iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VII **Month of publication:** July 2022

DOI: https://doi.org/10.22214/ijraset.2022.44551

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VII July 2022- Available at www.ijraset.com

Software Defect Estimation Using Machine Learning

Dr. Md Jaffar Sadiq[1], Shaik Ruma Sufian[2], K. Sowmya Reddy[3], Madapati Vinisha[4]
[1]*Associate Professor, Dept of IT, Sreenidhi Institute of Science and Technology*
[2, 3, 4]*B.Tech (IT) Student*

Abstract: ***In recent years, software has become increasingly vital in our daily lives. We come across it in various forms like Alexa to automatic mops.***
Building a software is a huge task, it involves many stages. The developer is required to build or design a software within the given deadline and with the allotted budget. During which few erroneous decisions might be taken which lead to poor logic, improper design etc.
That would make the user dissatisfied. In order to let the device achieve the users desired outcome we must build a flawless software. To build one as that, one must predict if any defects are present.
As a result, we employ seven machine learning algorithms to assist us in predicting any software flaws.
Keywords: Software development, Machine learning

I. INTRODUCTION

Software development is both necessary and time-consuming. As discussed before it includes various stages. Project planning, requirements, design, coding, testing, maintenance, and deployment are the seven steps that software development encompasses. A software engineer is expected to build or design a software within the given deadline and with the allotted budget. During this phase few erroneous decisions might be taken that lead to poor logic, wrong coding, improper design etc. Giving rise to rework. Rework increases the maintenance costs.
We want to see how well seven machine learning algorithms forecast software problems using quality metrics including precision, accuracy, F-measure, and recall.Bagging, Multilayer perceptron, Multinomial naive bayes, Naive bayes, Radial bias, Random forest, and Support vector machine were among the machine learning techniques used.

II. RELATED WORK

Numerous studies have developed and used factual and AI-based defect prediction algorithms in programming frameworks. Basili et al (1996) [2] have utilized calculated relapse to inspect what the impact of the set-up of item situated plan measurements is on the expectation of shortcoming inclined classes. Khoshgoftaar et al (1997) [3] have utilized the neural organization to group the modules of enormous media transmission frameworks as flaw inclined or not and compared it to a non-parametric discriminant model. To identify software problems, Fenton et al. (2002) [8] utilized a Bayesian belief.
But, Weaver(2003) [9] along with Ma et al. (2007) [10] discovered that limitations were present in the latter. Ceylan et al. (2006) [4] proposed a decision tree, radial bias function and multilayer perceptron based model which was applied on three huge companies in Turkey and detected the flaws in the software, which lead the companies to make necessary changes. Elish et al. (2008) [5] applied SVM on the four NASA data sets [12] against other machine learning algorithms and realizedSVM's performance was superior to that of others. Wang at all. (2013) [7] worked on ensemble algorithms and re-sampling techniques, they used various methods along with AdaBoost.NC.

III. METHODOLOGY

A. Data set

In this case, two data sets are employed. They're NASA data sets taken from the "PROMISE" repository that are open to the public.[1]. The data sets have 22 instances. To demonstrate how data size affects accuracy, data sets of various sizes were chosen.Language, number of attributes, number of instances, proportion of defective modules, and description are all included in each data set, it is depicted in Table II.Every data set contains equal number of attributes. The attribute details are shown below in Table I.

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

Table I

	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.

Table II

B. Algorithms

The seven machine algorithms used or selected for this experiment have been taken from Malhotra et. al. (2015) [6] andare as follows:

1) Bayesian Learners

- *Naive Bayes:* Based on the Bayes Theorem. It determines in every class the probability of every feature and returns the highest probable outcome.It used to tackle classification problems. P(A|B) = P(B|A)P(A) P(B)
- *Multinomial Naive Bayes:* To reach this outcome, size of the NB classifier is to be increased. A multinomial distribution is assigned to each feature.

2) Ensemble Learners

- *Bagging:* Bagging is obtained from bootstrapping and aggregating. It is the combination of both. Bootstrapping is nothing but selecting randomly some samples from the given original data set. In bootstrapping selections can be repeated. And each bootstrapped data set is converted into a decision tree. And the output is predicted based on the frequencies.
- *Random Forest:* Based on tree. It makes a prediction for each tree and chooses the best one to avoid over fitting and improve generalisation accuracy from the most frequently occurring results of all projected classes over trees. It is the most adaptable, efficient, and user-friendly tool for dealing with classification and regression issues.

3) Neural Networks

- *Multilayer Perceptron:* A multilayer perceptron is used in a Neural Network. The input, output, and at least one or more hidden layers are the three layers that make up this system. The feed forward technique is used.
- *Radial basis Function:* It is similar to mathematical function. It assigns a real value to each input which is taken from the original data set and the obtained value is always positive. It always gives a positive value as it is based on the distance measure. It also contains three layers input, output and hidden layers.

4) Support Vector Machines

The SVM basically create a boundary line it divides the classes. This boundary line is known as hyper plane. It uses two vectors i.e., points to create a hyper plane. These two vectors are generally outliers. As it uses two vectors for the support it is names as support vector machine algorithm. Svm is classified into two types. They are linear and non linear svm. Linear svm is used when the given original data is in linear format. Otherwise non linear svm is used.SVM selection helps to create limit points/vectors of the hyper plane.

C. Evaluation Metrics

To assess the above-mentioned learning algorithms, generally used assessment metrics such as the terms "accuracy," "precision," "recall," and "F-measure" are used. Confusion matrix is none other than the error matrix, it is a summary of expected outcomes on a a classification issue used to determine the efficacy of each algorithm's model. It is one of the commonly used as well as easy metrics that determines the accuracy of a model in classification problems where the output comprises of classes of two or more than two classes.

It is possible to obtain true positive and negative results as well as misleading positive and negative values.

- Positive (P):It's used to determine whether a finding is favourable.
- Negative (N): It's utilised to figure out if the observation result isn't good.
- True Positive (TP): When both the test data and estimated results are true, it is taken as true positive.
- False Negative (FN): When the estimated result is false and test data is true then it is considered as false negative.
- True Negative (TN): When the estimated result is false and test data is also false then it is considered as true negative.
- False Positive (FP):False positive occurs when the estimated result is correct but the test data is incorrect.

The four quality metrics are as follows:

Accuracy:

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

The classification rate is a measure of accuracy.

Recall:

The number of accurately predicted positive observations in the actual class is divided by the total number of observations.

$$Recall = \frac{TP}{TP + FN}$$

Precision:

The number of positive cases that were correctly classified divided by the number of positive examples that were expected.

$$Precision = \frac{TP}{TP + FP}$$

F-measure:

This statistic includes FNs and FPs. The precision and recall of the test as a weighted harmonic mean

$$Precision = \frac{2 * Recall * Precision}{Recall + Precision}$$

IV. RESULT ANALYSIS

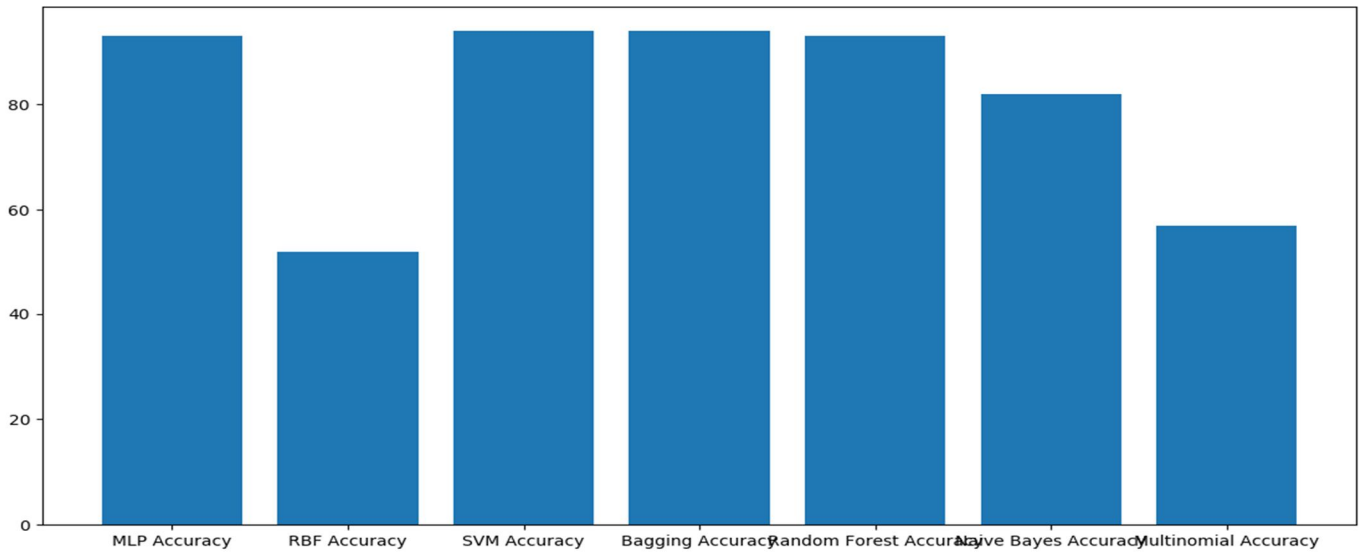


Table III

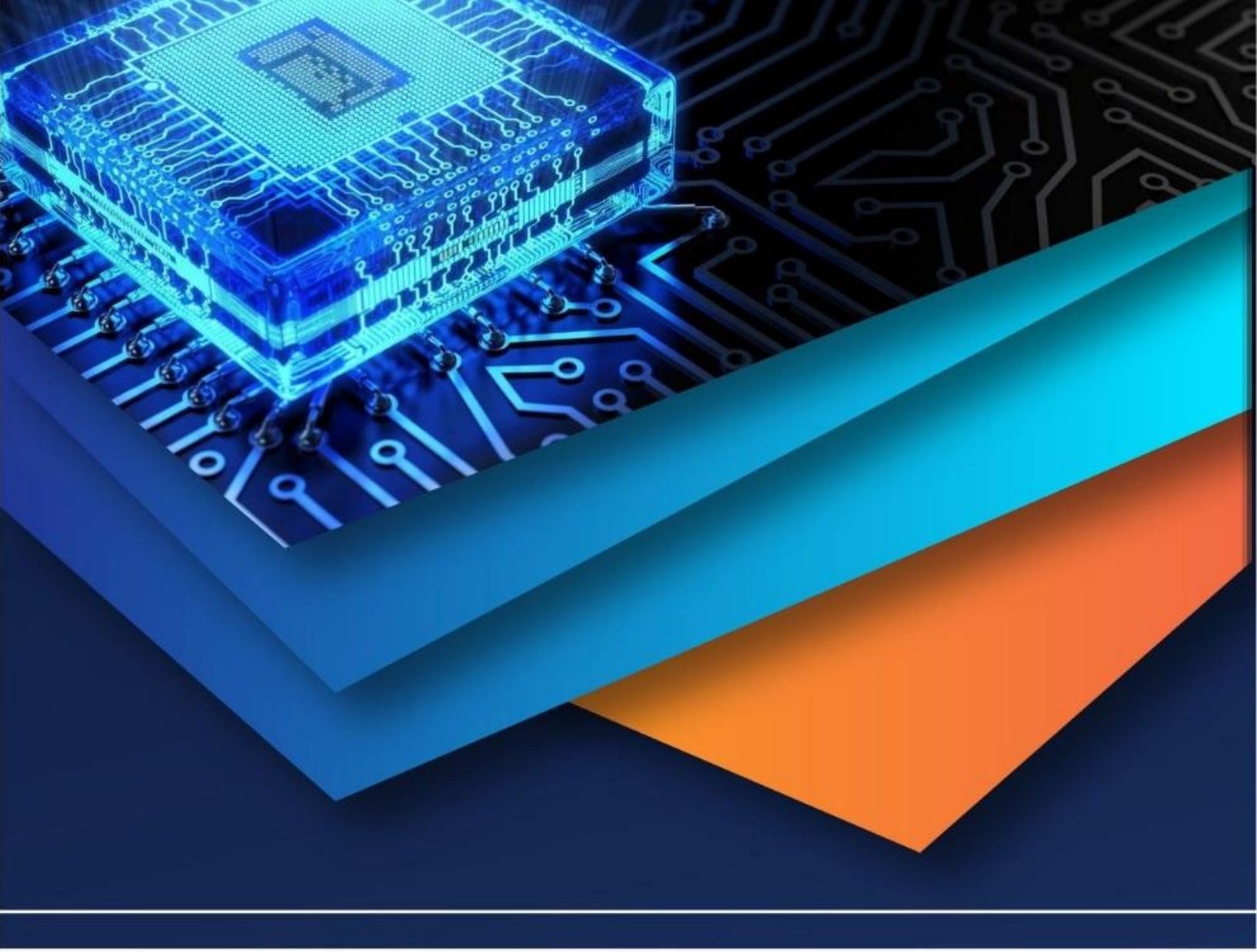
The graph (Table III) shows the accuracy percentages of all the seven algorithms which are used in our project. On the x axis different algorithms are displayed and on the y axis the accuracy levels of the algorithms are displayed. Our project results shows that the bagging and random forests algorithms which come under tree structured classifiers has high accuracy in predicting software defects while compared to the other algorithms. Out of all the seven different machine learning algorithms bagging stands as the best and the most accurate in estimating software defects. The bagging has overall accuracy and precision high when applied to all the data sets. But naive bayes has greater precision and f-measure when compared to bagging algorithm. Based on our experimental study it is evident that the tree structured classifiers are more accurate algorithms in detecting software defects. And based on our study it is clear that Rbf radial basis function has least accuracy when estimating software defects.

V. CONCLUSION

The project's goal is to find software flaws. It reduces the rework, time, manpower and maintenance costs of a company as the defects or flaws present in the software are detected during the beginning stages. By using the seven machine learning algorithms we predict the defect in the software by comparing the results based on the four metrics, namely Accuracy, Recall, Precision and F-Measure. After the prediction, necessary changes are made. This way the flawless software is released into the real environment which leads to customer satisfaction as well as it profits the company.

REFERENCES

[1] J. Sayyad Shirabad and T.J. Menzies. The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada, 2005.
[2] 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.
[3] 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.
[4] 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.
[5] 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.
[6] Ruchika Malhotra. A systematic review of machine learning techniques for software fault prediction. Applied Soft Computing, 27:504–518, 2015.
[7] Shuo Wang and Xin Yao. Using class imbalance learning for software defect prediction. IEEE Transactions on Reliability, 62(2):434–443, 2013.
[8] Norman Fenton, Paul Krause, and Martin Neil. Software measurement: Uncertainty and causal modeling. IEEE software, 19(4):116–122, 2002.
[9] Robert Andrew Weaver. The safety of software: Constructing and assuring arguments. University of York, Department of Computer Science, 2003.
[10] 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.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY