ISSN: 2424-8460(O)
2251-2608(P)

Original Research Article

Research on artificial intelligence-assisted software defect prediction and repair techniques

Yin Zheng

Liaoning University of Science and Technology,Anshan,Liaoning, 114051, China

Abstract: In this paper, the application of artificial intelligence technology in software defect prediction and repair is studied with respect to the shortcomings of traditional software defect prediction models in terms of interpretability and robustness. Focusing on the modeling method of Bayesian network in the field of software defect prediction, a software defect prediction model based on Bayesian network is established through data discretization, Bayesian network structure learning algorithm, and probabilistic inference technology. Further, this paper integrates Bayesian network with common predictors such as K-nearest neighbor, decision tree, logistic regression and so on in a soft-voting manner to construct an integrated software defect prediction model. Experimental simulations are carried out on six publicly available software defect datasets, and the results show that the integrated model based on Bayesian network significantly outperforms the traditional integrated model in terms of evaluation indexes such as F1, Recall, and G-Mean.

Keywords: Artificial intelligence; Software defect prediction; Bayesian networks; Causal analysis; Software defect repair that

1. Introduction

With the rapid development of information technology, software plays an increasingly important role in modern society. However, the existence of software defects (or known as software errors, faults) is a serious threat to the quality and reliability of software, which may lead to system crashes, data loss and even security problems. Therefore, software defect prediction and repair has become an important research direction in the field of software engineering. Software defect prediction aims to identify potential defect areas in software through historical data analysis, thus guiding the rational allocation of testing resources and improving testing efficiency and quality. On the other hand, software defect repair is to take corresponding measures to correct the defects after they are found, so as to ensure the normal operation of the software [1].

Traditional software defect prediction models are mainly based on statistics and machine learning techniques, such as logistic regression, support vector machine, decision tree and so on. These models can predict the probability of software defects to a certain extent, but there are still some limitations. First of all, there is insufficient interpretability, i.e., the prediction results of the model output are difficult to be understood by the developers, and it is difficult to guide the specific repair work. Second, the robustness is not strong, in the face of complex and changing software projects and data sets, the predictive performance of the model is often not stable enough. In addition, traditional models often ignore the complex relationship between variables, resulting in limited prediction accuracy [2-3].

2. Relevant technologies

2.1. Basic concepts and methods of software defect prediction

Software defect prediction is an important research direction in the field of software engineering, aiming to identify potential defect areas by analyzing the historical data of software projects, so as to guide the reasonable allocation of testing resources and improve testing efficiency and quality. The basic methods of software defect prediction include statistics-based prediction, machine learning-based prediction and deep learning-based prediction. These methods usually utilize software project metrics (e.g., lines of code, complexity, frequency of changes, etc.) as inputs to predict the probability of occurrence of software defects by constructing prediction models [4].

2.2. Theoretical Foundations of Bayesian Networks

A Bayesian network is a machine learning model based on probability theory and graph theory that is used to represent dependencies between variables, and consists of nodes and edges, where the nodes represent variables and the edges represent dependencies between the variables. a Bayesian network describes causal relationships between variables by defining a table of prior probabilities and conditional probabilities of the nodes. in the context of software defect prediction, a Bayesian network can be used to model the complex relationship between the data of the software measure and the probability of defect occurrence, thus enabling more accurate predictions. The Bayesian network consists of a graphical model (i.e., a directed acyclic graph) and a probability for each variable. defined as B=(G,P), where G=(V,E) is a directed acyclic graph representing the structure of the network, V denotes the set of nodes, E denotes the set of edges, and P is a vector of conditional probabilities, as described below. for a node v ∈ V, the parent of v is a directed connected node pointing to v. A simple Bayesian network is shown in **Figure 1.** [5].

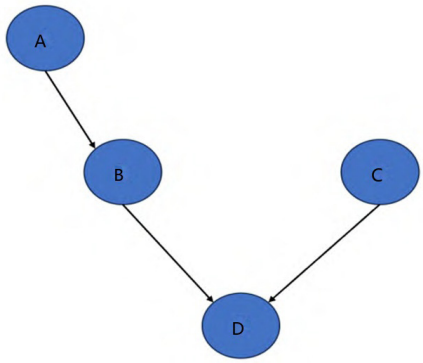

Figure 1. Schematic diagram of the Bayesian network that.

2.3. Data discretization techniques

Data discretization is the process of converting continuous data into discrete data, which is commonly used to deal with data sets with continuous attributes. In software defect prediction, since many metrics (e.g., lines of code, complexity, etc.) are continuous, some machine learning models, such as Bayesian networks, may face difficulties in dealing with continuous data. Therefore, data discretization techniques are widely used in software defect prediction to simplify the model structure and improve the prediction performance. Common data discretization methods include equal-width discretization, equal-frequency discretization and clustering-based

discretization [6].

2.4. Bayesian network structure learning algorithm that

Bayesian network structure learning algorithm is an algorithm used to automatically learn Bayesian network structure from data. In software defect prediction, since the complex relationship between software metrics and defect occurrence probability is difficult to be determined in advance, Bayesian network structure learning algorithms are needed to automatically construct prediction models. Common Bayesian network structure learning algorithms include algorithms based on scoring search (e.g., K2 algorithm, BIC scoring algorithm, etc.) and algorithms based on constraints (e.g., construction methods based on expert knowledge, etc.). These algorithms enable the model to more accurately describe the dependencies in the data by searching for the optimal network structure [7-8].

3. Integrated model building

3.1. Bayesian network-based software defect prediction model design

In constructing a Bayesian network-based software defect prediction model, we first need to determine the input features of the model, which usually include the code metrics of the software (e.g., lines of code, complexity), the metrics during the development process (e.g., frequency of changes, history of defect fixes), and other information related to the project. We then use these data to train Bayesian networks and learn the structure and parameters of the network. The learning of the structure of the Bayesian network is a key step that determines the dependencies between the variables, while the learning of the parameters determines the specific strength of these dependencies [9].

In order to enhance the interpretability of the model, we can use a Bayesian network with a simple structure, or reduce the number of input features through feature selection. In addition, we need to consider how to deal with missing data and outliers to ensure the robustness of the model. In the data preprocessing stage, we can use the data discretization technique to convert the continuous features into discrete features to adapt to the Bayesian network's processing requirements for discrete data.

3.2. Integration of model building strategies and soft voting methods that

Integrated learning is a method to improve the overall prediction performance by combining the prediction results of multiple base learners. In software defect prediction, we can integrate Bayesian networks with other commonly used software defect predictors (e.g., K-nearest neighbor, decision tree, logistic regression, etc.). In order to construct the integration model, we need to determine the integration strategy, among which the soft voting method is a commonly used strategy [10].

The soft voting approach allows each base learner to output a probability value (or confidence level) rather than a simple category label. These probability values are then averaged or weighted to obtain the final prediction. The advantage of this approach is that it can fully utilize the prediction information of each base learner and improve the prediction accuracy and robustness of the integrated model.

In the training phase, we can train Bayesian networks and other predictors separately and save their models. Then, in the prediction phase, we pass the input data to these models separately, collect their prediction results, and use the soft voting method to make the final prediction. In order to improve the efficiency of the integrated

models, we can employ parallel computing techniques to handle the prediction task of multiple base learners simultaneously.

4. Experimental results and analysis

4.1. experimental environment

In order to validate the effectiveness of the Bayesian network-based software defect prediction model and its integration method, we selected several representative software project datasets for experiments. These datasets cover software projects of different sizes, domains and development stages to ensure the generalizability and reliability of the experimental results.

Specifically, we selected the following datasets.

NASA dataset: This dataset contains metrics and defect information for several modules in the NASA software program, and is one of the benchmark datasets commonly used in the field of software defect prediction.

4.2. Evaluation indicators

In the task of software defect prediction, in order to comprehensively assess the performance of the model, we need to use a variety of evaluation indicators. These indicators can reflect the predictive ability of the model from different perspectives, help us better understand the strengths and weaknesses of the model, so as to carry out targeted optimization. The following are the definitions and calculation methods of several commonly used evaluation indexes.

4.2.1. F1 scores

The F1 score is a reconciled average of Precision and Recall, and is used as a combined measure of the predictive accuracy and recall ability of a model. It is especially suitable for datasets with imbalanced categories, because the precision or recall alone may not fully reflect the performance of the model.

Definition.

F1 score = 2*(precision*recall)/(precision+recall)

Calculation method.

Calculation of the precision rate: Precision rate = True case (TP)/(True case (TP) + False positive case (FP))

Calculate Recall:Recall=True cases (TP)/(True cases (TP)+False negative cases (FN))

Precision and recall were calculated by substituting them into the formula for the F1 score.

4.2.2. Recall

Recall is a measure of the number of true defects identified by the model as a proportion of the number of actual defects. It reflects how sensitive the model is to defects, i.e., how many real defects the model is able to identify.

Definition.

Recall = True cases (TP)/(True cases (TP) + False negative cases (FN))

Calculation method.

True cases (TP): the number of samples that the model predicted to be positive and were actually positive.

False Negative Examples (FN): the number of samples that the model predicts to be negative but are actually positive.

By statistically comparing the predicted results with the actual labels, the number of true and false-negative

cases can be calculated, and thus the recall rate.

4.2.3. G-Mean (geometric mean)

G-Mean is the geometric mean of precision and recall, which is used to synthesize and evaluate the performance of the model in the case of category imbalance. Similar to the F1 score, the G-Mean also considers both precision and recall dimensions, but is computed in a slightly different way.

4.3. Experimental results and analysis

The Bayesian network structure of the software defective data is a directed acyclic graph,the arrows between the nodes in **Figure 2.** indicate the causal relationship between the features.The experiment firstly discretizes the software defective data,and then,balances the data by using the random oversampling method with a random number of 42.Next,we learn the Bayesian network structure according to the k2 scoring method combined with the hill-climbing algorithm,and determine the conditional probability distribution of the nodes by combining with maximum likelihood estimation,which is shown in **Figure 5.** There are more nodes in the dataset, show the Bayesian network structure of Defective node of PC3 dataset as shown in **Figure 5.**, Defective node is the software defective node, taking the value of 0 and 1, which means that the software is defective and not defective, respectively, and the other nodes are the metrics of the software defects. **Figure 2.** The Bayesian network structure of the target node is illustrated in **Figure 2.**, the directed edge node LOC_ COMMENTS points to the node Defective, indicating that the number of annotated lines of the metrics of the software defective data leads to the occurrence of software defects, and other directed edges also indicate this causal relationship, and the pointing relationship of the directed edges in the Bayesian network characterizes the causal relationship between the nodes.

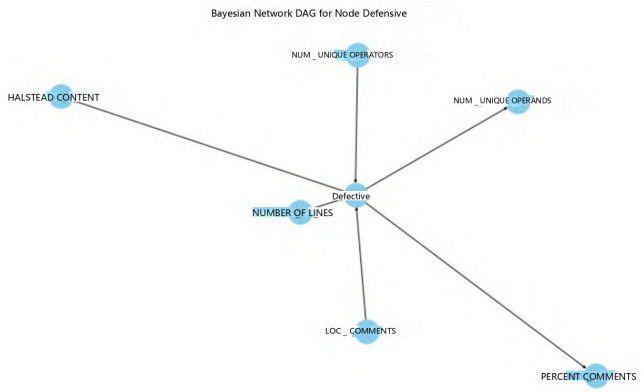

Figure 2. Bayesian network structure of the target node.

In summary, the integrated model based on Bayesian network shows significant advantages in software defect prediction, but there is still room for improvement. By optimizing the data quality, feature selection, model parameters and integration strategy, we can further improve the prediction performance of the model and provide more powerful support for software quality assurance.

5. Conclusion

This study mainly focuses on the software defect prediction model based on Bayesian network and its integration method. By constructing a Bayesian network-based prediction model and combining it with other commonly used software defect predictors to form an integrated model, we achieve significant performance

improvement in several evaluation metrics. Specifically, the integrated model outperforms the single model in terms of F1 score, recall, and G-Mean, demonstrating the effectiveness of the integrated approach in improving the performance of software defect prediction. Meanwhile, we also deeply analyze the advantages of the integrated model based on Bayesian network, including the diversity, complementarity and robustness of the model.

References

[1] Interpretability-Driven Sample Selection Using Self Supervised Learning for Disease Classification and Segmentation.[J]. Mahapatra Dwarikanath;Poellinger Alexander;Shao Ling;Reyes Mauricio.IEEE transactions on medical imaging.2021.23-24.

[2] Towards Visual Explainable Active Learning for Zero-Shot Classification.[J]. Jia Shichao;Li Zeyu;Chen Nuo;Zhang Jiawan.IEEE transactions on visualization and computer graphics.2021.45-48.

[3] LAL: Meta-Active Learning-based Software Defect Prediction[J]. Yubin Qu;;Fang Li;;Xiang Chen. International Journal of Performability Engineering.2023.56-57.

[4] ALTRA: Cross-Project Software Defect Prediction via Active Learning and Tradaboost[J]. Yuan Zhidan;Chen Xiang;Cui Zhanqi;Mu Yanzhou.IEEE Access.2024.67-68.