Using different ML methods for assessment of software defect prediction model

Meer Tauseef Ali[1], Dr .Syed Abdul Sattar[2]
1. Research Scholar,(PP.COMP.SCI.O398) Department of Computer Science, Rayalaseema University, Kurnool, A.P.
2. Professor, Principal, Nawab Shah Alam Khan College of Engg. & Tech., Hyderabad.

Abstract

In order to estimate the sensitivity of application modules to errors in software engineering, many computational approaches have been proposed. Classifying a program module as vulnerable to error means the execution of multiple verification operations. The wrong designation of a module as free of faults includes the possibility of device failure and has repercussions for costs as well. The selection of the "best" candidate from the several available models involves performance assessment and thorough comparisons, but due to the applicability of various performance indicators, these comparisons are not simple. This research identifies the benefits and disadvantages of performance assessment approaches and suggests that without taking into consideration the expense characteristics of the project unique to each implementation environment, the option of a "optimal" model will not be made. At the root of scientific software engineering research should be accurate methods for analyzing fault prediction models, but they have gained only scattered attention so far. In comparison to the numerous approaches currently used in software engineering research, this paper offers a description of model estimation methods by introducing and explaining the advantages of cost factors.

1 Introduction

Shantanu et al. [1] tends to illustrate the schema of how to make accurate prediction of the software. To forecast the consistency of the applications, several modeling approaches have been suggested and applied. Santosh S et al. [2] provides an overview of how to predict faults in the software system. Any of these techniques, such as logistic regression, attempt to create a relationship between inputs (software metrics) and outputs (software error rates) by using domain-specific knowledge [3]. In order to classify models and generalizations, Some use a single model to assess susceptibility to error Some predict the 'best' while others focus on collective learning to generate a selection of models from a single set of training data. [4][5]

The various aspects of the software fault prediction are investigated and discussed for all data sets, or at least most data sets, a modeling methodology could perform well. For various datasets, different classification algorithms have been implemented. Multiple experimental configurations restrict our ability to consider the algorithms' strengths and disadvantages. The collections of data can also be used to validate and compare the prediction models proposed. Challagulagla et al. suggested testing several sets of forecast models to evaluate accurate classification algorithms [6].Effective and reliable performance metrics should be employed to enable comparisons between models. The efficiency of the model can, however, be evaluated and evaluated to pick the "best" model from the several possible models. Inappropriate evaluation metrics have been used by several researchers, i.e. measures that do not provide enough information for possible comparisons. One of the purposes of this paper is, for these reasons, to

1) A description of the success metrics and their advantages for widely used versions.

2) Comparison of methods of model assessment and criteria for choosing them.

We conclude that this study's conclusions and guidelines have the ability to strengthen the predictive viability of subsequent studies and eventually to be at the forefront of the prediction of model failure.

2 The structure of the hypotheses

It's necessary to explain the features of the data set when reporting experiments to model error predictions. We make use of six well-known classification algorithms in our illustrative case. To remind readers: our aim is to illustrate the strengths and disadvantages of a specific measure of success, not to determine the "best" algorithm. The preference of classifiers is, thus, orthogonal to the planned contribution. Random Tree, Naïve Bayes, Bagging, J48, Logistic Regression and IBk are the six chosen classifiers. A broad approach to machine learning is defined by these algorithms. A recent research has shown that in detecting program failures, these classifiers do better than average [7].

Table 1: Characteristics of the AGI project used in the experiment

Project	Number of modules	Faulty (%)	Description
AGI	1,908	14	Storage management, receiving/processing
AT	494	19	Science data processing
PIMCO	98	51	Storage management , receiving/processing
ALH	1,096	7	Storage management, receiving/processing
AGA	16,894	3	Science data processing upgrade system

3 Analytical numerical implications of the predictive model

In the information engineering literature, numerical performance metrics are most frequently used. An summary of them is given in this section and their particular strengths and shortcomings are discussed.

3.1 Overall accuracy and error rate

Defective types are more likely to be a minority in the data collection, which can be a deceptive criterion (see Table 1). The total precision tests the probability that the error rate of individual modules will be accurately estimated. It lacks the delivery of data and information on prices. The drawbacks of overall precision as a success appraisal benchmark are evident. The average precision is the highest for any of the projects in Table 2 and the detection rate of failure groups is the lowest. The overall precision and detection rates of the fault modules (called PDs, as described in the next section) for four projects are shown in Table 2: AG, AGA, ALH and PIMCO.

3.2 Sensitivity, specificity and precision

Where an error-containing software module is called a positive case, sensitivity is characterized as the possibility that the error-containing module is correctly classified. Sensitivity is also known as probability of detection (PD) [8] in the field of software quality prediction. As already mentioned, for program failure prediction, modeling techniques that yield very low PDs are not suitable. Specificity is defined as the percentage of error-free modules which are correctly labeled. The percentage of incorrectly labeled and error-free modules is defined as the likelihood of false alarm (PF). The accuracy of PF = 1. That's what's obvious. For strict software testing and validation, often fault-free modules are "marked" where the precision is minimal, unnecessarily increasing the cost of the project and the time required for completion. Of necessity, a trade-off between sensitivity and precision is still available.

Table 2: Performance results for the three projects

	ALH		AGA		AGI	
Method	**Acc (%)**	**PD (%)**	**Acc (%)**	**PD (%)**	**Acc (%)**	**PD (%)**
Naïve Bayes	88.9	29.9	82.1	37.9	83.8	39.6
Logistic	92.2	6.6	86.1	21.1	81.9	39.4
IB1	90.8	43.9	84.2	41.2	79.2	51.2
J48	92.8	22.9	85.2	32.8	81.8	55.2
Bagging	94.1	17.2	84.8	25.3	83.3	46.9

3.3 Graphical evaluation methods

A innovation in software engineering literature is the consideration of cost curves, as far as we know. This chapter addresses the receiver operational characteristic curves (ROC), accuracy and recall curves (PR) and expense curves [9]. These diagrams are taken from and are thus closely related to the Uncertainty Matrix. Given the similarities, a particular feature of classification success is shown by each curve and should be considered in software engineering projects.

3.3.1 ROC curve

It is common to use AUC to compare the results of different grading approaches on the same outcomes; the region under the ROC curve (called AUC) is a qualitative performance metric that is explicitly related to the ROC curve. However, the whole field below the curve may not be of interest to software engineers. Comparisons in information engineering research [10] can be rendered using another basic non-parametric approach used by AUC. The main idea is to compare ROC curves using a sequence of sampling lines from a reference point; the ROC space is sampled by these lines. The intersection of these lines with the ROC curves can be defined and the Euclidean distance from the reference point can be computed.

3.3.2 Precision- Recall curve

The x-axis represents recall in the PR curve and the y-axis represents accuracy. Another name for PD is Recall. For visually comparing classifiers, the precision-recall (PR) curve provides an alternative approach [11]; the PR curve can display variations between algorithms that are not evident from the ROC curve. We suggest that only where the ROC curve does not indicate a discrepancy in the performance of the various grading algorithms can a PR curve have sufficient distinction. Although the purpose of the ROC curve and PR curve analysis is to maximize the area under the curve, the goal of the cost curve is to minimize the cost of misclassification, i.e. to minimize the area under the envelope.

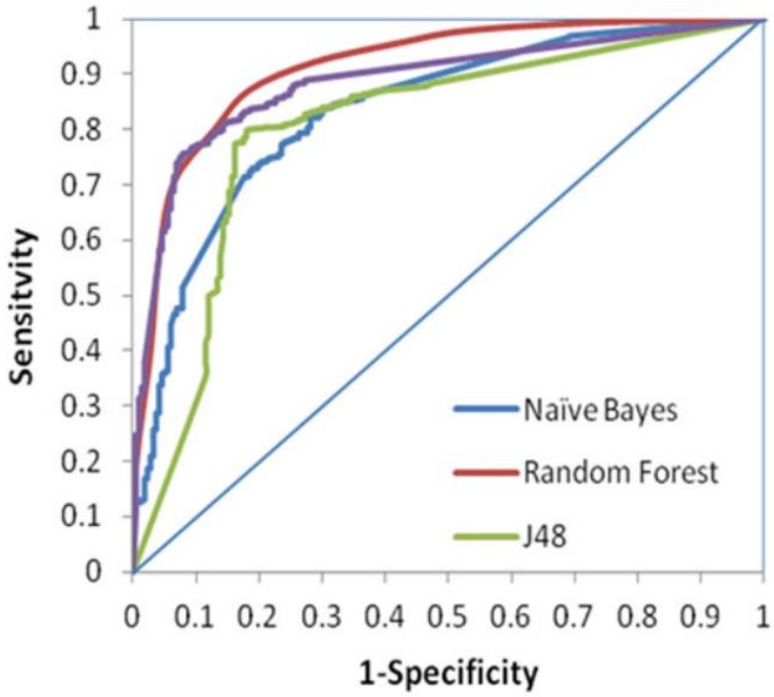

Figure 1: ROC curves for two the Naïve Bayes (nb) ,Random forest and j48 classifiers in the AGI project.

4 Statistical comparisons of classification models

Comparing the efficiency of various classification algorithms and making valid decisions is not straightforward. Statistical conclusions are also needed.[12][13] The goal of benchmarking is to choose among many candidates the best model. Suppose k-models have to be compared.

The statistical hypothesis is

Ho: There is no difference in performance between classifiers.vs.

H1: The performances of at least two classifiers are significantly different.

The Nemeni test is calculated as follows

If qa is the critical value of the Nemenyi measure, the difference in output between the two classifiers is important if the average rank difference between the two classifiers is greater than the CD value. We have N=8 records and k=6 classifiers in this situation. The average ranks of the IB1, J48, NB, Logistics, Bagging and Random Forest classifiers are respectively 5.50, 5.25, 3.625, 3.125, 2.5 and 1.

The Friedman test verifies the null hypothesis.

For k -1=5 and (k-1)(N-1)=5×7=35 degrees of freedom, the critical value of the F distribution is 2.48. From 33.41>2.48, the null hypothesis that there is no difference in the performance of these six models in the eight data sets is rejected. We then compare the performance of the classifiers through Nemenyi's post hoc test: for the six classifiers the critical value qa is 2.85, so the critical difference is CD =2:85 => 2.85((6(6-1))/(6*8))=2.85(1.067)=2.67.

Macskassy et al. analyze how confidence intervals are built as point-by-point or global confidence intervals along the ROC curve.[14][15] The consumer measures the difference between two PD values for a given two-value PF in a point-by-point approach. As an example of the global confidence band process, the AUC variations are statistically evaluated. The same form of statistical test can be performed on ROC curves, PR

curves and lift tables. The remainder of this section focuses on establishing confidence intervals for cost curves that have not been addressed in the software engineering literature previously.

To demonstrate the use of trust limits on cost curves, the approach illustrated by Drummond and Holte [16] is used. The technique is based on using the bootstrap approach to re-sample a standard 500 uncertainty matrix. The classifier's output is extracted from the uncertainty matrix. In the cost curve, confidence intervals are created by a series of uncertainty matrices, as in other power analysis graphs. For e.g., by extracting 2.5 percent of the highest and lowest predicted cost values, the 95 percent confidence interval for a given PC(+) value is gained.

The performance of the two classifiers which vary greatly in certain curve regions, but examples of the discrepancy in performance between the two IB1 and j48 classifiers in ALH recordings can be seen in some regions. The shaded area reflects the 95 percent confidence interval in the figure. Inside the shaded field, there are three lines. The discrepancy between the mean values of IB1 and j48 is shown by the middle line. The other two lines along the shaded region's edge represent the 95 percent confidence interval between the means of separation of the two classifiers. There is no significant distinction between the two classifiers when there is a zero (horizontal) line in the confidence interval; otherwise, there is one.

The larger distance from the zero line of the confidence interval implies that the gap between the two classifiers is more significant; a confidence interval above zero in comparison to PC(+) in the range of (0.0, 0.1) means that J48 performs better than IB1. In the (0.27, 0.59) interval, the confidence interval is below zero, indicating that IB1 performs better than J48. The confidence interval includes the zero line at the interval between (0.1, 0.27) and (0.59, 0.64), implying that there is no substantial gap between the two classifiers; the output of the two classifiers is the same for PC(+) >0.64; since the failure rate of modules in the ALH design is 7%, PC If (+) = 0.07, then the misclassification cost is the same for both defect-prone and defect-free classes; for PC(+) < 0.07, the misclassification cost of defect-free modules exceeds the misclassification cost of defect-prone modules.

(1)If PC (+) <0.07, then j48 is a bad choice (worse than a trivial classifier).

(2) j48 outperforms IB1 in this region (0,27,0,59).

IB1 and j48 have similar behavior.

5 Analysis of the experiment

Here, when designing and choosing a fault prediction model, we replicate the research that tech practitioners do. Second, six templates are generated using a classifier. In this analysis, the models built with the random forest algorithm have been omitted. Random forests outperform other classification methods for most of the dimensions of interest, rendering the topic of model selection very irrelevant. The removal of one of the classifiers does not limit the generality of our suggestions, since the object of this paper is to clarify the comparison and selection process of the model.

5.1 ALH data recording

From the MDP repository, we want to pick the best classifier for the ALH project. The most suitable classifier based on individual numerical metrics is illustrated in Table 7.4 for ease of comparison; the supremacy of the IB1 classification algorithm is apparent by using the basic parameters for each classifier, as provided in the Weka toolkit.

Table 3: Numerical performance indicators for ALH projects

Indices	Naïve Bayes	Logistic	IB1	J48	Bagging
PD	0.296	0.068	**0.439**	0.229	0.171
1−PF	0.941	0.991	0.949	0.991	**0.995**
Precision	0.261	0.291	0.420	0.538	**0.728**
Overall accuracy	0.888	0.918	0.922	0.929	**0.934**
G-mean1	0.280	0.140	**0.432**	0.360	0.349
G-mean2	0.532	0.248	**0.651**	0.478	0.409
F-measure (β=1)	0.282	0.111	**0.432**	0.331	0.269
F-measure (β=2)	0.288	0.076	**0.442**	0.258	0.198
ED (θ=0.5)	0.488	0.658	**0.402**	0.538	0.592
J-coeff	0.242	0.049	**0.402**	0.224	0.158

However, we would use the model produced by the bag classifier if we were to use overall precision, accuracy and specificity. The likelihood of identification (PD) during bagging, however, is limited (~17%) and its applicability to software engineering ventures is uncertain. Only IB1 satisfies this criteria if we verify the likelihood of identification and set the approval threshold to 0.40; G-mean, F-mean, J-coefficient and distance to complete classification all display IB1 as a candidate model. Therefore, with the visual model performance appraisal process, we must continue the research.

Table 4: AUC and AUCa for ALH fault prediction models

Dataset	Learner	AUC	AUCa
ALH	IB1	0.689	0.139
	J48	0.671	0.097
	Bagging	0.838	0.495
	Logistic	0.809	0.432
	NaiveBayes	0.672	0.109

In the AUC analysis, the models are classified as follows. Packaging>Logistics>IB1>Naive Bayes>J48. The actual AUCs are shown in Table 3. The average AUC values correspond to the order observed by the ROC curves From this we can conclude that.

(1) The highest average rank is held by J48 and Naïve Bayes, with the worst results.

(2) Bagging has an average rating of 1, which is higher than the five models but is indistinguishable from models built at 95 percent confidence intervals using logistic classifiers.

(3) The performance gap between J48, Naïve Bayes and IB1 is not statistically important and there is also no significant performance difference between IB1 and logistics.

In brief, the findings are much nuanced given the five versions of ALH projects chosen for comparison. The numerous models fulfill the various needs of software projects. If the software quality engineer is persuaded that the G or F test is correct, the IB1 classifier should be selected for the project. However, all the arguments in favour of selecting the model created by the bagging algorithm are precision, specificity, ROC index and cost curve. The model created by J48 provides the best output considering the lift curve, whereas the bagging model performs almost as well. The argument we are making is that when only a few pairs are registered, it is not possible to draw general conclusions about the error prediction model (TE, PF). The overall appraisal posed

here suggests that it is reasonable to challenge the relevance of those findings. One of the principal findings of this analysis is this statement.

5.2 AGI datasets

The AGI specification numerical efficiency metrics are seen in Table 6, with the minimum PD tolerance set at 40%, naive Bayes and logistic classifiers will not be further considered, whereas the former would provide the highest overall specificity, precision and accuracy. While the overall precision of the bagging model is the best, it is not as good as that of J48 for the other metrics, G-mean, F-mean, J-coff and distance to complete classification. The AUC values in Table 5 show a similar trend.

Table 5: Performance results for project AGI

Indices	Naïve Bayes	Logistic	IB1	J48	Bagging
PD	0.401	0.392	0.511	**0.552**	0.469
1−PF	**0.948**	0.928	0.862	0.902	0.928
Precision	**0.668**	0.602	0.479	0.581	0.642
Overall Accuracy	**0.842**	0.818	0.792	0.818	**0.842**
G-mean1	0.522	0.479	0.502	**0.558**	0.552
G-mean2	0.620	0.598	0.658	**0.711**	0.659
F-measure (β=1)	0.498	0.469	0.502	**0.559**	0.539
F-measure (β=2)	0.428	0.420	0.498	**0.549**	0.502
ED (θ=0.5)	0.431	0.428	0.358	**0.329**	0.381
ED (θ=0.67)	0.488	0.497	0.411	**0.382**	0.428
J-coeff	0.352	0.318	0.370	**0.439**	0.398

We discovered that

- Bayesian, logistic and naive bagging models are statistically indistinguishable in output.
- I can't say that the J48 and the luggage, the IB1 and the J48 work differently.
- Such comparisons indicate substantial variations statistically.

The mean values of the AUC help the Naïve Bayes-generated model. However, maximizing the area under the curve (AUCa) in the upper left corner of the ROC favors logistics (see Table 5). The statistical test shows us that the difference is not important between the naive Bayesian model, the logistic model, and the bagging model. Since the test analyses AUC values (not AUCa), it is reasonable to use the model built with the logistic classifier. For the AGI error estimation models, These results are consistent with those derived by evaluating the ROC curves and predictive tests; the PC(+)=0.21 vertical line shows that error performance occurs while error-prone and error-free groups are equal to the misclassification costs (21 percent of modules in AGI are error-prone).

The five AGI models are seen here; 523 modules are included in the AGI project (see Table 1). You can prefer the bagged model if you have to choose up to 10 percent of these modules. If open validation tools are sparse, this could be the most appropriate subset to use. Logistics is the model to be used because it is possible to review 10-25 percent of modules by capital (between 52-130); quite surprisingly, if AGI is a security and mission-conscious initiative, time and budget are not too tight, and it is possible to test more than 25 percent of modules.

Table 6: Average AUC values for AGI models

Dataset	Learner	AUC	AUCa
AGI	IB1	0.691	0.170
	J48	0.732	0.311
	Bagging	0.818	0.469
	Logistic	0.818	**0.478**
	NaiveBayes	**0.828**	0.458

A quick overview of the assessment of the AGI model reveals that the choice of the naïve Bayes model contributes to total precision, accuracy and specificity; the basic validation scenarios of G-means and F-means plots and lifts show that J48 is supported; the ROC and AUC review indicates that similarly good options are Naïve Bayes, Logistics and Bagging.

6 Conclusions.

Quality prediction of software tends to gain interest because it promises to increase performance and offers guidelines for verification and validation activities of software. Several datasets detailing module measurements and their error content have been made public over the past five years. As a result, in the quest for the 'right' modeling solution, numerous methods of software quality estimation have been suggested. It has become popular to suggest playing with various modeling techniques and selecting the most fitting one We assume that the methodology and appraisal measures presented in this paper are capable of delivering a broad understanding, enhancing significance, and improving the methodological viability of prospective research and experiments. Reasonable comparison of models and their assessment is, however, a requirement for model selection. The goal of this paper has been to describe strategies that are applicable to a software engineering project's particular application needs.

The clear implication is that a multidimensional question is the contrast of fault prediction models. Through analyzing different performance indicators, an in-depth debate on the effective and reliable measurement of fault prediction models in software engineering was carried out. It is not feasible to show that one model or modeling methodology in software quality evaluation is ideal for any possible use. The truth of software engineering program, though, tells us that the average success of the classification of models is not the real target. The capacity to characterize the variety of expense ratios for misclassification is critical because it is never a straightforward job to correctly calculate. In reality, we therefore firmly encourage the use and expect that they can become a common instrument for measuring the performance of software quality models.

REFERENCES

[1] Sutar, S., Kumar, R., Pai, S., Shwetha, B. R. (2019, February). “Defect Prediction based on Machine Learning using System Test Parameters”. In 2019 Amity International Conference on Artificial Intelligence (AICAI) (pp. 134-139). IEEE.

[2] Rathore, Santosh S., & Kumar, S. (2019). “A study on software fault prediction techniques” Artificial Intelligence Review, 51(2), 255-327.

[3] Basili VR, Briand LC, Melo WL (1996) A validation of object-oriented design metrics as quality indicators. IEEE Trans Softw Eng 22(10):751–761. doi:10.1109/32.544352.

[4] Ma Y (2007) An empirical investigation of tree ensembles in biometrics and bioinformatics. West VirginiaUniversity, PhD thesis, January 2007

[5] Challagulla VUB, Bastani FB, Yen I-L, Paul RA (2005) Empirical assessment of machine learning based software defect prediction techniques. Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'05), pp 263–270

[6] El-Emam K, Benlarbi S, Goel N, Rai SN (2001) Comparing case-based reasoning classifiers for predicting high-risk software components. J System Software 55(3):301–320. doi:10.1016/S0164-1212(00)00079-0

[7] Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques. Morgan Kaufmann

[8] Zhang H, Zhang X (2007) Comments on 'data mining static code attributes to learn defect predictors'. IEEE Trans Softw Eng 33(9):635–637

[9] Ling CX, Li C (1998) Data mining for direct marketing: problems and solutions. Proc. of the 4th Intern.Conf. on Knowledge Discovery and Data Mining, New York, pp 73–79

[10] Davis J, Goadrich M (2006) The relationship between precision-recall and ROC curves. Proceedings of the 23rd International Conference on Machine Learning. Pittsburgh, PA, pp 233–240

[11] Adams NM, Hand DJ (1999) Comparing classifiers when the misallocation costs are uncertain. PatternRecognit 32:1139–1147. doi:10.1016/S0031-3203(98)00154-X

[12] Siegel S (1956) nonparametric statistics. McGraw-Hill, New York

[13] Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

[14] Macskassy S, Provost F, Rosset S (2005a) Point wise ROC confidence bounds: an empirical evaluation. Proceedings of the Workshop on ROC Analysis in Machine Learning (ROCML-2005)

[15] Macskassy S, Provost F, Rosset S (2005b) ROC confidence bands: an empirical evaluation. Proceedings of the 22nd International Conference on Machine Learning (ICML). Bonn, Germany

[16] Drummond C, Holte RC (2006) Cost curves: an improved method for visualizing classifier performance. Mach Learn 65(1):95–130. doi:10.1007/s10994-006-8199-5

1		Meer Tauseef Ali[1], Research Scholar,(PP.COMP.SCI.O398) Department of Computer Science, Rayalaseema University, Kurnool, A.P.
2		Dr .Syed Abdul Sattar Professor, Principal, Nawab Shah Alam Khan College of Engg. & Tech., Hyderabad.