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.[6] 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.[4][7] 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.[5]. 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 an overview of the literature concerning software effort estimation. Then, in Section 3 the different techniques employed in the study are discussed. Section 4 reflects upon the data sets, evaluation criteria, and the statistical validation of the study. In Section 5, we apply the techniques as well as the generic backward input selection schema and discuss the results. The paper is concluded by Section 6 providing general conclusions and topics for future research.

2. 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. Paper 1[1] 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.
- Robust regression.
- Ridge regression.
- Least median squares regression.
- MARS.
- CART.
- Model tree.
- Multilayered perceptron neural network.

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

Table 1: Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
\$	4 in 5	Used in business
Ψ_1^2	1 in 40,000	Unexplained usage

- 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.

Paper 3[6] 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. Paper 2[2] 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.

Paper

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 try to build, improve or compare with regression model-based estimation methods.[3]

3. RESULTS

We say this since the experiments of this paper show that, at least for effort estimation, how data is collected is more important than what learner is applied to that data

3.1 Tables

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of tables, use the environment **table** to enclose the table's contents and the table caption. The contents of the table itself must go in the **tabular** environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules. Again, detailed instructions on **tabular** material is found in the *LaTeX User's Guide*.

Immediately following this sentence is the point at which Table 1 is included in the input file; compare the placement of the table here with the table in the printed dvi output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 2 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed dvi output of this document.

Figure 1: A sample black and white graphic.

Figure 2: A sample black and white graphic that has been resized with the `includegraphics` command.

3.2 Figures

Like tables, figures cannot be split across pages; the best placement for them is typically the top or the bottom of the page nearest their initial cite. To ensure this proper "floating" placement of figures, use the environment **figure** to enclose the figure and its caption.

This sample document contains examples of **.eps** files to be displayable with LaTeX. If you work with pdfLaTeX, use files in the **.pdf** format. Note that most modern TeX system will convert **.eps** to **.pdf** for you on the fly. More details on each of these is found in the *Author's Guide*.

As was the case with tables, you may want a figure that spans two columns. To do this, and still to ensure proper "floating" placement of tables, use the environment **figure*** to enclose the figure and its caption. and don't forget to end the environment with figure*, not figure!

3.3 Theorem-like Constructs

Other common constructs that may occur in your article are the forms for logical constructs like theorems, axioms, corollaries and proofs. There are two forms, one produced by the command `\newtheorem` and the other by the command `\newdef`; perhaps the clearest and easiest way to distinguish them is to compare the two in the output of this sample document:

This uses the **theorem** environment, created by the `\newtheorem` command:

THEOREM 1. *Let f be continuous on $[a, b]$. If G is an antiderivative for f on $[a, b]$, then*

$$\int_a^b f(t)dt = G(b) - G(a).$$

The other uses the **definition** environment, created by the `\newdef` command:

Definition 1. If z is irrational, then by e^z we mean the unique number which has logarithm z:

$$\log e^z = z$$

Two lists of constructs that use one of these forms is given in the *Author's Guidelines*.

There is one other similar construct environment, which is already set up for you; i.e. you must *not* use a `\newdef` command to create it: the **proof** environment. Here is a example of its use:

Table 2: Some Typical Commands

Command	A Number	Comments
`\alignauthor`	100	Author alignment
`\numberofauthors`	200	Author enumeration
`\table`	300	For tables
`\table*`	400	For wider tables

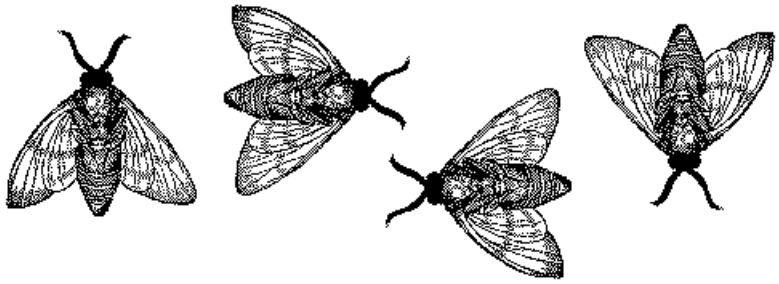

Figure 3: A sample black and white graphic that needs to span two columns of text.

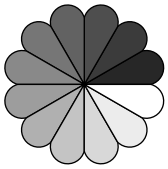

Figure 4: A sample black and white graphic that has been resized with the `includegraphics` command.

PROOF. Suppose on the contrary there exists a real number L such that

$$\lim_{x\to\infty}\frac{f(x)}{g(x)}=L.$$

Then

$$l=\lim_{x\to c}f(x)=\lim_{x\to c}\left[gx\cdot\frac{f(x)}{g(x)}\right]=\lim_{x\to c}g(x)\cdot\lim_{x\to c}\frac{f(x)}{g(x)}=0\cdot L=0,$$

which contradicts our assumption that $l\neq 0$. □

Complete rules about using these environments and using the two different creation commands are in the *Author's Guide*; please consult it for more detailed instructions. If you need to use another construct, not listed therein, which you want to have the same formatting as the Theorem or the Definition[**?**] shown above, use the `\newtheorem` or the `\newdef` command, respectively, to create it.

A *Caveat* for the TEX Expert

Because you have just been given permission to use the `\newdef` command to create a new form, you might think you can use TEX's `\def` to create a new command: *Please refrain from doing this!* Remember that your LATEX source code is primarily intended to create camera-ready copy, but may be converted to other forms – e.g. HTML. If you inadvertently omit some or all of the `\def`s recompilation will be, to say the least, problematic.

4. CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LATEX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

5. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

6. REFERENCES

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

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