Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 32 No. 2 (2025)

Improving Software Defect Prediction Accuracy through Modified Entropy Calculation in the Random Forest Algorithm

*** Ranjeetsingh Suryawanshi[1], Amol Kadam[2]**
[*1]ranjeetsinghsuryawanshi@gmail.com
[2]akkadam@bvucoep.edu.in
[1,2] Bharati Vidyapeeth Deemed to Be University, College of Engineering, India
*****Correspondence: ranjeetsinghsuryawanshi@gmail.com

Article History:

Received: *05-08-2024*

Revised: *12-09-2024*

Accepted: *21-09-2024*

Abstract:

Assume the scenario in which you are attempting to categorize software defects for a broad dataset. Which algorithm do you think will be the most effective for accomplishing that? Random Forest, Support Vector Machine, Neural Networks, Naive Bayes, K-Nearest Neighbours, Decision Tree, Logistic Regression, and other techniques are among those that can be utilized to solve the problem described above. The Random Forest technique, which allows for the generation of predictions through the utilization of many Decision Trees, is one of the most often utilized methods. Entropy, a complicated computation that examines the degree of uncertainty in the data, is the foundation upon which this algorithm is built. It is possible that the calculation of entropy, which is a function that uses natural logarithm, will take a significant amount of time. Does one know of a more accurate method for calculating entropy? The Taylor series expression was utilized in this investigation to investigate a different approach to calculating the natural logarithm. Any function may be approximated by utilizing its derivatives, and this series, which is made up of the sum of infinite terms, is what it is. Also, we updated the Random Forest algorithm by substituting the natural logarithm with the Taylor series equation in the Entropy calculation. This was done in order to achieve these modifications. Following the implementation of our improved algorithm on the dataset, we examined its performance in comparison to the Entropy formula that was first developed. An improvement in the algorithm's accuracy in predicting software defects was discovered by us as a result of our update to the technique.

Keywords: Random forest; decision tree; classification; prediction; entropy; Taylor series;

1. Introduction

There are bugs, glitches, or faults in software that can lead to inaccurate outcomes. These are referred to as software defects. As a result of the fact that people are responsible for a significant number of tasks during the creation of software, numerous software fault problems may arise. Some of the stages

that are involved in the process of building software, such as requirement analysis, design, coding, testing, deployment, and maintenance, are susceptible to discovering defects. In order to ensure that software products continue to be of high quality and reliability, rigorous software testing is essential. In order to keep up with the rapid growth of software applications, additional testing is required, which is both costly and time intensive. When compared to software testing and evaluations, software defect prediction approaches are significantly more cost-effective when it comes to the detection of software defects. Machine learning is a process that uses a number of methods to analyze input data and make predictions about output values. In addition to contributing to the automation of the labeling process, machine learning techniques contribute to the learning and improvement of performance.

The contributions of a large number of academics and academicians have demonstrated that the likelihood of identifying software defect prediction models may be more cost effective than the probability of identifying defects through software inspections. A subject of study that is still in progress, Software Defect Prediction is concerned with the discovery of software flaws through the application of algorithmic and data-level methodologies. Mahesh Kumar et al. provide an explanation of the characteristics of each technique as well as their applicability to the phenomenon of programming defect prediction throughout the various phases of the programming development life cycle. Expert opinions are the quickest and easiest way to receive software inspection; nevertheless, there is a large degree of prediction uncertainty, and the prediction may be impacted by personal biases. Despite this factor, expert opinions are the most straightforward technique. However, when dealing with huge volumes of data, models that are based on machine learning improve the accuracy of their predictions by altering the values of their parameters [1].Through the utilization of ensemble learning, it is possible to improve the performance of the model or reduce the risk of selecting an inaccurate model. When it comes to ensemble learning, there are three fundamental types: bagging, boosting, and stacking. It is possible to generate statistical distributions and confidence intervals with the use of bagging. By use boosting, bias can be reduced to a minimum. Through stacking, the most effective method of integrating machine learning models can be identified [2].

2. Related Work

When it comes to constructing prediction models, there are a variety of data level and algorithm approaches that may be utilized. Some examples of these approaches are data balance, feature selection, and machine learning algorithms. There are numerous academics and academicians who have made contributions to the field of defect prediction using both an empirical and conceptual approach. These contributions are described in the literature. The contributions of many researchers to the prediction of software defects are presented in Table 1.

Table 1. Summary of related works

Classifier used	Dataset used	Evaluation Measures	Future Work/Scope
SVM [3]	CM1, PC1, JM1, PC3, KC1, EQ and JDT	Precision, Recall, F1-score, and Accuracy	Model can be evaluated on different datasets to ensure its performance.
NB, J48, RF, SVM, AdaBoost [4]	Eclipse, Columba, and Scarab	F1-measure and ROC	Feature selection can be tested with deep learning
AdaBoost, KNN, LR, NB and RF [5]	PROMISE repository	Accuracy, F-Measure, Specificity, G Mean	can be extended to predict cross-project defects.
KNN, NB, SVM [6]	PROMISE repository	ROC curve	Can be tested on large datasets.
SVM, NB, RF [7]	CM1, KC1, MC1, PC1, JM1, MW1, PC2,	MCC, ROC, PRC, F1-score	Model can be build using deep learning

When it comes to improving the performance of defect prediction models, feature selection and data balancing are two crucial factors to account for. In the event where the inaccuracy increases as a result of changing feature values, the software feature turns into an important factor for the prediction. Robust machine learning models are built by means of the complicated interaction between software attributes, and Shapley values can be utilized in order to compute the degree of complexity that exists between these attributes [8]. It is possible to improve the quality of software defect prediction by including the best selected features into the Random Forest classification process. This will result in more accurate results than would otherwise be possible [9]. Using heterogeneous attribute selection and hierarchical stacking, the author presents a paradigm for forecasting software faults. This paradigm is founded on the idea that software might have errors. In order to improve the performance of the model Nested-Stacking involves the utilization of heterogeneous attribute selection methods in conjunction with normalization in order to enhance the quality of the data [10]. Using a decision tree induction, Gayatri and colleagues are able to determine which traits are appropriate. Each and every characteristic that was discovered through the application of the Decision Tree Induction Rule is included in the subset of attributes. The performance of the classifier is improved when it is taught this new feature set by utilizing the same models [11]. In their work, Pham et al. make use of probabilistic categorization, which identifies the class value that maximizes the posterior probability of the class for a particular set

of attributes [12]. In their article, Chennappan et al. discuss data balancing, which might lead to biassed results for a significant number of classes if the data are not balanced or balanced [13]

The prediction of defects appears to be a significant difficulty when dealing with vast volumes of data; hence, numerous machine learning techniques have been utilized in order to construct prediction models.

The categorization process is carried out by Random Forest through the utilization of many decision trees that collaborate to build a decision forest. With a relatively high level of accuracy, it is able to manage enormous datasets and inputs that contain multiple variables; it is particularly effective at regulating imbalanced datasets. Random forest is a technique that improves the interaction between decision trees [14]. Multiple decision trees cast their votes for the outcomes that better suit their preferences. Through the use of the Random Forests methodology, a limited number of rows are chosen at random from the overall quantity of data. Subsequently, a collection of decision trees is formed for each module in order to generate classification output [15]. A novel categorization and prediction strategy for enhancing accuracy is proposed by Premalatha et al. [16]. This technique is based on the Cost Random Forest algorithm, which significantly reduces the impact of errors that occur in software components that are not relevant to the problem at hand.

[17] A Bayesian network is a probabilistic visual model that displays a joint distribution of probabilities across a set of independent random variables. Bayesian networks are used to analyze Bayesian networks. It is possible for Bayesian networks to handle uncertainty in an effective manner. Through the utilization of conditional probability, a Bayesian network is able to effectively express the relationship between information components and acquire knowledge from data that is unclear [14].Through the utilization of data on boundary points of the hyperplane, the Support Vector Machine (SVM) is a linear classifier that is capable of performing binary classification. Support vector machines are able to conduct non-linear classification in addition to linear classification by utilizing the kernel technique, which implicitly transforms inputs into highly dimensional vector spaces [14].The support vector machine makes an effort to pick the ideal hyperplane that has the largest margins between data instances [18]. This is done in order to improve the accuracy of the classification process.

Due to the fact that it learns during the assessment step and continues to maintain data samples throughout the learning step, the KNN classifier is a method that is particularly sluggish. It is [14] Through the use of the KNN method, the class of samples that are to be classed is determined by comparing the samples that are most similar to one another.[15] Given that K continues to be positive, the selection of the neighbors is accomplished by using a collection of objects that have label information. [16] The AdaBoost classifiers algorithm employs a weak learner to assist in the training of a collection of classifiers in order to provide the best possible classifier. [17] An application that

combines the Backpropagation Neural Network approach with Adaptive Boosting is recommended by Yan Gao et al. for the purpose of training a software fault prediction model. The problem of overfitting is avoided by AdaBoost, which also functions as an efficient predictor for data that is not consistent [18].

Combined defect predictor is the name given to the integrated prediction model that was developed by Yun ZHANG and colleagues. This model was created by combining multiple fundamental classifiers. Integrated model was constructed by the author using two different sorts of voting techniques. On the other hand, Max Voting generates the highest confidence values from a wide variety of core classifiers, whereas First Ave Voting is responsible for aggregating confidence ratings [19]. With the use of the WEKA tool, Marwa Assim and her colleagues were able to classify the data using eight different machine learning techniques, including percentage split and k-fold cross-validation [20]

Having a conversation on machine learning without supervision A straightforward explanation of K-means clustering has been provided by Xin Dong and colleagues. After selecting K cluster centers at random, the next step is to place the element in the cluster that is the most similar to it by analyzing how it compares to the other clusters. Utilizing the average of all the objects contained within a particular cluster, one can ascertain the cluster center for each individual cluster [2].

There are researchers who construct software defect models by employing the deep learning methodology. The original minimal feature representation is transformed into a high-level feature by Deep Learning [22][23], which is accomplished through the use of multi-layer processing. It has the ability to accomplish complex categorization tasks by utilizing fundamental models [14]. Neural networks are formed from biological neural networks, which are composed of synapses and neurons. Neural network models are created from these systems. When neurons are modeled, they are represented as nodes in a graphical network, and synapses serve as weighted edges that connect the nodes [17]. In the context of machine learning applications, artificial neurons are comprised of artificial neural networks, which have the potential to function as a non-linear classifier. Interconnections between the input neurons make it possible for them to work simultaneously. They do this by exchanging signals with one another and calculating output using a non-linear function [21].

3. Materials and Methods

The Random Forest algorithm is a machine learning technique that use numerous decision trees to create predictions and classify the data. In this project, we have implemented a modified version of the Random Forest algorithm. Entropy is a measurement that is utilized to estimate the degree of impurity or randomness that exists within a collection of samples. The formula for calculating entropy using the natural logarithm is as follows:

$$E = -\Sigma p(y_i)\, log_2\big(p(y_i)\big) \quad (1)$$

Using a Taylor series expression, we have updated the algorithm in order to approximate the natural logarithm that is utilized in the entropy formula. This is done in order to measure the quality of each node split in the different splits that the Decision Tree algorithm uses. The workflow for our proposed framework for software defect prediction is depicted in detail in Figure 1.

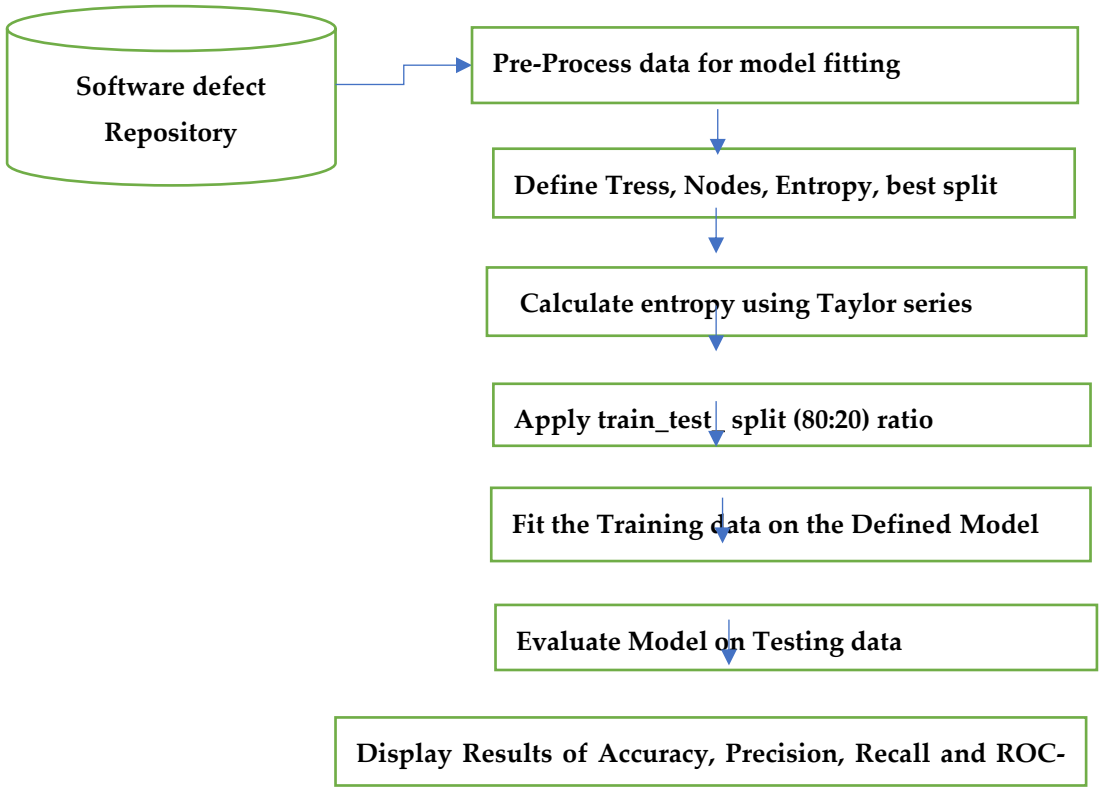

Figure 1. Proposed framework for software defect prediction

3.1 Main Modules of Prediction model

Class Node: Specify the decision tree node's structure.

Class Decision Tree: Define the decision tree's structure and put a few functions in place for the dataset's splitting, growth, and information-gathering, among other things.

Class Random Forest: Define the random forest's structure and put a few functions into place for things like model fitting and outcome prediction.

Main methodology: Apply the original and changed algorithms on the CM1 dataset, assess and analyze the outcomes, and contrast the updated algorithm's performance with the original.

3.2 Data Source:

We have tested our model's performance using the CM1 NASA dataset.

NASA Software Defect Repository: This is a dataset that is accessible to the general public and was obtained from a software defect repository that is administered by NASA. A wide variety of NASA

systems are included into the several programs that are run by NASA. As an illustration, the acronym CM1 refers to the instruments of spacecraft, but the acronyms KC1, KC3, and MC2 refer to the administration of ground data storage. MW1 is in charge of managing the data exchanges, and PC1, PC2, PC3, and PC4 are the programs that are responsible for the software that is used by earth-orbiting satellites. A number of characteristics are included in it, and these characteristics are used as independent variables to decide whether or not a particular piece of software may have some sort of defect. Some of these qualities include, but are not limited to, the complexity of the program, the amount of control flow statements, the number of lines of code, the number of comments contained within the code, and a great deal of other properties. The likelihood that a program is flawed is represented by the dependent variable known as "Defective."

3.3 Algorithm Selection and Modification:

We decided to use the Random Forest method as our foundational model because it offers a number of benefits, including the control of overfitting, the management of missing values, and the provision of feature importance scores. An approximation of the natural logarithm that is utilized in the Entropy formula has been incorporated into the Random Forest method through the utilization of the Taylor series expression. In the case of natural logarithm, the Taylor series formulation is as follows:

$$\text{Entropy} = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n}(x-1)^n \quad \text{from n = 0 to } \infty \qquad (2)$$

Here, the variable x can be any positive number between 0 and 1, inclusive. Through the utilization of this approximation, we are able to compute the natural logarithm for a certain value of x by employing a limited number of constituent terms.

When compared to the natural logarithm, the usage of Taylor series results in a significantly higher degree of precision throughout the entropy computation process. In contrast, the natural logarithm function necessitates the utilization of a complicated algorithm that incorporates iterative approaches, floating-point arithmetic, and error management. In addition, the Taylor series gives the user the ability to personalize the approximation by selecting the degree of the polynomial for the model. An array of probabilities was used as the input for this adjustment, and an array of approximated logarithms was returned as the output. This modification was implemented by constructing a custom function that follows this pattern. After that, we have incorporated the function described above into our Entropy formula in order to compute the split of each node.

3.4 Training and Testing Model:

Through the utilization of a wide range of hyperparameters, we have trained our modified Random Forest method on training data. These hyperparameters include the number of trees (n_trees), the maximum depth (max_depth), and the minimum samples split (min_samples_split). After that, we

used a grid search approach to find the most effective combination of hyperparameters for each dataset individually. This was then done in order to find the optimal combination. Using the CM1 training dataset, our revised Random Forest method was put through its paces, and its effectiveness was evaluated using a range of different measures.

4. Theory and Calculation

Out of the CYCLOMATIC_DENSITY feature that is included in the CM1 dataset, we have chosen ten sample values to represent it. Our next step is to determine the absolute value for each individual observation. Following is a list of the values:

[0.2, 0.13, 0.15, 0.17, 0.12, 0.2, 0.14, 0.28, 0.11, 0.17]. We can calculate entropy for value (0.2) as follow,

$$\mathbf{E\ (0.2)} = \frac{(-1)^{1+1}}{1}(0.2-1)^1 + \frac{(-1)^{2+1}}{2}(0.2-1)^2 + \frac{(-1)^{3+1}}{3}(0.2-1)^3 + \frac{(-1)^{4+1}}{4}(0.2-1)^4 + \frac{(-1)^{5+1}}{5}(0.2-1)^5 + \frac{(-1)^{6+1}}{6}(0.2-1)^6 + \frac{(-1)^{7+1}}{7}(0.2-1)^7 + \frac{(-1)^{8+1}}{8}(0.2-1)^8 + \frac{(-1)^{9+1}}{9}(0.2-1)^9 + \frac{(-1)^{10+1}}{10}(0.2-1)^{10} = \mathbf{1.60.}$$ Similarly, we can calculate for other points

$$\mathbf{E\ (0.13)} = \frac{(-1)^{1+1}}{1}(0.13-1)^1 + \ldots + \frac{(-1)^{10+1}}{10}(0.13-1)^{10} = \mathbf{2.04,}\quad \mathbf{E\ (0.15)} = \frac{(-1)^{1+1}}{1}(0.15-1)^1 + \ldots + \frac{(-1)^{10+1}}{10}(0.15-1)^{10} = \mathbf{1.89}$$

$$\mathbf{E\ (0.17)} = \frac{(-1)^{1+1}}{1}(0.17-1)^1 + \ldots + \frac{(-1)^{10+1}}{10}(0.17-1)^{10} = \mathbf{1.77,}\quad \mathbf{E\ (0.12)} = \frac{(-1)^{1+1}}{1}(0.12-1)^1 + \ldots + \frac{(-1)^{10+1}}{10}(0.12-1)^{10} = \mathbf{2.12}$$

$$\mathbf{E\ (0.2)} = \frac{(-1)^{1+1}}{1}(0.2-1)^1 + \ldots + \frac{(-1)^{10+1}}{10}(0.2-1)^{10} = \mathbf{1.60,}\quad \mathbf{E\ (0.14)} = \frac{(-1)^{1+1}}{1}(0.14-1)^1 + \ldots + \frac{(-1)^{10+1}}{10}(0.14-1)^{10} = \mathbf{1.96}$$

$$\mathbf{E\ (0.28)} = \frac{(-1)^{1+1}}{1}(0.28-1)^1 + \ldots + \frac{(-1)^{10+1}}{10}(0.28-1)^{10} = \mathbf{1.27,}\quad \mathbf{E\ (0.11)} = \frac{(-1)^{1+1}}{1}(0.11-1)^1 + \ldots + \frac{(-1)^{10+1}}{10}(0.11-1)^{10} = \mathbf{2.20}$$

$$\mathbf{E\ (0.17)} = \frac{(-1)^{1+1}}{1}(0.17-1)^1 + \ldots + \frac{(-1)^{10+1}}{10}(0.17-1)^{10} = \mathbf{1.77}$$

Table 2. Entropy calculation values by Taylor series

Value	0.2	0.13	0.15	0.17	0.12	0.2	0.14	0.28	0.11
Entropy	1.6094	2.0402	1.8971	1.7719	2.1202	1.6094	1.9661	1.2729	2.2072

5. Results

In the current method, the quality of each node split in the decision tree is determined by combining entropy with the natural logarithm. As a result of our desire to investigate an alternative approach to calculating the natural logarithm that was utilized in the initial Entropy formula, we are currently employing the modified Entropy formula that makes use of the Taylor series. When applied to the CM1 dataset, the results that were obtained by using both the original Entropy formula and the modified Entropy formula utilizing Taylor's series are presented in Table 3. We are able to clearly notice the change in accuracy, precision, and recall in both the original Random Forest method and the updated Random Forest algorithm by analyzing the results that were acquired by the random datasets they contained.

Table 3. Software defect prediction model result - Accuracy, precision, and recall

Parameters	**Precision (0)**	**Precision (1)**	**Recall (0)**	**Recall (1)**	**Accuracy**
Original formula	0.88	0.5	0.98	0.11	0.8695
Modified formula	0.88	1	1	0.11	0.884

With the help of this table, we are able to quickly draw the conclusion that we can increase the accuracy of our calculations by utilizing the Taylor series to calculate entropy. The ROC curve for the training dataset as well as the testing dataset is displayed in the figure. This curve is shown for both the original Random Forest algorithm and the modified Random Forest algorithm.

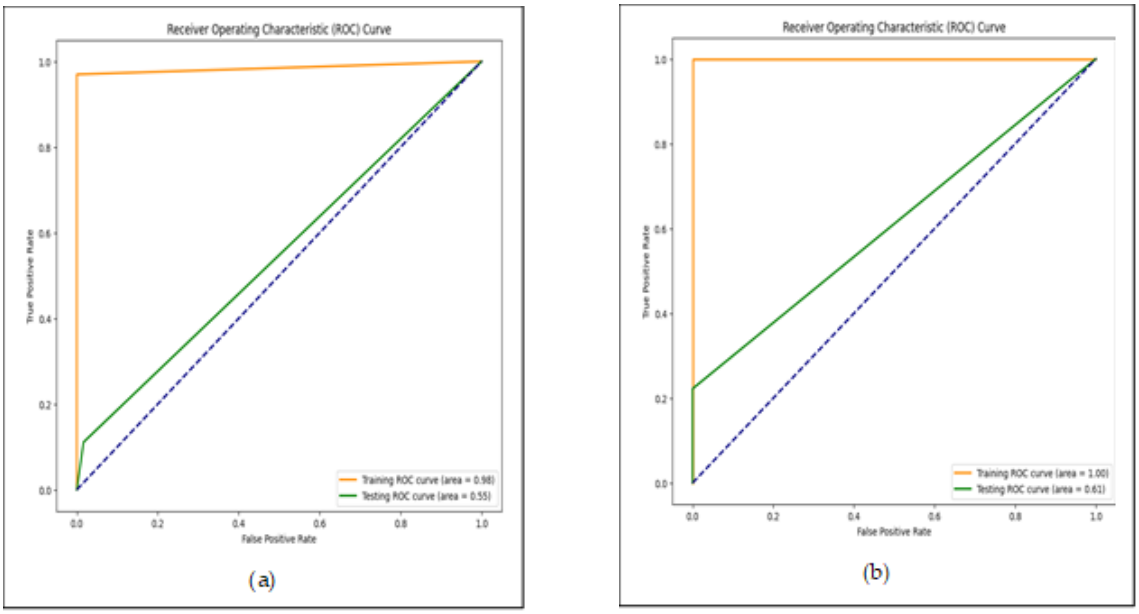

Figure 2. (a) ROC curve for original Random Forest Algorithm; (b) ROC curve for modified Random Forest Algorithm

The ROC curves that were presented earlier allow us to draw the conclusion that the area under the Training ROC curve and the area under the Testing ROC curve both exhibit considerable variations in their respective results.

6. Conclusions

The Taylor series expression was utilized in this study to approximate the natural logarithm that is utilized in the entropy formula to quantify the quality of each node split. This was done in order to investigate the computation of entropy. A CYCLOMATIC_DENSITY feature of the CM1 dataset was used to choose ten sample values, and we manually calculated the entropy for each of those values. This entropy computation has been implemented in the random forest algorithm, and evaluations have been performed on the CM1 dataset. The results suggest that it has the potential to enhance accuracy, and the ROC curve demonstrates that there is a shift in the true positive rate, which is beneficial for testing datasets. Future work could be expanded to include the prediction of software defects in commercial datasets and the construction of a generalized model through the testing of software defect prediction on cross-project instances.

Author Contributions:

Ranjeetsingh Suryawanshi: Data collection and analysis, developing methodology, Design and Development of an application.

Amol Kadam: Reviewing and Editing of the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] M. K. Thota, F. H. Shajin, and P. Rajesh, “Survey on software defect prediction techniques,” *Int. J. Appl. Sci. Eng.*, vol. 17, no. 4, pp. 331–344, 2020, doi: 10.6703/IJASE.202012_17(4).331.

[2] X. Dong, Y. Liang, S. Miyamoto, and S. Yamaguchi, “Ensemble learning based software defect prediction,” *J. Eng. Res.*, no. November, 2023, doi: 10.1016/j.jer.2023.10.038.

[3] M. Mustaqeem and T. Siddiqui, “A hybrid software defects prediction model for imbalance datasets using machine learning techniques: (S-SVM model),” *J. Auton. Intell.*, vol. 6, no. 1, pp. 1–19, 2023, doi: 10.32629/jai.v6i1.559.

[4] S. K. Pandey and A. K. Tripathi, “An empirical study toward dealing with noise and class imbalance issues in software defect prediction,” *Soft Comput.*, vol. 25, no. 21, pp. 13465–13492, 2021, doi: 10.1007/s00500-021-06096-3.

[5] K. K. Bejjanki, J. Gyani, and N. Gugulothu, “Class imbalance reduction (CIR): A novel approach to software defect prediction in the presence of class imbalance,” *Symmetry (Basel).*, vol. 12, no. 3, 2020, doi: 10.3390/sym12030407.

[6] K. J. Eldho, “Impact of Unbalanced Classification on the Performance of Software Defect Prediction Models,” *Indian J. Sci. Technol.*, vol. 15, no. 6, pp. 237–242, 2022, doi: 10.17485/ijst/v15i6.2193.

[7] S. K. Pandey, R. B. Mishra, and A. K. Tripathi, “BPDET: An effective software bug prediction model using deep

representation and ensemble learning techniques," *Expert Syst. Appl.*, vol. 144, p. 113085, 2020, doi: 10.1016/j.eswa.2019.113085.

[8] L. S. Shapley, "A Value for n-Person Games Contributions to the Theory of Games," *In Annals of Mathematical Studies,edited by Harold William Kuhn and Albert William Tucker,Princeton University Press*, vol. 2, no. 4. pp. 307–318, 1953. doi: 10.1515/9781400881970-018.

[9] K. Magal.R and S. Gracia Jacob, "Improved Random Forest Algorithm for Software Defect Prediction through Data Mining Techniques," *Int. J. Comput. Appl.*, vol. 117, no. 23, pp. 18–22, 2015, doi: 10.5120/20693-3582.

[10] L. qiong Chen, C. Wang, and S. long Song, "Software defect prediction based on nested-stacking and heterogeneous feature selection," *Complex Intell. Syst.*, vol. 8, no. 4, pp. 3333–3348, 2022, doi: 10.1007/s40747-022-00676-y.

[11] N. Gayatri, S. Nickolas, and A. V Reddy, "Feature Selection Using Decision Tree Induction in Class level Metrics Dataset for Software Defect Predictions," *World Congr. Eng. Comput. Sci. Vols 1 2*, vol. I, pp. 124–129, 2010, [Online]. Available: http://www.iaeng.org/publication/WCECS2010/WCECS2010_pp124-129.pdf

[12] D. T. Pham and G. A. Ruz, "Unsupervised training of Bayesian networks for data clustering," *Proc. R. Soc. A Math. Phys. Eng. Sci.*, vol. 465, no. 2109, pp. 2927–2948, 2009, doi: 10.1098/rspa.2009.0065.

[13] R. Chennappan and Vidyaathulasiraman, "An automated software failure prediction technique using hybrid machine learning algorithms," *J. Eng. Res.*, vol. 11, no. 1, p. 100002, 2023, doi: 10.1016/j.jer.2023.100002.

[14] H. Cao, "A Systematic Study for Learning-Based Software Defect Prediction," *J. Phys. Conf. Ser.*, vol. 1487, no. 1, 2020, doi: 10.1088/1742-6596/1487/1/012017.

[15] T. D. Buskirk, "Surveying the Forests and Sampling the Trees: An overview of Classification and Regression Trees and Random Forests with applications in Survey Research," *Surv. Pract.*, vol. 11, no. 1, pp. 1–13, 2018, doi: 10.29115/sp-2018-0003.

[16] H. M. Premalatha and C. V. Srikrishna, "Software fault prediction and classification using cost based random forest in spiral life cycle model," *Int. J. Intell. Eng. Syst.*, vol. 11, no. 2, pp. 10–17, 2018, doi: 10.22266/IJIES2018.0430.02.

[17] L. Perreault, S. Berardinelli, C. Izurieta, and J. Sheppard, "Using classifiers for software defect detection," *26th Int. Conf. Softw. Eng. Data Eng. SEDE 2017*, pp. 131–137, 2017.

[18] D. A. Pisner and D. M. Schnyer, *Support vector machine*. Elsevier Inc., 2019. doi: 10.1016/B978-0-12-815739-8.00006-7.

[19] Y. Zhang, D. Lo, X. Xia, and J. Sun, "Combined classifier for cross-project defect prediction: an extended empirical study," *Front. Comput. Sci.*, vol. 12, no. 2, pp. 280–296, 2018, doi: 10.1007/s11704-017-6015-y.

[20] Q. O. and M. H. M. Assim, "Software Defects Prediction Using Machine Learning Algorithms," *Int. Conf. Data Anal. Bus. Ind. W. Towar. a Sustain. Econ.*, pp. 1–6, 2020, doi: 10.1109/ICDABI51230.2020.9325677.

[21] S. P. Niculescu, "Artificial neural networks and genetic algorithms in QSAR," *J. Mol. Struct. THEOCHEM*, vol. 622, no. 1–2, pp. 71–83, 2003, doi: 10.1016/S0166-1280(02)00619-X.

[22] Mane, D., Ashtagi, R., Kumbharkar, P., Kadam, S., Salunkhe, D., Upadhye, G. (2022). An improved transfer learning approach for classification of types of cancer. Traitement du Signal, Vol. 39, No. 6, pp. 2095-2101. https://doi.org/10.18280/ts.390622

[23] Salunke, Dipmala & Mane, Deepak & Joshi, Ram & Peddi, Prasadu. (2022). Customized convolutional neural network to detect dental caries from radiovisiography(RVG) images. International Journal of Advanced Technology and Engineering Exploration. 9. 827-838. 10.19101/IJATEE.2021.874862.