International Journal of Current Advanced Research
ISSN: O: 2319-6475, ISSN: P: 2319-6505, Impact Factor: 6.614
Available Online at www.journalijcar.org
Volume 7; Issue 5(I); May 2018; Page No. 12849-12851
DOI: http://dx.doi.org/10.24327/ijcar.2018.12851.2274

Research Article

SOFTWARE POST BUG PREDICTION ANALYSIS OF ECLIPSE VERSIONS 2.0 AND 2.1 USING FACTOR ANALYSIS AND LINEAR REGRESSION MODELING

Anurag Gupta[1]., Mayank Sharma[1] and Amit Srivastava[2]

[1]Amity University, Noida
[2]Jaypee University, Noida

A R T I C L E   I N F O

Article History:

Received 10th February, 2018
Received in revised form 6th March, 2018 Accepted 24th April, 2018
Published online 28th May, 2018

Key words:

Software Bug prediction, fault prone, factor Analysis, Regression

A B S T R A C T

As Software's are fault prone and there is a need to predetermine the chances of existence of bugs while developing the software on the basis of Different Metrics/Factors which are important. Through Bug prediction models, Software developers can know in advance which factors are important so that they will make sure that Probability of coming of Software Bugs post Release/Implementation is minimal/least. This Research paper emphasizes mainly on prediction of the software post release bugs for the Eclipse Software. In order to reduce the number of dimensions of the input feature vector, factor analysis which uses concept of feature selection is applied and new matrix with lower number of dimensions is used as input to general linear regression based prediction models. Finally results are compared among different versions of Eclipse i.e., version 2.0 & 2.1 with correlation based dimension selection process and empirical study was conducted to compare prediction results.

INTRODUCTION

There are various methods used in the Software Bug Prediction Analysis such as Logistic Regression, Naive Bayes (NB), k-Nearest Neighbor, Neural Network, Decision trees, Support Vector Machines, Random Forest etc.

We have used Factor Analysis and linear regression model for finding the most important factors for software bug prediction. T. Zimmermann, R. Premraj, and A. Zeller *et al.* [2] have mapped defects from the bug database of Eclipse (one of the largest open-source projects) to source code locations. The resulting data set lists the number of pre- and post-release defects for every package and file in the Eclipse releases 2.0, 2.1.

Several researchers used historical data without taking bug databases into account. Khoshgoftaar *et al.* [1] classified modules as defect-prone whenever the number of lines of code added or deleted exceeded a threshold. Graves *et al.* [3] used the sum of contributions to a module in its history to predict defect density. Ostrand *et al.* [4] used historical data from up to 17 releases to predict the files with the highest defect density in the next release. Hudepohl *et al.* [5] predicted whether a module would be defect-prone by combining metrics and historical data. From several software metrics, Denaro *et al.* [6] learned logistic regression models for Apache 1.3 and verified them against Apache 2.0.

Corresponding author:* **Anurag Gupta
Amity University, Noida

Factor Analysis based on covariance and feature selection methodology is used for the reduction of number of variables. Bug Findings is very important for any software to be implemented and maintained. In software engineering, fault detection and diagnosis (FDD) aims to determine whether a software module is faulty, or to estimate the probability that it has at least one fault, defined as fault-proneness. Different from the FDD of mechanical and electronic systems, fault-proneness prediction of software modules has no measurement data from sensors as their predictive features but rather it has software measures such as static code or design metrics as predictors. To predict the fault-proneness, we need a training dataset that includes software metrics and the fault proneness state (fault-free or faulty) of the instances but it is usually agreed that the more faults a module has, the more likely that it will fail in operation. In this paper, we have used the feature Selection using factor analysis for reducing the number of factors and then used the linear regression model for checking the best factors which helps software developers in finding the Software bugs for further versions of Eclipse.

Feature Selection

Feature selection is the study of algorithms for reducing dimensionality of data to improve machine learning performance. For a dataset with *N* features and *M* dimensions (or features, attributes), feature selection aims to reduce *M* to *M'* and $M' \leq M$ (Sammut and Webb 2011). It is an important and widely used approach to dimensionality reduction. Another effective approach is feature extraction. One of the key distinctions of the two approaches lies at their outcomes.

Assuming we have four features *F1*, *F2*, *F3*, *F4*, if both approaches result in 2 features, the 2 selected features are a subset of 4 original features (say, *F1*, *F3*), but the 2 extracted features are some combination of the 4 original features. Feature selection is commonly used in applications where original features need to be retained. Some examples are document categorization, medical diagnosis and prognosis as well as gene-expression profiling. The benefits of feature selection are multifold: it helps improve machine learning in terms of predictive accuracy, comprehensibility, learning efficiency, compact models, and effective data collection. The objective of feature selection is to remove irrelevant and/or redundant features and retain only relevant features.

We have Covariance calculation using correlation coefficient used in the Factor Analysis.

Instead of using only covariance between variables we used correlation coefficient which is the ratio of covariance between two variables divided by the standard deviation of each variable calculated using the below equation.

$$\rho(x,y) = \mathrm{cov}(x,y) / \sigma(x).\sigma(y)$$

Factor Analysis is done on the dimensions and then we found some Factors for the prediction of Bugs in the Software .After finding the factors we have applied the Linear regression for getting the dependency on the independent factors came using factor Analysis. After the components have been chosen and the matrix has been set, the matrix of correlations (in general case –n matrices) between parameters can be calculated. Factor analysis transforms this matrix to the matrix of factors, where each of them reflects a set of components connected to a one system-forming element and represents a system-forming connection of elements. It is important to note that by using the technique of principal components all factors become orthogonal and caused by different properties of the system. Hence, we can see that the factor analysis follows the logic of the above mentioned theoretical ideas and their principles.

Data Description

Data consists of one file for each level (files, packages) and release (2.0, 2.1). Each case contains the following information:

Name: The name of the file, respectively, to which this case corresponds. It can be used to identify the source code in the release and may be needed for additional data collection.

Pre-release defects*:* The number of non-trivial defects that were reported in the last six months before release.

Post-release defects*:* The number of non-trivial defects that were reported in the first six months after release.

Complexity metrics*:* We computed for each case several complexity metrics. Metrics that are Computed for classes or methods are aggregate by using average (avg), maximum (max), and accumulation (sum) to file and package level.

Summary of Data set we have taken is:

Project: Eclipse (eclipse.org)
Content: Defect counts (pre- and post-release)
Releases: Version 2.0, 2.1
Level: Packages and files
URL: http://www.st.cs.uni-sb.de/softevo/bug-data/eclipseNo.
More data: Eclipse source code (for archived releases): http://archive.eclipse.org/eclipse/downloads/

Different metrics are used (No. given in Table 1) such as Assignment,, Block, Comment Boolean Literal, Break Statement, Cast Expression, CatchClause, Character Literal, TOC,, Compilation Unit, Conditional ExpresoConstructor Invocation, Continue Statement, DoStatement are used.

Table 1 No. of metrics and files Initially Taken

Version No	Number of Complexity Metrics	No. of files
Eclipse 2.0	218	6729
Eclipse 2.1	256	6729

Analysis

Analysis is being done on two versions of Eclipse 2.0 and 2.1 by using factor Analysis and then we used Linear Regression Modeling in SPSS to get the best factors which are most important for the Bug Prediction for coming versions of the Eclipse.

Table 2 Adjusted R Square Values for two versions of eclipse

Model Summary				
Eclipse Version	R	R Square	Adjusted R Square	Std. Error of the Estimate
2.0	.634	.403	.392	.719
2.1	.634	.401	.391	.719

Adjusted R-Square - A version of R-Squared that has been adjusted for the number of predictors in the model. R-Squared tends to over estimate the strength of the association especially if the model has more than one independent variable.

$$R^2_{adj} = 1 - \left[\frac{(1-R^2)(n-1)}{n-k-1} \right]$$

In our Comparative Analysis in Table no. 2 Adjusted R Square is almost same in all the two versions. Standard error for all the two versions are almost same .791 for version 2.0 and version 2.1

Analysis of Variance (ANOVA) consists of calculations that provide information about levels of variability within a regression model and form a basis for tests of significance. The regression line concept, DATA = FIT + RESIDUAL, is rewritten as follows:

$$(y_i - \bar{y}) = (\hat{y}_i - \bar{y}) + (y_i - \hat{y}_i).$$

The first term is the total variation in the response y, the second term is the variation in mean response, and the third term is the residual value. Squaring each of these terms and adding over all of the n observations gives the equation

$$\sum (y_i - \bar{y})^2 = \sum (\hat{y}_i - \bar{y})^2 + \sum (y_i - \hat{y}_i)^2.$$

This equation may also be written as SST = SSM + SSE, where SS is notation for sum of squares and T, M, and E are notation for total, model, and error, respectively.

The square of the sample correlation is equal to the ratio of the model sum of squares to the total sum of squares:
r^2 = SSM/SST.

This formalizes the interpretation of r^2 as explaining the fraction of variability in the data explained by the regression model.

The sample variance sy^2 is equal to

$\sum(yi - \bar{y})^2/(n - 1) = SST/DFT$,

the total sum of squares divided by the total degrees of freedom (DFT). For simple linear regression, the MSM (mean square model) = $\sum(\hat{y}_i - \bar{y})^2 / (1) = SSM/DFM$, since the simple linear regression model has one explanatory variable x. The corresponding MSE (mean square error) = $\sum(yi - \hat{y}i)^2/(n - 2) = SSE/DFE$, the estimate of the variance about the population regression line (σ^2).

ANOVA calculations are displayed in an analysis of variance Table 3, which has the following format for simple linear regression:

Table 3 General Anova Table

Source	Sum of squares	Degrees of freedom	Mean Square	F
Model	$\sum(\hat{y}_i - \bar{y})^2$	1	SSM/DFM	
Error	$\sum(yi - \hat{y}i)^2$	n-2	SSE/DFE	MSM / MSE
Total	$\sum(yi - \bar{y})^2$	n-1	SST/DFT	

Table 4 ANOVA TABLE for two versions of eclipse

ANOVA[a]						
Eclipse Version		Sum of Squares	df	Mean Square	F	Sig.
2.0	Regression	2301.853	113	20.370	39.439	.000[b]
	Residual	3416.695	6615	.517		
	Total	5718.548	6728			
2.1	Regression	2295.410	110	20.867	40.343	.000[b]
	Residual	3423.138	6618	.517		
	Total	5718.548	6728			

Table 4 shows that F test comes out to be same for both the versions and having almost the same value**.**

RESULT

Table 5 Shortlisted Factors Having Significance Level Between .000 And .003 And Then Common Factors Among Two Versions

S No.	Eclipse Version 2.0	Eclipse Version 2.1	Intersection
1	NOF_MAX	NBD_SUM	NSM_AVG
2	NSM_AVG	NOM_SUM	BLOCK
3	BLOCK	NSM_AVG	QUALIFIEDNAME
4	QUALIFIEDNAME	BLOCK	TRYSTATEMENT
5	TRYSTATEMENT	FIELDDECLARATION	PRE
6	PRE	IFSTATEMENT	NBD_SUM
7	NBD_SUM	TRYSTATEMENT	IFSTATEMENT
8	NOF_AVG	NULLLITERAL	TLOC
9	ARRAYINITIALIZER	QUALIFIEDNAME	MODIFIER
10	CATCHCLAUSE	MODIFIER	NUMBERLITERAL
11	IFSTATEMENT	TLOC	NOM_SUM
12	IMPORTDECLARATION	PRE	
13	STRINGLITERAL	NORM_FIELDACCESS	
14	VARIABLEDECLARATIONFRAGMENT	NBD_MAX	
15	NORM_THISEXPRESSION	VG_MAX	
16	TLOC	NUMBERLITERAL	
17	MODIFIER		
18	ASSIGNMENT		
19	NOM_SUM		
20	ARRAY_CREATION		
21	MLOC_MAX		
22	NORM_THROWSTATEMENT		
23	NORM_BLOCK		
24	NUMBERLITERAL		
25	EXPRESSIONSTATEMENT		

Table 5 displays Total No of factors coming from Factor Analysis & Significance Level Between .000 And .003 in Eclipse version 2.0 are **25 &** Total No of factors coming from Factor Analysis & Significance Level Between .000 And .003 in Eclipse version 2.1 are **16**.

This table also shows the intersection of the factors from version 2.0 and version 2.1 in the fourth column which are common factors and these comes out eleven These eleven factors are to be considered the most important and prone to Bugs post software release.

CONCLUSION

After the Analysis we come to the conclusion that Eleven predictors NSM_AVG, BLOCK, QUALIFIEDNAME, TRYSTATEMENT, PRE, NBD_SUM, IFSTATEMENT, TLOC, MODIFIER, NUMBERLITERAL, NOM_SUM are most important and useful for the prediction of Software Bugs.

References

1. T. M. Khoshgoftaar, E. B. Allen, N. Goel, A. Nandi, and J. McMullan, "Detection of software modules with high debug code churn in a very large legacy system." in *ISSRE '96: Proceedings of the The Seventh International Symposium on Software Reliability Engineering (ISSRE '96)*, Washington, DC, USA, 1996, p. 364.
2. T. Zimmermann, R. Premraj, and A. Zeller," Predicting Defects for Eclipse." PROMISE '07 Proceedings of the Third International Workshop on Predictor Models in Software Engineering.
3. T. L. Graves, A. F. Karr, J. S. Marron, and H. Siy, "Predicting fault incidence using software change history." *IEEE Transactions on Software Engineering,* vol. 26, 2000.
4. T. J. Ostrand, E. J. Weyuker, and R. M. Bell, "Predicting the location and number of faults in large software systems." *IEEE Trans. Software Eng.,* vol. 31, pp. 340-355, 2005..
5. J. P. Hudepohl, S. J. Aud, T. M. Khoshgoftaar, E. B. Allen, and J. Mayrand, "Emerald: Software metrics and models on the desktop." *IEEE Software,* vol. 13, pp. 56-60, September 1996.
6. G. Denaro and M. Pezzè, "An empirical evaluation of fault-proneness models." in *Proceedings of the International Conference on Software Engineering (ICSE 2002)*, Orlando, Florida, USA, 2002, pp. 241-251.
7. A. T. Nguyen, T. T. Nguyen, J. AI-Kofahi, H. V. Nguyen and T. N. Nguyen, "A topic-based approach for narrowing the search space of buggy files from a bug report", ASE '11, pp. 263-272.