# SOFTWARE DEFECT ESTIMATION USING DISTINCT LEARNING ALGORITHMS

**SUBRAH DIVYASRI, YALAMANCHILI RAVI, PAMARTHI DURGA BHAVANI,**

**PONUGUMATI ROHITH,** Student, Department of CSE, NRI INSTITUTE OF TECHNOLOGY, Vijayawada, A.P., India.

**Dr.CH. SURYA KIRAN**, Professor, Department of CSE, NRI INSTITUTE OF TECHNOLOGY, Vijayawada, A.P., India.

**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 the public PROMISE repository.

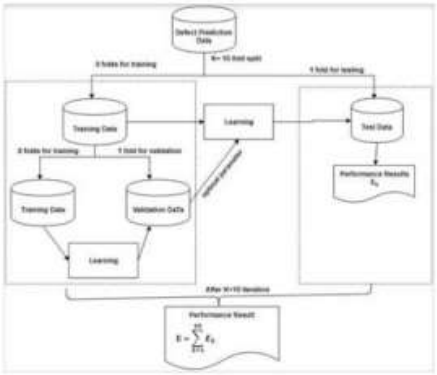
## INTRODUCTION

The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here. Supervised learning involves somehow modeling the relationship between measured

features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks. In classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data.



## LITERATURE SURVEY

**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) [2]** 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 [3], they made a case study by using regression trees to classify fault-prone modules of enormous telecommunication systems

## EXISTING SYSTEM

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. There are a great variety of studies which have developed and applied statistical and machine learning based models for defect prediction in software systems.

## PROPOSEDSYSTEM

In this paper author is evaluating performance of various machine learning algorithms such as SVM, Bagging, Naïve Bayes, Multinomial

Naïve Bayes, RBF, Random Forest and Multilayer Perceptron Algorithms to detect bugs or defects from Software Components. Defects will occur in software components due to poor coding which may increase software development and maintenance cost and this problem leads to dis-satisfaction from customers. To detect defects from software components various techniques were developed but right now machine learning algorithms are gaining lots of popularity due to its better performance. So in this paper also author is using machine learning algorithms to detect defects from software modules. In this paper author is using dataset from NASA Software components and the name of those datasets are CM1 and KC1. I am also using same datasets to evaluate performance of above-mentioned algorithms.

## IMPLEMENTATION

1) 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.

2) Neural Networks:

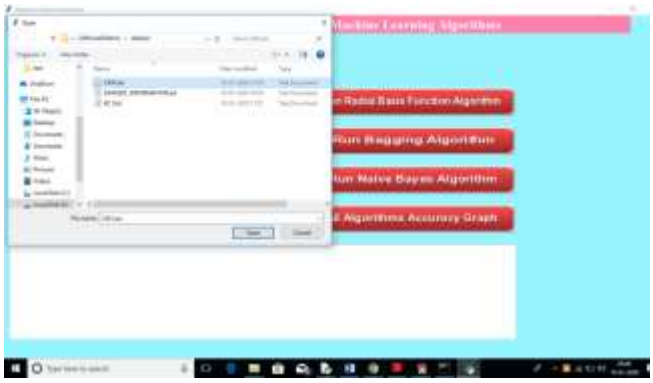
• Multilayer Perceptron: Multilayer Perceptron which is one of the types of Neural Networks comprises of one input layer, one output layer and at least one or more hidden layers. This algorithm transfers the data from the input layer to the output layer, which is called feedforward. For training, the 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: 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.

3) 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.

## SAMPLE RESULTS


In above screen click on 'Upload Nasa Software Dataset' button to upload dataset



In above screen uploading 'CM1.txt' dataset and information of this dataset you can read from internet of 'DATASET_INFORMATION' file from above screen. After uploading dataset will get below screen

In above screen we can see total dataset size and training size records and testing size records application obtained from dataset to build train model. Now click on 'Run Multilayer Perceptron Algorithm' button to generate model and to get its accuracy

In above screen we can see multilayer perceptron fmeasure, recall and accuracy values and scroll down in text area to see all details.

## 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.

## FUTURE SCOPE FOR FURTHER DEVELOPMENT

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.

## 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] Lan Guo, Yan Ma, Bojan Cukic, 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.

[9] Yan Ma, Lan Guo, and Bojan Cukic. A statistical framework for the prediction of fault-proneness. In Advances in Machine Learning Applications in Software Engineering, pages 237–263. IGI Global, 2007.

[10] Ruchika Malhotra. A systematic review of machine learning techniques for software fault prediction. Applied Soft Computing, 27:504–518, 2015.

[11] Jinsheng Ren, Ke Qin, Ying Ma, and Guangchun Luo. On software defect prediction using machine learning. Journal of Applied Mathematics, 2021.