SOFTWARE DEFECT ESTIMATION USING MACHINE LEARNING ALGORITHMS

Dr.M.Dhasaratham, Professor, Department of Computer Science and Engineering, Teegala Krishna Reddy Engineering College, Hyd.
ChepuriSpandana, B.Tech.,Department of Computer Science and Engineering, Teegala Krishna Reddy Engineering College, Hyd.
dasarath.m@gmail.com,spandanachepuri257@gmail.com

ABSTRACT:

Software Engineering is a comprehensive domain since it requires a tight communication between system stakeholders and delivering the system to be developed within a determinate time and a limited budget. Delivering the customer requirements include procuring high performance by minimizing the system. Thanks to effective prediction of system defects on the front line of the project life cycle, the project's resources and the effort or the software developers can be allocated more efficiently for system development and quality assurance activities. The main aim of this paper is to evaluate the capability of machine learning algorithms in software defect prediction and find the best category while comparing seven machine learning algorithms within the context of four NASA datasets obtained from public PROMISE repository [12]. All in all, the results of ensemble learners category consisting of Random Forests (RF) and Bagging in defect prediction is pretty much its counterparts.
Keywords—Software quality metrics, Software defect predic-tion, Software fault prediction, Machine learning algorithms

INTRODUCTION

Developing a software system is an arduous process which contains planning, analysis, design, implementation, testing, integration and maintenance. A software engineer is expected to develop a software system on time and within limited the budget which are determined during the planning phase. During the development process, there can be some defects such as improper design, poor functional logic, improper data handling, wrong coding, etc. and these defects may cause errors which lead to rework, increases in development and maintenance costs decrease in customer satisfaction. A defect management approach should be applied in order to improve software quality by tracking of these defects. In this approach, defects are categorized depending on the severity and corrective and preventive actions are taken as per the severity defined. Studies have shown that 'defect prevention' strategies on behalf of 'defect detection' strategies are used in current methods [10]. Using defect prevention strategies to reduce defects generating during the software development the process is a costly job. It requires more effort and leads to increases in project costs. Accordingly, detecting defects in the software on the front line of the project life cycle is crucial. The implementation of machine learning algorithms which is the binary prediction model enables identifydefectprone modules in the software system before a failure occurs during development process. In this research, our aim is to evaluate the software defect prediction performance of seven machine learning algorithms by utilizing quality metrics; accuracy, precision, recall, F-measure associated with defects as an independent variable and find the best category while comparing software defect prediction performance of these machine learning algorithms within the context of four NASA datasets obtained from public PROMISE repository [12]. The selected machine learning algorithms for comparison are used for supervised learning to solve classification problems. They are two tree-structured classifier techniques: (i) Bagging and (ii) Random Forests (RF); two neural networks techniques: (i) Multilayer Perceptron (MLP) and (ii) Radial Basis Function (RBF); two Bayesian classifier techniques: (i) Naive Bayes and (ii) Multinomial Naive Bayes; and one discriminative classifier Support Vector Machine (SVM). The remainder of the paper is organized as follows: Section 2 briefly describes the related work, while Section 3 describes the experimental methodology in detail. Section 4 contains the conclusion of the experimental study and underlined some possible future research directions.

LITERATURE SURVEY:

There are a great variety of studies which have developed and applied statistical and machine learning based models for defect prediction in software systems. Basili et al. (1996) [1] have used logistic regression in order to examine what the effect of the suite of object-oriented design metrics is on the prediction of fault-prone classes. Khoshgoftaar et al. (1997) [7] have used the neural network in order to classify the modules of large telecommunication systems as fault-prone or not and compared it with a non-parametric discriminant model. The results of their study have shown that compared to the non-parametric discriminant model, the predictive accuracy of the neural network model had a better result. Then in 2002 [6], they made a case study by using regression trees to classify fault-prone modules of enormous telecommunication systems. Fenton et al. (2002) [4] have used Bayesian Belief Network in order to identify software defects. However, this machine learning algorithm has lots of limitations which have been recognized by Weaver(2003) [14] and Ma et al. (2007) [9]. Guo et al. (2004) [5] have applied Random Forest algorithm on software defect dataset introduced by NASA to predict fault-prone modules of software systems and compared their model with some statistical and machine learning models. The result of this comparison has shown that compared to other methods, the random forest algorithm has given better predictive accuracy. Ceylan et al. (2006) [2] have proposed a model which uses three machine learning algorithms that are Decision Tree, Multilayer Perceptron and Radial Basis Functions in order to identify the impact of this model to predict defects on different software metric datasets obtained from the real*life projects of three big-size software companies in Turkey. The results have shown thatall of the machine learning algorithms had similar results which have enabled to predict potentially defective software and take actions to correct them. Elish et al. (2008) [3] have investigated the impact of Support Vector Machines on four NASA datasets to predict defect-proneness of software systems and compared the prediction performance of SVM against eight statistical and machine learning models. The results have indicated that the prediction performance of SVM has been much better than others. Kim et al. (2011) [8] have investigated the impact of the noise on defect prediction to cope with the noise in defect data by using a noise detection and elimination algorithm. The results of the study have presented that noisy instances could be predicted with reasonable accuracy and applying elimination has improved the defect prediction accuracy. Wang at all. (2013) [13] have investigated re-sampling techniques, ensemble algorithms and threshold moving as class imbalance learning methods for software defect prediction. They have used different methods and among them, AdaBoost.NC had better defect prediction performance. They have also improved the effectiveness and efficiency of AdaBoost.NC by using a dynamic version of it. Ren at al. (2014) [11] have proposed a model to solve the class imbalance problem which causes a reduction in the performance of defect prediction. The Gaussian function has been used as kernel function for both the Asymmetric Kernel Partial Least Squares Classifier (AKPLSC) and Asymmetric Kernel Principal Component Analysis Classifier (AKPCAC) and NASA and SOFTLAB datasets have been used for experiments. The results have shown that the AKPLSC had better impact on retrieving the loss caused by class imbalance and the AKPCAC had better performance to predict defects on imbalanced datasets. There is also a systematic review study conducted byMalhotra to review the machine learning algorithms for software fault prediction.

EXISTINGSYSTEM:

In the existing system,have proposed a model to solve the class imbalance problem which causes a reduction in the performance of defect prediction. The Gaussian function has been used as kernel function for both the Asymmetric Kernel Partial Least Squares Classifier (AKPLSC) and Asymmetric Kernel Principal Component Analysis Classifier (AKPCAC) and NASA and SOFTLAB datasets have been used for experiments. The results have shown that the AKPLSC had better impact on retrieving the loss caused by class imbalance and the AKPCAC had better performance to predict defects on imbalanced datasets. There is also a systematic review study conducted by Malhotra to review the machine learning algorithms for software fault prediction.

PROPOSEDSYSTEM:

The proposed system includes SVM, Multilayer Perceptron, Run Bagging algorithm, Naive Bayes algorithm, Random Forest algorithm, Multinomial NB and Radial Basis Functions to solve the class misbalancing problem which causes in the decreasing performance of defect prediction. The dataset has been trained and spitted according to the constraints and using the accuracies has been defined in order to measure the defect estimation capability of various algorithms proposed.
Advantages of proposed system:
1. Predicted model is used for evaluating the performance measures.
2. We can apply various datasets in this project. But we are using NASA datasets in our project.
3. Software defects are classified to the extent.
4. Advance measures can be taken on selection of algorithm
5. Provides Better results.
6. Identify defects in the early stage of the project which in turn results in Customer loyalty.

METHODLOGY

A. **Datasets**
The datasets which are available from the public PROMISE repository [12] and used for this task are detailed in Table II. These datasets have different number of instances. The dataset with the most data in terms of the number of instances is PJ1. Data sets of different sizes have been selected to demonstrate the effect of data size on accuracy. In Table II, each dataset explained with language, number of attributes, number of instances, percentage of defective modules and description. The number of attributes is equal for each dataset. Attribute information is shown in Table I.
B. Learning Algorithms In this experiment, the study of Malhotraet. al. (2015) [10] have been guiding us while deciding to select which machine learning algorithms we have used for defect prediction in software systems. They categorized the machine learning algorithms based on distinct learners such as Ensemble Learners, Bayesian Learners, Neural Networks and SVM. According to these categories, we selected seven different machine learning algorithms to estimate software defect. These algorithms used and their categories are shown in Figure 1. Each algorithm is detailed below. K-fold Cross-Validation (CV) model is employed for each learning algorithm to model validation. The k value is determined as 10 in this experiment. Since the number of samples TABLE I
B.

TABLE I. Attribute Definition

Metric	Definition
loc	numeric % McCabe's line count of code
v(g)	numeric % McCabe "cyclomatic complexity"
ev(g)	numeric % McCabe "essential complexity"
iv(g)	numeric % McCabe "design complexity"
n	numeric % Halstead total operators + operands
v	numeric % Halstead "volume"
l	numeric % Halstead "program length"
d	numeric % Halstead "difficulty"
i	numeric % Halstead "intelligence"
e	numeric % Halstead "effort"
b	numeric % Halstead
t	numeric % Halstead's time estimator
lOCode	numeric % Halstead's line count
lOComment	numeric % Halstead's count of lines of comment
lOBlank	numeric % Halstead's count of blank lines
lOCodeAndComment	numeric
uniq_Op	numeric % unique operators
uniq_Opnd	numeric % unique operands
total_Op	numeric % total operators
total_Opnd	numeric % total operands
branchCount	numeric % of the flow graph
defects	{false,true} % module has/has not one or more r

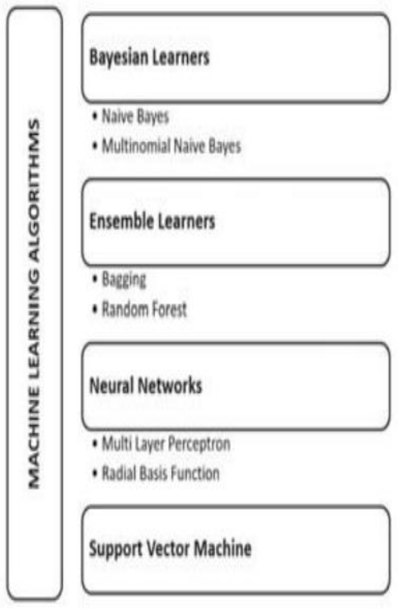
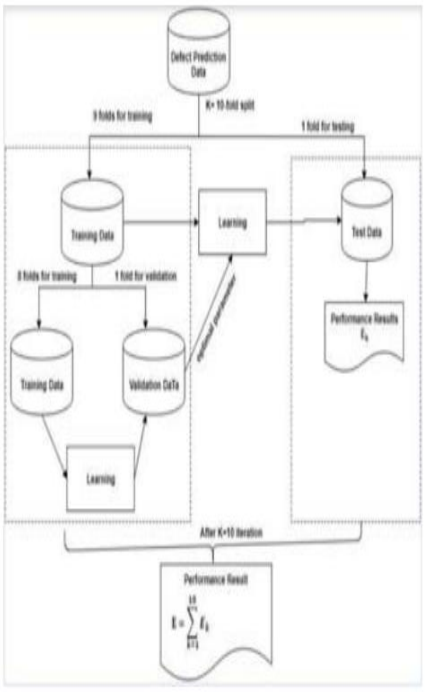


Fig. 1. Classification of ML Techniques for Software D

for Software Defect Prediction in the used datasets are equal to 10, the data is divided into 10 folds. That means k-1 objects in the dataset are used as training samples and one object is used as test sample in the each iteration. That is, every data fold is used as a validation set exactly once and falls into a training set k-1 times. Then the average error across all k trials which is equal to the number of samples in the dataset is computed. The process of k-Fold Cross-Validation of our study is as shown in Figure



ig. 2. Process of k-Fold Cross Validation for Software Defect Estimation

2. Fig. 2. Process of k-Fold Cross Validation for Software Defect Estimation

1) Bayesian Learners:

• Naive Bayes: Naive Bayes which is one of the most commonly used algorithms for classifying problems is simple probabilistic classifier and is based on Bayes

TABLE II

TABLE II. Dataset Properties

	Project	Language	# of Attributes	# of instances	% of Defective Modules	Description
Procedural	CM1	C	22	498	9.7	CM1 is a NASA spacecraft instrument written in "C".
	PC1	C	22	1109	6.9	Data from C functions. It is a flight software developed for earth orbiting satellite.
Object Oriented	KC1	C++	22	2109	15.4	KC1 is a system written by using C++ programming language. It implements storage management in order to receive and process ground data.
	KC2	Java	22	522	6.3	Data obtained from C++ functions. KC2 is a system developed for science data processing. It was developed by different developers than KC1 project as an extension of it. In this implementation, only some third-party software libaries of KC1 were used, the remainder of the software was developed differently.

Theorem. It determines the probability of each features occurring in each class and returns the outcome with the highest probability.

P(A|B) = P(B|A)P(A) P(B)

• 0XOWLQRPLDO 1DLYH %D\HV: Multinomial Naive Bayes classifier is obtained by enlarging Naive Bayes classifier. Differently from the Naive Bayes classifier, a multinomial distribution is used for each features.

2) Ensemble Learners:

• **Bagging**: This algorithm which is introduced by Leo Breiman and also called Bootstrap Aggregation is one of the ensemble methods. In this approach, N sub-samples of data from the training sample are created and the predictive model is trained by using these subset data. Sub-samples are chosen randomly with replacement. As a result, the final model is an ensemble of different models.

• **Random forest:** Random Forest algorithms which also called random decision forest is an ensemble tree-based learning algorithm. It makes a prediction over individual trees and selects the best vote of all predicted classes over trees to reduce overfitting and improve generalization accuracy. It is also the most flexible and easy to use for both classification and regression.

3) Neural Networks:

Multilayer Perceptron: Multilayer Perceptron which is one of the types of Neural Networks comprises of one input layer, one output layer and at least one or more hidden layers. This algorithm transfers the data from the input layer to the output layer, which is called feedforward. For training, the backpropagation technique is used. One hidden layer with (attributes + classes) / 2 units are used for this experiment. Each dataset has 22 attributes and 2 classes which are false and true. We determined the learning rate as 0.3 and momentum as 0.2 for each dataset.

•Radial Basis Function Network:
Radial Basis Function Network includes an input vector for classification, a layer of RBF neurons, and an output layer which has a node for each class. Dot products method is used between inputs and weights and for activation sigmoidal activation functions are used in MLP while in RBFN between inputs and weights Euclidean distances method is used and as activation function, Gaussian activation functions are used.

4) Support Vector Machines: Support vector machine (SVM) is a supervised machine learning method capable of both classification and regression. It is one of the most effective and simple methods used in classification. For classification, it is possible to separate two groups by drawing decision boundaries between two classes of data points in a hyperplane. The main objective of this algorithm is to find optimal hyperplane.

C.Evaluation Metrics To evaluate learning algorithms which are stated above, commonly used evaluation metrics are used such as accuracy, precision, recall, F-measure. The performance of the model of each algorithm is evaluated by using the confusion matrix which is called as an error matrix and is a summary of prediction results on a classification problem. Evaluation of model is the most important for classification problem where the output can be of two or more types of classes and the confusion matrix is one of the most commonly used and easiest metrics for determining the accuracy of the model. It has True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) values. • Positive (P) : Observation is positive (for example: is an defective).

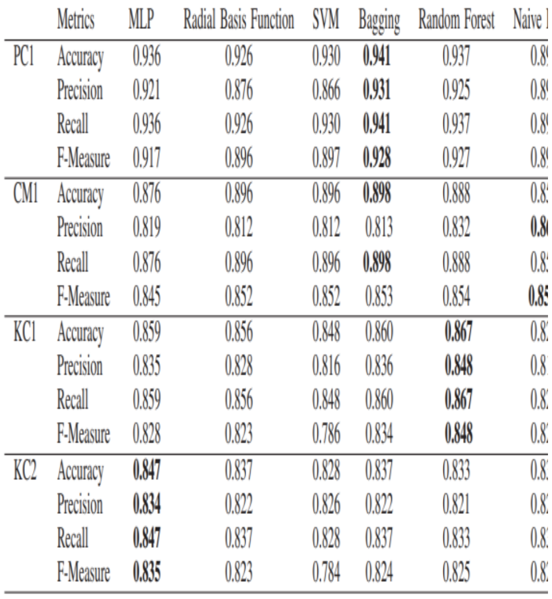
• Negative (N) : Observation is not positive (for example: is not an defective).

• True Positive (TP) : The model has estimated true and the test data is true.

• False Negative (FN) : The model has estimated false and the test data is true.
• True Negative (TN) : The model has estimated false and the test data is false.
• False Positive (FP) : The model has estimated true and the test data is false.

1) **Accuracy:** Accuracy which is called classification rate is given by the following relation: Accuracy = T P + T N /T P + T N + F P + F N

TABLE IIIRESULTS FOR EACH DATASET



	Metrics	MLP	Radial Basis Function	SVM	Bagging	Random Forest	Naive
PC1	Accuracy	0.936	0.926	0.930	**0.941**	0.937	0.8
	Precision	0.921	0.876	0.866	**0.931**	0.925	0.8
	Recall	0.936	0.926	0.930	**0.941**	0.937	0.8
	F-Measure	0.917	0.896	0.897	**0.928**	0.927	0.8
CM1	Accuracy	0.876	0.896	0.896	**0.898**	0.888	0.8
	Precision	0.819	0.812	0.812	0.813	0.832	**0.8**
	Recall	0.876	0.896	0.896	**0.898**	0.888	0.8
	F-Measure	0.845	0.852	0.852	0.853	0.854	**0.85**
KC1	Accuracy	0.859	0.856	0.848	0.860	**0.867**	0.8
	Precision	0.835	0.828	0.816	0.836	**0.848**	0.8
	Recall	0.859	0.856	0.848	0.860	**0.867**	0.8
	F-Measure	0.828	0.823	0.786	0.834	**0.848**	0.8
KC2	Accuracy	**0.847**	0.837	0.828	0.837	0.833	0.8
	Precision	**0.834**	0.822	0.826	0.822	0.821	0.8
	Recall	**0.847**	0.837	0.828	0.837	0.833	0.8
	F-Measure	**0.835**	0.823	0.784	0.824	0.825	0.8

2) **Recall:** To get the value of Recall, correctly predicted positive observations is divided by the all observations in actual class and it can be defined as below: Recall = T P T P + F N

3) **Precision**: Precision is the ratio of the total number of correctly classified positive examples to the number of predicted positive examples. As shown in Equation 4, As decreases the value of FP, precision increases and it indicates an example labeled as positive is indeed positive.

P recision = T P /T P + F P

3) F-measure: Unlike recall and precision, this metric takes into account both false positives(FP) and false negatives(FN). F-measure is the weighted harmonic mean of the precision and recall of the test. The equation of this metric is shown in Equation 5. P recision = 2 ∗ Recall ∗ P recision Recall + P recision (5) D. Experimental Results We performed experiments on the four different datasets which have a different number of attributes and results were shown in Table III. There are many ways to evaluate any machine learning algorithm and evaluation of the model is a very essential part of any project. In this experiment, different evaluation metrics which are given above are used to evaluate model performance. For each machine learning algorithm and each dataset, the best classification performance result is showed in boldfaced print. The first notable observation from these experimental results which are shown in Table III is that RF and Bagging learning algorithm which is a tree-based algorithm is better than other learning algorithm categories. The performance difference between each machine learning algorithm is shown in Figure 3 clearly. As shown in Figure 3, the results of the tree-based learning algorithm are better compared to other algorithms except for KC2 dataset. Although datasets of different sizes were used, no major differences were observed in performance. Figure 3 shows that ensemble learners are better at software defect estimation and it is also a powerful way to improve the performance of the model. It is a more successful model than individual models because of combining several diverse classifiers together.

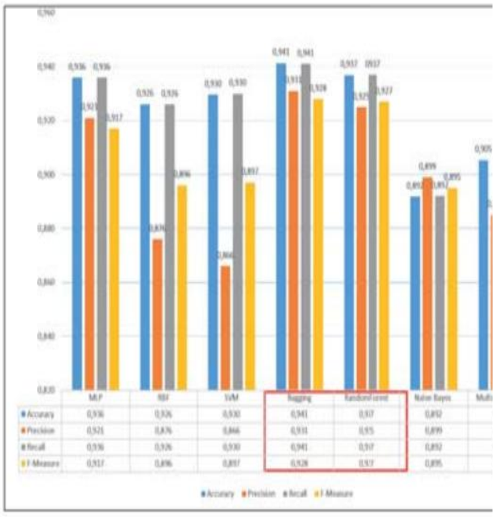
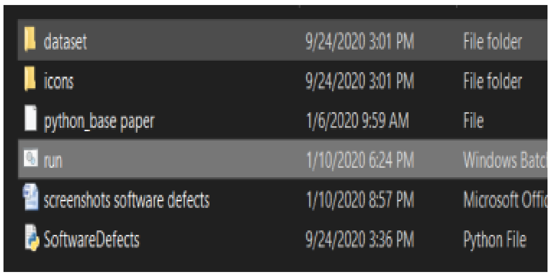


Fig. 3. Accuracies of ML Techniques of each dataset for Sof
Prediction

RESULTS:

OUTPUT SCREENS:

Step-1: Double click on windows batch file to load the GUI:



dataset	9/24/2020 3:01 PM	File folder
icons	9/24/2020 3:01 PM	File folder
python_base paper	1/6/2020 9:59 AM	File
run	1/10/2020 6:24 PM	Windows Batc
screenshots software defects	1/10/2020 8:57 PM	Microsoft Offic
SoftwareDefects	9/24/2020 3:36 PM	Python File

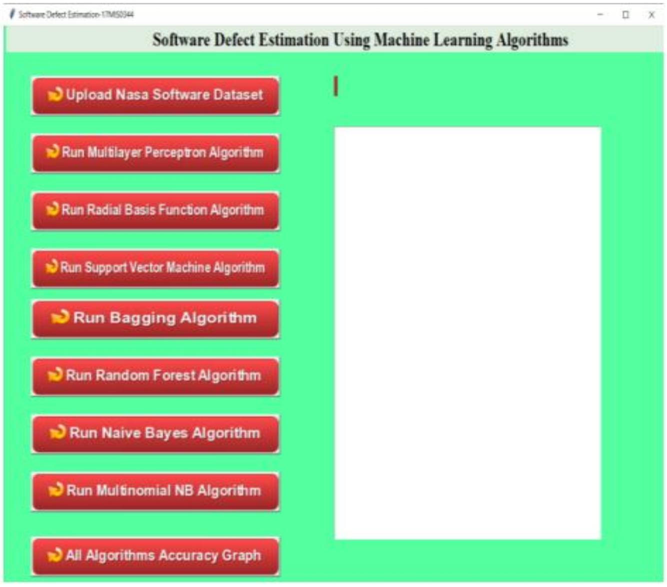
GUI:

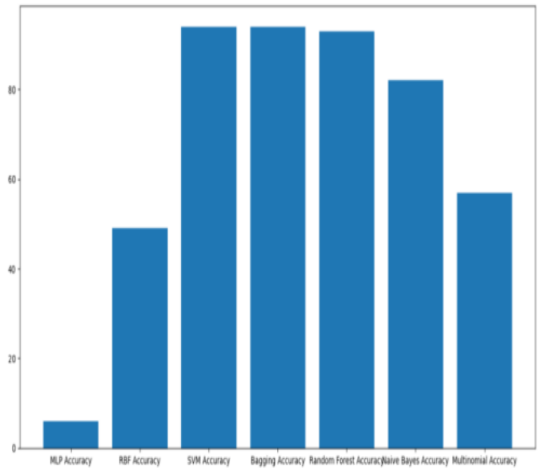
Step-2: Click on upload NASA Data Set: On uploading CM1 Data set we can see total dataset size and training size records and testing size records application obtained from dataset to build train model.

Step-3:Now click on 'Run Multilayer Perceptron Algorithm' button to generate model and to get its accuracy Step-4: Now click on "Run Radial Basis Function Algorithm" button to generate model and to get its accuracy Step-5: Click on Support vector Machine Algorithm to generate its accuracy.

Step-6: Click on Run Bagging Algorithm for generating accuracy.

Step-7: Click on Run Random Forest algorithm for generating accuracy. Step-8: Click on Naive Bayes algorithm for generating accuracy. Step-9: Click on Multinomial NB algorithm for generating accuracy. Step-10: Click on "All Algorithms Accuracy Graph" Button:


Software Defect Estimation-17MIS0344
Software Defect Estimation Using Machine Learning Algorithms
Upload Nasa Software Dataset
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph



In above graph x-axis represents algorithm name and yaxis represents accuracy of those algorithms.

CONCLUSION :

In this experimental study, seven machine learning algorithms are used to predict defectiveness of software systems before they are released to the real environment and/or delivered to the customers and the best category which has the most capability to predict the software defects are tried to find while comparing them based on software quality metrics which are accuracy, precision, recall and F-measure. We carry out this experimental study with four NASA datasets which are PC1, CM1, KC1 and KC2. These datasets are obtained from public PROMISE repository. The results of this experimental study indicate that tree-structured classifiers in other words ensemble learners which are Random Forests and Bagging have better defect prediction performance compared to its counterparts. Especially, the capability of Bagging in predicting software defectiveness is better. When applied to all datasets, the overall accuracy, precision, recall and FMeasure of Bagging is within 83,7-94,1%, 81,3-93,1%, 83,7- 94,1% and 82,4-92,8% respectively.For PC1 dataset, Bagging outperforms all other machine learning techniques in all quality metric. However, Naive Bayes outperforms Bagging in precision and F-Measure while Bagging outperforms it in accuracy and recall for CM1 dataset. Random Forests outperforms all machine learning techniques in all quality metrics for KC1 dataset. Finally, for KC2 dataset, MLP outperforms all machine learning techniques in all quality metrics for KC2 dataset. It is deductive from obtained results that tree-structured classifiers are more suitable for softwaredefect prediction. Moreover, it is recommended to software companies to utilize tree-structured classifiers for software defect prediction due to its performance. Utilizing these techniques enables them to save software testing and maintenance costs by identifying defects in the early phase of project life cycle and taking corrective and preventive actions before they becomes failures. Conducting additional experimental studies by using different datasets would be one direction of future work. These datasets would be obtained from the open repositories or software companies. Second direction of the future work would be conducting an experimental study by applying deep learning algorithms additional to these machine learning algorithms. Bringing into existence of new attributes by using combination of previous attributes would be another direction of the future work. In conclusion, it would be practical to carry out a case study by using distinct software quality datasets obtained from real-life projects of software companies having different company sizes.

REFERENCES:

[1] Victor R Basili, Lionel C. Briand, and Walcelio L Melo. ´ A validation of object-oriented design metrics as quality indicators. IEEE Transactions on software engineering, 22(10):751–761, 1996.

[2] EvrenCeylan, F OnurKutlubay, and Ayse B Bener. Software defect identification using machine learning techniques. In 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06), pages 240–247. IEEE, 2006.

[3] Karim O Elish and Mahmoud O Elish. Predicting defect-prone software modules using support vector machines. Journal of Systems and Software, 81(5):649–660, 2008.

[4] Norman Fenton, Paul Krause, and Martin Neil. Software measurement: Uncertainty and causal modeling. IEEE software, 19(4):116–122, 2002.

[5] LanGuo, Yan Ma, BojanCukic, and Harshinder Singh. Robust prediction of fault-proneness by random forests. In 15th International Symposium on Software Reliability Engineering, pages 417–428. IEEE, 2004.

[6] Taghi M Khoshgoftaar, Edward B Allen, and Jianyu Deng. Using regression trees to classify fault-prone software modules. IEEE Transactions on reliability, 51(4):455–462, 2002.

[7] Taghi M Khoshgoftaar, Edward B Allen, John P Hudepohl, and Stephen J Aud. Application of neural networks to software quality modeling of a very large telecommunications system. IEEE Transactions on Neural Networks, 8(4):902–909, 1997.

[8] Sunghun Kim, Hongyu Zhang, Rongxin Wu, and Liang Gong. Dealing with noise in defect prediction. In 2011 33rd International Conference on Software Engineering (ICSE), pages 481–490. IEEE, 2011.