Data Mining Techniques in Software Effort Estimation

Gregory Timmons
North Carolina State University
gbtimmon@ncsu.edu

Pooja Asher
North Carolina State University
pmasher@ncsu.edu

ABSTRACT

An important application of Data Mining is the creation of predictive models. In the domain of Software Engineering, data mining is being applied to Software Effort Estimation with improved results. The accuracy of this estimation is often crucial to these projects and a range of complex prediction models are being used with varied results. A number of studies have been done on which data mining techniques work best for the Software Effort Estimation premise, with no definitive results. In this study, we try to compile a few of the papers done in this domain. We also try to provide a comparison, and comment on improvements and future work for the same.

Keywords

Data mining, Software Effort Estimation, Regression, Prediction Model

1. INTRODUCTION

In software industry, project managers usually rely on their previous experience to estimate the number men/hours required for each software project. Since the software products are 'intangible', companies have a tough time evaluating the effort required for it. The accuracy of such estimates is a key factor for the efficient application of human resources. An study published by the Standish Group's Chaos states that 66 percent of the software projects analyzed were delivered with delay or above the foreseen budget, or worse, they were not finished.[14] This makes it necessary to evaluate the effort with minimal error but choosing the right method to do so is key.

Software Effort Estimation uses a predictive model that has been implemented by a number of techniques. Earlier formal techniques like COCOMO, COCOMO II and Function Points Analysis were moderately effective techniques for the domain. Various data mining techniques have been used since, to improve the accuracy of these prediction models. In this field of research, the required effort to develop a new project is estimated based on historical data from previous projects. This information can be used by management to improve the planning of personnel, to make more accurate tendering bids, and to evaluate risk factors.[7][17] Predictive data mining techniques use this past data, and analyse and compare it to the current project data to make an accurate estimate.

Also, a major concern for this research is that Effort estimation models suffer from very large performance deviations.These large deviations explain much of the contradictory results in the effort estimation literature.[9]. A number of papers we studied show varied results in estimates using data mining techniques.

The remainder of this paper is structured as follows: First, Section 2 presents our motivation to study data mining techniques for software effort estimation. Then, in Section 3 the different techniques employed in the studies are discussed. Section 4 reflects upon the data sets and performance measures. In Section 5, we discuss the results from all the studies and comment on their addition to the domain research. The paper is concluded by Section 6 providing general conclusions and topics for future research.

2. MOTIVATION

Our motivation for exploring this topic is that software effort estimation is a predictive model with diverse results across different techniques. Multiple studies on this topic focus on different aspects of the prediction process. None of the studies have given definitive irrefutable results on which data mining technique performs the best for software effort estimation. A number of techniques do compare old techniques to newer ones [4] [10] but are not enough to prove one techniques works over another. Also, among the found studies, there are huge discrepancies as well, which make comparison difficult. [9]

3. TECHNIQUES

We are studying papers with a wide variety of data mining techniques that could ideally work well with software effort estimation. We also account for and compare techniques not related to data mining for comparison. The first paper[3] is a comparative study of a number of techniques compiled from a number of other papers. Some of the techniques used were:

- Ordinary least squares regression.
- OLS regression with log transformation.
- OLS regression with Box Cox (BC) transformation.

 ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

- Robust regression.
- Ridge regression.
- Least median squares regression.
- MARS.
- CART.
- Model tree.
- Multilayered perceptron neural network.
- Radial basis function networks.
- Case-based reasoning.
- Least squares support vector machines

Note that since software effort estimation requires a continual target class and predictive model, regression is very commonly used for the application.

The third paper[14] talks about using Genetic Algorithm based method for feature selection and parameters optimization for machine learning regression as a means to aid effort estimation. This is in addition to Support vector regression, Multi-layer Perceptron (MLP) neural network, Model trees commonly used for effort estimation currently. The paper aims to prove that use of Genetic Algorithm enhances the results of either of these methods.

The second paper[5] uses decision trees and rule-based predictive models which could lend themselves well to effort estimation. This paper also talks about using decision trees as a good means of representation. The paper rigorously advocates decision making and it has been used multiple times for effort estimation as well with sufficiently accurate results.[2] [12] Often decision trees are used in the form of CART(Classification and Regression Trees) for the domain.[3][15]

In the fourth paper,[11] they evaluate different performance measures for software effort estimation. It uses machine learning approaches such as Multi-Layer Perceptrons (MLPs), Radial Basis Function networks (RBFs) and Regression Trees (RTs). MLPs are artificial neural networks which are composed of at least three layers of neurons. RBFs are artificial neural networks whose neurons of the first layer compute a radial basis function, allowing them to perform local learning. RTs are easy-to-understand structures that provide if-then rules to perform regression based on the values of the input attributes.[11]This paper also mentions that this approach has high implementation complexity and is not fully automated. It then goes on to suggest MOEA (multiobjective evolutionary algorithm).MOEAs are population-based optimisation algorithms that evolve sets of candidate solutions by optimising two or more possibly conflicting objectives.The specific algorithm they tested on was a harmonic MOEA.

Paper five,[8] compares learning from local or global data sources. This study proposes their algorithm for effort estimation and defect prediction. Their clustering algorithm, named WHERE, assumes that the dimensions of most interest are the dimensions of greatest variability. This assumption is shared by other researchers such as those using feature weighting based on variance or principal component analysis (PCA). They use the the WHICH contrast rule learner.WHICH builds these rules by looping over attribute value combinations, combining those that look most promising.

In the sixth paper[18], the article proposes an automatically transformed linear model (ATLM) as a suitable baseline model for comparison against SEE methods. The paper discusses that there is no definitive baseline model for comparison with other effort estimation techniques. A number of possibilities for models exist:

- a linear regression model
- a simple decision tree
- generalised linear models
- extensions to linear and tree-based modelling etc

The paper compares these as baseline techniques and comments the following: Why do these discrepancies exist in results of newer methods? The two possible reasons discussed include datasets that work well for one technique over the other, and tuning required on the data before regression can be applied.It evaluates that the current baseline can be improved and then suggests ATLM as a better baseline model compared to the above.

The next paper talks about selecting best practises in the domain and is focused on the large deviation in conclusions while researching effort estimation.Methods are divided into model-based and expert-based methods and evaluated accordingly, and the paper only focuses on model-based techniques. Deviation is caused on biases based on the methods, models, datasets and data miners. The paper then uses their method COSEEKMO to explore various estimation methods.The COSEEKMO effort-modeling workbench applies a set of heuristic rejection rules to comparatively assess results from alternative models. Using these rules, and despite the presence of large deviations, COSEEKMO can rank alternative methods for generating effort models and has been proven to show good results.

Another paper discusses that Software effort estimation techniques fall into four categoriesâĂŤempirical, regression, theory-based, and machine learning techniques.[15] This paper proposed a Bayesian belief network to learn and predict software development effort. Since they found no study at the time, that compares the performance of the Bayesian networks with other popular forecasting models, they benchmark the performance of the Bayesian networks with non-parametric neural networks and CART algorithm.

When analysing these techniques we find that techniques have evolved as preferred methods over time, from classic expert models, to formal methods like COCOMO and other predictive regression models, and currently complex data mining models like MLP, neural networks etc.

For formal techniques that do not use data mining, regression-based estimation approaches dominate. Notice that regression-based estimation approaches include most common parametric estimation models, e.g., the COCOMO model. Roughly half of all estimation papers in the past tried to build, improve or compare with regression model-based estimation methods.[6].

Even though we planned to study only data mining techniques in the study, COCOMO played a big role in process. Multiple papers discussed merits, used COCOMO as a comparison or used it as a baseline. Using COCOMO II or

an improvement of the same [1] can be surprising considering so many newer and more complex techniques. The recommendation in paper eight about When COCOMO-style attributes are available,(i) using that data and (ii) use COCOMO to generate predictions; can be surprising but they also prove that COCOMO is a powerful tool with the right datasets and when used correctly.

4. DATA AND PERFORMANCE MEASURES

4.1 Data

Detailed and complex datasets are often required for software effort estimation. As discussed with the professor, datasets for software effort estimation can be small with a large number of undesired attributes. Finding accurate datasets with all required parameters and also effective sampling are required for more accurate results. Many common datasets are used by a number of these studies and not always with the same results.

The data sets typically contain a unique set of attributes that can be categorized as follows:[3]

- Size attributes are attributes that contain information concerning the size of the software project. This information can be provided as Lines Of Code,(LOC), Function Points, or some other measure.
- Environment information contains background information regarding the development team, the company, the project itself (e.g., the number of developers involved and their experience), and the sector of the developing company.
- Environment information contains background information regarding the development team, the company, the project itself (e.g., the number of developers involved and their experience), and the sector of the developing company.
- Project data consist of attributes that relate to the specific purpose of the project and the project type. Also attributes concerning specific project requirements are placed in this category.

Paper one provides an interesting table with important characteristics of common data sets, shown in Fig 1 and Fig 2. [3]

The third paper uses common data sets like: Desharnais data set, a data set from NASA, the COCOMO data set, Albrecht data set, Kemerer data set, a data set used by Koten and Gray [16][14].

Similarly, paper 4 and paper 5 used common datasets from the PROMISE repository. The eighth paper uses common COCOMO datasets CocII and Nasa93 and also observes large variances in data.

4.2 Performance measures

Accurate comparison of the methods can only be done by using appropriate perfomance measures. All studies employ one or more performance measures for their results and they are largely varied as well. The first paper uses a selection of metrics (MdMRE, Pred25, and rs). The second paper uses measuring accuracy as the Percentage Correctly Classified (PCC) which is the number of correct answers divided by the total number of questions.[5]

Paper three, paper four as well as paper eight use MMRE and PRED for their measures. The sixth paper makes use of the error measurements MMRE, LSD(logarithmic standard deviation), and PRED(25) to assess model quality. Paper eight also uses correlation to compare results in addition to performance measures.

The fourth paper presents an analysis on which performance measure is best suited for the domain. Overall, even though a certain solution may appear better than another in terms of a certain measure, it may be actually worse in terms of the other measures. As none of the existing performance measures has a perfect behaviour, the software manager may opt to analyse solutions using several different measures, instead of basing decisions on a single measure. [11]

5. RESULTS

All the studied papers have tried to compare existing models and add a new model or an improvement to it. They also have variances in the results.

The first paper observes, ordinary least squares regression with logarithmic transformation (Log + OLS) is the overall best performing technique. However, a number of other techniques including least median squares regression, ordinary least squares regression with Box Cox transformation, and CART, are not significantly outperformed by Log + OLS. This paper compares the used performance measures to find different results. Since this is a comparative study, it finds Log + OLS to be more successful but also finds conflict among the performance measures regarding ranking of the rest of the techniques. It also presents a comparison where COCOMO is compared to Log + OLS and one works better for one data set and the other technique for the other. These results also indicate that data mining techniques can make a valuable contribution to the set of software effort estimation techniques, but should not replace expert judgment. Instead, both approaches should be seen as complements.

The second paper results showed that, on the aspect of comprehensibility, decision tables provide significant advantages. For each part of the experiment, the respondents were able to answer the questions faster, more accurately and more confidently using decision tables than using any of the other representation formats. Additionally, a majority of the users found decision tables the easiest representation format to work with. This user study clearly concluded benefits of decision trees and it's versions which about work well for effort estimation.

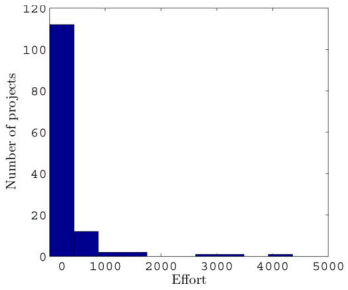
The third paper found results similar to earlier papers with positive results. In all the data sets, their method achieved the best performance in terms of PRED(25).Also,their simulations have shown that the GA-based method was able to improve performance of the three machine learning techniques considered in the problem of software effort estimation. Furthermore, the use of GA for feature selection has resulted in significant reduction of the number of input features for two of the data sets considered.[14] They conclude that a number of effort estimation techniques using data mining have improved results using their Genetic Algorithm for enhancement and this can add to a possible best solution for our problem.

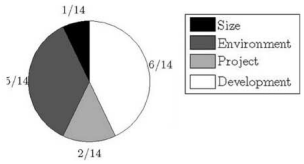
The fourth paper showed a complex set of results for each dataset and different performance measures(shown in Fig 3)[11] This result shows how using different datasets or different performance measures can alter what the ideal data

ESA

Skewness: 4.89
Kurtosis: 30.13
Minimum effort: 3 *Minimum Kloc: 1.5*
Mean effort: 264 *Mean Kloc: 56*
Max effort: 4361 *Max Kloc: 413*

Number of observations: 131
Number of attributes: 14
Previously used in other studies, e.g. [15], [64], [65]
Reference: The ESA initiative for Software Productivity Benchmarking and Effort Estimation
http://www.esa.int/esapub/bulletin/bullet87/greves87.htm

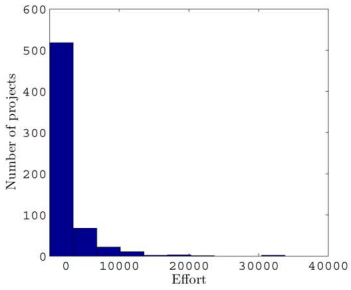


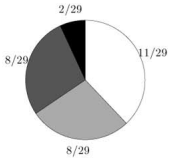


Experience

Skewness: 4.45
Kurtosis: 32.30
Minimum effort: 55 *Minimum FP: 7.14*
Mean effort: 4248 *Mean FP: 728*
Max effort: 67576 *Max FP: 14092*

Number of observations: 627
Number of attributes: 29
Previously used in other studies, e.g. [14], [50], [64], [66], [67]
Reference: http://www.fisma.fi

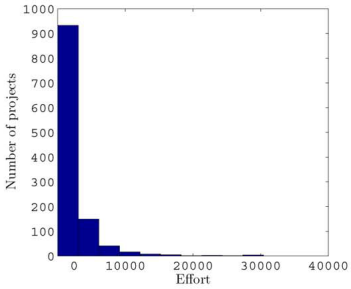


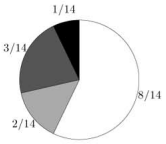


ISBSG

Skewness: 4.40
Kurtosis: 30.50
Minimum effort: 16 *Minimum FP: 4*
Mean effort: 4226 *Mean FP: 374*
Max effort: 60826 *Max FP: 7400*

Number of observations: 1160
Number of attributes: 14
Previously used in other studies, e.g. [13], [16], [68]
Reference: http://www.isbsg.org

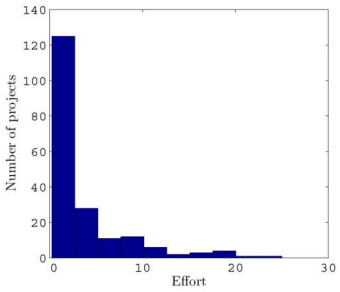




USP05

Skewness: 2.10
Kurtosis: 7.14
Minimum effort: 0.5 *Minimum FP: 0*
Mean effort: 7.25 *Mean FP: 25*
Max effort: 50 *Max FP: 321*

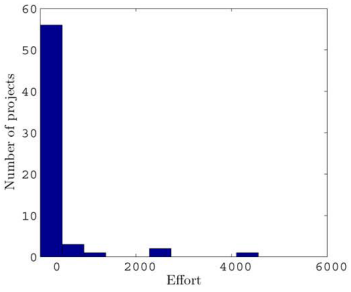
Number of observations: 193
Number of attributes: 14
Previously used in other studies, e.g. [16], [69]
Reference: [16], [69]
http://www.promisedata.org/?p=48



Coc81

Skewness: 4.37
Kurtosis: 23.08
Minimum effort: 5.9 *Minimum Kloc: 1.98*
Mean effort: 683 *Mean Kloc: 77*
Max effort: 11400 *Max Kloc: 1150*

Number of observations: 63
Number of attributes: 16
Previously used in other studies, e.g. [13], [55], [70]–[72]
Reference: [7]
http://www.promisedata.org/?p=6



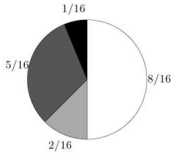
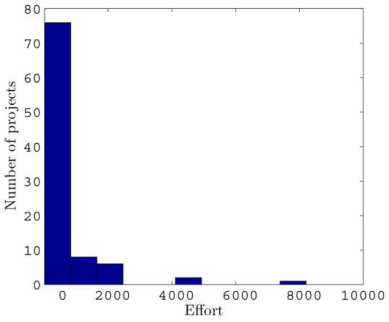
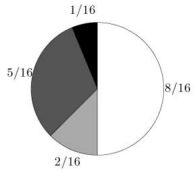


Figure 1: "Information on datasets"

Cocnasa
Skewness: 4.19
Kurtosis: 24.81
Minimum effort: 8.4 *Minimum Kloc: 0.9*
Mean effort: 624 *Mean Kloc: 94*
Max effort: 8211 *Max Kloc: 980*

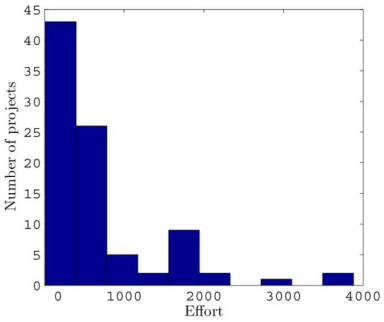
Number of observations: 93
Number of attributes: 16
Previously used in other studies, e.g. [70], [72]
Reference: http://www.promisedata.org/?p=35

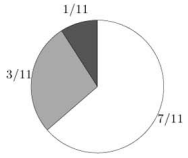




Euroclear
Skewness: 2.25
Kurtosis: 8.49
Minimum effort: 24
Mean effort: 1402
Max effort: 7758

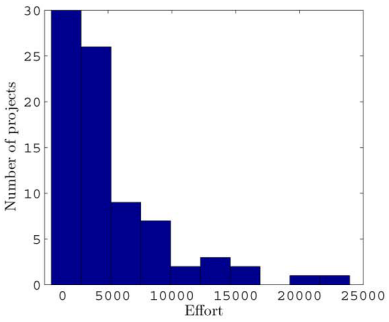
Number of observations: 90
Number of attributes: 11
Data set has not been used in previous studies
Reference: http://www.euroclear.com

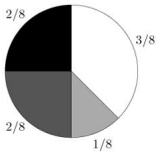




Desharnais
Skewness: 1.97
Kurtosis: 7.36
Minimum effort: 546 *Minimum FP: 62*
Mean effort: 5046 *Mean FP: 287*
Max effort: 23940 *Max FP: 1116*

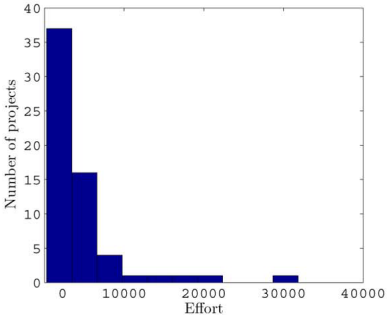
Number of observations: 81
Number of attributes: 9
Previously used in other studies, e.g. [20], [49], [64], [73]–[75]
Reference: [76]
http://www.promisedata.org/?p=9





Maxwell
Skewness: 3.27
Kurtosis: 15.52
Minimum effort: 583 *Minimum FP: 48*
Mean effort: 8223 *Mean FP: 673*
Max effort: 63694 *Max FP: 3643*

Number of observations: 62
Number of attributes: 23
Previously used in other studies, e.g. [13], [74]
Reference: [77]
http://www.promisedata.org/?p=108



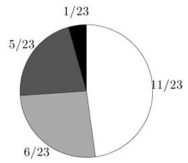


Figure 2: "Information on datasets[cont.]"

Table XII. Approaches Ranked as First.

Approach	LSD	MMRE	PRED(25)	MdMRE	MAE	MdAE
Cocomo81	RT	Bag+MLP	Bag+MLP	Bag+MLP	Bag+MLP	Bag+MLP
Sdr	RT	RT	Bag+RT	RT	RT	RBF
Nasa	Bag+RT	RT	Bag+MLP	Bag+MLP	Bag+RT	Bag+RT
Desharnais	Bag+RT	Bag+MLP	Pareto Ens	Pareto Ens	Pareto Ens	Pareto Ens
Nasa93	RT	RT	RT	RT	RT	RT
Org1	Bag+RBF	Pareto Ens	Pareto Ens	Pareto Ens	Pareto Ens	Pareto Ens
Org2	Bag+RT	Pareto Ens	Pareto Ens	Pareto Ens	Pareto Ens	Pareto Ens
Org3	Pareto Ens	Pareto Ens	Log+EBA	Log+EBA	Log+EBA	Log+EBA
Org4	Bag+RBF	Pareto Ens	RT	RT	Pareto Ens	Pareto Ens
Org5	Bag+RT	Log+EBA	Bag+RBF	Rand+MLP	Bag+RT	RT
Org6	Bag+RBF	Pareto Ens	Pareto Ens	Pareto Ens	Bag+RBF	Pareto Ens
Org7	Bag+RT	Log+EBA	Log+EBA	Log+EBA	Bag+RBF	Pareto Ens
OrgAll	RT	Pareto Ens	Pareto Ens	Pareto Ens	Pareto Ens	Pareto Ens

(a) Approaches per data set.

Figure 3: "Results for paper 4"

mining technique will be for it. It also observes that A MOEA can be used to produce and choose models that emphasize a particular measure without completely ignoring the performance using other measures, whereas the Pareto ensemble can be used as a good tradeoff among measures, if the software manager does not wish to emphasize any particular measure. The results of these paper embody what everyone believes about choosing the best technique for that particular data set or performance measure.

The fifth paper comments on use of local vs global data for prediction. The results of this paper suggest creation of a 'local team' that explores local data to make the best decision on this rather than making a fixed rule of what works best. Trite global rules are not sufficient for controlling such complex entities, like effort estimation, as concluded by the paper [8]

Paper six encounters large variances in results between different results of cross-validation. A jackknife, which is the cross-validation method used in the paper, procedure tends to produce a large variance for datasets with outliers, and therefore may not be suitable for SEE datasets.[18] The paper then argues that using ATLM as a baseline for these comparisons at least makes them more 'fair' to all techniques. An alternative solution suggested to this variation in performance is to perform many runs of cross-validation or training/test splits (many more than 30) to obtain a fair estimate of mean/variance in performance and to allow a meaningful statistical comparison.

Paper seven finds a number of conclusions on their study. They find that COSEEKMO can reduce both the standard deviation and the mean estimate error. Also, COSEEKMO was able to reduce variance in datasets like nasa93 which is the goal of the study. It also finds that there is tremendous variation in what treatment proved to the âĂIJbestâĂİ treatment. This pattern appears in the general data mining literature: Different summarization methods work best in different situations [19] and it remains unclear why that is so. This variation in the âĂIJbestâĂİ learner method is particularly pronounced in small data sets where minor quirks in the data can greatly influence learner performance.

The paper on negative results for SEE found results that the modeling effort associated with COCOMO-II could be reduced. Hence, it need not be expensive to deploy parametric estimation at some new site. Projects attributes do not need to be specified in great detail: a simple three point scale will suffice: below, nominal, above. As to how much data is required for modeling, a mere four to eight projects can suffice for calibration. Hence, it should be possible to quickly build many COCOMO-like models for various specialized sub-groups using just a three-point scale. This paper uses a simplistic approach with comparable/better results and Hence, in 2016, it is still a valid and a recommended practice to first try parametric estimation. This paper provides definitive results in comparison to most other studies and suggests consider that parametric methods while choosing a suitable technique for effort estimation. In conclusion, at least for effort estimation, how data is collected is more important than what learner is applied to that data. [10]

The eighth paper concludes that the Bayesian model is advantageous in a number of ways.Like, the Bayesian can be used to provide a point forecast for a software development effort. However, unlike most other forecasting models, the Bayesian model allows the managers to generate probability bounds on the forecast effort and combine managerial subjective estimates with the existing historical data to improve the software effort forecast. The Bayesian model is able to handle missing data and can be used to learn the causal relationships. Thus, the primary advantage of using the Bayesian model over other models is its capability of providing uncertainty in forecasted value and its capability of handling missing data[15] This result was a step over traditional methods in earlier stages and contributed to the growth of data mining techniques for software effort estimation.

6. CONCLUSION

The various papers presented in this study are an addition to the realm of good practices to improve data mining based effort estimation. The initial paper for this, paper one provides a comparison to large number of data mining techniques with a majority of regression based methods. This paper lends its way to further research by a number of stud-

ies to determine which data mining technique can be a clear winner for effort estimation.

The papers presented beyond are comparisons, or improve on the process to get better results. Paper two provides advantages of decision trees which clearly improve a large sector of data mining applications. Paper three offers feature selection and parameters optimization using genetic algorithm as a way to improve results. They also mention future work as particle swarm optimization (PSO), an alternative that may work better than GA. Another paper [13] also proves that use of GA combined with PSO gives better results and this may be a step ahead for regularly used regression methods for effort estimation when applied adequately.

The fourth paper talked about a new approach to SEE by seeing it as a multiobjective learning problem. It proved that considering different performance measures explicitly when creating a model may be used to produce good ensembles. It also discussed merits of MOEA, their proposed addition and proved it with positive results.

The growth of data mining in the domain has been clear from paper eight where using a bayesian network showed drastically better results in the study, to papers three, four, and so on where small tweaks and fine tuning are being done currently to improve the result.

Paper five debates another important aspect of the domain, the scope of the data itself. They do not provide a definitive result to which is better, local or global data, but rather provide a perspective on the choices and how to make that choice.

With the variations and multitude of techniques and enhancement, paper six was required to set an ideal baseline, there provided by ATLM, but also showing how a baseline could affect results. It provided one of the answers to the large variances in data mining results as well.

Paper seven and eight argue if all these complex models are really creating improvements by leaps and bounds compared to traditional methods that do the same job much more simplistically.

With all these options and adjustments and improvements possible, it does not answer necessarily how a user can choose one technique over the other for the best results. As a concise answer over multiple studies, we find that it just depends. There is no declared supreme method that can be the best in all situations. So, weighing different techniques for the particular application is very necessary. This is especially true for effort estimation where small changes can show big differences in the results, and also where accuracy is so necessary in the industry.

The future scope of many of these papers were interesting and many of them have also been made into studies of their own. Most of these papers present a small part of the whole process well but it would be insightful to create generalized paths on how and when to choose a particular data mining technique and also consider situations where data mining techniques may not be necessary at all.

7. REFERENCES

[1] Z. T. Abdulmehdi, M. S. Basha, M. Jameel, and P. Dhavachelvan. A variant of cocomo ii for improved software effort estimation. *International Journal of Computer and Electrical Engineering*, 6(4):346, 2014.

[2] A. S. Andreou and E. Papatheocharous. Software cost estimation using fuzzy decision trees. In *Automated Software Engineering, 2008. ASE 2008. 23rd IEEE/ACM International Conference on*, pages 371–374. IEEE, 2008.

[3] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens. Data mining techniques for software effort estimation: a comparative study. *IEEE Transactions on Software Engineering*, 38(2):375–397, 2012.

[4] A. Heiat. Comparison of artificial neural network and regression models for estimating software development effort. *Information and software Technology*, 44(15):911–922, 2002.

[5] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011.

[6] M. Jorgensen and M. Shepperd. A systematic review of software development cost estimation studies. *IEEE Transactions on software engineering*, 33(1):33–53, 2007.

[7] T. Jørgensen and S. W. Wallace. Improving project cost estimation by taking into account managerial flexibility. *European Journal of Operational Research*, 127(2):239–251, 2000.

[8] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmermann. Local versus global lessons for defect prediction and effort estimation. *IEEE Transactions on Software Engineering*, 39(6):822–834, 2013.

[9] T. Menzies, Z. Chen, J. Hihn, and K. Lum. Selecting best practices for effort estimation. *IEEE Transactions on Software Engineering*, 32(11):883–895, 2006.

[10] T. Menzies, Y. Yang, G. Mathew, B. Boehm, and J. Hihn. Negative results for software effort estimation. *Empirical Software Engineering*, pages 1–26, 2016.

[11] L. L. Minku and X. Yao. Software effort estimation as a multiobjective learning problem. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 22(4):35, 2013.

[12] J. Moeyersoms, E. J. de Fortuny, K. Dejaeger, B. Baesens, and D. Martens. Comprehensible software fault and effort prediction: A data mining approach. *Journal of Systems and Software*, 100:80 – 90, 2015.

[13] M. H. Moradi and M. Abedini. A combination of genetic algorithm and particle swarm optimization for optimal dg location and sizing in distribution systems. *International Journal of Electrical Power & Energy Systems*, 34(1):66–74, 2012.

[14] A. L. Oliveira, P. L. Braga, R. M. Lima, and M. L. Cornélio. Ga-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. *information and Software Technology*, 52(11):1155–1166, 2010.

[15] P. C. Pendharkar, G. H. Subramanian, and J. A. Rodger. A probabilistic model for predicting software development effort. *IEEE Transactions on Software Engineering*, 31(7):615–624, 2005.

[16] C. Van Koten and A. Gray. Bayesian statistical effort prediction models for data-centred 4gl software

development. *Information and Software Technology*, 48(11):1056–1067, 2006.

[17] V. Wable and S. Shinde. Ranking and clustering of software cost estimation models. *International Journal of Science and Research*, 3(7), 2014.

[18] P. A. Whigham, C. A. Owen, and S. G. Macdonell. A baseline model for software effort estimation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(3):20, 2015.

[19] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.