effort required to develop a software system [2].

Even you make initial estimations; there are several drawbacks in effort estimation like the software engineers involved in the project and their ability. But still organizations need to make effort estimation. And usually all these estimators are done to estimate the effort for a brand-new project by project managers who use their experience-based judgments and knowledge from previous projects.

The aim of this work is to estimate software effort objectively by using machine learning techniques instead of using subjective and time consuming estimation methods like expert judgment and estimation by analogy. To add more, unsuitable criteria and technique for estimation may be chosen by an expert. For these reasons, it is highly advantageous to use a more structured estimation process using machine learning techniques [3].

Omar Hidmi[1] was a master student at Bahçeşehir University, Department of Software Engineering, Istanbul, Turkey. (corresponding author's phone: +972527551723; e-mail: omar.hidmi@gmail.com).

Betul Erdogdu Sakar[2] is an assistant professor at Bahçeşehir University, Department of Software Engineering, Istanbul, Turkey. (e-mail: betul.erdogdu@eng.bau.edu.tr).

II. Data and Methods

It is a very great challenge for a software company to develop a new software project in a high quality within predetermined budget and time [4]. Effort estimation is the first step that is taken in budgeting for the new software project.

This research was prepared to predict effort for new software development projects by studying and analyzing previous data combined from earlier software projects. Data that is exploited from earlier projects consists of the actual effort (the dependent attribute) and factor values that are related to the effort (the independent attributes).

Many datasets have been used to estimate software development effort that are publicly available in PROMISE repository which is one of the most famous used repositories in Software Engineering Community to estimate effort, it is a well-known, useful and real set related to projects of software engineering, made publicly available in order to encourage repeatable, verifiable, refutable, and/or improvable predictive models of software engineering [5]. These datasets have been constructed and developed by various companies, some of them are cross-company and the others are single-company related projects.

In this research, we are going to use two publicly available datasets, desharnais [6] and maxwell [7], to build a model for estimating the effort for new software development projects. They are two of the most commonly used datasets in the field of software effort estimation.

Desharnais dataset consists of 81 projects collected by J.M. desharnais in the late 1980s from a Canadian software house [6], [8], and [9]. The original dataset consists of 12 attributes but in this study the ProjectID attribute was omitted from the original dataset because it has no meaning to the study, the left are ten independent attributes and one dependent attribute (effort), all the values in this dataset are numeric but only one nominal attribute that is Language. Many researchers used this dataset in their experiments including [1], [8], [10] and many others. Despite the fact that this dataset is now more than 25 years old, it is one of the largest and most used publicly available datasets [8]. Table 1 illustrates all the variables in desharnais dataset.

Maxwell dataset is a relatively new dataset consists of 62 projects between 1985 and 1993 [7]. Each project is described by 27 attributes in which all attributes are numerical. 26 independent attributes and one dependent attribute (effort). Table 2 illustrates all the variables in maxwell dataset.

TABLE I
LIST OF VARIABLES IN DESHARNAIS DATASET

Symbol	Name
TeamExp	Team experience – measured in years
ManagerExp	Manager experience – measured in years
YearEnd	Year project ended
Entities	The number of entities in the systems data model (function points)
Transactions	A count of basic logical transactions in the system (function points)
Length	Actual project schedule in months
PointsNonAjust	Transactions + Entities (function points)
PointsAdjust	Function points adjusted by the Adjustment factor = 0.65 + (0.01 * PointsNonAdjust)
Adjustment	Function point complexity adjustment factor (Total Processing Complexity)
Effort	Actual Effort is measured in person-hours (Dependent)
Language	Programming Language

TABLE II
LIST OF VARIABLES IN MAXWELL DATASET

Symbol	Name
Syear	Software Year
App	Application Type
Har	Hardware Platform
Dba	Database
Ifc	User Interface
Sourse	Where Developed
Telonuse	Telon Use
Nlan	Number of Development Languages
T01	Customer Participation
T02	Development Environment Adequacy
T03	Staff Availability
T04	Standards Use
T05	Methods Use
T06	Tools Use
T07	Software's Logical Complexity
T08	Requirements Volatility
T09	Quality Requirements
T10	Efficiency Requirements
T11	Installation Requirements
T12	Staff Analysis Skills
T13	Staff Application Knowledge
T14	Staff Tool Skills
T15	Staff Team Skills
Duration	Duration (months)
Size	Application Size (Function Points)
Time	Time
Effort	Work Carried-out (person-hours)

The two mentioned datasets (desharnais and maxwell) have some common features such as; having the same type of project size which is described using function points (FP), besides having the same measure type for effort which is person-hours. Table 3 summarizes the datasets in term of the features and projects within each dataset.

TABLE III
SUMMARY OF THE DATASETS

Dataset	Number of features	Number of projects
Desharnais	11	81
Maxwell	27	62

To form training set and testing set, we randomly divided the datasets using two techniques which are leave-one-out cross validation and k-fold cross-validation. The two datasets were analyzed in their own context by using two machine learning techniques which are *k*-nearest neighbor (*k*-nn) and support vector machine (svm).

III. MACHINE LEARNING TECHNIQUES USED

Machine learning is considered as a subfield of artificial intelligence and it is concerned with the development of methods and techniques that enable the machine to learn and perform activities and tasks [11]. Machine learning techniques resemble the human mind in some aspects, allowing us to solve complex problems in a fast way [12]. In the last 20 years or so, machine learning approaches have been proposed as an alternative way to predict software effort [8].

In this section, we will discuss two machine learning techniques that could be used to predict effort: *k*-nn and svm, these techniques have been selected because *k*-nn is non-parametric and svm is parametric and this difference will enable us to reveal what kind of a model works better on software effort estimation datasets.

K-nn is one of the techniques used for classification problems, and it is one of the most simple classification techniques that should be the first option for a classification study when there is no past knowledge about data description [13]. *K*-nn works first by computing distance between an instance with other instances and find the *k*-nearest neighbor for that instance, then it estimates the effort [3], as shown in fig.1.

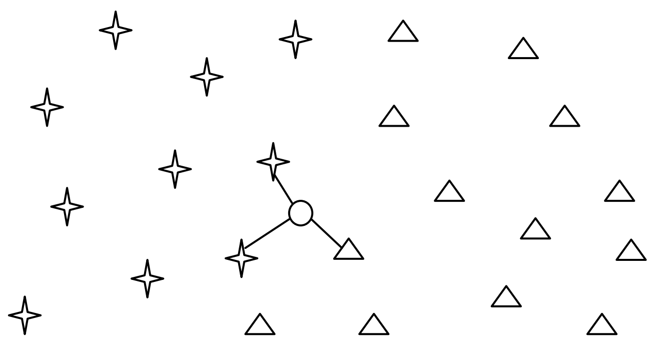

Fig. 1 Calculating the distance between new sample and the others

For the neighborhood parameter $k = 3$ (for example), first the nearest three samples are found. The distance between these samples are usually calculated by Euclidean distance (default in matlab) which is the strait-line distance between two points in Euclidean space. The majority of the samples that are belonging to one class determine the label of the new sample.

Svm is a set of machine learning methods used in many areas, such as classification and regression [14]. Svm classifier separates the instances from two different classes by using a hyper-plane which tries to maximize the margin [15]. This increases the generalization capability of the classifier. The instances that are close to or on the border are called the support vectors. The number of support vectors also represents the complexity of the model. A figurative representation of the algorithm is a good way to visualize how it works, as shown in fig. 2.

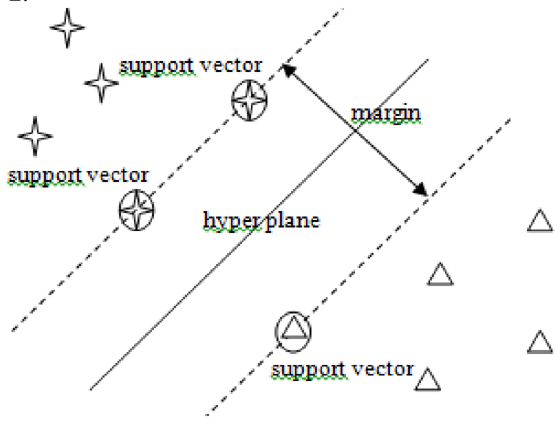

Fig. 2 Support vector machine algorithm (linear kernel)

We have transformed data into new scales using a procedure called Discretisation, to change the problem from regression to classification (i.e. to discretize numeric variables to a number of classes/intervals), it is a very important step to be done because the machine learning techniques we have picked work only with categorized data and do not predict exact numeric values but one of the classes, we used a code in matlab to convert the actual effort taken from the datasets to a specific number or label (1 to 5) according to an experiment done by Radlinski & Hoffmann [1], as explained in table 4.

TABLE IV
CLASSES FOR THE DATASETS

	Desharnais Dataset		Maxwell Dataset	
Class	Actual Effort	# of Cases	Actual Effort	# of Cases
1	0 – 1500	11	0 – 1500	9
2	1500 – 3000	21	1500 – 3000	10
3	3000 – 4500	20	3000 – 5000	11
4	4500 – 8000	14	5000 – 10000	18
5	> 8000	15	> 10000	14

IV. PROPOSED SOLUTION

The aim of this research is to improve the accuracy of estimations for software development projects, this will be done by constructing a system that will use multiple machine learning techniques and applying them to multiple datasets. In this research, we are going to propose a method to improve accuracy for effort estimation on software development projects, this method is called boosting, it is widely used to improve the accuracy, boosting is a method that is used to boost accuracy of any learning algorithm by fitting a series of models each having low error rate and then combining them into an ensemble that may perform better [16]-[18]. Many algorithms have been used for the purpose of boosting, and one of them is called Adaptive Boosting (Known as AdaBoost). AdaBoost algorithm was first introduce by Freund & Schapire [19], this algorithm was a solution to many of the difficulties for earlier boosting algorithms [20], the idea of AdaBoost is to construct a strong model sequentially by combining multiple weak classifiers into one single strong classifier, a weak classifier is a classifier which perform poorly but better than random guessing. As shown in fig. 3 each model tries to correct the mistakes of the previous one, to come up with a better accuracy for effort estimation.

V. EXPERIMENTAL RESULTS

In this section, we will present results for the experiments we have done using the two machine learning techniques on the mentioned datasets. All the experiments were conducted using a recent version of matlab software (r2014b); matlab is known to be simple, easy to use and has many ready functions built in its library.

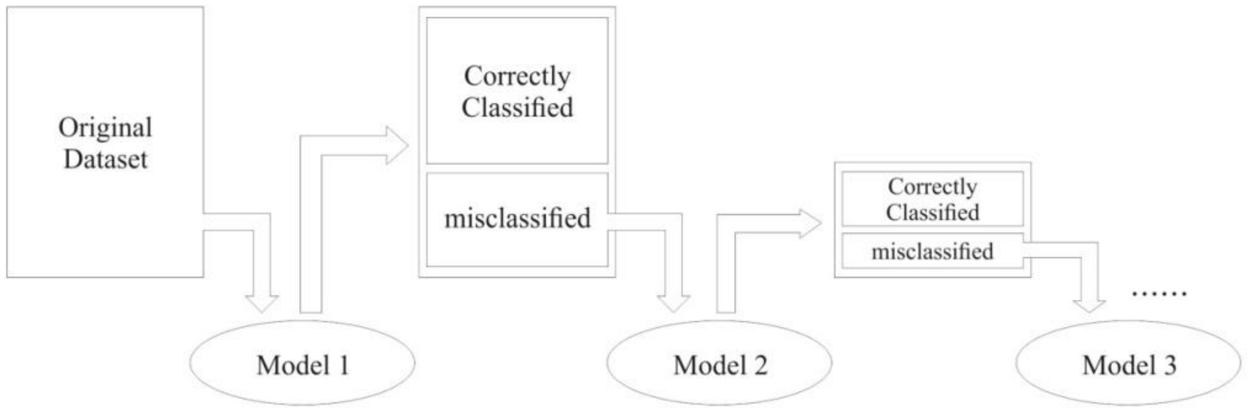

Fig. 3 How Adaptive Boosting Works

As explained earlier, desharnais dataset consists of 81 software projects and it has 4 projects missing some values, but we filled them by measuring the mean value for all the values in the same attribute, after filling all the missing values, we applied two machine learning techniques to calculate the accuracy and find the best one. On the other hand, maxwell dataset has no missing values so it is not necessary to apply a code to fill missing values in matlab like the previous dataset (desharnais), this is a very important thing to do to save time and memory usage when making the experiment.

All the experiments were conducted by scaling the dataset using a ready function in matlab called (autosc). Results are shown in tables 5 and 6.

TABLE V
RESULTS FOR THE EXPERIMENTS BEFORE BOOSTING

Accuracy results for k-nn technique

Technique	Value of *k*	On desharnais dataset	On maxwell dataset
leave-one-out cross validation	1	43.21	32.25
	3	41.98	38.71
	5	43.21	32.64
	7	50.62	35.48
10-fold cross validation	1	19.75	22.58
	3	23.46	22.58
	5	23.46	27.42
	7	23.46	32.25

Accuracy results for svm technique

# of Classes	Kernel Function	On desharnais dataset	On maxwell dataset
2 classes[a]	Linear	79.01	79.03
	Quadratic	81.48	62.90
	Polynomial	**85.18**	70.97
	RBF	79.01	51.61
5 classes	Linear	65.92	67.09
	Quadratic	76.29	75.80
	Polynomial	76.05	74.51
	RBF	81.97	**80.00**

TABLE VI
RESULTS FOR THE EXPERIMENTS AFTER BOOSTING

Boosting by using k-nn then svm

Accuracy	On desharnais dataset	On maxwell dataset
k-nn (for all data)	50.61	38.71
svm (for misclassified data)	80.00	76.31
Accuracy after boosting	90.12	**85.48**

Boosting by using svm then k-nn

Accuracy	On desharnais dataset	On maxwell dataset
svm (for all data)	85.18	79.03
k-nn (for misclassified data)	41.66	15.38
Accuracy after boosting	**91.35**	82.25

[a] We have also used only two classes for the datasets, for desharnais dataset we used one class for effort less than 3000 and the other is more than 3000, but for maxwell dataset we used one class for effort less than 5000 and the other is more than 5000.

VI. DISCUSSION AND CONCLUSION

A survey conducted by Molokken & Jorgensen [21] show that approximately between 70 and 85 percent of the respondents accepted and agreed to the importance of estimating the effort for new software development projects. This study and experiment was done to evaluate some machine learning methods which are *k*-nearest neighbor and support vector machine by applying them into two different datasets in order to make effort prediction for a new software development project.

We have seen from the results that when applying a single method alone, it has a good accuracy equals to 85% in the best scenario (this was a result when applying svm technique using 2 classes to desharnais dataset), but when we combined the classifiers, we get 91.35% accuracy when using desharnais dataset and 85.48% accuracy when using maxwell dataset. So, we can say that boosting one technique with another improves the accuracy of estimations.

For future work, other machine learning techniques can be used for classification problems, also other datasets can be used for experiments and training the model, in this way, we have more accurate estimations and predictions, also boosting with more than two techniques to get more accuracy reaching 100%.

REFERENCES

[1] Radlinski, L., & Hoffmann, W. (2010). On predicting software development effort using machine learning techniques and local data. International Journal of Software, 2(2).

[2] Sommerville, I. (2011). Software engineering. Tata McGraw Hill Publication Tata (Vol. 9th Ed).

[3] Nayebi, F., Abran, A., & Desharnais, J.-M. (2015). Automated selection of a software effort estimation model based on accuracy and uncertainty. Artificial Intelligence Research, 4(2), p45.

[4] Prakash, B.V.A., Ashoka, D. V & Aradhya, V.N.M., 2013. An Evaluation of Neural Networks Approaches used for Software Effort Estimation. Proc. of Int. Conf. on Multimedia Processing, Communication and Info. Tech., MPCIT. pp.292 – 296.

[5] Sayyad Shirabad, J. and Menzies, T.J., 2005. The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada.

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

[7] Maxwell, K. (2002). Applied statistics for software managers. Prentice Hall.

[8] Mair, C., Kadoda, G., Le, M., Phalp, K., Scho, C., Shepperd, M., & Webster, S. (2000). An investigation of machine learning based prediction systems. The Journal of Systems and Software, 53, 23–29.

[9] Menzies, T., Caglayan, B., Kocaguneli, E., Krall, J., & Turhan, B. (2012). The promise repository of empirical software engineering data. West Virginia University: Department of Computer Engineering.

[10] Braga, P. L., Oliveira, A. L. I., & Meira, S. R. L. (2007). Software Effort Estimation using Machine Learning Techniques with Robust Confidence Intervals. 7th International Conference on Hybrid Intelligent Systems (HIS 2007), 352–357.

[11] Prabhakar. (2013). Prediction of Software Effort Using Artificial Neural Network and Support Vector Machine. International Journal of Advanced Research in Computer Science and Software Engineering, 3(3), 2277–128.

[12] Schank, R. C. (1982). Dynamic memory: a theory of learning in computers and people.

[13] Peterson, L. (2009). K-nearest neighbor. Scholarpedia, 4(2), 1883.

[14] Smola, a J., & Schölkopf, B. (2004). A tutorial on support vector regression. Statistics and Computing, 14, 199–222.

[15] Vapnik, V. (2013). The Nature of Statistical Learning Theory (2nd edition). Springer Science & Business Media.

[16] Elish, M. O. (2009). Improved estimation of software project effort using multiple additive regression trees. Expert Systems with Applications, 36(7), 10774–10778.

[17] Monteiro, J. A. (2002). Multiple Additive Regression Trees a Methodology for Predictive Data Mining for Fraud Detection.

[18] Schapire, R. E., Labs, T., Avenue, P., Room, a, & Park, F. (1999). A Brief Introduction to Boosting Generalization error. Ijcai 99, 1401–1406.

[19] Freund, Y. & Schapire, R.E., 1997. A decision theoretic generalization of on-line learning and an application to boosting. Computer Systems Science. 57, pp.119–139.

[20] Schapire, R.E., 2009. A Short Introduction to Boosting. Society. 14(5), pp.771–780.

[21] Molokken, K., & Jorgensen, M. (2003). A review of software surveys on software effort estimation, 223–230.