Analysis of the Effectiveness of Artificial Intelligence Technology in Defect Prediction in Software Testing

Hao Wu[1*]

[1] Guangdong Aerospace 706 Information Technology Co., Ltd., Guangzhou, 510663, China

[*] 787730736@qq.com

https://doi.org/10.70695/IAAI202501A4

Abstract

With the continuous increase in the scale and complexity of software, software testing faces enormous challenges. Defect prediction, as a crucial part of software testing, plays a key role in improving software quality and reducing costs. This paper conducts an in-depth analysis of the application effectiveness of artificial intelligence technology in software testing defect prediction. It introduces the related principles and methods in detail and demonstrates their strengths and weaknesses through actual cases and data analysis. The aim is to provide a reference for the software industry to better utilize artificial intelligence technology for defect prediction.

Keywords Artificial Intelligence; Software Testing; Defect Prediction

1 Introduction

In today's digital age, software has widely permeated into various fields, and its quality is directly related to user experience, business operations, and even social security. Software testing is an important means to ensure software quality, and defect prediction can identify modules that may contain defects in advance, allowing for a more rational allocation of testing resources, thereby improving testing efficiency and software quality [1][2]. Traditional defect prediction methods often rely on manual experience and simple statistical models, which have limited accuracy and efficiency when faced with complex software systems. The rapid development of artificial intelligence technology has brought new opportunities to defect prediction, and its powerful data processing and pattern recognition capabilities are expected to significantly enhance the effectiveness of defect prediction.

2 The principle of artificial intelligence technology in defect prediction

Logistic Regression is a widely used classification algorithm that predicts the probability of a sample belonging to a certain category [3]. Its basic formula is:

$$P(Y=1|X)=\frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots+\beta_n X_n)}} \tag{1}$$

P (Y=1|X) represented by a given feature $X=(X_1,X_2,\cdots,X_n)$, the probability that the sample belongs to category 1(defective) given the feature set, β is a parameter of the model, which is estimated through training data. The logistic regression model determines the optimal parameters by maximizing the likelihood function, thereby achieving classification predictions for the samples.

A decision tree is a classification model based on a tree structure [4], which constructs decision rules by recursively partitioning features. The construction process of a decision tree involves selecting the best feature for splitting according to indicators such as information gain, information gain ratio, or Gini index until certain stopping conditions are met. For example, the criterion for splitting a decision tree based on information gain is calculated as follows:

$$IG(D,A)=H(D)-\sum_{v=1}^{V}\frac{|D^v|}{|D|}H(D^v) \tag{2}$$

D represents a subset of feature A, the decision tree model is intuitive and easy to understand, capable of generating interpretable rules, which are convenient for comprehension and application.

Integrating the information, logistic regression is used to predict the probability of a sample belonging to a certain category, such as determining whether a software module has defects. It achieves this by weighting numerous input features and synthesizing the likelihood of the sample belonging to a specific category [5]. The model training process involves continuously adjusting these weights to make the prediction results as close as possible to the actual situation, thereby enabling the classification prediction of samples. Decision trees construct decision rules based on a tree-like structure. They recursively partition the input features; for example, in software defect prediction, they gradually refine the dataset based on certain code features (like complexity and change frequency). Each internal node corresponds to a decision condition on a feature, the branches represent the different outcomes of that condition, and the leaf nodes correspond to the final classification results, such as determining whether a software module has defects. Decision tree models are very intuitive, and the generated decision rules are easy to understand, allowing developers to clearly see which features were used to make the respective predictions.

3 Artificial Intelligence-Based Defect Prediction Methods

3.1 Data Source

From the code repository, a wealth of code-related information can be obtained. For example, through version control systems (such as Git), historical records such as the modification time, author, and specific content of each code file can be retrieved. This information is of great value for analyzing the stability of the code and potential defects. Test reports meticulously record defect information discovered during the software testing process, including the location of the defect, detailed description, severity, and discovery time, and are the core data source for defect prediction [6]. Factors such as the experience and skill level of developers may also affect software quality. Collecting information about developers' years of work experience, the number of projects they have participated in, and the roles they have played in projects can serve as auxiliary data, providing a more comprehensive perspective for defect prediction.

3.2 Logical Framework

In the field of software testing, the integration of artificial intelligence technology into the defect prediction process has formed a systematic overall framework. Starting from the data level, it first widely collects information such as the number of lines of code, complexity, modification history from the code repository, defect location, severity in test reports, as well as auxiliary data like the work experience of developers. After that, these data are cleaned, missing values and outliers are processed, and the scale is standardized. In the feature processing phase, feature selection is carried out using filter methods, wrapper methods, and embedded methods, while feature extraction is conducted from the perspectives of code structure, text mining, and machine learning. In the model application stage, models such as logistic regression and neural networks are selected based on data characteristics and task requirements, training sets and test sets are divided for model training, and parameters are optimized through methods like cross-validation [7]. Finally, the model's performance is evaluated using metrics such as accuracy and recall rate. The various stages of the overall framework are closely linked and progressive, aiming to accurately predict software defects and help improve software quality and testing efficiency. The overall framework is shown as Figure 1.

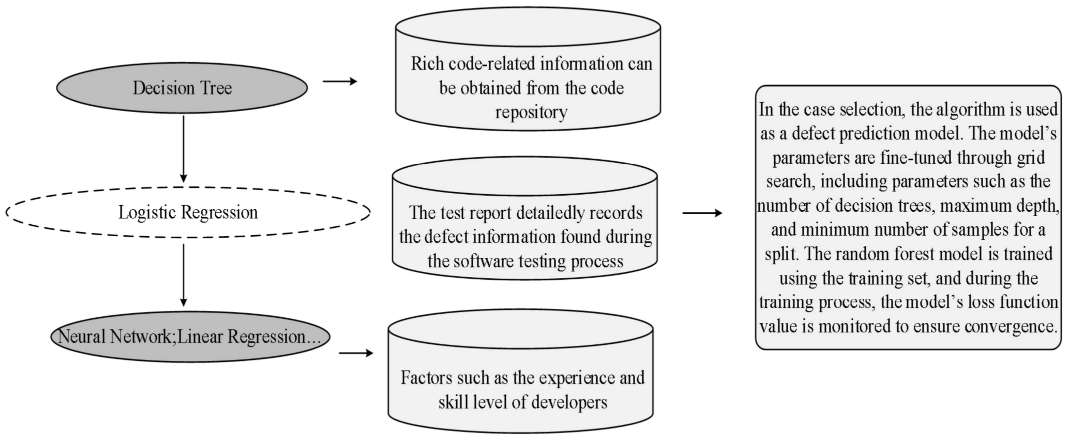
Decision Tree
Logistic Regression
Neural Network;Linear Regression...
Rich code-related information can be obtained from the code repository
The test report detailedly records the defect information found during the software testing process
Factors such as the experience and skill level of developers
In the case selection, the algorithm is used as a defect prediction model. The model's parameters are fine-tuned through grid search, including parameters such as the number of decision trees, maximum depth, and minimum number of samples for a split. The random forest model is trained using the training set, and during the training process, the model's loss function value is monitored to ensure convergence.

Fig. 1. Research Framework

3.3 Model Training and Evaluation

Step 1: Divide the dataset into training and testing sets. The collected data is divided into a certain proportion, usually the training set accounts for 70% - 80%, and the testing set accounts for 20% - 30%. The training set is used to train the model, allowing it to learn patterns and regularities in the data; the testing set is used to evaluate model performance. To improve the model's generalization ability, stratified sampling can be used to ensure that the proportions of various samples in the training and testing sets are consistent with the original data, avoiding model overfitting or underfitting due to unbalanced sample distribution.

Step 2: Select appropriate models and parameters. Choose suitable artificial intelligence models based on data characteristics and the requirements of the defect prediction task. For binary classification problems (determining whether a software module has defects), models such as logistic regression and decision trees are more applicable; for multi-classification problems (determining the type of defect), models like support vector machines and neural networks may perform better. During the parameter tuning process, methods such as grid search and random search can be used to traverse different parameter combinations and select the parameters that optimize model performance.

Step 3: Train the model. Use the training set to train the selected model, and adjust the model parameters continuously to allow the model to gradually learn the patterns in the data. During the training process, optimization algorithms such as batch gradient descent and stochastic gradient descent are used to update the model parameters in order to minimize the loss function. For example, for logistic regression models, the cross-entropy loss function is commonly used to measure the difference between the predicted values and the true values. By continuously iterating and updating the parameters, the value of the loss function is gradually reduced until the model converges.

4 Conclusion

The research indicates that with the continuous development of artificial intelligence technology, new algorithms and models will continue to emerge, which is expected to further improve the accuracy and efficiency of defect prediction. At the same time, the integration of cross-domain technologies, such as combining big data analytics and IoT (Internet of Things) technology to obtain more multi-dimensional data, will provide a more comprehensive perspective for defect prediction. Strengthening the research on model interpretability will not only allow artificial intelligence technology to deliver accurate results in software testing defect prediction but also enable developers to clearly understand the basis for predictions, thereby being more widely and deeply applied in the software industry and promoting the continuous improvement of software quality.

Conflicts of Interest

The authors declare no conflicts of interest.

References

1. Gu, W. H., Yu, S. F., Zhou, G. C., et al. (2024). Application of artificial intelligence technology in the field of software testing. Information Technology and Standardization, (11), 92-96.
2. Wei, H. Y. (2023). Design research of software reliability automatic testing system under artificial intelligence technology. Journal of Jiujiang University (Natural Science Edition), 38(04), 65-68.
3. Wang, D. (2023). Research on the application of artificial intelligence technology in the field of software testing. Modern Computer, 29(12), 55-59.
4. Gu, Y. S. (2023). Research on the application of artificial intelligence-based software testing. Intelligent IoT Technology, 6(02), 16-18.
5. Hu, Z. Q., & Zhi, C. J. (2021). Development and application of software testing in the era of artificial intelligence. Electronic World, (24), 101-103.
6. Zhou, C. (2021). Exploration of artificial intelligence-based software testing. Information Recording Materials, 22(10), 239-240.
7. Dai, J. X., Yang, P., & Zhao, J. X. (2021). Integration of artificial intelligence and software testing courses in higher vocational education. Industrial Control Computer, 34(05), 91-92.

Biographies

1. **Hao Wu** graduated from Northeastern University with a B.S. degree in Measurement and Control Technology and Instruments. An intermediate engineer with 10 years of testing work experience.

人工智能技術在軟件測試缺陷預測中的有效性分析

吳皓

摘要：隨著軟件規模和復雜性的不斷增加，軟件測試面臨著巨大的挑戰。缺陷預測作為軟件測試的重要組成部分，在提高軟件質量和降低成本方面發揮著關鍵作用。本文深入分析了人工智能技術在軟件測試缺陷預測中的應用效果。它詳細介紹了相關的原理和方法，並通過實際案例和數據分析論證了它們的優缺點。旨在為軟件行業更好地利用人工智能技術進行缺陷預測提供參考。

關鍵詞：人工智能；軟件測試；缺陷預測