IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 1, January 2024

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.423

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 1, January 2024 ||

| DOI:10.15680/IJIRSET.2024.1301014 |

Software Defect Prediction Analysis using Decision Tree Machine Learning Technique

Seema Rani[1], Prof. Suresh. S. Gawande[2]

M. Tech. Scholar, Department of Electronics and Communication, Bhabha Engineering Research Institute, Bhopal, India[1]

Guide, Department of Electronics and Communication, Bhabha Engineering Research Institute, Bhopal, India [2]

ABSTRACT: - Software Defect Prediction (SDP) is a dynamic research field in the software industry. A quality software product results in customer satisfaction. However, the higher the number of user requirements, the more complex will be the software, with a correspondingly higher probability of failure. SDP is a challenging task requiring smart algorithms that can estimate the quality of a software component before it is handed over to the end-user. Focus of the paper is on predicting software defects (SD) based on Machine learning (ML) techniques which is a challenging research area because of unbalanced nature of data sets. In general, a set of design metrics is used for fault prediction and for identifying the fault-proneness in software and recent studies shows that ML techniques is being applied for defect prediction. In this paper, we propose a decision tree approach to address this particular issue. Our approach combines the feature selection capability of the Optimized Networks algorithm with benchmark machine-learning classifiers for the better detection of bugs in software modules. The proposed approach is implemented python software and calculated precision, recall and accuracy.

KEYWORDS: - Software Defects, Accuracy, Precision, Recall, Decision Tree

I. INTRODUCTION

The development industry has already established deep roots in almost all businesses and industries worldwide. Although software applications have improved the efficiency of business operations, the software development process is nevertheless complex and fraught with difficulties. The complexity of its software makes it failure-prone (full of faults/defects). According to IEEE standards, a fault, bug or defect is considered to be an inappropriate step, process or data in a computer program [1]. A software bug or defect results in software failure, something which occurs when the actual behavior differs from the intended functionality; in other words, there is inconsistency between the actual and the required functionality of the software [1]. To overcome these issues, a great deal of effort is required in terms of maintaining software quality (defect-free products) and ensuring user satisfaction. According to Zhang, a large number of software systems have many defects. According to the World Quality Report (WQR 2020), end-user satisfaction is now a top priority for Quality Assurance (QA) and testing. The report indicates that 27% of the total development cost was spent on testing and quality assurance and this is in fact expected to rise soon to 32% of the total information technology (IT) budget [2]. Over the past few years, numerous studies focusing on SDP models have been reported in the literature. SDP is a means of locating bugs in the software being developed so that the quality and productivity can be improved. Defect prediction models forecast the defective modules in software which are likely to be more error-prone in the future, and the models also help to determine the impact of these defective modules. The building of defect prediction models involves training a machine-learning model using software metrics to predict the defects in software units. Generally, the aim of these models is to forecast the software quality using a set of directly-measurable internal software metrics, such as size and complexity metrics [3]. Software metrics are called ‘features’ or ‘attributes’ that themselves are independent explanatory variables. The effectiveness of the trained model is directly dependent on the worth of these features. Extraneous and redundant features reduce the performance of the prediction models. The greater the number of features in the model, the greater is the complexity, and the less accurate is the model’s performance. To address this issue, we applied Opt-aiNet as an FS technique, in conjunction with benchmark machine-learning classifiers for a better prediction of software bugs. We experimented with six machine-learning classifiers, namely, a Support Vector Machine (SVM), a K-Nearest Neighbor (KNN), a Naive Bayesian (NB), a Decision Tree (DT), a Linear Discriminate Analysis (LDA) and a Random Forest (RF), and then presented the results in terms of accuracy.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 1, January 2024 ||

| DOI:10.15680/IJIRSET.2024.1301014 |

FS is the process of selecting relevant features and eliminating noisy and less important features for model training. Our proposed technique reduces the model's complexity and improves its prediction accuracy. Our proposed method identifies and selects a useful subset of the features which represent patterns from a larger set of mutually-redundant features with different associated measurement risks. To date, much work has been done in the field of SDP but, regrettably, the importance of FS for building consistent and highly-performing prediction models is often overlooked. Some of the prediction models are based on metrics computed using data taken from the change and defect history of software products to predict those components that are expected to be more error-prone than the others [3]. The performance of the SDP model is usually influenced by the characteristics of the attributes/features [4]. However, a set of standard features that can be used for defect prediction has yet to be finalized. The performance of these models increases if an FS technique is applied to eliminate the irrelevant features from the data set [5,6].

Recently, researchers have used different approaches, such as metrics (feature) based defect prediction, optimization schemes, machine-learning techniques and hybrid techniques (a combination of optimization and machine-learning). The field of metric-based defect prediction, which includes both filter and wrapper methods, is considered to be a promising technique for prediction purposes [7]. We believe that, although powerful prediction models have been proposed, there is nevertheless still room for improvement in the prediction quality of models. Therefore, organizations are still curious about methods or techniques that could improve the prediction quality of their models.

A number of models have been proposed to address the FS issue based on computational intelligence approaches, such as genetic algorithms (GA) Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), etc. [8]. Artificial Immune System (AIS), a sub-field of computational intelligence is emerging as one of the most promising approaches for solving complex computational or engineering problems [9]. Opt-aiNet is a biology-inspired algorithm from the AIS family and was first proposed by de Castro et al. [9]. Opt-aiNet consists of a network of antibodies that are similar to population in GA. It also has selection and mutation methods similar to those of GA. All Opt-aiNet, PSO and ACO have a memory set [9,10]. Besides having these, only Opt-aiNet has several additional features, which gave us the confidence to delve in deeper and use it for our research. The additional features are:

- Dynamically adjustable population size.
- Capability of maintaining many optimal solutions.
- Multiple Interactions.
- Exploration of the search space.
- Defined stopping criteria.

II. SOFTWARE QUALITY AND SD PREDICTION

2.1 Software Quality

In computer science, software pertains to the processing of data by hardware, programs and other data to obtain information. Software has been described as a program developed in segments with source code. It is a set of agents handling complexity, testability, visibility, changeability, conformity, reliability and traceability [9]. Software has become the foundation of modern technology; it constitutes or controls the products and services that human beings rely on for a wide variety of daily activities, from the crucial to the trivial. It therefore refers to measurable characteristics. Juran's defines quality as a product feature that satisfies customer needs, providing customer satisfaction. The American Society for Quality (ASQ) defines quality as the foremost characteristics of the product and it must endure the quantified and inferred needs and the software distributed to the user community should be free of defects and scarcities. Software characteristics are measured with respect to its complexity, cohesion, streaks of code, amount of function points and other factors. Quality needs to be measured which imply to deliver a product designed in a stated way. When we design a product, the quality of the software designed must pertains to the features specified for that product by the designer [10].

Then, it is said to follow the conformance quality. Robert Glass believes that user satisfaction is the most significant factor for measuring quality by establishing the formula and is given by equation 1:

Satisfaction of the user = Submissive of the Product + software excellence + product provisioning within the schedule and budget (1)

Software quality is said to be as, the degree to which the product obeys its explicitly stated functional and non-functional requirements. Quality is an established benchmark for the following main reasons [11].

1. It is a must for existence.
2. It provides competitive edge.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 1, January 2024 ||

| DOI:10.15680/IJIRSET.2024.1301014 |

3. It reduces costs in the longer term.
4. It increases market scope.
5. It enables customer retention.

Although these requirements indicate that quality contributes to success, it is difficult to define and impossible to measure even as it is easily recognizable. Moreover, IEEE defines software quality and its assertion as:

1. It is defined as a scheduled and organized outline of the complete activities which are essential to offer the needed assurance that a product or the software adapts to the proven and specified strict standards and requirements.
2. A list of accomplishments or actions developed or manufactured. [12].

Waman believes that a successful project gives rise to customer satisfaction. It was advocated that meeting customer expectations leads to quality. Software quality assurance is achieved through designing test cases and using them as a control measure to ensure desired quality. Quality Assurance of the Software is defined as a list of events intended to appraise the product that will also help us to develop and maintain the software [13].

III. PROPOSED METHODOLOGY

The distribution in machine learning builds the module based on the training dataset with a classification algorithm. This learning can be categorized into all three possible classification algorithms. In a supervised learning class, labeled data is present at the beginning. In semi-supervised learning, some of the class labels are known. Whereas in unsupervised learning no class label for the entire dataset. Once the training phase is finished, features are extracted from the data based on term frequency, and then the classification technique is applied.

DT:-

A DT is a choice help instrument that utilizes a tree-like model of choices and their potential results, including chance occasion results, asset expenses, and utility. It is one method for showing a calculation that just holds back restrictive control explanations. DT are ordinarily utilized in tasks research, explicitly in choice examination, to assist with recognizing a technique probably going to arrive at an objective, but at the same time are a well-known device in ML.
A decision tree classifier is a simple but powerful classification model. It is a supervised learning model having a flowchart like structure that creates a training model from the dataset. A decision tree is a tool for the detection of patterns, relationships, and knowledge from data. Decision trees give a straightforward visualization of data. These are easy to interpret for decision-making in real-time applications. There are different issues related to training with large data sets and the construction of decision trees. To handle these problems different methods have been proposed in the literature by different researchers. These approaches related to decision trees are summarized in this chapter. The emphasis is given on the approaches to create accurate and optimized decision trees. A decision tree is a supervised classifier having a tree structure made up of a set of nodes. These nodes are categorized into internal nodes and leaves. Internal nodes are also called decision nodes because attributes are tested at these nodes to take a decision and partition the data set. Leaves are used to represent predicted class or decision class. In a decision tree construction, splitting criteria is used for partitioning attributes.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 1, January 2024 ||

| DOI:10.15680/IJIRSET.2024.1301014 |

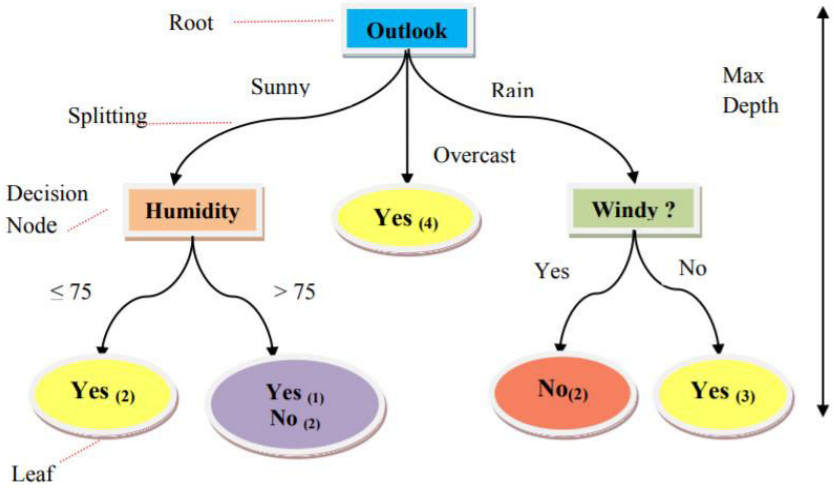

Figure 1: Simple Decision Tree

Decision tree algorithms divide the dataset into training and testing datasets. It uses a recursive method to construct trees. It recursively partitions a training set to create a trained classifier. The recursion process is completed when the subset at a node has the same prediction value as the target variable. A training dataset consists of a set of instances where each instance is made up of attributes and a class label. An attribute can have ordinal, real, or boolean values.

While constructing a decision tree, decision nodes are created which represents a test on selected attributes. For each possible output of the test, one branch from the decision node is created. Hence, the number of branches from the decision node is equal to the possible outputs of the test at that decision node. These decision nodes are called internal nodes (denoted by rectangles). This partitioning of the tree ends at leaf nodes (denoted by ovals). Each leaf in the decision tree represents a predicted class. The decision tree classifier uses a greedy method to construct a tree.

The greedy approach selects a globally optimal solution at each stage. The selection made by a greedy approach may depend on choices made so far in the construction of a tree. These choices do not depend on future choices or all the solutions of the sub problem. In each iteration, it makes one greedy choice after another. This leads to reducing each specified problem into a smaller one [16].

IV. SIMULATION RESULTS

In this test case, we considered other standard classification scheme such as Support vector machine (SVM), and decision tree (DT) classifier.

Data set: KC1 dataset contains total 2109 modules which are given in the table 1 as the predicted outcome using SVM classifier in terms of defective (D) and non-defective (ND) classes.

Table 1: KC1 Prediction for SVM

Actual class	Predicted class	
	ND	D
D	205	129
ND	203	1572

It is really tough to find a better separation between two groups. Certainly, the results obtained from the tests may overlay with one another. Sometimes, the defective module will be correctly classified as True Positive instances (TP)) and also in certain cases the defect will be classified negative instances (FN). Occasionally, without defect will be correctly classified as True negatives (TN) but sometimes, without defect will be classified as Positive (FP).

The above analysis shows that, when the threshold is low at a point both true positive fraction (TP) and sensitivity will rise and at times FP will also increase and therefore a decrease will be there in TN and in specificity.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 1, January 2024 ||

| DOI:10.15680/IJIRSET.2024.1301014 |

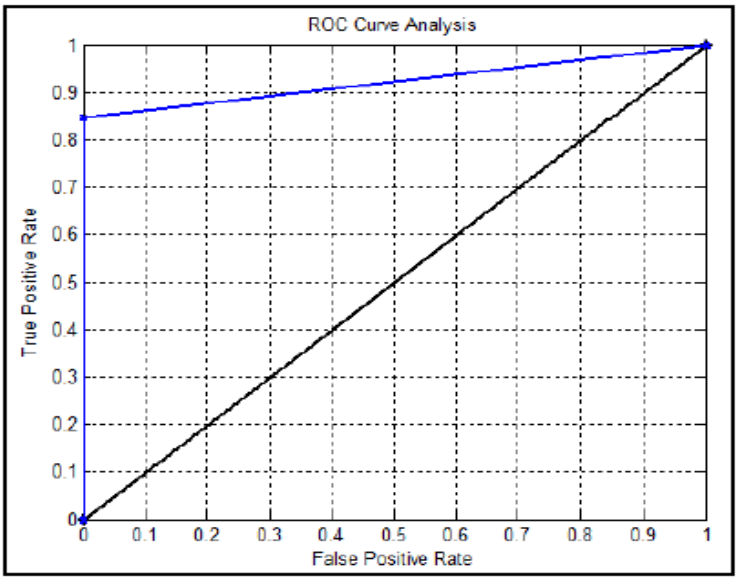

Figure 2: ROC Analysis of KC1 using SVM

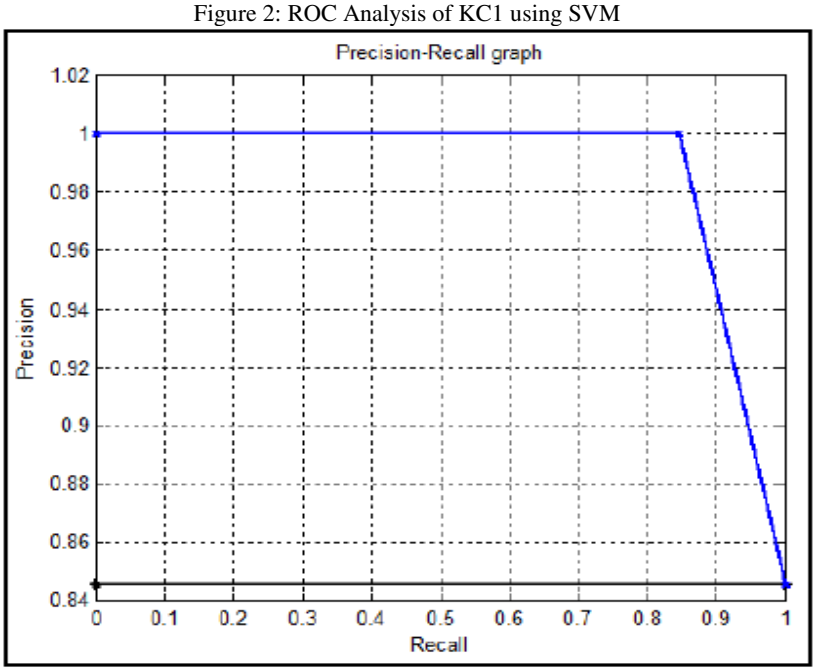

Figure 3: Precision and Recall Analysis of KC1 using SVM

Table II: KC1 Prediction for DT

Actual class	Predicted class	
	ND	D
D	178	201
ND	68	1662

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 1, January 2024 ||

| DOI:10.15680/IJIRSET.2024.1301014 |

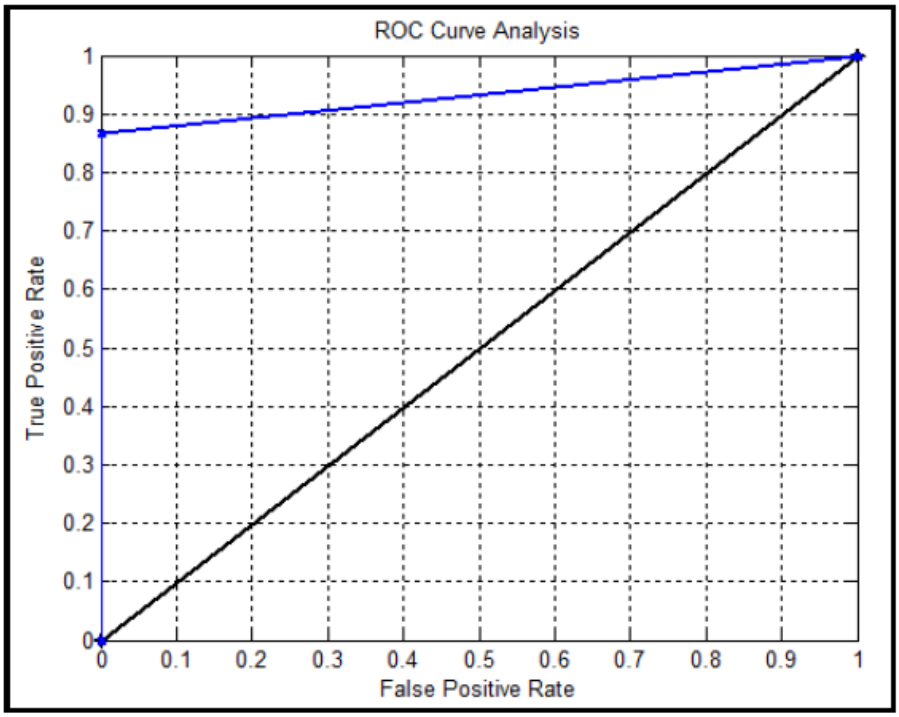

Figure 4: ROC Analysis of KC1 using DT

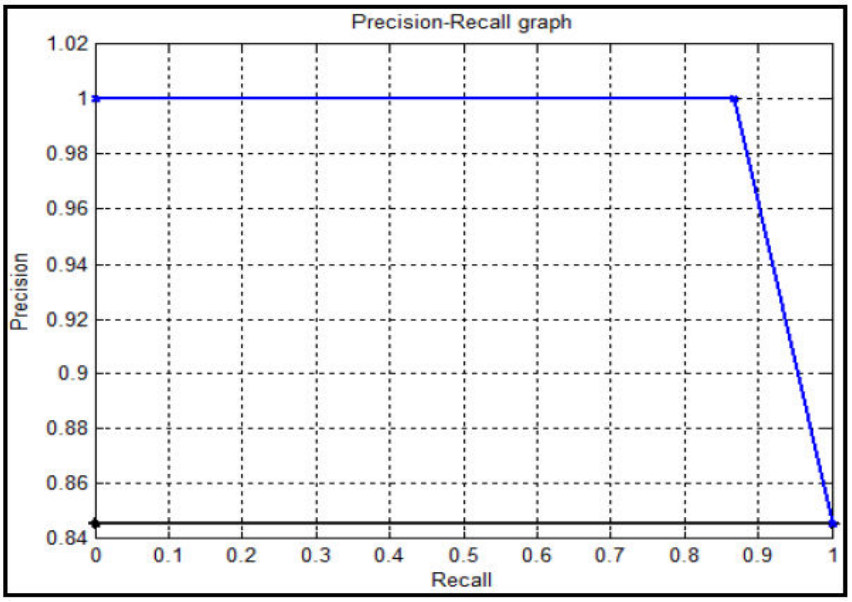

Figure 5: Precision and Recall Analysis of KC1 using DT

V. CONCLUSION

Our main intention is to use Machine learning based methods to cultivate an automated process framework or a prototype for predicting the defects in the software. The model mends the superiority of the software and identifies the fault prone modules by taking metric data into consideration. Usage of ML techniques is model construction and to identify and to forecast the defects in the software. To understand how metrics can be used by the developers to control, track and understand the software process and its ongoing activities. Then software defect prediction automatically replaces cumbersome traditional methods that sometimes lead to errors, and a system is implemented using DT and SVM ML technique.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 1, January 2024 ||

| DOI:10.15680/IJIRSET.2024.1301014 |

REFERENCES

[1] Iqra Mehmood, Sidra Shahid, Hameed Hussain, Inayat Khan, Shafiq Ahmad, Shahid Rahman, Najeeb Ullah and Shamsul Huda, "A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning", IEEE Access 2023.

[2] L.-Q. Chen, C. Wang, and S.-L. Song, ''Software defect prediction based on nested-stacking and heterogeneous feature selection,'' *Complex Intell. Syst.*, vol. 8, no. 4, pp. 3333–3348, Aug. 2022

[3] M. Pavana, L. Pushpa, and A. Parkavi, ''Software fault prediction using machine learning algorithms,'' in *Proc. Int. Conf. Adv. Elect. Comput. Technol.*, 2022, pp. 185–197

[4] A. Al-Nusirat, F. Hanandeh, M. K. Kharabsheh, M. Al-Ayyoub, and N. Al-Dhfairi, ''Dynamic detection of software defects using supervised learning techniques,'' *Int. J. Commun. Netw. Inf. Secur.*, vol. 11, no. 1, pp. 185–191, Apr. 2022.

[5] R. Bahaweres, F. Agustian, I. Hermadi, A. Suroso, and Y. Arkeman, ''Software defect prediction using neural network basedSMOTE,'' in *Proc. 7th Int. Conf. Electr. Eng., Comput. Sci. Informat. (EECSI)*, Oct. 2020, pp. 71–76

[6] N. Li, M. Shepperd, and Y. Guo, ''A systematic review of unsupervised learning techniques for software defect prediction,'' *Inf. Softw. Technol.*, vol. 122, Jun. 2020, Art. no. 106287.

[7] Y. Qiu, Y. Liu, A. Liu, J. Zhu, and J. Xu, ''Automatic feature exploration and an application in defect prediction,'' *IEEE Access*, vol. 7, pp. 112097–112112, 2019.

[8] A. Alsaeedi and M. Z. Khan, ''Software defect prediction using supervised machine learning and ensemble techniques: A comparative study,'' *J. Softw. Eng. Appl.*, vol. 12, no. 5, pp. 85–100, 2019.

[9] C. Manjula and L. Florence, ''Deep neural network based hybrid approach for software defect prediction using software metrics,'' *Cluster Comput.*, vol. 22, no. S4, pp. 9847–9863, Jul. 2019.

[10] R. Jayanthi and L. Florence, ''Software defect prediction techniques using metrics based on neural network classifier,'' *Cluster Comput.*, vol. 22, no. S1, pp. 77–88, Jan. 2019.

[11] A. Hammouri, M. Hammad, M. Alnabhan, and F. Alsarayrah, ''Software bug prediction using machine learning approach,'' *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 2, pp. 78–83, 2018.

[12] N. Kalaivani and R. Beena, ''Overview of software defect prediction using machine learning algorithms,'' *Int. J. Pure Appl. Math.*, vol. 118, pp. 3863–3873, Feb. 2018.

[13] M. A. Memon, M.-U.-R. Magsi, M. Memon, and S. Hyder, ''Defects prediction and prevention approaches for quality software development,'' *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 8, pp. 451–457, 2018.

[14] E. Naresh and S. P. Shankar, ''Comparative analysis of the various data mining techniques for defect prediction using the NASA MDP datasets for better quality of the software product,'' *Adv. Comput. Sci. Technol.*, vol. 10, no. 7, pp. 2005–2017, 2017.

[15] D. Kumar and V. H. S. Shukla, ''A defect prediction model for software product based on ANFIS,'' *Int. J. Sci. Res. Devices* vol. 3, no. 10, pp. 1024–1028, 2016.

[16] P. Mandal and A. S. Ami, ''Selecting best attributes for software defect prediction,'' in *Proc. IEEE Int. WIE Conf. Electr. Comput. Eng.*, Dec. 2015, pp. 110–113.

[17] M. C. M. Prasad, L. F. Florence, and A. Arya, ''A study on software metrics based software defect prediction using data mining and machine learning techniques,'' *Int. J. Database Theory Appl.*, vol. 8, no. 3, pp. 179–190, Jun. 2015.

[18] A. Chug and S. Dhall, ''Software defect prediction using supervised learning algorithm and unsupervised learning algorithm,'' in *Proc. 4th Int. Conf. Confluence Next Gener. Inf. Technol. Summit*, Noida, 2013, pp. 173–179.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details