INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

Design of a Hybrid Machine Learning Base-Classifiers for Software Defect Prediction

Hassan Adam
Post Graduate Student, Department of Computer Engineering, University of Maiduguri, Nigeria
Abatcha Muhammad
Lecturer, Department of Computer Engineering, University of Maiduguri, Nigeria
Abdulfattah A. Aboaba
Lecturer, Department of Computer Engineering, University of Maiduguri, Nigeria

Abstract:
Machine learning (ML) classifiers have attracted the research community's attention in the field of software defect prediction (SDP) in the last decade. Several comparative studies confirmed that algorithms like Naïve Bayes, Decision tree, or Random Forest have been used with good performance. Recently, several hybrid classifiers were also introduced in SDP. However, many of those can only provide a category for a given new sample. Instead, this paper proposes a methodology to build a hybrid of four (4) ML Base-classifiers made up of Gaussian Naive Bayes, Bernoulli Naive Bayes, Random Forest, and support vector machine (SVM). The study carefully ensemble the selected machine learning models with efficient feature selection to address these issues and mitigate their effects on the defect prediction performance using dataset CM1 from PROMISE repository. The study outcome is expected to show promising SDP performance in terms of F1-score compared to the benchmark work.

Keywords: *Machine learning, software defect, Gaussian naive Bayes, Bernoulli naïve Bayes, random forest and support vector machine (SVM)*

1. Introduction

In the past years, studies have contributed more to software fault prediction issues like static code metrics, used support vector machine (SVM)(Gray et al., 2009), defect classification (Laradji et al., 2015)., and quality of software prediction (Prasad et al., 2015). This is understandable according to CIO magazine 'Faulty software costs businesses over $78 billion per year' as stated in CIO Magazine (2001) (CIO is a magazine related to technology and IT. The magazine was founded in 1987 and is now entirely digital. The name refers to the job title of chief information officer. CIO is part of Boston-based International Data Group's enterprise publications business.). Furthermore, National Institute of Standards and Technology (NIST) report (NIST, 2002) states that lacking testing methods and tools costs the US economy between $22.2 billion and $59.5 billion every year, with generally 50% of these expenses borne by software engineers, as additional testing, and half by software clients, as disappointment aversion and alleviation endeavors. The above reports show that 25% to 90% of software improvement financial plans are frequently spent on testing. Several researchers used Machine Learning ML to automate SDP. ML is the subset of artificial intelligence which allows machines to analyze data. Hence, hundreds of software fault detection researches have been done using ML models (Tóth et al., 2016). As a result, Durelli et al. (2019) combined ML models to achieve good defects. Ensemble learning algorithms combine the predictions of two or more ML models. The idea of combined learning is closely related to the idea of the '*wisdom of crowds*'. This is where many different independent decisions, choices, or estimates are combined into a final outcome that is often more accurate than any single contribution. The study presently plans to recognize the hybrid of ML classifiers to improve the performance of defect prediction models. One of the features of software quality datasets is that there is no definite agreement about which metric is superior for defect prediction. Most defect prediction studies tend to use a combination of available metrics(Wang et al., 2011), (Song et al., 2019). Therefore, the study hybrids four ML classifiers: Gaussian Naive Bayes, Bernoulli Naive Bayes, Random Forest, and support vector machine, and benchmarks the obtained results with the work of (Kaur, 2021).

2. Methodology

The proposed approach deployed a Hybrid of Gaussian Naive Bayes, Bernoulli Naive Bayes, Random Forest, and support vector machine. The study carefully combined the ML classifiers using Sci-Kit library in python, as the CM1 dataset goes well in python. The selected machine learning models with efficient feature selection address these issues and mitigate their effects on the defect classification performance. The details of each Hybrid member are clarified underneath in the summary as obtained from (Shitole & Priyadarshini, 2022).

Naïve Bayes (NB).

The formula for Bayes' theorem is given as:

$$p(A|B) = \frac{P(B|A)P(A)}{P(B)} \qquad (1.0)$$

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.

Bernoulli Naive Bayes: The choice rule for Bernoulli Naive Bayes relies upon:

$$p(x_i|y) = p(i|y)x_i + \big(1 - p(i|y)\big)(1 - x_i) \quad (1.1)$$

2.1. Gaussian Naive Bayes

One run-of-the-mill way for taking care of the ceaseless elements in the NB classification is that Gaussian appropriations are used for the meaning of the probabilities of the highlights adapted to the classes. In this manner, each characteristic is portrayed through a Gaussian density function (PDF) as:

$$x_i \sim N(\mu, \sigma^2) \qquad (1.2)$$

The Gaussian PDF has the shape of a bell and is defined using the following equation:

$$N(\mu, \sigma^2)(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e - \frac{(x-\mu)^2}{2\sigma^2} \qquad (1.3)$$

In which μ is the mean and $\boldsymbol{\sigma^2}$ is the variance. In Naïve Bayes, the parameters required are in the order of $O(n, k)$ in, which is the number of attributes and k represents the number of classes. In particular, there is a requirement of defining a normal distribution $p(x_i \backslash c \sim \boldsymbol{N(\mu, \sigma^2)}$ for every continuous attribute. The parameters of such normal distributions are acquired with:

$$\mu x_{i|c=c} = \frac{1}{N_c}\sum_{i=1}^{N_c} x_i \qquad (1.4) \qquad {}^{2}x_{i|c=c} = \frac{1}{N_c}\sum_{i=1}^{N_c} x_i{}^2 - \mu^2 \qquad (1.5)$$

σ

In this, N_c is the number of instances in which $C = c$, and N_c is the number of total instances utilized while training. The calculation of $P(C = c)$ for all the classes is easy with the help of relative frequencies such that:

$$P(C = c) = \frac{N_c}{N} \qquad (1.6)$$

2.2. Random Forest

Mathematically, a Random Forest is a predictor where the collection of randomized base regression trees is comprised as $\{r_n(x, \ominus_m, D_n), m \geq 1\}$, *where* $\ominus_1, \ominus_2 \ldots\ldots\ldots$ are independent and identically distributed outputs of a randomized variable $\ominus$. This integration of random forests is done to develop the aggregated regression estimate.

$$r_n(x, D_n) = E_{\ominus}[r_n(x, \ominus, D_n)], \qquad (2.0)$$

In which $E_{\ominus}$represents the expectation in terms of the random parameter, conditionally on x and the data set D_n. In the following, to lessen a notation a little, the dependency of the estimates would be excluded in the sample and written to illustrate $r_n(X)$ rather than $r_n(x, D_n)$. In practice, Monte Carlo computed the above expectation when the M RTs are produced, and the average of the individual outcomes is taken. The randomizing variable $\ominus$ is carried out to determine the performance of the successive cuts while developing the individual trees in which the selection of the coordinate to split and position of the split are comprised. The variable $\ominus$ is deduced as independent of X and the training sample D_n.

2.3. Support Vector Machine

A support vector machine is a supervised machine learning algorithm mainly used to classify data. SVM can be used for face recognition, colon cancer classification, etc. SVM uses a hyperplane, also called a decision boundary, to divide data into various segments. The Support Vector Regressor (SVR) is used for regression-related problems. SVM can be performed on a non-linear dataset by using the kernel function.

2.4. Design

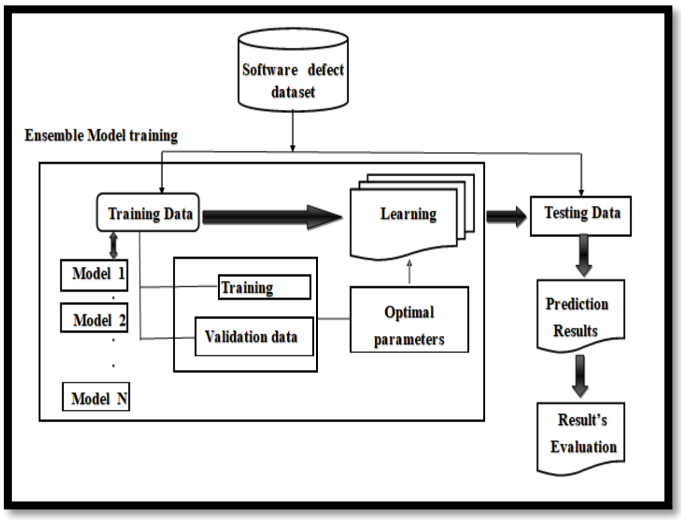

Figure 1: Hybrid Model for SDP Design Process

The design process in figure 1 shows an internal understanding of how the process plays out diagrammatically, while, Appendix I shows simple programming steps of the process.

3. Implementation

In this section, the outcomes of the developed system are discussed with the comparison of existing models for software defect prediction. The detail about the dataset and the parameters used for the performance analysis is discussed in detail.

The result of the Hybrid model implemented is featured in terms of accuracy, precision, recall, and F1-Score in tables.

3.1. Data Set

CM1 is a National Aeronautics and Space Administration (NASA) spacecraft instrument (data collection and processing) written in 'C'. At various times, researchers have negotiated access to the CM source code. Hence, it is somewhat more studied, thus, some of the other NASA data sets. CM1 publicly available data sets from PROMISE are used for this research, the repository commonly used by researchers in software defect prediction. CM1 contains 498 records, and 22 ascribes (5 unique lines of code measure, 3 McCabe metrics, 4 base Halstead measures, 8 determined Halstead measures, a branch count, furthermore 1 objective field). This dataset is picked in the work since it comes from a genuine source and is uninhabited.

4. Testing

Binary classification is a two-class prediction problem in which the outcomes are labeled as either positive or negative. For defect prediction, binary classification involves mapping all instances of a dataset containing defective and non-defective modules to defective and non-defective classes. There are four outcomes possible from this binary classification. Therefore, one must use a confusion matrix. Confusion matrices are used to visualize important predictive analytics like recall, specificity, accuracy, and precision. Confusion matrices are useful because they directly compare values like True Positives, False Positives, True Negatives, and False Negatives. If a module is defective and classified as defective, then it is called true positive (TP), but if it is non-defective but classified as defective, it is called false positive (FP). Similarly, if a module is non-defective and classified as non-defective, it is called true negative (TN). However, if a defective module is classified as non-defective, it is called false negative (FN).

Predicted Result	Actual Result		
		0	1
	0	TN	FP
	1	FN	TP

Table 1: Confusion Matrix of a Defect Prediction Classification

Accuracy: Accuracy is defined as the percentage of correct predictions out of all the observations.

$$Accuracy = \frac{Correct\ prediction}{Total\ cases} X\ 100\% \quad => \ Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} X\ 100\% \qquad (3.0)$$

Precision: Precision is defined as the percentage of true positive cases versus all the cases where the prediction is true.

$$Precision = \frac{True\ Positive}{All\ Predicted\ Positive} X\ 100\% \quad => precision = \frac{TP}{TP+FP}\ X\ 100\% \qquad (3.1)$$

Recall: It is defined as the fraction of positive cases that are correctly identified.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} X\ 100\% \Rightarrow Recall = \frac{TP}{TP+FN}\ X\ 100\% \qquad (3.2)$$

F1 Score: F1 score is defined as the measure of the balance between precision and recall.

$$F1\ Score = 2\ X \frac{Precision\ X\ Recall}{Precision + Recall} \qquad (3.3)$$

5. Results

The outcome of the hybrid ML classifiers is given in the table below.

Model	Accuracy %	Precision %	Recall %	F1-Score
(Kaur, 2021)	96.67	93	97	94.96
Hybrid ML	90.00	94	96	94.99

Table 2: Hybrid Classifier and (Kaur, 2021) Results

6. Discussion

The results presented confirm that SVM performs better than Decision Tree (C4.5) classifier in the Hybrid on F1-Score as it is required in confirming the good results of SVM for binary classification. It is possible to use CM1 dataset better for the hybrid. The Hybrid built also presents a superior performance to the base classifiers. This methodology presents an advantage that differentiates it from single ML classifier methods by combining more classifiers.

The proposed Model shows improvement in F1 Score compared to the existing ensemble software defect prediction.

7. Conclusion

Software defect, alongside an inherent component of a software item, is additionally a significant part of software quality. Software defects are an unavoidable co-item of the created software. What is more, the assurance of software quality affirmation is not that simple and requires much time. There are various approaches to characterize defects, for example, quality. Nonetheless, the defects are by and large characterized as deviations from determinations or assumptions, which might be the reason for disappointment in working. In this study, different ML classifiers like Gaussian Naive Bayes, Bernoulli Naive Bayes, Random Forest, and SVM are implemented in python for software defect prediction. The Hybrid classifier is created for the SDP, which is the blend of Gaussian Naïve Bayes, Bernoulli Naive Bayes, Random Forest, and SVM as Hybrid ML. As a result, it compares the work of Kaur (2021) and the proposed study considering F1-Score, and then, the proposed study is improved by 0.03, which shows an encouraging result.

8. Recommendation

This study revealed the effectiveness of ML hybrids in software fault detections. Thus, the following recommendations are hereby presented:

- Since defect detection has been proven, software developers should incorporate it in Software development Life cycle (SDLC) to maintain quality software for users and help them build a relational understanding of software testing/control.
- Implementation of software testing in python be encouraged by administrators and embraced by software developers to continually improve software quality.

9. References

i. Gray, D., Bowes, D., Davey, N., Sun, Y., & Christianson, B. (2009). Using the support vector machine as a classification method for software defect prediction with static code metrics. Communications in Computer and Information Science, 43 CCIS(May 2014), 223–234. https://doi.org/10.1007/978-3-642-03969-0_21.

ii. Kaur, B. (2021). An Ensemble Classification Model For Software Defect Prediction. 8(7), 33–40.

iii. Laradji, I. H., Alshayeb, M., & Ghouti, L. (2015). Software defect prediction using ensemble learning on selected features. Information and Software Technology, 58, 388–402. https://doi.org/10.1016/j.infsof.2014.07.005.

iv. Prasad, M. C. M., Florence, L. F., & Arya3, A. (2015). A Study on Software Metrics based Software Defect Prediction using Data Mining and Machine Learning Techniques. International Journal of Database Theory and Application, 8(3), 179–190. https://doi.org/10.14257/ijdta.2015.8.3.15.

v. Shitole, A. S., & Priyadarshini, I. (2022). Survey of Machine Learning Algorithms & its Applications. January, 0–5. https://doi.org/10.5281/zenodo.5090570.

vi. Song, Q., Guo, Y., & Shepperd, M. (2019). A Comprehensive Investigation of the Role of Imbalanced Learning for Software Defect Prediction. IEEE Transactions on Software Engineering, 45(12), 1253–1269. https://doi.org/10.1109/TSE.2018.2836442.

vii. Tóth, Z., Gyimesi, P., & Ferenc, R. (2016). A public bug database of GitHub projects and their application in bug prediction. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9789, 625–638. https://doi.org/10.1007/978-3-319-42089-9_44.

viii. Wang, H., Khoshgoftaar, T. M., Van Hulse, J., & Gao, K. (2011). Metric selection for software defect prediction. International Journal of Software Engineering and Knowledge Engineering, 21(2), 237–257. https://doi.org/10.1142/S0218194011005256.