I.J. Modern Education and Computer Science, **2020, 5, 29-40**
Published Online October 2020 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2020.05.03

Software Defect Prediction Using Variant based Ensemble Learning and Feature Selection Techniques

Umair Ali, Shabib Aftab, Ahmed Iqbal, Zahid Nawaz, Muhammad Salman Bashir, Muhammad Anwaar Saeed
Department of Computer Science, Virtual University of Pakistan, Lahore, Pakistan
Email: umair.ali.hamid@gmail.com, shabib.aftab@gmail.com, ahmedeqbal@gmail.com, mss.zahidnawaz@gmail.com, salman.vu@gmail.com, anwaar@vu.edu.pk

Received: 02 June 2020; Accepted: 28 June 2020; Published: 08 October 2020

Abstract: Testing is considered as one of the expensive activities in software development process. Fixing the defects during testing process can increase the cost as well as the completion time of the project. Cost of testing process can be reduced by identifying the defective modules during the development (before testing) stage. This process is known as "Software Defect Prediction", which has been widely focused by many researchers in the last two decades. This research proposes a classification framework for the prediction of defective modules using variant based ensemble learning and feature selection techniques. Variant selection activity identifies the best optimized versions of classification techniques so that their ensemble can achieve high performance whereas feature selection is performed to get rid of such features which do not participate in classification and become the cause of lower performance. The proposed framework is implemented on four cleaned NASA datasets from MDP repository and evaluated by using three performance measures, including: F-measure, Accuracy, and MCC. According to results, the proposed framework outperformed 10 widely used supervised classification techniques, including: "Naïve Bayes (NB), Multi-Layer Perceptron (MLP), Radial Basis Function (RBF), Support Vector Machine (SVM), K Nearest Neighbor (KNN), kStar (K*), One Rule (OneR), PART, Decision Tree (DT), and Random Forest (RF)".

Index Terms: Software Defect Prediction, Feature Selection, Classifier Variant, Ensemble Learning. Machine Learning Techniques

1. Introduction

Software testing is an important activity of quality assurance process, which ensures the delivery of qualitative product without any defects. As compared to other activities of software development life cycle, testing is the most expensive one as it consumes more resources than others [8,9,10]. This activity makes sure that all of the developed modules are bug free [6,7]. To bring down the overall development cost by keeping the quality intact, is a major issue for software development industry. The prediction of defective modules on the basis of historical development data can resolve this issue. In this procedure, the software modules which are more likely to be defective are identified before the testing activity. Due to which only those software modules are tested which are predicted as defective instead of all. With this approach, the cost of testing activity can be significantly reduced by keeping the quality intact [10,11,12]. The prediction of software defects is a binary classification problem as we have to identify that the particular module is defective or non-defective. Many researchers have focused on machine learning techniques to solve the binary classification problems such as: Network Intrusion Detection [21,22], Sentiment Analysis [13,14,15,16,17,18], Rainfall Prediction [19,20], and Software Defect Prediction [1,2,3,4,5]. The process of software defect prediction has been focused by many researchers in the last decade however improving the prediction accuracy has always been the main concern. This research proposes a classification framework for software defect prediction using variant based ensemble learning and feature selection techniques. There are five stages of the proposed framework: 1) Dataset Selection, 2) Variant Selection, 3) Pre-processing & Feature Selection, 4) Classification, and 5) Reflection of Results. The proposed framework is implemented on four of the cleaned datasets from NASA MDP repository including: JM1, KC1, PC4 and PC5. Moreover, three accuracy measure including: F measure, Accuracy and MCC are used for performance analysis. The performance of the proposed framework is compared with various supervised classifiers from a published research [1], which have used the same datasets and accuracy measures for performance analysis. The classifiers include: "Naïve Bayes (NB), Multi-Layer Perceptron (MLP), Radial Basis Function (RBF), Support Vector Machine (SVM), K

Nearest Neighbor (KNN), kStar (K*), One Rule (OneR), PART, Decision Tree (DT), and Random Forest (RF)". Results show that the proposed framework performed better than all of other classifiers from [1] in all of the three accuracy measures.

2. Related Work

Many researchers have focused on the use of machine learning techniques in order to predict the software defects before the testing stage by using the historical data (data of previously developed modules). The historical data consists of various software metrics collected during the development. This section discusses some recent studies, conducted on software defect prediction. All of the researchers which are going to be discussed in this section used cleaned version of datasets (D'') from NASA MDP repository. Researchers in [1] performed a detailed performance analysis of various supervised classification techniques on software defect prediction. The techniques include: "Naïve Bayes (NB), Multi-Layer Perceptron (MLP). Radial Basis Function (RBF), Support Vector Machine (SVM), K Nearest Neighbor (KNN), kStar (K*), One Rule (OneR), PART, Decision Tree (DT), and Random Forest (RF)". The default parameters of the classification techniques are used and performance was evaluated by using six measures including: "Precision, Recall, F-Measure, Accuracy, MCC, and ROC Area". The authors presented the results as a baseline for other studies so that any improvement in the accuracy by any proposed technique can be verified by comparing with the widely used base classifiers. Researchers in [2] presented a framework for software defect prediction. Feature selection and ensemble learning techniques are used to improve the performance of prediction. Two different dimensions are implemented in the framework, in one dimension feature selection is used and in second dimension feature selection activity is not performed. This process is done to identify the effect of feature selection process on the performance. Performance evaluation is performed through six measures such as: "Precision, Recall, F-Measure, Accuracy, MCC, and ROC Area". The results obtained from both of the dimensions are compared with each other. The results are also compared with other well-known and widely used classification techniques from [1] which have used the same datasets and performance measures. Researchers in [3] discussed the issue of imbalanced datasets and used three well known and widely used resampling techniques to resolve this problem during the process of software defect prediction. The used techniques include: "Random Under Sampling", "Random Over Sampling" and "Synthetic Minority Oversampling Technique (SMOTE)". For classification, various widely used machine learning techniques are used and effects of the resampling techniques on the performance is analyzed by using four measures such as: "F-measure, Accuracy, MCC and ROC". Researchers in [4] presented a framework to predict the defects at early stages of software development. To increase the performance of prediction, the researchers incorporated feature selection and ensemble learning techniques and for performance evaluation four measures are used: F-measure, Accuracy, MCC and ROC. Various feature selection techniques are used in the experiment. In the results, all feature selections techniques are compared with each other in all of the used performance measures. Then a detailed comparison is performed with various widely used supervised classification techniques from research [1]. Researchers in [5] proposed a classification framework to predict the defective modules before the testing. They used a multi filter feature section technique with an aggregated method and Artificial Neural Network (MLP). The framework is implemented with two different dimensions: first with oversampling technique and second without oversampling technique. The purpose behind introducing the oversampling technique in the framework is to analyze the effect of resampling on the performance of proposed classification framework. Performance evaluation is performed by using four accuracy measures: F-measure, Accuracy, MCC and ROC.

3. Materials and Methods

This paper contributes by proposing a classification framework for software defect prediction using variant based ensemble learning and feature selection techniques. The proposed framework (Figure. 1) consists of five stages: 1) Dataset Selection, 2) Variants Selection, 3) Data Preprocessing & Feature selection, 4) Classification, 5) Reflection of results. WEKA tool is used for the implementation of proposed classification framework.

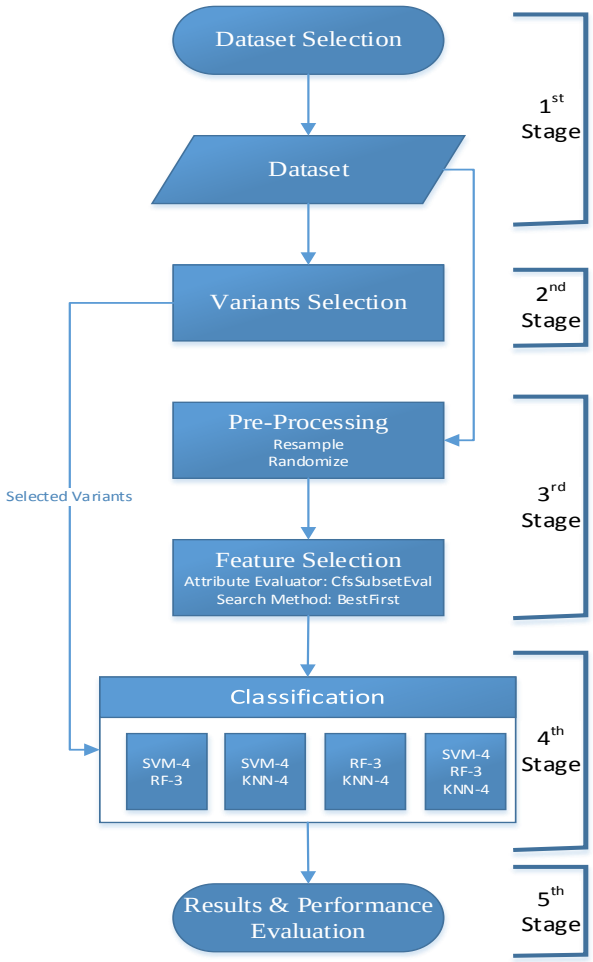

Fig.1. Proposed Classification Framework

Selection of appropriate dataset is the first stage of the proposed framework. In this study we have selected four widely used cleaned datasets from NASA MDP repository including: JM1, KC1, PC4 and PC5 (Table 1). These datasets reflect the module wise metrics of software systems of NASA (Table 2) and their testing results that either these modules are defective or not (Fig. 2,). Each dataset consists of various records whereas each record represents a module in the form of an attribute set. The attributes are the software metrics which are generated during the development (Table 2).

Table 1. NASA Cleaned Datasets (D") [23]

Dataset	Attributes	Modules	Defective	Non-Defective	Defective (%)
JM1	22	7,720	1,612	6,108	20.8
KC1	22	1,162	294	868	25.3
PC4	38	1,270	176	1094	13.8
PC5	39	1694	458	1236	27.0

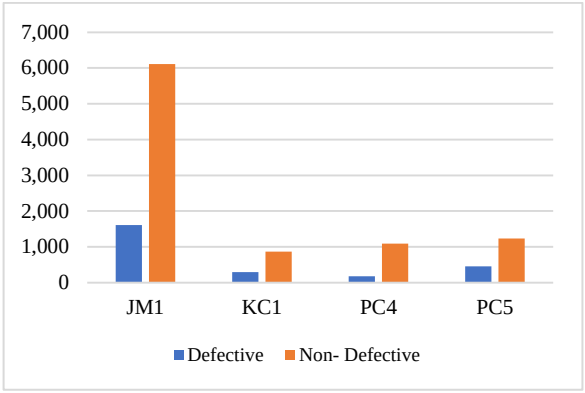

Fig.2. Target Class Distribution

Each of the used datasets contains various independent attributes, and only one dependent attribute. The dependent attribute is the one which is going to be predicted, also known as target class and the independent attributes are those which are used to predict the dependent attribute. The independent attributes of the used datasets are mentioned in Table 2. The target class (dependent attribute) can contain only one from either of two values: 'Y' or 'N'. 'Y' shows that the module is defective and 'N' reflects that the module is non-defective (Fig 2). Researchers in [23], provided two versions of cleaned NASA datasets: DS' and DS''. The instances in DS' included duplicate and inconsistent values however DS'' contains non-duplicate and consistent instances. These datasets are currently available at [24]. In this research we have used the DS'' version of NASA datasets which is already been used by many researchers [1,2,3,4,5,25,26,27].

Table 2. Independent Attributes of Datasets

Sr. #	Attributes	JM1	KC1	PC4	PC5
1	LOC_BLANK	✓	✓	✓	✓
2	BRANCH_COUNT	✓	✓	✓	✓
3	CALL_PAIRS			✓	✓
4	LOC_CODE_AND_COMMENT	✓	✓	✓	✓
5	LOC_COMMENTS	✓	✓	✓	✓
6	CONDITION_COUNT			✓	✓
7	CYCLOMATIC_COMPLEXITY	✓	✓	✓	✓
8	CYCLOMATIC_DENSITY			✓	✓
9	DECISION_COUNT			✓	✓
10	DECISION_DENSITY			✓	
11	DESIGN_COMPLEXITY	✓	✓	✓	✓
12	DESIGN_DENSITY			✓	✓
13	EDGE_COUNT			✓	✓
14	ESSENTIAL_COMPLEXITY	✓	✓	✓	✓
15	ESSENTIAL_DENSITY			✓	✓
16	LOC_EXECUTABLE	✓	✓	✓	✓
17	PARAMETER_COUNT			✓	✓
18	GLOBAL_DATA_COMPLEXITY				✓
19	GLOBAL_DATA_DENSITY				✓
20	HALSTEAD_CONTENT	✓	✓	✓	✓
21	HALSTEAD_DIFFICULTY	✓	✓	✓	✓
22	HALSTEAD_EFFORT	✓	✓	✓	✓
23	HALSTEAD_ERROR_EST	✓	✓	✓	✓
24	HALSTEAD_LENGTH	✓	✓	✓	✓

25	HALSTEAD_LEVEL	✓	✓	✓	✓
26	HALSTEAD_PROG_TIME	✓	✓	✓	✓
27	HALSTEAD_VOLUME	✓	✓	✓	✓
28	MAINTENANCE_SEVERITY			✓	✓
29	MODIFIED_CONDITION_COUNT			✓	✓
30	MULTIPLE_CONDITION_COUNT			✓	✓
31	NODE_COUNT			✓	✓
32	NORMALIZED_CYLOMATIC_COMPLEXITY			✓	✓
33	NUM_OPERANDS	✓	✓	✓	✓
34	NUM_OPERATORS	✓	✓	✓	✓
35	NUM_UNIQUE_OPERANDS	✓	✓	✓	✓
36	NUM_UNIQUE_OPERATORS	✓	✓	✓	✓
37	NUMBER_OF_LINES			✓	✓
38	PERCENT_COMMENTS			✓	✓
39	LOC_TOTAL	✓	✓	✓	✓

Second stage of the framework deals with the selection of best variants from different classifiers (Fig.3). Six classifiers are optimized (tuned) to create the variants. The classifiers include: Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbor (KNN), Naive Bayes (NB), Random Forest (RF), and Multi-layer Perceptron (MLP). First, these classifiers are applied on all the datasets to predict the defective modules with default parameters (without tuning), and then different variants from each of these classifiers are created by optimizing their parameters. All of these variants are then used to predict the defective modules and those variants are selected for ensembles which have higher accuracy on all of the datasets. Only one variant is selected from one family (family included base classifier and its variants), suppose if four different variants of MLP are created then only one variant would be selected which performs higher than its own base classifier (with default parameters). If more than one variants perform better than base classifier then the one with the highest performance will be selected. Multiple variants are created of each base classifier. The first variant of each classifier is the base classifier itself with its default parameters. The accuracy of each of the later variants is compared with the first variant (base classifier) as well as with other variants within the family on each of the dataset. If any of the variant within the family cannot perform well than base classifier then no variant will be selected from that family.

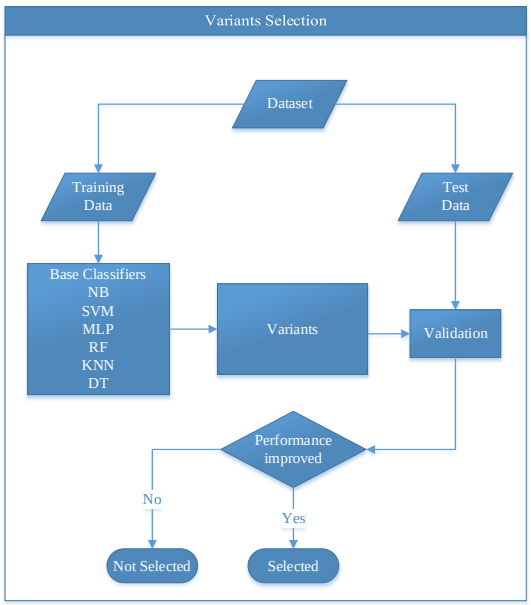

Fig.3. Variant Selection Process

Two variants of Naive Bayes (NB) are created (Table 3) by optimizing two parameters ("UseKernalEstimator" and "Use SupervisedDiscretization"). In first variant the default parameters are used, in which both the parameters are not selected whereas in later variants both the parameters are selected one by one.

Table 3. Variants of Naive Bayes

Optimized parameter	NB-1 (Default)	NB-2	NB-3
UseKernalEstimator / UseSupervisedDiscretization	None-selected	UseKernal Estimator	UseSupervisedDiscretization

For Support Vector Machine (SVM), three variants are created (Table 4). The value of complexity parameter "C" is considered for optimizing, which defines hyper-plan and controls the misclassification. In the case of small value of C, a hyper-plan with large margin is defined which results in high misclassification rate in training data. If the value of C is large then it defines a hyper-plan with small margin which gives better results and classify maximum points correctly.

Table 4. Variants of Support Vector Machine

Optimized parameter	SVM-1 (Default)	SVM-2	SVM-3	SVM-4
C	1	10	25	50

In Multi-layer Perceptron (MLP), the parameter "h" is optimized, which defines the two things 1) no of hidden layers, and 2) no of neurons in each hidden layer. The default value is "a" which defines one hidden layer, and the no of neurons in the hidden layer are decided by the following formula:

$$a = \frac{(attributes + Classes)}{2}$$

Only one hidden layer is used in all variants of MLP with different no of neurons as shown in Table 5.

Table 5. Variants of Multi-Layer Perceptron

Optimized parameter	MLP-1 (Default)	MLP-2	MLP-3	MLP-4	MLP-5
h	a	3	5	7	9

In Random Forest (RF), the parameter of "max depth" is optimized. By default RF has value 0 however two variants are created by changing its number (Table 6).

Table 6. Variants of Random forest

Optimized parameter	RF-1 (Default)	RF-2	RF-3
MaxDepth	0	10	15

During the optimization of K-Nearest Neighbor, three variants are created by optimizing the number of neighbors (Table 7).

Table 7. Variants of K-nearest Neighbours

Optimized parameter	KNN-1 (Default)	KNN-2	KNN-3	KNN-4
No of Neighbours	1	3	5	7

From Decision Tree (DT), 4 variants are created by tuning the parameter of confidence factor which controls the pruning (Table 8).

Table 8. Variants of Decision Tree

Optimized parameter	DT-1 (Default)	DT-2	DT-3	DT-4	DT-5
Confidence Factor	0.25	0.20	0.15	0.10	0.05

All variants including the base classifiers (with default parameters) are used to classify the datasets with training: testing split ratio of 70:30. After the performance analysis on accuracy measure only three variants are selected: SVM-4, RF-3 and KNN-4, as these variants showed higher performance than the base classifiers as well as than other variants within their family. The detailed results (accuracy of the selected variants) will be discussed in next section.

Third stage of proposed framework deals with two activities Data preprocessing and Feature Selection. In Data preprocessing, two tasks are performed: Resampling [31, 32] and Randomization. Resampling is performed to resolve the issue of class imbalance in the datasets as this issue can compromise the accuracy of proposed classification framework [2,3,4,5]. To perform this task, the builtin function of WEKA is used (weka.filters.supervised.instance.Resample). On the other hand, the randomization technique shuffles the instances of datasets. This process is also performed by using a builtin function of WEKA tool (weka.filters.unsupervised.instance.Randomize).

Feature Selection activity is applied on all the datasets in order to select only those features which highly participate in the classification process. As, it has been proved now from the studies [2], [5] that those features should be removed from the dataset which do not participate in classification process as these features may reduce the performance. In this research, feature selection is performed by "Cfs Subset Evaluator" [28, 29, 30] with BestFirst search method, whereas full dataset is given for training. In this approach, three directions are used for feature subset selection: Forward, Backward, and Bi-Directional. In result, three subsets are generated for each dataset. To choose the best subset, those attributes are selected which are common in each direction as shown in Table 9.

Table 9. Feature Subsets

BestFirst (Direction)	JM1	KC1	PC4	PC5
Forward	1,3,4,6,7,8,9,11,20,21	1,3,8,9,10,14,19,20	4,8,17,26	3,4,10,13,17,19,20,21,35
Backward	1,3,4,6,7,8,9,16,20,21	1,3,8,9,10,14,19,20	4,8,17,26	3,4,13,17,19,20,25,35
Bi-Directional	1,3,4,6,7,8,9,16,20,21	1,3,8,9,10,14,19,20	4,8,17,26	3,4,13,17,19,20,21,35
Common Features	1,3,4,6,7,8,9,20,21	1,3,8,9,10,14,19,20	4,8,17,26	3,4,13,17,19,20,35

Classification is the fourth stage of the proposed framework. In this stage the ensembles are created by using three variants which are selected in second stage of framework. Purpose of ensemble learning is to combine the power of multiple classifiers in one classification model. Many ensemble techniques are available today such as Bagging, Boosting, Voting and Stacking etc. In this research, voting is selected to create ensembles. In this technique multiple sub models are created and each models performs its own prediction and then those predictions are combined on the basis of voting. Three variants selected in second stage including: SVM-4, RF-3 and KNN-4 are used to create various ensembles by using Voting technique. Ensemble with all possible combinations of selected variants are used one by one for classification using the subsets selected in third stage, and finally one ensemble having the combination of RF-3 and KNN-4 is chosen as it performed highest among all.

4. Results and Discussion

Fifth and the last stage of the proposed framework deals with the presentation and analysis of results. Three measures are used for the performance evaluation of proposed framework including: F-measure, Accuracy, and MCC. All of these performance measures are based on confusion matrix (Fig. 4)

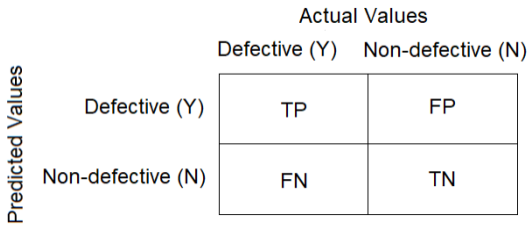

Fig. 4. Confusion Matrix

A confusion matrix is consists of four parameters: TP, FP, FN, and TN. These parameters are explained below:
TP (True Positive): "Instances which are actually positive and also classified as positive".
FP (False Positive): "Instances which are actually negative but classified as positive".
FN (False Negative): "Instances which are actually positive but classified as negative".
TN (True Negative): "Instances which are actually negative and also classified as negative".

All of three performance measures: F-measure, Accuracy, and MCC are calculated by using the parameters of confusion matrix. The brief description along with the calculation formula of each of the used performance measure is given below:

'F-measure' is one of the widely used measures to analyze the performance of classification techniques. To calculate this measure, two measures named 'Precision' and 'Recall' have to be calculated as F-measure is the average of these both measures.

Precision represents the ratio between True Positive instances and the instances which are classified as positive (True Positive + False Positive) as shown below:

$$\text{Precision} = \frac{TP}{(TP + FP)} \tag{1}$$

Recall represents the ratio between True Positive instances and the instances which are actually positive (True Positive + False Negatives) as shown below:

$$\text{Re}\,call = \frac{TP}{(TP + FN)} \tag{2}$$

And finally F-measure is the average of Precision and Recall, as shown below"

$$\text{F-measure} = \frac{\text{Precision} * \text{Recall} * 2}{(\text{Precision} + \text{Recall})} \tag{3}$$

Accuracy represents the ratio of the instances which are classified correctly and the total number of classified instances, as shown below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

MCC represents the ratio of the observed classifications and the predicted classifications.

$$MCC = \frac{TN * TP - FN * FP}{\sqrt{(FP + TP)(FN + TP)(TN + FP)(TN + FN)}} \tag{5}$$

First the results of the variants from stage two are discussed. Multiple variants are created in each of the classifier family (base classifier family) and then those variants are used to predict the defect prone software modules by using all of the used datasets. Three variants from 6 families of the classifiers are selected as only those three performed with highest accuracies by outperforming the base classifiers of their families. The performance of selected variants including SVM-4, RF-3, and KNN-4 are reflected in the tables below (Table 10).

Table 10. Accuracy of Selected Variants

Dataset	JM1		KC1		PC4		PC5	
	Default Classifier	Variant	Default Classifier	Variant	Default Classifier	Variant	Default Classifier	Variant
SVM-4	79.1883	79.4041	75.3582	76.7908	88.189	90.5512	74.2126	74.8031
RF-3	80.1813	80.6131	77.937	79.0831	87.4016	87.4016	75.9843	76.7717
KNN-4	73.9637	80.0086	69.341	77.9370	85.8268	85.8268	73.0315	76.5748

The final results of the proposed framework in terms of F-Measure, Accuracy and MCC are reflected and compared (Table 11 to Table 14) with the results of a published paper [1]. That paper [1] has used 10 widely used supervised classification models on the same version of cleaned NASA MDP datasets, which are used for experiments in this research. The classifiers used in the published papers include: "Naïve Bayes (NB), Multi-Layer Perceptron (MLP). Radial Basis Function (RBF), Support Vector Machine (SVM), K Nearest Neighbor (KNN), kStar (K*), One Rule (OneR), PART, Decision Tree (DT), and Random Forest (RF)".

In this research, the performance measures are only discussed for the defective class ('Y') as we are predicting the defective modules not the non-defective modules. Results show that the proposed framework outperformed all 10 classifiers from published paper in all three performance measures on all of the used datasets. The results from published paper show the symbol '?' in few places which indicates that the score in that measure cannot be calculated due to the issue of imbalance data. The proposed framework also solved this issue besides the overall high performance.

Table 11. JM1 Results

Classifier	F-Measure	Accuracy	MCC
NB	0.318	79.835	0.251
RBF	0.181	80.397	0.215
SVM	?	79.188	?
kNN	0.348	73.963	0.186
kStar	0.355	75.993	0.212
OneR	0.216	77.158	0.126
PART	0.037	79.490	0.104
DT	0.348	79.101	0.252
RF	0.284	80.181	0.244
MLP	0.146	80.354	0.206
Proposed Framework	**0.507**	**84.974**	**0.488**

The results of JM1 dataset are shown in Table 11. It can be seen that the proposed framework outperformed in F-Measure, Accuracy, and in MCC with the scores of 0.507, 84.974, and 0.488 respectively.

Table 12. KC1 Results

Classifier	F-Measure	Accuracy	MCC
NB	0.400	74.212	0.250
RBF	0.362	78.796	0.347
SVM	0.085	75.358	0.151
kNN	0.395	69.341	0.190
kStar	0.419	72.206	0.238
OneR	0.256	73.352	0.147
PART	0.255	76.504	0.239
DT	0.430	75.644	0.291
RF	0.454	77.937	0.346
MLP	0.358	77.363	0.296
Proposed Framework	**0.548**	**83.9542**	**0.482**

KC1 results are reflected in Table 12. It can be seen that the proposed framework shows high performance in all of the three measures with the scores of 0.548, 83.9542 and 0.482 respectively.

Table 13. PC4 Results

Classifier	F-Measure	Accuracy	MCC
NB	0.404	86.089	0.334
RBF	0.250	87.401	0.279
SVM	0.286	88.189	0.342
kNN	0.438	85.826	0.359
kStar	0.330	81.889	0.225
OneR	0.361	87.926	0.352
PART	0.481	85.301	0.396
DT	0.583	86.876	0.514
RF	0.532	90.288	0.516
MLP	0.562	89.763	0.515
Proposed Framework	**0.68**	**91.8635**	**0.649**

Table 13 reflects the results of PC4 dataset. It can be observed that the proposed framework performed better with highest performance in F-Measure, Accuracy, and in MCC with the score of 0.68, 91.8635, and 0.649 respectively.

Table 14. PC5 Results

Classifier	F-Measure	Accuracy	MCC
NB	0.269	75.393	0.245
RBF	0.235	75.590	0.251
SVM	0.097	74.212	0.173
kNN	0.498	73.031	0.314
kStar	0.431	69.881	0.227
OneR	0.387	71.259	0.209
PART	0.335	75.787	0.274
DT	0.531	75.000	0.361
RF	0.450	75.984	0.322
MLP	0.299	74.212	0.216
Proposed Framework	**0.75**	**87.7953**	**0.669**

PC5 results are shown in Table 14. It can be observed that the proposed framework outperformed in F-Measure, Accuracy, and in MCC with the score of 0.75, 87.7953 and 0.669 respectively.

5. Conclusion

In this paper, the researchers proposed a classification framework for the prediction of defect prone software modules in order to reduce the cost of testing process in software development life cycle. The key activities performed by the researchers in order to improve the performance include: feature selection and variant based ensemble classification. Feature selection is performed to eliminate those features which do not participate in the classification process and even reduce the performance of framework besides the high processing cost. In the process of variant selection, first the variants are created by optimizing six base classifiers, including: Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbor (KNN), Naive Bayes (NB), Random Forest (RF), and Multi-layer Perceptron (MLP). From the variants of each base classifier (classifier family), one variant is selected which performed higher than all other variants of its family including the base classifier. Three variants are selected in this research due to high performance including SVM-4, RF-3 and KNN-4. These variants are then integrated by using an ensemble technique: "Voting" with all possible combinations. One combination of variants: RF-3 and KNN-4 outperformed all the other combinations and is selected for classification in the proposed framework. The results of the framework are compared with the results of various widely used supervised classifiers from published paper, which have used the same datasets and performance measures for performance analysis. The comparative analysis have reflected the fact that the proposed framework outperformed all other classification techniques from the published paper and also resolved the issue of class imbalance. However it is suggested for future work to optimize more classifiers with extensive set of parameters so that more variants can be selected for ensemble learning.

References

[1] Iqbal, S. Aftab, U. Ali, Z. Nawaz, L. Sana, M. Ahmad, and A. Husen, "Performance Analysis of Machine Learning Techniques on Software Defect Prediction using NASA Datasets," Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 5, 2019.

[2] A. Iqbal, S. Aftab, I. Ullah, M. S. Bashir, and M. A. Saeed, "A Feature Selection based Ensemble Classification Framework for Software Defect Prediction," Int. J. Mod. Educ. Comput. Sci., vol. 11, no. 9, pp. 54-64, 2019.
[3] A. Iqbal, S. Aftab, and F. Matloob, "Performance Analysis of Resampling Techniques on Class Imbalance Issue in Software Defect Prediction," Int. J. Inf. Technol. Comput. Sci., vol. 11, no. 11, pp. 44-53, 2019
[4] F. Matloob, S. Aftab, and A. Iqbal, "A Framework for Software Defect Prediction Using Feature Selection and Ensemble Learning Techniques," Int. J. Mod. Educ. Comput. Sci., vol. 11, no. 12, pp. 14-20, 2019.
[5] A. Iqbal, and S. Aftab, "A Classification Framework for Software Defect Prediction Using Multi-filter Feature Selection Technique and MLP," Int. J. Mod. Educ. Comput. Sci., vol. 12, no. 1, pp. 18-25, 2020.
[6] A. Dadwal, H. Washizaki, Y. Fukazawa, T. Iida, M. Mizoguchi, and K. Yoshimura, "Prioritization in automotive software testing: Systematic literature review," CEUR Workshop Proc., vol. 2273, no. QuASoQ, pp. 52–58, 2018.
[7] A. Bertolino, "Software testing research: Achievements, challenges, dreams," FoSE 2007 Futur. Softw. Eng., no. September, pp. 85–103, 2007.
[8] R. M. De Castro Andrade, I. De Sousa Santos, V. Lelli, Ḱathia Marçal De Oliveira, and A. R. Rocha, "Software testing process in a test factory from ad hoc activities to an organizational standard," ICEIS 2017 - Proc. 19th Int. Conf. Enterp. Inf. Syst., vol. 2, no. Iceis, pp. 132–143, 2017.
[9] D. Kumar and K. K. Mishra, "The Impacts of Test Automation on Software's Cost, Quality and Time to Market," Procedia Comput. Sci., vol. 79, pp. 8–15, 2016.
[10] S. Huda et al., "A Framework for Software Defect Prediction and Metric Selection," IEEE Access, vol. 6, no. c, pp. 2844–2858, 2017
[11] E. Erturk and E. Akcapinar, "A comparison of some soft computing methods for software fault prediction," Expert Syst. Appl., vol. 42, no. 4, pp. 1872–1879, 2015.
[12] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," Inf. Softw. Technol., vol. 54, no. 3, Mar. 2012.
[13] M. Ahmad, S. Aftab, I. Ali, and N. Hameed, "Hybrid Tools and Techniques for Sentiment Analysis: A Review," Int. J. Multidiscip. Sci. Eng., vol. 8, no. 3, 2017.
[14] M. Ahmad, S. Aftab, S. S. Muhammad, and S. Ahmad, "Machine Learning Techniques for Sentiment Analysis: A Review," Int. J. Multidiscip. Sci. Eng., vol. 8, no. 3, p. 27, 2017.
[15] M. Ahmad and S. Aftab, "Analyzing the Performance of SVM for Polarity Detection with Different Datasets," Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 10, pp. 29–36, 2017.
[16] M. Ahmad, S. Aftab, and I. Ali, "Sentiment Analysis of Tweets using SVM," Int. J. Comput. Appl., vol. 177, no. 5, pp. 25–29, 2017.
[17] M. Ahmad, S. Aftab, M. S. Bashir, and N. Hameed, "Sentiment Analysis using SVM: A Systematic Literature Review," Int. J. Adv. Comput. Sci. Appl., vol. 9, no. 2, 2018.
[18] M. Ahmad, S. Aftab, M. S. Bashir, N. Hameed, I. Ali, and Z. Nawaz, "SVM Optimization for Sentiment Analysis," Int. J. Adv. Comput. Sci. Appl., vol. 9, no. 4, 2018.
[19] S. Aftab, M. Ahmad, N. Hameed, M. S. Bashir, I. Ali, and Z. Nawaz, "Rainfall Prediction in Lahore City using Data Mining Techniques," Int. J. Adv. Comput. Sci. Appl., vol. 9, no. 4, 2018.
[20] S. Aftab, M. Ahmad, N. Hameed, M. S. Bashir, I. Ali, and Z. Nawaz, "Rainfall Prediction using Data Mining Techniques: A Systematic Literature Review," Int. J. Adv. Comput. Sci. Appl., vol. 9, no. 5, 2018.
[21] A. Iqbal and S. Aftab, "A Feed-Forward and Pattern Recognition ANN Model for Network Intrusion Detection," Int. J. Comput. Netw. Inf. Secur., vol. 11, no. 4, pp. 19–25, 2019.
[22] A. Iqbal, S. Aftab, I. Ullah, M. A. Saeed, and A. Husen, "A Classification Framework to Detect DoS Attacks," Int. J. Comput. Netw. Inf. Secur., vol. 11, no. 9, pp. 40-47, 2019.
[23] M. Shepperd, Q. Song, Z. Sun and C. Mair, "Data Quality: Some Comments on the NASA Software Defect Datasets," IEEE Trans. Softw. Eng., vol. 39, pp. 1208–1215, 2013.
[24] "NASA Defect Dataset." [Online]. Available: https://github.com/klainfo/NASADefectDataset. [Accessed: 27-October-2019].
[25] B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," Proc. - Int. Conf. Softw. Eng., vol. 1, pp. 789–800, 2015.
[26] G. Czibula, Z. Marian, and I. G. Czibula, "Software defect prediction using relational association rule mining," Inf. Sci. (Ny)., vol. 264, pp. 260–278, 2014.
[27] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, and J. C. Riquelme, "Preliminary comparison of techniques for dealing with imbalance in software defect prediction," in Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. ACM, p. 43, 2014.
[28] A. O. Balogun, S. Basri, S. J. Abdulkadir, and A. S. Hashim, "Performance Analysis of Feature Selection Methods in Software Defect Prediction: A Search Method Approach," Appl. Sci., vol. 9, no. 13, p. 2764, 2019.
[29] N. Sánchez-Maroño, A. Alonso-Betanzos, and M. Tombilla-Sanromán, "Filter methods for feature selection - A comparative study," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 4881 LNCS, pp. 178–187, 2007.
[30] M. R. Malik, L. Yining, and S. Shaikh, "Analysis of Software Deformity Prone Datasets with Use of Attribute Selected Classifier," Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 7, pp. 14–21, 2019
[31] U. R. Salunkhe and S. N. Mali, "A hybrid approach for class imbalance problem in customer churn prediction: A novel extension to under-sampling," Int. J. Intell. Syst. Appl., vol. 10, no. 5, pp. 71–81, 2018.
[32] N. F. Hordri, S. S. Yuhaniz, N. F. M. Azmi, and S. M. Shamsuddin, "Handling class imbalance in credit card fraud using resampling methods," Int. J. Adv. Comput. Sci. Appl., vol. 9, no. 11, pp. 390–396, 2018.

Authors' Profiles

Umair Ali is student of MS Computer Science with the specialization of Software Engineering in Virtual University of Pakistan. He received the degree, BS Computer Science from Virtual University of Pakistan in 2016. His research interest includes Software Engineering and Data Mining.

Shabib Aftab received MS degree in Computer Science from COMSATS Institute of Information Technology Lahore, Pakistan, and M.Sc degree in Information Technology from Punjab University College of Information Technology (PUCIT) Lahore, Pakistan. Currently he is serving as Lecturer Computer Science at Virtual University of Pakistan. His research areas include Data Mining and Software Process Improvement.

Ahmed Iqbal received MS degree in Computer Science with the specialization of Software Engineering from Virtual University of Pakistan. His research interest includes Software Engineering and Data Mining

Zahid Nawaz received MS degree in Computer Science with the specialization of Software Engineering from Virtual University of Pakistan. Currently he is serving as Head of Computer Science Department at Punjab College (Dunyapur Campus). His research interest includes Software Requirement Engineering, Software Process Models, Agile Process models, Software Design and Software Quality Assurance.

Muhammad Salman Bashir received MS degree in Computer Science from COMSATS Institute of Information Technology Lahore, Pakistan, and M.Sc degree in Computer Science from Punjab University College of Information Technology (PUCIT) Lahore, Pakistan. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Engineering and Technology (UET), Lahore. Currently he is working as Assistant Professor with the Department of Computer Science, Virtual University of Pakistan. His research interests include HCI, Usability Evaluation, Software Processes, and Software Requirements Engineering.

Muhammad Anwaar Saeed obtained his PhD degree in Computer Science from National College of Business Administration & Economics (NCBA&E), Lahore, Pakistan. He is currently working as Assistant Professor with the Department of Computer Science, Virtual University of Pakistan. His area of research is key generation for data encryption and information security. He is also interested in Quantum computing especially encryption mechanisms used in this field. He is also the author of book "Framework for Self Organizing Encryption in Ubiquitous Environment", published by VDM Verlag in 2010. He has published many research papers on his area of interest. Before joining VU, he has ample experience of both software development and network management.

How to cite this paper: Umair Ali, Shabib Aftab, Ahmed Iqbal, Zahid Nawaz, Muhammad Salman Bashir, Muhammad Anwaar Saeed, " Software Defect Prediction Using Variant based Ensemble Learning and Feature Selection Techniques", International Journal of Modern Education and Computer Science(IJMECS), Vol.12, No.5, pp. 29-40, 2020.DOI: 10.5815/ijmecs.2020.05.03

