Software Fault-Prediction using Combination of Neural Network and Naive Bayes Algorithm

Bahman Arasteh
Tabriz Branch, Islamic Azad University, Tabriz
Iran
b_arasteh2001@yahoo.com

ABSTRACT: *Nowadays, the role of software has becoming increasingly important in many safety-critical applications and the reliability is a key issue in the software systems. One of the ways for improving software Reliability is predicting its faults before tasting phase. Ability of predicting fault–proneness software modules can reduce software testing cost and consequently overall software project cost. In this paper, a combined method includes Neural Network and Naive Bayes algorithm are used to build a software fault prediction-model. Five traditional fault-datasets are used to construct and evaluate the prediction model using proposed method. The results of experiments indicate that the constructed model by the proposed method have higher prediction accuracy and precision than the other methods.*

Keywords: Software, Fault Prediction, Fault – Proneness Modules, Neural Network, Accuracy

DOI: 10.6025/jnt/2018/9/3/94-101

Received: 29 April 2018, Revised 5 June 2018, Accepted 13 June 2018

1. Introduction

Software systems are becoming more and more complex and many software systems have been delivered to customers with non-negligible number of faults despite applying quality control activities (such as testing) [١, ۶, ٨, 15, 28]. On the other hand, nowadays, the role of software has becoming increasingly important in many safety-critical applications[1] where failures may endanger users or cause the financial losses. Hence, the reliability is a key issue in the software systems. About 80% of the faults come from 20% of the modules and about half the modules are defect free [5]. Predicting software faults before tasting phase helps us to develop more reliable software with a limited project's cost [1]. Faults of software system cannot be directly measured, but they can be estimated based on software metrics [2, 3, 4, 9, 12].

Software fault prediction methods contain one dependent variable and *n* independent variables. The dependent variable illustrates whether the software module is faulty or fault-free; the independent variables are software product or process metrics such as Cyclomatic complexity and lines of code. Some of the software fault-prediction methods classify the software modules as faulty and fault-free using prior fault datasets and data-mining techniques [4, 11, 13, 14,]. Data mining algorithms are used widely for software fault prediction purpose; some powerful machine-learning algorithms that are used for fault prediction are Random Forest, J48, Neural Network and SVM [4, 10, 12]. There are different public and free datasets, such as PROMISE and NASA MDP repositories, that are used by corresponding researchers [4, 12]. In this paper, the proposed method uses a combination of Neural Network and Naive Bayes algorithm to classify the software modules into faulty or faulty-free classes. Using the combination of Neural Network and Naive Bayes algorithm improves the accuracy and precision of the module classification. Five traditional fault-datasets are used to construct the prediction model using proposed method; datasets are accessible in PROMISE [16] data repository. This paper's contributions are as follows:

• Accuracy of the constructed prediction model has been improved using the combination of Neural Network and Naive Bayes algorithm.

• Precision of the constructed prediction model has been improved using the combination of Neural Network and Naive Bayes algorithm.

In section 2, related works are reviewed. In section3, proposed method is described and in section 4, experiments and results of this paper are discussed. Then, in section 5 conclusion and future works are given.

2. Related Works

Metric	Description
WMC	Weighted Methods per Class.
DIT	Depth of Inheritance Tree.
NOC	Number of Children.
CBO	Coupling Between Object Classes.
RFC	Response for a Class.
LCOM	Lack of Cohesion in Methods.
Ca	Afferent Couplings. This is measure of how many other classes are used by the specific class.
Ce	Efferent Couplings. This is measure of how many other classes are used by the specific class.
NPM	Number of Public Methods
LCOM3	Lack of Cohesion in Methods. LCOM3 varies between 0 and 2
LOC	Lines of Code.
DAM	Data Access Metric.
MOA	Measure of Aggregation. This metric measures the extent of the relationship, realized by using attributes.
MFA	Measure of Functional Abstraction. This metric is the ratio of the number of methods inherited by a class to the total number of methods.
CAM	Cohesion among Methods of Class.
IC	Inheritance Coupling.
CBM	Coupling Between Methods.
AMC	Average Method Complexity. This metric measures the average method size for each class.

Table 1. Object-oriented metrics (attributes) of datasets used in the related studies and our study

Software fault-prediction methods have been used since 1990s until now. The usages of public datasets and machine learning methods have been increased since 2005. Software fault-prediction methods use previous software-metrics (attributes) to predict fault-prone modules. Software metrics are categorized into process-level, file-level, class-level, method-level, component-level and quantitative-level categories. CK metrics are the most popular class-level metrics [9] that are shown in Table 1. Some related studies have used Object-Oriented metrics rather than traditional source code-metrics to predict fault prone software [4]; class-level metrics (object-oriented metrics) are only used for object-oriented programs. Object-Oriented metrics are more successful than the other metrics in software fault-prediction. Method-level, class-level, component-level and file-level metrics are the most traditional product metrics used by the previous methods [4, 7]; but the software prediction methods can be enhanced in terms of accuracy and performance by the process-level metrics. Combination of software requirements metrics with code-level metrics has been used in [7] to improve the performance of prediction methods. CPU usage, disk-space usage are the other metrics that used as ‘‘quantitative-level metrics” in software fault-prediction methods [18].

Datasets are the other required resource in software fault-prediction methods. The used datasets in the relevant papers have been divided into four categories: public, private and unknown [13, 14]. NASA MDP (metrics data program) repositories are the main sources of public datasets that used in the previously proposed methods.

Datasets	**Measures**	**Methods**			
		BayesNet	**SMO**	**Multilayer Perceptron**	**AdaboostM1**
kc1	Accuracy	69.89%	84.77%	85.91%	84.96%
	Precision	79.10%	51.65%	77.14%	74.27%
kc2	Accuracy	78.35%	82.75%	84.67%	81.41%
	Precision	82.40%	59.70%	82.80%	78.40%
cm1	Accuracy	64.65%	89.55%	87.55%	90.16%
	Precision	68.93%	49.76%	73.40%	70.00%
pc1	Accuracy	74.39%	92.96%	93.59%	93.05%
	Precision	70.38%	50.19%	72.36%	80.30%

Table 2. Accuracy and precision of the constructed models by different previous methods [4, 13, 14, 17]

Private datasets belong to private companies and they are not free. Some of studies have used datasets that there is no information about them; this type of dataset is called ‘‘unknown”. PROMISE repository is the main sources of public datasets have been used by different researchers since 2005 [18].

The proposed software fault prediction methods in the previously published papers are categorized into following groups: statistical methods, machine learning based methods, statistical methods + expert opinion, statistical methods + machine learning based methods [13, 14]. In [17], a method has been proposed for fault prediction using Gradient Descent Adoptive techniques (GDA); the prediction accuracy of the method is better than the similar methods but this method is time-consuming and complex. The accuracy and precision of different previous methods are compared in Table 2. In this study a combination of Neural Network and Naive Bayes algorithm are used to enhance the accuracy of the software fault prediction.

3. Proposed Method

In this paper, our objective is constructing a learning-based fault predictor for software in such away the proposed prediction

model has higher accuracy and performance than the previously proposed models. To this end the proposed method combines Neural Network and Naive Bayes algorithms by stacking approach. Figure1 shows the proposed method's steps. The proposed method includes the following steps:

• **Preparing the Required Datasets:** The required dataset for constructing fault-prediction model were prepared from PROMISE data-repository [16]; it is the largest public software data-repository which has been used by the researchers to construct the fault-prediction model.

Constructing the Predicting Model using Proposed Method: Combination of Neural Network and Naive Bayes by the stacking approach is used to construct a software fault prediction model with higher accuracy and precision.

• Validating (Testing) the prediction model by the testing data.

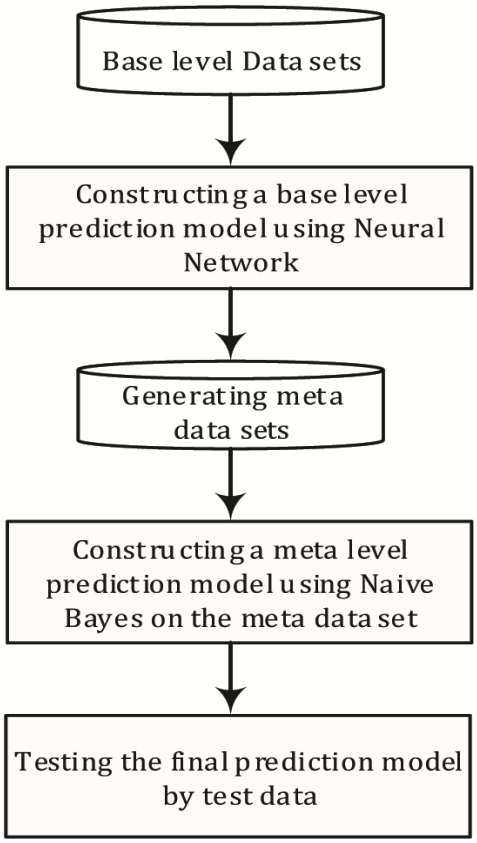

Figure 1. The diagram of proposed methods

Preparing the Required Datasets: PROMISE Software Engineering Repository (as a public software data) is increasingly used in the software engineering studies [16, 18]. PROMISE includes a subset of NASA datasets and categorized as defect prediction, cost estimation, text mining applications.

The NASA dataset were collected from different projects (satellite instrumentation, ground control systems, attitude control system and etc.) in the United States over different years. The fault-prediction datasets contains McCabe, Halstead and line of code metrics; the common attributes explained in Table 1. Table 2 illustrates datasets used in this study. In this study, the objective is to develop a fault prediction model using object oriented metrics at source-code level. The object oriented metrics (shown in Table 1) are considered as independent variables and *fault* (last attribute) is the dependent variable.

Constructing and Evaluating the Predicting Model: The aim of this step is to construct a fault prediction model using the

proposed combined method (Neural Network + Naive Bayes). Stack generalization is a method for combining different classification algorithms to achieve higher prediction accuracy and precision. In this study, the stacking of Neural Network and Naive Bayes is used (as a first time in the literature) to construct a software fault-prediction model; the main motivation of combining these learning algorithms is to reduce the probability of misclassification of software modules. In this method, the final prediction model is constructing in two levels as follows:

• **Base Level:** In this level, the method takes the Neural Network as the base learning algorithms and base dataset D includes *Kc1*, *Kc2*, *pc1*, *cm1* and *jm1* and runs the algorithm on the dataset D. A base prediction model is constructed by running the base algorithm on the base datasets. Then, a new dataset is constructed by replacing the final attributes (faulty/non-faulty) of each instance in the base-level dataset by the predicted value for that instance. is known as Meta dataset.

• **Meta Level:** In this level, the Naive Bayes algorithm, as the Meta learning algorithm, has been conducted on the Meta dataset. Combining two Classifiers, as a multi-level classifier, can improve the accuracy and precision of classification.

The explained datasets in Table 2 were used to construct and evaluate (test) a predictive model using the combined method. In order to construct the model, we used 2/3 of the instances in each available dataset (training data) and to test the model we used the remaining 1/3 instances. One of the important points is that the training and testing data should have same distributions of every attribute. To this end, we develop a program to randomly split the datasets into training and testing data. The developed program repeats until the train and test data include a same percentage of faulty and non-faulty instances.

Dataset	Language	Instances	attributes	Faulty modules (%)
Kc1	C++	2109	21	15.45
Kc2	C++	522	21	20.49
pc1	C	1109	21	6.94
cm1	C	498	21	9.83
Jm1	C	10885	21	19.35

Table 3. Description of datasets

4. Experiments and Results

In order to evaluate the performance of the proposed method (accuracy and precision), the constructed predicting-model was evaluated (tested) using the remaining 1/3 instances of datasets. As mentioned previously the distribution of attributes in train and test data are same. In this study, *Kc1*, *Kc2*, *pc1, cm1* and *jm1* datasets have been used for evaluating the proposed fault-prediction model. These datasets are described in Table 3. The proposed method uses the combination of Neural Network (as a base learning algorithm) and Naive Bayes (as a second level learning algorithm) to construct fault-predicting model. The proposed method has been implemented in WEKA environment which is one of open source and java based tool for data mining applications. The outputs of predicting model in testing step are binary (faulty of fault free) and categorized as follows:

• **True Positive (TP):** The predicting model correctly predicts a positive outcome.

• **True Negative (TN):** The predicting model correctly predicts a negative outcome.

• **False Positive (FP):** The predicting model incorrectly predicts a positive outcome.

• **False Negative (FN):** The predicting model incorrectly predicts a negative outcome.

Equation (2) is used to calculate the accuracy of the predicting model. Error rate is another performance criterion that should be calculated for our predicting model. Equation (3) is used to calculate the precision of the method.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

In order to evaluate the effectiveness of the proposed method, the similar experiments have been performed using the following algorithms on the same datasets:

• Support Vector Machine (SVM)

• Artificial Neural Network (ANN)

• Naive Bayes

The obtained results have analyzed and compared with each other in terms of accuracy and precision. Figure 2 compares the prediction accuracy of the constructed model by different methods. On average the accuracy of the constructed prediction models by Naive Bayes, ANN, SVM and the proposed method are 84.16%, 90.79%, 87.46% and 96.91% respectively. Regarding the results of experiments, the constructed model by the proposed combined method has higher accuracy than the accuracy of the constructed models by the other methods. Precision, as another criterion, is used to evaluate the performance of the proposed method. To this end, the precision of the constructed models by different methods were quantified and compared to each other.

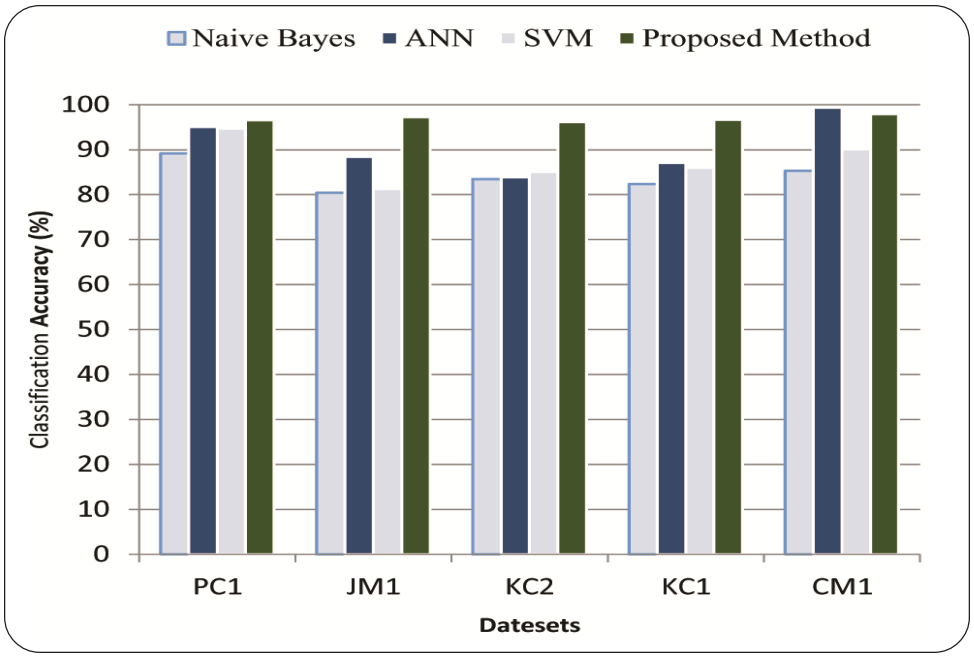

Figure 2. The accuracy of the constructed prediction-models by different methods

Figure 3 shows the precision of different prediction models on different datasets. The average precision of the constructed prediction models by Naive Bayes, ANN, SVM and the proposed method are 0.72, 0.74, 0.50 and 0.99. It is clear that the prediction model by the proposed method is better than the other prediction model in terms of accuracy and precision. The proposed prediction model helps the software testers to focus their effort on the fault prone modules (predicted by the proposed method) instead of all modules. Hence, the software developers can develop more reliable software with a limited project's cost.

The other criterion to evaluate the performance of the proposed method is the required time to construct the prediction-model.

Figure 4 shows the required time to build a prediction model by different methods. On average, the model construction time by Naive Bayes, ANN, SVM and the proposed method are respectively 9.70, 15.42, 10.88 and 10.86 minutes. The model construction time in the proposed method is lower than the ANN and SVM.

5. Conclusion and Future Studies

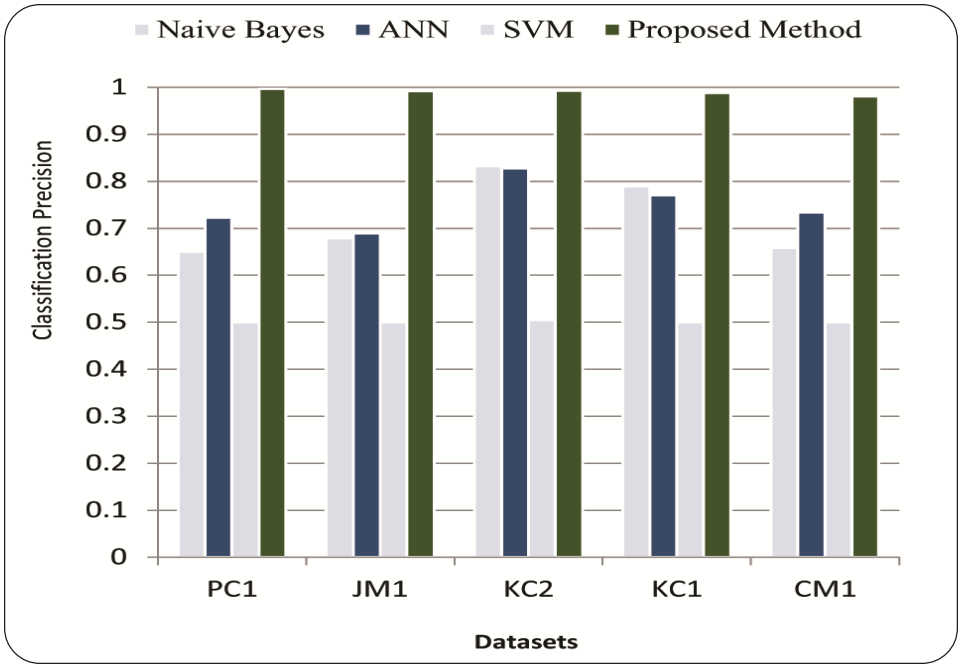

Figure 3. The precision of the constructed prediction-models by different methods

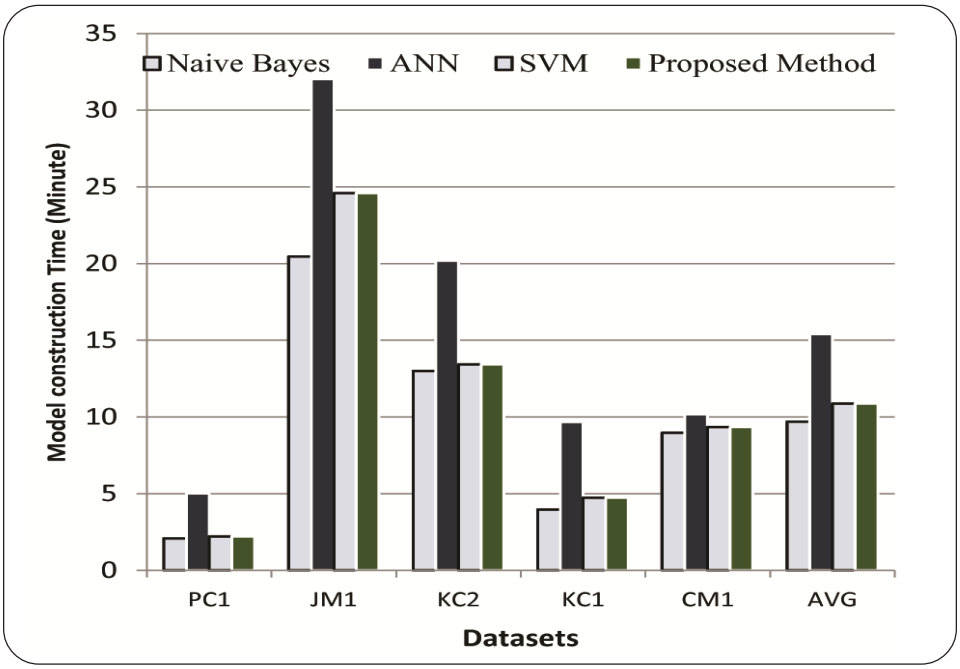

Figure 4. The required time for constructing the prediction-model by different methods

In this study, a combined method has been proposed to predict software fault using combination of Neural Network and Naive Bayes algorithm. The constructed model using the proposed method has higher prediction accuracy and prediction precision than the other methods. Combination of other learning algorithms to build an efficient prediction model is recommended as a future work.

References

[1] Arora, I., Tetarwal, V., Saha, A. (2015). Open Issues in Software Defect Prediction, *Procedia Computer Science*, 46, p. 906–912.

[2] Mahajan, R., Gupta, S. K., Bedi, R. K. (2015). Design of Software Fault Prediction Model using BR Technique, *Procedia Computer Science*, 46, p. 849–858.

[3] Kanmani, S., Uthariaraj, V. R., Sankaranarayanan, V., Thambidurai, P. (2007). Object-oriented Software Fault Prediction using Neural Networks, *Information and Software Technology*, 49 (5) 483–492, (May).

[4] Radjenovi, D., Hericko, M., Torkar, R., •ivkovi, A. (2013). Software Fault Prediction Metrics: A Systematic Literature Review, *Information and Software Technology*, 55 (8) 1397–1418, (August).

[5] Ahmed Yousef, H. (2015). Extracting Software Static Defect Models using Data Mining, *Ain Shams Engineering Journal*, 6 (1) 133-144.

[6] Arasteh, B. (2017). Improving the Resiliency of Software Against Soft-Errors Without External Redundancy and Performance Overhead, *Journal of Circuits, Systems and Computers,* 26 (7) 1750124-28.

[7] Jiang, Y., Cukicc, B., Menzies, T. (2007). Fault Prediction using Early Lifecycle Data, *In*: Proceedings of 17th IEEE international symposium on software reliability, Sweden, p. 237–246.

[8] Aghdam, Karimi., Z., Arasteh, B. (2017). An Efficient Method to Generate Test Data for Software Structural Testing Using Artificial Bee Colony Optimization Algorithm, *International Journal of Software Engineering and Knowledge Engineering*, 27 (6) 967-993.

[9] Catal, C., Diri, B. (2009). Investigating the Effect of Dataset Size, Metrics Sets and Feature Selection Techniques on Software Fault Prediction Problem, *Information Sciences*, 179 (8) 1040–1058, (March).

[10] Jin, C., Jin, S. W. (2015). Prediction Approach of Software Fault-Proneness based on Hybrid Artificial Neural Network and Quantum Particle Swarm Optimization, *Applied Soft Computing*, Vol. 35, p. 717–725 (October).

[11] Okutan, A. (2002). Software Defect Prediction using Bayesian Network and Kernel Methods, Phd, Dissertation, Dept. Computer Engineering, Isik University.

[12] He, P., Li, B., Liu, X., Chen, J., Ma, Y. (2015). An Empirical Study on Software Defect Prediction with a Simplified Metric Set, *Information and Software Technology*, 59, p. 170–190.

[13] Malhotra, R. (2015). A systematic Review of Machine Learning Techniques for Software Fault Prediction, *Applied Soft Computing*, 27, p. 504–518, (February).

[14] Catal, C. (2011). Software Fault Prediction: A Literature Review and Current Trends, *Expert Systems with Applications*, 38 (4) 4626–4636 .

[15] Arasteh, B. (2015). A Bouyer and S Pirahesh, An Efficient Vulnerability-driven Method for Hardening a Program Against Soft-error using Genetic Algorithm, *Computers and Electrical Engineering*, 48, p. 25-43.

[16] *http://openscience.us/repo/*

[17] Erturk, E., Akcapinar Sezer, E. (2015). A Comparison of Some Soft Computing Methods for Software Fault Prediction, *Expert Systems with Applications*, 42 (4) 1872-1879.

[18] Moeyersoms, J., Junqu, E., Dejaeger, K., Baesens, B., Martens, D. (2015). Comprehensible Software Fault and Effort Prediction: A Data Mining Approach, *Journal of Systems and Software ,* 100, p. 80-90 (February).