International Journal of Engineering and Management Research e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-13, Issue-2 (April 2023)
https://ijemr.vandanapublications.com https://doi.org/10.31033/ijemr.13.2.37

A Review Paper on Software Defect Prediction Based on Rule Mining

Fareha Bashir[1] and Dr. Akbar Shaun[2]
[1]PG Student, Department of Computer Science & Engineering, Integral University, Lucknow, INDIA
[2]Assistant Professor, Department of Computer Science & Engineering, Integral University, Lucknow, INDIA

[1]Corresponding Author: farehabashir62@gmail.com

Received: 29-03-2023 **Revised:** 12-04-2023 **Accepted:** 29-04-2023

ABSTRACT

Software defect prediction is an important task in software engineering, aimed at identifying and mitigating software defects before they become major problems. Rule mining is a technique used to discover interesting patterns and relationships in data, and can be applied to software defect prediction by analyzing past data on software development and testing. This abstract discusses the process of software defect prediction based on rule mining, including data collection, data pre-processing, feature extraction, rule mining, model evaluation, and model deployment. By accurately predicting the likelihood of defects occurring in future software releases, developers can take proactive measures to prevent defects from occurring, thereby improving software quality and reducing the time and resources spent on fixing bugs.

Keywords— Software Defect Prediction, Classification Algorithm, Confusion Matrix, Rule Mining

I. INTRODUCTION

Software defects are a common occurrence in software development, and can result in delays, increased costs, and damage to a company's reputation. As such, predicting and preventing defects has become an important research area in software engineering. Software defect prediction based on rule mining is a technique that leverages machine learning algorithms to identify potential software defects by analyzing past data on software development and testing[1].

Rule mining is a process of discovering interesting patterns and relationships in data. In the context of software defect prediction, rule mining involves analyzing data on software development and testing to identify patterns that are indicative of potential defects. By identifying these patterns, developers can take proactive measures to prevent defects from occurring, such as improving code quality, adjusting development processes, or providing additional training to developers.

Software defect prediction based on rule mining has become increasingly popular in recent years, as it provides a data-driven approach to identifying potential defects. By analyzing large datasets, machine learning algorithms can identify patterns that may not be apparent to human developers, thereby improving the accuracy of defect predictions. This can result in improved software quality and reduced costs, as developers can identify and fix defects before they become major problems.

Software defect prediction based on rule mining is a powerful technique that can help improve software quality by identifying potential defects before they occur. By leveraging machine learning algorithms to analyze past data on software development and testing, developers can take proactive measures to prevent defects, thereby reducing costs and improving customer satisfaction.

II. EXISTING APPROACH ON SOFTWARE DEFECT PREDICTION BASED ON RULE MINING

There are several existing approaches to software defect prediction based on rule mining, including:

1. Association Rule Mining

This approach involves using association rule mining algorithms to identify correlations between different variables in software development and testing data. These correlations can be used to predict the likelihood of defects occurring in future software releases[2].

Association Rule Mining is a data mining technique that is widely used for analyzing large datasets to identify patterns, correlations, and associations among different variables. The mathematical model for Association Rule Mining involves the following steps:

1. Let D be a dataset of transactions where each transaction t consists of a set of items {i1, i2, ..., in}. Let T be the set of all transactions in D.
2. Calculate the support and confidence of each rule R in the form $X \rightarrow Y$, where X and Y are sets of items. The support of a rule R is defined as the fraction of transactions in T that contain both X and Y. The confidence of a rule R is defined as the fraction of transactions in T that contain X and also contain Y.
3. Select rules with a minimum support and confidence threshold. This threshold is set based on the specific

application and the goals of the software development team.

4. Use the selected rules to predict the likelihood of defects occurring in future software releases. For example, if a rule X → Y has a high support and confidence, it may indicate that there is a strong correlation between the items in X and Y, and that the occurrence of X may increase the likelihood of defects in software releases.
5. Evaluate the accuracy of the predictions using metrics such as precision, recall, and F1-score. This involves comparing the predictions made by the model against actual outcomes to determine how well the model is performing.

Association Rule Mining is a powerful technique for software defect prediction that can be used to identify patterns and correlations in software development and testing data. By accurately predicting the likelihood of defects occurring in future software releases, developers can take proactive measures to prevent defects from occurring, thereby improving software quality and reducing the time and resources spent on fixing bugs[3].

2. Decision Tree-based Rule Mining

This approach involves building decision tree models to identify the most important features or variables that are predictive of software defects. These decision tree models can be used to generate rules that predict the occurrence of defects[4].

Decision Tree-based Rule Mining is a machine learning technique that involves building decision tree models to identify the most important features or variables that are predictive of software defects. The mathematical model for Decision Tree-based Rule Mining involves the following steps

1. Let D be a dataset of software development and testing data, where each record d consists of a set of features {f1, f2, ..., fn} and a binary label indicating whether the record corresponds to a defective software release.
2. Build a decision tree model using the dataset D. The decision tree model consists of a set of decision nodes and leaf nodes, where each decision node corresponds to a feature or variable, and each leaf node corresponds to a predicted label.
3. Split the dataset at each decision node based on the value of the corresponding feature or variable. The goal is to maximize the purity of each split, such that records with the same label are grouped together.
4. Calculate the impurity of each split using a measure such as Gini impurity or information gain. The impurity of a split is a measure of how well the split separates records with different labels.
5. Prune the decision tree to reduce overfitting and improve generalization. This involves removing nodes that do not contribute to the predictive power of the model.
6. Generate rules from the decision tree model by traversing the tree from the root to each leaf node. Each path from the root to a leaf node corresponds to a rule that predicts the label of a record based on the values of its features.

Use the generated rules to predict the likelihood of defects occurring in future software releases. For example, if a rule predicts a high likelihood of defects for software releases with certain combinations of features, developers can take proactive measures to prevent defects from occurring in those releases.

Evaluate the accuracy of the predictions using metrics such as precision, recall, and F1-score. This involves comparing the predictions made by the model against actual outcomes to determine how well the model is performing.

Decision Tree-based Rule Mining is a powerful technique for software defect prediction that can identify the most important features or variables that are predictive of defects. By accurately predicting the likelihood of defects occurring in future software releases, developers can take proactive measures to prevent defects from occurring, thereby improving software quality and reducing the time and resources spent on fixing bugs.

3. Bayesian Network-based Rule Mining

This approach involves building Bayesian network models that represent the relationships between different variables in software development and testing data. These models can be used to generate rules that predict the occurrence of defects[5].

Bayesian Network-based Rule Mining is a probabilistic graphical model that uses Bayesian inference to predict the likelihood of software defects. The mathematical model for Bayesian Network-based Rule Mining involves the following steps:

1. Let D be a dataset of software development and testing data, where each record d consists of a set of features {f1, f2, ..., fn} and a binary label indicating whether the record corresponds to a defective software release.
2. Build a Bayesian network model using the dataset D. The Bayesian network model consists of a set of nodes and edges, where each node corresponds to a feature or variable, and each edge represents a probabilistic dependency between two nodes.
3. Specify prior probabilities for each node in the network. The prior probabilities reflect the domain knowledge or beliefs about the probability distribution of each variable before observing the data.

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-13, Issue-2 (April 2023)
https://doi.org/10.31033/ijemr.13.2.37

4. Learn the conditional probability distributions of each node given its parent nodes using maximum likelihood estimation or Bayesian parameter estimation. The conditional probability distributions capture the probabilistic dependencies between different variables.
5. Use the Bayesian network model to predict the likelihood of defects occurring in future software releases. This involves computing the posterior probabilities of the defective label given the values of the observed features.
6. Generate rules from the Bayesian network model by examining the conditional probabilities and their dependencies. Each rule corresponds to a probabilistic relationship between the features and the likelihood of defects.
7. Use the generated rules to identify the most important features or variables that are predictive of defects. By examining the conditional probabilities and their dependencies, developers can determine which features are most strongly associated with defects.
8. Evaluate the accuracy of the predictions using metrics such as precision, recall, and F1-score. This involves comparing the predictions made by the model against actual outcomes to determine how well the model is performing.

Bayesian Network-based Rule Mining is a powerful technique for software defect prediction that can capture complex probabilistic dependencies between different features and variables. By accurately predicting the likelihood of defects occurring in future software releases, developers can take proactive measures to prevent defects from occurring, thereby improving software quality and reducing the time and resources spent on fixing bugs.

4. Random Forest-based Rule Mining

Random Forest-based Rule Mining is a machine learning technique that involves building an ensemble of decision trees to predict the likelihood of software defects. The mathematical model for Random Forest-based Rule Mining involves the following steps:

1. Let D be a dataset of software development and testing data, where each record d consists of a set of features {f1, f2, ..., fn} and a binary label indicating whether the record corresponds to a defective software release.
2. Build an ensemble of decision trees using the dataset D. The ensemble consists of a set of decision trees, where each decision tree is built using a subset of the features and a subset of the records in D.
3. Train each decision tree using a randomized feature selection and a bagging technique to improve the diversity and robustness of the ensemble. The randomized feature selection involves selecting a random subset of features at each node in the tree, while the bagging technique involves sampling a random subset of records with replacement.
4. Predict the likelihood of defects occurring in future software releases by aggregating the predictions of the decision trees in the ensemble. This involves computing the majority vote or weighted average of the predictions made by the decision trees.
5. Generate rules from the Random Forest model by examining the most important features or variables that are predictive of defects. The importance of each feature can be measured using metrics such as mean decrease impurity or mean decrease accuracy.
6. Use the generated rules to identify the most important features or variables that are predictive of defects. By examining the importance scores of each feature, developers can determine which features are most strongly associated with defects.
7. Evaluate the accuracy of the predictions using metrics such as precision, recall, and F1-score. This involves comparing the predictions made by the model against actual outcomes to determine how well the model is performing.

This approach involves building random forest models that can identify the most important features or variables that are predictive of software defects. These models can be used to generate rules that predict the occurrence of defects[6].

5. Neural Network-based Rule Mining

This approach involves building neural network models that can identify patterns in software development and testing data and generate rules that predict the occurrence of defects[7].

These approaches have been shown to be effective in identifying potential defects in software development and testing data. However, the choice of approach may depend on the specific characteristics of the data and the goals of the software development team. It is important to evaluate and compare different approaches to determine the most effective approach for a given application.

1. Neural Network-based Rule Mining is a machine learning technique that involves building a neural network to predict the likelihood of software defects. The mathematical model for Neural Network-based Rule Mining involves the following steps:
2. Let D be a dataset of software development and testing data, where each record d consists of a set of features {f1, f2, ..., fn} and a binary label indicating whether the record corresponds to a defective software release.
3. Build a neural network model using the dataset D. The neural network model consists of a set of nodes and edges, where each node corresponds to a neuron or unit, and each edge represents a weighted

connection between two neurons.

4. Specify the architecture of the neural network, including the number of layers, the number of neurons per layer, the activation functions, and the learning algorithm. The architecture of the neural network can be optimized using techniques such as grid search or random search.
5. Train the neural network model using the dataset D. This involves adjusting the weights of the connections between neurons to minimize the prediction error on the training data. The training can be performed using techniques such as backpropagation or stochastic gradient descent.
6. Predict the likelihood of defects occurring in future software releases using the trained neural network model. This involves feeding the values of the observed features into the input layer of the network and propagating them through the hidden layers to the output layer.
7. Generate rules from the neural network model by examining the weights of the connections between neurons. Each rule corresponds to a weighted relationship between the features and the likelihood of defects.
8. Use the generated rules to identify the most important features or variables that are predictive of defects. By examining the weights of the connections between neurons, developers can determine which features are most strongly associated with defects.
9. Evaluate the accuracy of the predictions using metrics such as precision, recall, and F1-score. This involves comparing the predictions made by the model against actual outcomes to determine how well the model is performing.
10. Neural Network-based Rule Mining is a powerful technique for software defect prediction that can capture complex nonlinear relationships between different features and variables. By accurately predicting the likelihood of defects occurring in future software releases, developers can take proactive measures to prevent defects from occurring, thereby improving software quality and reducing the time and resources spent on fixing bugs.

6. Web Document Information Extraction using Class Attribute Approach for Rule Mining

Web document information extraction using class attribute approach for rule mining is a technique that involves extracting structured data from unstructured web pages using class attributes and applying rule mining techniques to extract patterns and rules from the extracted data. the class attribute approach for web document information extraction and rule mining is a powerful technique for extracting structured data from unstructured web pages and deriving useful patterns and rules from the data. By applying rule mining techniques to the extracted data, developers can gain valuable insights into the structure and content of the web pages and use this information to improve search engine optimization, content management, data integration, and other applications [8].

7. Different Technique of Load Balancing for Defect Prediction

Load balancing is an important technique for defect prediction that involves distributing the workload across multiple computing resources in order to optimize performance and improve accuracy[9]. There are several different techniques of load balancing that can be used for defect prediction. load balancing is a critical technique for defect prediction that can help to optimize performance, improve accuracy, and reduce the time and resources required for software testing and development. By choosing the right load balancing technique for a given application or environment, developers can ensure that their defect prediction models are robust, scalable, and reliable [10,11].

8. An Unsupervised Deep Learning Ensemble Model for Defect Prediction

An unsupervised deep learning ensemble model for defect prediction is a type of machine learning model that uses unsupervised learning techniques to identify patterns and anomalies in software code that may indicate the presence of defects. The model consists of a group of interconnected neural networks that work together to analyze different aspects of the code and identify potential issues. the unsupervised deep learning ensemble model for defect prediction is a powerful approach that can help to improve the accuracy and efficiency of software testing and development. By using unsupervised learning techniques to identify patterns and structures in the code, developers can more quickly and accurately identify potential defects and reduce the risk of introducing errors into the final product[12,13].

9. An IOT Based Approach for Defect Prediction

An IoT-based approach for defect prediction involves using data from sensors and other connected devices to identify potential defects in software systems. The approach uses machine learning and data analytics techniques to analyze the data and identify patterns or anomalies that may indicate the presence of defects[14]. an IoT-based approach for defect prediction can provide real-time insights into the performance of software systems, allowing developers to identify potential issues before they become major problems. By using data from connected devices to monitor system performance and predict defects, this approach can help to improve the reliability and efficiency of software systems[15].

III. CONCLUSION

Defect prediction is an important area of software engineering that aims to identify potential issues and defects in software systems before they can cause serious problems. There are various mathematical models and approaches that can be used for defect prediction, ranging from rule-based methods to machine learning and data analytics techniques. Each approach has its own strengths and limitations, and the choice of method will depend on the specific needs and characteristics of the software system being analyzed. Despite the challenges involved in defect prediction, there are many potential benefits to using these techniques in software engineering. By identifying and addressing defects early in the development process, software engineers can reduce the risk of introducing errors into the final product, improve overall system reliability, and increase user satisfaction. Moreover, by using machine learning and data analytics techniques to analyze system data and identify potential defects, developers can gain real-time insights into system performance and make informed decisions about how best to optimize and improve software systems. defect prediction is a valuable area of research in software engineering that can help to improve the quality, reliability, and efficiency of software systems. By using a combination of mathematical models and data analytics techniques, software engineers can gain a deeper understanding of system performance and make more informed decisions about how best to optimize and improve software systems.

REFERENCES

[1] Sahana, D. C. (2013). Software defect prediction based on classication rule mining. *Doctoral Dissertation*.

[2] Shao, Y., Liu, B., Wang, S. & Li, G. (2018). A novel software defect prediction based on atomic class-association rule mining. *Expert Systems with Applications*, *114*, 237-254.

[3] Jadhav, R. B., Joshi, S. D., Thorat, U. G. & Joshi, A. S. (2020). Software defect prediction utilizing deterministic and probabilistic approach for optimizing performance through defect association learning. *International Journal*, *8*(6).

[4] Yoo, K., Shukla, S. K., Ahn, J. J., Oh, K. & Park, J. (2016). Decision tree-based data mining and rule induction for identifying hydrogeological parameters that influence groundwater pollution sensitivity. *Journal of Cleaner Production*, *122*, 277-286.

[5] Yang, Z., Wan, C., Yang, Z. & Yu, Q. (2021). Using Bayesian network-based TOPSIS to aid dynamic port state control detention risk control decision. *Reliability Engineering & System Safety*, *213*, 107784.

[6] Wang, S., Wang, Y., Wang, D., Yin, Y., Wang, Y. & Jin, Y. (2020). An improved random forest-based rule extraction method for breast cancer diagnosis. *Applied Soft Computing*, *86*, 105941.

[7] Boutorh, A. & Guessoum, A. (2016). Complex diseases SNP selection and classification by hybrid association rule mining and artificial neural network—based evolutionary algorithms. *Engineering Applications of Artificial Intelligence*, *51*, 58-70.

[8] Srivastava, S., Haroon, M. & Bajaj, A. (2013, Sep). Web document information extraction using class attribute approach. In: *4th International Conference on Computer and Communication Technology (ICCCT)*, pp. 17-22. IEEE.

[9] Khan, R., Haroon, M. & Husain, M. S. (2015, Apr). Different technique of load balancing in distributed system: A review paper. In: *Global Conference on Communication Technologies (GCCT)*, pp. 371-375. IEEE.

[10] Haroon, M. & Husain, M. (2015, Mar). Interest attentive dynamic load balancing in distributed systems. In: *2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 1116-1120. IEEE.

[11] Haroon, M. & Husain, M. (2013). Analysis of a dynamic load balancing in multiprocessor system. *International Journal of Computer Science engineering and Information Technology Research*, *3*(1).

[12] Khan, W. & Haroon, M. (2022). An unsupervised deep learning ensemble model for anomaly detection in static attributed social networks. *International Journal of Cognitive Computing in Engineering*, *3*, 153-160.

[13] Khan, W. & Haroon, M. (2022). An efficient framework for anomaly detection in attributed social networks. *International Journal of Information Technology*, *14*(6), 3069-3076.

[14] Husain, M. S. & Haroon, D. (2020). An enriched information security framework from various attacks in the IoT. *International Journal of Innovative Research in Computer Science & Technology (IJIRCST)*.

[15] Khan, A. M., Ahmad, S. & Haroon, M. (2015, Apr). A comparative study of trends in security in cloud computing. In: *Fifth International Conference on Communication Systems and Network Technologies*, pp. 586-590. IEEE.