A Comparative Evaluation of Machine Learning based Software Fault Prediction Models using Principle Component Analysis

[1] Harsimran Kaur, [2] Hardeep Singh,[3]Amitpal Singh Sohal
[1]Research Scholar, [2] Professor, [3]Assitant Professor
[1] Department of Computer Science, GNDU, Amritsar
[2] Department of Computer Science, GNDU, Amritsar
[3]Department of Computer Engineering and Technology, GNDU RC, Gurdaspur
[1] harsimrankaur004@@gmail.com, [2] hardeep.dcse.gndu.ac.in, [3]amitpal.csegsp@gndu.ac.in

ABSTRACT

Software fault prediction assists in identifying flaws in the early stages of software development and makes software testing more convenient and reliable. This study investigates the effect of the Principle Component Analysis techniques on software fault prediction models. It empirically compares the performance of six machine learning classifiers: Naïve Bayes, Random Forest, Logistic Regression, Support Vector Machine, Gradient Boosting, and Decision Tree with and without Principle Component Analysis (PCA). Furthermore, this paper aims to measure the capability of the software fault predictability in terms of accuracy, precision, f1 score, and AUC. The Area under Curve (AUC) Receiver Operating Characteristic (ROC) is used to check the validity of the models. The CAMEL dataset is used, which contains twenty-one Object-Oriented Software metrics available on the PROMISE repository. The Comparative evaluation indicates that all classifiers performed well with the Principal Component Analysis technique, whereas Random Forest and Decision Tree outperformed other classifiers.

Keywords: machine learning, fault prediction, principle component analysis, AUC

INTRODUCTION

Testing is essential in software development because it reveals hidden errors and ensures software quality. Although it is impossible to create a fault-free software system, the number of errors can be reduced by thoroughly inspecting the software during the testing phase. The testing process requires a high amount of resources in terms of time and effort to diminish the fault list [1]. Overall, identifying and correcting faults require around 50% of the project budget [2]. The authors claimed that the testing process consumes more than half of the total software development budget, which may further arise in the case of critical software systems. As a result, testers must implement practical strategies to cover more and more errors in less time. Identification of software faults in the early stage of the Software Development life cycle is known as the Software fault prediction process. Software fault prediction (SFP) is developing a model that software developers can utilize to detect the faulty class model before the testing [3]. Software fault prediction models are often built using the fault dataset from earlier releases of related software projects and utilized to forecast problems in the software system currently being developed. Some structural elements are used to detect faulty classes in the software. To reduce software failure, fault prediction aids in organizing, controlling, and executing software development activities and classifying any software module as faulty or non-faulty, eventually increasing software efficiency and effectiveness. The SFP also decreases the amount of time and effort that must be spent on software [4].

Machine learning techniques are becoming increasingly popular in the software industry for predicting early faults. It includes Decision Trees, a Bayesian approach, Support Vector Machine, Random Forest, and a neural network. The performance of these classifiers is entirely dependent on the training dataset fed to them [5][3]. A machine learning algorithm is utilized for software fault prediction, which works on the principle of automatically learning from training data sets and providing the best results with test data sets [6]. Machine learning algorithms that have been supervised are fed a predefined dataset of training data. The algorithms learn from the training dataset and generate rules to predict the class label for new data sets [7].

Principle Component Analysis (PCA) is used for dimensionality reduction and it is an unsupervised linear transformation technique that also improves the classification outcome of the model [8]. PCA is the most widely used and popular dimension reduction technique Because of its calculation flexibility and reversible approach. It is accomplished by eliminating less significant attributes in high dimensional space and attributes into a low dimensional subspace, which speeds up the computation time of the model during training the dataset [9]. It is also a feature selection technique used to produce strong patterns in datasets in the fault prediction to select the feature of OO metrics [10]. This technique leads to the formulation of the following research questions:

RQ1: How effective is the PCA technique for Fault Prediction Models?
RQ2: Which machine learning algorithms perform best with the PCA technique?

In this paper, C&K metrics are used and its suite has been used the majority of times in the prediction studies for Object Oriented systems. It also observed that machine learning techniques provide better results than classical methods [11].In this study; we used the CAMEL dataset, which includes 21 Object-Oriented Software Metrics available on the PROMISE repository.

The rest of the paper is organized as follows: Section 2 provides the overview of the related works; Section 3 discusses the proposed methodology, Section 4 for evaluating predicting models and presents the results of the comparative evaluation, and in the last Section 5 gives the conclusion and future work of the paper.

RELATED WORK

C. Lakshmi Prabha [12] defines the hybrid approach that includes the PCA, random forest, Naïve Bayes, and the SVM Software framework applied on five different datasets and evaluates the results using the WEKA simulation tool. In this study, the author introduced to acquisition of a feature reduction template, where overall probability is also applied to minimize the data recovered by PCA. In this work, performance measure parameters are used, namely confusion, precision, recall, recognition, and accuracy, compared with the prevailing schemes. The analytical analysis indicates that the hybrid approach will provide more valuable solutions for defect prediction.

M. Massoudi et al.[8] Studied software defect prediction using dimensionality reduction and deep learning. The author used NASA Promise Data Repository to evaluate the comparative study. The model performance evaluates through Accuracy, F1 scores, and Areas under the Receiver Operating Characteristic curve. The results show that Kernel PCA and Artificial Neural Network as a classifier outperformed all the other techniques.

K. J. Chabathul [9] highlight the importance of various machine learning techniques developed to date for identifying various network attacks and recommends a suitable Intrusion Detection System (IDS) using the available system resources while optimizing speed and accuracy. Support Vector Machines (SVM), K-Nearest Neighbors (KNN), J48 Tree algorithm, Random Forest Tree classification algorithm, Ad boost algorithm, Nearest Neighbors generalized Exemplars algorithm, and Naïve Bayes probabilistic classifier are used to test the reduced dimension dataset. The experimental results show that the with-PCA-based approach takes less time to detect than the without-PCA-based approach.

G. P. Bhandari and R. Gupta [13] describe the utilization of software metrics to predict the faults of the software. The capability of software fault predictability is measured in terms of accuracy, f-measure, precision, recall, and reliability. The research looks into the impact of feature selection techniques on software fault prediction. In most cases, the result predicted by the Random Forest technique outperforms the results expected by other machine learning techniques.

This study further extends the above contributions by considering the performance parameters' specificity to check the proposed approach's validity. We have regarded as specified dataset and experimented with them using several machine learning techniques along with Principle Component Analysis to measure the capability of the machine learning techniques for software fault prediction models.

METHODOLOGY

In this section, we describe machine learning techniques (section A), Principle Component Analysis (section B), datasets description (section C), and performance evaluation metrics (section D), and machine learning algorithms and techniques to create Fault Prediction Models (section E) given below:

A. Techniques used

Twenty- Object-Oriented Software Metrics are used to predict the faults with the help of the following ML techniques:

- **Logistic regression:** It is a supervised learning classification approach used for binary data classification, and the view involves a more probabilistic type [14][15]. Fast Classify unknown records and gives good accuracy for the linearly separable data set [16].

- **Decision Tree (DT):** A decision tree represents a simple learning technique in data mining. It works both ways forward and reverses to improve accuracy [17]. It is a tree-based classifier where internal nodes indicate the dataset's characteristics, branches for the decision-making processes, and leaves for the results [18].
- **Support Vector Machine (SVM):** This classifier considers different sample points in space mapped to the class to which they belong, with the most significant separation from other types [17].

- **Naïve Bayesian:** The Nave Bayes classification method is the most well-known and widely used. Classification is assigning a class label to a sample from a given set of labelled samples [19]. It is an efficient and straightforward probabilistic classifier [17].

- **Random Forest:** Random Forest is a popular supervised learning machine learning algorithm. It is used for both classification and regression problems in machine learning. It's based on the idea of ensemble learning, which is the process of combining multiple classifiers to solve a complex issue and improve the model's performance [20].

- **Gradient Boosting:** Gradient Boosting Machine algorithm is used for both continuous and categorical target variables. It provides predictive accuracy and flexibility [21]. This model is proposed to predict the failure and its classification superiorities. It is widely researched, has high accuracy performance, fast training/testing, and sound generalization across many different data sets [22].

B. Principle Component Analysis (PCA)

PCA is a technique for extracting essential variables from a large set of variables in a data set. It removes a low-dimensional set of features from a high-dimensional data set by projecting outside dimensions onto it to capture as much information as possible. With fewer variables obtained, minimizing information loss, visualization becomes significantly more meaningful, and also when dealing with three or more data dimensions [23].

C. Dataset Description

In this work, the dataset was gathered from the PROMISE repository for evaluating the models and considered datasets include data from Open Source Software Systems such as CAMEL. The dataset contained Twenty-one Object-Oriented Software Metrics, which are used to identify the faults in each Module. The dataset is categorized into faulty and non-faulty classes. The defective Module was marked as one, and the non -faulty Module scored 0 in the given dataset.

Features Name	**Abbreviations**	**Features Description**
WMC	Weighted Methods for Class	It indicates a weighted method per class
DIT	Depth of Inheritance Tree	It measures the depth of the inheritance tree
NOC	Number of Children	It indicates cohesion in methods
CBO	Coupling between Objects	It indicates coupling between object classes
RFC	Response For Class	It indicates the response to class
LCOM	Lack of Cohesion of Methods	It defines a Lack of cohesion in methods
CA	Afferent coupling	It defines an afferent coupling
CE	Efferent coupling	It defines an efferent coupling
PM	public methods	It defines the number of public methods for class also known
LCOM3	Lack of Cohesion of Methods	It is a normalized version of LCOM having a value between 0 and 2
LOC	lines of code	It measures the lines of code
DAM	Data access metric	It indicates the data access metric
MOA	measure of aggregation	It indicates the measure of aggregation

Features Name	Abbreviations	Features Description
MFA	Measure of functional abstraction	It evaluates the measure of functional abstraction
CAM	Cohesion among methods	It indicates the cohesion among methods of a class
IC	Inheritance coupling	It defines the inheritance coupling
CBM	Coupling between methods	It measures the coupling between methods
AMC	Average method complexity	It evaluates the complexity of the average method
MAX_CC	Maximum cyclomatic complexity	Maximum cyclomatic complexity
AVG_CC	Average cyclomatic complexity	It evaluates the average cyclomatic complexity
BUG	Reported Fault	It indicates whether the module has any reported fault or not

Table (1) - Description of the features present in the Dataset

D. Performance Evaluation Metrics

- **Accuracy:** It defines the overall performance of the model and the ratio of the number of modules correctly predicted to the total number of modules. It is calculated as follows:

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

- **Precision:** It is used to identify the portion of the correctly predicted faulty modules out of all modules that are predicted faulty. It is defined as:

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

- **Recall:** It is used to identify how many correct faulty modules are predicted. It is defined as:

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

- **F1-Score:** The F1-score combines a classifier's precision and recall into a single metric by calculating their harmonic mean. In F1-score where the best possible score is 1 and the worst possible score is 0. The F1 Score is calculated as follows:

$$F1\ Score = \frac{2*(precision+Recall)}{Precision+Recall} \quad (4)$$

- **AUC and Confusion Matrix:** It stands for the area under the receiver operating characteristic curve. It is a graphical representation or plot that depicts the diagnostic capabilities of a prediction model under different threshold values. It plots the true positive rate on the y-axis and the false positive rate on the x-axis. The area under the curve shows the probability that a classifier will classify a randomly chosen positive module higher than a randomly chosen negative module.

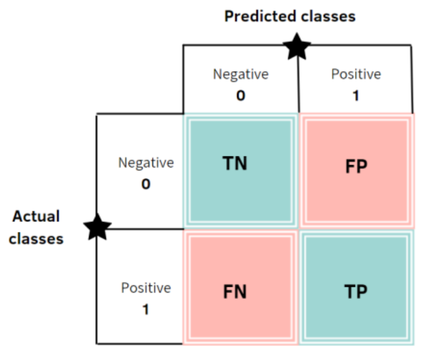


Fig- 1. Confusion Matrix [8]

E. Using Machine Learning Algorithms and techniques to create Fault Prediction Models

In this section, the following research methodology is proposed based on the steps carried out by a numerical study to predict and classify software faults. The stepwise approach is as follows:

Step I: First, access the dataset from the Object Oriented metrics Camel Dataset available on the PROMISE repository.
Step II: First, use the Over Sampling method to balance the imbalanced data set.
Further, split each dataset into training and testing datasets and validate the proposed model.
Step III: Construct the model using different simple classification algorithms with the help of a test and training dataset. Further, we construct the classification model with and without Principle Component Analysis (PCA) tool with the training and testing dataset.

Next, we plot the ROC Curve and calculate the AUC to check the validity of our models.
In the last step, we discuss the performance measures of the models, and comparisons have been made.

RESULTS AND DISCUSSION

This study uses six classifiers, namely Naïve Bayes, SVM, Logistic Regression, Decision Tree, Gradient Boosting, and Random Forest Tree with and without PCA, to predict the faults using Fault Prediction Model. We compare the results of each classifier with and without the PCA technique. It has been noticed that the classifier produces good results when using the PCA technique. Decision Tree and Random Forest Tree algorithms outperformed other algorithms in terms of performance measurement metrics like Accuracy, precision, F1 Score, and AUC. To assess classifier output quality, use the Receiver Operating Characteristic (ROC) metrics. In the given results, ROC curves typically show the actual positive rate on the Y axis and the false positive rate on the X axis. This means that the top left corner of the plot is the *ideal* point, with a false positive rate of zero and an actual positive rate of one. To test the validity of the performance of our classifier, we plotted the ROC curves and calculated the AUC. The ROC curves for the respective matrices are shown below:

RQ1: How effective is the Principle Component Analysis technique for Fault Prediction Models?

Principle Component Analysis is a popular unsupervised learning technique. Its basic concept is to reduce the dimensionality of a dataset while retaining as much 'variability' as possible. Table 3 demonstrates that fault prediction models perform better with the PCA technique than the results shown in Table 2. It reduces high- to low-dimensional data and shortens the time required for dataset training and testing.

Classifier	**Accuracy**	**Precision Score**	**Recall**	**F1 Score**	**AUC**
Naïve Bayes	0.78	0.79	0.78	0.78	0.800
SVM	0.84	0.84	0.84	0.84	0.849
Logistic Regression	0.82	0.82	0.82	0.82	0.923
Decision Tree	0.91	0.92	0.91	0.91	0.910
Gradient Boosting	0.82	0.83	0.82	0.82	0.926
Random Forest	0.95	0.95	0.95	0.95	0.960

Table (2) - Classifiers Results without Principle Component Analysis (PCA)

Classifier	Accuracy	Precision Score	Recall	F1 Score	AUC
Naïve Bayes+ PCA	0.89	0.89	0.89	0.89	0.893
SVM+PCA	0.86	0.86	0.86	0.86	0.903
Logistic Regression + PCA	0.85	0.85	0.85	0.85	0.944
Decision Tree + PCA	0.98	0.98	0.97	0.98	0.976
Gradient Boosting +PCA	0.95	0.95	0.95	0.95	0.934
Random Forest+ PCA	0.98	0.98	0.98	0.98	0.980

Table (3) - Classifiers Results with Principle Component Analysis (PCA)

RQ2: Which machine learning algorithms performs best with the PCA technique?

For the CAMEL dataset, Random Forest and decision tree perform best with their high performance measure metrics and, the AUC values of the Decision Tree and Random Forest 0.976 and 0.980 with the PCA technique. Further analysis shows that SVM, LR, GB, and NB have significantly improved their performances after the applied the PCA techniques. Figure 2 shows the performance comparisons of different classifiers with and without PCA.

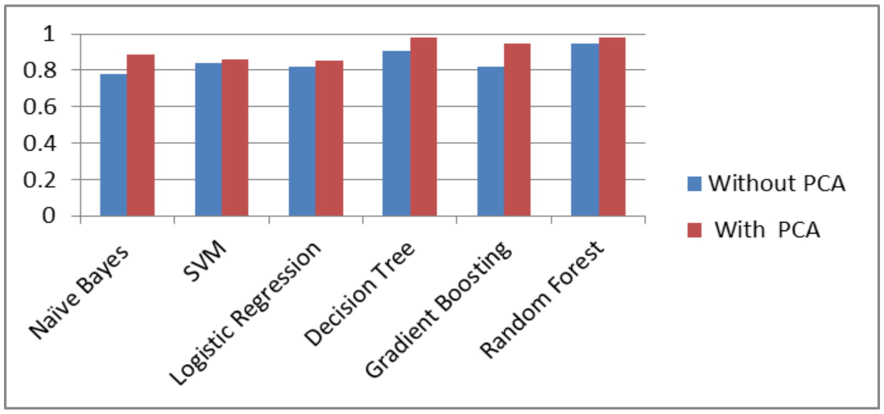
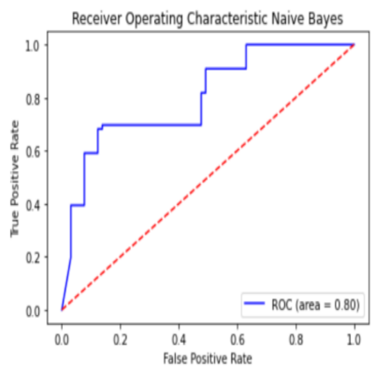


Fig-2. Comparision of AUC values among different classifiers on the CAMEL dataset

To check the validity of our model ,we plot the Receiver Operating Charateristic (ROC)Curves that shown below (Fig. 3 to Fig. 14):



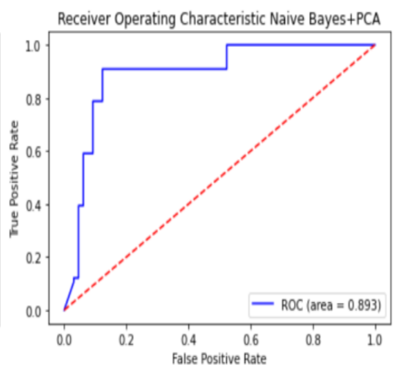
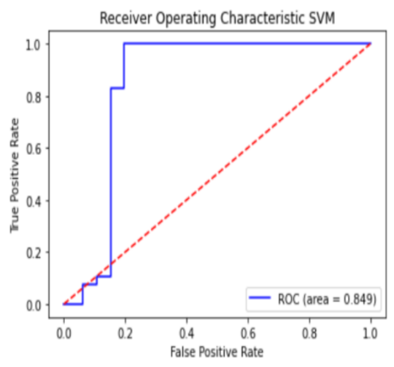


Fig- 3 .Curve for classifier before PCA Fig - 4. Curve for classifier After PCA



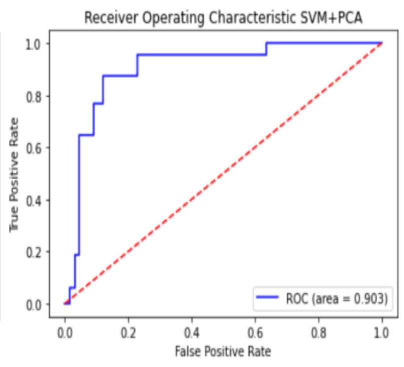
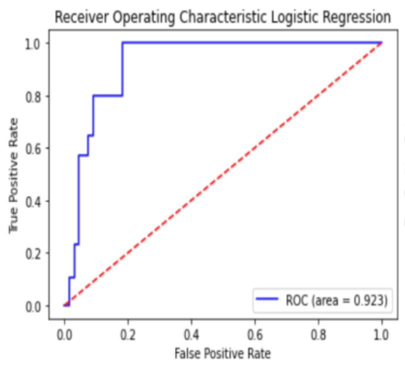


Fig- 5. Curve for classifier before PCA Fig- 6.Curve for classifier After PCA



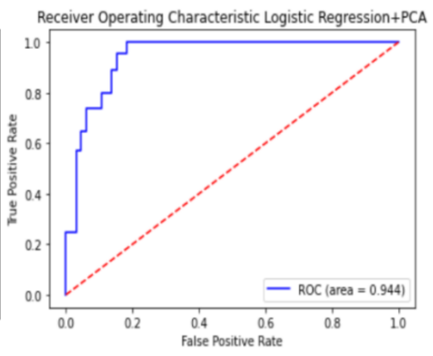
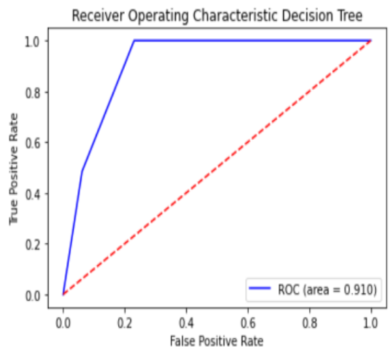


Fig- 7. Curve for classifier before PCA Fig- 8. Curve of the classifier After PCA



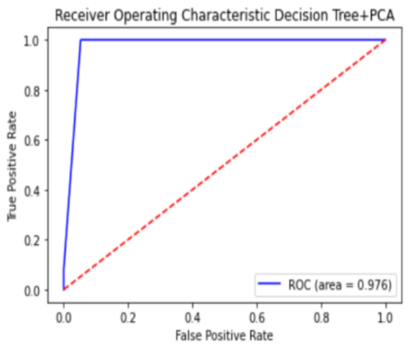
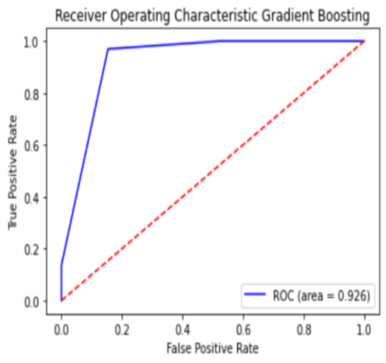


Fig- 9. Curve for classifier before PCA Fig.-10. Curve for classifier After PCA



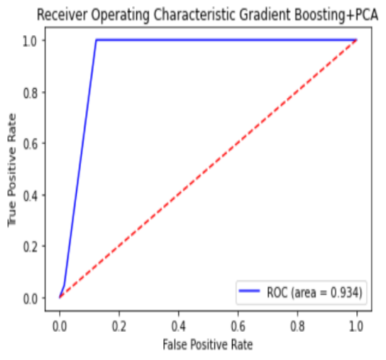
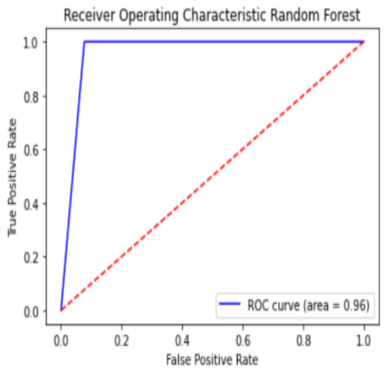


Fig.- 11. Curve for classifier before PCA Fig- 12. Curve for classifier After PCA



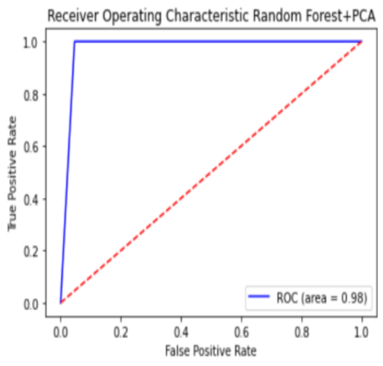


Fig- 13. Curve for classifier before PCA Fig-14. Curve for classifier After PCA

CONCLUSION AND FUTURE WORK

This paper compares with and without-PCA-based fault prediction approaches using six classifiers, likely Support Vector Machines (SVM), Logistic Regression, Decision Tree, Random Forest Tree classification, Gradient Boosting, and Naïve Bayes classifier. It is observed through the experimental result that with-PCA based approach exhibits more accuracy, Precision, F1- Score, and AUC as compared to the simple machine learning approach to predict the faults using Fault Prediction Model. Decision Tree and Random Forest Tree algorithms outperformed all other classifiers. We are planning to test the same experiment by taking large no. of datasets and using the Kernel PCA tool to compare the performance of different classifiers on large datasets.

REFERENCES

[1] Y. J. & A. B. Tim Menzies, Zach Milton, Burak Turhan, Bojan Cukic, “Defect prediction from static code features: current results, limitations, new approaches.” 2010.

[2] T. B. (Author) Glenford J. Myers (Author), Corey Sandler (Author), *The Art of Software Testing*. .

[3] S. K. Pandey, R. B. Mishra, and A. K. Tripathi, “Machine learning based methods for software fault prediction: A survey,” *Expert Syst. Appl.*, vol. 172, no. December 2020, p. 114595, 2021, doi: 10.1016/j.eswa.2021.114595.

[4] O. Al Qasem and M. Akour, “Software fault prediction using deep learning algorithms,” *Int. J. Open Source Softw. Process.*, vol. 10, no. 4, pp. 1–19, 2019, doi: 10.4018/IJOSSP.2019100101.

[5] S. Goyal, “Effective software defect prediction using support vector machines (SVMs),” *International Journal of System Assurance Engineering and Management*, vol. 13, no. 2. pp. 681–696, 2022, doi: 10.1007/s13198-021-01326-1.

[6] I. Kaur and A. Kaur, “Comparative analysis of software fault prediction using various categories of classifiers,” *Int. J. Syst. Assur. Eng. Manag.*, vol. 12, no. 3, pp. 520–535, 2021, doi: 10.1007/s13198-021-01110-1.

[7] N. Babu, Himagiri, V. Vamshi Krishna, A. Anil Kumar, and M. Ravi, “Software defect prediction analysis by using machine learning algorithms.,” *Int. J. Recent Technol. Eng.*, vol. 8, no. 2 Special Issue 11, pp. 3544–3546, 2019, doi: 10.35940/ijrte.B1438.0982S1119.

[8] M. Massoudi, N. K. Jain, and P. Bansal, “Software defect prediction using dimensionality reduction and deep learning,” *Proc. 3rd Int. Conf. Intell. Commun. Technol. Virtual Mob. Networks, ICICV 2021*, no. Icicv, pp. 884–893, 2021, doi: 10.1109/ICICV50876.2021.9388622.

[9] K. J. Chabathula, C. D. Jaidhar, and M. A. Ajay Kumara, “Comparative study of Principal

Component Analysis based Intrusion Detection approach using machine learning algorithms," *2015 3rd Int. Conf. Signal Process. Commun. Networking, ICSCN 2015*, pp. 1–6, 2015, doi: 10.1109/ICSCN.2015.7219853.

[10] A. Singh, R. Bhatia, and A. Sighrova, "Taxonomy of machine learning algorithms in software fault prediction using object oriented metrics," *Procedia Comput. Sci.*, vol. 132, pp. 993–1001, 2018, doi: 10.1016/j.procs.2018.05.115.

[11] A. Kumar and A. Bansal, "Software Fault Proneness Prediction Using Genetic Based Machine Learning Techniques," *Proc. - 2019 4th Int. Conf. Internet Things Smart Innov. Usages, IoT-SIU 2019*, pp. 1–5, 2019, doi: 10.1109/IoT-SIU.2019.8777494.

[12] C. L. Prabha and N. Shivakumar, "Software Defect Prediction Using Machine Learning Techniques," *Proc. 4th Int. Conf. Trends Electron. Informatics, ICOEI 2020*, no. Icoei, pp. 728–733, 2020, doi: 10.1109/ICOEI48184.2020.9142909.

[13] G. P. Bhandari and R. Gupta, "Machine learning based software fault prediction utilizing source code metrics," *Proceedings on 2018 IEEE 3rd International Conference on Computing, Communication and Security, ICCCS 2018*. pp. 40–45, 2018, doi: 10.1109/CCCS.2018.8586805.

[14] M. Maalouf, "Logistic regression in data analysis: An overview," *Int. J. Data Anal. Tech. Strateg.*, vol. 3, no. 3, pp. 281–299, 2011, doi: 10.1504/IJDATS.2011.041335.

[15] S. Pirzada, "Machine Learning and Logistic Regression Umme Salma," *Mach. Learn. Algorithms Logist. Regres.*, no. May, 2020.

[16] L. Connelly, "Logistic regression," *MEDSURG Nursing*, vol. 29, no. 5. pp. 353–354, 2020, doi: 10.46692/9781847423399.014.

[17] B. Murgante *et al.*, *Proceedings - 14th International Conference on Computational Science and Its Applications, ICCSA 2014*. 2014.

[18] "Decision Tree Classification Algorithm," *JavaTpoint*. p. 1, [Online]. Available: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm.

[19] C. Catal, U. Sevim, and B. Diri, "Practical development of an Eclipse-based software fault prediction tool using Naive Bayes algorithm," *Expert Syst. Appl.*, vol. 38, no. 3, pp. 2347–2353, 2011, doi: 10.1016/j.eswa.2010.08.022.

[20] Javatpoint, "Machine Learning Random Forest Algorithm," *Www.Javatpoint.Com*. 2021, [Online]. Available: https://www.javatpoint.com/machine-learning-random-forest-algorithm.

[21] N. Tarbani, "Gradient Boosting Algorithm | How Gradient Boosting Algorithm Works," *This article was published as a part of the Data Science Blogathon*. pp. 1–1, 2021, [Online]. Available: https://www.analyticsvidhya.com/blog/2021/04/how-the-gradient-boosting-algorithm-works/.

[22] D. K. Thai, T. M. Tu, T. Q. Bui, and T. T. Bui, "Gradient tree boosting machine learning

on predicting the failure modes of the RC panels under impact loads," *Engineering with Computers*, vol. 37, no. 1. pp. 597–608, 2021, doi: 10.1007/s00366-019-00842-w.

[23] "Importance of Principal Component Analysis | by Mukesh Chaudhary | Medium." [Online]. Available: https://medium.com/@cmukesh8688/importance-of-principal-component-analysis-e9184a47ffa8.