Comparing Efficiency of Software Fault Prediction Models Developed Through Binary and Multinomial Logistic Regression Techniques

Dipti Kumari and Kumar Rajnish

Abstract Software fault prediction method used to improve the quality of software. Defective module leads to decrease the customer satisfaction and improve cost. Software fault prediction technique implies a good investment in better design in future systems to avoid building an error prone modules. The study used software metrics effectiveness in developing models in 2 aspects (binary and multinomial) Logistic Regression. We are developing multivariate (combined effect of object-oriented metrics) models in both aspects for finding the classes in different error categories for the three versions of Eclipse, the Java-based open-source Integrated Development Environment. The distribution of bugs among individual parts of a software system is not uniform, in that case Multinomial aspects helps the tester to prioritize the tests with the knowledge of error range or category and therefore, work more efficiently. Multinomial models are showing better result than Binary models.

Keywords OO-metrics · Fault-prone · Prediction · Logistic regression · Multinomial · LR · BMLR · MMLR

1 Introduction

Over the past years, software quality has become one of the most important requirements in the development of systems. Fault-proneness estimation could play a key role in quality control of software products [1]. To maintain this quality and develop fault free software, almost in every organization that is involved in the

D. Kumari (✉) · K. Rajnish
Department of Computer Science and Engineering, BIT Mesra, Ranchi 835215, Jharkhand, India
e-mail: Kumari_dipti0511@yahoo.co.in

K. Rajnish
e-mail: krajnish@bitmesra.ac.in

J.K. Mandal et al. (eds.), *Information Systems Design and Intelligent Applications*, Advances in Intelligent Systems and Computing 339,
DOI 10.1007/978-81-322-2250-7_19

software development there are various activities that have to be performed. Such an activity is testing. Software testing is a critical and essential part of software development that consumes maximum resources and effort. Software testing almost takes at least half of the resources and still doesn't assure the 100 % correctness of the system. Also every part of the software seems impossible to test. That's why everyone wants to focus or test only those parts of the software those have high probability to be fault-prone. The aim of various researches is to find such parts of system. Such parts includes classes, methods, inheritance paths etc. Software fault prediction is one of the quality assurance activities in Software Quality Engineering such as formal verification, fault tolerance, inspection, and testing. Software metrics [2, 3] and fault data (faulty or non-faulty information) belonging to a previous software version are used to build the prediction model. The fault prediction process usually includes two consecutive steps: training and prediction. In the training phase, a prediction model is built with previous software metrics (class or method-level metrics) and fault data belonging to each software module. After this phase, this model is used to predict the fault proneness labels of modules that locate in a new software version [4].

Until now, software engineering researchers have used Case-based Reasoning, Neural Networks, Genetic Programming, Fuzzy Logic, Decision Trees, Naive Bayes, Dempster-Shafer Networks, Artificial Immune Systems, and several statistical methods to build a robust software fault prediction model [5–8]. Some researchers have applied different software metrics to build a better prediction model, but recent papers [9] have shown that the prediction technique is much more important than the chosen metric set. In this paper, we attempt to develop binary and Multinomial LR models to estimates the fault-proneness in object oriented environment. We organised this paper as: Sect. 2 provides the Research Background and Research Methodology, Sect. 3 provides Model Evaluation, Sect. 4 provides Analysis of result. Section 5 provides the Conclusion and Sect. 6 provides Future scope.

2 Research Background

In this section, we present the selection of data source (Sect. 2.1), selection of metrics in this article (Sect. 2.2), Collection of Fault data and its categorization (Sect. 2.3) and Research methodology (Sect. 2.4).

2.1 Selection of Data Source

This study makes use of the data collected from three major releases of Eclipse (Eclipse2.0, Eclipse2.1 and Eclipse3.0). We select Eclipse2.0, 2.1, and 3.0 as the subjects of our study for two reasons: First, their fault data are publicly available

(Therefore, it is easy to externally validate our empirical results by other researchers. Second, they are major releases of Eclipse and have been widely used for several years.

2.2 Selection of Independent Variables

The selection of software metrics was a difficult task because there are many available metrics. We used two criteria in our selection process:

The set of metrics cover all aspects of OO design.
We have to be able to collect the metrics by using automated tool.

These metrics are characterized into coupling, cohesion, inheritance, class complexity and class-size metrics. We used JHAWK [10] automated tool metric to collect these metrics from the Eclipse source code [11]. JHAWK compiled the source code and give output as each module name and their set of OO metrics. These OO-metrics are independent variables.

2.3 Collection of Dependent Variable

The binary and multinomial dependent variable in this study is fault proneness. Fault proneness is defined as the probability of fault detection in a class. We collected the fault data from three releases of Eclipse (Versions 2.0, 2.1, and, 3.0) provided by the publicly available data set promise2.0a [12–14]. This data set lists the number of pre-release faults (reported in the first 6 months after release) for each java file in Eclipse2.0, Eclipse2.1 and Eclipse3.0. Pre release bug data are used for study and two types of categorization has been done on the pre release error data:

1. Binary Categorization: In this we only used two values 0 (means no error) and 1 (means with error). If a class contains error in it then we put 1 in error column otherwise 0.
2. Multinomial Categorization: In this we divide the error severity into 4 classes.

For classification our followed steps are as follows:

- We find the descriptive statistics of pre error data. From that we are able to know the min, different number of occurrences of error (nonzero) and max value of error data in all classes of every versions of Eclipse.
- After that, we again find the descriptive statistics of (Min, 25, 50, 75 % and Max) the different occurrences of number of errors (from min (nonzero) to max). Based on that we classified class error data into one of five categories that are defined as follows:
- No Error: class containing zero error

- Nominal: class containing error in the range Min ≤ error < 25 %
- Low: class containing error in the range 25 % ≤ error < 50 %
- Medium: class containing error in the range 50 % ≤ error < 75 %
- High: class containing error in the range 75 % ≤ error < Max

More specifically, we attempt to answer the following questions by appropriate statistical analysis technique:

- How accurate do the investigated metrics distinguish between fault-prone and not fault-prone classes (binary categorization)?
- How accurate do the investigated metrics classifies classes into four categories: Nominal, Low, Medium, and High based on the error severity level (multinomial categorization)?

2.4 Research Methodology

In this section, we describe logistic regression (LR) analysis in binary and multinomial aspects. LR is the most widely used technique in literature. It is used here to predict dependent variable from a set of independent variables (a detailed description is given by [15, 16]). Binary LR is used to construct models when the dependent variable is binary and Multinomial LR is used to construct models when the dependent variable is not binary but having more than two values.

2.4.1 Binary Logistic Regression Model

Binary Logistic regression is a standard statistical modeling method in which the dependent variable Y can take on only one of two different values [17]. Assume that X_1, X_2, …, X_n represents the independent variables (i.e. the metrics in this study) and $Pr(Y = 1|x_1, x_2, \ldots, x_n)$ represents the probability that $Y = 1$ when $X_1 = x_1$, $X_2 = x_2$, …, and $X_n = x_n$. Then, the logistic regression model assumes that $Pr(Y = 1|x_1, x_2, \ldots, x_n)$ is related to x_1, x_2, …, x_n by the following equation:

$$\Pr(Y = 1|x_1, x_2, \ldots, x_n) = \frac{e^{\alpha+\beta_1 x_1+\cdots+\beta_n x_n}}{1 + e^{\alpha+\beta_1 x_1+\cdots+\beta_n x_n}} \quad (1)$$

where β is are the regression coefficients and can be estimated through the maximization of a log-likelihood.

2.4.2 Multinomial Logistic Regression

Multinomial Logistic regression is modified form of binary logistic regression, it is appropriate when the outcome is a polytomous variable (i.e. categorical with more than two categories) and the predictors are of any type. Y can take on more than two different values depending on the no. of different categories [17]. In the following, let the values be 0, 1, 2, 3 and 4. Here, Y = 1, 2, 3 and 4 represents the corresponding class have fault according to the nominal category, low category, mid category and high category and Y = 0 represents the corresponding class have no fault. As in other forms of linear regression, multinomial logistic regression uses a linear predictor function $f(k, i)$ to predict the probability that observation i has outcome k, of the following form:

$$f(k,i) = \beta_{0,k} + \beta_{1,k} \times x_{1,i} + \beta_{2,k} \times x_{1,i} + \cdots + \beta_{M,k} \times x_{M,i} \tag{2}$$

where $\beta_{M,k}$ is a regression coefficient associated with the mth explanatory variable and the kth outcome.

3 Model Evaluation Criteria

In the literature, many other measures have been proposed for evaluating the predictive effectiveness of classification models such as logistic regression models [18–20]. Accuracy of the model is considered as the comparison factor with the earlier traditional models and may be obtained using Confusion Matrix [21]. A confusion matrix contains information about actual and predicted classifications done by a classification system. TP and TN are the number of correct predictions that an instance is positive and negative respectively. FP and FN are the number of incorrect predictions that an instance is positive and negative respectively.

- Accuracy: the number of classes that are correctly classified divided by the total number of classes. Where TP and TN are the number of correct predictions that an instance is positive and negative respectively. FP and FN are the number of incorrect predictions that an instance is positive and negative respectively. For binary categorization

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

  For Multinomial categorization

$$Accuracy = \frac{TP + TN_i}{TP + TN_i + FP + FN_i} \tag{4}$$

where TP = T_0, TN_i = T_1, T_2, T_3 and T_4, $FP = \sum_{j=1}^{4} F_{0j}\, FN_i = \sum_{i=1}^{4} \sum_{j=1}^{4} F_{ij}$ where $i \neq j$.

Where T_0, T_1, T_2, T_3 and T_4 are the number of correct predictions that an instance is positive, negative in nominal category, low category, mid category and high category. Where F_{01} are the number of incorrect predictions that an instance is actually in positive but predicted in nominal category. In F_{ij} first subscript showing the actual instance and second one is showing the predicted result.

- The general rule to evaluate the classification performance is to find the area under the curve (AUC): AUC = 0.5 means no good classification; 0.5 < AUC < 0.6 means poor classification; 0.6 ≤ AUC < 0.7 means fair classification; 0.7 ≤ AUC < 0.8 means acceptable classification; 0.8 ≤ AUC < 0.9 means excellent classification; AUC ≥ 0.9 means outstanding classification.

4 Experimental Analysis

We have developed binary and multinomial model by taking the combined effect of OO-metrics to identify faulty classes. The model is built using backward elimination in the model and model statistics are shown in Tables 1 and 2 for all 3 versions of Eclipse in binary and multinomial aspects respectively. From table we find that Multinomial categorization are showing good result compare to Binary Categorization.

4.1 Validation Result

Tables 3 and 4 summarizes the TP, TN, FP, FN, Sensitivity, (1-Specificity), AUC and Accuracy results from 18-fold cross-validation for the Multivariate models for binary and multinomial aspect respectively for all 3 version of Eclipse. Accuracy result for nominal category is 58–65 % and AUC comes in poor category, but accuracy for low, mid and high category is in 50–60 % and their classification power comes in fair and acceptable classification. PACK is showing best result among all metrics for Eclipse2.0 in MBLR model. Accuracy of nominal category is high compare to other but its discriminating power is in poor class. Other 3 categories have accuracy 69–71 % and low and part of mid comes in acceptable class, but high comes is excellent classification for Eclipse2.1 in MBLR model. PACK is showing the best result among all. Same incidence is found in Eclipse3.0 for MBLR

Table 1 Multivariate model statistics for binary categorization

	Eclipse2.0				Eclipse2.1				Eclipse3.0			
	Metric	B	S.E	Sig	Metric	B	S.E	Sig	Metric	B	S.E	Sig
Nominal	NOM	0.014	0.006	0	AVCC	0.137	0.03	0	CC	0.011	0.002	0
	AVCC	0.136	0.033	0.027	NOS	0.003	0.002	0.026	MPC	−0.017	0.005	0
	NOS	0.004	0.001	0.004	PACK	0.067	0.005	0	FOUT	0.044	0.013	0.001
	PACK	0.063	0.005	0	NLOC	−0.003	0.001	0.002	RFC	0.014	0.004	0.002
	CBO	0.018	0.008	0.025	TCC	0.009	0.004	0.021	PACK	0.056	0.004	0
	NLOC	−0.002	0.001	0.014	Constant	−2.167	0.064	0	AVCC	0.115	0.026	0
	FOUT	0.076	0.016	0					Constant	−2.113	0.056	0
	Constant	−1.5	0.062	0								
Low	NOS	0.01	0.004	0.018	PACK	0.079	0.008	0	CC	0.019	0.004	0
	PACK	0.115	0.009	0	RFC	0.012	0.004	0.004	LMC	−0.044	0.018	0.016
	NLOC	-0.009	0.003	0.005	CBO	0.037	0.018	0.035	PACK	0.066	0.007	0
	LMC	0.041	0.02	0.044	Constant	−5.881	0.224	0	Constant	−6.541	0.267	0
	MAXCC	0.04	0.014	0 006								
	Constant	−5.217	0.268	0								
Mid	PACK	0.124	0.013	0	PACK	0.071	0.011	0	CBO	0.084	0.025	0.001
	CBO	0.079	0.022	0	RFC	0.024	0.007	0	PACK	0.085	0.009	0
	NLOC	−0.017	0.007	0.02	CBO	0.069	0.022	0.002	AVCC	0.279	0.127	0.028
	TCC	0.039	0.019	0.045	Constant	−8.1	0.513	0	Constant	−8.762	0.501	0
	Constant	−6.861	0.703	0								
High	PACK	0.135	0.018	0	NOM	−0.241	0.121	0.047	CC	0.024	0.007	0.001
	CBO	0.099	0.036	0.007	NOS	0.041	0.017	0.017	PACK	0.089	0.011	0
	LMC	0.112	0.044	0.01	PACK	0.076	0.028	0.007	Constant	−9.991	0.902	0
	FOUT	0.102	0.05	0.042	RFC	0.061	0.021	0.005				
	Constant	−8.983	1.056	0	NLOC	−0.035	0.015	0.018				
					Constant	−7.364	1.334	0				

Table 2 Multivariate model statistics for multinomial categorization

Eclipse2.0				Eclipse2.1				Eclipse3.0			
Metric	B	S.E	Sig	Metric	B	S.E	Sig	Metric	B	S.E	Sig
NOS	0.003	0.001	0.012	UWCS	−0.007	0.004	0.045	AVCC	0.122	0.02	0
PACK	0.064	0.004	0	RFC	0.006	0.002	0.004	UWCS	0.011	0.002	0
CBO	0.019	0.008	0.017	CC	0.01	0.004	0.007	PACK	0.056	0.004	0
FOUT	0.073	0.015	0	MAXCC	0.025	0.007	0	FOUT	0.041	0.013	0.001
AVCC	0.115	0.023	0	NLOC	−0.003	0.001	0.003	TCC	0.007	0.002	0
NLOC	−0.002	0.001	0.011	PACK	0.068	0.004	0	CBO	0.015	0.007	0.023
Constant	−1.443	0.057	0	Const	−1.966	0.047	0	Const	−2.085	0.051	0

Table 3 Evaluation result of the performance of binary multivariate model for all 3 version of eclipse

Version	TP	TN	FP	FN	Sensitivity	1-Specificity	AUC	Accuracy
Eclipse2.0	3,649	879	437	1,683	0.34	0.11	0.62	68.11
Eclipse2.1	5,386	595	309	1,507	0.28	0.05	0.61	76.71
Eclipse3.0	7,250	727	376	2,151	0.25	0.05	0.60	75.94

model. In MMLR model the accuracy of nominal category is 62–66 %,its AUC come in the poor class. But all other 3 categories has accuracy 55 % and AUC for low (in Eclipse2.1 and 3.0) and high (in Eclipse2.0) comes under acceptable class and except high for Eclipse2.1 comes in excellent class. The AUC result for Eclipse2.1 in high category gives the outstanding classification. Binary model is not best way to find the fault-prone and fault-free classes, but Multinomial Logistic regression is the best way to classify the classes in different categories depending on the number of errors in classes and work more efficiently.

5 Conclusion

In this paper, we re-examine the ability of metrics (metrics covering all aspects of software's property) for predicting fault-prone classes in OO systems. Our results are summarized as follows:

- When Multivariate logistic regression models built with combined effect of all chosen metrics. This shows better result in prediction. But, among both aspect (i.e. Binary, Multinomial) multivariate model shows better result in multinomial aspect of fault prediction.
- Of the investigated metrics, PACK is the only metric which is used to develop model in both aspects as well as both way bivariate and multivariate. It shows that pack metric has the best discrimination ability.
- We conclude that prioritizing the test on the basis of different number of errors in different error categories is more effective than the category of fault prone and fault free classes in binary categorization for developing fault prediction model.

6 Future Work

In the future work, we will replicate this study using these investigated metrics and other modeling techniques to draw stronger conclusion for getting best predictor and model also. In this study we have used all those metric which are capable for giving the significant threshold for differentiating the classes in binary categorization (error-free and error-prone) and also multinomial categorization (nominal, low, mid and high) from our previous study [22].

Table 4 Evaluation result of the performance of multinomial multivariate model for all 3 version of eclipse

Version	Multinomial multivariate model																							
	True				False				Sensitivity				1-Specificity				AUC				Accuracy			
	Nom	Low	Mid	High	Nom	Low	Mid	High	Nom	Low	Mid	High	Nom	Low	Mid	High	Nom	Low	Mid	High	Nom	Low	Mid	High
Eclipse2	694.00	12.00	2.00	2.00	495.00	20.00	2.00	0.00	0.29	0.79	0.75	0.75	0.12	0.17	0.18	0.18	0.59	0.81	0.81	0.80	65.91	55.66	55.51	55.51
Eclipse2.1	453.00	5.00	2.00	3.00	368.00	11.00	1.00	1.00	0.26	0.58	0.75	1	0.07	0.10	0.11	0.11	0.59	0.74	0.83	0.97	62.29	55.55	55.51	55.52
Eclipse3	622.00	7.00	3.00	1.00	409.00	10.00	0.00	0.00	0.23	0.64	0.74	0.89	0.06	0.10	0.10	0.10	0.59	0.78	0.83	0.90	64.83	55.58	55.52	55.49

References

1. Bellini, P., Bruno, I., Nesi, P., Rogai, D.: Comparing fault-proneness estimation models. ICECCS '05 Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems, pp. 205–214 (2005)
2. Misra, S.: Evaluation criteria for object-oriented metrics. Acta Polytech. Hung. **8**(5), 109–136 (2011)
3. Pusatli, O.T., Misra, S.: Software measurement activities in small and medium enterprises: an empirical assessment. Acta Polytech. Hung. **8**(5), 21–42 (2011)
4. Seliya, N.: Software Quality Analysis with Limited Prior Knowledge of Faults. Wayne State University, Department of Computer Science, Graduate Seminar (2006)
5. Mittal, P., Singh, S., Kahlon, K.S.: Empirical model for fault prediction using.object-oriented metrics in mozilla firefox. Int. J. Comput. Technol. Res. **1**(6), 151–161 (2013)
6. Kayarvizhy, N., Kanmani, S.: High precision cohesion metric. WSEAS Trans. Inform. Sci. Appl. **10**, 79–89 (2013)
7. Zhou, Y., Xu, B., Leung, H.: On the ability of complexity metrics to predict fault-prone classes. J. Syst. Softw. **83**, 660–674 (2010)
8. Shatnawi, R., Li, W., Swain, J., Newman, T.: Finding software metrics threshold values using ROC curves. J. Softw. Maintenance Evol. Res. Pract. **22**(1), 1–16 (2010)
9. Menzies, T., Greenwald, J., Frank, A.: Data mining static code attributes to learn defect predictors. IEEE Trans. Softw. Eng. **32**(1), 2–13 (2007)
10. JHAWK.: Metrics reference. http://www.virtualmachinery.com/jhawkreferences.html. Accessed March 2014
11. Eclipse source code (for archived releases).: http://archive.eclipse.org/eclipse/downloads/. Accessed 3 Dec 2013
12. Eclipse bug data (for archived releases).: http://www.st.cs.uni-sb.de/softevo/bug-data/eclipse. Accessed 20 Nov 2013
13. Zimmermann, T., Premrai, R., Zeller, A.: Predicting defects for eclipse. In: Proceedings of the Third International Workshop on Predictor models in Software Engineering, 2007
14. Schroter, A., Zimmermann, T., Premraj, R., Zeller, A.: If your bug database could talk. In: Proceedings of the Fifth International Symposium on Empirical Software Eng. **2**, 18–20 2006
15. Hosmer D, Lemeshow S.: Applied Logistic Regression. Wiley, New York (1989)
16. Basili, V., Briand, L., Melo, W.: A validation of object oriented design metrics as quality indicators. IEEE Trans. Softw. Eng. **22**(10), 751–761 (1996)
17. Georage,D., Mallery,P.: SPSS for Windows STEP BY STEP. Pearson Education (2011)
18. Jiang, Y., Cukic, B., Ma, Y.: Techniques for evaluating fault prediction models. Empirical Softw. Eng. **13**(5), 561–595 (2008)
19. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recogn. **30**(7), 1145–1159 (1997)
20. Hopkins, W.G.: A New View of Statistics. Sport Science, New Zealand (2003)
21. Chidamber, S., Darcy, D., Kemerer, C.: Managerial use of metrics for object-oriented software: an exploratory analysis. IEEE Trans. Softw. Eng. **24**(8), 629–639 (1998)
22. Kumari, D., Rajnish, K.: Finding error-prone classes at design time using class based object-oriented metrics threshold through statistical method. Infocomp J. Comput. Sci. **12**(1), 49–63 (2013)