

Machine Learning-Based False Positive Software Vulnerability Analysis

Shahid M.¹, Gupta S.^{2*}, Pillai S.³

DOI: <https://doi.org/10.58260/j.iet.2202.0105>


¹ Mohammad Shahid, Department of Computer Science and Engineering, Noida Institute of Engineering and Technology, Greater Noida, Uttar Pradesh, India.

^{2*} Sunil Gupta, Department of Computer Science and Engineering, Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab, India.

³ Sofia Pillai, Department of Artificial Intelligence, Noida Institute of Engineering and Technology, Greater Noida, Uttar Pradesh, India.

Measurements and fault data from an older software version were used to build the fault prediction model for the new release. When past fault data isn't available, it's a problem. The software industry's assessment of programme module failure rates without fault labels is a difficult task. Unsupervised learning can be used to build a software fault prediction model when module defect labels are not available. These techniques can help identify programme modules that are more prone to errors. One method is to make use of clustering algorithms. Software module failures can be predicted using unsupervised techniques such as clustering when fault labels are not available. Machine learning clustering-based software failure prediction is our approach to solving this complex problem.

Keywords: Machine Learning, Supervised Learning, Vulnerabilities, Software, Clustering Algorithm

| Corresponding Author | How to Cite this Article | To Browse |
|--|--|---|
| Sunil Gupta, , Department of Computer Science and Engineering, Chitkara University Institute of Engineering and Technology, Chitkara University, , Punjab, India. Email: mohd86shahid@gmail.com | Mohammad Shahid, Sunil Gupta, Sofia Pillai, Machine Learning-Based False Positive Software Vulnerability Analysis. Glo.Jou.of.Innov.and.Eme.Tech. 2022;1(1):29-35. Available From http://iet.adsrs.net/index.php/iet/article/view/6 |  |

| | | | | |
|--|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------|
| Manuscript Received 2022-05-14 | Review Round 1 2022-05-16 | Review Round 2 2022-05-23 | Review Round 3 2022-05-30 | Accepted 2022-06-05 |
| Conflict of Interest Nil | Funding Nil | Ethical Approval Yes | Plagiarism X-checker 19% | Note |
|  © 2022 by Mohammad Shahid, Sunil Gupta, Sofia Pillai and Published by ADSRS Education and Research. This is an Open Access article licensed under a Creative Commons Attribution 4.0 International License https://creativecommons.org/licenses/by/4.0/ unported [CC BY 4.0].  | | | | |

Introduction

To meet the objectives of a software quality management project, SOFTWARE QUALITY MODELS are effective tools. A software quality model can be used to identify defective programme modules. Since only those programme modules will benefit from the limited resources allotted for software quality inspection and enhancement, these resources can be used more efficiently while also saving money. One of the most important aspects of developing high-assurance systems is being able to identify and discover any software flaws early in the development process.

Data from prior releases or similar projects' software measurement and defect (quality) data is frequently used to train a software quality model. Using the newly-trained model, we can now estimate the quality of the existing project components. This kind of supervised learning technique assumes that the development team has previously worked on systems similar to the current project and that defect data for all programme modules in the training data is available. There are a number of practical limits on the availability of defect data for modules in training data in the actual world of software development. Earlier releases or similar projects, for example, may not have had software defect data collected. Using software measurement and defect data from previous projects for modelling purposes is wrong due to the absence of business expertise in building a similar system. Distributed software development is widespread in today's globalised world of technology. Consequently, software defect data may not be gathered by all development sites depending on the organisational structure and resources of specific development sites. The supervised learning approach to software quality modelling cannot be applied since the training data lacks defect data or quality-based class labels. So the software engineering specialist is in charge of assessing software quality or determining if programme modules are failure prone (fp) or not (nfp). The process of individually labelling each programme module is time-consuming, costly, and labour-intensive. We provide a semi-supervised clustering strategy to help the expert in the labelling process. The proposed method relies on constraint-based clustering and k-means as the underlying algorithm. When using k-means clustering, the

Constraint makes sure modules (instances) remain in clusters that have already been labelled as either fp or nfp. Unlabelled S/W modules are subjected to a two-stage quality assessment using the proposed approach. Using single metrics thresholds in the first case and Fuzzy C in the second Means clustering is compared to mean squared error in terms of time and value. To organise things into clusters, clustering algorithms group things that are similar while separating those that are unlike. Using clustering approaches, modules with similar metrics can be grouped according to how similar or dissimilar they are (distances). After the clustering process is complete, an expert or an automated approach can examine the representative modules of each cluster to determine whether or not it is prone to failure. In this work, we demonstrate that software metrics thresholds can be used to name clusters instead of consulting an expert during this time-consuming step. The process of grouping is referred known as clustering. Partition Clustering: For the partition clustering algorithm, the database of n objects is partitioned into many clusters, each of which optimises a clustering criterion within the cluster, such as minimising squared distance from the mean. Partitional clustering is one sort of analysis that uses K-means clustering as a method. Machine Learning and Clustering There are two forms of clustering: simple clustering and complex clustering.

Engaging with a Machine to Learn: To learn from data, machine learning latches on to and extracts attributes from that data. The learning and prediction performance will have an impact on the data set's size and quality. Engaging with a Machine to Learn

Types of machine learning

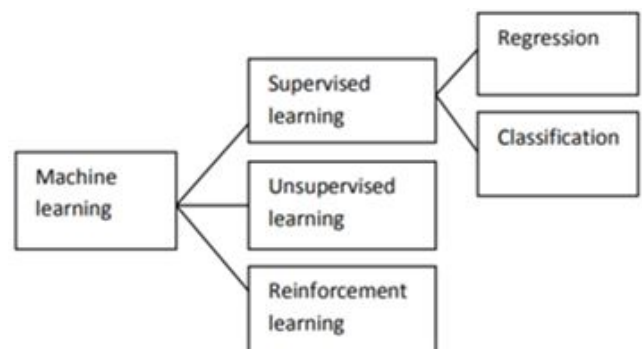


Figure 1 Types of Machine Learning

- **Supervised learning:** In order to train supervised learning, it is necessary to use a labelled data set.

- A common application of supervised learning is to forecast future use from historical data. This method has two subtypes: regression and classification. In regression, the label is a number that is either positive or negative. The label has little significance when it comes to classification, on the other hand.
- Unsupervised learning uses data that hasn't been previously classified and uses that information to produce predictions.
- Games, navigation, and robots all use reinforcement learning as a method of teaching. Each of these three factors plays a critical role in how this learning approach works [10,14].
- strategies for machine learning: The development of algorithms for the construction of machine learning models and major machine learning processes. Three classifiers are discussed in this research: decision trees, nave bayes, and support vector machines.
- **Decision tree:** A decision tree is constructed through the use of supervised learning. It is a predictive model method in the fields of artificial intelligence, data mining, and statistics. In decision analysis, a decision tree can serve as an illustration of decision making and decisions. Despite the fact that a data mining decision tree characterises data without making decisions, the resulting allocation tree can be utilised as a decision-making input in some cases.

Many different decision tree algorithms are available:

- A. Working on a C4.5 (which is the next version of ID3) (classification and regression tree)
- B. MARS: extends the decision tree to better handle numerical input.
- C. The probabilistic learning approach Nave Bayes has been extensively studied. The naive Bayesian classifier says there is no such thing as attribute addiction. This is called conditional independence.
- D. Nave Bayes is a wonderful approach to learn about expectations.
- E. The Nave Bayes classifier uses fast calculation to make judgments.

Using supervised machine learning, classification and regression issues can be solved. But it's largely used for classification. SVM can classify nonlinear

Data and linear data. This approach assigns a value to each coordinate, making each data point an n-dimensional space point (where n is the number of characteristics you have).

Actions to help the vector machine: SVM classification, support vectors, and support vector machine Support vectors; Support vector machines Preparation: Prep data for training.

A real-world vulnerability prediction scenario was the purpose of the Deep Learning Vulnerability Detection Project. They were dismayed to see their output had fallen by more than half. A recent study found issues with deep learning-based vulnerability forecasting systems' training data and model selection (e.g., data duplication, unrealistic distribution of vulnerable classes, etc). Token-based techniques are a good example.) So-called "fixes" often neglect the root causes of They hunt for insignificant artefacts in the dataset rather than significant ones. Including real vulnerability prediction variables in data collection and model creation can lead to more empirically based solutions. The new tools outperform the old ones in precision (33.57%) and recall (90%) (128 percent). The research roadmap emphasises the inadequacies of present approaches. Please read it. Thanks. They provide VIVA, a binary-level summarising and patch matching tool, to detect reoccurring vulnerabilities. To find incomplete vulnerabilities and repair signatures, slice and refine pseudocode traces. Security risks are promptly identified via pre-filtering and signature matching. When it comes to one-day and recurring vulnerabilities, VIVA's source code and binary matching solutions outperform the competition (28.58s per signature search in 4M functions). 92 new vulnerabilities in real-world project series and versions. Eleven are still unresolved in this release.

AI Security Vulnerabilities Found (AI) Analyze the D2A dataset. Error detection uses differential analysis. The Open-Source Version Pairings dataset contains open-source version pairs. Static analysis was done on each project version before and after bug patches. If no further vulnerabilities were found, a fix was issued. Its massive tagged dataset was utilised to train vulnerability detection models. To begin, they utilised a dataset to train a classifier to detect false positives.

Classifying Software Error Messages Static Code Analysis Reports [4]

In the software development industry, static source code is regularly inspected for faults and flaws. Several technologies are used to avoid identifying bad code. Various bug-finding methodologies can result in numerous notifications for the same set of faults. Code analysis is time-consuming and costly because of the enormous number of warnings. Using principal component analysis, they built an analytical model by combining software warning categories from commonly used Java and C++ issue detection tools. Respondents ranked false positives as the least important problem, whereas the most challenging problems were prioritised.

Software Vulnerability Analysis Using Advanced Techniques and Tools [4].

Our goal is to find ways to make various vulnerability detection technologies more effective (increasing the detection rate and decreasing the number of false-positives). It was decided to employ static code analysis (SCA) as the primary method because of its high detection rate. Because of the high number of false-positive outcomes, they want to include SCA into other detection systems. (An illustration would be software metrics.) It will be easier for developers and patchers to work on less vital jobs if problems are fixed first.

CURRENT INFRASTRUCTURES Previously, software module defects were predicted using Quad Tree-based K-Means. Using a quad tree as a starting point, the K-Means Algorithm clusters data using its results. To achieve the desired number of cluster centres, simply adjust the threshold value. A Quad Tree was predicted to be preferable for grouping in the beginning. The most significant advantages have already been identified. 4 Leaf Trees (c) It's possible to utilise this strategy to foresee software module errors.

Currently Existing System Issues:

- A. For this reason, K-Means is a challenge to employ.
- B. The cluster centres are assigned at the beginning of the Quad Tree-based technique.
- C. Using Quad Tree and K-Means takes longer.
- D. It's impossible to say what the failure rate of a piece of software is.

Fourth, a Methodology is Suggested: We created a machine learning clustering technique to help find

Software bugs. We utilise an attribute selection technique to narrow the solution space after starting with the input dataset. Using the attribute selection strategy, you can focus on the most important attribute while dealing with less traits. An attribute's importance is determined by how highly it is regarded. The most critical aspects can be scored. When one Attribute is selected, the accessible Attributes are reduced. There is only a certain quantity available. This trimmed-down characteristic is used in clustering. By using clustering criteria such as squared distance from the mean minimization, this technique divides a database of items into groups that may be compared. Partitional clustering can be approached using fuzzy-C methods, such as fuzzy-C means. After the data has been clustered, Fuzzy-C Means are used to further cluster it. Clustering (FCMC) keeps track of the centroid values, which can be thought of as a mechanism to calculate a threshold. In order to be sure that this value is correct, the centroid data values must be compared to it. VULNERABILITIES y was assigned if the centroid data metric value exceeded the threshold; otherwise, it was designated not VULNERABILITIES y.

Advantage in proposed system

- Propose a single technique
- FUZZY C means is used for Clustering.
- Processing time is reduced
- Metric Threshold has to Clear description

System Architecture

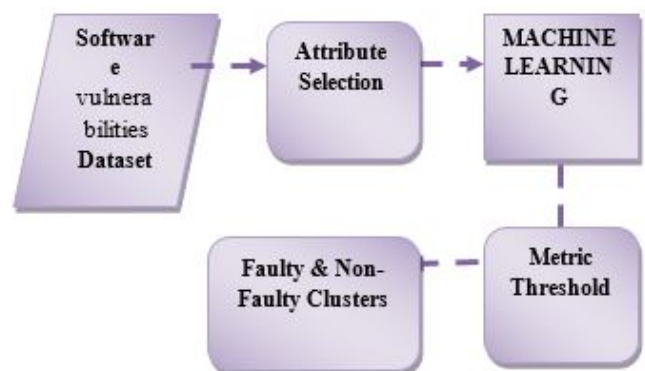


Figure 2 Proposed System

A. VULNERABILITIES Dataset: to put together a database of software (AR3, AR4 and AR5). Reading and storing the Dataset's data has completed. To use Attributes Selection, you must first get the attributes' names.

B. Attribute Selection: A list of attributes for our dataset will be returned. The dataset consists of 30 different variables. It's possible to reduce the amount of useful learning features by focusing on a few key criteria. It will take a long time and a lot of hard work to incorporate all of these characteristics. As a result, we're better able to focus on what really matters: our core business. Determine the relevance of an attribute by using Attribute Evaluation. We pick and rank qualities with the help of the Weka programme.

C. Using machine learning to create clusters: A group of objects may be described as "similar," but they are also "different" due to their differences. In order to include the same set of information in different clusters, FCM (machine learning data clustering) can be employed. Only when the data has been grouped can Machine Learning be applied to it. The cluster centroid is obtained and the data is split after the clustering phase is complete.

D. Metric Threshold: Decide on the right metric thresholds based on a set of criteria. It is determined by characteristics such as the number of code lines and Cyclomatic Complexity. Other parameters include the number of operators and operands in the programme, as well as the number of unique operators and operands in the programme (TOPnd).

The final step is to calculate the threshold vector [LoC, CC, UOp, UOpnd, TOP, TOPnd].

Detect VULNERABILITIES: Once the clustering process is complete, a single data point is stored in each cluster. Determine the metric threshold for each cluster. If the centroid data point's metric value exceeded the threshold, the cluster was deemed defective; otherwise, it was classed as non-defective.

As the name implies, this technique uses just-in-time defect prediction to determine whether or not the file in question in the current changeset is defective at the time of the change. Just-in-time defect prediction systems such as the ones listed below are traditional:

1. Training Data Extraction Based on the project's revision history and issue tracking system, label each change as buggy or clean. A faulty change contains one or more flaws, whereas a flawless change is error-free.

2. Feature Extraction. Calculate new values for relevant characteristics based on historical data. Researchers have previously utilised a wide variety of features to classify changes.

3. Model Learning. Use a classification approach to create a model based on the labelled changes and their related attributes.

4. Model Application. For a new change, extract the values of various features. Input these values to the learned model to predict whether the change is buggy or clean.

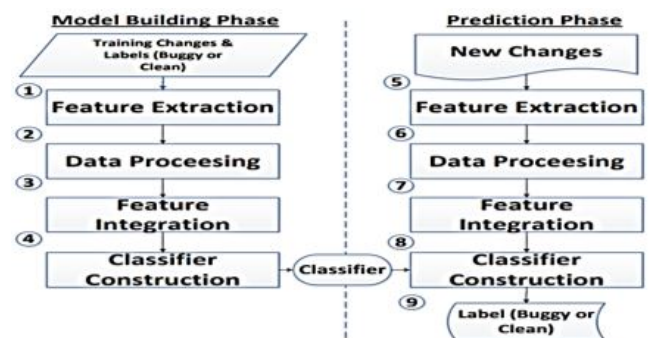


Figure 3 Flow of Proposed System

Model construction and prediction are the two main aspects of the system. In the model-building phase, our goal is to generate a classifier (i.e., a statistical model) utilising deep learning and machine learning approaches from past changes with well-defined labels (i.e., buggy or clean). In the prediction phase, this classifier would be used to assess whether an incoming modification would be buggy or clean.

Performance measures: Machine learning evaluates prediction accuracy using a variety of performance metrics:

- Precision
- Recall
- Accuracy
- F measure
- Roc (receiver operating characteristics)

| | | PREDICTED | |
|--------|---|---------------------|---------------------|
| | | P | N |
| ACTUAL | P | TRUE POSITIVE (TP) | FALSE NEGATIVE (FN) |
| | N | FALSE POSITIVE (FP) | TRUE NEGATIVE (TN) |

Table 1: Confusion Matrix

There were calculated using the prediction classification confusion matrix table:

A. TP (true positive): Number of correct predictions that an instance is positive.

B. TN (true negative): Number of correct predictions that an instance is negative.

C. FN (false negative): Number of incorrect predictions that an instance is positive.

D. FP (false positive): Number of incorrect predictions that instance is negative.

Accuracy: The total number of predictions that were correct

$$\text{Accuracy (\%)} = (TP+TN)/(TP+FP+FN+TN)$$

Precision: The predicted true pages those were correct:

$$\text{Precision (\%)} = TP/(TP+EP)$$

Recall: The predicted true pages that were correctly identify.

$$\text{Recall (\%)} = TP/(FN+TP)$$

F-Measure: Derives from precision and recall values:

$$\text{F-Measure (\%)} = (2 \times \text{recall} \times \text{precision}) / (\text{recall} + \text{precision})$$

Results and Analysis

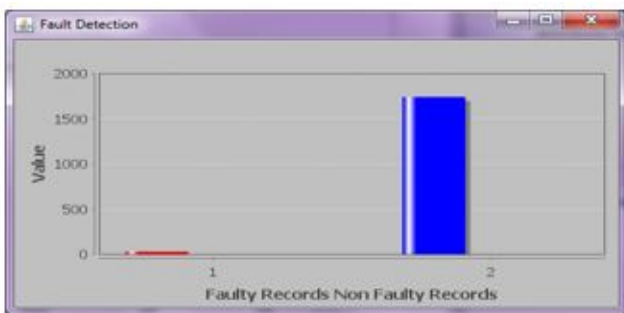


Figure 4 Fault Detection Rate



Figure 5 Error Rate

So far as result & analysis is concern from Figure 4 we have seen the Fault Detection Rate by applying the formula explained in above confusion matrix, with help of confusion matrix easily we can predict the result with the help of machine learning & the matrix compares the actual target values with those predicted by the machine learning model.

We looked for trends in samples that had been misclassified (false positives and false negatives). n-gram statistics from the best n-grams combination we discovered (69 percent accuracy) were evaluated to perform error analysis. We determined the mean value of each attribute across all samples. Real negatives, false positives, true positives, and false negatives were all treated independently as shown in figure 5.

Conclusion

A technique based on Machine Learning clustering and Metrics threshold-based VULNERABILITIES prediction has been developed for situations in which there is no prior data on VULNERABILITIES available. When using the weighted attribute selection approach, just the most critical attributes are displayed. Organizing the data is accomplished through clustering, with each cluster having a unique centroid value. For each cluster, the threshold was set using the Metric Threshold function and its value compared to the centroid. The data for VULNERABILITIES y and non-VULNERABILITIES y were then displayed. Based on the first experiment's findings, it's clear that changing hyperparameters enhanced precision and other performance measures. Trains for a particular vulnerability, like this one, may have an influence since they are more effective because they are targeted. It is the goal of the new Fast scan to construct models that anticipate only one type of vulnerability before merging the components into a global analyser in an end system, whereas the previous Fast scan aimed to predict a wide range of vulnerabilities.

Reference

1. Saikat Chakraborty;Rahul Krishna;Yangruibo Ding;Baishakhi Ray "Deep Learning based Vulnerability Detection: Are They have There Yet" IEEE Transactions on Software Engineering Year: 2021 | Early Access Article | Publisher: IEEE

2. Yang Xiao;Zhengzi Xu;They have they havei Zhang;Chendong Yu;Longquan Liu;They havei Zou;Zimu Yuan;Yang Liu;Aihua Piao;They havei Huo "VIVA: Binary Level Vulnerability Identification via Partial Signature" 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) Year: 2021 | Conference Paper | Publisher: IEEE
3. Yunhui Zheng;Saurabh Pujar;Burn Lewis;Luca Buratti;Edward Epstein;Bo Yang;Jim Laredo;Alessandro Morari;Zhong Su "D2A: A Dataset Built for AI-Based Vulnerability Detection Methods Using Differential Analysis" 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) Year: 2021 | Conference Paper | Publisher: IEEE
4. Binh Hy Dang "A Practical Approach for Ranking Software Warnings from Multiple Static Code Analysis Reports" 2020 SoutheastCon Year: 2020 | Volume: 2 | Conference Paper | Publisher: IEEE
5. José D'Abruzzo Pereira "Techniques and Tools for Advanced Software Vulnerability Detection" 2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW) Year: 2020 | Conference Paper | Publisher: IEEE
6. V. Bhattacharjee and P.S. Bishnu, "Software VULNERABILITIES Prediction and Defect Estimation Using Machine Learning and KMedoids Algorithm" – 2011
7. P.S. Bishnu and V. Bhattacharjee , "Application of K-Medoids with kd-Tree for Software VULNERABILITIES Prediction" –, 2011
8. P.S. Bishnu and V. Bhattacharjee, "Outlier Detection Technique Using Quad Tree" –, 2009
9. Goyal A.,Sharma V.K. and K. Sandeep, "Development of hybrid ad hoc on demand distance vector routing protocol in mobile ad hoc network" , International Journal on Emerging Technologies,11(2), pp. 135–139,2020.
10. Goyal A.,Rathore L. and k. sandeep, "A Survey on Solution of Imbalanced Data Classification Problem Using SMOTE and Extreme Learning Machine", Lecture Notes in Networks and Systems, 204, pp. 31–44, 2021
11. S. Zhong, T.M. Khoshgoftaar, and N. Seliya, "Analyzing Software Measurement Data with Clustering Techniques" , 2004
12. C. Catal, U. Sevim, and B. Diri, "Clustering and Metrics Threshold Based Software VULNERABILITIES Prediction of Unlabeled Program Modules" , 2009
13. Goyal A, Sharma V.K., "Modifying the MANET routing algorithm by GBR CNR-efficient neighbor selection algorithm", International Journal of Innovative Technology and Exploring Engineering, 8(10), pp. 912–917, 2019
14. Philip K Chan and Richard P Lippmann. Machine learning for computer security. Journal of Machine Learning Research, 7(Dec):2669–2672, 2006.
15. Brian Chess and Gary McGraw. Static analysis for security. IEEE security & privacy, 2(6):76–79, 2004
16. Wes Felter, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. An updated performance comparison of virtual machines and linux containers. In 2015 IEEE international symposium on performance analysis of systems and software (ISPASS), pages 171–172. IEEE, 2015
17. Samuel Gonçalves Ferreira. Vulnerabilities fast scan - tackling sast performance issues with machine learning. Master's thesis, University of Minho, 2019.
18. Dr. Mohammad Shahid" "Black Hole Detection and Prevention Using Digital Signature and SEP in MANET" in the 10th IEEE International Conference on Emerging Trends in Engineering & Technology Signal and Information Processing (ICETET SIP-22) held during April 29-30, 2022 at G H Rasoni College of Engineering, Nagpur (India).
19. MOHAMMAD SHAHID" Efficient and Reliable Packet Routing Solutions for Wireless Sensor Networks" 3RD INTERNATIONAL CONFERENCE (ONLINE) ON INNOVATIONS IN COMMUNICATION COMPUTING AND SCIENCES (ICCS-2021)
20. Rahma Mahmood and Qusay H Mahmoud. Evaluation of static analysis tools for finding vulnerabilities in Java and C/C++ source code. arXiv preprint, 2018. arXiv:1805.09040
21. D. Steinley and M.J. Brusco, "Initializing K-Means Batch Clustering: A Critical Evaluation of Several Techniques" , 2007