Combining Multiple Learners Induced on Multiple Datasets for Software Effort Prediction

Ekrem Kocaguneli
Bogazici University
Computer Engineering
34342, Bebek, Istanbul,
Turkey
+90 542 660 19 56
ekrem.kocaguneli@boun.edu.tr

Yigit Kultur
Bogazici University
Computer Engineering
34342, Bebek, Istanbul,
Turkey
+90 212 359 72 27
yigit.kultur@boun.edu.tr

Ayse Basar Bener
Bogazici University
Computer Engineering
34342, Bebek, Istanbul,
Turkey
+90 212 359 72 26
bener@boun.edu.tr

Abstract

Background: *First approaches in software effort prediction depended on regression based models, whereas later models investigated more sophisticated methods like machine learning algorithms.* ***Discussion Points:*** *Single methods or models can discover only a certain part of the high dimensional space of software effort data and a common practice to increase accuracy values is to combine multiple learners. However, merely comparing models over a single dataset on the basis of precision values is not a healthy practice, since each dataset may favor a certain method. Therefore, a solid statistical test is required for comparison.* ***Method:*** *In this study, we adapt a previous study conducted by Khosgoftaar et. al. [1] in the field of software quality analysis to the field of software effort estimation and evaluate our results on the basis of statistical significance tests.* ***Conclusions:*** *Khosgoftaar et. al.'s work[1] was the first of its kind in software quality and we adapted their novel work to software effort prediction. We exploited 14 methods over 3 different software effort prediction datasets under 4 different scenarios and observed similar results to Khosgoftaar et. al., that is multiple learners induced on single dataset do not produce significantly better results.*

1. Introduction

Effort prediction in software industry enables practitioners to predict the cost of a future project and thereby allows them to allocate available resources to optimize the quality, budget and schedule in projects. Furthermore effort prediction models serve to a number of purposes such as risk analysis, project planning and control as well as improving investment analysis [2]. Therefore, particularly after the 60s, topic of effort prediction in software projects became a point of focus for researchers [3]. Although a large number of approaches have been proposed over the years and many alternatives have been investigated, we can group those approaches under two main categories: 1) expert based and 2) model based effort prediction models [4]. Expert based models rely on the experience and judgment of human experts. On the other hand model based approaches make use of a certain type of algorithmic model and they try to predict the effort of a future project by analyzing a dataset of past projects' effort data.

Regardless of the method we exploit for software effort prediction, we need to bear in mind that each method and model make its own assumptions and come with its own bias [4]. Therefore, it is mostly recommended to complement the prediction of a model with the judgment of an expert [2].Moreover, effort prediction data is inherently a high dimensional data with a limited number of instances and a single method may fall short of fully exploring a high dimensional space. A common practice to address the problem is resorting to combination of multiple learners. This approach is referred as multi-learner or multi-expert system [15]. As Khoshgoftaar et. al. claimed [1], although combination of multiple learners with complementary biases may seem to offer a solution to increase predictive accuracy of a model, that does not always prove to be efficient in practice. Khoshgoftaar et. al. observed the validity of this fact in their extensive work [1] in the context of software quality and have come up with the suggestion that increasing the information content of training datasets for knowledge discovery is the key to increase the ultimate predictive accuracies [1,16,17]. In our study, we adapted the experimental settings proposed by Khoshgoftaar et. al. to an alternative context: Software effort prediction. Although we have used the same experimental settings in a different context, we reached similar conclusions, i.e. combination of learners do not significantly improve accuracy values and as it was the case for software quality. Similar to software quality, in software effort prediction the volume and diversity of the data also need to be increased for better prediction performances.

2. Related Work

Exploiting software project data for effort prediction purposes has been extensively elaborated and a number of models have been developed. Among the parametric methods COCOMO[7], COCOMO II[8], COCOTS[8], and RUPS[18] can be given. However, parametric methods like COCOMO have their own drawbacks; for instance they can only be calibrated locally [10]. Machine learning approaches are good means to address local tuning related problems and a variety of machine learning approaches have been used for effort prediction purposes as well [10,19,20]. However, when dealing with software effort prediction, all of these methods suffer from the large deviation problem and low accuracy values [4]. To address those problems, pre-processing methods such as PCA and wrapper have been combined with various types of learners. Among those methods, variable reduction (column pruning) and stratification (row pruning using nearest neighbor) were reported to be most effective tuning methods [21]. Although single learners combined with row and/ or column selection methods have addressed the deviation problem to some extent, they are can not remove the inductive bias inherent in each learner. In other words, each learner is based on its own premises and come with its own assumptions [16]. To address these problems, a combination of complementary learners seems to be a promising solution. Khoshgoftaar et. al. [1] investigated this in the software quality domain and they exploited 17 learners on 7 datasets for 4 different scenarios. Khoshgoftaar et. al. have used voting to combine multiple learners. Their study is unique in terms of its content and application domain [1] and they came up with the conclusion that combination of learners induced on single datasets does not produce a significant increase in the accuracy values. In our study, we replicate their study in software effort estimation domain and we use 14 learners that are induced on 3 different datasets. The problem of Khoshgoftaar et. al. was a classification problem whereas in our case the problem is a regression type of a problem. Therefore, the algorithms used in two studies are not all the same. However, we also chose our algorithms from the same machine learning toolbox (WEKA) [17]. We will provide more details regarding the algorithms and the methodology we adopted in Section 3.

3. Learners and Adopted Methodology

In our study, we follow the same base principles as Khoshgoftaar et. al., that is, we are using multiple experts induced on different datasets. As Alpaydin suggests [23], there is no point in combining multiple learners that always make similar decision. Hence, we selected 14 learners belonging to different families. All the selected learners can be found in WEKA data mining tool [17].

3.1 Selected Learners

In Table 2, we give the brief descriptions of the learners we have used as well as their families in the same way as they were given in WEKA toolkit[17]. Furthermore, we also included the acronyms that are used for each algorithm in WEKA data mining tool [17]. The method we have used in our experiments while combining the learners was voting, as it was also used in the study of Khoshgoftaar et. al. We can regard voting as a regularizer, such that it smoothes out the predictions of learners [23].

3.2 Adopted Methodology

While applying learners on the datasets, we followed the same experimental design described in the work of Khoshgoftaar et. al., consisting of 4 different scenarios. Below, we give the fundamental idea laying behind each scenario. The experimental results of the scenarios will be provided in Section 5.

Scenario 1 - Single Learner, Single Dataset: In this scenario, each single learner is trained and tested on each single datasets one by one. We have used ten-fold cross validation in this scenario. Scenario 1 yielded the results, which are given on Table 3.

Scenario 2: Multi Learner, Single Dataset: The intuition in that scenario is to use one of the datasets for training and the remaining 2 for testing and making this for each dataset. Furthermore, we do that for every learner and combine their predictions via voting.

Scenario 3: Single Learner, Multi Dataset: Each learner is trained with 2 of the 3 datasets and is tested on the remaining dataset. This procedure is repeated for each dataset, so that each dataset is used as a test set at least for once.

Scenario 4: Multi Learner, Multi Dataset: In the last scenario, each learner is trained on 2 datasets and is tested on the remaining dataset. Difference of scenario 4 from scenario 3 is that the predictions of learners are again combined via voting.

Table 1. Selected Learners [17]

Family	Description	Acronym
Function Based Learners	Gaussian Processes for regression without hyperparameter-tuning	GaussianProcesses
	Uses backpropagation for prediction	MultilayerPerceptron
	A normalized Gaussian radial basis function network.	RBFNetwork
	Alex Smola and Bernhard Scholkopf's sequential minimal optimization algorithm for training a support vector regression model	SMOReg
	Support vector machine for regression	SVMReg
Instance Based Learners	K-nearest neighbours learners	IBk
	Locally weighted learning	LWL
Meta Learner	Uses bagging a learner to reduce variance	Bagging (with fast decision tree learner)
	Meta learner that enhances the performance of a regression base classifier	Additive Regression (with decision stump)
	Constructs a decision tree based learner to provide highest accuracy on training data while improving on generalization accuracy	RandomSubSpace (with fast decision tree learner)
Tree Based Learners	Uses decision stump for learning	DecisionStump
	M5 Base	M5P
Rule Based Learners	Single conjunctive rule learner that can predict for numeric and nominal class labels	ConjunctiveRule
	Builds and uses a simple decision table majority learner	DecisionTable

4. Dataset and Evaluation Criteria

4.1 Dataset

In our study, we used 3 different datasets: *coc81*, *nasa93* and cocomo*nasa_v1*. All of the three databases are publicly available in PROMISE data repository [22] and all of them were collected in the form of COCOMO software cost model, where project efforts are measured in terms of calendar months [2] and is linearly correlated with software size.

Table 2. Datasets

	coc81	*nasa93*	*cocomonasa_v1*
Number of Features	17	17	17
Number of Instances	63	93	60

4.2 Evaluation Criteria

In evaluating the accuracy of a learner in the context of software effort prediction for COCOMO datasets, there are standardized criteria such as Pred(K) and MMRE [19]. MMRE stands for Mean Magnitude Relative Error and it is the average of the absolute error over all test instances. The formula of MMRE is given in Equation 1.

$$MMRE = 100 * \frac{1}{N} * \sum_{i=1}^{N} \frac{|actual_i - predicted_i|}{predicted_i} \quad (1)$$

Pred(K) is another very commonly exploited evaluation criteria for software effort prediction. Pred(K) is the ratio of the number of test instances whose predicted cost is within ±K% range of the actual effort. K value may change from application to application, however K = 30 is a common choice [4] and so will we use K = 30 in our work. Equation 2 describes how Pred(30) is calculated, where T_i30 is the i^{th} test instance whose accuracy is within 30 percent and T_inot30 is the i^{th} instance whose accuracy is not within 30 percent.

$$Pred(30) = \frac{(\sum_{i=1}^{N} T_i 30)}{(\sum_{i=1}^{N} T_i 30 + \sum_{i=1}^{M} T_i not\,30)} \quad (2)$$

Although MMRE and Pred(K) have been seriously condemned due to multiple reasons, they still remain extensively utilized and they do not have a strong replacement[20]. However, since MMRE is greatly influenced by noises in the dataset, we will heavily depend on Pred(30) values while evaluating our experimental results.

5. Experimental Results

In Section 3.2 we have provided the methodology of our experiments as well as the possible scenarios, which would be investigated. In this section, we will provide the results elicited from each scenario.

In Table 3, we provide our results of Scenario 1, where each single learner is tested on each one of the datasets. As we see in Table 3, application of single learners to each single dataset can yield pred(30) values as high as 50%. For ease of reading, we have written top 2 performing classifier's pred(30) values in bold face in Table 3. From Table 4, we see that when we combine the predictions of multiple learners for each dataset, the accuracy is far from outperforming single learners. In Table 4, we also give Pred(30) value of the best performing single learner (inside parenthesis with bold face). When we also compare the MMRE values between Table 3 and Table 4, we see that combination of learners yields higher MMRE rates, which is not an improvement either. For the validity of our results that are proposed here, we applied ANOVA tests and the details are provided in Section 6.

The results of Scenario 3 and Scenario 4 are provided in Table 5 and Table 6 respectively. When the results of the 3rd and the 4th scenarios are compared to see whether combining multiple learners has provided a significant increase in the Pred(30), we see a situation that is similar to the comparison of Scenario 1 and Scenario 2; that is, for the multiple dataset case, we cannot observe a significant increase in the Pred(30) values as well.

Table 3. Scenario 1: Results for Single Learner, Single Dataset

	coc81		nasa93		nasa_v1	
LEARNER	pred(30)	mmre	pred(30)	mmre	pred(30)	mmre
Gaussian Process	15.71	669.92	22.09	454.75	20	295.33
MultilayerPerceptron	15.47	667.5	23.78	511.91	36.67	228.62
RBFNetwork	7.61	931.17	17.08	623.48	12	440.14
SMOReg	18.57	484.18	**49.48**	**332.53**	48.33	157.04
SVMReg	**20.47**	**483.08**	**45.25**	**328.56**	**50**	**155.92**
IBk	17.38	756.5	33.49	619.3	41.67	273.87
LWL	9.76	678.26	15.06	555.99	11	280.74
Additive Regression	4.52	829.34	22.05	446.6	38.33	221.92
Bagging	8.8	718.13	30.97	422.25	36.67	184.47
RandomSubSpace	7.38	779.94	25.86	474.72	13.33	346.61
DecisionStump	11.19	717.47	20.05	567.64	13.33	303.21
M5P	**20.95**	**516.62**	35.84	346.85	**50**	**158.31**
ConjunctiveRule	6.19	858.36	18.94	621.83	16.67	299.81
DecisionTable	12.14	598.95	17.65	564.74	33.34	261.12

Our findings after application of all 4 scenarios are similar to those of . Khoshgoftaar et. al. In their study [1], conducted on software quality, they found that combination of learners trained on single dataset does not improve the predictive accuracy when compared to that of single learner induced on a single dataset.
Indeed, the observation which was found for software quality by Khoshgoftaar et. al. can be observed for software effort prediction data as well. From our findings, we can also suggest that predictive accuracy of multiple learners in terms of Pred(30) values do not increase significantly, when compared to the results of single learners both on single datasets and on multiple datasets.

Table 4. Scenario 2: Results for Multiple Learners, Single Dataset

coc81		nasa93		nasa_v1	
Pred (30)	mmre	Pred (30)	mmre	Pred (30)	mmre
13.55 **(20.95)**	705.06	25.45 **(49.48)**	496.06	26.3 **(50.00)**	338.08

6. Threats to Validity

We address the threats to validity in two areas: 1) Internal validity and 2) external validity. Internal validity deals with the extent to which cause and effect relationships between dependent and independent variables holds. For the internal validity of our results, we have applied one-way ANOVA (analysis of variance) tests. One-way ANOVA tests are applied to check whether two groups are statistically different from one another in terms of the measured quantity. Multiple learner-single dataset values are significantly different than the single learner-single dataset values with a p-value of 0.00. As for the external validity of our results, we need to see that results elicited from domain specific datasets do hold for other datasets coming from different domains. Since we adopted a similar experimental design from Khoshgoftaar et. al. and came up with similar results in a different domain, we can say that our study is an external validation of Khoshgoftaar et. al.'s study and vice versa. However, software effort datasets are very limited in number and trying the same experiments on more datasets can further improve external validity of this study.

Table 5. Scenario 4: Results for Single Learner, Multiple Datasets

coc81		nasa93		nasa_v1	
Pred (30)	mmre	Pred (30)	mmre	Pred (30)	mmre
13.41	705.06	25.45	496.06	34.36	250.25

7. Conclusion

In our work, we have adapted a previous novel work of Khoshgoftaar et. al. from software quality domain to software effort prediction domain and wanted to see whether the findings that came out of software quality data would hold for software effort prediction as well. Both in terms of selected learners and in terms of the number of datasets, we needed to make some changes, due to difference of domains as well as due to the nature of problems (i.e. software quality prediction is a classification problem, whereas software effort prediction is a regression one).

Table 6. Scenario 3: Results for Single Learner, Multiple Datasets

	coc81		nasa93		nasa_v1	
LEARNER	pred(30)	mmre	pred(30)	mmre	pred(30)	mmre
Gaussian Process	11.905	747.63	27.205	494.04	25	345.905
MultilayerPerceptron	11.905	1460.425	41.065	342.705	27.5	234.92
RBFNetwork	10.315	778.355	13.81	594.32	13.335	522.58
SMOReg	15.37	583.64	**44.82**	**363.115**	45	185.51
SVMReg	13.735	581.31	43.06	364.68	46.665	187.865
IBk	**19.05**	**633.685**	23.88	499.12	**55.835**	**280.725**
LWL	13.74	615.435	15.275	505.47	16.665	288.74
Additive Regression	16.165	702.425	24.1	705.88	18.565	491.74
Bagging	16.165	563.405	28.07	416.14	28.33	222.385
RandomSubSpace	12.25	589.645	19.54	521.345	14.165	403.07
DecisionStump	11.905	665.48	14.625	548.83	16.67	424.245
M5P	13.075	524.675	20.885	589.63	29.935	341.705
ConjunctiveRule	10.28	711	16.875	554.13	10.67	503.38
DecisionTable	11.89	713.85	23.125	445.5	19.885	300.35

The empirical experimental results of the two studies point to the same direction, although their domains are different. Khoshgoftaar et. al. found that combination of learners in software quality does not significantly improve average predictive accuracy [1] and concluded that rather than trying to derive high accuracy values out of noisy or information content-wise weak datasets, practitioners shall put more effort into information gathering. Also our findings suggest that utilization of complex machine learning algorithms do not necessarily result in higher prediction performances. Therefore the same conclusion is also valid for software effort estimation: the information content of a dataset has a more decisive role on the prediction accuracies than complex algorithms.

Acknowledgement

This research is supported in part by Tubitak under grant number EEEAG108E014, and in part by Ibtech A.Ş.

8. References

[1] Khoshgoftaar, T. M., Rebours, P., and Seliya, N. 2009. Software quality analysis by combining multiple projects and learners.*Software Quality Control* 17, 1 (Mar.2009),25-49. DOI=http://dx.doi.org/10.1007/s11219-008-9058-3

[2] B. Boehm, C. Abts and S. Chulani, Software development cost estimation approaches—a survey. Annals of Software Engineering 10 (2000), pp. 177–205.

[3] Chrysler, E. 1978. Some basic determinants of computer programming productivity. *Commun. ACM* 21, 6 (Jun. 1978), 472-483. DOI= http://doi.acm.org/10.1145/359511.359523

[4] T. Menzies, Z. Chen, J. Hihn, and K. Lum. Selecting best practices for effort estimation. IEEE Transactions on Soft-ware Engineering, 32:883–895, 2006.

[5] M. Jørgensen, A review of studies on expert estimation of software development effort, Journal of Systems and Software 70 (1–2) (2004), pp. 37–60.

[6] Rowe, G., Wright, G., 2001. Expert opinions in forecasting: The role of the Delphi process. In: Armstrong, J.S. (Ed.), Principles of Forecasting: A Handbook for Researchers and Practitioners. Kluwer Academic Publishers, Boston, pp. 125–144.

[7] B. Boehm. Software Engineering Economics. Prantice Hall, 1981.

[8] B. W. Boehm, Clark, Horowitz, Brown, Reifer, Chulani, R. Madachy, and B. Steece. Software Cost Estimation with Cocomo II. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

[9] L. C. Briand. A pattern recognition approach for software engineering data analysis. IEEE Transactions on Software Engineering, 18:931–942, 1992.

[10] Y. Kultur, B. Turhan, and A. B. Bener. Enna: software effort estimation using ensemble of neural networks with associa-tive memory. In SIGSOFT '08/FSE-16: Proceedings of the 16th ACM SIGSOFT International Symposium on Founda-tions of software engineering, pages 330–338, New York, NY, USA, 2008.

[11] B. Stewart. Predicting project delivery rates using the naive-bayes classifier. Journal of Software Maintenance and Evolution: Research and Practice, 14:161–179, May 2002

[12] Jalali, O., Menzies, T., Baker, D., and Hihn, J. 2007. Column Pruning Beats Stratification in Effort Estimation. In *Proceedings of the Third international Workshop on Predictor Models in Software Engineering* (May 20 - 26, 2007). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 7. DOI= http://dx.doi.org/10.1109/PROMISE.2007.3

[13] Chen, Z., Menzies, T., Port, D., and Boehm, B. 2005. Feature subset selection can improve software cost estimation accuracy. In *Proceedings of the 2005 Workshop on Predictor Models in Software Engineering* (St. Louis, Missouri, May 15 - 15, 2005). PROMISE '05. ACM, New York, NY, 1-6. DOI= http://doi.acm.org/10.1145/1083165.1083171

[14] Kohavi, R. and John, G. H. 1997. Wrappers for feature subset selection. *Artif. Intell.* 97, 1-2 (Dec. 1997), 273-324. DOI= http://dx.doi.org/10.1016/S0004-3702(97)00043-X

[15] Gamberger, D., Lavrac, N., and Dzeroski, S. 1996. Noise Elimination in Inductive Concept Learning: A Case Study in Medical Diagnosois. In *Proceedings of the 7th international Workshop on Algorithmic Learning theory* (October 23 - 25, 1996). S. Arikawa and A. Sharma, Eds. Lecture Notes In Computer Science, vol. 1160. Springer-Verlag, London, 199-212.

[16] E. Alpaydın, 2004. Introduction to Machine Learning, MIT Press, 2004

[17] Witten, I. H., & Frank, E. (2000). Data mining, practical machine learning tools and techniques with Java implementations. San Francisco, CA: Morgan Kaufmann.

[18] P. Kruchten. The Rational Unified Process: An introduction. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[19] Burgess, C. J. and Lefley, M., 2001. Can genetic programming improve software effort estimation? A comparative evaluation", Information and Software Technology 43, pp 863-873

[20] Srinivasan, K. and Fisher, D., 1995. Machine Learning Approaches to Estimating Software Development Effort, IEEE Transactions on Software Engineering, Vol. 21, No. 2 , 1995, pp (126-137)

[21] K. Lum, T. Menzies, and D. Baker. 2cee, a twenty first cen-tury effort estimation methodology. ISPA / SCEA, pages 12 – 14, 2008.

[22] G. Boetticher, T. Menzies, and T. Ostrand. The PROMISE Repository of Empirical Software Engineering Data, 2007. http://promisedata.org/repository.

[23] Alpaydin, E. (1998). Techniques for combining multiple learners. In E. Alpaydin (Ed.), Proceedings of engineering of intelligent systems conference (Vol. 2 of 6–12). ICSC Press.