SOFTWARE FAULT PREDICTION USING MACHINE LEARNING

Yasaswini Kandru, Naveen Syamala
Department of Computer Science
Georgia State University
ykandru1@student.gsu.edu, nsyamala1@student.gsu.edu

Abstract: Detecting faults early in the development process saves time and money. Predicting software defects is an important element of the software development process since it can improve software quality, performance, and maintenance costs. Defective software has a direct impact on software quality, resulting in increased software expenses, project delays, and maintenance expenditures. Applying a prediction model to the software before deployment, on the other hand, enhances software performance, reliability, and user happiness. The software prediction performance was assessed using eight machine learning (ML) algorithms. The algorithms include artificial neural networks (ANNs), random forest (RF), random tree (RT), decision table (DT), linear regression (LR), gaussian processes (GP), SMOreg, and M5P. Weka 3.8.3 tool has been used to run the algorithms using the k-fold cross-validation and percentage split techniques.

Keywords: Software Faults, Software Fault Prediction, Machine Learning, Weka tool, SMOreg classifier

I.INTRODUCTION

Dealing with defects is a major issue in the software development process. The presence of various defects entails a high level of risk, which might lead to project failure. Furthermore, it is a factor in lowering project quality and raising project maintenance costs. Detecting faults early in the development process saves time and money. Predicting software defects is an important element of the software development process since it can improve software quality, performance, and maintenance costs. Defective software has a direct impact on software quality, resulting in increased software expenses, project delays, and maintenance expenditures. Applying a prediction model to the software before deployment, on the other hand, enhances software performance, reliability, and user happiness. Developers have additional opportunities to focus on faulty modules and successfully resolve them with fault prediction activities. During the growth stage, defect prediction models are extensively used by industries. These models aid in forecasting defects, estimating effort, testing software dependability, hazard analysis, and other tasks.

There are two types of machine learning techniques: supervised learning and unsupervised learning. A supervised machine learning predictive algorithm is consumed with the predefined collection of training data. After learning from the training dataset, the algorithm generates rules for predicting the class label or several defects for a testing data set. The predictor function is generated and strengthened using mathematical techniques in the learning phases. This procedure uses training data with an attribute input value and a defined output value. The quality of the expected ML algorithm is compared to the well-known result. This is repeated numerous times with different sets of training data until the best prediction accuracy is achieved or the maximum number of loops is reached. The class label output value is unknown in data in the realm of unsupervised learning methods. Alternatively, the software is loaded with a cluster of data, and the algorithm finds a pattern and relationships within it. The main emphasis is on the Relationship among data attributes.

Predicting faulty modules helps improve software quality. Defect prediction is a method of developing models that are used early in the process to identify problematic systems, such as units or classes. This is accomplished by categorizing modules as defect-prone or not. Support vector classifier (SVC), random forest, naive Bayes, decision trees (DT), and neural networks are some of the most frequent approaches for identifying the classification module (NN). In the progress testing phases, the found defect-prone modules are given top priority, while the non-defect-prone modules are investigated as time and expense allow.

The classifier method is used in the ML classification models to find the relationship between the attributes and the training dataset class label, and the equations are used to analyze the categorization of the targets. These rules will also be required to determine the labels for future dataset classes. Thus, utilizing classification patterns and a classifier, unknown datasets can be classified. The ML Regression model aids in the discovery of variable correlations and allows us to forecast a continuous output variable based on one or more predictor variables. Regression analysis, in

particular, allows us to see how the value of the dependent variable changes about an independent variable while the other independent variables are maintained constant. This method is commonly utilized. The classifier method is used in the ML classification models to find the relationship between the attributes and the training dataset class label, and the equations are used to analyze the categorization of the targets. These rules will also be required to determine the labels for future dataset classes. Thus, utilizing classification patterns and a classifier, unknown datasets can be classified. The ML Regression model aids in the discovery of variable correlations and allows us to forecast a continuous output variable based on one or more predictor variables. Regression analysis, in particular, allows us to see how the value of the dependent variable changes about an independent variable while the other independent variables are maintained constant. This method is commonly utilized.

Because of the huge deployment of software, defining and discovering software faults is repeated work for researchers. The basic purpose of categorizing the software dataset as a model for bug prediction is to forecast the number of defects in a given dataset by categorizing it into defective and non-defective datasets.

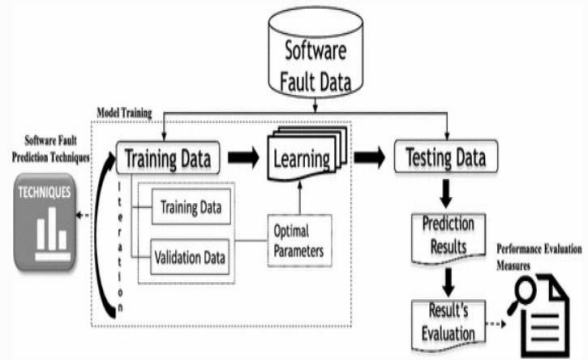

Fig 1. Software Fault Prediction model

The following are the steps involved in predicting faults using the machine learning model:

- Divide the dataset into training and testing datasets.
- Feed the training data as input for the ML model.
- The model learns the relation between input features and tries to predict the target value. It develops a predicting function.
- Use the testing data and try to predict the target value.
- Calculate the evaluation metrics for the model.

The other sections of the paper are structured as follows: the literature review related to software prediction techniques is presented in Section II; background knowledge and description of the used Machine Learning algorithms are discussed in Section III; the dataset with evaluation methodology and experimental results are presented in Sections IV and V, respectively; the conclusion is discussed in Section VI.

II. LITERATURE REVIEW

Static code metrics and process metrics are two types of software metrics in fault prediction. Lines of Code (LOC), Cyclomatic Complexity Number (CCN), and other static code metrics, such as compiler instruction and data declaration counts, can be retrieved directly from the source code. Object-oriented metrics, such as Depth of Inheritance Tree (DIT), Coupling Between Objects (CBO), Number of Children (NOC), and Response for Class, are a subgroup of static code metrics (RFC). Import coupling measures are significantly associated with fault proneness, according to Aggarwal et al [7], and can accurately identify problematic classes in object-oriented systems.
Based on historical changes to source code over time, process metrics can be retrieved from a Source Code Management system. Object-oriented metrics are frequently used to evaluate source code's testability, maintainability, and reusability. Object-oriented metrics have recently been used to anticipate faults. Many object-oriented metrics are studied and proposed to access the quality of software.

In [1], the RAPIDMINER machine learning tool is used to implement the random forest (RF) algorithm. The performance evaluation of different numbers of trees in RF will be compared in this research. As a result, if the number of trees is increased, the accuracy will be slightly improved. The greatest accuracy is 99.59 percent, while the minimum accuracy is 85.96 percent. Another comparison is based on the AUC curve, which in the field of software defect prediction is the most informative metric of predictive accuracy. All of the data suggest that the RF method is effective in this prediction and that the use of hundred trees in the RF is more appropriate.

Dataset	10 Trees	100 Trees	1000 Trees
ar1	0.803	0.958	0.969
ar3	0.980	0.989	0.989
ar4	0.958	0.970	0.972
ar5	0.991	0.996	0.996
ar6	0.917	0.907	0.961

kc1	0.623	0.792	0.821
kc2	0.856	0.896	0.904
kc3	0.770	0.911	0.920
pc1	0.652	0.659	0.668
pc2	0.565	0.629	0.714
pc3	0.629	0.851	0.854
pc4	0.630	0.884	0.919

Table I. Accuracy of SFP model using RF algorithm

In [2], the author looked into various Machine Learning models to see which one was the most effective in terms of performance and accuracy. It primarily examines the performance of machine learning algorithms like as the Naive Bayes Decision Tree, Random Forest, Logistic Regression, K-nearest neighbors, Multilayer Perceptron, and SVM in defect prediction on both class-level and method-level datasets from PROMISE Java projects. In this study, thirteen datasets were used to predict fault, including JM1, PC4, KC2, MC1, KC1, PC3, CM1, MW1, PC1, Class, MC2, KC3, and PC2. In addition, the effectiveness of these strategies is calculated in terms of F1, AUC, and Accuracy. The Receiver Operating Characteristic (ROC) is used to examine the performance of the approaches to ensure that the results are reliable.

Algorithm\Metric	F1	AUC	Accuracy
Logistic Regression	0.34	0.68	0.64
K-nearest Neighbors	0.43	0.55	0.52
Decision Tree	0.41	0.56	0.55
Random Forest	0.43	0.61	0.56
Naïve Bayes	0.32	0.61	0.65
SVM	**0.48**	0.58	**0.63**
Multilayer Perceptron	0.44	0.62	0.58

Table II. Class Level Fault Prediction Result

Experimental results show that among traditional techniques, Support Vector Machine achieves high accuracy and the F1 value outperforms that of other techniques at class-level.

Algorithm\Metric	F1	AUC	Accuracy
Logistic Regression	0.45	**0.90**	**0.91**
K-nearest Neighbors	0.49	0.71	0.87
Decision Tree	0.42	0.72	0.86
Random Forest	0.47	0.88	0.9
Naïve Bayes	0.24	0.85	0.88
SVM	0.32	0.66	0.9
Multilayer Perceptron	**0.59**	0.91	0.91

Table III. Method Level Fault Prediction Result

Multilayer Perceptron beats all other techniques in terms of accuracy, F1, and AUC for method-level datasets. As a result, Multilayer Perception is the best method for predicting faults at the method level, whereas Support Vector Machine is best for predicting mistakes at the class level. Support Vector Machine outperforms other techniques for class-level datasets, while Multilayer Perceptron beats other techniques for method-level datasets, according to the findings.

In [3], a hybridized technique employing the PCA and Random Forest, nave Bayes, SVM is used to analyze five datasets, including PC3, MW1, KC1, PC4, and CM1. Confusion, precision, recall, identification accuracy, and other factors are examined and compared to existing systems in a systematic study investigation. The suggested hybrid technique appears to provide more relevant solutions for device defect prediction, according to the analytical investigation.

Dataset	SVM	Random Forest	Naïve Bayes
KC1	1.00	0.99	0.96
CM1	0.99	0.98	0.97
PC4	0.96	0.98	1.00
MW1	1.00	0.97	0.99
PC3	0.97	1.00	0.98

Table IV. Accuracy results for a hybrid approach

PCA is used to obtain a feature reduction template, and overall probability is used to reduce the amount of data retrieved by PCA. In addition, program faults are detected using the neural network classification method. According to the research, the proposed method is efficient and produces an AUC of 98.70 percent, which is a significant improvement over certain state-of-the-art models. To demonstrate the significance of the feature selection effect empirically. When there is no methodology for selecting features and when the strategies for selecting features are used, there is no noticeable variance in classifier accuracy. When there is no collection of features and the methods for picking the function is utilized, there is a gap in classifier accuracy. As a result, it is noticed that employing features reduces the period and space difficulty of defect prediction without decreasing prediction accuracy by using feature selection techniques.

III. METHODOLOGY

A. SOFTWARE METRICS FOR FAULT PREDICTION

A publicly available dataset from the PROMISE repository, notably class-level data for KC1 (a NASA

product), was used in this analysis. NUMDEFECTS is the last numeric attribute, and it reflects the total number of class defects recorded. It was employed as a response variable in the prediction process by machine learning algorithms. The dataset provides genuine measured data for one of NASA's products for KC1 and has 145 instances. The first ten class-level attributes, as well as the NUMDEFECT attribute, are represented in Table 5.

No	Attribute	Description
1	PERCENT_PUB_DATA	The percentage of data that is public and protected data in a class.
2	ACCESS_TO_PUB_DATA	The number of times that a class's public and protected data is accessed.
3	COUPLING_BETWEEN_OBJECTS	The number of distinct non-inheritance-related classes on which a class depends.
4	DEPTH	The level for a class.
5	LACK_OF_COHESION_OF_METHODS	The percentage of the methods in the class.
6	NUM_OF_CHILDREN	The number of classes derived from a specified class.
7	DEP_ON_CHILD	Whether a class is dependent on a descendant.
8	FAN_IN	Count of calls by higher modules.
9	RESPONSE_FOR_CLASS	Count of methods implemented within a class.
10	WEIGHTED_METHODS_PER_CLASS	Count of methods implemented within a class.
11	NUMDEFECTS	No. of Defects

Table V. Numeric attribute description of KC1 dataset

The software prediction performance was assessed using eight machine learning (ML) algorithms. The algorithms include artificial neural networks (ANNs), random forest (RF), random tree (RT), decision table (DT), linear regression (LR), gaussian processes (GP), SMOreg, and M5P. Weka 3.8.3 tool has been used to run the algorithms using the k-fold cross-validation and percentage split techniques. Dataset used in this work was obtained from a publicly available repository from NASA Promise. The data set was validated using two test modes: 10-fold cross-validation and percentage split with different training set percentages (60%, 70%, 80%, 90%). Using the WEKA tool, the two test modes applied the data classification on the eight ML algorithms.

The concept and research approach for software defect prediction is depicted in Figure 2. For the prediction and analysis, the KC1 dataset was used. The dataset was trained and tested using a 10-fold cross-validation method. The classifiers' output is then assessed using various error measurements and metrics.

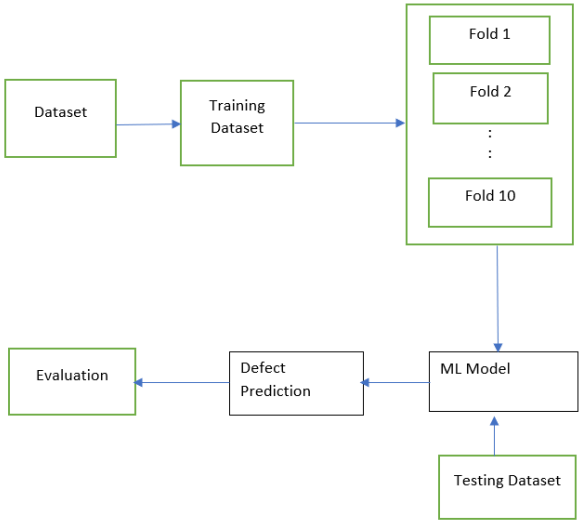

Fig. 1. Software Defect Prediction Model

B. USED MACHINE LEARNING ALGORITHMS

Machine learning (ML) uses multiple algorithms to analyze input data and predict output values. Machine learning algorithms learn and improve performance and help with automating the labeling process. The performance of eight ML algorithms was measured to predict software defects. These ML algorithms are summarized as the following:

Artificial Neural Networks (ANNs):

ANNs are complex systems that mimic the operation of the nervous system in humans. They are made out of artificial neurons and can be used as a non-linear classifier in machine learning systems, receiving numerous inputs and providing a single output. The input neurons are connected and work in parallel by

delivering a signal to other neurons and calculating the output using a non-linear function.

Random Forest (RF):

The RF method is a learning technique for supervised classification and regression. It generates a forest of decision trees and uses committee voting rather than individual tree decisions to determine the optimal solution.

Random Tree (RT):

A Random Tree (RT) is a graphical depiction of all possible solutions for a given set of circumstances. It's called a tree because it starts with a root and branches out to provide several sets of data to form a decision tree.

Decision Table (DT):

DT is a well-organized model for creating requirements based on business rules. It's a type of classifier that can be used to simulate complex reasoning. True (T) or false (F) conditions are labeled in a DT (F). Each column in the table contains a business logic rule that describes a specific set of instances.

Linear Regression (LR):

The LR algorithm is a supervised learning method. It estimates the outcome by establishing a linear regression relationship between an independent variable (x) and the value of a dependent variable (y).

Gaussian Processes (GP):

GP is a probability distribution of possible outcomes. It is a collection of random variables represented by time or space, where each group of random variables has a multivariate distribution.

SMOreg:

SMOreg uses a vector machine for regression and a variety of parameter learning algorithms. The method fills in the blanks and converts nominal properties to binary attributes.

M5P:

M5P is a variant of the M5 technique that can generate regression model trees. At the nodes, M5P uses both traditional decision trees and linear regression functions.

IV. EVALUATION MEASURES

Different evaluation measures will be used to assess software defect prediction accuracy. According to the results of these measures, we can decide which ML algorithms we can use for software defect prediction. The evaluation metrics include correlation coefficient (R^2), mean absolute error (MAE), root mean squared error (RMSE), relative absolute error (RAE), and root-relative squared error (RRSE). The ML algorithm with the lowest error rates can be considered the most appropriate algorithm to apply in the prediction process. The error rate between the number of actual faults and the number of anticipated defects in the dataset is measured by these measures. Assume, E is the number of real faults, $\tilde{E}$ denotes the number of expected defects, $\bar{E}$ denotes the mean of E, and n denotes the number of instances. The formulas below represent the performance metrics that were employed in the prediction procedure:

1. $R^2 = 1 - \frac{\sum_{i=1}^{n}(Ei - \bar{E}i)^2}{\sum_{i=1}^{n}(Ei - \bar{E}i)^2}$

2. $MAE = \frac{1}{n}\sum_{i=1}^{n} |Ei - \tilde{E}i|$

3. $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Ei - \tilde{E}i)^2}$

4. $RAE = \frac{\sum_{i=1}^{n} |Ai - \tilde{A}i|}{\sum_{i=1}^{n} |Ai - \bar{A}i|}$

5. $RRSE = \sqrt{\frac{\sum_{i=1}^{n}(Ai - \tilde{A}i)^2}{\sum_{i=1}^{n}(Ai - \bar{A}i)^2}}$

The R2 coefficient indicates how closely the real and projected values are connected. These numbers range from 1 to 1, with 1 indicating a correct positive linear relationship, 0 indicating no relationship, and 1 indicating a correct negative linear relationship. The average difference between the actual and projected values is known as the MAE. RMSE is similar to MAE, except that it represents the square of the difference between the actual and expected values. As a result, the attention will be on more serious errors rather than minor ones [32]. The error is the entire absolute error because RAE offers the average of the actual values. Because it is the average of the actual values, the RRSE is comparable to the RAE, but the error is the total squared error.

V. EXPERIMENT RESULTS

The LR technique has the highest correlation coefficient (R2) value in the KC1 dataset, as shown in Table 5, followed by the SMOreg algorithm. The SMOreg method, on the other hand, had the lowest error rates for the remaining metrics. As a consequence, among the nine ML algorithms, the SMOreg classifier had the best performance results. The ANN algorithm, on the other hand, had the worst performance outcomes.

Furthermore, the error rates for both RAE and RRSE exceeded 100%, indicating that the algorithm was unable to predict legitimate values because the values were constant across all input data.

classifier	R^2	MAE	RMSE	RAE	RRSE
ANN	0.16	6.98	14.72	115.67	134.89
Random Forest	0.59	4.42	9.39	73.18	86.00
Random Tree	0.13	5.50	12.20	91.20	111.78
Decision Table	0.26	4.92	11.22	81.50	102.84
Linear Regression	0.75	6.58	11.17	109.08	102.35
Gaussian Process	0.67	4.63	7.94	76.70	72.75
SMOreg	0.73	4.31	7.43	73.10	68.11
M5P	0.69	4.36	7.78	72.21	71.28

Table VI. Results for The ML Algorithms Using 10 Folds Cross-Validation Testing Mode

Classifier	60%	70%	80%	90%
ANN	6.35	5.09	5.74	5.74
Random Forest	3.99	3.60	1.94	1.94
Random Tree	4.36	4.29	1.30	1.30
Decision Table	6.28	5.56	2.56	2.56
Linear Regression	5.94	5.52	4.24	4.24
Gaussian Process	5.14	5.09	3.40	3.40
SMOreg	4.43	3.95	2.73	2.73
M5P	4.08	5.99	3.02	3.02

Table VII. Results For ML Algorithms Using Percentage Split Testing Mode

Scored results of the eight ML algorithms using percentage split testing mode are shown in Table III. Different percentages for the training and assessment sets were used, ranging from 60% to 90%. The MAE statistical metric was employed as a performance evaluation metric for the assessment. We may deduce that the error rate will decrease when the percentage set is increased. For both the 60% and 70%training sets, the random forest (RF) method had the lowest MAE error rate. The random tree (RT) method, on the other hand, had the lowest MAE error rate in both the 80 percent and 90 percent training sets.

VI. CONCLUSION

Because software must evolve to meet changing consumer and market demands, it is modified over time. Many software flaws can occur during software development. To detect these new faults, the software developer should update the test suites, for as by adding new test cases. Software defect prediction is an important technique that the software developer should think about before releasing the software. A machine learning-based prediction model can be used to predict software defects. The approach can forecast future software problems using previous data. Various machine learning techniques were used to assess the accuracy of software fault prediction in this work. ANN, RF, RT, DT, LR, GP, SMOreg, and M5P were among the eight ML methods utilized. Different statistical indicators were utilized to analyze the error rates of the anticipated values using a publicly available dataset from the NASA promise repository, where the algorithms were applied.

R2, MAE, RMSE, RAE, and RRSE are the measures. We can deduce from the findings that machine learning algorithms are effective tools for forecasting future software flaws. To begin, the findings were measured using a 10-fold cross-validation testing mode. Among the other methods, the SMOreg algorithm, which uses a vector machine classification for regression and replacing missing values, as well as normalizing all characteristics by default, produced the best results. The RT algorithm, on the other hand, achieved the lowest MAE error rate utilizing percentage split testing mode for both (80% and 90%) training sets.

VII. REFERENCES

[1] Y. N. Soe, P. I. Santosa and R. Hartanto, "Software Defect Prediction Using Random Forest Algorithm," *2018 12th South East Asian Technical University Consortium (SEATUC)*, 2018, pp. 1-5, doi: 10.1109/SEATUC.2018.8788881.

[2] T. M. Phuong Ha, D. Hung Tran, L. T. My Hanh and N. Thanh Binh, "Experimental Study on Software Fault Prediction Using Machine Learning Model," 2019 11th International Conference on Knowledge and Systems Engineering (KSE), 2019, pp. 1-5, doi: 10.1109/KSE.2019.8919429.

[3] C. L. Prabha and N. Shivakumar, "Software Defect Prediction Using Machine Learning Techniques,"

2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), 2020, pp. 728-733, doi: 10.1109/ICOEI48184.2020.9142909.

[4] M. Assim, Q. Obeidat and M. Hammad, "Software Defects Prediction using Machine Learning Algorithms," 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI), 2020, pp. 1-6, doi: 10.1109/ICDABI51230.2020.9325677.

[5] D. R. Ibrahim, R. Ghnemat and A. Hudaib, "Software Defect Prediction using Feature Selection and Random
Forest Algorithm," 2017 International Conference on New Trends in Computing Sciences (ICTCS), 2017, pp.
252-257, doi: 10.1109/ICTCS.2017.39.