EXPERIMENTAL STUDY ON SOFTWARE FAULT PREDICTION USING MACHINE LEARNING MODEL

Thi Minh Phuong Ha
School of Information and Communication Technology – The University of Danang
Danang, Viet Nam
htmphuong@sict.udn.vn

Duy Hung Tran
The University of Danang – University of Science and Technology
Danang, Vietnam
hungtran1196@gmail.com

LE Thi My Hanh
Information Technology Faculty
The University of Danang – University of Science and Technology
Danang, Vietnam
ltmhanh@dut.udn.vn

Nguyen Thanh Binh
Information Technology Faculty
The University of Danang – University of Science and Technology
Danang, Vietnam
ntbinh@dut.udn.vn

***Abstract*—Faults are the leading cause of time consuming and cost wasting during software life cycle. Predicting faults in early stage improves the quality and reliability of the system and also reduces cost for software development. Many researches proved that software metrics are effective elements for software fault prediction. In addition, many machine learning techniques have been developed for software fault prediction. It is important to determine which set of metrics are effective for predicting fault by using machine learning techniques. In this paper, we conduct an experimental study to evaluate the performance of seven popular techniques including Logistic Regression, K-nearest Neighbors, Decision Tree, Random Forest, Naïve Bayes, Support Vector Machine and Multilayer Perceptron using software metrics from Promise repository dataset usage. Our experiment is performed on both method-level and class-level datasets. The experimental results show that Support Vector Machine archives a higher performance in class-level datasets and Multilayer Perception produces a better accuracy in method-level datasets among seven techniques above.**

Keywords—software metrics, fault prediction, machine learning, NASA, PROMISE.

I. INTRODUCTION

Dealing with faults is a significant problem in software development process. The existence of serious errors contain high potential risks which are causes of project failure. In addition, it is also a cause for decreasing the quality of project and increasing the maintenance cost [15]. Detecting errors in early phase leads to reduce the time and cost of development. Hence, fault prediction is a crucial activity prior to the developing and testing phase contributes to the success of the overall project. Fault prediction activities offer developers more chances to focus on defective modules and resolve them effectively. In order to predict defects in early stage, many software fault prediction models have been constructed for practitioners to find the modules identified as faulty in early stage. Later, these models can be applied in future real systems to detect faulty modules. We can use two approaches are software metrics and bug report to build a predictive model. Many researches proved that software metrics are useful for software fault prediction [24]. In this paper, we focus on fault prediction based on metrics. First, metrics are measurable properties for identifying error prone parts. Many researches have investigated the significant relationship between software metric and fault proneness. For instance, Enam [1] have considered metrics had the strongest association with fault-proneness in software. Several metrics were proposed for Object Oriented (OO) software ([3], [7]). Second, predictive models were built by using historical bug data from similar projects. Using software metrics to identify errors in early development cycle is considered for fault prediction. However, some metrics are not effective for defecting faults. Therefore, it is important to identify good metrics for defecting faulty modules.

Many approaches have been investigated to predict faulty classes such as statistical techniques and machine learning techniques. Statistical techniques are to classify classes as either defective or non-defective. Otherwise, machine learning techniques are superior to traditional statistical techniques for classification of faults. Thus, in order to make predictive models more accurate *i.e* building models is capable of predicting modules containing numerous errors, many machine learning techniques have been used for predicting faults using software metrics to build a high performance prediction model. Currently, a number of machine learning techniques using software metrics consist of static metrics and process metrics to predict faults [10,17,21], such as Logistic Regression, K-nearest Neighbors, Decision Tree, Random Forest, Naïve Bayes, SVM and Multilayer Perceptron. These techniques were proposed to be applied in different contexts for predicting the probability of possible errors that can occur in a module. They were assessed as the good techniques in fault prediction. In this work, we investigate which one is the best in terms of performance and accuracy. The main contribution of this paper focus on analyzing the performance of some machine learning techniques in fault prediction. We conduct an experiment to compare the efficiency of seven techniques: Naïve Bayes Decision Tree, Random Forest, Logistic Regression, K-nearest neighbors, Multilayer Perceptron, and SVM on both class-level and method-level datasets from Java projects which are part of the PROMISE repository. In this study, we use thirteen datasets including JM1, PC4, KC2, MC1, KC1, PC3, CM1, MW1, PC1, Class, MC2, KC3 and PC2 extensively utilized to predict fault. In addition, we examine the effectiveness of these techniques in terms of F_1, Area under the Curve (AUC) and Accuracy. For ensuring reliability of the results, Receiver Operating Characteristic (ROC) is considered to analyze the performance of the techniques [25].

Experimental results show that among traditional techniques, Support Vector Machine achieves high accuracy and the F_1 value outperforms that of other techniques at class-level. For method-level datasets, Multilayer Perceptron outperforms all the others techniques in terms of accuracy, F_1 and AUC. Thus, Multilayer Perception is the best technique to predict faults at method-level and Support Vector Machine is suitable to predict errors at class-level datasets.

The paper consists of the five sections. In section 2, an overview of machine learning approaches and the software metrics is provided for predicting fault. Section 3 presents the details of the experimental design. The results of experiments and discussion are given in session 4. Finally, the last section give conclusions of the research and future works.

978-1-7281-3003-3/19/$31.00 ©2019 IEEE

II. BACKGROUND

A. Software metrics in fault prediction

Object-oriented metrics are often used to assess the testability, maintainability or reusability of source code. Recently, object-oriented metrics are also employed to predict faults. Many object-oriented metrics are studied and proposed to access the quality of software. This section presents briefly some following typical metrics which are based on the popular object-oriented characteristics as encapsulation, inheritance, polymorphism and abstraction.

Chidamber and Kemerer [7] proposed six metrics based on classes and hierachical relationships between classes, called CK metrics. CK metrics are most widely used and inherited by many researchers to further develop.

Lorenz and Kidd [26] proposed four metric groups for class level based on size and inheritance; and three metrics for method level based on size, complexity and number of parameters.

Harrison et *al*. [12] proposed six metrics to assess the object-oriented properties of the design including method hiding, attribute hiding, method inheritance, attribute inheritance, coupling and polymorphism.

The object-oriented metrics have been employed by researchers to predict faults. In [6], Emam et *al*. constructed a prediction model based on object-oriented metrics. Their results showed that the prediction model had a high accuracy. Aggarwal et *al*. [8] indicated the metrics Coupling Between Object Classes (CBO), Response For a Class (RFC), Weighted Method per Class (WMC) are strongly correlated to errors and high accuracy prediction in object-oriented software. In [9], the authors showed that the metrics Response For a Class (RFC), Weighted Method per Class (WMC) and Lake Of Cohesion in Methods (LCOM) are closely related to predict the fault prone module.

Some common OO metrics are described in Table I.

TABLE I. OO METRICS

Metric Name	Definition
Weighted Methods per Class (WMC)	The total of the complexities of all methods in a class
Number of Children (NOC)	The number of immediate subclasses inheriting directly from a given class.
Depth of Inheritance Tree (DIT)	The maximum of steps in the inheritance tree from a class node to the root of tree.
Coupling Between Objects (CBO)	CBO for a class measure for a class to how much other classes it is coupled
Response for a Class (RFC)	Number of other classes directly called by any methods on a given class.
Source lines of code (SLOC or LOC)	Number of lines of the program's source code.
Lacks of cohesion of method (LCOM)	Number of methods in a class that access one or more of the same attributes.

There are a lot of object-oriented metrics are investigated to predict errors. According to Ruchika [24], not all of OO metrics can be used effectively for detecing faults. It would be valuable to identify valuable metrics relate to fault-proneness.

B. Machine learning techniques for fault prediction

Several approaches have been studied to construct fault prediction model including statistical analysis, expert estimation, machine learning and mixed technique, etc. Amongst them, machine learning has been proposed as the most efficient technique for predicting the defect proneness of modules [11]. The widely used machine learning techniques for predicting software defects consist of Logistic Regression [8], K-nearest Neighbors [16], Naïve Bayes [17], Decision Tree [20], Random Forest [19], Support Vector Machine [20] and Multilayer Perceptron [21].

Decision Tree [28] is a predictive model called as a hierarchical one which can be used to represent regression and classification models in form of a tree. It includes decision node having at least two branches and leaf node showing a classification and decision. The popular algorithm of Decision Tree is C4.5.

Logistic Regression [29] is used to predict the dependent variable from a set of independent variables. In Logistic Regression, univariate logistic regression is used to construct a prediction model for finding faulty classes. In [23], Zhou and Leung identified WMC, CBO and SLOC are good metrics for predicting errors.

Multilayer Perceptron (MLP) [30] is as part of artificial neural network. MLP uses supervised learning technique called back-propagation algorithm to construct a neural model from history data.

K-nearest Neighbors (KNN) [27] chooses the closest neighbors in the training data to export the target data. This algorithm finds the k-closest training points according to some metrics like Euclidean [4].

Support Vector Machine is another machine learning technique which Elish and Elish [2] used to propose a model for predicting the probability of error by combining the k-means kernel and the DSMOPSO-D techniques with Halstead metrics. They showed that their approaches reached best values for AUC in large datasets and that Halstead metrics were closely correlated with errors compared to others metrics.

Naïve Bayes [18] is a classification technique with independence assumption between independent variables. Among techniques, this technique is also reported to archive better performance in predicting errors [9].

Random Forest is another machine learning technique used in data mining that Guo [14] proposed to predict faults in software system. Catal and Diri [5] reported that Naïve Bayes is the best technique for predicting errors if there are small datasets, whereas Random Forest technique is the best option if there are large datasets.

C. The Fault Prediction Process

Most of the machine learning techniques used in software fault prediction are supervised learning techniques.

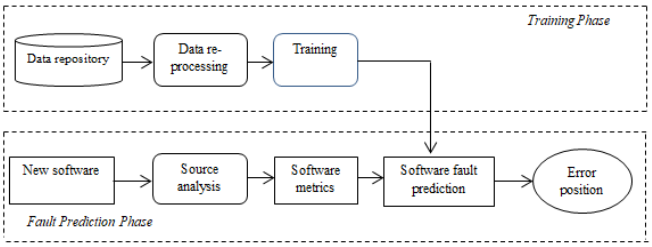

Fig. 1. The process of fault prediction

Before implementing the fault prediction phase, the datasets should be processed in a training phase. The process of fault prediction is shown in Fig. 1.

1. Data Collection

Data collection is performed in the first stage of software fault prediction. In this work, the training datasets was collected from *tera - Promise* data source [13]. This training material is organized into two parts: data for predicting object-oriented faults and method-oriented faults.

2. Pre-processing

Training data influences greatly on the fault prediction result of supervised machine learning techniques. To increase the accuracy of the techniques, data should be preprocessed before training. The steps of the preprocessing include standardizing data and reducing number of dimensions.

3. Training software fault prediction

The fault prediction problem is a Binary Classification, in which the error source code belongs to the positive class and the remaining source code belongs to the negative class.

III. EXPERIMENTAL DESIGN

From previous studies [22], researchers proved that machine leaning technique is the most successful approach to predict the fault proneness. Our objective is evaluation empirically predicting faults of the traditional machine learning techniques using public projects in Java on the NASA defect datasets available from PROMISE repository and applying a cross-validation approach. In *k-fold* cross approach, our original dataset is randomly divided into k mutually exclusive $D_1, D_2, \ldots, D_k$ of approximately equal size. The technique is trained and tested with k times. In each time, one fold is used to test and the remaining folds are used to train the model. The process is repeat until each fold of the *k-fold* is used as the test set. The overall accuracy is estimated by averaging the results of each loop.

The experimental study is conducted into the following steps:

- Chooosing the machine learning techniques: Logistic Regression, K-nearest Neighbors, Decision Tree, Random Forest, Naïve Bayes, Support Vector Machine and Multilayer Perceptron; Choosing the dataset from NASA MDP PROMISE.
- Using cross-validation approach to evaluate the effectiveness of the machine learning techniques in predicting faults.
- Identifying the acuracy, precision, recall, F-measure, AUC for each technique and analyzing the ROC curve comparison between diferent techniques at class-level and method-level.
- Finally analyzing the results and determining which techniques are effecive for fault prediction.

A. Dataset Preparation

The experimentation depends on the quality of used datasets, hence choosing the datasets plays an important role. According to Catal and Diri [11], by using PROMISE repository [13], the results of prediction are more reliable. In fault prediction, the class-level metrics and object-level metrics are used regularly to analyze the performance of machine learning techniques.

B. Metrics considered for evaluating performance of techniques

In order to examine the effectiveness of seven popular techniques for defect prediction, well-known performance metrics are used such as Accuracy Recall, AUC and F-measure. To increase the reliability of experimental results, the area of the receiver operating curve (ROC) values are also analyzed. Table II describes a confusion matrix for fault prediction performance.

- True Positive (TP): indicate that the positive samples were correctly labeled by the classifier.
- True Negative (TN): indicate that to the negative samples were correctly labeled by the classifier.
- False Positive (FP): indicate that the negative samples were incorrectly labeled as positive.
- False Negative (FN): indicate that the positive samples were mislabeled as negative.

TABLE II. CONFUSION MATRIX

Observed	Predicted	
	Faulty	Non-Faulty
Fault Prone	TP	FP
Not Fault Prone	FN	TN

- Accuracy is defined as number of positively predicted instances divided by total number of instances. In the experiment, the value of accuracy is ranged from 0 to 1.
- Precision shows the correctness of predicted results. It is calculated as the ratio between number of error instances for a particular class including faulty and non-faulty. The high precision shows that the predictive model is more accurate.

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

- Recall measures the ratio of correct instance for a particular class including faulty and non-faulty. Recall also presents probability of detection.

$$\text{Re}call = \frac{TP}{TP + FN} \quad (2)$$

- F_1: Indicate the average of precision and recall for a particular class (faulty or non-faulty). F_1 assesses whether the increase in precision (recall) is greater than the reduction in recall (precision).

$$F_1 = 2\frac{precision.recall}{precision + recall} \quad (3)$$

- ROC curve and AUC curve: Beside three above metrics, the effectiveness of the predictive models is evaluated by the ROC curve. The nearer the curve gets the left border of the ROC space and then gets the top border of the ROC space, the more correct the test. Otherwise, the

curve approaches the diagonal line and AUC value is small; the fault prediction model brings low efficiency.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

Initially, we collect the results of predictive techniques applied to two different datasets which are class-oriented datasets and object-oriented datasets. The collected results are calculated by using *tenfold cross-validation* approach. Accordingly, the datasets are divided into 10 mutually exclusive folds whose size is approximately equal so that the error rate/ no error in each part are the same. The training of fault prediction model is repeated by 10 times. In each loop, 1 subset will be used to test and the other subsets (9 subsets left) are training datasets. The final result is average of 10 above loops results.

A. Class-level fault-proneness prediction results

We have applied the GridSearch technique to choose the appropriate parameters for particular technique and datasets. In the GridSearch, based on the combination of different parameter values, the results returns the combination with the highest accuracy. Basing on the GridSearch technique, we chose the parameters of class-level datasets for each machine learning technique (MLT) as follows:

TABLE III. THE PARAMETERS OF CLASS_LEVEL DATASETS

MLT	Parameters
LogisticRegression	c=1.0, max_iter=1000
K-nearest Neighbor	n_neighbors=5, weight='uniform'
Decision Tree	min_samples_leaf=1 min_samples_split=2
Random Forest	min_samples_leaf=1 min_samples_leaf=2 n_estimators=10
SVM	c=0.9, kernel='sigmoid'
Multilayer Perceptron	activation='tanh', alpha=0.0002

Table IV summaries the results obtained from applying seven traditional techniques on class-level datasets.

TABLE IV. CLASS-LEVEL FAULT – PRONENESS PREDICTION RESULTS

Metrics / Algorithm	F1	AUC	Accuracy
Logistic Regression	0.34	0.68	0.64
K-nearest Neighbors	0.43	0.55	0.52
Decision Tree	0.41	0.56	0.55
Random Forest	0.43	0.61	0.56
Naïve Bayes	0.32	0.61	0.65
SVM	**0.48**	0.58	**0.63**
Multilayer Perceptron	0.44	0.62	0.58

The values of the best results from these techniques are bold in the table. We can realize that SVM achieves the best F_1 value that reaches to 0.48. Moreover, SVM also has the high accuracy (63%) and this is not significantly lower than the Logistic Regression's accuracy (64%) and Naïve Bayes's accuracy (65%). In summary, the prediction performance of Support Vector Machine is better than other techniques.

We also uses the ROC graphs to analyze the performance of these techniques. In Fig. 2, the ROC result shows that Logistic Regression is the best possible technique with the ROC curve near to the line y=1.

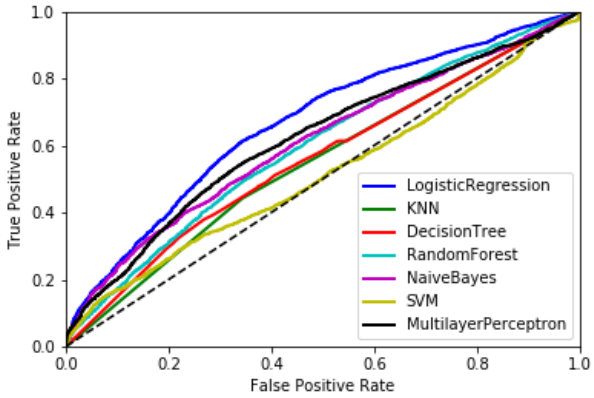

Fig. 2. Comparing the ROC curves between the techniques at class-level

B. Method-level fault-proneness prediction results

This section presents the experimentation on method-level datasets. Similarly, we chose the parameters for method-level datasets as follows:

TABLE V. THE PARAMETERS OF METHOD_LEVEL DATASETS

MLT	Parameters
LogisticRegression	c=0.9, max_iter=1000
K-nearest Neighbor	n_neighbors=5, weight='uniform'
Decision Tree	min_samples_leaf=1 min_samples_split=2
Random Forest	min_samples_leaf=1, min_samples_leaf = 2, n_estimators=10
SVM	c=0.1
Multilayer Perceptron	activation='tanh', alpha=0.0002

TABLE VI. METHOD-LEVEL FAULT- PRONENESS PREDICTION RESULTS

Table VI presents the overall performance for the techniques of predicting fault applied to method-level datasets.

Metrics / Algorithm	F1	AUC	Accuracy
Logistic Regression	0.45	**0.90**	**0.91**
K-nearest Neighbors	0.49	0.71	0.87
Decision Tree	0.42	0.72	0.86
Random Forest	0.47	0.88	0.9
Naïve Bayes	0.24	0.85	0.88
SVM	0.32	0.66	0.9
Multilayer Perceptron	**0.59**	**0.91**	**0.91**

The results show that Logistic Regression and Multilayer Perceptron get the best values. The AUC values for both

Logistic Regression and Multilayer Perceptron are similar (respectively 0.90 and 0.91). Moreover, Logistic Regression and Multilayer Perceptron achieve the highest accuracy (91%). However, the F_1 value of Multilayer Perceptron (0.59) is better than Logistic Regression (0.45). Therefore, Multilayer Perceptron is the best technique in predicting errors for method-level datasets compared to other techniques

Fig. 3 presents the ROC curves of the experiment on method-level datasets. The ROC curve for Multilayer Perception gives the highest AUC value (0.91). Hence, Multilayer Perception is the best technique for predicting software faults.

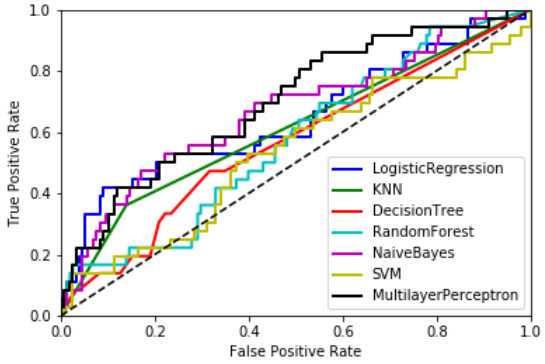

Fig. 3. Comparing the ROC curves between the techniques at method-level

IV. CONCLUSION

Software metrics are often used for evaluating and improving software quality.We studied the using of object-oriented metrics in applying different machine learning techniques for predicting fault. Our experimental study was conducted on seven popular machine learning techniques for predicting fault by using the PROMISE datasets at both method-level and class-level. The obtained results conclude that Support Vector Machine has the highest performance for class-level datasets and Multilayer Perceptron outperforms other techniques for method-level datasets. In the future work, we will study the classification techniques in order to deal with the imbalance issue of datasets for fault prediction.

ACKNOWLEDGEMENT

This research was supported by the Ministry of Education and Training [grant number B2019-DNA-03].

REFERENCES

[1] El-Enam, K.Benlarbri, S., Goel, N., Rai, S. "Avalidation of Object-Oriented Metrics," National Research Council of Canada, NR/ ERB 1063

[2] Elish, K.O.; Elish, M.O, "Predicting Defect-Prone Software Modules Using Support Vector Machines," J.Syst. Softw. 2008, 81, 649–660.

[3] T. J. McCabe and C. W. Butler, "Design complexity measurement andtesting," Communications of the ACM, 32(12), December 1989..

[4] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," Machine Learning, vol. 6, pp. 37-66, January 01 1991.

[5] Cagatay Catal, Banu Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," Information Sciences, Pages 1040-1058, March 2009.

[6] Emam, K. E., Melo, W., & Machado, J. C, "The prediction of faulty classes using object-oriented design metrics," Journal of Systems and Software, 56(1), 63–75, 2001

[7] S. Chidamber and C. Kemerer, "A Metrics Suite For Object Oriented Design," IEEETransactions Software Engineering, vol.20, no.6, pp. 476,493, June 1994,

[8] Aggarwal, K. K., Singh, Y., Kaur, A., and Malhotra, R, "Empirical analysis for investigating the effect of object - oriented metrics on fault proneness: a replicated case study," Software Process: Improvement and Practice, Vol. 14, No. 1, 2009, pp. 39-62.

[9] T. Menzies, J. Greenwald, A. Frank, "Data mining static code attributes to learn defect predictors," IEEE Transactions on Software Engineering 33, 2007, 2-13.

[10] Ahmet Okutan Olcay Taner Yıldız, "Software defect prediction using Bayesian networks", Empirical Software Eng (2014) 19:154–181 © Springer Science+Business Media, LLC, 2012

[11] Catal, C., &Diri, B. cag J, "Expert Systems with Applications," Vol. 36 (4), 2009; 73467354

[12] Harrison, R. S.J. Counsell, and R. V. Nithi, "An Evaluation of the MOOD set of Object-Oriented Software Metrics," IEEE Trans. Software Engineering, vol. SE-24, no. 6, June 1998, pp. 491-496.

[13] Menzies, T.; Krishna, R.; Pryor, D., "The Promise Repository of Empirical Software Engineering Data," 2015.http://openscience.us/repo

[14] Guo, L., Ma, Y., Cukic, B., Singh, H., "Robust prediction of fault proneness by random forests," In: Proceedings of the 17th International Conference Symposium on Software Reliability Engineering (ISSRE'04), 2004, pp. 417-428.

[15] MenziesT, MiltonZ, BurakT, CukicB, JiangY, BenerA, "Defect prediction from static code features: current results, limitations, new approaches," Autom Softw Eng J 17(4):375–407, 2010

[16] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," Machine Learning, vol. 6, pp. 37-66, January 01 1991.

[17] Catal, C., U. Sevim and B. Diri, "Analysis of Naive Bayes' Assumptionson software fault data:An empirical study," Data &Knowledge Engineering, 68(2): 221-290, 2009.

[18] Dumais, S., Platt, J., Heckerman, D., and Sahami, M, "Inductive learning algorithms and representations for text categorization," In Proceedings of the seventh international conference on Information and knowledge management, 1998, pages 148–155. ACM.

[19] Breiman, L., "RandomForests", MachineLearning, vol. 45, no. 1, pp. 5-32, 2001

[20] Lessmann, S, "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings," IEEE transactions on software engineering vol 34, NO 4, 2008.

[21] Gayathri, M. and Sudha, "A. Software Defect Prediction System using Multilayer Perceptron Neural Network with Data Mining," International Journal of Recent Technology and Engineering, 3, 54-59, 2014.

[22] Cagatay Rukshan Catal, Banu Diri, "A systematic review of software fault prediction studies," Expert Systems with Applications, May 2009, Pages7346-735.

[23] Y. Zhou, and H. Leung, H, "Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults," IEEE Transactions on Software Engineering, Vol.32, No.10, 2006, pp.771-789.

[24] Y. Singh, A. Kaur, and R. Malhotra, "Empirical vlidation of object-oriented metrics for predicting fault proneness models," Software Quality Journal, Vol.18,No.1, 2010,pp.3-35

[25] K. El Emam, S. Benlarbi, N. Goel, and S. Rai, "A validation of object-oriented metrics," NRC Technical report ERB-1063,1999

[26] Mark Lorenz and Jeff Kidd, Object Oriented Metrics. Englewood, NJ: Prentice Hall, 1994.

[27] Oliver Sutton, "Introduction to k Nearest Neighbour Classification and Condensed Nearest Neighbour Data Reduction", Feb-2012.

[28] Kotsiantis, S. B., Zaharakis, I., and Pintelas, P, "Supervised machine learning: A review of classification techniques," Emerging artificial intelligence applications in computer engineering, 2007, 160:3–24.

[29] Hosmer, D. W., Jr. and Lemeshow, S, Applied Logistic Regression,Wiley, ISBN: 0-471-35632-8, 2000.

[30] J. B. Bradley, "Neural networks: A comprehensive foundation: S. HAYKIN," New York: Macmillan College (IEEE Press Book) (1994). v + 696 pp. ISBN 0-02-352761-7," Information Processing & Management, vol. 31, p. 786, 1995/09/01/ 1995.