SOFTWARE DEFECT ESTIMATION USING MACHINE LEARNING ALGORITHMS

[1]S.Pranav, [2]Ramnath Srivatsa, [3]Ch. Karthik Reddy, [4]Mrs. G.Bindu Madhavi,
[1]Department of CSE, Anurag Group of Institutions, Venkatapur, Ghatkesar, Hyderabad, Telangana 500038
16h61a0549@cvsr.ac.in
[2]Department of CSE, Anurag Group of Institutions, Venkatapur, Ghatkesar, Hyderabad, Telangana 500038
16h61a0551@cvsr.ac.in
[3]Department of CSE, Anurag Group of Institutions, Venkatapur, Ghatkesar, Hyderabad, Telangana 500038
16pq1a0523@cvsr.ac.in
[4]Associate Professor, Department of CSE, Anurag Group of Institutions, Venkatapur, Ghatkesar, Hyderabad, Telangana 500038

ABSTRACT

Software Engineering is a comprehensive domain since it requires a tight communication between system stake holders 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 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 NASA datasets obtained from public repository.

1. 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. 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 identify defect prone 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. 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).

2. RELATED WORK

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 machinelearningalgorithmsthatareDecisionTree, MultilayerPerceptronandRadialBasisFunctionsi n 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 that all 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 by Malhotra to review the machine learning algorithms for software faultprediction.

3.EXPERIMENTAL METHODOLOGY

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 1.

B. Learning Algorithms

In this experiment, the study of Malhotra et. 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.

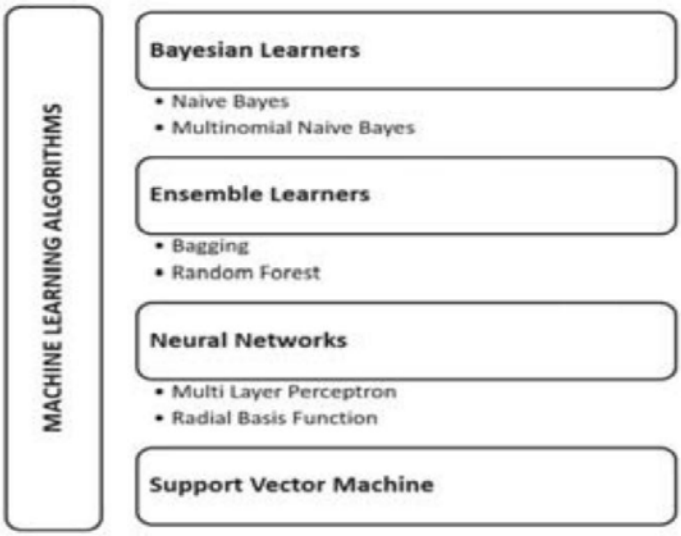

Fig 1: Classification of ML techniques for Software Defect Prediction

Table 1: 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 comments
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 reported defects

Four measures of performance that are commonly used with classification and prediction models are Precision score, Recall score, F-score (or) F1-measure, and Accuracy.

- Precision score: Precision score refers to the ratio of the correctly positively labeled by our program to all positively labeled. With respect to this project, it refers to the ratio of images correctly classified as tumor-affected by the model to all images that are labeled as tumor-affected by the model.

Precision = TP / (TP + FP)

- Recall score: Recall score refers to the ratio of the correctly positively labeled images by our program to all images who are positive in reality. In other words, it refers to the ratio of images classified as tumor-affected by the model to all images that are tumor-affected in reality.

Recall = TP / (TP + FN)

In the above formulae,

- TP => TruePositive
- FP => FalsePositive
- FN => False Negative

- F-score (F1-measure): F-score is the harmonic mean of precision and recall. It is considered to be most apt for measuring performance as it considers both precision and recall values. A good F- score indicates a good balance between precision and recall of a model.

F-Score = 2 * (Recall * Precision) / (Recall + Precision)

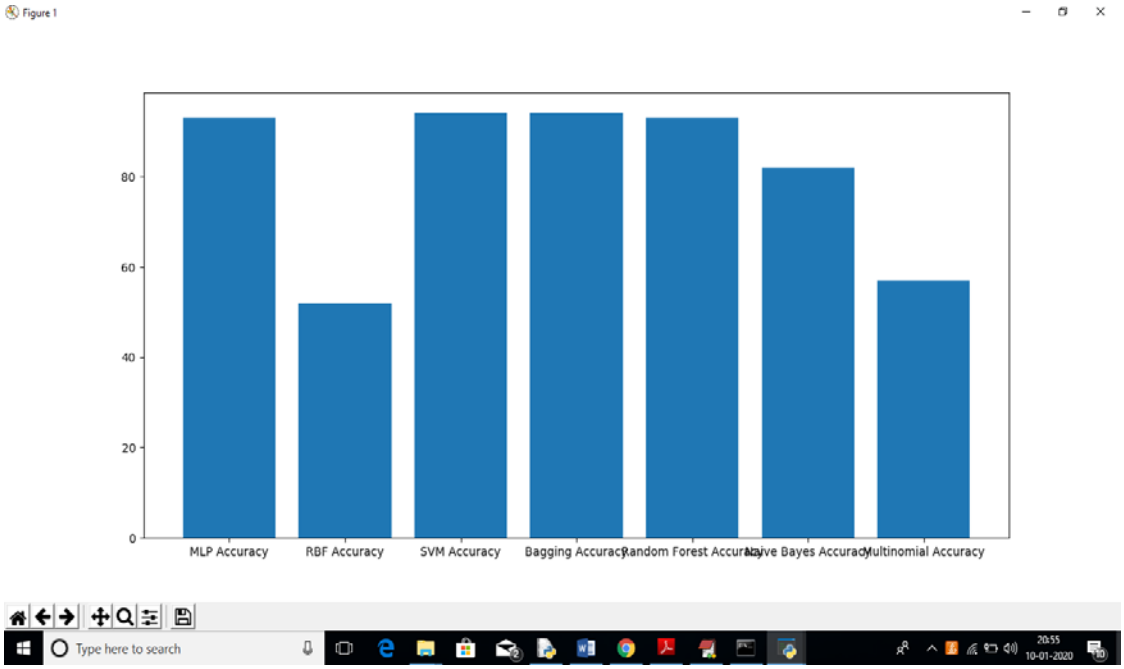

Fig 2: Comparision between different algorithms

4. 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 comparingthembasedonsoftwarequalitymetrics whichareaccuracy,precision,recallandF-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 software defectprediction.

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. Evren Ceylan, F Onur Kutlubay, 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. Iqbal S, Ghani MU, Saba T, Rehman A. Brain tumor segmentation in multi-spectral MRI using convolutional neural networks (CNN). Microsc Res Tech. 2018;00:1–9.