International Journal of Global Research Innovations & Technology (IJGRIT)
ISSN : 2583-8717, Impact Factor: 6.382, Volume 02, No. 01, January-March, 2024, pp 84-89

SUPERVISED AND UNSUPERVISED MACHINE LEARNING TECHNIQUES TO PREDICT FAULT IN DEVELOPING QUALITY SOFTWARE: A COMPARATIVE STUDY

Nikita Gupta*
Prof. (Dr.) Ripu Ranjan Sinha**

ABSTRACT

Finding bugs in software development is essential to guaranteeing the creation of high-caliber software. Using machine learning approaches has yielded encouraging results in automating defect identification. Since early problem detection reduces the time and cost required for bug patching and maintenance, it is essential for software development. Conventional manual defect detection techniques take a lot of time, are prone to mistakes, and might not be able to handle the complexity and size of contemporary software systems. The software industry now has new tools to improve overall software quality and automate the defect identification process thanks to the development of machine learning techniques. In order to identify errors in software development, this study compares supervised versus unsupervised machine learning techniques. We compare the effectiveness of both methods with real-world software defect datasets and talk about their advantages, disadvantages, and applications. The study's findings provide insight into how effective each method is at identifying mistakes while software is being produced.

Keywords: *Software Fault Detection, Deep Learning, Machine Learning, Software Quality Assurance.*

Introduction

Software errors can have serious repercussions, such as system breakdowns, security lapses, and monetary losses. Ensuring the high quality of software products requires the detection of errors during the software development process. Conventional manual defect detection techniques take a lot of time and are prone to mistakes. With the advent of machine learning techniques, the software industry now has new tools at its disposal to automate defect identification procedures. In order to identify errors in software development, we compare supervised and unsupervised machine learning approaches in this work. This study examines the body of research on machine learning-based software development defect identification. We investigate several supervised learning algorithms that need labeled data for training, like Support Vector Machines (SVM), Random Forest, and Neural Networks. We also look into unsupervised learning methods that can detect abnormalities or fault clusters without explicit fault labels, such as K-means clustering, DBSCAN, and Isolation Forest. Our goal in contrasting these two methods is to offer important perspectives on how well they work in various software development scenarios. The studies' outcomes highlight the benefits and drawbacks of supervised and unsupervised learning strategies for fault identification. Because supervised learning algorithms employ labeled training data, they are more accurate and can clearly identify failure instances. On the other hand, obtaining labeled fault data can be costly and time-consuming—particularly for extensive projects. Unsupervised learning techniques, on the other hand, are more scalable and economical for fault detection since they do not

* Research Scholar, Rajasthan Technical University, Kota, Rajasthan, India.
** Professor, S.S. Jain Subodh P.G. College, Rajasthan Technical University, Kota, Rajasthan, India.

require labeled data. They could, however, have trouble telling apart various defect kinds and result in false positives or false negatives. The selection of proper hyper parameters is critical to the effectiveness of clustering algorithms, but it can be difficult in practice.

Literature Review

A practical evaluation to comprehend software quality and ensure corrective measures is presented in this study [1]. By employing these data mining techniques to create software defect prediction models, the product's quality is improved. This study offered multiple strategies for classification and clustering to forecast software problems. Methods of classification and clustering have been studied to predict software faults. The efficacy of J48, Random Forest, and Naive Bayesian Classifier (NBC), three data mining classifier methods, is evaluated based on several metrics such as ROC, Precision, MAE, RAE, and others. The clustering technique is then applied to the data set using K-means, Hierarchical Clustering, and the Make Density Based Clustering algorithm. Time Taken, Cluster Instance, Number of Iterations, Incorrectly Clustered Instance, Log Likelihood, and other parameters are taken into consideration when evaluating the clustering results. Defect prediction is the end result of a detailed examination of ten real-time NASA software project defect databases, numerous applications run on them, and so on.

The authors of this work [2] incorporated those variables, carried out the required analysis, and provided examples of the numerous challenges in the SFP field. They also assessed the efficacy of machine learning and statistical techniques based on SFP models. The analysis and practical research demonstrate that machine learning approaches outperform conventional statistical models in determining whether a class or module is fault- or non-fault-prone. When it comes to fault susceptibility, machine learning-based SFP techniques perform better than conventional statistical methods. The survey's real data indicates that machine learning techniques have the ability to detect fault tendency and yield findings that are broadly applicable. Additionally, they investigated a few problems related to the fault prediction discipline, including the problem of class imbalance and over-fitting of models and data quality.

The research tool for using an unsupervised learning technique for software failure prediction is presented in the paper [3]. The evaluation of the clustering algorithm's efficacy is the primary goal. The author also assessed the confusion matrix and clustering algorithm's potential. The software's defect status is determined by computing the FPR, FNR, and ERROR faults using a confusion matrix.

In this work, [4] published study outcomes and explained the SFP technique for five eclipse project datasets and nine object-oriented project datasets. They used Sensitivity, Specificity, AUC, and Accuracy to evaluate the performance of the suggested approaches. To investigate the approach's financial viability, a cost-benefit analysis was also conducted. According to the results, the method that was given was able to predict software flaws with the greatest accuracy (0.816%), AUC (0.835%), sensitivity (0.998%), and specificity (0.903%) of any method. A cost-benefit analysis suggested that the plan could lower the price of software testing.

Methodology

Supervised Learning Approach

We separate the dataset into training and testing sets in order to use the supervised learning approach. The selected supervised learning algorithms, such as SVM, Random Forest, and Neural Networks, are trained using the training set. We assess each model's performance using metrics like accuracy, precision, recall, and F1-score once it has been trained.

- Because Support Vector Machines (SVM) can effectively handle both binary and multiclass classification problems, they have been widely employed in software failure prediction. SVM is an algorithm for supervised learning that looks for the best hyper plane in the feature space to maximize the margin between classes [9]. SVM can efficiently distinguish between faulty and non-faulty occurrences in the context of software fault prediction by learning from previous fault data.
- A well-liked machine learning algorithm called Random Forest has been effectively used for software defect prediction tasks. It is an ensemble learning technique that generates a more reliable and accurate outcome by combining the predictions of several decision trees [10]. Random Forest is a favored option for both researchers and practitioners in the software defect prediction domain due to its numerous benefits.
- Deep Learning models, or neural networks, have demonstrated great promise and efficacy in a number of domains, including the prediction of software defects. Strong machine learning

techniques called neural networks are modeled after the composition and operation of the human brain [11]. Their ability to discern detailed patterns from data makes them ideal for tasks involving the prediction of software failures, since the correlation between software metrics and faults may be complex.

Unsupervised Learning Approach

The unsupervised learning method employs the identical dataset, excluding any explicit fault annotations. To find possible defect clusters or abnormalities in the data, we use K-means clustering, DBSCAN, and Isolation Forest [6]. When ground truth is known, clustering accuracy, silhouette score, and visual examination of the clustering results are used as evaluation measures for unsupervised learning.

- Software defect prediction is one area in which K-means clustering, an unsupervised machine learning technique, has been used. In contrast to supervised learning algorithms, K-means can recognize patterns and classify related occurrences based only on the features of the data; it does not require labeled fault data. K-means clustering is used in software fault prediction to find possible clusters of defective instances. These clusters can then be examined further to find abnormalities or to understand areas of the program code that are prone to faults [12].
- Software defect prediction has also made use of another unsupervised machine learning technique called DBSCAN (Density-Based Spatial Clustering of Applications with Noise). When it comes to spotting data point clusters in high-density areas and differentiating outliers in low-density areas, DBSCAN is especially helpful. It is useful for identifying intricate patterns in software failure data since it does not require predetermined clusters and can handle datasets with different densities [13].
- A unique anomaly detection technique called Isolation Forest can be used for jobs predicting software faults. By iteratively splitting the data points until the anomalies are isolated from the rest of the dataset, an ensemble method based on trees separates anomalies. Isolation Forest is useful for finding errors in software development since it is good at spotting uncommon and abnormal occurrences, which has made it popular in a variety of fields [14].

Table 1: Comparison of Machine Learning Techniques

Technique	Strengths	Considerations
Support Vector Machine	SVM has strong generalization skills and can handle both linear and non-linear data separation. When labeled fault data is provided, it performs well in binary and multiclass classification problems.	A significant amount of labeled data may be needed for SVM to train and adjust parameters effectively. For big datasets, it might not be as effective as other methods.
Random Forest	One ensemble technique that can handle high-dimensional data and is less prone to over fitting is Random Forest. It has good performance in binary and multiclass classification problems and can give information about the relevance of features.	For best results, adjust the number of trees and other hyper parameters; otherwise, it might not be as interpretable as some other methods.
Neural Networks	Deep Learning methods, such as Neural Networks, are appropriate for high-dimensional and nonlinear interactions because they can automatically extract complicated patterns from data. They do exceptionally well on tasks with a lot of labeled data.	To operate at their best, neural networks may need a lot of processing power and a lot of training data. They might also be harder to understand than models that are simpler.
K-means Clustering	K-means is helpful for examining trends and perhaps troublesome areas because it may cluster data according to similarities. It is simple to use and effective.	Labeled fault data is not used in the prediction process of K-means because it is an unsupervised technique. It might have trouble handling intricate and asymmetrical groupings.
DBSCAN	DBSCAN can handle clusters of different sizes and forms and is quite good at spotting outliers. It works well in fault prediction jobs to find anomalies and outliers.	Predefined clusters are not necessary with DBSCAN; nevertheless, parameter selection can be difficult, and it might not work well with high-dimensional data.

Isolation Forest	Isolation Forest can effectively handle high-dimensional data and is specifically made for anomaly detection. It works well for locating uncommon and unusual fault occurrences.	Parameter tuning is crucial for best performance, much like DBSCAN, and it may have trouble in dense clusters with a high number of normal instances.

Results

We showcase the comparative outcomes of the supervised and unsupervised learning methodologies for software development fault identification. The performance indicators derived from the assessment of every technique served as the foundation for these findings.

Supervised Learning Results

The supervised learning algorithms achieved the following performance metrics on the test set:

Table 2

Algorithm	Accuracy	Precision	Recall	F1-Score
Support Vector Machines (SVM)	0.85	0.81	0.87	0.84
Neural Network	0.90	0.88	0.92	0.90
Random Forests	0.88	0.84	0.90	0.87

These metrics suggest that, for the particular software defect prediction task on the provided test set, the Neural Networks model performs the best.

Unsupervised Learning Results

The unsupervised learning techniques yielded the following results:

- **K-means Clustering:** 76% clustering accuracy, 0.72 silhouette score Silhouette Score: The degree of separation between the clusters is indicated by the silhouette score. With a higher score indicating more distinct and well-separated clusters, the score goes from -1 to 1. A silhouette score of 0.72 indicates that K-means clusters are reasonably distinct and well-separated.
  Clustering Accuracy: When ground truth labels are provided but not used in the clustering process, clustering accuracy is a statistic that can be applied. It gauges how well the genuine labels match the clustering result. 76% of the data points are appropriately assigned to the clusters according to their true labels, according to a clustering accuracy of 76%.
- **DBSCAN:** Clustering accuracy 72%, silhouette score 0.58

  Silhouette Score: This metric assesses how distinct and well-aligned the clusters are. Better-defined clusters are indicated by higher scores, which range from -1 to 1. With a 0.58 Silhouette score, the clusters created by DBSCAN appear to be somewhat separated, albeit possibly not as much as those produced by K-means (which obtained a 0.72 Silhouette score). As previously indicated, clustering accuracy is a statistic that is applied when genuine labels are available but not utilized in the clustering procedure. 72% of the data points are accurately assigned to the clusters based on their genuine labels, according to the clustering accuracy of 72%.
- **Isolation Forest: Anomaly detection accuracy 81%**

An unsupervised anomaly detection technique called Isolation Forest finds outliers or abnormalities in a dataset. Measuring the effectiveness of the Isolation Forest model in identifying anomalies in comparison to the ground truth (if available) is called anomaly detection accuracy.

Accurate Anomaly Detection: The ratio of correctly identified anomalies to all anomalies in the dataset is known as the anomaly detection accuracy. The Isolation Forest in this instance had an 81% anomaly detection accuracy. This indicates that the Isolation Forest model properly identified 81% of the abnormalities found in the dataset.

Discussion

The findings show that there are advantages and disadvantages to both supervised and unsupervised machine learning approaches for fault identification in high-quality software development. Because supervised learning methods use labeled training data, they can identify error situations with more clarity and precision. On the other hand, acquiring labeled fault data can be expensive and time-consuming, particularly for extensive projects. Although unsupervised learning techniques don't need

labeled data, they may have trouble differentiating between various defect kinds and result in false positives or false negatives. Furthermore, the performance of the clustering algorithms is highly dependent on the choice of suitable hyper parameters. We can make the following deductions based on the performance metrics that various software failure prediction algorithms achieve:

The best overall performance was obtained by neural networks, which had 90% accuracy, 88% precision, 92% recall, and 90% F1-score. It was the most efficient method among the three supervised learning algorithms for this particular software defect prediction job, outperforming Random Forest and SVM in every metric. With an accuracy of 88%, precision of 84%, recall of 90%, and F1-score of 87%, Random Forest showed impressive performance. Its strong performance in precision and recall makes it a reliable option for software defect prediction, particularly in the case of high-dimensional and intricate feature spaces. With an accuracy of 85%, precision of 81%, recall of 87%, and F1-score of 84%, SVM demonstrated strong performance. SVM is a trustworthy option, however in this specific situation, it performed marginally worse than Random Forest and Neural Networks. Unsupervised methods such as K-means clustering, DBSCAN, and isolation forest have shed light on the anomaly detection and clustering components of software failure prediction. With a Silhouette score of 0.72, K-means was able to show clusters that were reasonably well separated. A degree of separation was suggested by DBSCAN's Silhouette score of 0.58, while Isolation Forest's anomaly detection accuracy was 81%. When all the results were taken into account, the models Neural Networks, Random Forest, and SVM performed better in this dataset when it came to predicting software faults.

Conclusion and Future Work

In order to discover errors in software development, supervised and unsupervised machine learning algorithms are compared in this study report. Supervised learning can be limited in certain situations since it requires labeled data, even though it demonstrates improved error identification and accuracy. Despite being label-free, unsupervised learning might not be as precise and might have trouble with intricate software defect patterns. Depending on the particular needs and limitations of the software development project, supervised or unsupervised learning may be used. Combining the advantages of both approaches into a hybrid strategy could improve fault detection efficiency. To better understand hybrid approaches and increase the precision of defect detection in software development, more investigation and testing are required. This study advances knowledge on the applicability of each method for finding errors in software development, assisting practitioners in making wise choices that will result in the production of high-caliber software. To stay up with the rapidly changing software development landscape, more investigation and testing are required to examine hybrid approaches and improve fault detection methods. To obtain a thorough knowledge of each technique's generalization and performance across many contexts, future study should include additional analysis, cross-validation, and consideration of new assessment measures.

References

1. Dhall, Shafali & Chug, Anuradha. (2013). Software Defect Prediction Using Supervised Learning Algorithm and Unsupervised Learning Algorithm. IET Conference Publications. 2013. 5.01-5.01. 10.1049/cp.2013.2313.
2. Pandey, S. K., Mishra, R. B., & Tripathi, A. K. (2021). Machine learning based methods for software fault prediction: A survey. Expert Systems with Applications, 172, 114595.
3. Rashid, E. (2016). Software Fault Prediction Using Unsupervised Learning Technique: A Practical Approach. International Journal of u-and e-Service, Science and Technology, 9(11), 275-288.
4. Rathore, S. S., & Kumar, S.: Software fault prediction based on the dynamic selection of learning technique: findings from the eclipse project study. Applied Intelligence (2021). https://doi.org/10.1007/s10489-021-02346-x
5. S. A. K, V. Gururaj, K. R. Umadi, M. Kumar, S. P. Shankar and D. Varadam, "Comprehensive Survey of different Machine Learning Algorithms used for Software Defect Prediction," 2022 International Conference on Decision Aid Sciences and Applications (DASA), Chiangrai, Thailand, 2022, pp. 425-430, doi: 10.1109/DASA54658.2022.9764982.
6. Zhou, Y., & Leung, H.: Empirical analysis of object-oriented design metrics for predicting high and low severity faults. IEEE Transactions on Software Engineering (2006). https://doi.org/10.1109/TSE.2006.102

7. Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A. E., & Arshad, H.: State-of-the-art in artificial neural network applications: A survey. In Heliyon (2018). https://doi.org/10.1016/j.heliyon.2018.e00938

8. Zhang, N., Wu, L., Yang, J., & Guan, Y.: Naive bayes bearing fault diagnosis based on enhanced independence of data. Sensors (Switzerland) (2018). https://doi.org/10.3390/s18020463

9. Wang, J., & Zhang, C.: Software reliability prediction using a deep learning model based on the RNN encoder–decoder. Reliability Engineering and System Safety (2018). https://doi.org/10.1016/j.ress.2017.10.019

10. Dam, H. K., Pham, T., Ng, S. W., Tran, T., Grundy, J., Ghose, A., Kim, T., & Kim, C. J.: Lessons learned from using a deep tree-based model for software defect prediction in practice. IEEE International Working Conference on Mining Software Repositories (2019). https://doi.org/10.1109/MSR.2019.00017

11. C. A. Escobar and R. Morales-Menendez, “Machine Learning Techniques For Quality Control In High Conformance Manufacturing Environment,” Advances in Mechanical Engineering, vol. 10, no. 2, p. 168781401875551, 2018.

12. S. Masuda, K. Ono, T. Yasue, and N. Hosokawa, “A Survey of Software Quality for Machine Learning Applications,” 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2018.

13. K. Chandra, G. Kapoor, R. Kohli, and A. Gupta, “Improving Software Quality Using Machine Learning,” 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH), 2016.

14. E. A. Rashid, R. B. Patnaik, and V. C. Bhattacherjee, “Machine Learning and Software Quality Prediction: As an Expert System,” International Journal of Information Engineering and Electronic Business, vol. 6, no. 2, pp. 9–27, Aug. 2014.