Performance Analysis of Machine Learning Techniques on Software Defect Prediction using NASA Datasets

Ahmed Iqbal[1], Shabib Aftab[2], Umair Ali[3], Zahid Nawaz[4], Laraib Sana[5], Munir Ahmad[6], Arif Husen[7]
Department of Computer Science, Virtual University of Pakistan, Lahore, Pakistan[1, 2, 3, 4, 6, 7]
Department of Computer Science, Lahore College for Women University, Lahore, Pakistan[5]
Department of Computer Science, COMSATS University Islamabad, Lahore Campus, Pakistan[7]

***Abstract*—Defect prediction at early stages of software development life cycle is a crucial activity of quality assurance process and has been broadly studied in the last two decades. The early prediction of defective modules in developing software can help the development team to utilize the available resources efficiently and effectively to deliver high quality software product in limited time. Until now, many researchers have developed defect prediction models by using machine learning and statistical techniques. Machine learning approach is an effective way to identify the defective modules, which works by extracting the hidden patterns among software attributes. In this study, several machine learning classification techniques are used to predict the software defects in twelve widely used NASA datasets. The classification 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). Performance of used classification techniques is evaluated by using various measures such as: Precision, Recall, F-Measure, Accuracy, MCC, and ROC Area. The detailed results in this research can be used as a baseline for other researches so that any claim regarding the improvement in prediction through any new technique, model or framework can be compared and verified.**

Keywords*—Software defect prediction; software metrics; data mining; machine learning; classification; class imbalance

I. Introduction

Prediction of defective modules in an early stage of software development is considered as one of most challenging aspect of quality assurance activity [11]. The identification of defects in an early stage is crucial as the cost of correcting these defects increases exponentially in the later phases of software development life cycle (SDLC). In software engineering, testing and bug fixing is very expensive and require huge amount of resources [12]. Predicting defective modules in the developing software has been investigated by many studies since the last two decades. An efficient software defect identification technique depends upon various factors, most importantly the extraction of software metrics from historical data. Various software metrics are used to classify the software instance/class/module as defective or non-defective. [13-16]. Furthermore, many empirical studies have also reflected that the subsets of software metrics can improve the performance of classifiers [17]. The activity of software defect prediction is necessary in order to enhance the effectiveness of quality assurance process. It can help to develop a qualitative product with limited amount of resources in a limited time period. Machine learning techniques are considered a promising way to predict the software defects in an early stage of SDLC by detecting the hidden pattern in historical software data. The purpose of this paper is to analyze the performance of supervised machine learning techniques on software defect prediction by using NASA datasets. Machine learning techniques used in this research are: 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). Supervised machine learning techniques need the pre-classified data (training data) for training. During the training process these techniques make rules to classify the unseen data (test data) [18-19], [20-23], [26-27]. In this study, NASA's clean software defect datasets are used for experiments, including: CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4 and PC5. This research performs a detailed performance analysis of widely used machine learning classification techniques by using the 70:30 proportion of training and test data. The benchmark datasets are used in experiments so that any researcher can compare these results with the results of his/her proposed technique and claim the high accuracy which would be easy to validate for all research community.

Further organization of this paper is as follows. Section II discusses the related work. Section III describes about materials and methods used for the experiments. Section IV reflects the results and findings of the experiments. Section V finally concludes this study.

II. Related Work

Many researchers have used machine learning techniques to predict the software defects at an early stage of software development, some of the selected studies are discussed here. Researchers in [1] compared six classification techniques by using the data of 27 academic projects. Classification techniques include: Principal Component Analysis (PCA), Discriminant Analysis, Logistic Regression (LR), Holographic Networks, Logical Classification, and Layered Neural Networks model. Back propagation learning technique was used to develop Neural Network. Performance was evaluated

by using Predictive Validity, Verification Cost, Misclassification Rate, and Achieved Quality. According to results, no model performed well in predicting the software defects. Researchers in [2] used SVM for software defect prediction by using four publicly available NASA datasets: PC1, CM1, KC1 and KC3. The performance is compared with eight machine learning and statistical techniques i.e. K-Nearest Neighbours (KNN), Logistic Regression (LR), Multilayer Perceptron (MLP), Decision Trees, Radial Basis Function (RBF), Bayesian Belief Networks (BBN), Naïve Bayes, and Random Forest (RF). Parameters generated from confusion matrix were used for performance evaluation. The results reflected that SVM performed better than some of the other techniques. Researchers in [3] studied and explored the significant software metrics to predict the software defects. Significant metrics were identified through sensitive analysis by ANN model which was trained using the historical data. The identified metrics then used to develop separate Neural Network models to predict the defective modules. The performance was compared with the Gaussian kernel SVM. JM1 dataset was used for experiment from NASA MDP repository. The results reflected that SVM performed better than ANN in binary defect classification. In [4], researchers performed an experiment using three Cost-Sensitive Boosting algorithms and Back-Propagation learning techniques. From these three, two based on weight updating architectures and one based on threshold value. Four NASA datasets were used for experiment and performance was evaluated using Normalized Expected Cost of Misclassification (NECM). According to results, the threshold based Feed Forward Neural Network performed better than other methods particularly for object oriented software modules. Researchers in [5] compared the statistical and machine learning techniques on software defect prediction by using public domain datasets of AR1 and AR6. The techniques included: Artificial Neural Networks, Decision Trees, Cascade Correlation Network, Support Vector Machines, Group Method of Data Handling Method, and Gene Expression programming. Performance was evaluated by using AUC values. Results reflected that Decision Tree achieved 0.8 and 0.9 AUC scores for AR1 and AR6 respectively which were better than other used techniques. Researchers in [6] presented a software defect prediction technique using Conventional Radial Basis Function along with novel Adaptive Dimensional Biogeography-based optimization model. For experiment, five NASA datasets from PROMISE repository were used and the results showed the higher accuracy of proposed technique as compared to early used techniques. In [7], researchers developed a GUI tool in MATLAB for software defect prediction. The proposed tool was based on Bayesian Regularization (BR) technique which reduced the software cost by limiting the squared errors and weights. The performance of used technique was compared with Levenberg Marquardt (LM) Algorithm and according to results BR performed better. Researchers in [8] compared Artificial Neural Network (ANN) and Support Vector Machine (SVM) on software defect prediction. For experiment, seven NASA datasets from PROMISE repository were used. The performance was evaluated in terms of Specificity, Recall, and Accuracy. Results showed that SVM performed better. In [9], the researchers proposed a GUI tool in MATLAB which used CK (Chidamber and Kemerer) object-oriented metrics for software defect prediction. For experiment, NASA datasets from PROMISE repository were used and performance of Levenberg-Marquardt (LM) algorithm is compared with Polynomial Function-based Neural Network on software defect prediction. According to results the proposed model performed better than other techniques.

III. MATERIALS AND METHODS

This study analyzes the performance of various machine learning classifiers on software defect prediction by using NASA benchmark datasets. Each dataset includes several features along with known output class. The output/target class is one which is predicted on the basis of other available attributes. The attribute which is predicted is known as dependent attribute whereas other attributes which are used to predict the dependent attribute are known as independent attributes. The selected datasets for this study contains dependent attribute which has values either "Y" or "N". "Y" means the specific software instance or module has tendency to be defective and "N" means it is not defective. In this research, total of 12 cleaned NASA datasets [26] are used in experiments. The datasets includes CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4 and PC5 (Table I). Each selected dataset represents a NASA's software system, which includes different metrics, closely related to software quality.

Two versions of clean datasets are provided by [26]: DS' (which included duplicated and inconsistent instances) and DS'' (which does not include duplicated and inconsistent instances). These datasets were initially available at [27] but removed later. We have taken these datasets from [28], where backup of NASA datasets are stored. These cleaned datasets are already used and discussed by [29-31]. Table II reflects the cleaning criteria implemented by [26].

The experiments are performed in Weka [10], one of the most popular data mining tools. This tool is developed in Java language at the University of Waikato, New Zealand and widely accepted due to its portability, General Public License and ease of use.

TABLE I. NASA CLEANED DATASETS [26]

Dataset	Attributes	Modules	Defective	Non-Defective	Defective (%)
CM1	38	327	42	285	12.8
JM1	22	7,720	1,612	6,108	20.8
KC1	22	1,162	294	868	25.3
KC3	40	194	36	158	18.5
MC1	39	1952	36	1916	1.8
MC2	40	124	44	80	35.4
MW1	38	250	25	225	10
PC1	38	679	55	624	8.1
PC2	37	722	16	706	2.2
PC3	38	1,053	130	923	12.3
PC4	38	1,270	176	1094	13.8
PC5	39	1694	458	1236	27.0

TABLE II. CLEANING CRITERIA [26], [29]

Criterion	Data Quality Category	Explanation
1.	Identical cases	'Instances that have identical values for all metrics including class label'.
2.	Inconsistent cases	'Instances that satisfy all conditions of Case 1, but where class labels differ'.
3.	Cases with missing values	'Instances that contain one or more missing observations'.
4.	Cases with conflicting feature values	'Instances that have 2 or more metric values that violate some referential integrity constraint. For example, LOC TOTAL is less than Commented LOC. However, Commented LOC is a subset of LOC TOTAL'.
5.	Cases with implausible values	'Instances that violate some integrity constraint. For example, value of LOC=1.1'

IV. RESULTS AND DISCUSSION

This section aims to analyze the performance of used classification techniques. The performance is analyzed and evaluated through various measures generated from confusion matrix (shown in Fig. 1). A confusion matrix consists of the following parameters:

True Positive (TP): Instances which are actually positive and also classified as positive.

False Positive (FP): Instances which are actually negative but classified as positive.

False Negative (FN): Instances which are actually positive but classified as negative.

True Negative (TN): instances which are actually negative and also classified as negative.

The classification techniques are evaluated through following measures: Precision, Recall, F-measure, Accuracy, MCC and ROC.

Precision is defined as the ratio of True Positive (TP) modules with respect to total number of modules which are classified as positive [2].

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

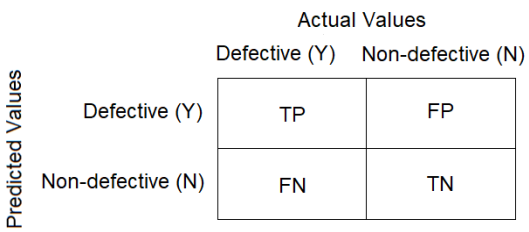

Fig. 1. Confusion Matrix.

Recall is defined as the ratio of True Positive (TP) modules with respect to the total number of modules that are actually positive [2].

$$\text{Recall} = \frac{TP}{(TP + FN)} \tag{2}$$

F-measure provides the average of Precision & Recall [2].

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

Accuracy indicates that how much the prediction is accurate [2], [32].

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

Matthew's Correlation Coefficient (MCC) is defined as a ratio of the observed and predicted binary classifications and ranges from -1 to +1. The results closer to 1 depicts the good prediction whereas closer to or below 0 indicates the bad performance [24], [32].

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

The area under the ROC curve (AUC) is a measure of how well a parameter can distinguish between two classes (defective/non defective) [25], [31].

$$AUC = \frac{1 + TP_r - FP_r}{2} \tag{6}$$

All these performance measures are given by Weka tool. The results of Precision, Recall and F-Measure for each class (Y and N) are reflected in the tables (Table III to Table XIV). These accuracy measures are sensitive to class imbalance problem and reflect the symbol of '?' in case of such issue. Highest scores in each class are highlighted in bold for easy identification.

Results of CM1 datasets are given in Table III. It can be seen that in Precision, NB performed better in both the classes (Y and N). In Recall, NB and DT both performed better in Y class whereas RBF, SVM and PART showed better performance in N class and finally in F-measure, NB showed better performance in Y class whereas RBF, SVM and PART performed better in N class.

Results of JM1 datasets are reflected in Table IV. In precision, PART performed better in Y class whereas kStar performed better in N class. In Recall, kNN performed better in Y class and SVM performed better in N class. In F-measure, kStar outperformed in Y class whereas MLP and RBF outperformed in N class.

Table V reflects the results of KC1 dataset. It can be observed that in precision, SVM performed better in Y class whereas RF performed better in N class. In Recall, kNN and kStar both performed better in Y class and SVM performed better in N class. And finally in F-measure, RF performed better in Y class and RBF outperformed in N class.

TABLE III. CM1 DATASET RESULTS

Classifier	Class	Precision	Recall	F-Measure
NB	Y	**0.167**	**0.222**	**0.190**
	N	**0.919**	0.888	0.903
MLP	Y	0.000	0.000	0.000
	N	0.904	0.955	0.929
RBF	Y	?	0.000	?
	N	0.908	**1.000**	**0.952**
SVM	Y	?	0.000	?
	N	0.908	**1.000**	**0.952**
kNN	Y	0.067	0.111	0.083
	N	0.904	0.843	0.872
kStar	Y	0.067	0.111	0.083
	N	0.904	0.843	0.872
OneR	Y	0.000	0.000	0.000
	N	0.903	0.944	0.923
PART	Y	?	0.000	?
	N	0.908	**1.000**	**0.952**
DT	Y	0.118	**0.222**	0.154
	N	0.914	0.831	0.871
RF	Y	0.000	0.000	0.000
	N	0.907	0.989	0.946

TABLE IV. JM1 DATASET RESULTS

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.537	0.226	0.318
	N	0.823	0.949	0.882
MLP	Y	0.765	0.081	0.146
	N	0.804	0.993	**0.889**
RBF	Y	0.694	0.104	0.181
	N	0.807	0.988	**0.889**
SVM	Y	?	0.000	?
	N	0.792	**1.000**	0.884
kNN	Y	0.363	**0.334**	0.348
	N	0.829	0.846	0.837
kStar	Y	0.403	0.317	**0.355**
	N	**0.830**	0.876	0.853
OneR	Y	0.378	0.151	0.216
	N	0.807	0.935	0.866
PART	Y	**0.818**	0.019	0.037
	N	0.795	0.999	0.885
DT	Y	0.496	0.268	0.348
	N	0.828	0.929	0.876
RF	Y	0.572	0.189	0.284
	N	0.819	0.963	0.885

Results of KC3 dataset is reflected in Table VI. It is reflected that in Precision, MLP and OneR showed highest performance in Y class whereas NB performed better in N class. In Recall, NB and kNN performed better in Y class and in N class, SVM outperformed the others. In F-measure, NB performed better in Y class whereas SVM performed better in N class.

Results of MC1 dataset are reflected in Table VII. In Precision, kNN and PART showed better performance in Y class whereas NB performed better in N class. In Recall, NB performed better in Y class whereas MLP, RBF, SVM and DT performed better in N class. In F-Measure, kNN and PART performed better in Y class whereas MLP, RBF, SVM and DT performed better in N class.

TABLE V. KC1 DATASET RESULTS

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.492	0.337	0.400
	N	0.795	0.881	0.836
MLP	Y	0.647	0.247	0.358
	N	0.787	0.954	0.863
RBF	Y	0.778	0.236	0.362
	N	0.789	0.977	**0.873**
SVM	Y	**0.800**	0.045	0.085
	N	0.753	**0.996**	0.858
kNN	Y	0.398	**0.393**	0.395
	N	0.793	0.796	0.795
kStar	Y	0.449	**0.393**	0.419
	N	0.801	0.835	0.817
OneR	Y	0.444	0.180	0.256
	N	0.767	0.923	0.838
PART	Y	0.667	0.157	0.255
	N	0.771	0.973	0.861
DT	Y	0.533	0.360	0.430
	N	0.803	0.892	0.845
RF	Y	0.615	0.360	**0.454**
	N	**0.808**	0.923	0.862

TABLE VI. KC3 DATASET RESULTS

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.444	**0.400**	**0.421**
	N	**0.878**	0.896	0.887
MLP	Y	**0.500**	0.300	0.375
	N	0.865	0.938	0.900
RBF	Y	0.000	0.000	0.000
	N	0.818	0.938	0.874
SVM	Y	?	0.000	?
	N	0.828	**1.000**	**0.906**
kNN	Y	0.333	**0.400**	0.364
	N	0.870	0.833	0.851
kStar	Y	0.300	0.300	0.300
	N	0.854	0.854	0.854
OneR	Y	**0.500**	0.300	0.375
	N	0.865	0.938	0.900
PART	Y	0.250	0.100	0.143
	N	0.833	0.938	0.882
DT	Y	0.300	0.300	0.300
	N	0.854	0.854	0.854
RF	Y	0.286	0.200	0.235
	N	0.843	0.896	0.869

TABLE VII. MC1 DATASET RESULTS

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.156	**0.357**	0.217
	N	**0.984**	0.953	0.968
MLP	Y	?	0.000	?
	N	0.976	**1.000**	**0.988**
RBF	Y	?	0.000	?
	N	0.976	**1.000**	**0.988**
SVM	Y	?	0.000	?
	N	0.976	**1.000**	**0.988**
kNN	Y	**0.400**	0.286	**0.333**
	N	0.983	0.990	0.986
kStar	Y	0.250	0.143	0.182
	N	0.979	0.990	0.984
OneR	Y	0.333	0.143	0.200
	N	0.979	0.993	0.986
PART	Y	**0.400**	0.286	**0.333**
	N	0.983	0.990	0.986
DT	Y	?	0.000	?
	N	0.976	**1.000**	**0.988**
RF	Y	0.000	0.000	0.000
	N	0.976	0.998	0.987

Results of MC2 dataset are reflected in Table VIII. In precision, NB performed better in Y class whereas PART performed better in N class. In Recall, PART performed better in Y class whereas NB and RBF performed better in N class. In F Measure, PART performed better in both the classes.

TABLE VIII. MC2 DATASET RESULTS

Classifier	Class	Precision	Recall	F-Measure
NB	Y	**0.833**	0.385	0.526
	N	0.742	**0.958**	0.836
MLP	Y	0.500	0.538	0.519
	N	0.739	0.708	0.723
RBF	Y	0.800	0.308	0.444
	N	0.719	**0.958**	0.821
SVM	Y	0.400	0.154	0.222
	N	0.656	0.875	0.750
kNN	Y	0.667	0.462	0.545
	N	0.750	0.875	0.808
kStar	Y	0.400	0.308	0.348
	N	0.667	0.750	0.706
OneR	Y	0.500	0.231	0.316
	N	0.677	0.875	0.764
PART	Y	0.727	**0.615**	**0.667**
	N	**0.808**	0.875	**0.840**
DT	Y	0.500	0.385	0.435
	N	0.704	0.792	0.745
RF	Y	0.500	0.462	0.480
	N	0.720	0.750	0.735

Table IX reflects the result of MW1 dataset. It can be seen that in Precision, MLP performed better in both the classes. In Recall, MLP performed better in Y class whereas OneR performed better in in N class. In F-measure, MLP performed better in both the classes.

TABLE IX. MW1 DATASET RESULTS

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.333	0.625	0.435
	N	0.95	0.851	0.898
MLP	Y	**0.545**	**0.75**	**0.632**
	N	**0.969**	0.925	**0.947**
RBF	Y	?	0.00	?
	N	0.893	1.000	0.944
SVM	Y	?	0.000	?
	N	0.893	1.000	0.944
kNN	Y	0.400	0.500	0.444
	N	0.938	0.910	0.924
kStar	Y	0.143	0.125	0.133
	N	0.897	0.910	0.904
OneR	Y	0.500	0.125	0.200
	N	0.904	**0.985**	0.943
PART	Y	0.250	0.125	0.167
	N	0.901	0.955	0.928
DT	Y	0.250	0.125	0.167
	N	0.901	0.955	0.928
RF	Y	0.333	0.125	0.182
	N	0.903	0.970	0.935

TABLE X. PC1 DATASET RESULTS

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.280	**0.700**	0.400
	N	0.983	0.907	0.944
MLP	Y	**1.000**	0.300	0.462
	N	0.965	**1.000**	**0.982**
RBF	Y	0.333	0.100	0.154
	N	0.955	0.990	0.972
SVM	Y	?	0.000	?
	N	0.951	**1.000**	0.975
kNN	Y	0.273	0.300	0.286
	N	0.964	0.959	0.961
kStar	Y	0.125	0.300	0.176
	N	0.961	0.892	0.925
OneR	Y	0.333	0.100	0.154
	N	0.955	0.990	0.972
PART	Y	0.375	0.600	0.462
	N	0.979	0.948	0.963
DT	Y	0.389	**0.700**	**0.500**
	N	**0.984**	0.943	0.963
RF	Y	0.750	0.300	0.429
	N	0.965	0.995	0.980

Results of PCI datasets are shown in Table X. It can be seen that in Precision, MLP performed better in Y class whereas DT performed better in N class. In Recall, NB and DT performed better in Y class whereas MLP and SVM both performed better in N class. In F-measure, DT performed better in Y class whereas MLP performed better in N class.

Results of PC2 datasets are shown in Table XI. According to results in Precision, kStar performed well in both the classes. In Recall, kStar performed well in Y class whereas RBF, SVM, DT and RF performed well in N class. In F-measure, kStar performed well in Y class however RBF, SVM, DT and RF performed well in N class.

Results of PC3 dataset is reflected in Table XII. It can be seen that in Precision, OneR and RF performed better in Y class however NB performed better in N class. In Recall, NB performed better in Y class whereas RBF, SVM and PART performed better in N class. In F-measure, DT performed better in Y class whereas OneR and RF performed better in N class.

Results of PC4 dataset are shown in Table XIII. It is reflected that in Precision, SVM performed better in Y class whereas DT performed better in N class. In Recall, DT performed better in Y class whereas SVM performed better in N class. In F-Measure, DT performed better in Y class whereas RF performed better in N class.

Results of PC5 dataset are shown in Table XIV. It can be seen that in Precision, SVM performed better in Y class whereas DT performed better in N class. In Recall, DT performed better in Y class whereas SVM performed better in N Class. In F Measure, DT performed better in Y class whereas RBF performed better in N class.

TABLE XI. PC2 DATASET RESULTS

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.000	0.000	0.000
	N	0.976	0.967	0.972
MLP	Y	0.000	0.000	0.000
	N	0.977	0.991	0.984
RBF	Y	?	0.000	?
	N	0.977	**1.000**	**0.988**
SVM	Y	?	0.000	?
	N	0.977	**1.000**	**0.988**
kNN	Y	0.000	0.000	0.000
	N	0.977	0.991	0.984
kStar	Y	**0.143**	**0.200**	**0.167**
	N	**0.981**	0.972	0.976
OneR	Y	0.000	0.000	0.000
	N	0.977	0.995	0.986
PART	Y	0.000	0.000	0.000
	N	0.977	0.991	0.984
DT	Y	?	0.000	?
	N	0.977	**1.000**	**0.988**
RF	Y	?	0.000	?
	N	0.977	**1.000**	**0.988**

TABLE XII. PC3 DATASET RESULTS

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.150	**0.907**	0.257
	N	**0.929**	0.190	0.316
MLP	Y	0.346	0.209	0.261
	N	0.883	0.938	0.909
RBF	Y	?	0.000	?
	N	0.864	**1.000**	0.927
SVM	Y	?	0.000	?
	N	0.864	**1.000**	0.927
kNN	Y	0.480	0.279	0.353
	N	0.893	0.952	0.922
kStar	Y	0.313	0.233	0.267
	N	0.884	0.919	0.901
OneR	Y	**0.600**	0.140	0.226
	N	0.879	0.985	**0.929**
PART	Y	?	0.000	?
	N	0.864	**1.000**	0.927
DT	Y	0.500	0.279	**0.358**
	N	0.894	0.956	0.924
RF	Y	**0.6000**	0.140	0.226
	N	0.879	0.985	**0.929**

TABLE XIII. PC4 DATASET RESULTS

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.486	0.346	0.404
	N	0.901	0.942	0.921
MLP	Y	0.676	0.481	0.562
	N	0.922	0.964	0.942
RBF	Y	0.667	0.154	0.250
	N	0.881	0.988	0.931
SVM	Y	**0.818**	0.173	0.286
	N	0.884	**0.994**	0.936
kNN	Y	0.477	0.404	0.438
	N	0.908	0.930	0.919
kStar	Y	0.333	0.327	0.330
	N	0.894	0.897	0.895
OneR	Y	0.650	0.250	0.361
	N	0.892	0.979	0.933
PART	Y	0.464	0.500	0.481
	N	0.920	0.909	0.914
DT	Y	0.515	**0.673**	**0.583**
	N	**0.946**	0.900	0.922
RF	Y	0.778	0.404	0.532
	N	0.912	0.982	**0.946**

Accuracy results are shown in Table XV. It can be seen that RBF showed better performance with higher accuracy in 5 datasets. MLP, SVM, RF each showed higher accuracy in 4

datasets. On the other hand NB, kStar and kNN did not show higher accuracy in any of the dataset. MCC results are shown in Table XVI, it can be seen that scores in most of the classifiers could not be drawn due to class imbalance. NB and DT each performed better in 3 datasets whereas MLP showed higher performance in 2 datasets. RBF, kNN, kStar, PART and RF each showed higher performance in 1 dataset. SVM and OneR did not perform well in any single dataset. Table XVII shows the results of ROC area, it can be seen that RF reflected high performance in 8 datasets whereas NB, MLP, kStar and PART each showed high performance in 1 dataset. Remaining algorithms did not show high performance in any of the used dataset. It has been noted that Besides the Precision, Recall and F measure, MCC is also sensitive to the class imbalance problem as it could not give the scores in many of the classifiers. However accuracy and ROC both did not show any symbol of class imbalance in the results which make them non sensitive to this issue. Therefore, besides the Accuracy and ROC, other performance measures including Precision, Recall, F-Measure and MCC should be used for effective performance analysis. Class imbalance issue can be resolved in the used datasets with many techniques [31]. However the purpose of this research is to use the exact snapshot of NASA cleaned dataset that is why no preprocessing or class balancing techniques are used.

TABLE XIV. PC5 DATASET RESULTS

Classifier	Class	Precision	Recall	F-Measure
NB	Y	0.676	0.168	0.269
	N	0.759	0.970	0.852
MLP	Y	0.560	0.204	0.299
	N	0.762	0.941	0.842
RBF	Y	0.760	0.139	0.235
	N	0.756	0.984	**0.855**
SVM	Y	**0.875**	0.051	0.097
	N	0.740	**0.997**	0.850
kNN	Y	0.500	0.496	0.498
	N	0.815	0.817	0.816
kStar	Y	0.439	0.423	0.431
	N	0.790	0.801	0.795
OneR	Y	0.455	0.336	0.387
	N	0.776	0.852	0.812
PART	Y	0.646	0.226	0.335
	N	0.770	0.954	0.852
DT	Y	0.537	**0.526**	**0.531**
	N	**0.826**	0.833	0.830
RF	Y	0.588	0.365	0.450
	N	0.794	0.906	0.846

TABLE XV. ACCURACY RESULTS

Dataset	NB	MLP	RBF	SVM	kNN	kStar	OneR	PART	DT	RF
CM1	82.6531	86.7347	**90.8163**	**90.8163**	77.551	77.551	85.7143	**90.8163**	77.551	89.7959
JM1	79.8359	80.3541	**80.3972**	79.1883	73.9637	75.9931	77.1589	79.4905	79.1019	80.1813
KC1	74.212	77.3639	**78.7966**	75.3582	69.341	72.2063	73.3524	76.5043	75.6447	77.937
KC3	81.0345	**82.7586**	77.5862	**82.7586**	75.8621	75.8621	**82.7586**	79.3103	75.8621	77.5862
MC1	93.8567	**97.6109**	**97.6109**	**97.6109**	97.2696	96.9283	97.2696	97.2696	**97.6109**	97.4403
MC2	75.6757	64.8649	72.973	62.1622	72.973	59.4595	64.8649	**78.3784**	64.8649	64.8649
MW1	82.6667	**90.6667**	89.3333	89.3333	86.6667	82.6667	89.3333	86.6667	86.6667	88.000
PC1	89.7059	**96.5686**	94.6078	95.098	92.6471	86.2745	94.6078	93.1373	93.1373	96.0784
PC2	94.47	96.7742	**97.6959**	**97.6959**	96.7742	95.3917	97.235	96.7742	**97.6959**	**97.6959**
PC3	28.7975	83.8608	86.3924	86.39 24	86.0759	82.5949	**87.0253**	86.3924	86.3924	**87.0253**
PC4	86.0892	89.7638	87.4016	88.189	85.8268	81.8898	87.9265	85.3018	86.8766	**90.2887**
PC5	75.3937	74.2126	75.5906	74.2126	73.0315	69.8819	71.2598	75.7874	75.000	**75.9843**

TABLE XVI. MCC RESULTS

Dataset	**NB**	**MLP**	**RBF**	**SVM**	**kNN**	**kStar**	**OneR**	**PART**	**DT**	**RF**
CM1	**0.097**	-0.066	?	?	-0.037	-0.037	-0.074	?	0.041	-0.032
JM1	0.251	0.206	0.215	?	0.186	0.212	0.126	0.104	**0.252**	0.244
KC1	0.250	0.296	**0.347**	0.151	0.190	0.238	0.147	0.239	0.291	0.346
KC3	**0.309**	0.295	-0.107	?	0.218	0.154	0.295	0.056	0.154	0.111
MC1	0.208	?	?	?	**0.325**	0.174	0.206	**0.325**	?	-0.006
MC2	**0.444**	0.243	0.371	0.040	0.374	0.062	0.137	0.512	0.189	0.216
MW1	0.367	**0.589**	?	?	0.373	0.038	0.211	0.110	0.110	0.150
PC1	0.400	**0.538**	0.161	?	0.247	0.128	0.161	0.440	0.490	0.459
PC2	-0.028	-0.015	?	?	-0.015	**0.146**	-0.010	-0.015	?	?
PC3	0.088	0.183	?	?	0.294	0.173	0.245	?	**0.304**	0.245
PC4	0.334	0.515	0.279	0.342	0.359	0.225	0.352	0.396	0.514	**0.516**
PC5	0.245	0.216	0.251	0.173	0.314	0.227	0.209	0.274	**0.361**	0.322

TABLE XVII. ROC AREA RESULTS

Dataset	NB	MLP	RBF	SVM	kNN	kStar	OneR	PART	DT	RF
CM1	0.703	0.634	0.702	0.500	0.477	0.538	0.472	0.610	0.378	**0.761**
JM1	0.663	0.702	0.713	0.500	0.591	0.572	0.543	0.714	0.671	**0.738**
KC1	0.694	0.736	0.713	0.521	0.595	0.651	0.551	0.636	0.606	**0.751**
KC3	0.769	0.733	0.735	0.500	0.617	0.528	0.619	0.788	0.570	**0.807**
MC1	0.826	0.805	0.781	0.500	0.638	0.631	0.568	0.684	0.500	**0.864**
MC2	**0.795**	0.753	0.766	0.514	0.668	0.510	0.553	0.724	0.615	0.646
MW1	0.791	**0.843**	0.808	0.500	0.705	0.543	0.555	0.314	0.314	0.766
PC1	0.879	0.779	0.875	0.500	0.629	0.673	0.545	**0.889**	0.718	0.858
PC2	0.751	0.746	0.724	0.500	0.495	**0.791**	0.498	0.623	0.579	0.731
PC3	0.773	0.796	0.795	0.500	0.616	0.749	0.562	0.79	0.664	**0.855**
PC4	0.807	0.898	0.862	0.583	0.667	0.734	0.614	0.776	0.834	**0.945**
PC5	0.725	0.751	0.732	0.524	0.657	0.629	0.594	0.739	0.703	**0.805**

V. CONCLUSION

Software defect prediction using machine learning techniques is considered as one of the emerging research areas now days. Identification of defects at the early stage of development can contribute to the delivery of high quality software by using limited amount of resources. This study deals with the detailed performance analysis of various machine learning classification techniques on software defect prediction using 12 widely used and publically available NASA datasets. The classification 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 performance is evaluated by using various measures extracted from confusion matrix such as: Precision, Recall, F-Measure, Accuracy, MCC, and ROC Area. It is reflected from the results that neither the Accuracy and nor the ROC can be used as an effective performance measure as both of these did not react on class imbalance issue. However, Precision, Recall, F-Measure and MCC reacted to class imbalance problem in the results with the symbol of '?'. The results presented in this research can be used as baseline for other researches so that the results of any proposed technique, model or framework can be compared and easily verified. For future work, class imbalance issue should be resolved in NASA cleaned datasets. Moreover, to improve the performance, feature selection and ensemble learning techniques should also be explored.

REFERENCES

[1] F. Lanubile, A. Lonigro, and G. Vissagio, "Comparing models for identifying fault-prone software components." Seke, no. July, pp. 312–319, 1995.

[2] K. O. Elish and M. O. Elish, "Predicting defect-prone software modules using support vector machines," J. Syst. Softw., vol. 81, no. 5, pp. 649–660, 2008.

[3] I. Gondra, "Applying machine learning to software fault-proneness prediction," J. Syst. Softw., vol. 81, no. 2, pp. 186–195, 2008.

[4] J. Zheng, "Cost-sensitive boosting neural networks for software defect prediction," Expert Syst. Appl., vol. 37, no. 6, pp. 4537–4543, 2010.

[5] R. Malhotra, "Comparative analysis of statistical and machine learning methods for predicting faulty modules," Appl. Soft Comput. J., vol. 21, pp. 286–297, 2014.

[6] P. Kumudha and R. Venkatesan, "Cost-Sensitive Radial Basis Function Neural Network Classifier for Software Defect Prediction," Sci. World J., vol. 2016, 2016.

[7] R. Mahajan, S. K. Gupta, and R. K. Bedi, "Design of software fault prediction model using BR technique," in Procedia Computer Science, vol. 46, no. Icict 2014, pp. 849–858, 2015.

[8] I. A. and A. Saha, "Software Defect Prediction: A Comparison Between Artificial Neural Network and Support Vector Machine," Adv. Comput. Commun. Technol., pp. 51–61, 2017.

[9] M. Singh and D. Singh Salaria, "Software Defect Prediction Tool based on Neural Network," Int. J. Comput. Appl., vol. 70, no. 22, pp. 22–28, 2013.

[10] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2016.

[11] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," Inf. Softw. Technol., vol. 54, no. 3, pp. 248–256, 2012.

[12] Y. Singh and R. Malhotra, Object-Oriented Software Engineering. PHI Learning Pvt. Ltd. New Delhi, 2012.

[13] R. Moser, W. Pedrycz, and G. Succi, "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction," pp. 181, 2008.

[14] E. Giger, M. D'Ambros, M. Pinzger, and H. C. Gall, "Method-level bug prediction," pp. 171, 2012.

[15] S. E. S. Taba, F. Khomh, Y. Zou, A. E. Hassan, and M. Nagappan, "Predicting bugs using antipatterns," IEEE Int. Conf. Softw. Maintenance, ICSM, pp. 270–279, 2013.

[16] K. Herzig, S. Just, A. Rau, and A. Zeller, "Predicting defects using change genealogies," 2013 IEEE 24th Int. Symp. Softw. Reliab. Eng. ISSRE 2013, pp. 118–127, 2013.

[17] S. Moustafa, M. Y. ElNainay, N. El Makky, and M. S. Abougabal, "Software bug prediction using weighted majority voting techniques," Alexandria Eng. J., vol. 57, no. 4, pp. 2763–2774, 2018.

[18] 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, pp. 27-32, 2017.

[19] 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, pp. 29-33, 2017.

[20] 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.

[21] 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.

[22] 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, pp. 393-398, 2018.

[23] 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, pp. 254-260, 2018.

[24] N. Farnaaz and M. A. Jabbar, "Random Forest Modeling for Network Intrusion Detection System," Procedia Comput. Sci., vol. 89, pp. 213–217, 2016.

[25] T. Fawcett, "An introduction to ROC analysis," Pattern Recognit. Lett., vol. 27, no. 8, pp. 861–874, 2006.

[26] 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, no. 9, pp. 1208–1215, 2013.

[27] "NASA – Software Defect Datasets [Online]. Available: https://nasa softwaredefectdatasets.wikispaces.com. [Accessed: 01-April-2019].

[28] "NASA Defect Dataset." [Online]. Available: https://github.com/klainfo/NASADefectDataset. [Accessed: 01-April-2019].

[29] 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.

[30] 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.

[31] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, and J. C. Riquelme, "Preliminary comparison of techniques for dealing with imbalance in software defect prediction," Proc. 18th Int. Conf. Eval. Assess. Softw. Eng. ACM, pp. 1–10, 2014.

[32] 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.