Software Fault Prediction Using Machine Learning

Tushar Arora
Information Technology
Maharaja Agrasen Institute of Technology
tushararora1401@gmail.com

Harshit Saini
Information Technology
Maharaja Agrasen Institute of Technology
harshitsaini3582@gmail.com

ABSTRACT

The IT and software industry has grown tremendously over the past few years, creating an increasing impact on the lives of people and on society as a whole. Consequently, we must make the software and applications more accurate, free of major errors, and more reliable. Therefore, predicting software flaws could be very useful in the IT field and will have a profound impact on society at large.

Keywords— supervised; unsupervised; reinforced; linear regression; Linear regression decision tree; python programming; Jupyter Notebook; confusion matrix;

1. INTRODUCTION

The vast area of software development and different applications makes it challenging for software developers and also customers to observe, maintains and manage software applications. Moreover, the fourth industrial revolution employs artificial intelligence by software industry is one of the promising sectors of modern times that observes a constant transformation in its practices because of the automating large quantities of software technologies . The size and complexity of current software is increasing day by day. As a result, software engineers are struggling continuously with faults from the beginning of the development phase.

The classification of the software faults is important in real-time, otherwise, the effort and cost of finding defects hiding in an application are also rising fast. This inspires the development of automated fault prediction models for software fault prediction that can forecast the software defects. If software defects are identified before the release of software that can help the developer to allocate and fix those defect modules easily.

2. MATERIALS & METHODS

2.1 Data Collection

In this project, we have used 3 open source publicly available data from PROMISE Software Engineering Database. These datasets Tim Menzies et al. have been used in their research paper [3]. In another study, Jureczko et al. [13] have been assembled a software fault prediction model to predict the software defects using machine learning algorithms. They have discussed in their paper about 8 projects (PROMISE Repository) data and by taking 19 CK metrics and McCabe metrics for constructed a predictive model. In our study, we have used 22 attributes for building our automated fault predict model. Table 1 shows 22 different attributes from software defect datasets including 21 independent metrics and one is outcome information. i.e. which is faulty and no-fault.

Table 1. List of the metrics

No	Metrics name	Type
1	Line of code	McCabe
2	Cyclomatic complexity	McCabe
3	Essential complexity	McCabe
4	Design complexity	McCabe
5	Halstead operators and operands	Halstead
6	Halstead volume	Halstead
7	Halstead program length	Halstead
8	Halstead difficulty	Halstead
9	Halstead intelligence	Halstead
10	Halstead effort	Halstead
11	Halstead time estimator	Halstead
12	Halstead line count	Halstead
13	Halstead comments count	Halstead
14	Halstead blank line count	Halstead
15	IO code and comments	Miscellaneous
16	Unique operators	Miscellaneous
17	Unique operands	Miscellaneous
18	Total operators	Miscellaneous
19	Total operands	Miscellaneous
20	Branch count	Miscellaneous
21	b: numeric	Halstead
22	Defects	False or true

We are using JM1, CM1, PC1 datasets which were implemented in C language. Table 2 depicted details about detail of all datasets with their features.

Table 2: Details about datasets

No	Dataset	Missing attribute	Instance	Class distribution	
				True	False
1	JM1	None	10885	8779 (80.65%)	2106 (19.35%)
2	CM1	None	498	49 (9.83%)	449 (90.16%)
3	PC1	None	1109	1032 (93.05%)	77 (6.94%)

2.2 Classification Techniques

Machine learning algorithm has been creating a significant role in software engineering fields. In recent years, machine learning techniques are one of the most operational techniques what are gained significantly high performance in real-world problems for the research and technical community. Harshita et al. [14] discussed in their review, there are common use of machine learning techniques for constructing software fault prediction models such as fuzzy logic-based software defect prediction, Naïve Bayes (NB), neural network (NN), random forest (RF), support vector machine (SVM), P-SVM, k-nearest neighbour's (KNN), etc. Ruchita Malhotra [15] described in her systematic mapping study, the top five machine learning techniques were used to software defect analysis such as DT (46%), NB (74%), MLP in NN (85%), RF (59%), SVM (27.7%), etc.

In this study, 4 machine learning (ML) techniques have been considered to construct the defect model: k-nearest neighbour's (KNN), Decision Tree (DT), Naïve Bayes (NB) and Random Forest (RF)

2.3 Machine Learning Algorithms

Machine Learning relies on different algorithms to solve data problems. Data scientists like to point out that there's no single one-size-fits-all type of algorithm that is best to solve a problem. The kind of algorithm employed depends on the kind of problem you wish to solve, the number of variables, the kind of model that would suit it best and so on.

A. Decision Tree

Decision Tree is a supervised learning technique that can be used for both classification and regression problems, but mostly it is preferred for solving classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision Tree, there are two nodes, which are the Decision Node and Leaf Node.

The goal of this algorithm is to create a model that predicts the value of a target variable, for which the decision tree uses the tree representation to solve the problem in which the leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

B. Random Forest

Random Forest classifier is a supervised learning technique in machine learning. It can be used to solve both Classification and Regression problems in machine learning. It is based on the process of combining multiple classifiers to solve a complex problem and to improve the performance of the model, which is known as ensemble learning. Random Forest consists of several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Rather than relying on a single decision tree, the random forest acquires the prediction from each tree, and based on the majority of votes for predictions, it predicts the final output. The higher number of trees in the forest leads to better accuracy and also prevents the problem of over fitting. The final output is taken by using the majority voting classifier for a classification problem while in the case of a regression problem the final output is the mean of all the outputs.

C. Naive Bayes

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.It is mainly used in text classification that includes a high-dimensional training dataset.Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

The formula for Bayes' theorem is given as:

$$P(A|B)= \frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

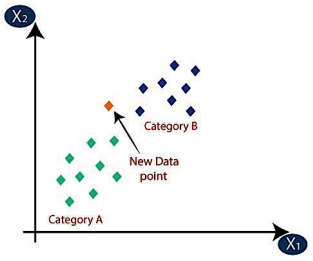
P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

D. K-nearest Neighbours (KNN)

It work on the basis of distance between the location of data and on the basis of this distinct data are classified with each other. All the other group of data are called neighbour of each other and number of neighbour are decided by the user which play very crucial role in analysis of the dataset.

In the above Fig. k=3 shows that there are three neighbor that means three different type of data are there. Each cluster represented in two dimensional space whose coordinates are represented as (Xi,Yi) where Xi is the x-axis, Y represent y- axis and i= 1,2,3,….n.



2.4 Performance Measurement

Once the predictive model has been built, it can be applied to perform a test to predict the fault modules inside the software fault datasets. In this work, we examined the ML prediction models, utilizing six classification algorithms, based on different statistical techniques such as confusion matrix (True Positive = TP, True Negative = TN, False Positive = FP, False Negative = FN), recall, precision, F1 measure, etc. Table 3 shows a quality measure of predictive model based on confusion matrix as below

Table 3. Performance measurement criteria

Metrics	Mathematical formula
Accuracy	$\frac{(TP+TN)}{(TP+FP+TN+FN)}$
Precision	$\frac{TP}{(TP+FP)}$
Recall = TPR	$\frac{TP}{(TP+FN)}$
F1 measure	$\frac{2*(Recall*Precision)}{(Recall+Precision)}$
Specificity = TNR	$\frac{TN}{(TN+FP)}$

3. RESULTS & DISCUSSION

We used the machine learning techniques to predict defects in software. In this study, we focused on automated fault recovery inside software through a predictive model, besides we also observed Requirement Specification Requirement Analysis Physical Design HLD (High Level Diagram) DLD (Detail Level Diagram) Deployment Software Defect Performance 3 Defect Datasets Split the datasets (Training 80% & Testing 20%) Approve the Outperform Model. Table 4 shows the performance evaluation of six supervised classification techniques for software fault prediction.

Table 4. Classification performance of ml techniques

Algorithms	Performance Measurement	Datasets		
		JM1	CM1	PC1
DT	Precision	1.0	1.0	.97
	Recall	0.99	1.0	1.0
	Accuracy	0.99	1.0	.99
	F1	0.99	1.0	.98
NB	Precision	0.94	1.0	.91
	Recall	0.93	1.0	.85
	Accuracy	0.98	1.0	.98
	F1	0.93	1.0	.87
RF	Precision	.99	1.0	.97
	Recall	1.0	1.0	.97
	Accuracy	0.99	1.0	.99
	F1	0.99	1.0	.97
KNN	Precision	0.95	.95	.86

With respect to the precision: DT and SVM achieved the highest performance (i.e. 100%) on JM1 datasets; DT, NB, SVM, and RF achieved the best performance on CM1 datasets, (it's respectively 100%); DT, SVM, and RF obtained the highest performance (i.e. 97%) on PC1 datasets. Relatively, all of the classifiers have shown good performance in terms of precision. However, considering the recall of the analysis, SVM and RF achieved the highest performance on JM1 datasets; LR and NB attained the lowest performance on CM1 and PC1 datasets. Not that all of the classifiers achieved very similar scores in terms of recall. Another measure for classification is F1 measure. With respect to F1 measure: SVM achieved the highest value (100%) on JM1 datasets and NB obtained the lowest score (93%). By Looking CM1 datasets, we can monitor that the f1 scores are mostly similar (NB, DT, SVM, RF = 100% and KNN = 97%, LR = 95%). Moreover, RF achieved the best score (i.e. 99%) and KNN performed lowest (86%) on PC1 datasets.

4. CONCLUSION

In this paper, we proposed an automated software engineering approach for defect prediction model development (SDPD) on software development life cycle. After that, the main objective of our study was to evaluate the abilities of 4 supervised based machine learning classifications techniques to predict the software defect modules using 3 NASA datasets (JM1, CM1, PC1). The results (i.e. accuracy: 90 - 98%) of the experiment with different attributes showed the capability and efficiency of SDPD model to identify the fault and improve software quality.

In addition, this SDPD model can be able to early detection of software faults by collecting real-time software development data from the target applications. The proposed approach can be used for software fault recovery inside a system and enhanced by applying machine learning techniques.

For future work, we will implement more classification algorithms, such as hybrid or ensemble model to verify the performance of the software fault prediction.

5. ACKNOWLEDGEMENTS

The authors are grateful to all the researchers in this research study.

6. REFERENCES

[1] Md. Razu Ahmed, Md. Asraf Ali, Md Fahad Bin Zamal, Nasim Ahmed, "The Impact of Software Fault Prediction in Real-World Application: An Automated Approach for Software Engineering".

[2] Bartłomiej Wójcicki, Robert Dąbrowski, "Applying Machine Learning to Software Fault Prediction, Institute of Informatics, University of Warsaw".

[3] Tim Menzies, Justin DiStefano, Andres Orrego, Robert (Mike) Chapman, "Assessing Predictors of Software Defects".

[4] Manzura Jorayeva, Akhan Akbulut, Cagatay Catal and Alok Mishra, “Machine Learning-Based Software Defect Prediction for Mobile Applications: A Systematic Literature Review”.

[5] Fatih Yucalara , Akin Ozcifta , Emin Borandaga, Deniz Kilinc, “Multiple-classifiers in software quality engineering: Combining predictors to improve software fault prediction ability”.

[6] Mitt Shah, Nandit Pujara, “Software Defects Prediction Using Machine Learning”.

[7] Revoori Veeharika Reddy, Nagella Kedharnath, Mandi Akif Hussain, S. Vidya, “Software Defect Estimation using Machine Learning Algorithms”.

[8] Jaswitha Abbineni, Ooha Thalluri, “Software Defect Detection Using Machine Learning Techniques”.

[9] Awni Hammouri, Mustafa Hammad, Mohammad Alnabhan, Fatima Alsarayrah, “Software Bug Prediction using Machine Learning Approach”.

[10] Marwa Assim, Qasem Obeidat, Mustafa Hammad, “Software Defects Prediction using Machine Learning Algorithms”.

[11] Burcu Yalçıner, Merve Özdeş, “Software Defect Estimation Using Machine Learning Algorithms”.

[12] Awni Hammouri, Mustafa Hammad, Mohammad M Alnabhan, “Software Bug Prediction using Machine Learning Approach”.

[13] M. Jureczko and L. Madeyski, “Towards identifying software project clusters with regard to defect prediction,” in Proceedings of the 6th International Conference on Predictive Models in Software Engineering - PROMISE ’10, 2010, p. 1.

[14] H. Tanwar and M. Kakkar, “A Review of Software Defect Prediction Models,” Springer, Singapore, 2019, pp. 89–97.

[15] R. Malhotra, “A systematic review of machine learning techniques for software fault prediction,” Appl. Soft Comput., vol. 27, pp. 504–518, Feb. 2015.