International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Empirical Model for Fault Prediction On the Basis of Regression Analysis

Manbir Kaur Dhillon[1], Prof. Birinder Singh[2], Prof. Jatinder Singh[3]

[1,2,3]Department of Computer Science and Engineering, Baba Banda Singh Bahadur Engineering College
Fatehgarh Sahib, (Punjab) India

Abstract: ***Software fault prediction is the most efficient and systematic approach to improve the quality of the software products. It is essential to find the defect or fault as quick as possible to improve the quality of the software. In fault prediction model development research, combination of metrics significantly can compare the fault prediction of the different model. In this study, binary logistic regression technique is used for the prediction of the faults in the software. Binary logistic regression measures the relationship between the categorical dependent variable and one or more independent variables. This work takes in account the software metrics to improve the quality of the object oriented software. We compared different prediction models based on regression analysis for the fault free software. Through the results derived, it is empirically established and validated that the performance of binary logistic regression remains ordinarily unaffected and simultaneously provides superior performance for the prediction model.***

Keywords: Fault prediction, Binary logistic regression, Software metrics, software quality, prediction model.

1. Introduction

Software metrics are vital tools to audit software quality throughout the project life cycle. Software metrics are indicators of many externally recognized quality factors including fault proneness, reusability, and maintenance time of the software code. Identifying which classes are more likely to have faults is necessary to guide software testers and reduce the costs of quality assurance. Development of software is a creative process where efficiency of each person is different. Software developers and researchers are using different techniques and are more concerned about accurately predicting the faults of the software product being developed. Important decisions need to be made during the development process of software products. Software fault prediction is the process of locating defective modules in software. To produce high quality software, the final product should have as few faults as possible. Early detection of software faults could lead to reduced development costs and rework effort and more reliable software. So, the study of the fault prediction is important to achieve software quality.

2. Literature Survey

Many researchers have carried out significant work in the area of fault prediction in software. A number of regression techniques have been developed to build the fault prediction model from the existing datasets. The purpose of regression analysis is to analyze relationships among variables. Lin et al. [1] proposed a hierarchical grey clustering approach in which the similarity measure was a globalized modified grey relational grade instead of traditional distances. Pradeep Singh et al. [6] present a novel fault prediction technique that reduces the probability of false alarm (pf) and increases the precision for detection of faulty modules. The general expectation from a predictor is to get very high probability of false alarm (pf) to get more reliable and quality software product. Kuk lida lee et al. [16] demonstrate that logistic regression can be a powerful analytical technique for use when the outcome variable is dichotomous. The effectiveness of the logistic model was shown to be supported by (a) significance tests of the model against the null model, (b) the significance test of each predictor, (c) descriptive and inferential goodness-of-fit indices, (d) and predicted probabilities. Taghi m. Khoshgoftaar et al. [14] compares the fault prediction accuracies of six commonly used prediction modelling techniques, CART-LS, CART-LAD, S-PLUS, CBR, ANN, and MLR. Walter Bouwmeester et al. [3] compared prediction models that were developed with random intercept or standard logistic regression analysis in clustered data. Shilpa Sharma et al. [7] depicts that a variety of software fault predictions techniques have been proposed, but there is very less work on the prediction of Level of severity of faults present in the software system. Giuseppe Scanniello et al. [8] suggest that defect prediction approaches use software metrics and fault data to learn which software properties associate with faults in classes. This paper proposes an intra-release fault prediction technique, which learns from clusters of related classes, rather than from the entire system. Yogesh Singh et al. [12] proposed several metrics that can be used in predicting important quality attributes such as fault proneness, maintainability, effort, productivity and reliability. Banu Diri et al. [11] aims at classifying studies with respect to metrics, methods, and datasets that have been used in these prediction papers. PROMISE repository includes a collection of public datasets to build repeatable, refutable and verifiable models of software engineering and it was inspired by UCI Machine Learning Repository which is widely used by researchers in Machine Learning area. Piotr Gawrysiak et al. [5] The paper contains description of a new clustering methodology that partitions data set into clusters, such that regression in determination coefficient for data from each cluster is minimized. Jiang et al. [20] using ISBSG data, had built linear regression models with a logarithmic

transformation based on function points. Regression model had been used as an activation function in a neural network for the calibration of weights in the function point model.

3. Research Objectives

Earlier studies research results of mining the faults with better mining techniques are not leading to the acceptable results for fault prediction in software metrics. With the focus on this idea the following objectives of the study are formulated:

- To study previously designed framework developed for fault prediction.
- To do the classification of heterogeneous data to homogeneous form with normalization of data set.
- To do clustering of homogeneous dataset.
- To design threshold of software metrics for the purpose of accuracy.
- To do prominent clustering and design the fault prediction model.
- To Improve test process by focusing on fault-prone modules.

4. Proposed Methodology

The methodology used for the detection of the faults in software system is based upon the regression analysis after which fault prediction model will be developed. We will introduce a model for fault prediction using logistic regression technique. In this work, first step is to normalize the datasets, K-mean clustering techniques for the reduction of heterogeneity. After that GRG is calculated based on grey relational analysis and logistic regression is applied on GRG dataset. At last fault prediction model is build using based on regression analysis.

4.1. Data Collection

The PROMISE Repository is a research dataset repository specializing in software engineering research datasets. In this study, experiments are conducted on version 1.5 of velocity that was taken for analysis. This dataset is taken from promise repository which helps us to collect all CK metrics values.

4.2. Tools Used

Weka is a collection of machine learning algorithms for data mining tasks [27]. The algorithms can either be applied directly to a dataset or taken from your own Java code.

SPSS (Software Package used for Statistical Analysis) is a Windows based program that can be used to perform data entry and analysis and to create tables and graphs. SPSS is capable of handling sizably voluminous amounts of data and can perform all of the analyses covered in the text.

4.3. Clustered regression using grey relational analysis:

In this methodology, to reduce the faults that exists in the datasets, the initial focus on clustering of dataset.

The four main steps involved in the algorithm are:

Step 1: Finding feature weights using grey relational analysis.
Step 2: Clustering the datasets based on feature-weights those found using grey relational algorithm.
Step 3: Applying regression techniques on the clustered datasets.
Step 4: After that model for fault prediction by regression techniques

4.4. Grey Relational Analysis

GRA is comparatively a novel technique in software fault prediction. It is used for analyzing the relationships that exists between two series. The magnetism of GRA to software fault prediction shoots from its flexibility to design the fault prediction for software reliability and better performance of the software.

4.5. Grey Relational Grade:

GRA is used to quantify all the influences of various factors and the relationship among data series that is a collection of measurements. The main steps involved in the process are:

Data Processing: The first step is the standardization of the various attributes. Every attribute has the same amount of influence as the data is made dimensionless by using various techniques like upper bound effectiveness, lower bound effectiveness or moderate effectiveness. Upper-bound effectiveness (i.e., larger-the-better) is given by:

$$X(k) = \frac{x_i(k) - min_i x_i(k)}{max_i x_i(k) - min_i x_i(k)} \qquad (i)$$

Where i=1,2,….,m and k=1,2,…n.

$$\gamma(x_0(k), x_i(k)) = \frac{\Delta_{min} + \zeta\Delta_{max}}{\Delta_{vj}(k) + \zeta\Delta_{max}} \qquad (ii)$$

where;

$\Delta_{0,i}(k) = |x_0(k) - x_i(k)|$ is the difference of the absolute value between $x_0(k)$ *and* $x_i(k)$;
$\Delta_{min} = min_j min_k |x_0(k) - x_j(k)|$ is the smallest value of $\Delta_{0,j} \forall_j \in \{1, 2, \ldots, n\}$;
$\Delta_{max} = \max_j \max_k |x_0(k) - x_j(k)|$ is the largest value of $\Delta_{0,j} \forall_j \in \{1, 2, \ldots, n\}$; and
ζ is the distinguishing coefficient, $\zeta \in (0, 1]$.

The ζ value will change the magnitude of $\gamma(x_0(k), x_i(k))$. In this study the value of ζ has been taken as 0.5.

4.6. Logistic Regression:

Logistic regression is used to explain the relationship between one dependent binary variable and one or more metric (interval or ratio scale) independent variables. Logistic regression can be binomial, ordinal or multinomial. In this work, binary logistic regression is used. Binomial or binary logistic regression deals with situations in which the observed outcome for a dependent variable can have only two possible types.

The logistic regression can be understood simply as finding the β parameters that best fit:

$$f(x) = \beta_1 x_2 + \beta_2 x_2 + \beta_3 x_3 + \cdots + c$$

Where*; x1, x2, x3* are the metric values.

An explanation of logistic regression can begin with an explanation of the standard logistic function. The logistic function is useful because it can take an input with any value from negative to positive infinity, whereas the output always takes values between zero and one and hence is interpretable as a probability. The logistic function σ is defined as follows:

$$\sigma = \frac{e^{f(x)}}{1 + e^{f(x)}}$$

4.7. Evaluation Parameters:

After the prediction of fault-proneness data, a confusion matrix, which is useful in the process of performance evaluation, has been calculated.The confusion matrix is used to measure the performance of using threshold model in identifying actual fault classes using three measures, Recall, Precision, and F-measure. The actual labels of data items are placed along the rows while the predicted labels are placed along the columns.

These measures are calculated as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F - measure = \frac{(\beta + 1) * Precision * Recall}{\beta * Precision + Recall}$$

Results & Discussions

In order to evaluate the model based upon the proposed methodology, the well established dataset from the Promise repository have been used for validating the model. The skewness statistics for WMC, DIT, NOC, CBO, RFC, and LCOM metrics are shown in the table I for the the release of velocity 1.5. Many experimental studies have developed the metrics model or used the metrics values to predict the faults in the software classes for the better software quality. The descriptive statistics for the WMC, DIT, NOC, CBO, RFC and LCOM are shown in tables for the the dataset of velocity. Study includes the discussion of results in comparison to previous studies done by Rosenberg [26]. Following are the results discussed for each of the metric separately.

- WMC: WMC is an indicator of a class or interface complexity. Large values of WMC should be considered problematic. Table II shows the descriptive statistic for the WMC metric on the basis of which performance is evaluated for the velocity 1.5. The mean value is 8.9 and median is 5 and mode is 5.
- DIT: DIT is an indicator of depth of inheritance in a class. Large DIT values lead to more complex classes, which are difficult to understand, maintain, and reuse. DIT threshold can be used to mark classes that need more attention during both testing and maintenance phases. Although there have been many studies on the impact of the depth of inheritance, there is no consensus on the acceptable depth. Table II shows the potential descriptive statistic of velocity 1.5 for DIT. In table I, the skewness statistics for DIT for velocity 1.5 is -0.504. It is noticed that the mean value for the DIT metric is 1and close to median and mode.

Table I: Skewness statistics of velocity 1.5.

Metric	Skewness
WMC	-2.781
DIT	-0.504
NOC	-8.91
CBO	-1.406
RFC	-1.407
LCOM	-9.363

WMC, weighted methods complexity; DIT, depth of inheritance hierarchy; NOC, number of child classes; CBO, coupling between objects; RFC, response for class; LCOM, lack of cohesion of methods.

- NOC: NOC is an indicator of both abstraction and inheritance in classes. Large values of NOC could be marked as problematic in maintenance and testing phases and could increase the effort required for both phases. NOC thresholds have been considered in many previous works, and corresponding threshold values were identified. Table II shows the descriptive statistics values of velocity 1.5 for NOC.
- CBO: CBO is a well-known metric for measuring coupling among classes. Coupling is considering a problem maker in both the design and code. High coupling makes a class more complex, difficult to understand and increases testing and maintenance efforts. The developers can reduce coupling among classes throughout code cleaning and refactoring continuously. However, where high coupling that should be marked by the developers is not certain. There have been many threshold values reported previously for the CBO. Table II shows the descriptive statistics values of velocity 1.5 for CBO.
- RFC: RFC is an indicator of the amount of responsibility amounted to a class. Classes that have large response set are prone to faults and need more maintenance. Such classes usually need more testing effort to make sure the class implementation meets all requirements assigned to the classes. The response set should be limited with a threshold values for all classes. RFC values have a large variance, and there is no one threshold found suitable for all projects. Table II shows the descriptive statistics values of velocity 1.5 for RFC.
- LCOM: LCOM is one of the significant metrics that can be used to evaluate an object-oriented software system. LCOM is useful for estimating the amount of cohesion in the system. For example, in an object-oriented system, the LCOM can be used to measure the cohesion of each class of the system. A high LCOM value could indicate that the design of the class is poor. Table I shows the value of the skewness statistics for LCOM for velocity 1.4, velocity 1.5 and velocity 1.6. Table II shows the values of descriptive statistics for all the velocity release.

Table V: Descriptive statistics for velocity 1.5

Metric	Minimum	Maximum	Median	Mean	Mode
WMC	0	152	5	8.9	5
DIT	1	5	1	1.6	1
NOC	0	43	0	0.4	0
CBO	0	74	8	10.6	5
RFC	0	265	16	22.6	6
LCOM	0	7900	4	78.8	0

WMC, weighted methods complexity; DIT, depth of inheritance hierarchy; NOC, number of child classes; CBO, coupling between objects; RFC, response for class; LCOM, lack of cohesion of methods.

In this work, the F-measure of the three metrics is calculated collectively i.e. first of WMC, DIT and NOC and after that of CBO, RFC and LCOM in release of velocity 1.5. The value of F-measure calculated from the velocity 1.5 dataset is considered as the standard F-measure as its values are close to 1. The value of F-measure of dataset equal to or close to one is considered to be the standard F-measure and best dataset version. This standard F-measure of the velocity dataset 1.5 is calculated after finding the GRG and applying the binary logistic regression on that GRG and bugs. Therefore dataset of velocity 1.5 is considered to be as the best dataset model.

- WMC, DIT and NOC metrics has value of F-measure of 0.790 in velocity 1.5 dataset shown in table V. This value of F-measure of metrics WMC, DIT and NOC of velocity 1.5 have the value close to 1 and is considered to be the standard F-measure values for the metrics WMC, DIT and NOC. This also means the value of F-measure close to 1 in the velocity release are the faults predicted in that dataset.
- CBO, RFC and LCOM has value of F-measure to be 0.768 in velocity 1.5 as shown in table V. Low value of F-measure means that there are more faults in that velocity dataset. Following tables show the values of precision, recall and their corresponding F- measures.

Table V: Values of Precision, Recall and F-measure of velocity 1.5.

Metric	Precision	Recall	F-measure
WMC	0.654	1	0.790
DIT	0.654	1	0.790
NOC	0.654	1	0.790
CBO	0.673	0.9	0.768
RFC	0.673	0.9	0.768
LCOM	0.673	0.9	0.768

WMC, weighted methods complexity; DIT, depth of inheritance hierarchy; NOC, number of child classes; CBO, coupling between objects; RFC, response for class; LCOM, lack of cohesion of methods.

The values of both the Recall and Precision are between 0 and 1. Values that are close to 1 mean better results. If the value is 1, then the classifier is ideal and without FN or FP. The high values of Recall and Precision do not coincide. In practice, it is hard to achieve high Recall and Precision, that is, high Recall occurs often with low Precision. Then the F value of different versions of each metric are compared and the above results are concluded. The results show that the performance of most of the systems is evaluated by their derived thresholds and then apply it over the other datasets. Following table show the values of precision, recall and their corresponding F-measure.

Software metrics has been used to describe the complexity of the program and, to estimate software development time. Software metrics elucidate quantitative measurements of the software product or its specifications. Software metrics can be used to evaluate software quality from various perspectives including software fault proneness and maintenance effort. A software fault is an error, flaw, bug, mistake, failure, or defect in a computer program or system that may generate an inaccurate or unexpected outcome, or precludes the software from behaving as intended. Fault prediction is extremely essential in the field of software quality and software reliability. Defect prediction is comparatively a novel research area of software quality engineering. Many software metrics are available that were proposed to measure the internal structure of object-oriented systems. We limit our study on only one set of software metrics which is the CK metrics that were proposed and validated as measures of six internal attributes of the software classes including coupling, cohesion, complexity, inheritance depth, inheritance breadth, and class responsibilities. Many commercial and open-source tools are available to assess and provide quality indicators of software code using software metrics. Most of these tools collect metrics and analyze metrics using software threshold values that are already injected in these tools. Software metrics can be used to evaluate software quality from various perspectives including software fault proneness and maintenance effort.

- The proposed techniques in previous works could not identify thresholds for all CK metrics.
- The identified thresholds using data distribution (histogram analysis and distribution parameters) were based on either small number of projects (except the work of [8]) or no empirical work has been conducted to identify thresholds.
- The effect of skewness in metrics has not been accounted for although data skewness has been reported frequently.

5. Conclusion

Software fault prediction model is extremely essential in the field of software quality and software reliability. Software fault prediction can be regarded as one piece of the solution for timely and cost effective software development. In this work, the fault prediction model for the dataset has been done for the evaluation of the best model. The model is based on software metrics and faults. For calculating the predicted values of every metric, logistic regression technique is applied to the datasets. Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. On the basis of the

f-measure calculated for the datasets using logistic regression, it is concluded that the model with the value of F-measure close to 1 is the best model. Value of F-measure close to 1 means that the software is fewer faults predicted and reliable and is good quality software model. In future, it is planned to use the linear regression technique for the fault prediction model.

References

[1] Lin, C. T. and Tsai, H. Y. (2005) "Hierarchical Clustering Analysis Based on Grey Relation grade",Information and Management Sciences, Vol. 16, No. 1, pp. 95-105.
[2] Pradeep Singh (2014), "An Efficient Software Fault Prediction Model using Cluster based Classification", International Journal of Applied Information Systems (IJAIS), Volume- 7, No. 3, pp. 206-216.
[3] Kuk Lida Lee (2002), "An Introduction to Logistic Regression Analysis and Reporting" The Journal of Educational Research, Vol. 96, No. 1, pp. 546-555.
[4] Taghi m. khoshgoftaar and Naeem seliya. (2003) "Fault Prediction Modeling for Software Quality Estimation: Comparing Commonly Used Techniques". Kluwer Academic Publishers. Vol:8, pp. 255-283.
[5] Geeta nagpal, Moinuddin and Arvinder kaur(2013), "Estimating Project Development Effort Using Clustered Regression Approach", CCSIT, SIPP, AISC, PDCTA, pp. 493–507.
[6] Albrecht, A.J. and Gaffney, J.R. (1983) "Software measurement, source lines of code, and development effort prediction: a software science validation", IEEE Transactions on Software Engineering, Vol. 9, No. 6, pp. 639-648.
[7] Putnam and Lawrence H. (1978) "A General Empirical Solution to the Macro Software Sizing and Estimating Problem", IEEE Transactions on Software Engineering, Vol. 4, No. 4, pp. 345-361.
[8] El-Zaghmouri, B. M. and Abu-Zanona, M. A. (2012) "Fuzzy C-Mean Clustering Algorithm Modification and Adaption for Application", World of Computer Science and Information Technology Journal, ISSN: 2221-0741, Vol.2, No.1, pp. 42-45.
[9] Lin, C. T. and Tsai, H. Y. (2005) "Hierarchical Clustering Analysis Based on Grey Relation grade",Information and Management Sciences, Vol. 16, No. 1, pp. 95-105.
[10] Wong, C.C. and Chen, C.C. (1998) "Data clustering by grey relational analysis", J. Grey Syst, Vol. 10,No. 3, pp. 281-288.
[11] Hu, Y.C., Chen, R. S., Hsu, Y. T., and Tzebg, G. H. (2002) "Grey self-organizing feature maps", Neuro computing, Vol. 48, No. 4, pp. 863-877.
[12] Duda, R.O., and Hart, P.E. (1973), "Pattern classification and scene analysis", John Wiley &Sons ,New York, pp. 443-458.
[13] Bezdek, J. C., Ehrlich, R. and Full, W. (1984) "FCM: The Fuzzy c- Means Clustering Algorithm",Computers and Geoscience, Vol. 10, No. 2-3, pp. 191-203.
[14] Runkler, T.A. &Bezdek, J.C. (1999) "Alternating cluster estimation: a new tool for clustering and function approximation", IEEE Trans. Fuzzy Syst., Vol. 7, No. 4, pp. 377-393.
[15] Boehm. B (1981), "Software Engineering Economics Englewood Cliffs", NJ, Prentice Hall, pp. 123-131.
[16] Pedrycz. W (1996) "Conditional fuzzy c-means", Pattern Recogn. Lett.,Vol. 17, No. 6, pp 625-632.
[17] Kung C. C and Su J. Y. (2007) "Affine Takagi-Sugeno fuzzy modeling algorithm by Fuzzy c regression models clustering with a novel cluster validity criterion", IET Control Theory Appl., pp. 1255 – 1265.
[18] Wang, W., Wang, C., Cui, X. and Wang, A. (2008) "A clustering algorithm combine the FCM algorithm with supervised learning normal mixture model", pp. 1-4.
[19] Chang, K. C. and Yeh, M. F. (2005) "Grey Relational Based Analysis approach for data clustering",IEEE Proc.-Vis. Image Signal Process, Vol.152, No.2, pp.354-366.
[20] Song, Q., Shepperd M. and Mair C. (2005) "Using Grey Relational Analysis to Predict Software Effort with Small Data Sets". Proceedings of the 11th International Symposium on Software Metrics (METRICS'05), pp. 35-45.
[21] Cagatay Catal (2012), "Performance Evaluation Metrics for Software Fault Prediction Studies". Acta Polytechnica Hungarica, Vol. 9, No. 4, pp. 199-204.
[22] Mrinal Singh Rawat (2012), "Software Defect Prediction Models for Quality Improvement", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 2, pp. 132-141.
[23] Satwinder singh (2013), "Analysis of CK Metrics to predict Software Fault-Proneness using Bayesian Inference", International Journal of Computer Applications, Volume 74– No.2, pp. 43-53.
[24] Lilly Florence (2015), "A Study on Software Metrics based Software Defect Prediction using Data Mining and Machine Learning Techniques", International Journal of Database Theory and Application, Vol.8, No.3 , pp.179-190.
[25] Geeta Nagpal (2014), "Grey Relational Effort Analysis Technique Using Regression Methods for Software Estimation",The International Arab Journal of Information Technology, Vol. 11, No. 5, pp. 255-263.
[26] Aditi Puri(2014), "Genetic Algorithm Based Approach For Finding Faulty Modules In Open source Software Systems", International Journal of Computer Science & Engineering Survey (IJCSES) Vol.5, No.3, pp. 196-205.
[27] http://www.cs.waikato.ac.nz/ml/weka/