Journal of Basic and Applied Engineering Research
p-ISSN: 2350-0077; e-ISSN: 2350-0255; Volume 4, Issue 1; January-March, 2017, pp. 5-9

http://www.krishisanskriti.org/Publication.html

Identification of Error Prone Classes and Quality Estimation using Semi-supervised Learning Mechanism with Limited Fault Data

Aarti[1], Geeta Sikka[2] and Renu Dhir[3]

[1,2,3]Dr B R Ambedkar NIT Jalandhar
E-mail: [1]aarti.1208@gmail.com, [2]sikkag@nitj.ac.in, [3]dhirr@nitj.ac.in

Abstract—*We address the problem of limited faulty data as detection models are always trained using the previous releases. Moreover, small set of trained data with limited faulty information is not sufficient to predict the future versions. It also has been investigated that software metrics plays significant role for fault prediction. Our study focuses on relationship between fault-proneness and software metrics using Expectation and maximization (EM) based semi-supervised learning scheme. The quality model is evaluated with multiple datasets from PROMISE repository and comparison is made with the different quality model.*

Keywords*: Clustering; quality; semi-supervised; fault-pronness*

1. INTRODUCTION

Software quality assurance is becoming more crucial activity and major subset of testing, verification, validation, fault tolerance and prediction. Software testing is time consuming task and sometimes, errors may still left in the software projects even after testing. It becomes very difficult to do changes after release of software[6]. Software metrics plays vital role for locating the error-prone modules. Software metrics can be used to improve the quantitative nature of model. But there is needed to choose the proper quality model before testing that can reduce effort, resources and time with limited historical data [8].

The quality prediction can be obtained either by predicting the number of faulty projects [1][11] or predicting the fault-prone class[2]. Most of the previous fault-prediction mechanism focussed on supervised mechanism and these technique uses these information to identifies the previously similar known faults associated with the projects called as classification [3][4]. Our main goal is to identify the software metrics and essential parameters that effect proneness of faults. The quality classification model classifies the dataset into fault-prone (fp) and non fault-prone (nfp) modules.

The dataset with historical information are used to train the model to identify the faulty modules in the newer projects. But, supervised training with labels may not always yield a good quality software[6]. Some of the issues have been noticed such as: Firstly, the process of collecting faulty data totally dependent on the some project components. Secondly, collection of defect data may be error-prone. Thirdly, in case of multi-release software projects, collection of data can extract the qualitative data based on the small portion of module due to the practical issues of iterative versions.

In this paper, we investigate the semi-supervised clustering technique [5][7] known as Expectation and Maximization (EM) algorithm which uses the iterative augmentation of unlabeled modules with their estimated class labels.

The rest of this paper is organized as follows: Section 2 gives description of data sets and the evaluation criteria opted for quality model. Section 3 reviews on prerequisite knowledge based on modeling techniques have been discussed. Section 4 so discuss about the methodology used for fault proneness with design of experiment. Section 5 gives the results and comparison of proposed work. In the end the conclusion is made in Section 6.

2. DATA SETS AND METRICS

The datasets are taken from PROMISE repository where various research groups contribute and which are publicly available. We have considered the data from SOFTLAB which is software company dealing with embedded controller applications. Table 1 provides the detailed information about three projects considered in this project with the defect information. These projects are single version in their respective research groups.

Table 1: Dataset

System	Languages	No of Classes	% Defect
ant 1.3	Java	125	16
jedit 4.1	Java	312	25.32

The projects considered in Table 1 have various matrices but we have considered those which are common in all analyzed projects. The set of selected metrics are depicted in Table 2. The data of software defects have been calculated by ckjm tool [12] along with other software metrics.

We used logistic regression and machine learning techniques that used to estimate the faults during various stages of software development life cycle. The prediction capability of the model is based on chidamber and kemerer[9][10] metrics tabulated in Table 2.

The focus of our research is identifying the relationship between OO design metrics and fault proneness. The classification model is constructed with logistic regression and machine learning methods to distribute elements in two categories of fault-prone and not fault-prone. The ability of fault-proneness model is evaluated on the basis of classification / prediction of projects in fault prone and non fault prone modules.

Table 2: CK metrics

Software Metric	Description
CBO(Coupling between Objects)	Two classes are said to be coupled if one class calls method of other class. Inheritance and polymorphism are used in it.
DIT(Depth of inheritance)	Maximum length of class hierarchy that counts the number of ancestor nodes.
LCOM(Lack of cohesion among methods)	Measure degree of dissimilarity of methods in a class along with attributes
NOC(Number of children)	Counting the number of immediate decedents of the class
RFC(Response for a class)	Number of methods that can be executed in response to a message
WMC(Weighted method count)	summing up of complexity of all methods
SLOC(Source line of code)	Total numbers of lines

a) Evaluation parameters

We will use the commonly used performance measures: accuracy, recall, specificity and precision to evaluate the prediction. The first metric we used is Precision. Precision defines where repeated measurement shows same result under unchanged condition. It is given by the Eq. 1

$$Precision = \frac{TP}{FP+TP} \quad (1)$$

The second metric to consider is Recall (also called probability of detection, PD) which defines how many relevant items that are to be identified. It is given by Eq. 5.

$$Recall = \frac{TP}{FN+TP} \quad (2)$$

The third metric is Probability of false alarms PF. It is calculated as Eq. 6.

$$Probaility\ of\ false\ alarm(PF) = \frac{FP}{TN+FP}. \quad (3)$$

The fourth metric included is accuracy which defines as proportion of predicted fault-prone fault that are inspected out of all module given in Eq. 4

$$Accuracy = \frac{TN+TP}{TP+TN+FP+FN} \quad (4)$$

The fifth metric is the Specificity which identifies how classifier identifies negative labels. This is calculated according to Eq. 5

$$Specificity = \frac{TN}{FP+TN} \quad (5)$$

3. PREREQUISITE KNOWLEDGE

3.1 EM algorithm

EM algorithm is iterative method for maximizing the posteriori estimations of parameters to deal with the unobserved data. The EM statistical model given with X as observed data and Z as the set of unobserved data with θ be unknown parameters and Likelihood function can be represented as: $L(\theta; X, Z) = p(X, Z|\theta)$. Maximum likelihood of function (MLE) is represented by the marginal likelihood of the observed data as $L(\theta; X) = p(X|\theta) = \sum_Z p(X, Z|\theta)$.. EM technique performs two computation to find the MLE as follows:

Expectation (E): It creates function for the expectation of log-likelihood for estimation of parameters under the current estimation of parameters$\theta^{(t)}$:

$$Q(\theta|\theta^t) = E_{Z|X,\theta^t}[\log L(\theta; X, Z)] \quad (6)$$

Maximization (M): It maximizing log-likelihood found on Expectation step.

$$\theta^{t+1} = argmax_\theta\ Q(\theta|\theta^t) \quad (7)$$

To apply this algorithm, following steps are used:

Steps for algorithm

- First initialize parameter θ to some random values.
- Compute the best value of Z
- Use the value of Z to estimate the value of θ.
- Iterate the step 2 and 3 until convergence.

Logistie regression

Regression analysis is statistical technique to find the relationship among OO metrics. The main focus for using regression is to assets the relationship between dependent and independent variables. It is basically used to estimates the likelihood of dependent variable given by independent variables. In this, dependent variable(Y) is given by function of independent variable(X) with some unknown parameters (β) called regression function.

$$Y \approx f(X, \beta) \quad (8)$$

It is also used to interpret which among the independent (OO metrics) are related to the dependent variable. Two techniques of linear regression are used in this study: univariate and multi-variate analysis. Univariate is the simplest method to understand the impact of each metric on fault proneness. Mutivariate analysis is to observe the simultaneous analysis of more than one target variable. To use both regression techniques, we opt logistic regression method because dependent variable is categorical in our study. The estimation of probability for binary independent variable based on one or more independent variables (OO metrics) is done by the logistic model.

4. EXPERIMENTAL DESIGN

This section discusses about prediction of faults using clustering approach. The comparative analysis is done with well known machine learning techniques.

Experiment 1: The following steps give detailed description of experiment:

Input: Clustered data set extracted with CK parameters

Output: Classification model to identify parameters like precision, recall etc.

Process of Experiment

Step 1: Extract CK metrics from dataset

Step2: Validate extracted metrics using univariate regression analysis. (using B and standard errors.)

Step 3: Apply EM clustering on reduced dataset.

Step 4: Apply logistic regression to classify testing file

Different experiments are conducted to evaluate the effectiveness of the proposed scheme with contrast from the other techniques. The following statics are used for each significant metrics using logistic regression in our research:

B: It defines the coefficient for the constant (intercept) in the null model and coefficients for other attributes. These are log-odd units that used to define relationship between dependent and independent variables. These coefficients are in log-odd units that are difficult to interpret, so these are converted to odds-ratios (Exp (B)).

$$log\left({}^{p}/_{1-p}\right) = b_0 + b_1.x_1 + b_2.x_2 \ldots\ldots\ldots b_n.x_n \quad (9)$$

Exp(B): It is the exponentiation of the B coefficient, called as odds ratio. This value is taken as defaults because of easier interpretation than the log-odds units coefficients.

Goodness of fit: It is used to evaluate the fitness of logistic regression on concurrent measures of sensitivity and specificity. The curve is generated between sensitivity (y-axis) and specificity (x-axis). This curve is called receiver operating characteristics (ROC). The area ranges from 0.5 to 1.0 where larger value indicates better fit.

Statistical significance: It measures the significance levels of the coefficients of attributes measured using logistic regression. The higher is value of significance; lower the estimated impact of independent attribute.

5. RESULT ANALYSIS

We use ant-1.3 and jedit-4.1 dataset for prediction of fault-proneness classes. Firstly CK extracted metric are clustered using EM into 5 clusters. Then, they are classified using statistical (logistic) as well as machine learning (multilayer-preceptron and J48). Table 3 gives descriptive analysis of Ant 1.3 dataset. We investigated value of NOC remains zero at 25, 50 and 75 percentile. The standard error is lower for WMC and higher for the LOC. **Sig. (p-value)** indicates the statistical significance of the regression model. If p value is less than 0.05, then we considered regression model as statistically significant and if it greater than 0.05, then it is considered as non-statistically significant. Std. Error indicates that number of observed values falls below the regression line. On the other side, it also discusses about how wrongly regression model uses entities of response variable

Table 3: Descriptive statistics of Ant 1.3

	Mean	**Std. Error**	**Std. De**	**Percentiles**		
				25	***50***	***75***
wmc	10.59	0.93	10.36	4.00	8.00	14.00
dit	2.28	0.11	1.28	1.00	2.00	3.00
noc	0.58	0.29	3.25	0.00	0.00	0.00
cbo	10.43	1.33	14.89	4.00	7.00	10.50
rfc	34.37	2.60	29.02	15.0	28.0	46.50
lcom	69.32	23.2	259.28	1.00	6.00	46.00
loc	301.5	30.3	339.11	83.	168	421.50
bug	0.26	0.06	0.69	0.00	0.00	0.00

Table 4 provides the coefficient (B), statistical significance (sig), and standard error (SE) values for each measure. The quality prediction are said to be successful if they meet criteria of less error rate. The highest value of S.E. is found for the DIT. From value of S.E., we found that this metric cannot provide much contribution. We found that two out of seven metrics (NOC and LOC) are found insignificant as their Sig value is higher than 0.5.

Table 4: Statistical parameters using regression

	Unstandardized Coefficients		**Sig.**
	B	***Std. Error***	
(Constant)	-.003	.137	.982
wmc	-.008	.015	.585
dit	-.085	.048	.077
noc	-.010	.020	.618

cbo	.004	.005	.487
rfc	.015	.006	.011
lcom	-.001	.000	.043
loc	.000	.000	.689

Table 5 discusses about the classification results using different techniques. The ROC parameters and F-measure are found to be higher in logistic based classification model and precision is higher in case of hierarchical technique (J48) .

Table 5: Prediction result using logistic and machine learning techniques

	TP	FP	Precision	Recall	F-Measure	ROC
Logistic	0.8	0.08	0.83	0.83	0.83	0.94
Multi layer-preceptron	0.8	0.08	0.792	0.8	0.793	0.94
J48	0.86	0.04	0.85	0.86	0.858	0.89

Table 6 elaborates descriptive analysis of Jedit 4.1 dataset. We investigated value of NOC again remains zero at 25, 50 and 75 percentile. This means this metric does not have impact on the prediction strategy. The standard error is lower for DIT and NOC and higher for the LOC. Table 7 indicates the statistical Table 7 describes the statistical parameters using regression, all the metrics except LOC are found significant as there Sig value is less than 0.05. Some of the metrics have negative B value means that larger the value of metric have higher impact on the prediction strategy.

Table 6: Descriptive analysis of Jedit-4.1

	Mean	Std. Error	Std. Dev	Percentiles		
				25	***50***	***75***
wmc	13.13	1.74	30.73	4.00	6.00	12.00
dit	2.74	0.12	2.12	1.00	2.00	4.00
noc	0.48	0.16	2.76	0.00	0.00	0.00
cbo	12.98	1.07	18.95	5.00	8.00	13.75
rfc	39.87	3.29	58.03	10.0	23.0	46.50
lcom	187.89	61.4	1084.9	1.00	5.00	31.75
loc	490.66	88.9	1571.9	62.5	176	415.8
bug	0.70	0.11	1.89	0.00	0.00	1.00

Table 7: Statistical parameters using regression

	Unstandardized Coefficients		Sig.
	B	***Std. Error***	
(Constant)	-.029	.117	.807
wmc	-.037	.010	.000
dit	-.103	.034	.003
noc	-.018	.024	.446
cbo	.012	.005	.026
rfc	.032	.003	.000
lcom	.001	.000	.000
loc	5.000E-05	.000	.694

Table 8 discusses about the classification results using logistic and machine learning technique. The ROC parameters and F-measure are found to be higher in logistic as well as multi layer preceptron based classification model. The logistic model and NN model gives similar result on this dataset and hierarchical model lack in fp rate, precision, and ROC values.

Table 8: Prediction result using logistic and machine learning techniques

	TP	FP	Precision	Recall	F-Measure	ROC
Logistic	0.98	0.01	0.98	0.98	0.98	0.99
Multi layer-preceptron	0.98	0.01	0.98	0.98	0.98	0.99
J48	0.97	0.03	0.97	0.97	0.97	0.98

6. CONCLUSION

We investigated that the object-oriented metrics plays most important role to predict the fault-proneness in software projects. To check error prediction accuracy, we used ROC curve parameter for different techniques (statistical as well as machine learning technique). To deal with the missing data, EM provides the likelihood parameter for the missing data. So, we can conclude that predicted model can provides the satisfactory solution.

REFERENCES

[1] T.M.Khoshgoftaar, & N. Seliya, Tree-based software quality models for fault prediction. *In Proceedings of 8th international software metrics symposium, Ottawa, Ontario, Canada,* June 2002, pp. 203–214, IEEE Computer Society.

[2] T.M. Khoshgoftaar & N. Seliya, Analogy-based practical classification rules for software quality estimation*. Empirical Software Engineering Journal*, vol. 8, no. 4,2003, pp. 325–350

[3] V. Basili, L. Briand L and W. Melo, A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering,* vol. 22, no. 10: 1996, pp. 751–761

[4] J.M. Bieman and B.K. Kang ,Cohesion and Reuse in an Object-Oriented System. *Proceedings of the Symposium on Software reusability-ACM 2*0: , 1995, pp. 259-262

[5] S. Bhattacharya , S. Rungta and N. Kar N, Software fault prediction using fuzzy clustering & genetic algorithm. *International Journal of Digital Application & Contemporary Research*, vol. 2, no. 5, 2013, pp. 1-7

[6] N. Bettenburg , M. Nagappan, and A E Hassan, Think locally, act globally: Improving defect and effort prediction models*. In Proc. 9th IEEE Working Conference on Mining Software Repositories (MSR), Zurich, Switzerland*, 2012, pp. 60–69

[7] C. Catal, U. Sevim, and B. Diri, B. Clustering and metrics threshold based software fault prediction of unlabeled program modules*, In proceeding of Sixth International Conference on Information Technology*: New Generations, 2009, pp. 199-204.

[8] C. Catal, Software fault prediction: A literature review and current trends, *Expert Systems with Applications*, vol. 38, no. 4, 2011, pp. 4626-4636.

[9] S. Chidamber and C. Kemerer, A Metrics Suite for Object-Oriented Design, *IEEE Transaction of Software Engineering*, vol. 20, no.6, 1994, pp. 476-493.

[10] K. El, Emam, S. Benlarbi, N. Goel and S.N. Rai, The confounding effect of class size on the validity of object-oriented metrics, *IEEE Transactions on Software Engineering*, vol. 27, no.7, 2001, pp. 630–650.

[11] T. Hall, S. Beecham, D. Bowes, D. Gray and S. Counsell, A systematic literature review on fault prediction performance in software engineering, *IEEE Transaction Software Engineering*, vol. 38, no.6, 2012, pp. 1276-1304.

[12] S. Chidamber and C.Kemerer C , A metrics suite for object-oriented design. *IEEE Transactions on Software Engineering*., vol. 20, no.6 , 1994, pp476–493