Cost-Based Evaluation Framework for Software Fault Classification

Dr.Raghav Mehra [1,*], Vinod Parihar[2,*]

[1]Professor, [2]Research Scholar; Computer Science & Engineering Department,
Bhagwant University Ajmer (Rajasthan)

Abstract

Cost-based evaluation framework is necessary to assess the usability of designed fault prediction models. In this chapter, classification of faults using logistic regression and various neural network models as classifiers is discussed in detail. Data classification techniques help in enhancing not only the efficiency of the training process, but also the performance of the predictive model in terms of precision. The proposed approach is applied on a case study discussed in previous chapter viz., Apache Integration Framework version 1.6.

Keywords:
Cost-Based Evaluation Framework, Software Fault Prediction, Cost Analysis, Soft Computing

---

Introduction

Effectiveness of fault-prediction is premeditated by affecting a element of beforehand known data related to faults and expecting its recital beside other part of the fault data. Several researchers have worked on building prediction models for software fault prediction. But, it was noticed that proving the effectiveness of a fault forecast model needs further study. Table 1 lists the proposed criteria considered by various authors in the design of their respective cost evaluation framework.
The table emphasizes on the studies carried out by different authors to compare the techniques, and the evaluation criterion considered in choosing an effective fault classification model.
This paper intends to appraise in ounce of classifier models in predicting errors by using metrics as requisite input to the prediction models, to put the results of a fault-prediction technique in proper perspective. Also an attempt has been made to assess the in ounces of fault removal cost to know whether performing fault prediction analysis is useful or not.

Cost-Based Evaluation Framework:

In literature it is observed that, the work done on classifying the object- oriented classes as a faulty or not-faulty one has been carried out by numerous authors. This can be viewed as a two class" classification problem. The objective of this problem is to identify the *dependent variable* (accuracy) using various classifier models based on several *independent variables*. Independent variables can be considered as some sort of metrics or combination of different metrics.

Table 1: Cost assessment framework for fault classification

Author	Cost evaluation criteria
Ostrand (2005)	Performed prediction using defect densities and concluded that this will be able to nd more defects in a xed percentage of code .
Jiang (2008)	Introduced cost curve based on Receiver Operating Characteristic .
Lessmann (2018)	Identied a common evaluation framework based on ROC Curve (AUC) and used the proposed concept of Demsar to compare the performance of models.
Mende (2009)	Introduced a performance measure (P_{opt}) and compared prediction model with an optimal model. P_{opt} accounted module size to evaluate the performance of a fault-prediction model .
Mende (2012)	Proposed two strategies namely AD (e ort-aware binary prediction) and DD (e ort-aware prediction based on defect density) to include the notion of e ort awareness into fault-prediction model .
Arisholm (2015)	Proposed a cost performance measure - Cost E ectiveness (CE), a variation of lift charts where the x-axis contains the ratio of lines of code instead of modules .

Wagner has designed the cost-based evaluation framework based on certain constraints, as mentioned below:

i. Dierent phases of testing account for varying fault removal cost.
ii. No testing phase can detect 100 % faults.
iii. It is not practically possible to perform unit testing on all modules, so a limited number of important logical paths should be selected, and testing should be exercised to selectively ensure proper working of the software to be delivered [4].

The fault removal costs summarized by Wagner are shown in Table 2.

Table 2: Removal costs of test techniques (in sta hour per defect)

Type	Min	Max	Mean	Median
Unit	1.5	6	3.46	2.5
System	2.82	20	8.37	6.2
Field	3.9	66.6	27.24	27

The fault identification efficiencies for different testing phases are taken from the study of Jones [1. The e cienciesof testing phases are summarized in Table 3. Wilde et al stated that more than fifty percent of modules are usually very small in size, hence performing unit testing on these modules is not fruitful .

Table 3: Fault identification efficiencies of different test phases

Type	Min	Max	Median

Unit	0.1	0.5	0.25
System	0.25	0.5	0.65

The formulation of Ecost, Tcost and the NEcost of the anticipated cost based valuation framework is described in the following subsections:

Anticipated fault removal cost (E_{cost})

The fault removal cost in system testing is computed using Equation 1.1.

$$Cost_{system} = \delta_s * C_s * (FN + (1 - \delta_u) * TP) \quad (1.1)$$

where, δ_u and δ_s represent fault identi cation e ciency of unit testing and fault identification effciency of system testing respectively.

Remaining faulty classes which were not identied in system testing will be further identi ed in eld testing. The fault removal cost in case of eld testing (C_f) is computed using Equation 1.2.

$$Cost_{field} = (1 - \delta_s) * C_f * (FN + (1 - \delta_u) * TP) \quad (1.2)$$

The estimated overall fault removal cost can be determined by using Equation 1.3.ie.,

Ecost = $Cost_{unit}$ + $Cost_{system}$ + $Cost_{field}$

$$Ecost = C_i + C_u * (FP + TP) + \delta_s * C_s * (FN + (1 - \delta_u) * TP) + (1 - \delta_s) * C_f * (FN + (1 - \delta_u) * TP) \quad (1.3)$$

The other notations in this cost evaluation framework are as follows:

i. M_p: percentage of classes unit tested.

ii. FP : Number of false positive, FN : Number of false negative, TP : Num- ber of true positive, TN : Number of true negative, TC: Total number of classes, FC: Total number of faulty classes.

iii. δ_u: Fault identi cation e ciency of unit testing, δ_s: Fault identi cation e ciency of system testing.

In this experiment, the values tabulated in Table 4.2 are used in designing cost evaluation framework. δ_u and δ_s show the fault identification efficiency of entity testing and system testing respectively. The values of δ_u, and δ_s are collected from the survey report Software Quality in 2010" M_p shows the fraction of modules unit tested, obtained from the paper of Wilde [12]. Median values have been chosen in this cost-based analysis.

The objective is to provide the benchmarks to approximate the overall fault removal cost. Figure 1 shows the owchart optedfor the proposed cost-based evaluation framework for software fault classification.

The proposed framework clearly states that if a technique accounts for having high false negative and/or high false positive rates, then it results in higher fault removal cost. When this approximated cost surpass the unit testing cost (T cost)

It is enhanced to analysis all the modules at unit level instead of using fault prediction technique.

Results and Analysis:

In this segment, the association between value of metrics and the fault found in a class is determined. In this approach, the proportional learning involves via six CK metrics as input joins and the output is the achieved fault classifcation rate for AIF version 1.6.

This section highlights on the design and use of neural network as a classifer and also presents the obtained cost-based analysis results for classifcation of faults obtained by applying Logistic regression, ANN, RBFN, FLANN and PNN techniques

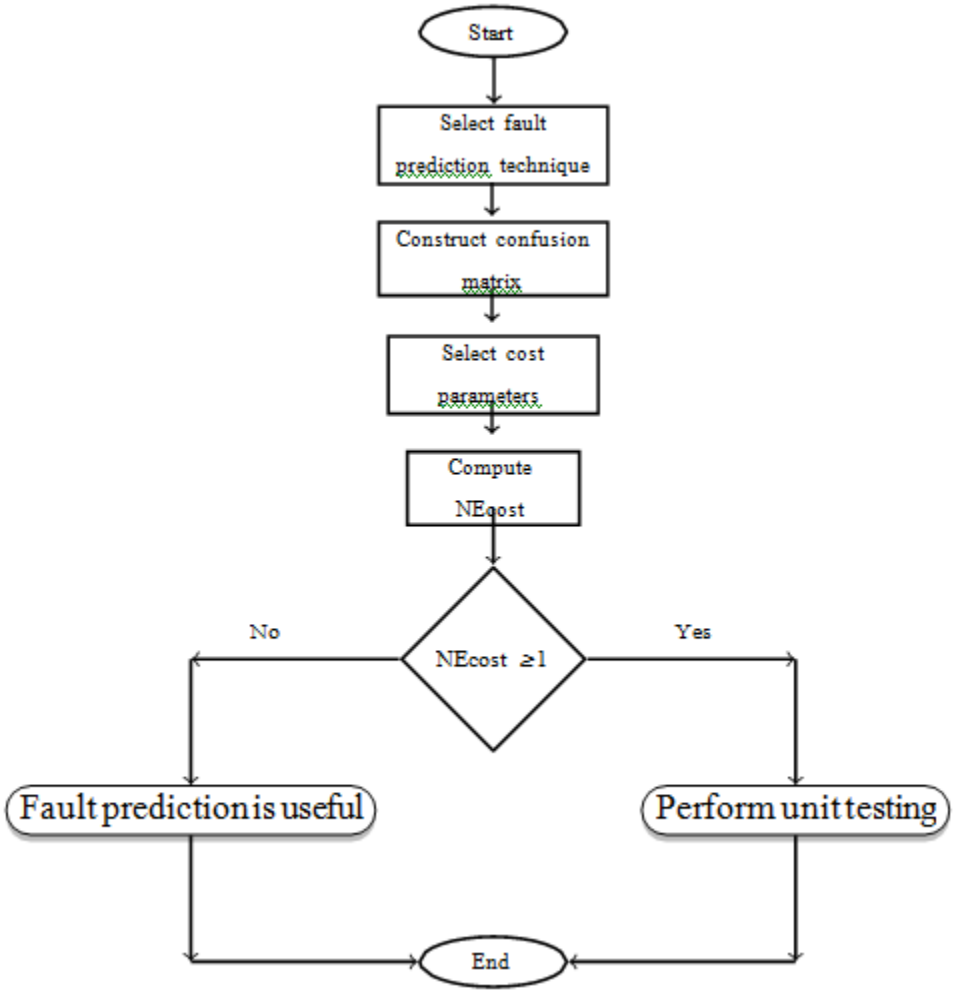

Figure 1: Cost-based evaluation framework for software fault classification.

Neural network as a classifier:

In the design of neural network as a classifier, the target output y' determines the type of classification result of a class as faulty or not faulty. The following conditions are taken into account to predict the accuracy of classification which are mentioned below for several neural network approaches as classifier:

$$Classifier = y' \Longrightarrow \begin{cases} > 0, & Class\ is\ Faulty \\ < 0, & Not\ Faulty \end{cases} \quad (4.16)$$

This table contains a total number of 865 classes, among which 779 classes have zero bugs and the remaining 188 classes have at least one bug.

Table 4.4: Confusion matrix

	Not-Faulty	Faulty
Not-Faulty	777	0
Faulty	188	0

ANN as a classifier:

Table 4.5 and Table 4.6 illustrate the classification matrix for Gradient Decent and Leven berg arquardt learning techniques of ANN.

Table 4.5: Confusion matrix for Gradient Descent

	Not-Faulty	Faulty
Not-Faulty	761	16
Faulty	157	31

Gradient Descent ANN (Table 4.6) is able to classify a total of 792 (761+31) classes as not-faulty with a accuracy of 82.07%.

Table 4.6: Perplexity matrix for Lederberg Marquardt

	Not-Faulty	Faulty
Not-Faulty	756	21
Faulty	171	17

In assessment with Table 4.4, LM model (Table 4.7) is competent to classify a total of 773 (756+17) classes as not-faulty with an accuracy of 80.10%.

RBFN as a classifier:

In Table 4.8 shows the classification matrix when Basic RBFN is used as a classifier. After applying Basic RBFN classifier (Table 4.8), a total of 607 (517 + 90) classes are correctly classified as not-faulty with an accuracy of 62.9%.

Table 4.8: Confusion matrix for Basic RBFN

	Not-Faulty	Faulty
Not-Faulty	517	260
Faulty	98	90

FLANN as a classifier:

Table 4.11 shows the obtained classification matrix when FLANN technique is used as a classifier.

Table 4.11: Confusion matrix for FLANN

	Not-Faulty	Faulty
Not-Faulty	742	35
Faulty	160	28

After applying FLANN classifier (Table 4.11), a total of 770 (742+28) classes are correctly classified as not-faulty with an accuracy of 79.79%.

PNN as a classifier:

Table 4.12 shows the classification matrix obtained by applying PNN tech- nique as a classi er.

Table 4.12: Confusion matrix for PNN

	Not-Faulty	Faulty
Not-Faulty	775	2
Faulty	181	7

In assessment with Table 4.4, it is observed that, after applying PNN as a classifier (Table 4.12), a total of 782 (775 + 7) classes are correctly classified as not-faulty with an accuracy of 81.03%.

The fault removal cost for AIF version 1.6 obtained by applying Logistic regression, ANN, RBFN, FLANN and PNN techniques are tabulated in Table 4.13..

Table 4.13: Fault removal cost for AIF 1.6 using various classifier models

Classification model	Precision	TP Rate	FP Rate	TN Rate	FN Rate	Accuracy	Ecost	NEcost
Logistic regression	61.54	08.51	01.29	98.71	91.49	81.13	3119.4	0.8795
Gradient Descent	65.96	16.49	02.06	97.94	83.51	82.07	3109.7	0.8762
Levenberg Marquardt	44.74	09.04	02.70	97.30	90.96	80.10	3145.3	0.8868
RBFN Basic	25.71	47.87	33.46	66.54	52.13	62.90	3622.3	1.0213
RBFN Gradient	99.19	64.89	00.13	99.87	35.11	93.50	2922.0	0.8238
RBFN Hybrid	40.00	10.64	03.86	96.14	89.36	79.40	3162.8	0.8917
FLANN	44.44	14.89	04.50	95.50	85.11	79.79	3162.1	0.8915
PNN	77.78	03.72	00.26	99.74	96.28	81.03	3114.3	0.8780

Comparison of cost analysis:

Data set of AIF version 1.6 from PROMISE repository is chosen to estimate the impact of fault prediction technique. The fault removal cost (NEcost) computed using the proposed framework is used to evaluate the models.

To illustrate efficiency of the anticipated cost-based evaluation framework, classifier models such as Logistic regression, ANN, RBFN, FLANN and PNN are used for computing misclassification cost. The goal is to demonstrate the cost evaluation framework and suggest whether performing fault prediction using particular prediction model is useful or not rather than identifying the

best" fault-prediction model.

Table 4.13 shows the various parameters related to cost evaluation frame- work along with NEcost. NEcost is the criterion used in evaluating a classification model to show the usefulness of fault prediction. From Table 4.13, it can be observed that:

i. The Gradient Descent approach of RBFN classifier obtained the best classification rate of 93.50% when compared with other four models, and

ii. Gradient Descent RBFN incurs less cost involved in testing (with a NEcost ratio of 0.8238).

This indicates that performing fault prediction on the basis of classification cost involved using Hyrbid RBFN method is very much effective in assessment with LR, GD, LM, FLANN and PNN models.

It is observed that, Normalized fault removal cost (NEcost), which is the ratio of Ecost and Tcost (Equation 4.9) determines the fault prediction model's e ectiveness in a succinct manner.

The proposed cost-based evaluation framework provides:

1. A binary yes or no scale whether to perform fault prediction analysis or not.
2. A criterion to choose a better fault prediction model based not only on the obtained accuracy rate but also taking NEcost into consideration.

Conclusion :

This paper comprises detailed survey on software fault prediction for classifying software modules which is faulty or non-fault. In this paper, an attempt has been made to design a cost based evaluation framework for finding the efficiency of the developed fault prediction model using different neural network models as classifiers. Models such as LR, ANN, RBFN FLANN and PNN were used as classifiers..All described methods are collectively called as computational intelligence techniques. The precision rate of fault prediction using different techniques show that fuzzy has more useful than other methods due to save in training time. The FIS is technically better, it has power of rule for which handle the different size of the software. It has also independent of the size of the software.

References:

[1] Ian Somerville, "Software Engineering Eight edition", Pearson Edition India, 2012

[2] Aditya P.Mathur, Foundation of software Testing,(Pearson Education, India,2007)

[3] Sushant Kumar , Prabhat Ranjan and R.Rajesh "A Concept for Test Case Prioritization Based Upon the Priority Information of Early Phase", Lecture Notes in Electrical Engineering, (2016)

[4] Yogesh Singh, Ruchika Malhotra, "Object-Oriented Software Engineering", PHI India (2012)

[5] Luciano S.de Souza, Ricardo B.C.Prudencio,Flavia de A. Barros, "Search Based constrained test case selection using execution effort",Expert Systems with Applications40(2018)

[6] Manoj kumar, Arun Sharma,Rajeshkumar, "Towards Multi-Faceted Test Cases Optimization", Journal of Software Engineering and Applications,4, 550 (2019)

[7] J. Brownlee, Clever Algorithms: Nature-Inspired Programming Recipes. Lulu Enterprises Inc., 2020.

[8] F. Wu, Empirical validatin of object-oriented metrics on NASA for fault pre- diction, in Proceedings of International Conference on Advances in Information Technology and Education. Springer, 2018.

[9] Y. Singh, A. Kaur, and R. Malhotra, Empirical validation of object-oriented metrics for predicting fault proneness models, Software Quality Journal, vol. 18, no. 1, pp. 3 35, March 2010.

[10] J. Brownlee, Clever Algorithms: Nature-Inspired Programming Recipes. Lulu Enterprises Inc., 2011.

[11] L. N. De Castro and F. J. Von Zuben, Learning and optimization using the clonal selection principle, IEEE Transactions on Evolutionary Computation, vol. 6, no. 3, pp. 239 251, June 2002.

[12] K. K. Aggarwal, S. Yogesh, K. Arvinder, and R. Malhotra, Empirical analysis for investigating the e ect of object-oriented metrics on fault proneness: a replicated case study, Software Process: Improvement and Practice, vol. 14, no. 1, pp. 39 62, August 2009.

[13] H. M. Olague, L. H. Etzkorn, S. Gholston, and S. Quattlebaum, 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 Transactions on Software Engineeering, vol. 33, June 2007.

[14]]M. Cartwright and M. Sheppered, _An empirical investigation of an object ori-

[15] ented software system,_ IEEE Transactions on Software Engineeering, vol. 26,no. 8, pp. 786_796, August 2000.

[16] F. Wu, Empirical validatin of object-oriented metrics on NASA for fault pre- diction, in Proceedings of International Conference on Advances in Information Technology and Education. Springer, 2021.

[17] G. Pai and J. Dugan, Empirical analysis of software fault content and fault proneness using bayesian methods, IEEE Transactions on Software Engineeer- ing, , October 2007.

[18] Y. Singh, A. Kaur, and R. Malhotra, Empirical validation of object-oriented metrics for predicting fault proneness models, Software Quality Journal, vol. 18, no. 1, pp. 3 35, March 2010.

[19] Sushant Kumar ,Prabhat Ranjan and R.Rajesh, "An Overview of Test Case Optimization using Meta-Heuristic Approch", Recent Advances in mathematics, Statics and Computer Science, (2015)