A Comparative Study of Threshold-Based and Machine Learning Models

Navneet Kaur[1], Deepali Bassi[2]

[1]*Department of Computer Science and Applications, Khalsa College, Amritsar*
[2]*Department of Computer Science and Applications, Guru Nanak Dev University, Amritsar*

---***---

Abstract -Fault prediction in software systems is essential for ensuring software quality and reliability. Traditionally, binary classification based on threshold values has been seen as a simpler and more intuitive method than machine learning and statistical techniques. Threshold-based methods allow developers and testers to classify software components as faulty or non-faulty by checking if specific metric values exceed set thresholds. However, it is important to determine if threshold techniques can achieve prediction performance that matches machine learning algorithms. This paper looks into this issue by building threshold-based models using the concordance probability method, Alves Rankings, and the Vales method, and compares their results with those from machine learning and statistical methods like Naive Bayes, Logistic Regression, AdaBoost, Random Forest, and Bagging. The results show the better classification ability of threshold techniques as compared to machine learning based classifiers.

Key Words: Fault prediction, machine learning, threshold techniques, ROC curve

1. INTRODUCTION

The process of binary classification based on the concept of threshold is easier than the repetitive process of constructing models, as in the former case, developers and testers can sense the peculiarity in the class just by checking if its metric value exceeds the calculated threshold [1]. Although, researchers claimed the effectiveness of threshold concept as compared to Machine Learning (ML) and statistical methods, as the former method can be easily used by the software development team to maintain the quality of the software system, however, it is also mandatory to verify whether the threshold techniques can produce the discrimination results as good as the ML algorithms. Therefore, in this paper, the authors have compared the prediction peRFormance of the threshold techniques with ML and statistical techniques.

In this paper, the threshold based models is developed by using the concordance probability method, Alves Rankings, and Vales method and their prediction outcome is compared with the results of Naive Bayes (NB), Logistic Regression (LR), Adaboost, Random Forest (RF), and Bagging. The concordance probability method is selected in this study due to its capability to produce the best prediction results [2]. Furthermore, the Vales method is evaluated for its ability to determine an effective threshold in splitting the faulty and non-faulty classes. This method was originally introduced to find the lazy and god classes of the software system [3]. God class performs too much task, whereas, lazy class performs little task in the software system. In (Vale & Figueiredo, 2015), the authors reported results signalized the best outcome in case of Vales method as compared to the Alves Rankings method for detecting the lazy and God classes. In order to validate the published results of the referenced study in the fault prediction domain, the current study compares the prediction efficiency of the Vales method with that of the Alves method.

The common method to predict faulty classes using a threshold value includes declaring a class as faulty when a particular software metric exceeds its threshold value. But in [4], the authors also tested the scenario when more than one software metric exceed their threshold values. The authors declared a class as faulty when at least single software metric exceeds its threshold value and evaluate the results. The authors declared a class as faulty when at least single software metric exceeds its threshold value and evaluate the results. Similarly, performance was also evaluated when at least two, three and four software metrics were exceeding their threshold values. The best results were obtained when the number of software metrics that exceeded the threshold values reached three or four. The authors also found that with an increase in the number of software metrics, the number of false positive predictions gets low and the number of false negative predictions gets high. In this paper, the authors adopted the same evaluation criteria to assess the suitability of the threshold and ML algorithms for the classification purpose. Herein, ROC-1 denotes a class as faulty when at least one of the selected software metrics exceeds its threshold value obtained through the ROC method. Likewise, ROC-2, ROC-3, and ROC-4 denote a class as faulty when at least two, three, and four software metrics exceed their respective threshold values. Similarly, ALVS-1, ALVS-2, ALVS-3, and ALVS-4 represent the results obtained through the Alves Rankings method, while VAL-1, VAL-2, VAL-3, and VAL-4 represent the results obtained through the Vales method.

The main contributions of this paper are

1. To compare the prediction efficiency of ML based and threshold based models.

2. To explore the contribution of the Vales method in identifying the optimal threshold value. The prediction model has been constructed using Vale method and the results have been compared with Alves based models. Section 5 contains the results and Section 6 concludes the paper.

The rest of the paper is organized as follow. Section 2 contains review of the studies conducted in the related area. Section 3 contains the review of the fault prediction related studies. Section 4 introduces the classification algorithms.

2. RELATED WORK

This section contains the review of existing studies conducted in the fault prediction area.

Grcer and Tarhan (2025) applies five machine learning models—RF, GB, NB, MLP, and NN—for software defect prediction and integrates an explainable AI (XAI) framework to enhance interpretability [5]. Using six XAI methods on the KC2 dataset, the research provides both global and local explanations, reducing the “black-box” nature of ML models and offering clearer insights into defect prediction.

In [6], the authors responded to the need for software quality and defect prediction to prevent expensive maintenance. The authors compared algorithms like GNB, Bernoulli NB, RF, and MLP, and suggested a PCA-based ensemble with class balancing as well. Performance was measured in terms of metrics such as accuracy, precision, and recall.

Shatnawi et al. proposed the ROC curve for threshold-based classification in fault prediction, choosing the point with the highest sum of specificity and sensitivity [7]. While thresholds of 12 OO metrics effectively partitioned classes by fault severity, they failed to divide Eclipse classes into faulty and non-faulty. Catal et al. also applied the ROC approach but maximized for maximum area under the curve [8]. Ferreira et al. developed yet another method based on data fitting to probability distributions [9].

Boucher and Badri (2018) compared ROC, Alves Rankings, and VARL, and concluded that ROC had the highest predictive efficiency, next was Alves, whereas VARL thresholds were inadmissible in over half of the cases [4]. Subsequently, Kaur and Singh (2020) compared ROC and Alves Rankings on 20 OO measures and found much superior results for ROC for measures like RFC, NPM, LOC, CAM, and AMC [10].

3. THRESHOLD TECHNIQUES

In this paper, threshold-based models are constructed using three techniques: the concordance probability method, the Alves Rankings method, and the Vales method. Although the classification ability of the Alves method is lower than that of the concordance probability method, it is selected for threshold identification in this study due to its advantage of deriving threshold values without relying on previous fault data. Some companies are unable to collect fault information because of the high costs associated with data collection tools. In such cases, unsupervised methods can be employed to extract threshold values. Therefore, in this paper, along with the Alves method, another unsupervised method proposed by in the study [3], is also tested for the identification of optimal threshold values. The Vales method is similar to the Alves Rankings method as it can show the proficient results in case of skewed metric values. However, it does not take into account the size metric in order to find the threshold value. The steps to find an optimal value using Vales method are explained as follows.

a) The first step is the extraction of values of the software metric whose threshold value is to be calculated.

b) The next step is the calculation of weight ratio of each entity (here entity represents class), which can be computed by dividing the class weight by the total number of classes and the result is multiplied by 100. Each class is assigned with the same weight and their aggregation should produce 100%. For e.g., if dataset contains 1000 classes, then the weight ratio of each class will be 0.1% [0.1%*1000=100].

c) The next step is entity aggregation, this step requires the summation of weight ratios of the classes having similar value of the selected software metrics. For e.g., if the dataset contains five classes with CBO value 6 and weight ratio of each class is 0.1, then the results after entity aggregation will be 0.5.

d) The last step involves the threshold derivation. It requires the construction of a cumulative line chart, where the X-axis represents weight ratio and the Y-axis represents metric value. Here the metric value corresponding to the weight ratio producing efficient results is selected as an optimal threshold value.

4. CLASSIFICATION ALGORITHMS

The following statistical and machine algorithms are considered and evaluated for their relative performances on the fault dataset. The selected algorithms are NB, LR, Adaboost, RF, and Bagging.

NB is a simple probabilistic classifier that trains the model by assuming that features are independent of each other. Although, the stated assumption seems unrealistic, however the technique works efficiently and produces results as well as other sophisticated classifiers.

LR is the standard technique to handle the classification problem by providing the estimates of event outcome probability. It measures the association between the dependent variable and one or more independent variables by estimating the likelihood of occurrence of certain case using a logistic function. LR has been selected n the multitude of studies for identifying the faulty classes.

Adaboost is a prevalent method for constructing an ensemble classifier. It is a process of making classification by creating several weak classifiers also known as decision stumps and aggregating them into a strong classifier. The stumps are weak in the sense as they use only one variable to make a decision. The next weak classifier is created by taking into account the mistake made in the last classifier.

Another method utilized in this paper is Bagging, which was put forward by Breiman, with the motive to enhance the classification outcome by randomly generated trainings set [11]. Decision Trees (DT) are known for their ease in construction, use and interpretation. However, they can efficiently manage the problem of over-fitting by using the pruning method. But, in the arena of predictive learning, DT produce inaccurate results because of their inflexibility in classifying the new samples. Consequently, RF can be used which possess easiness property of DT along with their improved accuracy. RF works by constructing a large number of DT from the training dataset and makes predictions by aggregating the predictions resulting from these DT. The process involves the construction of bootstrapped dataset from the original dataset. For this, samples are selected randomly from the original dataset. The size of bootstrapped dataset should be equal to that of original one, in this process the duplicated samples can be chosen i.e. the former can pick any sample more than one time. Then, a DT is created from the bootstrapped dataset. The process of creating a DT is repeated for another bootstrapped dataset many times. After the formation of a forest, test data is run down on all trees and mark the option (true or false) containing the highest number of votes.

5. RESULTS

In this paper, the selected classification algorithms are used to find the optimal cut-off values of CK and LOC metrics. The experiment is executed on Ant-1.7, Arc, Prop-6, Synapse-1.2, Tomcat, Velocity, Antlr, BroadleafCommerce, Junit, McMMO, and Netty. The metrics selected based on the univariate analysis are WMC, CBO, RFC, and LOC. These metrics were showing their significant relationship with the target class in a high number of the software systems.

5.1 *Threshold values*

The threshold values obtained with the application of chosen threshold methods are given in Table-1. In the case of the Alves method, selecting the appropriate weight ratio is essential, and the metric values corresponding to a weight ratio of 20 produce the best results. In this paper, the metric value corresponding to the same weight ratio is also selected as the optimal threshold value. A similar approach is adopted to find the optimal threshold values in case of Vales method. To determine the appropriate weight ratio the G-mean results of the thresholds acquired at weight ratios 10%, 20%, up to 100% are compared, and the ratio producing the highest G-mean results are considered as an optimal weight ratio. The findings revealed that the metric values corresponding to ratio 50% gives the best classification outcome, therefore, the metric values corresponding to this ratio are selected as an optimal threshold value.

Table-1 Threshold values computed using Concordance, Alves, and Vales method

	Concordance Probability				Alves Rankings (20%)				Vales method (50%)			
Soft-ware	WMC	CBO	RFC	LOC	WMC	CBO	RFC	LOC	WMC	CBO	RFC	LOC
Ant-1.7	10	7	32	178	8	5	30	221	9	7	26	194
Arc	8	12	19	132	9	5	22	122	8	6	17	122
Prop-6	10	8	18	229	5	6	19	134	5	8	19	134
Synapse -1.2	5	12	31	200	4	9	27	174	5	11	28	181
Tomcat	11	6	18	375	9	3	27	248	9	5	27	181
Velocity	4	8	12	210	8	4	28	239	6	8	28	128
Antlr	23	7	20	101	10	3	12	125	8	5	9	58
BroadleafCommerce	14	3	7	63	9	1	8	66	10	3	8	64
Junit	3	4	3	27	7	4	10	92	2	3	3	15
McMMO	11	3	12	62	13	2	9	81	13	4	12	80
Netty	3	5	10	34	7	2	9	76	6	5	10	54

5.2 *Performance results*

In this paper, three performance parameters, i.e., FPR, FNR, and G-mean, are considered to express the predictive performance of the prediction models.

In Ant-1.7, ROC-2 shows the best classification ability to discriminate the faulty and non-faulty classes followed by Alvs-3 and VAL-4 (as shown in Fig-1). The lowest false predictions (FPR- 0.328 and FNR- 0.115) and the highest G-mean value, i.e. 0.771, are obtained when at least two of the software metrics transcend the threshold values of the corresponding metrics obtained using the ROC method. With the execution of ROC-3, the false positive predictions slightly decline, i.e. 0.328, as compared to ROC-2 and false negative predictions slightly rise, i.e. .115. For ROC-1, average results are obtained when a class is declared faulty if at least one software metric exceeds the computed cut-off value. No classification ability is observed for ROC-4 due to its high rate of false negative predictions. The Alves method produces the best results when at least three software metrics exceed the derived threshold values, achieving a G-mean of 0.758. Similarly, for the Vales method, the best results are observed for VAL-2 and VAL-3, with G-mean values of 0.74. Among the selected machine learning techniques, AdaBoost reports the best classification results, with a G-mean of 0.715, while the others produce only average classification performance. Although AdaBoost shows the strongest discrimination ability among the machine learning techniques, its performance remains lower than that of the threshold-based models.

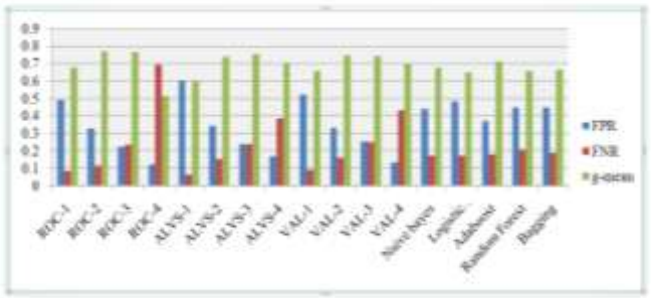


Fig -1: Results of threshold and ML based models on Ant-1.7

The prediction results obtained by the threshold-based and ML models are displayed in the Fig-2. In case of Arc, the ML based models fail to achieve the required classification ability. Although, the threshold based models produced better results than ML based models, but the former still fail to produce the acceptable classification power. The threshold based model produced the maximum of average classification ability in case of ROC-2 and ROC-3 with G-mean score of 0.67, in case of ALVS-2 with G-mean score of 0.636 and in case of VAL-3 with G-mean value of 0.652. As compared to threshold based models the techniques like NB, LR, Adaboost, RF, and Bagging showed considerably low prediction outcome.

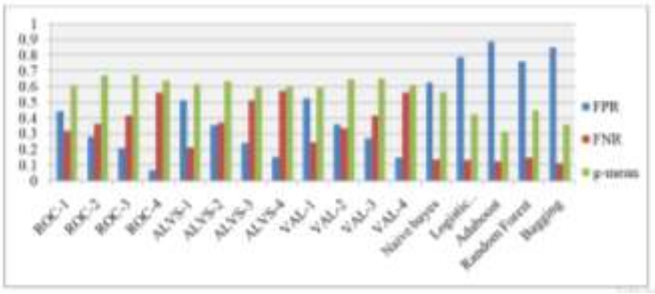


Fig -2 Results of threshold and ML based models on Arc

In case of Prop, model built using ROC gives the best result (as depicted in Fig-3). The acceptable results are also observed in the case of Vales method. The results produced by Alves method are showing average ability to classify the faulty and non-faulty groups. LR, Adaboost, RF, and Bagging fail to classify the software classes into the accurate group, whereas, Naïve Bayes shows the average classification ability to separate the software classes. Therefore, based on the above results it can be concluded that between the threshold based and ML, the former models are better choice for the development of fault prediction models.

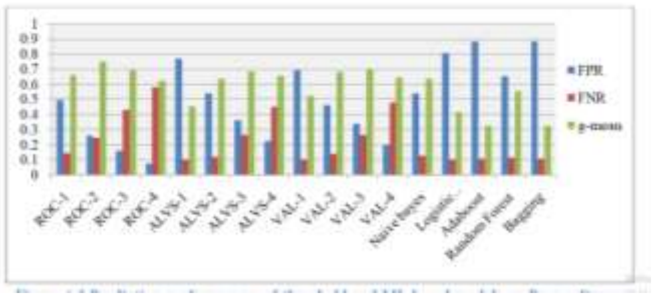


Fig -3 Prediction performances of the selected models on Prop

In case of Synapse, Adaboost gives slightly better results than other ML based models (as shown in Fig-4). The G-mean value obtained by Adaboost is 0.753, followed by ROC-3 (with G-mean score of 0.743), VAL-3 (with G-mean score of 0.739), ALVS-3 (with G-mean score of 0.727), and RF (with G-mean value of 0.715). Therefore, no significant difference can be observed in the prediction results of ROC and Adaboost methods. All of these previously mentioned methods showed acceptable results, however, the average results were shown by other ML techniques (excluding Adaboost and RF).

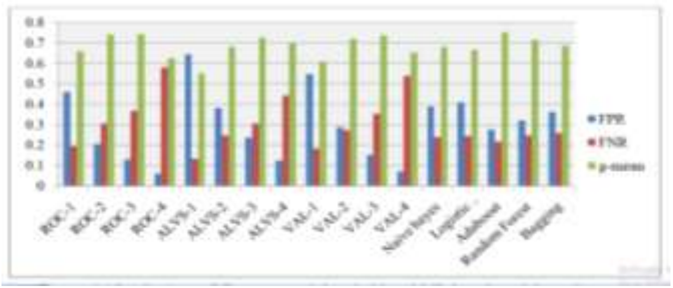


Fig -4 Prediction performance of threshold and ML based models on Synapse

Similarly, In Tomcat, the prediction model developed using ROC and Vales method produced the best result, as here the excellent results were revealed when at least three software metrics exceeds the derived threshold values. The detailed prediction peRFormance hold by the different classification models is depicted in Fig-5. The ML models such as LR, Adaboost, Bagging showed no classification ability, RF revealed a poor ability, and Naïve Bayes achieved average results. Therefore, in case of Tomcat, the threshold based models provided significantly better results than ML based models.

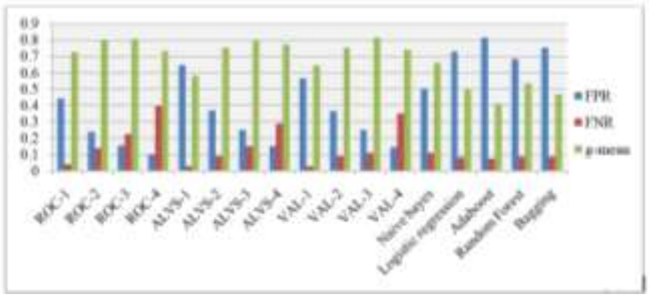


Fig -5 Prediction performance of the selected models on Tomcat

In case of velocity (as shown in Fig-6), the threshold based models gave the best results as compared to the ML based models. Among ROC, Vales, and Alves, the prediction models developed using the former two techniques gave the best classification results. In case of ML algorithms, Naïve Bayes and LR depicted poor classification ability, whereas, RF and Bagging managed to produce the average discrimination outcome.

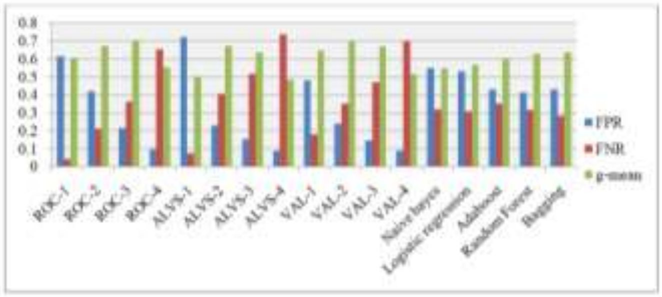


Fig -6 Performance of the selected models on Velocity

In Antlr (as shown in Fig-7), the threshold based models developed using ROC and Alves method showed the excellent discrimination result, whereas, ML lagged behind the threshold based models in terms of classification efficiency. The ML techniques such as Adaboost, RF, and Bagging showed no classification ability. Similarly, the models built using LR and Naïve Bayes failed to produce an efficient discrimination outcome.

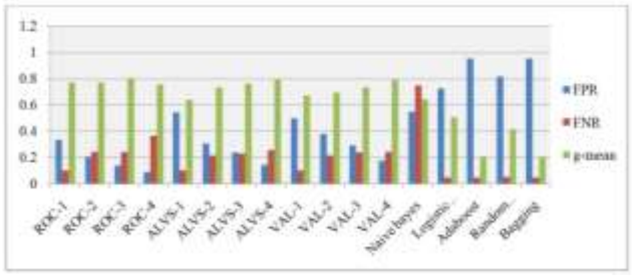


Fig-7 Prediction performance of the selected models on Antlr

In Broadleaf Commerce (as depicted in Fig-8), among the selected threshold based models, the models constructed using ROC and Alves give the best results. As compared to threshold based models, the model developed using ML techniques depicts significantly low prediction ability. The misclassification rate of LR, AdaBoost, RF, and Bagging is too high; most of the software classes that are predicted as faulty are actually non-faulty. Although NB demonstrates better classification ability compared to its counterparts, its performance is still not up to the mark.

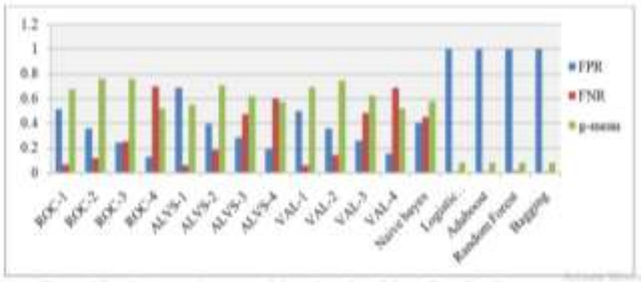


Fig-8 Performance of the selected models on BroadleafCommerce

Similarly, in case of Junit, the threshold based models outperforms the ML based models with significant performance difference. The detailed performance outcome is depicted in Fig-9. Here, ROC and Alves based models shows the excellent prediction ability. All of the ML based models shows very high false positive rate, which further impacts the overall classification performance**.**

In case of McMMO (as shown in Fig-10), slightly different results are observed as compared to the previous software systems, as here two of the ML techniques, i.e., RF and bagging, shows the prediction results as good as the ROC based threshold model. On the other hand, the prediction performance of NB, LR and boosting algorithm is not satisfctory.

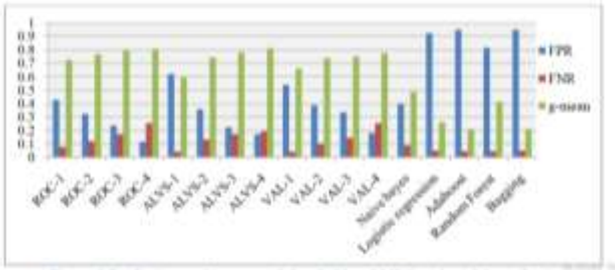


Fig-9 Performance of threshold and ML based models on Junit

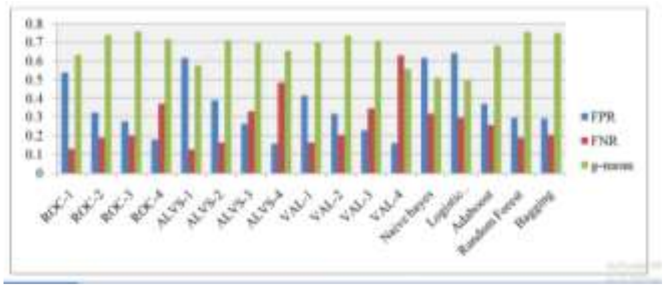


Fig -10 Performance of the selected models on McMMO

In case of Netty, the ROC based model gives the best results followed by Vales and Alves methods. The G-mean score of the threshold based and other classifiers are shown in Fig-11. The discrimination strength of the RF-based model is close to that of the threshold-based models. By contrast, NB, LR, and AdaBoost depict no classification ability.

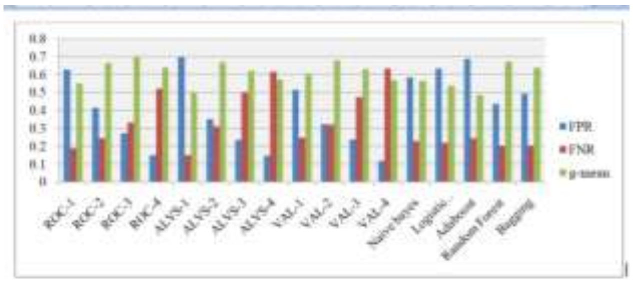


Fig -11 Performance of the selected models on Netty

Based on the previous results, it can be observed that threshold based models show better prediction results as compared to ML algorithms. Furthermore, among the selected threshold based models, the one constructed using ROC curve produces the best results. No significant difference is found between the prediction performance of the Alves and Vales methods, as in some cases the former gives better results and in other cases the latter produces a better outcome. As opposed to the [3] where the Vales method was proven significantly better than the Alves method in detecting the God and Lazy classes, in the fault prediction domain the Vales method fails to show a significant difference from the Alves method.

Overall, the selected threshold-based methods show better prediction performance than ML. The acceptable results are observed only for AdaBoost in Ant, AdaBoost and RF in Synapse, and RF and Bagging in McMMO.

5.3 Comparison results

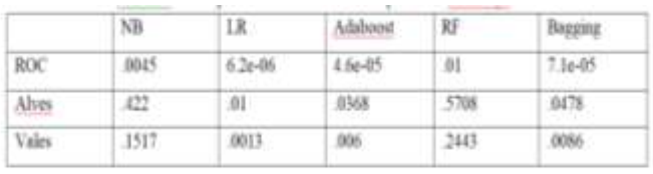
Although, it is imperative from the results, obtained so far, that threshold techniques perform better than ML techniques for the detection of faulty classes. However, Freidman test is applied to verify whether the performance difference between them is actually significant. Among ROC-1, ROC-2, ROC-3, and ROC-4, the model that gave the highest G-mean score was selected for the comparison process. The similar selection criterion is applied to Alves and Vales models. The null and alternate hypotheses of Friedman test based on this paper are as follows.

Null hypothesis: Both ML and threshold techniques shares same predictive capability.

Alternate hypothesis: The predictive efficiency of ML and threshold techniques is not same.

The null hypothesis of Friedman test was rejected with p-value 0.000, which indicates that there exists a significant difference between the predictive performances of selected techniques. The outcome of post-hoc Nemenyi test is shown in Table-2. The results reported the effectiveness of the ROC method than all of the selected ML techniques. The Alves based model succeeded to hold significant difference with only LR, Adaboost, and bagging. Similar findings were observed in the case of Vales method, as here also no significant difference was found with the prediction performance of NB and RF.

Table -2: Comparison results based on post-hoc Nemenyi test



	NB	LR	Adaboost	RF	Bagging
ROC	.0045	6.2e-06	4.6e-05	.01	7.1e-05
Alves	.422	.01	.0368	.5708	.0478
Vales	.1517	.0013	.006	.2443	.0086

6. CONCLUSIONS

This paper contains the comparison of the classification ability of the models constructed using threshold and ML techniques. The threshold techniques selected for the comparison purpose are concordance, Alves, and vales method. The ML methods chosen are NB, LR, Adaboost, RF, and Bagging. In the comparison between threshold-based and ML-based models, the threshold-based model built using the concordance method gives significantly better results than all of the selected ML techniques, whereas the prediction results of the Alves and Vales based models are significantly better only than LR, AdaBoost, and Bagging. Furthermore, no significant difference is found between the Alves and Vales methods.
This paper has considered only CK and LOC metric. The future work can be conducted on more software metrics.

REFERENCES

1. Shatnawi, R., Li, W., Swain, J., Tim, N.: Finding software metrics threshold values using ROC. J. Softw. Maint. Evol.: Res. Pract. **22**, 1–16 (2010).
2. Kaur, N., Singh, H.: An empirical assessment of threshold techniques to discriminate the fault status of software. J. King Saud Univ.-Comput. Inf. Sci. **34**(8), 6339–6353 (2022)
3. Vale, G., Figueiredo, E.: A method to derive metric thresholds for software product lines. In: Proc. 29th Brazilian Symposium on Software Engineering (SBES), pp. 110–119 (2015).
4. Boucher, A., Badri, M.: Software metrics thresholds calculation techniques to predict fault-proneness: An empirical comparison. Inf. Softw. Technol. 96, 38–67 (2018).
5. Gezici Geçer, B., Kolukısa Tarhan, A.: Explainable AI framework for software defect prediction. J. Softw.: Evol. Process **37**(4), e70018 (2025).
6. Setia, S., Ravulakollu, K.K., Verma, K., Garg, S., Mishra, S.K., Sharan, B.: Software defect prediction using machine learning. In: Proc. 11th Int. Conf. on Computing for Sustainable Global Development (INDIACom), pp. 560–566. IEEE, February 2024.
7. Shatnawi, R.: A quantitative investigation of the acceptable risk levels of object-oriented metrics in open-source systems. IEEE Trans. Softw. Eng. **36**(2), 216–225 (2010).
8. Catal, C., Sevim, U., Diri, B.: Practical development of an Eclipse-based software fault prediction tool using Naive Bayes algorithm. Expert Syst. Appl. **38**(3), 2347–2353 (2011).
9. Ferreira, K.A.M., Bigonha, M.A.S., Bigonha, R.S., Mendes, L.F.O., Almeida, H.C.: Identifying thresholds for object-oriented software metrics. J. Syst. Softw. **85**(2), 244–257 (2012).
10. Kaur, N., Singh, H.: An empirical analysis of threshold techniques for identifying faulty classes in object-oriented software systems. Int. J. Next-Gener. Comput. **11**(3) (2020).
11. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001).