Software Defect Classification and Defect Prioritization using Machine Learning techniques: A Comparative Study

S. Aarthi Priyadarshni[1], T. Sathya Priya[2]

[1]Research Scholar, PG & Research, Department of Computer Science, Shri Sakthi Kailassh Women's College, Salem, Tamil Nadu, India
[2]Assistant Professor, PG & Research, Department of Computer Science, Shri Sakthi Kailassh Women's College, Salem, Tamil Nadu, India

---***---

Abstract - *Defects are imperfections in a product or a system that indicates the product has failed to meet the stated expectation. Software defects are deviations in actual results from those of the expected results. A Software defect prediction model when paired with machine learning algorithms helps in identifying defects even while the software is still being developed. When the defects are further classified and prioritized based on its impact, the model greatly helps the development team to focus keenly on potential areas of risk. With a focus to choose an appropriate framework to build a defect prediction model that incorporates the goals of classification and prioritization of software defects as its key aspects, this paper demonstrates a comparative study over the methods used in existing models. This paper briefly outlines some of the important techniques such as Data Preprocessing, Feature Selection, Feature Reduction methods and Performance Measures which are to be considered while constructing the defect prediction framework and pursues a literature survey on commonly used machine learning classifiers that are recommended for such prediction models.*

Key Words: **Software defect prediction, Defect Classification, Defect Prioritization, Data Preprocessing, Machine Learning Algorithms, Performance Measures, Feature Selection and Feature Reduction.**

1.INTRODUCTION

Defect prediction models on software products has been a very active area of research in the field of Software Engineering. Software Defects are abnormalities that impair the quality, function and utility of a system. They may come from misconceptions perceived during the design; or due to faulty modules that disrupts the integration of the software. Most organizations follow a traditional Time-Based Release (TBR) where the software product is planned for a release on quarterly, half-yearly or yearly delivery. The team may decide to postpone the release dates so that they can focus on delivering a defect-free product; or may decide to release the product with defects to keep up with their timelines. In the latter case, the impact of a defective product has a huge effect on the cost plan during the maintenance phase spending most of their time in defect prediction and fix.

As the Software Defect Prediction models are one of the dynamic methods to predict the reliability of the software, the need for a technology that has the ability to change over time becomes necessary. In this regard, machine learning methods have the efficiency to learn the changes through training datasets. Learning methodologies such as supervised learning, unsupervised learning, and reinforcement learning helps the training datasets to adapt to new changes without any manual intervention. Thus, machine learning approaches provide a dynamic platform for Software Defect Prediction models.

The rest of the paper is organized as follows: Section 2: Significance of Machine learning techniques – Section 3: Software Defect Classification Models – 3.1 Learning Algorithms – 3.2 Performance Measurements – 3.3 Literature Survey – Section 4: Software Defect Prioritization Models – 4.1 Feature Selection methods – 4.2 Feature Reduction methods – 4.3 Literature Survey – Section 5: Comparative Analysis – Section 6: Conclusion

2. SIGNIFICANCE OF MACHINE LEARNING

Machine learning approaches when incorporated in Software Defect Prediction models tends to dynamically predict software defects since the model matures along the development cycle. This section outlines a general machine learning process to understand its benefits when used in such models. (Fig.1)

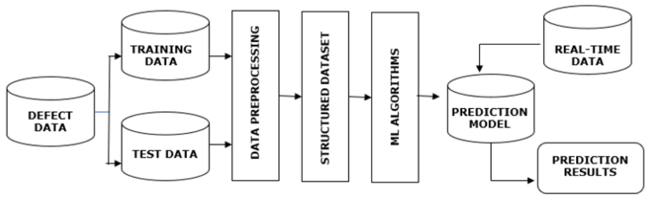

Fig.1 Defect Prediction Model using Machine Learning techniques

(1) *Data Collection:* Defect data is collected from historical data repositories or software archives. This data is used in training the prediction model

and evaluating the performance of the prediction model. Hence the defect data is divided into two parts, namely the training datasets and test datasets.

(2) *Data Preprocessing:* Machine learning models largely depend on training datasets and test datasets, and hence it is crucial to prepare the data before training the prediction model. This step is intended to clean the data using the Data Preprocessing techniques as it greatly helps in constructing a structured dataset. Basically, this step involves the Data Selection process which selects a subset of data from the given data; Data Preprocessing cleanses and formats the selected data and Data Transformation process scales the preprocessed data and collates the selected feature subsets into a single subset. The output of this step is a Structured dataset.

(3) *Machine learning algorithms:* Before starting to build a prediction model, a learning scheme is first selected. This scheme involves the various machine learning approaches such as the supervised, unsupervised, and reinforcement learning methodologies and selection of machine learning algorithms. Different machine learning algorithms serves different purposes, therefore choosing appropriate algorithms is important for building an effective model.

(4) *Prediction Model:* A Prediction model is built using the selected learning scheme, appropriate machine learning algorithms and training datasets which trains the model in prediction.

(5) *Real-time Data:* These are instances which is used in evaluating the performance of the prediction model. In a Software defect prediction model, these instances can be the current version of the software.

(6) *Prediction Results:* Predicted results which shows the classification and prioritization of defects are generated.

3. SOFTWARE DEFECT CLASSIFICATION MODELS

Defect Classification models categorizes the software code attributes into defective and non-defective labels(fig.2) based on the data collected from previous versions of the software. This classification helps the developers to plan their resources and focus their work keenly on defective modules. A variety of classification techniques have been used to build defect prediction models ranging from simple (e.g., logistic regression) to advanced techniques (e.g., Multivariate Adaptive Regression Splines (MARS)).

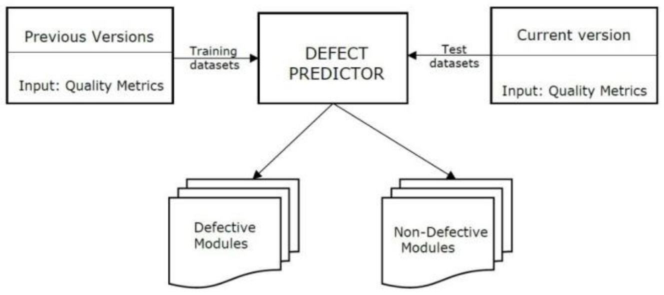

Fig.2 Most commonly used defect classification structure

3.1 MACHINE LEARNING ALGORITHMS

In this section, we have taken five most commonly used machine learning algorithms from the literature survey to discuss their functionalities, advantages and limitations in order to understand their capability and performance accuracy while used in classification models. Our goal is to determine a best fitting classifier for a software defect classification model by comparing their performances. This study involves five machine learning classifiers namely, Naïve Bayes (NB), Support Vector Machine (SVM), Random Forest (RF), K-nearest neighbors (KNN) and Decision Tree (DT) which are briefly outlined below:

A. *Naïve Bayes (NB):* Naïve Bayes classifiers are a family of simple probabilistic classifiers that are based on Bayes' theorem. They classify each pair of features with strong assumptions that the features are independent of each other. Naïve Bayes classifiers are easy to implement and requires a small amount of training data to estimate the test data. But on the other side as disadvantages, if the assumptions that the features are independent holds true then it can perform better, if not, they face 'zero-frequency problem' where the test data contains variable that is not present in the training data.

B. *Support vector machine (SVM):* This classifier has the capability of both classification and regression using a supervised learning method. It uses decision boundaries called the hyperplane to classify datapoints that falls on either side and categorizes them into different classes. It is one of the most effective and simple methods used in classification and are considered to be effective in separating non-linear data. But the classifier causes overlapping issues if the hyperplane is not well-defined. Therefore, it is important to choose an optimal hyperplane.

C. *Random Forest (RF):* It is a combination of different decision trees that functions on various subsets of the given dataset. It takes a prediction from each tree and selects the best vote of all predicted classes over all trees, thus reducing the problem of overfitting. It yields high accuracy when more noticeable number of trees are incorporated, but becomes less instinctive when we have enormous collection of trees.

D. *K-Nearest Neighbor (KNN):* The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. 'k' in KNN is a parameter that refers to the number of nearest neighbors that are included in the majority of the voting process. We need to select an appropriate K value in order to achieve the maximum accuracy of the model. Classification is done by a majority vote to its neighbors and the data is assigned to the class which has the nearest neighbor. It is very easy to implement and has a quick calculation time, but becomes slower as the variables increase.

E. *Decision Tree (DT):* They can handle both numeric and categorical data. It breaks down the dataset into smaller subsets as the nodes of the tree gets developed. The output of the decision tree is a leaf node that represents the decision. It is simple to understand and easy to visualize but becomes unstable when there are constant changes in the input dataset.

3.2 PERFORMANCE MEASURES

Selection of algorithms largely influences the performance of the working model. Therefore, it is important to evaluate the performance of the algorithms in the prediction model by using performance measures or also known as performance metrics. Based on the literature survey, this section briefly elaborates on commonly used performance measures such as Accuracy, Precision, Recall, ROC AUC and F-measure to indicate its appropriate place of usage.

A confusion matrix that summarizes the prediction results contain variables such as True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) whose values evaluate the performance of the algorithms. These variables are elaborated below:

- True Positives (TP): When the model has correctly predicted the positive classes.
- True Negative (TN): When the model has correctly predicted the negative classes.
- False Positive (FP): When the model's prediction of positive classes is incorrect
- False Negative (FN): When the model's prediction of negative classes is incorrect

Each performance measures are briefly elaborated below:

a) *Accuracy:* Accuracy is the proportion of true results among the total number of cases examined. This is a valid choice of measurement when there is no class imbalance problem. It cannot be used when the target class is sparse. The accuracy rate on the performance of the classification algorithms is calculated as below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

b) *Precision:* This determines on what proportion of predicted positives are truly positive. This is a valid choice of measurement when we want to be very sure of our predictions. Precision values are calculated as below:

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

c) *Recall:* This determines on what proportion of actual positives are correctly classified. This is a valid choice of measurement when we want to capture as many positives as possible. Recall values are calculated as below:

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

d) *ROC AUC:* An ROC curve (Receiver Operating Characteristic Curve) is a graph that shows the performance of the classification algorithms at all classification thresholds. This curve plots two parameters namely True Positive Rate (TPR) and False Positive Rate (FPR) which is calculated as below:

$$TPR = \frac{TP}{TP + FN} \quad (4)$$

$$FPR = \frac{FP}{FP + TN} \quad (5)$$

AUC stands for Area Under the ROC curve that measures the two-dimensional area underneath the entire ROC curve.

e) *F- Measures:* This measurement is a harmonic mean of precision and recall values and the F1 score is between 0 and 1. This is mainly used in binary predictions. F-Measure is calculated as below:

$$Precision = \frac{2 \times Recall \times Precision}{Recall + Precision} \quad (6)$$

3.3 LITERATURE SURVEY

Romi Satria Wahono et al [1] conducted analysis to test the performance of some of the widely used classification algorithms and proposed an algorithm that best suited a software defect prediction model. The research involved 10 classifiers to build classification models and their performance were tested in 9 NASA MDP datasets. Area under curve (AUC) was used as an accuracy indicator to evaluate the performance of classifiers. Friedman and Nemenyi post hoc tests were used to test the significance of AUC differences between classifiers. The results showed that the models using logistic regression performed best in most NASA MDP datasets. Models based on Naïve bayes, neural network, support vector machine and k* classifiers also performed well although there was no significant difference between them. However, models that involved the Decision tree-based classifiers, linear discriminant analysis and k-nearest neighbor tend to underperform.

David Bowes et al[2] conducted a research to determine if different classifiers detect different types of defects. For this purpose, four classifiers namely Random Forest, Naïve Bayes, RPart and SVM classifiers were involved to predict defects in NASA, open source and commercial datasets. The level of the defect predictions produced by these classifiers were captured in a confusion matrix and their predictions were analyzed for each individual classifier. Although all four classifiers showed consistent performance in predicting the defects, the research identified that each classifier identifies different sets of defects when exposed to different software domains that comprised of both open-source as well as commercial datasets. This research concluded that using a classifier ensembles method combined with decision-making strategies will enhance the performance of the individual classifiers to identify a specific set of defects.

Cagatay Catal[3] suggested a semi-supervised learning approach to classify defects. This research identified that predicting defective modules by using only the labeled modules was a challenging problem in the software industry. Supervised classification approaches that were destined to predict defects based on only a few defect data could not perform high. Therefore, this study proposed a solution to the above problem, by combining labeled data points with unlabeled data points during the training phase, thus using a semi-supervised learning method. The study involved four semi-supervised classification methods such as Low-density separation (LDS), Support Vector Machine (SVM), Expectation-Maximization (EM-SEMI), and Class Mass Normalization (CMN) methods on NASA data sets, which were CM1, KC1, KC2, and PC1. This research concluded that SVM and LDS algorithms outperformed CMN and EM-SEMI algorithms. LDS algorithm performed even much better than SVM when the data set is large. Therefore, this study suggested LDS-based prediction approach for software defect prediction model using a semi-supervised learning method.

Nitish Pandey et al [4]. Misclassification of defects often costs valuable time of the developers. Hence this research focused on automating the classification of the defect reports. The researchers first conducted an empirical study to select appropriate machine learning algorithms that helps in classifying the defect reports. For this purpose, classification algorithms such as Naive Bayes (NB), Linear Discriminant Analysis, K-nearest neighbors (KNN), Support Vector Machine (SVM) with various kernels, Decision Tree (DT) and Random Forest (RF) were applied separately to classify defects from three open-source projects. Performance of different algorithms were evaluated using F-measure, average accuracy and weighted average f-measures. Results identified that RF performs best, while SVM with certain kernels also achieved high performance.

4. SOFTWARE DEFECT PRIORITIZATION MODELS

Defect priority defines the order in which defects will be fixed by the developers as they indicate the impact on the system. Higher the impact of the defect on the business, higher is the priority of the defect. A general defect prioritization process is depicted in fig 3.

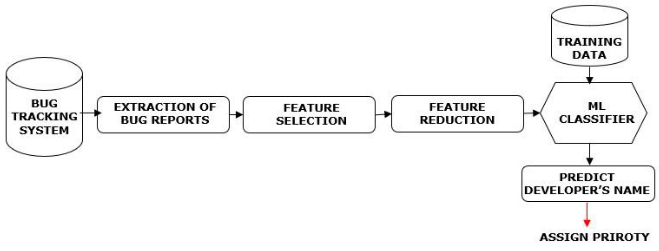

Fig.3 A general Defect Prioritization Process

Priority levels differ from model to model such as the one shown in fig.4 and they determine how quickly the defects are to be fixed. However, this assignment of the priority levels is determined manually. When a dynamic model such as the software defect prediction model considers defect prioritization as one of its aspects, the model exhibits reliable advancements and helps to overcome this manual effort.

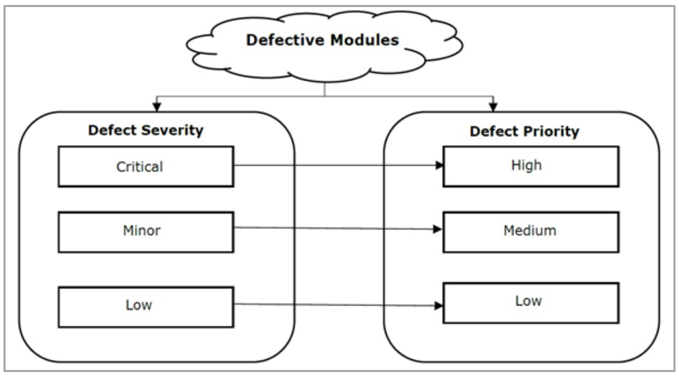

Fig.4 Most commonly used defect priority levels

4.1 FEATURE SELECTION METHODS

Reported defects are commonly maintained using a bug tracking system such as JIRA. Each defect report contains details such as the summary of the defect, the affected module and its impact on the system. A defect prioritization model begins with feature selection process that selects a subset of the above state features from the defect reports in order to determine their priorities. We briefly summarize some of the feature selection techniques commonly used in the existing defect prioritization models as below:

A. *Information Gain (IG):* This technique selects features based on the information gain with the frequency of items. It uses a filtering approach that ranks the subset of features in descending order, so that we can prioritize defects with high information gain values on the top order of the list and defects with low information gain values are lesser prioritized.

B. *Chi-Square Test:* This method helps to choose the best features by testing the relationship between independent category feature (predictor) and dependent category feature (response). In feature selection, this method aims to select features which are highly dependent on response. Given the data of two variables, we can get observed count O and expected count E. Chi-Square test measures how expected count E and observed count O deviates from each other. This is calculated as below:

$$X_c^2 = \sum \frac{(O_i - E_i)^2}{E_i} \quad (7)$$

where:

c = degree of freedom
O = observed value(s)
E = expected value(s)

Information Gain and Chi-Square test are feature selection methods used when the given set of features are categorical. Both methods can remove distractions without losing the categorization accuracy and hence they are highly recommended in classification problems.

4.2 FEATURE REDUCTION METHODS

Feature reduction which is also known as dimensionality reduction, is the process of reducing the number of features from the given set of features without losing its originality and important information. It decreases the number of dimensions, making the data less sparse and more statistically significant for machine learning applications. Some of the commonly used feature reduction techniques in defect prioritization models are explained below:

A. *Non-negative Matrix Factorization (NMF):* It is a feature reduction technique used when there are many unclear or weak attributes which have weak predictability. By combining such attributes, NMF can produce expressive patterns. NMF is often used in text mining. In a text document, the same word can occur in different places with different meanings. For example, "hike" can be applied to the outdoors or to interest rates. By combining attributes, NMF introduces context, which is essential for predictive power: 'hike' + 'mountain'= 'outdoor sport' and 'hike' + 'interest' = 'interest rates'

B. *Principle Component Analysis (PCA):* Principal component analysis (PCA) is a mathematical algorithm that reduces the dimensionality of the data while retaining the originality and accuracy in the data set. It achieves this reduction by identifying directions, called principal components, along which the accuracy in the data is maximal. PCA is applied to a dataset represented in the form of n x m matrix A that results in a projection of A which is a subset of reduced features.

4.3 LITERATURE SURVEY

Aladdin Baarah et al [5]. This research aimed at building a prediction model that automatically determines the defect priorities using machine learning approaches. The study involved Information Gain (IG) for feature selection and evaluated the proposed methodology by comparing eight popularly used machine learning algorithms such as Naive Bayes (NB), Naive Bayes Multinomial (NBM), Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), Logistic Model Trees (LMT), Decision Rules (JRip) and K-Nearest Neighbor. Datasets were constructed from historical bug reports stored in JIRA bug tracking system from open-sourced projects. Accuracy was measured using F-measure and Area under the curve (AUC). Experimental results showed that Logistic Model Trees (DT: LMT) outperformed other classifiers and the overall performance has been enhanced after applying feature selection method. The results showed that the LMT algorithms reported the best performance results based on all performance measures (Accuracy= 86.31, AUC= 0.90, F-measure= 0.91).

Lian Yu et al [6]. This research created an ANN model to prioritize software defects by adding an extra layer to resolve the reliance between attributes of a tester and severity. This model adopts an interactive training method to handle new feature problems. The inputs to this ANN model were based on dynamic testing results, such as testing milestones and defect categories and the output was the defect priority prediction. Researchers alienated their work into four main steps: Defining levels for detect priorities – Extracting attributes from defect reports – creating a three-layer model of ANN – Error function and Backward propagation to calculate errors and adjust weights in output layer and middle layers. The results of the proposed model were compared with the Bayes algorithm. The 3-fold cross-test in both the closed test and opening test ways were executed and the ANN model shows better qualification in terms of recall, precision and F-measure when compared to the Bayes approach. The comparison was carried on the RIS2.0 software project with sample size about 2000 bug reports.

Gitika Sharma et al [7] The proposed approach created dictionary of critical terms which indicated the severity of the bugs. Data sets were extracted from open-sourced Eclipse development environment and Pre-processing was performed on the text summary of the bug reports. Text-Document matrix were created which consisted of columns that represented the terms that occurred in the documents and rows represented each bug report. Feature selection methods such as Info-Gain and Chi Square retrieved the most informative terms from the corpus of the matrix. A dictionary consisting of 125 informative terms were created and sorted in descending order according to their weights using top k-terms approach. Bug reports were classified into binary levels as severe and non-severe bug reports, and the classifiers were trained using Naïve Bayes Multinomial

(NBM) algorithm for probabilistic classification and K-nearest neighbor (KNN) algorithm for similarity-based classification. Results were validated using 5-fold cross validation method, and the performance of the classifier was evaluated using Accuracy and Precision performance measures. Experimental results showed that KNN performed better as it classified bug reports using common value attributes and on the current distance, whereas NBM did not utilize the common values and were based on some initial probabilities. Hence, this study recommends KNN classifiers for bug severity prediction.

Ishrat-Un-Nisa Uqaili et al[8]. By analyzing source code modules, the research proposed an approach to categorize and prioritize defects based on each class of the source code. This approach trained the defect prediction model to categorize defects into two classes: clean and buggy and prioritize defects into three levels: clean, ordinary, and complex where complex priority was further classified into three sublevels: Critical bugs, Major Bugs and Non-Trivial bugs. The proposed model used three classifiers: Random Forest (RF), MultiLayer Perceptron (MLP) and Naive Bayes (NB), with 4 groups of transformed datasets. The proposed method used genetic algorithms such as Principal Component Analysis (PCA) for feature selection technique, Genetic Search (GenSch) to minimize correlation problem, and SMOTE (Synthetic Minority Oversampling Technique), Resample Filter (Res) to minimize class imbalance issues. The research used 37 metric values as datasets which were publicly available in software metrics and bug repository data of four projects namely Eclipse, Pde, Mylyn, and Equinox. The performance measures were in terms of precision and recall. Experimental results showed that Random forest classifier when combined with the genetic algorithms, performed best in both defect classification and defect severity by showcasing its precision and recall values to be above 90%.

5. COMPARATIVE ANALYSIS

A comparative analysis [9] of the algorithms and machine learning techniques used in the above-sited literature survey greatly helps in understanding the efficiency of the learning algorithms when used in such defect prediction models. Based on the performance measures, we have tabulated a comparison of the machine learning algorithms as shown in Table 1 and Table 2.

Table -1: Defect Classification Models

Author	Experimental Results		
	Algorithms Used	**Evaluation**	**Results**
Romi Satria Wahono[1]	LR, NB, neural network, SVM, k* classifiers	AUC	LR had the highest value of AUC
David Bowes et al[2]	RF, NB, RPart and SVM	F-measure and Matthews' Correlation Coefficient (MCC)	All algorithms performed best when applied to different datasets
Cagatay Catal[3]	Low-density separation (LDS), SVM, Expectation-Maximization (EM-SEMI), and Class Mass Normalization (CMN)	ROC AUC	SVM and LDS algorithms outperform CMN and EM-SEMI algorithms
Nitish Pandey et al[4].	NB, linear discriminant analysis, KNN, SVM with various kernels, DT and RF	F-measure, average accuracy and weighted average f-measures	RF performed best while SVM with some kernels also achieved high performance

Table -2: Defect Prioritization Models

Author	Experimental Results			
	Feature Selection	**Algorithms Used**	**Evaluation**	**Results**
Aladdin Baarah et al [5]	Information Gain (IG)	NB, NBM, SVM, DT (J48), RF, LMT, Decision Rules (JRip) and KNN	F-measure and Area Under the Curve (AUC).	LMT achieved better performance with Accuracy= 86.31, AUC= 0.90, F-measure= 0.91
Lian Yu et al [6]	Extracted during data preprocessing	ANN and Bayes algorithm	Recall (R), Precision (P) and F-measure (F)	ANN model performed better
Gitika Sharma et al[7]	Text-Document matrix (TDM), Info-Gain and Chi Square	NBM and KNN	Accuracy and Precision	KNN classifiers showed better performance
Ishrat-Un-Nisa Uqaili et al[8]	Extracted during data preprocessing	RF, Multilayer Perceptron (MLP) and NB	Precision and Recall	RF performed best with precision and recall values above 90%

6. CONCLUSION

This paper presented a comparative study of the most commonly used machine learning algorithms and highlighted some of the key factors such as feature selection methods and performance metrics to be considered while developing a defect classification and prioritization model. By citing the literature survey of some of the existing models, this study comprehends the various approaches used for building a classification and prioritization model. The comparative analysis of the performance evaluation shows that Decision Tree(DT) and Support Vector Machine have performed well for most cases in Defect Classification models, while Random Forest (RF) and K-nearest neighbor (KNN) algorithms along with feature selection methods have performed well in most cases of Defect Prioritization models. However, each algorithm behaves contrarily to different problems and different data types making it difficult to recommend one algorithm that best suits a defect prediction model. Hence, this study suggests that the choice of the algorithm be made specific to the problem statement.

REFERENCES

[1] Romi Satria Wahono, Nanna Suryana Herman, Sabrina Ahmad; Article in Advanced Science Letters. 20. 1945-1950. 10.1166/asl.2014.5640

[2] David Bowes, Tracy Hall, Jean Petri´c; Software Qual J (2018) 26, pp.525–552, Published online: 7 February 2017

[3] Cagatay Catal; Journal of Intelligent Systems 2014; vol. 23, no. 1, pp. 75–82.

[4] Nitish Pandey, Debarshi Kumar Sanyal, Abir Hudait, Amitava Sen; Innovations Syst Softw Eng 13, pp. 279–297 (2017), published on 24 July 2017

[5] Aladdin Baarah, Ahmad Al-oqaily, Mannam Zamzeer, Zaher Salah, Mohammad Sallam; International Journal of Advanced Computer Science and Applications, Vol. 10, No. 8, 2019

[6] Lian Yu, Wei-Tek Tsai, Wei Zhao, Fang Wu; ADMA 2010, Part II, LNCS 6441, pp. 356–367, 2010, Springer-Verlag Berlin Heidelberg 2010

[7] Gitika Sharmaa,, Sumit Sharmaa, Shruti Gujrala; 4th International Conference on Eco-friendly Computing and Communication Systems, ICECCS 2015; Procedia Computer Science 70 (2015); pp: 632 – 639

[8] Ishrat-Un-Nisa Uqaili and Syed Nadeem Ahsan; The International Arab Journal of Information Technology, Vol. 17, No. 1, January 2020

[9] T. Sathya Priya and Dr. T. Ramaprabha; “A Comparative Study on the Methods Used for the Detection of Breast Cancer”; International Journal on Recent and Innovation Trends in Computing and Communication; Volume: 5 Issue: 9;pp: 143 – 147