Effect of Feature Selection in Software Fault Detection

Shamse Tasnim Cynthia, Md. Golam Rasul, and Shamim Ripon(✉)

Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh
cnth999@gmail.com, grpranto@gmail.com, dshr@ewubd.edu

Abstract. The quality of software is enormously affected by the faults associated with it. Detection of faults at a proper stage in software development is a challenging task and plays a vital role in the quality of the software. Machine learning is, now a days, a commonly used technique for fault detection and prediction. However, the effectiveness of the fault detection mechanism is impacted by the number of attributes in the publicly available datasets. Feature selection is the process of selecting a subset of all the features that are most influential to the classification and it is a challenging task. This paper thoroughly investigates the effect of various feature selection techniques on software fault classification by using NASA's some benchmark publicly available datasets. Various metrics are used to analyze the performance of the feature selection techniques. The experiment discovers that the most important and relevant features can be selected by the adopted feature selection techniques without sacrificing the performance of fault detection.

Keywords: Fault detection · Feature selection · Feature classification

1 Introduction

Nowadays the role of software has gained much more importance in every known field. This makes the software system more complex than before and some of the software systems have to be delivered with the least or non-negligible number of faults possible. Faults are basically the errors in the code that prevents the software system to work as expected [6]. Faults are often generated for misunderstanding, lacking knowledge in the working area or even for the deadlines. Some of the software faults can be detected at the early stage of developing a software but when fault is detected in the later stage, not only it takes a lot of time to fix the faults but also the quality of software is compromised. Several studies reveal that almost 80% of the software faults occur from 20% of the modules and defect free portion covers the rest half of the module [29]. Software faults demand the rework process that has negative impact on for Software Quality Assurance [10]. Ability in detecting software faults mostly assist software developers in the testing phase about maintaining software standards [8]. So predicting software faults

R. Chamchong and K. W. Wong (Eds.): MIWAI 2019, LNAI 11909, pp. 52–63, 2019.
https://doi.org/10.1007/978-3-030-33709-4_5

at the early stage of software development can support the development of more efficient and reliable software within the stipulated limited time and cost [18].

Feature selection is one of the most significant techniques for kind of any data analysis. Features are mainly referred as the attributes which are given in a dataset and have strong correlation with the class attributes [4]. The main aim of feature selection in a dataset is to select the essential features which will help to improve the fruitfulness of a model. The feature selection techniques not only increases the accuracy and efficiency of a classifier but also decreases the chance of overfitting, reduces the dimensionality and eliminates noise [11,21]. In addition, it can seek out useful information deliberately and lessens the effect of variance in the result. Proper selection of features can help researchers looking for the exact fault in the model. Again when the most essential features are selected, the reduced dimensionality of a dataset boosts the performance of some algorithms, delivers more accurate results in the less amount of time. Most of the feature selection techniques extract features that ranges from sub-optimal to near optimal solutions [26]. By ranking the features with different scores, feature selection techniques reach to near optimal solutions.

Considering the significance of feature selection in predicting and classifying software faults, this work aims at investigating the effect of various feature selection techniques upon the performance of various classification algorithms. In particular, several feature selection techniques are applied to some used fault prediction datasets and then apply classification and predictive techniques on the datasets having only those features selected earlier.

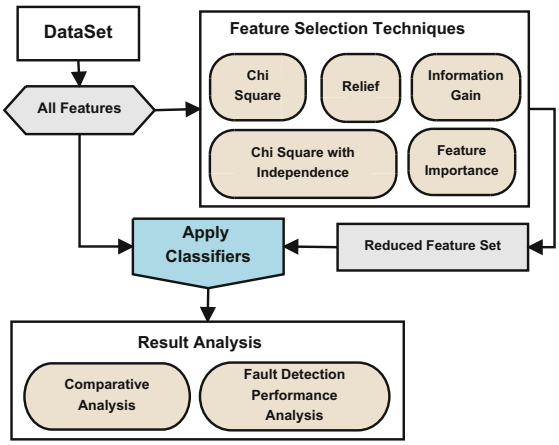

Fig. 1. Proposed model

The schematic view of the proposed framework is illustrated in Fig. 1. Five feature selection techniques along with five datasets are considered in the framework for experimental purpose. For each dataset, all the feature selection techniques are applied and relevant features are selected for each type of technique.

Classification algorithms are then applied to the selected features obtained from each feature selection technique. Several classification algorithms are applied in the experiments. Various metrics are used to analyze the performance of each classification algorithm for each type of feature selection technique and for each type of dataset. For comparative analysis, experiment has also been conducted considering all the features in the dataset and then compare the result with that of the selected features.

The rest of the paper is organized as follows. A brief overview of the dataset used in the paper is shown in Sect. 3. After briefly demonstrated the adopted feature selection techniques in Sect. 4, the following section shows the effect of feature selection on the applied classification algorithms for various datasets. A thorough analysis of the obtained result is presented in Sect. 6. A brief review of similar works are described in Sect. 2. Finally, Sect. 7 concludes the paper by summarizing the paper and outlining our future plan.

2 Related Work

A framework model has been proposed by Oinbao Song et al. [25] to follow-up the MGF on defect prediction using Scheme evaluation and defect prediction for feature selection. Naive Base, J4.8 and OneR were used for comparing the performance. Jiang et al. [13] used ROCUS for software defect prediction. They proposed a disagreement-based semi-supervised learning method to exploit the abundant unlabeled data but higher misclassification rate is the limitation for this technique.

Kakkar et al. [14] tried to build a framework by selecting important attributes using five classifiers: IBK, KStar, LWL, Random Tree and Random Forest. The values of accuracy and ROC was used in evaluating the performance of these classifiers. Attributes selection was done through CfxSubsetEval evaluator where ChiSquaredAttributeEval and CorrelationAttributeEval ranked the attributes based on their individual evaluation. A hybrid feature selection approach has been introduced by Jia [12] where different feature selection techniques have been combined. Those techniques are Chi Square, Information Gain and Pearson Correlation Coefficient techniques. Finally, random forest classifier has been used to build the model.

To measure the correlation among the attributes Qiao et al. [30] introduced a feature selection approach on the basis of similarity measurement for software defect prediction. By updating the feature weights and by sorting them according to their rankings feature list is created in descending order. Finally, K-nearest model classifier is applied on the selected dataset for the detection of faults. Xu et al. [28] proposed a feature selection framework named MICHAC which stands for Maximal Information Coefficient with Hierarchical Agglomerative Clustering. To remove irrelevant features, this framework extracts one feature from each feature subset groups. Three different classifiers and four performance metrics were used to evaluate the performance of the model built with selected featured datasets.

Ibrahim et al. [10] in their work, used Bat-Based Search Algorithm for the feature selection purpose. Similarly, Wahono et al. in their research [26] on software defect prediction gave priority on imbalance nature of the NASA dataset and for feature selection, Genetic Algorithm has been used only. Anbu et al. in their research [2] used Firefly algorithm for feature selection and classifiers like Support Vector Machine, Naïve Bayes and K- nearest neighbor to predict the defects.

In comparison to the existing works, in this paper five feature selection techniques have been applied to NASA MDP's five datasets and five search-based classifiers are applied for analysis of classification performance. Combination of all these three factors results in a large number experiments. Such experiment can give better intuition regarding the effectiveness of the feature selection techniques.

3 DataSet Overview

NASA MDP dataset [26] has been used in this paper. This dataset contains 96 datasets, but among them only 13 datasets are provided by NASA [22]. For experimental purpose we have taken only five of them. These datasets are collected from several projects (satellite instrumentation, ground control systems, attitude control system etc.) in USA for several years. Each dataset is the representative of NASA software system/subsystem containing metrics of static code and fault data for each comprising module. These datasets have been used very widely for detecting software fault. The fault prediction dataset has McCabe, Halstead and line of code metrics [3]. The attributes of these datasets are mostly of numeric types except the class attribute which consists polynomial data. Table 1 shows the total instances of the selected datasets and their class percentages.

Table 1. Dataset Overview

Dataset	Total sample	Defective	Not defective	Number of attributes	Programming language
CM1	344	42 (12.2%)	302 (88%)	37	C
KC3	200	36 (18%)	164 (82%)	39	Java
PC4	1399	178 (13%)	1221 (87%)	37	C
PC2	1585	16 (1%)	1569 (99%)	37	C
MW1	264	27 (10%)	237 (90%)	37	C

4 Feature Selection Techniques

The *Chi-Square* test is introduced by Karl Pearson is a statistical hypothesis test that determines the goodness of fit between a set of observed and expected values [5]. It is a nonparametric test that is used for testing the hypothesis

of no association between two or more groups, population or criteria and to test how well the observed distribution of data fits with the distribution that is expected [23]. The formula of Chi-square is shown in Eq. 1

$$\chi_c^2 = \sum_{1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{1}$$

where, c is degree of freedom, O is observed value and E is expected value

The *Chi-square test of independence* is used to detect if there is a significant relationship between two nominal (categorical) variables. Each category's frequency for one nominal variable is compared with the categories of another nominal variable. Each row of the data in a contingency table represents a category for one variable and each column represents a category for other variable [16]. The corresponding formula is show in Eq. 2,

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{2}$$

where, r is number of rows, c is number of rows, O is observed value and E is expected value.

Information Gain is a measure of the change of entropy which reduces the uncertainty of the result. Entropy gives the measure of impurity of the classes. The value of the entropy should be less for getting the best output. When a node in a decision tree is used for partitioning the training instances into smaller subsets, the value of the entropy changes. Information gain specifies the importance of an attribute and decides the ordering of the attributes in the nodes of a Decision Tree.

$$Gain(T, x) = Entropy(T) - Entropy(T, x) \tag{3}$$

Relief is a feature selection algorithm which uses a statistical method and avoids heuristic research. The algorithm inspired by instance-based learning. It needs linear time for the number of given features and the number of training instances regardless of the target concept to be learned.

From given training data, sample size, and a threshold of relevancy, Relief finds those features that are statistically relevant to the target concept. Relief collects the total number of triplets of an instance, its Near-hit instance and Near-miss instance. Euclidian distance is used for selecting Near-hit and Near-miss. A routine is also called by Relief to update feature weight vector for every triplet and finds the average feature weight vector Relevance (of all the features to the target concept) and those features whose average weight is above the given threshold are selected by Relief [15].

Feature Importance returns a score for each feature and based on that score, the features which have higher score get more privilege towards the output variable. It uses ensembles of decision trees which computes the relative importance of each attribute.

5 Effect of Feature Selection

We have evaluated the performance of our five feature selection processes using the True Positive Rate, True Negative Rate, Precision and Accuracy. These metrics help us to examine whether the methods can correctly and efficiently recognize the optimized features and show us the effects of feature selection in the classification [1]. The experimental result shown here are obtained by considering 20 features selected by applying the feature selection techniques mentioned here.

In the experiment of feature selection techniques with classification algorithms, various matrices are used to measure the performance, namely TPR and TNR and *accuracy*. Four important information is obtained from confusion matrix to calculate these matrices: True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). True positive rate or Sensitivity is the result where the positive class is correctly predicted by the model.

$$TPR = \frac{TP}{TP + FN} \tag{4}$$

Similarly, true negative rate or Specificity is the result where the negative class is correctly predicted by the model.

$$TNR = \frac{TN}{TN + FP} \tag{5}$$

Classification accuracy is the fraction of prediction to see whether the model works right.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

True positive rate, true negative rate and accuracy these three metrics need to be higher for better prediction. We have calculated all these three metrics on the five datasets using some selected classifiers to see their performances. All these experiments have been conducted considering not only the selected feature but also for all the features. While choosing the classifiers, only search-based classifiers as mentioned in [24] are selected for experimental purpose. The classifiers used here are: Decision Tree [20], Random Forest [7], Naïve Bayes [27], Logistic Regression [19] and Artificial Neural Network [17].

Table 2 illustrates True Positive Rate (TPR), True Negative Rate (TNR) and Accuracy of the the classifiers over the datasets after choosing the features by using Relief test. Table 3, on the other hand illustrates the TPR, TNR and Accuracy values of different datasets where features are selected through Chi Square test. Tables 4, 5, and 6 shows the same for Information gain, Chi Square Test of Independence and Feature Importance respectively. For this experiment, 20 relevant features have been selected from each dataset by using the feature selection techniques. We have also conducted a similar experiment by considering 10 most relevant features. The F-measure and precision are also calculated for all the datasets considering all the algorithms and feature selection techniques. The comparative analysis of performance between 10 features and 20 features is illustrated in Sect. 6.

Table 2. Performance with relief test

Dataset	Decision Tree			Random Forest			Naïve Bayes			Logistic Regression			ANN		
	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy
CM1	0.00	1.00	88%	0.38	1.00	92%	0.29	0.90	83%	0.26	0.98	89%	0.10	0.99	88%
KC3	0.56	1.00	83%	0.61	1.00	93%	0.39	0.89	81%	0.36	0.96	86%	0.14	0.98	83%
PC2	0.38	1.00	99%	0.88	1.00	99%	0.31	0.96	95%	0.13	1.00	99%	0.00	1.00	99%
PC4	0.35	0.99	91%	0.23	1.00	91%	0.56	0.86	82%	0.38	0.98	90%	0.48	0.98	92%
MW1	0.00	1.00	90%	0.63	1.00	96%	0.56	0.85	82%	0.40	0.98	91%	0.30	0.98	91%

Table 3. Performance with Chi-Square

Dataset	Decision Tree			Random Forest			Naïve Bayes			Logistic Regression			ANN		
	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy
CM1	0.43	0.99	92%	0.33	1.00	92%	0.29	0.90	83%	0.24	0.97	87%	0.09	0.99	88%
KC3	0.06	1.00	83%	0.67	1.00	94%	0.36	0.91	81%	0.31	0.96	84%	0.11	0.98	83%
PC2	0.25	1.00	99%	0.88	1.00	99%	0.19	0.97	96%	0.19	0.99	99%	0.00	1.00	99%
PC4	0.29	0.99	91%	0.25	1.00	90%	0.26	0.94	85%	0.39	0.98	91%	0.41	0.98	91%
MW1	0.11	1.00	91%	0.67	1.00	97%	0.56	0.85	82%	0.33	0.99	92%	0.44	0.99	93%

Table 4. Performance of information gain

Dataset	Decision Tree			Random Forest			Naïve Bayes			Logistic Regression			ANN		
	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy
CM1	0.38	0.99	92%	0.36	1.00	92%	0.36	0.90	84%	0.26	0.99	90%	0.17	0.99	90%
KC3	0.50	0.99	90%	0.64	0.99	93%	0.34	0.91	82%	0.44	0.97	88%	0.31	1.00	87%
PC2	0.25	1.00	99%	0.69	1.00	99%	0.19	0.97	96%	0.19	1.00	99%	0.00	1.00	99%
PC4	0.30	0.99	90%	0.26	1.00	90%	0.54	0.92	88%	0.43	0.99	91%	0.51	0.98	92%
MW1	0.11	1.00	91%	0.59	1.00	96%	0.59	0.86	83%	0.33	0.99	93%	0.44	0.99	94%

Table 5. Performance of Chi Square test of independence

Dataset	Decision Tree			Random Forest			Naïve Bayes			Logistic Regression			ANN		
	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy
CM1	0.43	0.99	92%	0.33	1.00	92%	0.29	0.90	83%	0.24	0.97	89%	0.09	0.99	88%
KC3	0.06	1.00	83%	0.67	1.00	94%	0.36	0.91	81%	0.31	0.96	84%	0.11	0.98	82%
PC2	0.25	1.00	99%	0.88	1.00	99%	0.19	0.97	96%	0.19	0.99	99%	0	1	99%
PC4	0.29	0.99	91%	0.25	1.00	90%	0.26	0.94	85%	0.39	0.98	91%	0.41	0.98	91%
MW1	0.11	1.00	91%	0.63	1.00	96%	0.51	0.99	83%	0.52	0.87	93%	0.44	0.98	93%

Table 6. Performance of feature importance

Dataset	Decision Tree			Random Forest			Naïve			Logistic Regression			ANN		
	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy	TPR	TNR	Accuracy
CM1	0.43	0.99	92%	0.38	1.00	92%	0.38	0.89	83%	0.26	0.98	90%	0.14	0.98	88%
KC3	0.06	1.00	83%	0.67	1.00	94%	0.36	0.91	82%	0.42	0.96	87%	0.22	0.98	85%
PC2	0.31	1.00	99%	0.88	1.00	99%	0.19	0.97	96%	0.25	0.99	99%	0.00	1.00	99%
PC4	0.30	0.99	91%	0.30	1.00	91%	0.30	0.94	85%	0.40	0.98	91%	0.43	0.98	91%
MW1	0.11	1.00	92%	0.67	1.00	97%	0.56	0.86	83%	0.33	0.99	92%	0.41	0.98	93%

6 Result Analysis

The aim of feature selection is to find the features which are more important and relevant to the target class and discard those which are less important or the correlation between them and the target class is not enough to be considered during classification. NSA dataset consisting 13 datasets, each of them has a large numbers of attributes. If it is possible to select only the important features, the computational time can be reduced, the utilization of resources can be improved and the classification efficiency can be increased. In this paper, five (5) feature selection processes are applied to select features from the datasets and only 20 most important features are selected for the classification process.

Our experiment reveals that when Decision Tree classifier is applied on CM1 dataset considering all the feature, DT can only predict N values (not defective class) and failed to predict any Y value (defective class). The Not defective class only has the class precision value. But when the same experiment is conducted by selecting features using Chi-Square Test, DT can then predict both N and Y value. The same experiment is conducted for all the five feature selection techniques for all the classifiers. The experimental result from DT is shown in Table 7. From the table it can be shown that among the five processes Relief could not predict the Defective class like the other classes.

Table 7. Class detection comparison after feature selection

	TP	TN
Chi-Square	300	18
Information gain	299	16
Feature importance	299	18
Chi-Square test of independence	300	18
Relief	302	00

On the other hand, in KC3 dataset, the feature selection process has less effect compared to the result that was calculated considering all the features. The TP and TN values found usually the same or a little different from the main dataset calculation. For the PC2 dataset, the total number of defective class is very less, so the algorithms could not work better in the classification. The Naïve Bayes and Logistic Regression algorithms perform better in detecting defective classes. The datasets with all features and the datasets with the selected features show almost the same result in both cases. In the PC4 dataset, the features selected by Relief process worked better for Naïve Bayes and Decision Tree algorithms and Chi Square test of independence performed better for Random Forest algorithm compared to the result computed when all the features were present. For example, the TP and TN values calculated with all the features are 69 and 1158 respectively, but with 20 selected features via Relief,

the TP and TN values become 100 and 1049 respectively. Lastly, for the MW1 dataset, all the algorithms with all the features and with the selected features performed almost the same.

In Fig. 2 the accuracies of the different dataset are shown for the Decision tree classifier. Each dataset is tested with all the features and also with the selected features. Average values of the accuracies are taken here for comparison due to multidimensional result. Similar results are obtained for other classifiers, however due to limited scope they are omitted from here.

It is mentioned earlier that experiments have been conducted also by selecting ten (10) most relevant features in the same way as of 20 features. Figure 3 illustrates the accuracy comparison between 10 features' average accuracy and 20 features' average accuracy when random forest classifier is applied for the prediction. Here we can see that for dataset MW1, PC4, and PC2, the accuracies in both subsets of the datasets are almost same. Whereas, for KC3 the classifier gives better result 20 features and for CM1 the classifier gives better result for 10 features. For other classifiers, it has been observed that some classifiers worked better for 10 features' and some are better for 20 features' subsets but the differences are negligible.

The experiments conducted so far do not consider cross validation of the datasets. We conduct a similar experiment considering 10-fold cross validation on all the datasets by applying all five feature selection techniques using all the already chosen classifiers. As the result is multi-dimensional it cannot be presented in a single table or diagram. Figure 4 illustrates the accuracies of various classifiers on the five datasets after applying Chi-Square feature selection technique. The result shows that feature selection improves the classification accuracies for most of the datasets. This experiment has also been conducted for all the feature selection techniques to observe their performances. The resultant tables can be found in Appendix for reviewing purpose only.

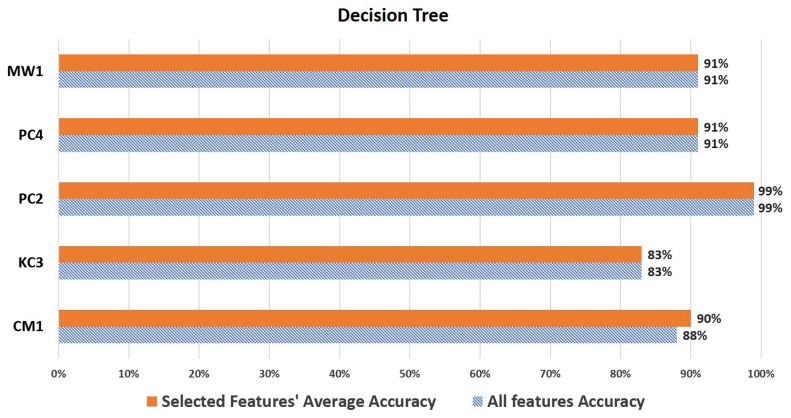

Fig. 2. Accuracy of different datasets before and after applying feature selection

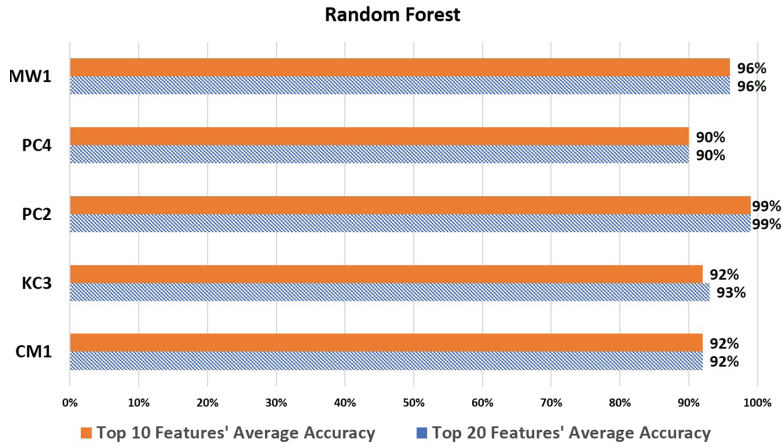

Fig. 3. Classification accuracy of random forest after applying 10 and 20 features in the datasets

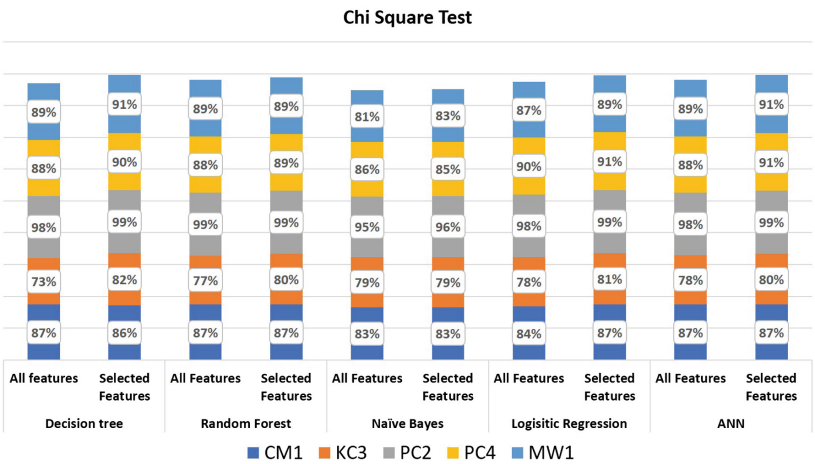

Fig. 4. Comparison of classification accuracies for all classifiers using Chi-square feature selection and 10-fold cross validation

7 Conclusion

Proper selection of relevant features in a large dataset can immensely improve the performance of classifiers and significantly reduces the training time. Among the various feature techniques, this paper shows the effect of feature selection of only five approaches. Five search-based classifiers are applied here for our experiments. The experimental results reveal that after feature selection the performance of the classifiers are almost similar to that of without feature selection. Experiments have been conducted by considering both 10 and 20 features

from the datasets. The variation among the obtained results are not significant. Such result implies that feature selection approaches do not compromise the performance of the classifiers while taking less time and resource during the experiments.

It is mentioned earlier that only a subset of feature selection techniques have been considered in this work. For a better comprehension of the proposed approach, our future plan is to consider both filter and wrapper based feature selection techniques in our experiment. Another major concern for our future work is that NASA MDP datasets that we used in our experiment requires some important preprocessing because these datasets are imbalance. The preprocessing can be done using several methods [9] like eliminating module identifier, extra error data attributes and also by replacing missing data.

References

1. Agarwal, S., Tomar, D.: A feature selection based model for software defect prediction. Int. J. Adv. Sci. Technol. **65**, 39–58 (2014)
2. Anbu, M., Anandha Mala, G.S.: Feature selection using firefly algorithm in software defect prediction. Cluster Comput., 1–10 (2017)
3. Arasteh, B.: Software fault-prediction using combination of neural network and Naive Bayes algorithm. J. Netw. Technol. **9**(3), 94 (2018)
4. Chen, X., Shen, Y., Cui, Z., Ju, X.: Applying feature selection to software defect prediction using multi-objective optimization. In 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), pp. 54–59. IEEE, July 2017
5. Crack, T.F.: A note on Karl Pearson's 1900 Chi-squared test: two derivations of the asymptotic distribution, and uses in goodness of fit and contingency tests of independence, and a comparison with the exact sample variance chi-square result. SSRN Electron. J. (2018)
6. Akalya Devi, C., Surendiran, B., Kannammal, K.E.: A study of feature selection methods for software fault prediction model. In: Proceedings of the International Conference on Network, Intelligence and Computing Technologies (ICNICT 2011), Tamil Nadu, India, pp. 1–5 (2011)
7. Fawagreh, K., Gaber, M.M., Elyan, E.: Random forests: from early developments to recent advancements. Syst. Sci. Control Eng. **2**(1), 602–609 (2014)
8. Felix, E.A., Lee, S.P.: Integrated approach to software defect prediction. IEEE Access **5**, 21524–21547 (2017)
9. Gray, D., Bowes, D., Davey, N., Sun, Y., Christianson, B.: The misuse of the NASA metrics data program data sets for automated software defect prediction. In: 15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011), pp. 96–103. IET (2011)
10. Ibrahim, D.R., Ghnemat, R., Hudaib, A.: Software defect prediction using feature selection and random forest algorithm. In: 2017 International Conference on New Trends in Computing Sciences (ICTCS), pp. 252–257. IEEE, October 2017
11. Jakhar, A.K., Rajnish, K.: Software fault prediction with data mining techniques by using feature selection based models. Int. J. Electr. Eng. Inf. **10**(3), 447–465 (2018)
12. Jia, L.: A hybrid feature selection method for software defect prediction. IOP Conf. Ser. Mater. Sci. Eng. **394**(3), 032035 (2018)

13. Jiang, Y., Li, M., Zhou, Z.-H.: Software defect detection with ROCUS. J. Comput. Sci. Technol. **26**(2), 328–342 (2011)
14. Kakkar, M., Jain, S.: Feature selection in software defect prediction: a comparative study. In 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence), pp. 658–663. IEEE, January 2016
15. Kira, K., Rendell, L.A.: A practical approach to feature selection. In: Proceedings of the Ninth International Workshop on Machine Learning, pp. 249–256 (1992)
16. McHugh, M.L.: The Chi-square test of independence. Biochemia Medica, 143–149 (2013)
17. Mishra, M., Srivastava, M.: A view of artificial neural network. In: 2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014), pp. 1–3. IEEE, August 2014
18. Nugroho, A., Chaudron, M.R.V., Arisholm, E.: Assessing UML design metrics for predicting fault-prone classes in a Java system. In: 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), pp. 21–30. IEEE, May 2010
19. Joanne Peng, C.-Y., Lee, K.L., Ingersoll, G.M.: An introduction to logistic regression analysis and reporting. J. Educ. Res. **96**(1), 3–14 (2002)
20. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
21. Rokach, L.: Ensemble-based classifiers. Artif. Intell. Rev. **33**(1–2), 1–39 (2010)
22. Shepperd, M., Song, Q., Sun, Z., Mair, C.: Data quality: some comments on the NASA software defect data sets. **2010**(9), 1–13 (2013)
23. Singhal, R., Rana, R.: Chi-square test and its application in hypothesis testing. J. Pract. Cardiovasc. Sci. **1**(1), 69 (2015)
24. Son, L.H., et al.: Empirical study of software defect prediction: a systematic mapping. Symmetry **11**(2) (2019)
25. Song, Q., Jia, Z., Shepperd, M., Ying, S., Liu, J.: A general software defect-proneness prediction framework. IEEE Trans. Software Eng. **37**(3), 356–370 (2011)
26. Wahono, R.S., Herman, N.S.: Genetic feature selection for software defect prediction. Adv. Sci. Lett. **20**(1), 239–244 (2014)
27. Webb, G.I., Keogh, E., Miikkulainen, R., Sebag, M.: Naïve Bayes. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning, pp. 713–714. Springer, Boston (2011). https://doi.org/10.1007/978-0-387-30164-8_576
28. Xu, Z., Xuan, J., Liu, J., Cui, X.: MICHAC: defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering. In: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), pp. 370–381. IEEE, March 2016
29. Yousef, A.H.: Extracting software static defect models using data mining. Ain Shams Eng. J. **6**(1), 133–144 (2015)
30. Qiao, Y., Jiang, S., Wang, R., Wang, H.: A feature selection approach based on a similarity measure for software defect prediction. Front. Inf. Technol. Electron. Eng. **18**(11), 1744–1753 (2017)