Towards Logistic Regression Models for Predicting Fault-prone Code across Software Projects

Ana Erika Camargo Cruz, Koichiro Ochimizu
Japan Institute of Science and Technology
School of Information Science
1-1 Asahidai, Nomi, Ishikawa
erika.camargo@jaist.ac.jp, ochimizu@jaist.ac.jp

Abstract

In this paper, we discuss the challenge of making logistic regression models able to predict fault-prone object-oriented classes across software projects. Several studies have obtained successful results in using design-complexity metrics for such a purpose. However, our data exploration indicates that the distribution of these metrics varies from project to project, making the task of predicting across projects difficult to achieve. As a first attempt to solve this problem, we employed simple log transformations for making design-complexity measures more comparable among projects. We found these transformations useful in projects which data is not as spread as the data used for building the prediction model.

1. Introduction

There exist several research studies which have shown that design-complexity metrics, while measured from the code, are good predictors of fault-prone object-oriented classes [2, 4, 11] using Logistic Regression (LR) models. However, these models are based on design-complexity metrics, which lack genuine thresholds [9]. Therefore, it might be possible that LR models, built with metrics of a specific project, will only work efficiently with other projects of similar characteristics.

In order to enhance the usability of prediction LR models across softwares, we focused on the analysis and solution of the reduction of the gap which might exist between the levels of the design-complexity metrics across different software projects. As a possible solution, we studied the application of data transformation to design-complexity measures. If we can normalize these measures by using an appropriate data transformation, then we might be able to use code-based prediction models across software projects.

This paper is structured as follows: In section 2, we discuss the usability of LR models across different software projects. In section 3, we present an experimental study to test the performance of univariate LR models with two different projects, using raw values and data transformations. Finally, in section 4, we draw our conclusions and our plans for future work.

2. Problem Analysis

We have mentioned that there are several studies using LR models to predict faulty classes, given design-complexity metrics measured from the code. Some of the most used metrics, and better for predicting fault-proneness, are the Chidamber and Kemerer metrics (CK).

The CK metrics were designed to measure unique aspects of the object-oriented approach and complexity of the design. Thresholds of these metrics cannot be determined before their use, they should be derived and used locally for each dataset (each project). Usually, it is recommended to use the 80th and 20th percentiles of the distributions to determine high and low values of the metrics [5, 12, 8].

Therefore, it might be that that prediction models of fault-proneness based on this kind of metrics should also be used locally, and they can not be used in different projects. Let us say that we have a LR model which considers the number of methods per class (NOM) as predictor of fault-proneness. And it was built using data of a large-size software project *A*, where the most faulty classes are those with $NOM >= 22$. If we would like to use the same prediction model with a much smaller project *B*, with a maximum $NOM = 20$, then we would fail to find the most faulty classes of B, as illustrated in Figure 1.

In the next sections, we present three univariate LR models, using RFC, CBO and WMC CK metrics of a large-size software project. We show the results of testing each of these models with other software projects.

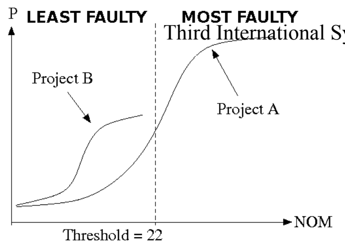

Figure 1. One prediction model - two projects

3. Case Study

For the construction of the models, we used the code of the first released version of the Mylyn software, which is a task-focused interface for Eclipse and is an open-source project written in Java by six developers.

We obtained the number of faults per class of the Mylyn system using the information from its bugzilla database[1] and parsed CVS logs per class. Since LR uses a dichotomous variable as a dependent variable, we chose to classify our classes as Most Faulty and Least Faulty. Classes with a number of faults greater or equal to the median of the entire dataset of faults of the system are Most Faulty, otherwise they are Least Faulty. From a total of 638 classes, 419 were classified as Most Faulty and 219 as Least Faulty.

Furthermore, the independent variables used were the RFC, CBO and WMC metrics, which were measured using the CKJM tool [2] to build the three univariate LR models.

In the next sections, we provide the details of the construction of the models, including information of the data transformation used as an attempt to improve the results obtained with these models.

3.1. Data Analysis

Using boxplots, we analyze the CBO, RFC and WMC measures within various projects. From a boxplot, we can have at a glance important characteristics of our data such as: location, spread, skewness, tail length and outlying data points [7]. A boxplot mainly represents the interquartile range (IQR), which is the difference between the 75th and 25th percentiles. The values of these percentiles are the edges of the box and are called upper hinge and lower hinge respectively. Data points 1.5 times the IQR above the upper hinge or below the lower hinge are considered potential outliers. The presence of significant outliers produces biased regression estimates and reduces the predictive power of regression models [6].

There are seven projects whose measures will be compared: The Mylyn system (MYL) previously mentioned, another large open-source project from Eclipse named Graphic Modeling Framework (GMF) with 1523 Java classes, and five small size projects. These are an e-commerce system (ECS), a banking system (BNS), a cruise control and monitoring system (CRS), a distributed factory automation system (FACS) and an elevator control system (ELCS), which were also written in Java by different teams of 2 or 3 first-year-master students[3]. The number of classes for these systems varies from 11 to 84.

Due to space limitations, we are only showing, in Figure 2, the boxplots for the CBO measures in the mentioned projects. Because the median values are connected by a doted line, we can easily observe that there is a tendency for spread to increase as level (location) does. This was particularly observed in the CBO measures. Moreover, a large number of outliers and more dispersed data were found in the largest projects.

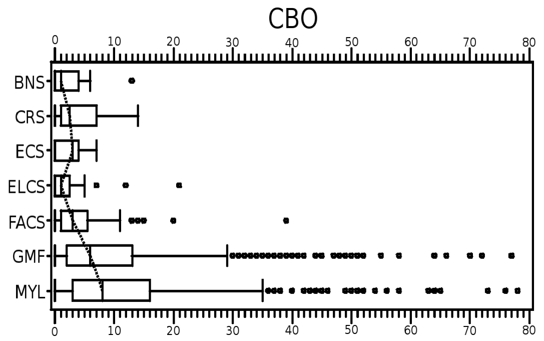

Figure 2. CBO Boxplots

3.2. Data Transformation

Transformations in general enable us to re-express data in a new scale to promote equality of spread among the batches, and thus decrease variability. They also improve symmetry and normality and reduce the number of outliers. Furthermore, when there is a relationship between spread and level, *power transformations* can be used to reduce or eliminate this dependence [7].

Power transformations are defined as the transformation that replaces x by x^p. If $p = 0$ then $logx$ is used instead. At this time, we chose to work with $p = 0$. Because we might obtain zero values which would result on infinitive values using the logarithmic function, we applied a simple and common solution, transforming as follows: $x' = log(x + 1)$ [10]. Additionally, we aligned all of the batches to a common reference point: The median of the MYL system, which design-complexity measures were used to build the LR models. The final transformation used is $x' = log(x + 1) + b$, where: x is a particular value of

[1]It is an open-source/free software bug tracking system.

[2]It calculates CK metrics by processing the byte-code of compiled Java files (by Diomidis Spinellis)

[3]These systems were developed during the lecture "Software Design Laboratory" offered by The Japan Institute of Science and Technology

Table 1. Estimated Coefficients for UL models

x	A	B
CBO	-0.5604 | -1.80	0.1329 | 2.8487
RFC	-0.5257 | -3.01	0.0491 | 2.90
WMC	-0.0935 | -1.27	0.0973 | 2.31

CBO, RFC or WMC, x' is the transformed value of x, and b is the difference between the median of the transformed dataset of the MYL system, and the median of the dataset to be transformed. In the MYL system, the median values for the CBO, RFC and WMC transformed datasets are 0.95, 1.34 and 0.85 respectively.

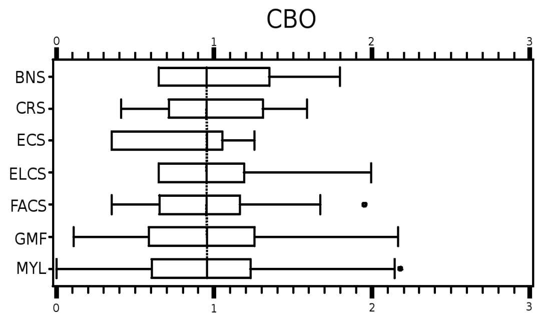

Figure 3. CBO Boxplots after transformations

We are only showing the resultant boxplots for CBO in Figure 3. The RFC and WMC boxplots had a similar effect to the CBO boxplots, where a large number of outliers have disappeared and spread among the batches resulted less variable.

We expect that these transformations contribute to a better performance of our LR models in detecting fault-prone classes in different software projects.

3.3. Logistic regression models

The estimated coefficients for each built univariate LR model (see Equation 1) are shown in Table 1. The x column contains the independent variable of the model, the A column refers to the intercept values and the B column to the regression coefficients of the independent variable. While coefficients on the left side of columns A and B are those for the models using raw values, the coefficients on the right are for the models using data transformations.

$$P = \frac{1}{1 + e^{-(A+Bx)}} \tag{1}$$

3.4. Validation

We present the necessary steps to assess the appropriateness, adequacy and usefulness of every model. First, we assesses the importance of each of the variables in every model by carrying out statistical tests of significance. Second, the overall goodness of fit of every model is tested. Third, we measure their ability to discriminate between the Most Faulty (MF) and Least Faulty (LF) classes. Finally, we validate them using other projects [3].

1) Importance of each independent variable. For this, we use the *-2LogL* statistic, which has a χ^2 (chi square) distribution with 1 degree of freedom (*df*). If we choose a significance level[4] (α) of 0.05 then $-2LogL > \chi^2_{1,0.05} = 3.8$ is desirable. The resultant *-2LogL* values of our models suggest that the three variables are significantly important for its correspondent model to fit the data, using raw and transformed values.

2) Overall goodness of fit. For testing this, we use the Hosmer-Lemeshow (*HL*) test. This statistic has a χ^2 distribution with 8 *df*. Large values of *HL* indicate that the fit may not be good [1]. Thus, considering an $\alpha = 0.05$, $HL < \chi^2_{8,0.05} = 16$ is desirable. The resultant *HL* values of our models suggest that for the models using raw values the overall fit may not be good, while the models using the data transformation yield a very good overall fit.

3) Discrimination. We need to measure the ability of the models to discriminate between the MF and LF classes. For this, we used Hubert's statistical procedure [13]. The following statistics are required: e is the expected number of correct classifications due to chance, O is the number of correct classifications, and Z statistic, which follows a normal distribution, and if its value is significant at an $\alpha = 0.05$, it suggests that the number of correct classifications is significantly greater than that obtained due to chance.

The values of e are 275.71, 75.17 and 350.34 for the groups MF, LF and for the entire dataset respectively. The resultant values of O are greater than those of e, for all of the models, which indicate a good discrimination between the MF and the LF classes. And the resultant Z values suggest that the number of correct classifications is significantly greater than that obtained due to chance.

The cutoff probabilities to discriminate between MF and LF classes are 0.5, 0.5 and 0.56 for CBO, RFC and WMC respectively. This selection was based on the best discrimination results obtained from the models using raw values.

Notice that although the models using raw values discriminate well, they did not do well in the goodness of fit test, according to the HL statistic, which considers all estimated probabilities of the predicted outcomes, and tells us how close predicted values are to the observed values.

4) Validation using different projects. Finally our models are tested using other projects different from the MYL. These are the ECS and BNS projects described previously.

[4]It is defined as the probability of making a decision to REJECT the null hypothesis. If a test of significance gives a p-value lower than the α-level, the null hypothesis is rejected.

Table 2. Validation using the BNS

Group	x	e	O	Z	DTx
MF 6 classes	CBO	3.27	2 | 5	-1.04 | 1.41	⇑
	RFC	3.27	5 | 5	1.41 | 1.41	=
	WMC	3.27	6 | 6	2.23 | 2.23	=
LF 5 classes	CBO	2.27	5 | 5	2.44 | 2.44	=
	RFC	2.27	3 | 3	0.65 | 0.65	=
	WMC	2.27	4 | 4	1.55 | 1.55	=
Both 11 classes	CBO	5.54	7 | 10	0.87 | 2.68	⇑
	RFC	5.54	8 | 8	1.48 | 1.48	=
	WMC	5.54	10 | 10	2.68 | 2.68	=

⇑ : Improvement, ⇓ : Decay, = : No change
DTx : Effect After Data Transformation

Table 3. Validation using the ECS

Group	x	e	O	Z	DTx
MF 9 classes	CBO	6.2	3 | 7	-2.3 | 0.5	⇑
	RFC	6.2	9 | 8	2 | 1.27	⇓
	WMC	6.2	7 | 6	0.55 | -0.16	⇓
LF 4 classes	CBO	1.2	4 | 4	3 | 3	=
	RFC	1.2	0 | 3	-1.3 | 1.91	⇑
	WMC	1.2	0 | 4	-1.3 | 3	⇑
Both 13 classes	CBO	7.5	7 | 11	-0.2 | 1.98	⇑
	RFC	7.5	9 | 11	0.86 | 1.98	⇑
	WMC	7.5	7 | 10	-0.25 | 1.42	⇑

⇑ : Improvement, ⇓ : Decay, = : No change
DTx : Effect After Data Transformation

Tables 2 and 3 show the resultant values of e, Z and O of our three models for the groups: MF, LF and for the entire dataset, using the same convention described for Table 1.

For the BNS project, the CBO model using the data transformation yields slightly improved results as compared to the model using simply raw values. As for the RFC and WMC models, the results remain the same.

On the other hand, for the ECS project, models using the data transformation yield better results than those models using raw values. Although the RFC and the WMC models using the data transformation have decreased slightly their discrimination ability at detecting the MF classes, they can discriminate the LF classes, while models based on raw values cannot.

In general, log transformations improved the results of our prediction models in projects, which measures were not as spread as those used in the construction of the models. As it occurred in BNS-CBO, ECS-CBO, ECS-RFC and ECS-WMC datasets.

4. Conclusion

In this paper, we have discussed the problem of making LR models that use design-complexity metrics to predict fault-prone object-oriented classes, usable across different software projects. As a first attempt to solve this problem, we studied the application of a data transformation for building and using LR models. We built three univariate LR models, using CBO, RFC and WMC CK measures of a large-size software project. And we tested these models with other two small-size software projects. Our results indicate that log transformations can be useful when measures are not as spread as those measures used in the construction of the model. The findings exposed in this paper are limited to the number of software projects used in our experiments. However, they encourage us to conduct further data exploration and to extend our knowledge of data transformations.

References

[1] A. Afifi, V. Clark, and S. May. *Computer-Aided Multivariate Analysis*. Chapman & Hall/CRC, USA, 2004.

[2] V. R. Basili, L. C. Briand, and W. L. Melo. A validation of object-oriented design metrics as quality indicators. *IEEE Trans. Softw. Eng.*, 22(10):751–761, 1996.

[3] V. Bewick, L. Cheek, and J. Ball. Statistics review 14: Logistic regression. *Critical Care*, 9(1), 2005.

[4] L. C. Briand, J. Wüst, J. W. Daly, and D. V. Porter. Exploring the relationship between design measures and software quality in object-oriented systems. *J. Syst. Softw.*, 51(3):245–273, 2000.

[5] S. R. Chidamber, D. P. Darcy, and C. F. Kemerer. Managerial use of metrics for object-oriented software: An exploratory analysis. *IEEE Trans. Softw. Eng.*, 24(8):629–639, 1998.

[6] G. Fernández. *Data mining using SAS applications*. CRC Press, Inc., Boca Raton, FL, USA, 2002.

[7] D. C. Hoaglin, F. Mosteller, and J. W. Tukey. *Understanading Robust and Exploratory Data Analysis*. John Wiley & Sons, Inc., USA, 2000.

[8] S. H. Kan. *Metrics and Models in Software Quality Engineering*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[9] E.-E. Khaled. Object-oriented metrics: A review of theory and practice. *Advances in software engineering*, pages 23–50, 2002.

[10] F. Mosteller and J. W. Tukey. *Data Analysis and Regression*. Addison-Wesley Publishing Company, USA, 1977.

[11] H. M. Olague, S. Gholston, and S. Quattlebaum. Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. *IEEE Trans. Softw. Eng.*, 33(6):402–419, 2007. Senior Member-Letha H. Etzkorn.

[12] L. Rosenberg. Applying and interpreting object oriented metrics. In *Software Assurance Technology Conference*.

[13] S. Sharma. *Applied Multivariate Techniques*. Addison-Wiley & Sons, Inc., USA, 1996.