International Journal of Current Engineering and Technology
©2016 INPRESSCO®, All Rights Reserved

E-ISSN 2277 – 4106, P-ISSN 2347 – 5161
Available at http://inpressco.com/category/ijcet

Research Article

AUC based Software Defect Prediction for Object-Oriented Systems

Dharmendra Lal Gupta[#]* and Kavita Saxena[#]

[#]Department of Computer Science & Engineering, Meware University, Chittorgarh, Rajasthan, India

Accepted 24 Sept 2016, Available online 25 Sept 2016, **Vol.6, No.5 (Oct 2016)**

Abstract

In this paper object oriented defect datasets have been collected from open source promise data repository. Out of 20 provided metrics in each dataset, most prominent 14 metrics have been selected using feature selection process. These metrics are directly responsible for bug prediction in such systems. In this paper most prompting classifier Logistic Regression based on 10-cross validation has been used. The findings have been analyzed using Area under Curve (AUC) values. This information can be used by software developers to enhance the quality of a system. WEKA tool has been used for finding and analysis of our result. A Comparative study has also been done in terms of AUC values obtained by proposed model and Mamdouh Alenezi Model. Our proposed model is providing better result that Mamdouh Alenezi Model.

Keywords: *Software Bug, Software Defect Prediction, AUC*

1. Introduction

Defect as well as cost both are the deriving forces in software development. Software defect prediction using machine learning approaches are now becoming the popular techniques. But before to predict the quality we must be ensured that the software metrics have empirically validated using machine learning methods and they also have the practical relevance in the assessment of quality factors.

It is very difficult to produce a fault free software whereas within the complexity and constraints the development environment are increasing gradually. To solve such problems we have to focus our prediction on some software quality attributes such as fault proneness, effort required, testability issues, maintainability and reliability factors in the early phases of software development.

Based on ROC (Receiver operating characteristics) analysis, one can easily predict that such models (as our model is discussed) can be helpful in planning and performing testing issues by focusing resources on fault-prone parts at design and coding levels.

Rest of the paper is organized as follows: Section 2 describes about the work done by various person as related work. Methodology is described in section3. In section 4 different experiments have been done on all the datasets. Section 5 describe, summary and conclusion portion of the paper. Future scope is mentioned is section 6.

*Corresponding author: **Dharmendra Lal Gupta** is a Research Scholar **and Kavita Saxena** is working as Associate Professor

2. Related Work

Software defect prediction has been under study for a very long time. Various software defect prediction models are available in the literature. In the last few decades many software developers and researchers studies have been carried out in this area of software defect prediction. Regression tree, Neural Networks, Fuzzy logic etc. and its applications have been used for Software quality prediction.

Support Vector Machine approaches have been used by Xing F *et al.*., Singh Y *et al.*. and Gondra I. *et al.*. Classification of software modules based on complexity metrics have been used and a module has been proposed for early software quality prediction for small number of sample data. Singh Y *et al.*. have manifested a SVM based software fault prediction model which investigate the relationship between object-oriented metrics and fault-proneness. They have preferred KC1 a NASA dataset and computed ROC (Receiver Operating Characteristics) as performance measure. Gondra I. *et al.*. have shown software fault prediction model based on Artificial Neural Network using sensitivity analysis and also using SVM and on the basis of result it is advocated that SVM is better than ANN.

Amasaki S. *et al.*. have used Bayesian belief network approach to predict the final quality of software. In this paper they have done empirical evaluation based on the metrics data gathered from development projects in a certain company. As a result of empirical evaluation they confirmed that the proposed model can predict the amount of residual faults the software reliability growth model is unable to handle.

Catal C. has reported in his paper about a complete survey of 90 research papers regarding the direction of research work done during 1990 to 2009.

In addition to it he has also focused on machine learning approach and statistical data calculation for the fault prediction.

Chidamber *et al.*. have shown the key requirements of measurement to improve the quality of software with the help of new metrics suite which consists of six design level metrics named WMC, DIT, NOC, CBO, RFC and LCOM.

Churcher N. I. *et al.*. have commented on the metrics suite developed by CK (Chidamber S.R., and Kemerer C. F.) and they have illustrated their views by counseling a fundamental property NOM (Number of Methods in a Class) and demonstrated that this is open to a variety of interpretations, they have manifested that the number of methods per class is required directly for the computation of WMC (Weighted Method per Class) and indirectly for other metrics provided by CK.

Olague H.M. *et al.* have shown an empirical validation of Chidamber & Kemerer , Brito e Abreu's MOOD Metrics and Bansiya and Davis's Quality Model for Object Oriented Design (QMOOD) Metric suite which helps during prediction of fault-proneness of object-oriented classes. In this paper authors have used Mozila Rhino project and its versions as datasets and metrics have been collected using SSML (Software System Markup Language) Tool. Finally they have concluded that CK and MOOD metrics contain almost similar components and in these cases product statistical models are much effective in the detection of error-prone classes. They have also commented that the class components in MOOD metrics suite are not to good class fault-prone predictors. The authors have applied Multivariate binary logistic regression models across six Rhino Versions which indicates that these models may be useful in assessing the quality in object oriented classes shaped using highly iterative software development processes.

Jureczko M. *et al.*. have presented an analysis regarding defect prediction using clustering technique on software projects. They have used a data repository with 92 versions of 38 proprietary, academic and open source projects. In this paper Hierarchical and K-means clustering as well as Kohonen's neural network has used to find the groups of similar projects. Two defect prediction models were created for each of the identified groups. In this study Ckjm Tool has been used for the retrieval of all metrics which will be used for defect prediction model. JUnit and FitnNess have been used as test Tools. The authors have identified two clusters and compared with results obtained by other researchers. Finally clustering is suggested and applied.

Karabulut E. M. *et al.* have shown the effect of feature selection method (in which irrelevant features from the original dataset is removed). The authors have used 15 datasets from University of California Irvine's. Data Mining Repository to compare three classification algorithms with respect to their effect from six feature selection filters. The authors have done their experimental work using WEKA data mining tool. Authors have investigated the impact of feature selection on aforementioned classifiers and observed that MLP is most effective classifier. It is also discovered for classifier that the Gain Ratio, for MLP the Chi-square and for J48 the Information Gain is the most effective feature selection algorithm.

Alenezi M. has empirically investigated the relationship between metrics of open source defective software systems and their fault proneness. For this, he has applied Feature Selection technique on all the twenty metrics of datasets and selected nine metrics out of twenty for his model which is based on Random Forest classifier and he has computed AUC as well as F-Measure value on eight datasets (Camel 1.6, Ant 1.7, Xerces 1.4.4 , jEdit 4.3, POI, Lucene and Synapse) and suggested that by focusing on a small and precise set of internal attributes (metrics) the quality of the model can be improved as well as quality assurance team can also save time and resources while getting high accuracy fault proneness predictions.

Table 1: Data Description

S No.	Dataset Name	Instances	*Instances after Normalization
1	Camel1.6	965	884
2	Tomcat 6.0	858	796
3	Ant 1.7	745	724
4	jEdit4.3	492	476
5	Ivy 2.0	352	345
6	arc	234	215
7	e-learning	64	57
8	berek	43	43
9	forrest 0.8	32	31
10	zuzel	29	29
11	Intercafe	27	27
12	Nieruchomosci	27	26

*These values have been obtained after Normalization. The duplicate entries have been deleted except one entry

3. Methodology

In this paper fault proneness have been calculated in terms of AUC (Area under Receiver operating characteristics Curve) values, by delineating the confusion matrix for each of the techniques involved separately. All the defect datasets must be having relation with their different metrics set and bug values corresponding to each and every instances of the dataset/datasets. If any instance is lacking the value of some metrics then in that case all such metrics values will be considered as zero.

3.1 Data Description

12 open source object-oriented defect projects and each of which are of Marian Jureczko datasets. (Camel1.6, Tomcat 6.0, Ant 1.7, jEdit4.3, Ivy 2.0, arc, e-

learning, berek, forrest 0.8, zuzel, Intercafe and Nieruchomosci) have been taken under study which are the latest release of their retrospective versions and are collected from Promise Software Engineering Repository, which is available at [8] and shown in Table 1. Normalized (Duplicity remove) instance values of the retrospective datasets are also shown in this table.

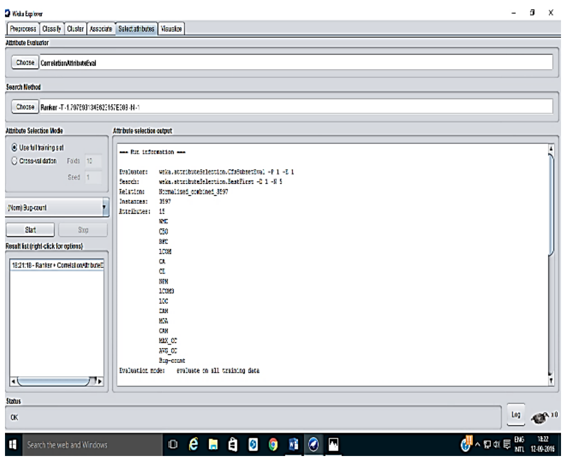
3.2 Metrics Description

Dependent and Independent Variables

Here Bug is dependent variable which shows that whether there is any bug in a class of the each data set or not. The independent variables are WMC DIT NOC CBO RFC LCOM, CA CE, LOC, LCOM3, NPM DAM MOA MFA CAM IC CBM AMC MAX_CC and AVG-CC.

3.3 Feature Selection

First of all we have removed the duplicate entries from each of the datasets mentioned in Table 1 and then combine all the normalized datasets and named it as Combined Dataset. Then feature selection process has been applied on the Combined Data set which is having 3597 instances. In this process correlation attribute evaluation is applied which is based on Pearson correlation method. Ranker attribute selection is selected in WEKAtool. Form this selection most prominent and deserving 14 metrics have been found, which are WMC, CBO, RFC, LCOM, CA, CE, MFA, LCOM3, LOC, DAM, MOA, CAM, MAX_CC and AVG_CC. BUG-Count value is dependent value, which is actually dependent on above mentioned 14 metrics. The most prominent metrics or features selection is shown in Figure 1.



Attribute Evaluator: CorrelationAttributeEval
Search Method: Ranker -T -1.7976931348623157E308 -N -1
Attribute Selection Mode: Use full training set; Cross-validation Folds 10 Seed 1
(Nom) Bug-count
Result list: 18:21:18 - Ranker + CorrelationAttributeE

```
=== Run information ===

Evaluator:    weka.attributeSelection.CfsSubsetEval -P 1 -E 1
Search:       weka.attributeSelection.BestFirst -D 1 -N 5
Relation:     Normalised_combined_3597
Instances:    3597
Attributes:   15
              WMC
              CBO
              RFC
              LCOM
              CA
              CE
              MFA
              LCOM3
              LOC
              DAM
              MOA
              CAM
              MAX_CC
              AVG_CC
              Bug-count
Evaluation mode:    evaluate on all training data
```

Status: OK

Figure 1: Feature selection

3.4 Classifier Selection

In this section all 14 different classifiers Naïve Bayes, Logistic Regression, LivSVM (Support Vector Machine), Multi-Layer Perceptron, SGD (Stochastic Gradient Descent), SMO (Sequential Minimal Optimization), Voted Perceptron, Attribute Selected Classifier, Classification Via Regression, Logit Boost, Tree Decision Stamp, Random forest, Random Tree and REP (Reduce Error Pruning) Tree are applied on validated combined dataset with 14 most prominent metrics at 10-cross validation. It is found that Logistic Regression is providing best result and it is now kept as the most prominent classifier.

3.5 AUC Measurement

ROC (Receiver operating Characteristics) curve is commonly used to measure the performance of classifier techniques in case of imbalanced classes.
Generally AUC (Area under receiver operating characteristics) is used for single numeric measurement to predict the potential of a classifier. Commonly a higher AUC is better. It also indicates that the used classifier within whole possible range of decision threshold has a higher sensitivity or true positive rate. It is found that higher the AUC value, the better is the performance.

4. Experimental Work

Proposed model with fourteen most prominent metrics WMC, CBO, RFC, LCOM, CA, CE, MFA, LCOM3, LOC, DAM, MOA, CAM, MAX_CC and AVG_CC with Logistic Regression Classifier is applied and it is trained on all the validated datasets and tested also on alone. Cross project training-testing process is applied. All the computed results are mentioned in Table No. 2.

Experiment No. 1

Training is applied on Camel 1.6 with 884 instances and cross testing is applied on all the datasets. It is found that the proposed model is providing minimum AUC value i.e. 0.469 on dataset jEdit 4.3 whereas Intercafe dataset is providing maximum AUC value i.e. 0.837.The Average AUC value provided by the model on dataset Camel 1.6 is 0.672.

Experiment No. 2

Model is trained on dataset Tomcat 6.0 (796) and tested on all the datasets. It is seen that Dataset Nieruchomosci is providing 0.375 AUC value which is minimum, whereas berek dataset is providing maximum AUC value i.e.0.887. The average value is 0.704.

Experiment No. 3

Training is applied on Ant 1.7 with 724 instances and cross testing is applied on all the datasets. It is found that the proposed model is providing minimum AUC value i.e. 0.594 on dataset Camel 1.6 whereas berek dataset is providing maximum AUC value i.e. 0.958.The Average AUC value provided by the model on dataset Ant 1.7 is 0.798.

Table 2: AUC measured on Training and Testing datasets using Logistic Regression Classifier

Training and Testing Both on Normalized Datasets																			
S No.	Trained on Datasets with AUC			Tested on Datasets with AUC															
	Datasets	**Instances**	**AUC**	**Camel1.6**	**Tomcat 6.0**	**Ant 1.7**	**jEdit 4.3**	**ivy 2.0**	**Arc**	**e-learning**	**berek**	**Forrest 0.8**	**zuel**	**Intercafe**	**Nieruchomosci**	**Combined Dataset**	**Min**	**Max**	**Average**
				884	**796**	**724**	**476**	**345**	**215**	**57**	**43**	**31**	**29**	**27**	**26**	**3597**			
1	**Camel1.6**	**884**	**0.719**	0.72	0.75	0.67	0.47	0.67	0.6	0.8	0.72	0.48	0.64	0.84	0.71	0.65	0.47	0.84	**0.7**
2	**Tomcat 6.0**	**796**	**0.833**	0.58	0.83	0.78	0.82	0.82	0.63	0.78	0.89	0.48	0.83	0.63	0.38	0.7	0.38	0.89	**0.7**
3	**Ant 1.7**	**724**	**0.834**	0.59	0.81	0.83	0.82	0.81	0.68	0.92	0.96	0.71	0.88	0.8	0.85	0.71	0.59	0.96	**0.8**
4	**jEdit 4.3**	**476**	**0.859**	0.5	0.51	0.64	0.86	0.59	0.55	0.74	0.31	0.47	0.27	0.55	0.43	0.56	0.27	0.86	**0.5**
5	**ivy 2.0**	**345**	**0.844**	0.59	0.83	0.79	0.8	0.84	0.68	0.81	0.89	0.67	0.82	0.76	0.4	0.71	0.4	0.89	**0.7**
6	**Arc**	**215**	**0.825**	0.54	0.77	0.56	0.16	0.41	0.83	0.75	0.24	0.9	0.2	0.42	0.32	0.51	0.16	0.9	**0.5**
7	**e-learning**	**57**	**1**	0.51	0.46	0.58	0.51	0.57	0.49	1	0.57	0.33	0.46	0.41	0.75	0.53	0.33	1	**0.6**
8	**berek**	**43**	**1**	0.51	0.65	0.65	0.51	0.66	0.62	0.49	1	0.48	0.75	0.65	0.9	0.6	0.48	1	**0.7**
9	**Forrest 0.8**	**31**	**1**	0.52	0.59	0.64	0.72	0.6	0.74	0.85	0.65	1	0.76	0.71	0.31	0.62	0.31	1	**0.7**
10	**zuel**	**29**	**0.998**	0.53	0.69	0.7	0.72	0.7	0.51	0.8	0.84	0.45	1	0.69	0.69	0.62	0.45	1	**0.7**
11	**Intercafe**	**27**	**1**	0.56	0.53	0.65	0.74	0.71	0.67	0.65	0.72	0.9	0.4	1	0.8	0.62	0.4	1	**0.7**
12	**Nieruchomosci**	**26**	**1**	0.63	0.73	0.63	0.68	0.68	0.44	0.31	0.92	0.28	0.84	0.36	1	0.61	0.28	1	**0.6**
13	**Combined Dataset**	**3597**	**0.723**	0.61	0.82	0.81	0.82	0.83	0.7	0.89	0.95	0.74	0.87	0.77	0.5	0.72	0.5	0.95	**0.8**
Average=0.661																			

Table 2: Comparative results of AUC between Proposed Model and MA Model

S. No	Datasets	Instances	AUC at 10-cross validation					
			Proposed Model	MA Model(maximum value)				Average
				NB	B NET	J48	RF	AUC of MA Model
1	**Camel1.6**	**884**	**0.719**	0.632	0.584	0.579	0.656	0.61275
2	**Tomcat 6.0**	**796**	**0.833**	0.781	0.761	0.722	0.803	0.76675
3	**Ant 1.7**	**724**	**0.834**	0.784	0.79	0.748	0.814	0.784
4	**jEdit 4.3**	**476**	**0.859**	0.747	0.491	0.398	0.572	0.552
5	**ivy 2.0**	**345**	**0.844**	0.761	0.78	0.634	0.76	0.73375
6	**Arc**	**215**	**0.825**	0.704	0.604	0.465	0.681	0.6135
7	**e-learning**	**57**	**1**	0.458	0.221	0.338	0.596	0.40325
8	**berek**	**43**	**1**	0.944	0.921	0.862	0.934	0.91525
9	**Forrest 0.8**	**31**	**1**	0.448	0.221	0.338	0.596	0.40075
10	**zuel**	**29**	**0.998**	0.856	0.873	0.781	0.877	0.84675
11	**Intercafe**	**27**	**1**	0.549	0.75	0.598	0.826	0.68075
12	**Nieruchomosci**	**26**	**1**	0.763	0.472	0.506	0.706	0.61175
Average AUC			**0.909333**					**0.6601**

Experiment No. 4

In cross project training-testing process based on dataset jEdit 4.3 it is found that zuzel is showing result 0.267 AUC value whereas 0.859 AUC is found on dataset jEdit 4.3. The average AUC value at this dataset is 0.535.

Experiment No. 5

Model is trained on dataset ivy 2.0 (345) and tested on all the datasets. It is seen that Dataset Nieruchomosci is providing minimum AUC value i.e. 0.400 whereas berek dataset is providing maximum AUC value i.e.0.887.The average AUC value is 0.737.

Experiment No. 6

Training is applied on Arc with 215 instances and cross testing is applied on all the datasets. It is found that the proposed model is providing minimum AUC value i.e. 0.159 on dataset jEdit 4.3 whereas Forrest0.8 dataset is providing maximum AUC value i.e. 0.897.The Average AUC value provided by the model on dataset Arc is 0.508.

Experiment No. 7

Model is trained on dataset e-learning (57) and tested on all the datasets. It is seen that Dataset Forrest 0.8 is providing minimum AUC value i.e. 0.328 whereas e-learning itself is providing maximum AUC value i.e. 1.0.The average AUC value is 0.552.

Experiment No. 8

In cross project training-testing process based on dataset berek it is found that Forrest 0.8 is showing result 0.483 AUC value whereas 1.0 AUC is found on dataset berek itself. The average AUC value at this dataset is 0.653.

Experiment No. 9

Model is trained on dataset Forrest 0.8 (31) and tested on all the datasets. It is seen that Dataset Nieruchomosci is providing minimum AUC value i.e. 0.313 whereas Forrest 0.8 itself is providing maximum AUC value i.e. 1.0.The average AUC value is 0.671.

Experiment No. 10

Training is applied on zuzel with 29 instances and cross testing is applied on all the datasets. It is found that the proposed model is providing minimum AUC value i.e. 0.448 on dataset Forrest 0.8 whereas zuzel itself is providing the maximum AUC value i.e.0.998.The Average AUC value provided by the model on dataset zuzel is 0.687.

Experiment No. 11

In cross project training-testing process based on dataset Intercafe it is found that zuzel is showing result 0.401 AUC value whereas 1.0 AUC is found on itself. The average AUC value at this dataset is 0.687.

Experiment No. 12

Model is trained on dataset Nieruchomosci (26) and tested on all the datasets. It is seen that dataset Forrest 0.8 is providing minimum AUC value i.e. 0.276 whereas Nieruchomosci itself is providing maximum AUC value i.e. 1.0.The average AUC value is 0.623.

Experiment No. 13

Finally training is applied on Combined dataset with 3597 instances and cross testing is applied on all the datasets. It is found that the proposed model is providing minimum AUC value i.e. 0.5 on dataset Nieruchomosci whereas berek dataset is providing maximum AUC value i.e. 0.951.The Average AUC value provided by the model on dataset Combined is 00.771.

Experiment No. 14

Comparative Study of Proposed Model with Mamdouh Alenzi Model

It is observed from Table 3 that for all the datasets our proposed model is providing better result in terms of AUC than the model provided by Mamdouh Alenzi.
Therefore from mentioned comparative analysis it is seen that the proposed model is providing on average AUC value i.e. 0.909 whereas Mamdouh Alenzi model is providing an average AUC which is 0.660.

Summary and Conclusions

From the experiments computed in previous section it is seen that some datasets are providing the similar AUC value (prediction result) and some are prompting the different result, while cross training-testing process is applied on all the datasets. Finally we got that our proposed model is providing an average AUC value i.e. 0.661.

Future Work

The results gathered from the experiments conducted in the paper is based on open source datasets. This study may be replicated on different data sets which may be some real one to generalize our findings.

References

Alenzi M. I (2014), Fault-Proneness of Open Source Systems: An Empirical Analysis, in International Conference (ACIT2014), University of Nizwa, Oman.

Amasaki S., Takagi Y., Mizuno O. and Kikuno T. (2003), A Baysian Belief Network for Asssessing the Likelihood of Fault content, Proceeding of the 14th IEEE International Symposium on software Reliability Engineering (ISSRE'03).

Catal C. (June 1994), Software fault prediction: A literature reviews and current trends, Proceeding of Expert Systems with Applications 38 (4626-4636) of Elsevier, 2011.

Chidamber S.R., and Kemerer C. F., A Metrics Suite for Object Oriented Design, IEEE Transactions on Software Engineering, Vol. 20, No.6.

Churcher N.I., and Shepperd M.J. (March 1995), Comments on – A Metrics Suite for Object Oriented Design, IEEE Transactions of Software Engineering, Vol. 21, No 3.

Gondra I. (2008), Applying Machine Learning to Software Fault-Proneness Prediction, The Journal of Systems and Software, Elsevier, Vol. 81, pp. 186-195.

Henderson-Sellers, B. (1996), Object-Oriented Metrics measures of Complexity, Prentice Hall.

https://code.google.com/p/promisedata/w/list.

https://code.google.com/p/promisedata/wiki/MarianJureczko.

Jureczko M. and Madeyski L. (2010), Towards identifying software project clusters with regard to defect prediction, Proceedings of the 6th International Conference on Predictive Models in Software Engineering (Promise-10) held at Timisoara, Romania, published in ACM, New York, USA. http:// doi.acm.org/ 10.1145/1868328. 1868342.

Karabulut E. M., Ozel S.A. and Ibrikci T. (2012), A Comparative study on the effect of feature selection on classification accuracy, Procedia Technology, Elsevier, Vol. 1, pp. 323-327

Martin R. (1994), O O Design Quality Metrics-An Analysis of Dependencies, Proceeding of Workshop Pragmatic and Theoretical Directions in Object-Oriented Software Metrics, OOPSLA'94.

Newman D. J. , Hettich S., Blake C. L. and Merz C. J. (1998), UCI Repository of machine learning databases, University California Irvine, Department of Information and Computer Science

Olague H.M.,Etzkorn L.H., Gholston S. and Quattlebaum S. (June 2007), 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 Transaction on Software Engineering, Vol. 33, No. 6.

Singh Y., Kaur A. and Malhotra R. (2009), Software Fault Proneness Prediction Using Support Vector Machine, Proceedings of the World Congress on Engineering, Vol. 1.

Witten I. H. and Frank E. (October 1999), Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann,. http://www.cs.waikato.ac.nz/ml/weka/.

Xing F., Guo P. and Lyu M. R. (2005), A Novel Method for Early Software Quality Prediction Based on support Vector Machine, Proceeding of the 16th IEEE International Symposium o software Reliability Engineering (ISSRE'05).