Machine learning based software fault prediction utilizing source code metrics

Guru Prasad Bhandari
DST-Centre for Interdisciplinary Mathematical Sciences
Institute of Science, Banaras Hindu University
Varanasi, India
guru.bhandari@gmail.com

Ratneshwer Gupta
School of Computer & Systems Sciences
Jawaharlal Nehru University
New Delhi, India
ratnesh@mail.jnu.ac.in

Abstract—In the conventional techniques, it requires prior knowledge of faults or a special structure, which may not be realistic in practice while detecting the software faults. To deal with this problem, in this work, the proposed approach aims to predict the faults of the software utilizing the source code metrics. In addition, the purpose of this paper is to measure the capability of the software fault predictability in terms of accuracy, f-measure, precision, recall, Area Under ROC (Receiver Operating Characteristic) Curve (AUC). The study investigates the effect of the feature selection techniques for software fault prediction. As an experimental analysis, our proposed approach is validated from four publicly available datasets. The result predicted from Random Forest technique outperforms the other machine learning techniques in most of the cases. The effect of the feature selection techniques has increased the performance in few cases, however, in the maximum cases it is negligible or even the worse.

Keywords—Fault, Fault prediction, Software, Machine Learning, Feature Selection, Software Metrics;

I. INTRODUCTION

Literally, fault is an abnormal condition in the component, sub-systems or module of the system [1]. Failure can be defined as the state or condition of not meeting a desired or intended objective such as a software stops performing its required operation. As increasing the size and the complexity of the software, the identification of the fault location in a timely manner is important otherwise the cost of correcting its severity increases exponentially in the later phases of the software development. This encourages the advancement of fault prediction models that can predict the software fault i.e. whether it is faulty or nonfaulty.

Machine learning techniques are being popular among the researchers and the practitioners in variety of areas. Basically, learning is the process of knowledge acquisition as human naturally [2]. In this study, some of the machine learning techniques suggested in the literature are applied over different benchmark datasets related to the fault prediction of the software. Few feature selection strategies have also been applied to the datasets to reduce the features and only extracts the most relevant features.

Chidamber and Kemerer (CK) metrics [3], Object-Oriented (OO) metrics [4], Process metrics, McCabe metrics [5], HalStead metrics [5] are the most frequently used metrics for software quality estimation such as fault prediction, effort and performance estimation to enhance the software maintainability. Software fault prediction in the early stages of the software development can assist to the software practitioners to emphasize on the weaker areas. Thus, this paper presents the analysis and comparison of several machine learning techniques for the software fault prediction.

The remainder of this paper is structured as follows. Section II presents the related works contributed by the researchers in the area of software fault prediction. Feature selection techniques with ranking methods are explained in Section III. And, Section IV presents experimental evaluation and results. Finally, Section V concludes the paper.

II. RELATED WORKS

Machine learning techniques are being used in software fault prediction to assist testing and maintainability. Fault prediction approaches are explored by researchers in the literature [6] [7] [8] [9] [10]. Several researchers [11] [12] have also contributed in distributed software systems using machine learning. When the PROMISE data repository was released in 2005 [13], the number of studies in software fault prediction increased significantly [14].

In the literature, Kumar et al. [15] have constructed a machine learning based model to predict the software fault prediction. They have used 30 projects of PROMISE data repository by taking 20 CK metrics and McCabe metrics. Their work has focused on Least square support vector machine with linear, polynomial and radial basis kernel functions. Likewise, Twala [9] has performed software fault predict using Decision Tree, K-Nearest Neighbour, Naïve Bayes and Support Vector Machine (SVM). In experiment, 21 CK metrics, McCabe metrics and HalStead metrics of 3 NASA projects from PROMISE data repository are taken for the performance calculation, it has been shown that Decision Tree produced most accurate result than other techniques. Malhotra [16] has taken 2 projects of PROMISE and NASA data repository (AR1 and AR6) to compare and check the applicability of the software fault prediction using 36 (McCabe, HalStead, LOC (Line of Code) metrics and Miscellaneous) metrics of source code. It has been observed from her experiment that with Logistic regression, Artificial Neural Network (ANN), SVM, Decision Tree (DT), Cascade Correlation Network (CCN), GMDH (Group Method of Data Handling) polynomial network, Decision Tree performed the best result than other machine learning techniques.

Boucher and Badri [17] have performed a comparative study on software metrics thresholds calculation techniques to predict fault-proneness. Three thresholds calculation techniques; ROC (Receiver Operating Characteristic) curves, VARL (Value of an Acceptable Risk Level) and Alves rankings are compared. The experiment on 12 PROMISE and Eclipse projects shows that ROC curves is the best

978-1-5386-6227-4/18/$31.00 ©2018 IEEE

TABLE 1 Characteristics of the software fault prediction datasets

System	Language	No. of attributes	No. of Instances	Total LOC	No. of undefected Modules	No. of fp modules	%Faulty
JM1	C	22	10885	457346	8779	2106	19.35 %
KC1	C++	22	2109	42965	1783	326	15.46 %
PC1	C	22	1109	25924	1032	77	6.94 %
jEdit	Java	9	369	83127	204	165	44.72 %

performing method among three thresholds calculation methods investigated. Similarly, Moeyersoms et al. [18] have performed comparative study on Random Forests, Support Vector Machine, C4.5 and Regression Tree machine learning techniques. They have used four publicly available datasets from Android with 27 metrics and three from PROMISE data repository with 21 and 37 metrics. In experiment, it has been found that Random forest has the highest accuracy among other machine learning techniques.

This study further extends the above contributions by considering majority of the performance parameters to check the validity of the proposed approach. We have considered four benchmarked datasets and experimented them using several machine learning techniques along with two feature selection techniques to measure the capability of the machine learning techniques for software fault prediction. In the following section, we present feature selection techniques investigated in this study.

III. FEATURE SELECTION TECHNIQUES

In the following subsections, feature selection techniques are briefly explained which are inspected in this study.

A. Principal component analysis

Principal Component Analysis (PCA) is an applied linear algebra technique used for dimensionality reduction procedure. It is also known orthogonal linear transformation that transforms the data to a new coordinate system. PCA is also called a feature extraction technique of the dataset. The new features are extracted by a linear combination of the original features. In PCA, the features with the highest variance are taken to do the reduction [19]. The PCA transforms n_s vector $\{x_1, x_2, \ldots\ldots, x_n\}$ from the d-dimensional space to n vectors $\{x'_1, x'_2, \ldots\ldots, x'_n\}$ in a new d' dimensional space.

$$x' = \sum_{k=1}^{d'} a_k, ie_k, d' \leq d,$$

where e_k are eigenvectors corresponding to d' largest eigen vectors for the scatter matrix S and $a_{k,i}$ are the projections of the original *vectors* x_i on the eigenvectors e_k .

B. Correlation-based feature selection

Evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. Subsets of features that are highly correlated with the class while having low intercorrelation are preferred [20] in correlation-based feature selection (CFS).

IV. EXPERIMENTAL EVALUATION AND RESULTS

In this section, a brief explanation about independent and independent variables, datasets, simulation setup, performance measures for experimental evaluation of the proposed model is presented.

A. Dependent and independent variables

In this study, fault-proneness is the binary dependent variable which is either faulty or non-faulty (true or false/defeated or undefeated). Software code metrics are the independent variables that assist to predict whether the instance is faulty or non-faulty. In *JM1, KC1* and *PC1* there are 21 independent variables and a dependent variable as a classifier. While, in *jEdit 4.2-4.3*, there are only 8 independent variables as software metrics and a dependent variable as a classifier.

B. Datasets

In this study, we have selected four small to large scale datasets from the PROMISE repository [13]. The largest dataset is *JM1* which belongs to real time predictive ground system project of the NASA, with 10,885 instances having 19% of the instances are defected. Whereas, *jEdit* represents the small-scale datasets with 369 instances, among them 44.72% of the instances are defected. *JM1, KC1* and *PC1* are consisting of 22 features as independent variables including 1 classifier as dependent variable. Whereas, *jEdit 4.2-4.3* (say *jEdit*) has 9 features including a classifier. Table 1 presents the characteristics of the software fault prediction datasets.

C. Simulation selection

All the experiments are performed in WEKA (Weka Environment for Knowledge Analysis) Ver. 3.8 [21], an open-source software and implemented in JAVA; it is developed in the University of Waikato, Hamilton, New Zealand and used for machine learning studies.

D. Performance measures/parameters

We used different performance measures/parameters to evaluate the classification performance. Each classifier's accuracy is calculated by taking the average of 10 accuracy scores for each fold during cross-validation. Recall/sensitivity or True Positive (TP) is a measure of the number of positive predictions with respect to the number of positive class values in the test data. Whereas, precision/specificity gives the number of positive prediction of the classifier with respect to the total number of positive class values predicted. *F*-measure/*F1*-score is a measure that calculates the harmonic mean of the precision and recall measures of the classifier

E. Results without using feature selection techniques

All the selected machine learning techniques for this study are comparatively evaluated by the above performance measures. Moreover, the ROC (Receiver Operating Characteristic) curves for each dataset are plotted. AUC (Area Under the ROC Curve) is also a measure to calculate the effectiveness of the fault prediction capability of the classifier. Table 2 (withoutFS columns) presents each classifier's performance statistics i.e. accuracy, *F*-measure, recall and precision for different machine learning techniques without any feature selection technique applied. In such case,

Random Forest outperforms others and produces the most accurate result followed by the ANN (however, in jEdit, SVM is the best), whereas, Naïve Bayes performed the worst accuracy.

F. Effect of feature selection techniques

The effect of feature selection techniques on the software metrics is evaluated by comparing the accuracy of classification techniques before and after applying the PCA (Principle Component Analysis) and CFS (Correlation-based Feature Subset Selection) along with *Ranker* and *GreedyStepwise* methods respectively.

1. Ranker: It is a common strategy to select the attributes which ranks attributes by their individual evaluations, used in conjunction with attribute evaluators (ReliefF, GainRatio, Entropy etc.) [22]. Ranking in which members of the ensemble are called low-level classifiers and they produce not only a single result but a list of choices ranked according to their likelihood [9]. And, high-level classifier chooses from these set of classes using additional information that is not usually available to or well represented in a single low-level classifier.

2. GreedyStepwise: It performs a greedy forward or backward search through the space of attribute subsets. May start with no/all attributes or from an arbitrary point in the space and stop when the addition/deletion of any remaining attributes results in a decrease in evaluation. Can also produce a ranked list of attributes by traversing the space from one side to the other and recording the order that attributes are selected [22].

Table 2 shows each classifier's performance statistics i.e. accuracy, *F*-measure, recall and precision for different machine learning techniques after the application of PCA and CFS, as well as without any feature selection techniques applied. When PCA applied to the datasets, Random Forest achieved the best accuracy followed by the ANN whereas, Naïve Bayes performed the worst accuracy.

The ROC curves without using any feature selection algorithms on the datasets, noticed that Random Forest achieved the highest AUC values in comparison with other classifiers in case of all datasets. SVM could not perform well. It can be observed that the AUC values after the application of PCA and CFS are increased. It is also observed that the AUC values of the classification increased by a margin of 0-9% when CFS was applied than in PCA as dimension reduction of the datasets. Random Forest achieved the highest AUC values in comparison with other classifiers. SVM could not perform well. This shows that Random Forest can work with CFS dimension reduction very well and accuracy is also obtained good, whereas SVM cannot perform well with multidimensional reduction in this case. The ROC curves obtained for fault prediction of the software using each machine learning techniques for the datasets after the application of PCA are shown in Fig. 1-4 over *JM1, KC1, PC1* and *jEdit* respectively. The ROC curves after the consideration of CFS are presented in Fig. 5-8 over *JM1, KC1, PC1* and *jEdit* respectively.

TABLE 2 Accuracy, F-measure, Recall and Precision for different algorithms without using any feature selection techniques and with consideration of PCA and CFS feature selection techniques

Algorithms	**Measures**	**withoutFS**				**PCA + Ranker**				**CfsSubsetEval + GreedyStepwise**			
		JM1	***KC1***	***PC1***	***jEdit***	***JM1***	***KC1***	***PC1***	***jEdit***	***JM1***	***KC1***	***PC1***	***jEdit***
DT	Accuracy	80.90	84.86	92.92	62.03	81.01	84.80	92.98	61.08	**81.01**	84.86	93.01	62.25
	F-measure	0.89	0.92	0.96	0.64	0.89	0.91	0.96	0.65	0.89	0.92	0.96	0.64
	Recall	0.97	0.97	0.99	0.64	0.98	0.97	1.00	0.66	0.98	0.97	0.99	0.63
	Precision	0.82	0.87	0.94	0.67	0.82	0.87	0.93	0.66	0.82	0.87	0.94	0.67
RF	Accuracy	**81.70**	**86.21**	**93.71**	**68.21**	**81.20**	**85.28**	**93.22**	**64.77**	80.95	**85.72**	**93.84**	**69.41**
	F-measure	0.89	0.92	0.97	0.71	0.89	0.92	0.96	0.68	0.89	0.92	0.97	0.73
	Recall	0.96	0.96	0.99	0.73	0.95	0.95	0.99	0.69	0.95	0.96	0.98	0.74
	Precision	0.84	0.89	0.95	0.71	0.84	0.88	0.94	0.68	0.84	0.88	0.95	0.72
NB	Accuracy	80.42	82.46	89.00	53.53	*80.32*	*82.83*	*89.73*	*54.59*	*80.39*	*82.40*	*89.17*	*55.56*
	F-measure	0.89	0.90	0.94	0.34	0.89	0.90	0.94	0.38	0.89	0.90	0.94	0.38
	Recall	0.95	0.91	0.93	0.23	0.95	0.92	0.95	0.26	0.94	0.91	0.93	0.26
	Precision	0.83	0.89	0.95	0.77	0.81	0.85	0.93	0.58	0.83	0.89	0.95	0.80
SVM	Accuracy	80.73	84.77	93.00	**62.81**	80.71	84.54	**93.06**	57.40	80.65	84.50	93.06	62.89
	F-measure	0.89	0.92	0.96	0.67	0.89	0.92	0.96	0.69	0.89	0.92	0.96	0.67
	Recall	1.00	1.00	1.00	0.69	1.00	1.00	1.00	0.87	1.00	1.00	1.00	0.70
	Precision	0.81	0.85	0.93	0.66	0.81	0.85	0.93	0.58	0.81	0.85	0.93	0.66
ANN	Accuracy	**81.00**	**85.65**	**93.52**	61.82	**81.08**	**85.08**	92.89	61.59	**81.05**	**85.75**	**93.59**	60.13
	F-measure	0.89	0.92	0.97	0.63	0.89	0.92	0.96	0.63	0.89	0.92	0.97	0.62
	Recall	0.99	0.98	0.99	0.62	0.98	0.98	0.99	0.62	0.98	0.98	0.99	0.61
	Precision	0.82	0.87	0.94	0.68	0.82	0.87	0.93	0.68	0.82	0.87	0.94	0.67
Adaboost	Accuracy	80.73	84.77	93.06	57.57	80.79	84.61	**93.06**	60.59	80.65	84.79	93.00	58.00
	F-measure	0.89	0.92	0.96	0.57	0.89	0.92	0.96	0.62	0.89	0.92	0.96	0.57
	Recall	1.00	0.99	1.00	0.53	1.00	0.98	1.00	0.61	1.00	0.99	1.00	0.53
	Precision	0.81	0.86	0.93	0.65	0.81	0.86	0.93	0.67	0.81	0.85	0.93	0.66

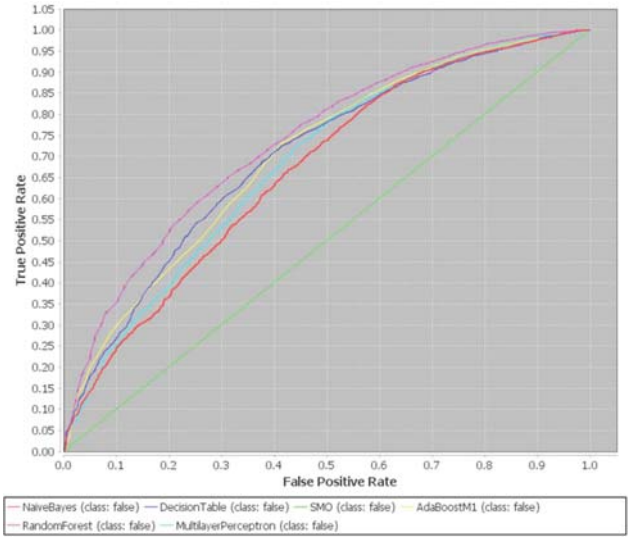


Fig. 1 ROC curves of the classifiers when PCA is applied on JM1

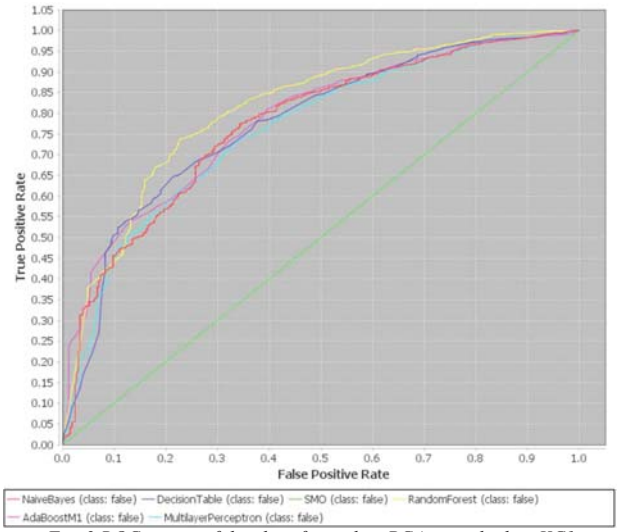


Fig. 2 ROC curves of the classifiers when PCA is applied on KC1

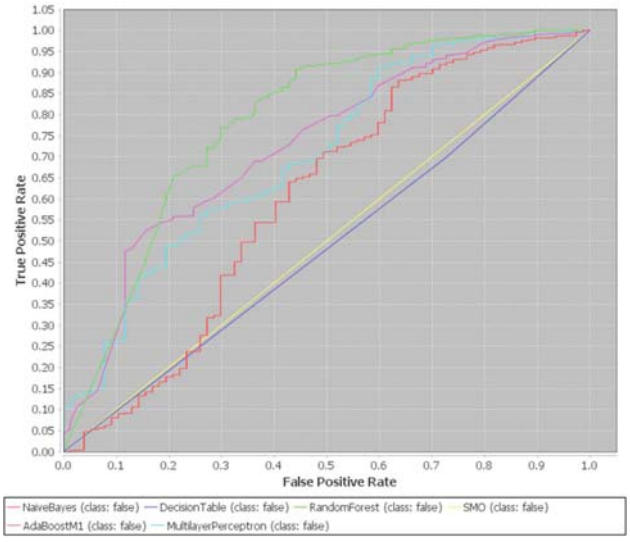


Fig. 3 ROC curves of the classifiers when PCA is applied on PC1

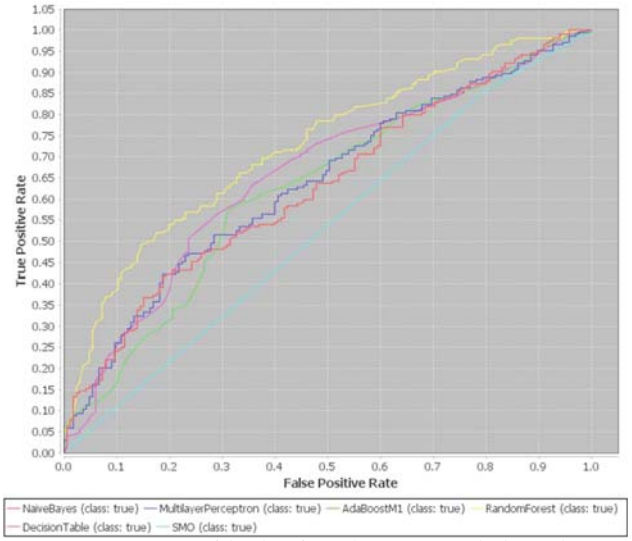


Fig. 4 ROC curves of the classifiers when PCA is applied on jEdit

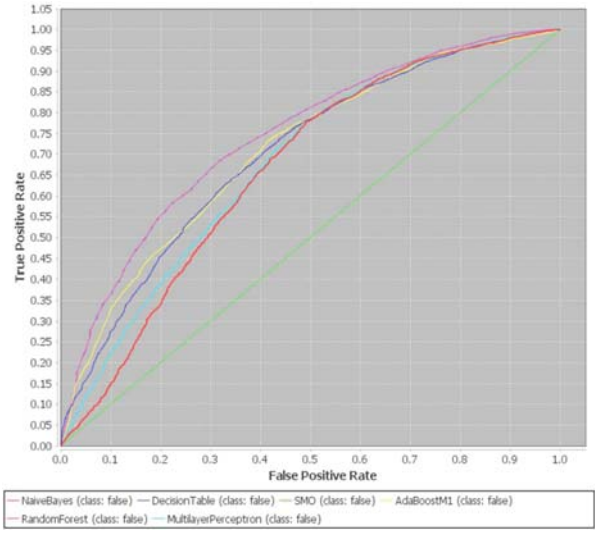


Fig. 5 ROC curves of the classifiers when CFS is applied on JM1

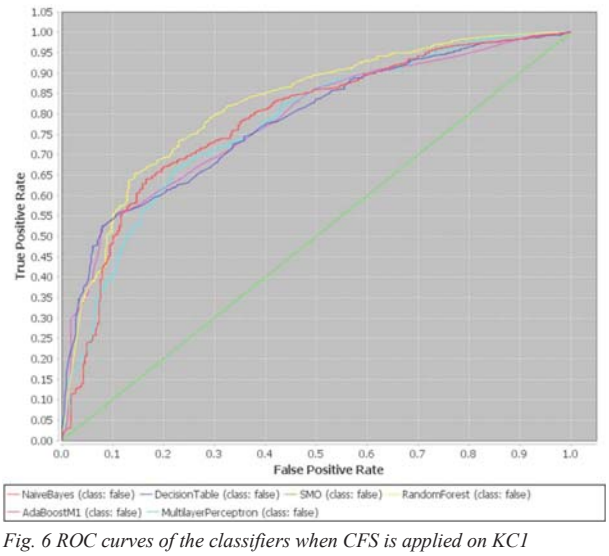


Fig. 6 ROC curves of the classifiers when CFS is applied on KC1

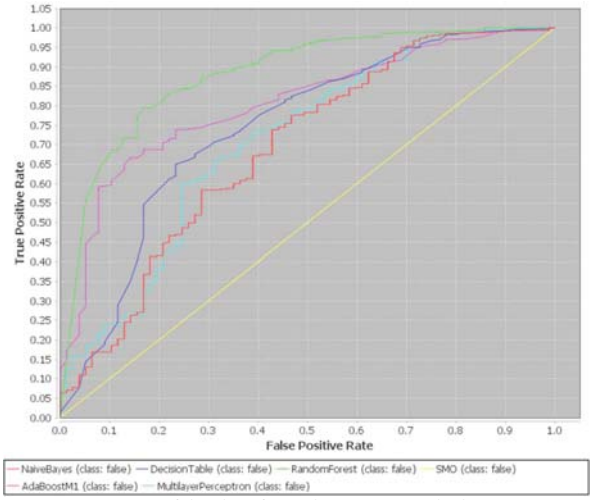


Fig. 7 ROC curves of the classifiers when CFS is applied on PC1

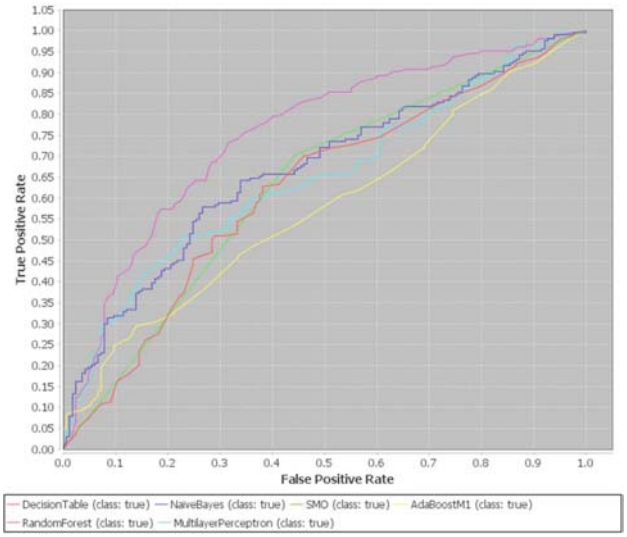


Fig. 8 ROC curves of the classifiers when CFS is applied on jEdit

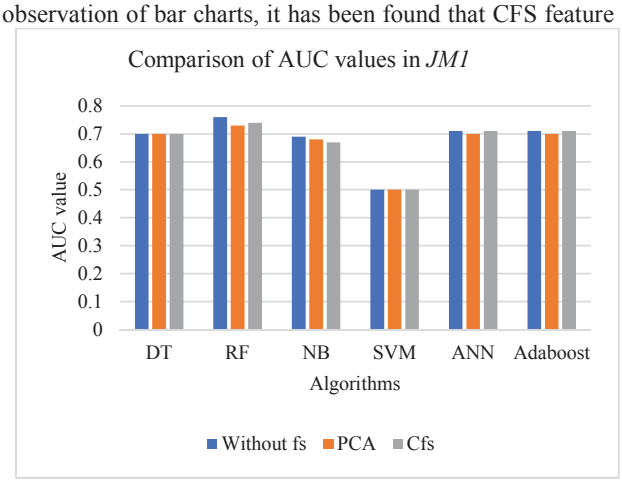
The bar charts comparing AUC values without using any feature selection algorithms and with PCA and CFS feature selection algorithms among different machine learning algorithms on the datasets are presented in Fig. 9 (*(a)JM1, (b)KC1)* and Fig.10 *((a)PC1 and (b)jEdit*). From the observation of bar charts, it has been found that CFS feature selection algorithm in most of the datasets works better than without using feature selection algorithms. For the software fault prediction, it can be said that CFS performs better than PCA from small to large-scale datasets. PCA is not recommended for the feature reduction purpose here because in most of the cases, it decreases the AUC values as it applied as feature selection technique.

Machine learning techniques i.e. Decision tree, Random forest, Naïve Bayes, Support vector machine, Artificial Neural Networks and Adaboost, in order to predict the software faults using the software source code are found to be the feasible fault prediction techniques. Through the experiment, we analyzed and compared the performance measures of the different machine learning techniques.

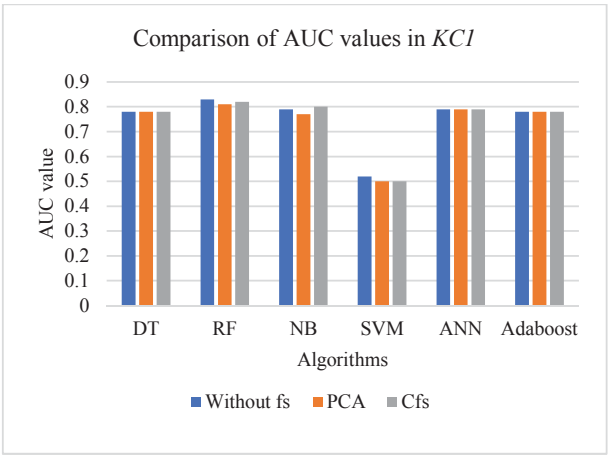
The result confirmed that the predicted result from Random Forest technique have the highest accuracy among other machine learning techniques in most of the cases. Similar nature of studies with large scale software could be performed to generalize and validate the results that is obtained by this study.

V. CONCLUSION

The main objective of our study was to examine capability of the machine learning techniques i.e. Decision tree, Random forest, Naïve Bayes, Support vector machine, Artificial Neural Networks and Adaboost, and to check the effectiveness of the feature selection techniques in order to predict the software faults using the software metrics. Through the paper, we analyzed and compared the performance measures of the different machine learning techniques. To claim the validity of our model we experimented the model on four publicly available datasets. The result predicted from Random Forest technique outperformed the other machine learning techniques in most of the cases. Even in using feature selection, result of Random Forest is better than rest of other five machine learning techniques. Thus, the paper confirms that the machine learning techniques such as Decision tree, Random forest, Naïve Bayes, Support vector machine, Artificial Neural Networks and Adaboost, have the predictability for software modules classifying as faulty and nonfaulty modules.

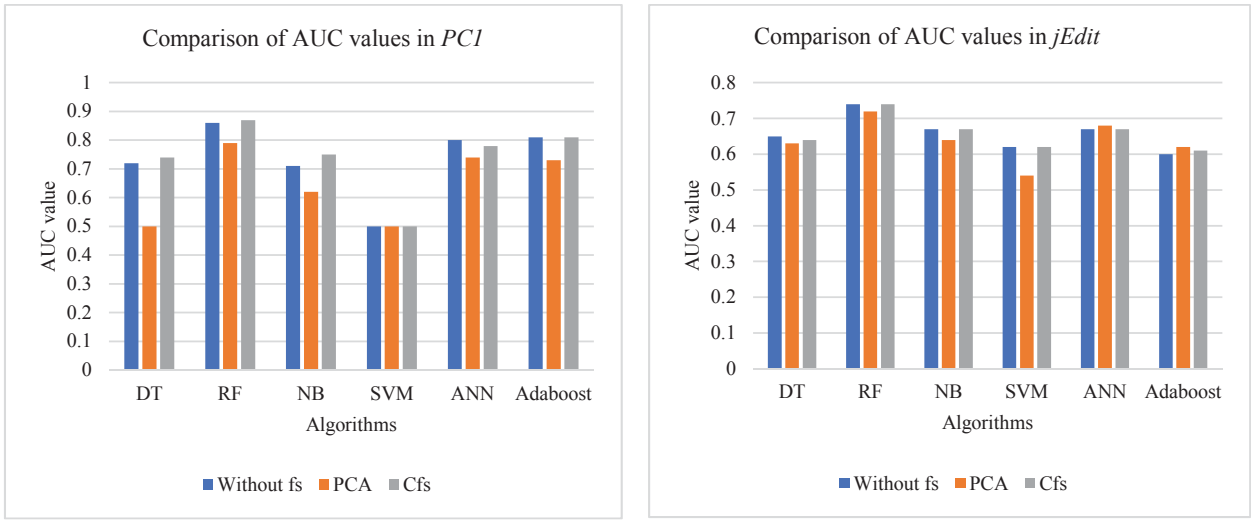


(a)



(b)

Fig. 9 Comparison of AUC values among different algorithms on the datasets (a) JM1 and (b)KC1



(a) *(b)*

Fig. 10 Comparison of AUC values among different algorithms on the datasets (a)PC1 and (b)jEdit

ACKNOWLEDGMENT

The authors would like to thank Indian Council for Cultural Relations (ICCR), Ministry of Foreign Affairs, India for providing funds and DST-CIMS, Institute of Science, Banaras Hindu University, India for providing necessary infrastructure and facilities for undertaking this research work.

REFERENCES

[1] A. Avižienis, J. C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 1, pp. 11–33, 2004, https://doi.org/10.1109/TDSC.2004.2.

[2] I. Portugal, P. Alencar, and D. Cowan, "The use of machine learning algorithms in recommender systems: A systematic review," *Expert Syst. Appl.*, vol. 97, pp. 205–227, 2018, https://doi.org/10.1016/j.eswa.2017.12.020.

[3] S. R. Chidamber and C. F. Kemerer, "a Metrics Suite for Object Oriented Designa Metrics Suite for Object Oriented Design," *PhD Propos.*, vol. 1, no. 6, pp. 476–493, 1992, https://doi.org/10.1017/CBO9781107415324.004.

[4] M. Lorenz and J. Kidd, *Object-oriented software metrics: a practical guide*. Prentice-Hall, Inc., 1994.

[5] T. Love, S. B. Sheppard, P. Milliman, and M. A. Borst, "Measuring the Psychological Complexity of Software Maintenance Tasks with the Halstead and McCabe Metrics," *IEEE Trans. Softw. Eng.*, vol. SE-5, no. 2, pp. 96–104, 1979, https://doi.org/10.1109/TSE.1979.234165.

[6] R. Malhotra and A. Jain, "Fault prediction using statistical and machine learning methods for improving software quality," *J. Inf. Process. Syst.*, vol. 8, no. 2, pp. 241–262, 2012, https://doi.org/10.3745/JIPS.2012.8.2.241.

[7] G. P. Bhandari and R. Gupta, "Fault Analysis of Service-Oriented Systems: A Systematic Literature Review," *IET Softw.*, 2018, https://doi.org/10.1049/iet-sen.2018.5249.

[8] G. Abaei and A. Selamat, "A survey on software fault detection based on different prediction approaches," *Vietnam J. Comput. Sci.*, vol. 1, no. 2, pp. 79–95, 2014, https://doi.org/10.1007/s40595-013-0008-z.

[9] B. Twala, "Predicting software faults in large space systems using machine learning techniques," *Def. Sci. J.*, vol. 61, no. 4, pp. 306–316, 2011, https://doi.org/10.14429/dsj.61.1088.

[10] G. P. Bhandari and R. Gupta, "Extended Fault Taxonomy of SOA-Based Systems," *J. Comput. Inf. Technol.*, vol. 25, no. 4, pp. 237–257, 2018, https://doi.org/10.20532/cit.2017.1003569.

[11] L. Kumar, A. Krishna, and S. K. Rath, "The impact of feature selection on maintainability prediction of service-oriented applications," *Serv. Oriented Comput. Appl.*, vol. 11, no. 2, pp. 137–161, 2017, https://doi.org/10.1007/s11761-016-0202-9.

[12] L. Kumar, S. K. Rath, and A. Sureka, "Using source code metrics to predict change-prone web services: A case-study on ebay services," in *MaLTeSQuE 2017 - IEEE International Workshop on Machine Learning Techniques for Software Quality Evaluation, co-located with SANER 2017*, 2017, pp. 1–7.

[13] J. Sayyad Shirabad and T. J. Menzies, "The PROMISE Repository of Software Engineering Databases," *Sch. Inf. Technol. Eng. Univ. Ottawa, Canada*, vol. 24, 2005.

[14] D. Radjenović, M. Heričko, R. Torkar, and A. Živkovič, "Software fault prediction metrics: A systematic literature review," *Inf. Softw. Technol.*, vol. 55, no. 8, pp. 1397–1418, 2013, https://doi.org/10.1016/j.infsof.2013.02.009.

[15] L. Kumar, S. K. Sripada, A. Sureka, and S. K. Rath, "Effective fault prediction model developed using Least Square Support Vector Machine (LSSVM)," *J. Syst. Softw.*, vol. 137, pp. 686–712, 2018, https://doi.org/10.1016/j.jss.2017.04.016.

[16] R. Malhotra, "Comparative analysis of statistical and machine learning methods for predicting faulty modules," *Appl. Soft Comput. J.*, vol. 21, pp. 286–297, 2014, https://doi.org/10.1016/j.asoc.2014.03.032.

[17] A. Boucher and M. Badri, "Software metrics thresholds calculation techniques to predict fault-proneness: An empirical comparison," *Inf. Softw. Technol.*, vol. 96, no. November 2017, pp. 38–67, 2018, https://doi.org/10.1016/j.infsof.2017.11.005.

[18] J. Moeyersoms, E. Junqué De Fortuny, K. Dejaeger, B. Baesens, and D. Martens, "Comprehensible software fault and effort prediction: A data mining approach," *J. Syst. Softw.*, vol. 100, pp. 80–90, 2015, https://doi.org/10.1016/j.jss.2014.10.032.

[19] A. Malhi and R. X. Gao, "PCA-based feature selection scheme for machine defect classification," *IEEE Trans. Instrum. Meas.*, vol. 53, no. 6, pp. 1517–1525, 2004, https://doi.org/10.1109/TIM.2004.834070.

[20] J. Von Hagen, "Money growth targeting by the bundesbank*," University ofWaikato, 1999.

[21] University of Waikato, "http://www.cs.waikato.ac.nz/ml/weka," *University of Waikato, Hamilton, New Zealand*, 2017. .

[22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software," *ACM SIGKDD Explor. Newsl.*, vol. 11, no. 1, p. 10, 2009, https://doi.org/10.1145/1656274.1656278.