International Journal of Research in Engineering, Science and Management
Volume 7, Issue 4, April 2024
https://www.ijresm.com | ISSN (Online): 2581-5792

Software Defect Density Prediction Using Deep Learning

V. Raaga Varsini[1], S. Abitha Lakshmi[2], Devarshi[3], B. Gokul[4*]
[1]Assistant Professor, Department of Computer Science and Engineering, Velalar College of Engineering and Technology, Erode, India
[2,3,4]Student, Department of Computer Science and Engineering, Velalar College of Engineering and Technology, Erode, India

***Abstract*: Software defect prediction is the use of various approaches and procedures to discover and anticipate future flaws in a software project before they become costly and disruptive problems. Organizations may use data analysis, machine learning, and historical defect data to make educated decisions about resource allocation, software testing methodologies, and, ultimately, software product quality. Software defect prediction is the technique of identifying software modules that are likely to have flaws. The suggested system is intended for software defect prediction, combining PCA with a software business analysis model and employing Rank SVM as its primary predictive modeling approach. It includes data collecting from historical defects and relevant data sources, preprocessing to clean and convert the data, feature selection to identify critical indicators, and training a PCA using a software business analysis model utilizing Rank SVM. This methodology, once installed, forecasts the risk of faults in software modules based on their characteristics. The system's output influences resource allocation and testing procedures, hence improving software quality and development efficiency. Continuous monitoring and adjustments assure continuing correctness, making this an efficient option for proactive defect control in software development.**

***Keywords*: Defect Density Prediction, Data Sparsity, Machine Learning.**

1. Introduction

A. Defect Density Prediction

In the realm of software development and quality assurance, the ability to forecast defect density is paramount for ensuring software product reliability and performance. Defect density, denoting the quantity of bugs or defects within a software system relative to specific measurement units like lines of code, function points, or modules, plays a pivotal role. This predictive capability empowers software development teams to proactively identify potential quality issues, optimize resource allocation, and make informed decisions throughout the development lifecycle. Software defects span from minor nuisances to critical vulnerabilities that can trigger system failures or security breaches. Thus, the skill to anticipate defect density is indispensable for the efficient management of software projects and the delivery of top-notch products to end-users.

B. Data Sparsity

In today's data-driven landscape, both organizations and individuals constantly gather and analyze extensive data to glean insights, make informed choices, and foster innovation. Nevertheless, not all data holds the same value, and a key challenge in data analysis often emerges as data sparsity, wherein the available data is either incomplete or insufficient for meaningful analysis or modeling. Data sparsity takes various forms, including missing values, limited data points, or imbalanced datasets, and it transcend s diverse domains like finance, healthcare, natural language processing, and machine learning. Recognizing and addressing data sparsity is crucial, as it can significantly affect the quality and dependability of analytical outcomes.

C. Machine Learning

In the digital era, the exponential growth in data collection and generation has presented both remarkable opportunities and challenges. Within this landscape, machine learning, a subset of artificial intelligence, has emerged as a formidable tool for leveraging this data to automate tasks, make predictions, and extract valuable insights. Its transformative impact spans across diverse industries such as healthcare, finance, transportation, and entertainment, fundamentally altering our problem-solving and decision-making approaches. At its core, machine learning is the science of enabling computers to learn from data and make informed predictions or decisions without explicit, fixed programming. Instead, these algorithms continuously adapt and enhance their performance as they process more data, filling crucial roles where conventional programming methods fall short.

2. Literature Review

In their paper, Chakra Tanti thamthavorn [1] et al. address the critical issue of defect prediction models trained on imbalanced datasets, where the proportion of defective and clean modules is uneven. The authors highlight the conflicting findings in prior research on the impact of class rebalancing techniques on defect prediction model performance, attributing these disparities to variations in dataset choices, classification methods, and performance metrics. To resolve this uncertainty,

*Corresponding author: gokul161002@gmail.com their study comprehensively investigates the effects of four widely-used class rebalancing techniques on ten commonly-applied performance measures and the interpretability of defect prediction models. By analyzing a diverse set of 101 datasets from both proprietary and open-source systems, the authors ultimately recommend the adoption of class rebalancing techniques when aiming to enhance the identification of software defects, particularly in terms of Recall, providing valuable insights for quality assurance teams.

In their paper, Yenny Villuendas Rey [2] et.al. Introduce the significance of software defect prediction within the realm of software engineering, emphasizing the key metric of defect density (DD) as a pivotal measure for assessing software process effectiveness. They advocate for the adoption of a transformed k-nearest neighborhood output distance minimization (TkDM) algorithm as a novel approach for predicting DD in software projects.

This approach is pitted against established techniques such as statistical regression, support vector regression, and neural networks to evaluate its predictive accuracy, marking a notable contribution to the field of software prediction.

In their system, Lei Qiao [3] et al. address the increasing complexity of software systems, emphasizing the pressing need to predict software defects automatically. Such predictions are essential for optimizing resource allocation among developers. Despite various existing approaches for defect identification and rectification, their performance falls short of expectations. Therefore, their paper introduces an innovative method harnessing deep learning techniques to forecast the number of defects in software modules. Their methodology involves preprocessing a publicly available dataset, which includes log transformation and data normalization, followed by data modeling to prepare input for the deep learning model. Subsequently, the modeled data is fed into a specially designed deep neural network-based model for defect prediction. Evaluation on established datasets demonstrates the accuracy and superiority of their approach, with a substantial reduction of over 14% in mean square error and an increase of more than 8% in squared correlation coefficient compared to state-of-the-art methods on average.

In this system, Mr. Suraj Rathaur [4] et al. propose software quality as a key indicator of a software system's ability to align with customer requirements. The maintenance of satisfactory software quality hinges significantly on defect density, a pivotal factor influenced by the escalating software complexity. This paper introduces a machine learning-driven model, utilizing the multiple linear regression technique, to forecast defect density in forthcoming versions of open-source software (OSS). Data pertaining to OSS versions has been sourced from the Git version control system, encompassing software metrics like source lines of code, developer count, commit frequency, and code churn, all chosen as predictor variables. Furthermore, normality tests have been conducted on each variable to affirm their adherence to a normal distribution.

In their work, Iqbal H. Sarker [5] et al. emphasize the pivotal role of Deep Learning (DL) in the Fourth Industrial Revolution (4IR or Industry 4.0), highlighting its significance as a core technology within the domains of machine learning (ML) and artificial intelligence (AI). DL, stemming from artificial neural networks (ANN), has garnered immense attention due to its data-driven learning capabilities, finding extensive applications across diverse fields such as healthcare, visual recognition, text analytics, cybersecurity, and more. Nonetheless, constructing effective DL models poses formidable challenges owing to the dynamic and variable nature of real-world problems and data. Additionally, the lack of interpretability often renders DL methods as black-box solutions, hindering their widespread adoption. This article offers a structured and comprehensive overview of DL techniques, featuring a taxonomy encompassing various real- world tasks including supervised, unsupervised, and hybrid learning, among others.

3. Existing System

Delivering a reliable and high-quality software system to clients poses a significant challenge throughout the software development and evolution process. Defect density serves as a crucial metric to assess system quality, often required by practitioners during both development and operational phases to gauge software reliability. However, predicting defect density before module testing can be time- consuming, prompting managers to seek predictive models that can identify potentially defective modules. This approach aims to reduce testing costs and optimize resource allocation. A central issue in software defect datasets is the inherent data sparsity within defect density, which can introduce bias into predictions. To address this challenge, we employ deep learning techniques, which have proven effective in handling sparse data. Our constructed deep learning model undergoes

4. Proposed System

The suggested system is a sophisticated approach to software fault prediction that employs PCA and Rank SVM techniques. It begins with the gathering and preparation of historical software fault data and pertinent variables. Feature selection reveals significant predictors of faults. Rank SVM is used to rank software modules based on their likelihood of harboring flaws and feature relevance. Simultaneously, PCA generates a prediction model based on the rankings. The system's evaluations use defined measures to assure accurate forecasts. The integrated method directs resource allocation for software testing, promotes quality, and is intended for ongoing improvement to meet growing development requirements.

A. Load Data

This module collects and loads Eclipse, Lucene, Mylyn, and pde datasets from diverse sources to anticipate software defects. This data usually contains historical information on software development projects, such as code metrics, defect reports, developer experience, and other pertinent elements. Loading data entails extracting, formatting, and arranging information into a structured dataset ready for analysis. This dataset serves as the basis for developing prediction algorithms to detect probable flaws in software code.

B. Data Preprocessing

Data preprocessing is a vital component in defect prediction. It entails preparing raw software development data for analysis. This process consists of numerous essential activities, including data cleansing, feature engineering, and normalization. Data cleaning include removing missing numbers, outliers, and inconsistencies that may bias the results.

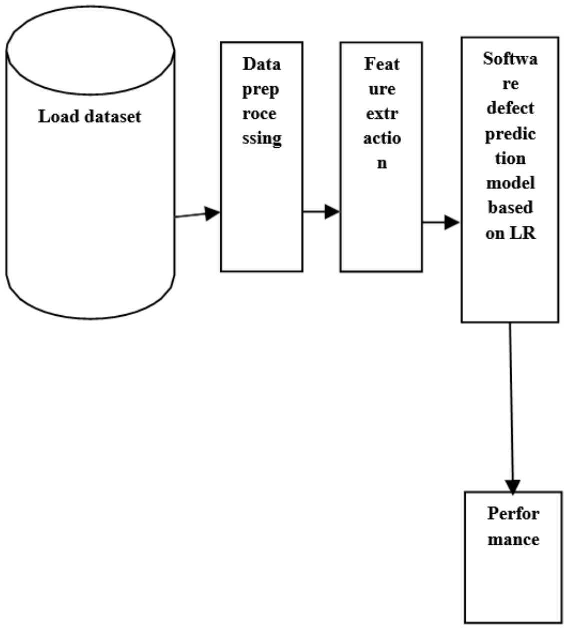

Fig. 1. Block diagram

C. Feature Selection

Feature selection is a vital step in identifying the most relevant features. This module reduces dimensionality and improves model performance by identifying important indications linked to software faults. The selection process can be guided by a variety of methodologies, including statistical tests, feature significance ratings from machine learning models, and domain expertise. Effective feature selection improves model performance by minimizing noise and dimensionality.

D. Training and Testing

This lesson covers training a PCA with a software business analysis model utilizing Rank SVM on pre- processed data. During the training phase, the model learns about the correlations between the selected characteristics and the likelihood of software module errors. The training dataset, which is usually a subset of the available data, is utilized to train the model. It offers historical instances of software modules, together with defect information.

E. Evaluation and Performance

The module measures the model's performance using a variety of metrics, including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R- squared, and others. These metrics indicate how effectively the model predicts software faults.

5. Result Analysis

Table 1
Comparison table

Algorithm	Accuracy
Existing system	75
Proposed system	81

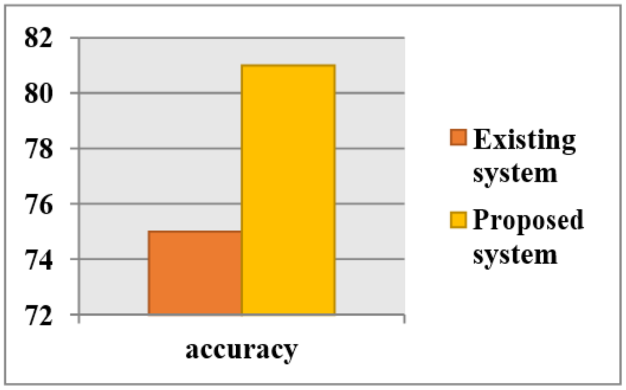

Fig. 2. Comparison graph

There is a noticeable increase in defect prediction capabilities when comparing the algorithm accuracy of the proposed system to the current system. The proposed system exhibits a notable development with an accuracy rate of 81%, compared to the previous system's 75% accuracy. The predictive modeling procedure of the suggested system makes use of advanced methods PCA with Rank Svm. It includes all the necessary procedures to proactively detect and handle possible flaws in software modules, such as data collection, preprocessing, feature selection, model training, and deployment. The suggested system's improved accuracy, together with its ongoing monitoring and improvement processes, highlight how well it may improve resource allocation, testing methodologies, and overall software development efficiency to produce software that is of a higher caliber.

6. Conclusion

To summarize, the suggested software defect prediction system, which combines PCA with a software business analysis model utilizing Rank SVM, provides a robust and proactive approach for improving software quality and resource allocation efficiency. It accurately forecasts faults using historical data and complex rating methodologies, allowing enterprises to invest resources wisely and improve overall software quality. Continuous monitoring and flexibility keep the system relevant in the changing world of software development, providing as a vital tool for effective defect management and avoidance, resulting in a more dependable and simplified software development process.

7. Future Work

Future research in this area should focus on improving the integration of sophisticated machine learning and ranking algorithms in order to increase fault prediction accuracy. Exploring real-time data streams for dynamic defect monitoring and expanding the system to accommodate changing software development processes are also attractive directions. It will be vital to develop user-friendly interfaces for widespread adoption, as well as to investigate automation for defect management and response. Collaboration between researchers and industry practitioners for benchmarking and standardizing defect prediction algorithms is critical for advancing the discipline and facilitating the widespread implementation of proactive defect management solutions across various software development ecosystems.

References

[1] Tantithamthavorn, Hassan, and Matsumoto, "Examining the Influence of Class Rebalancing Techniques on the Performance and Interpretability of Defect Prediction Models," in IEEE Transactions on Software Engineering, vol. 46, no. 11, pp. 1200 to 1219, November 2020.

[2] Lopez-Martín, Villuendas-Rey, Azzeh, Bou Nassif, and Banitaan, "Transformed k-Nearest Neighborhood Output Distance Minimization for the prediction of defect density in software projects," Journal of Systems Software, Volume 167, September 2020.

[3] Qiao, Li, Umer, and Guo, "Deep Learning- Based Software Defect Prediction," in Neurocomputing, April 2020.

[4] Rathaur, Kamath, and Ghanekar, "Software defect density prediction based on multiple linear regression," in the proceedings of the 2nd International Conference on Inventive Research in Computer Applications (ICIRCA), pp. 434 to 439, July 2020.

[5] Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications, and Research Directions," in journal Social Networking and Computational Sciences, vol. 2, no. 6, pp. 420, November 2021.

[6] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh and K. Shaalan, "Speech Recognition Using Deep Neural Networks: A Systematic Review," in IEEE Access, vol. 7, pp. 19143-19165, 2019.

[7] S. Wang, T. Liu, J. Nam and L. Tan, "Deep Semantic Feature Learning for Software Defect Prediction," in IEEE Transactions on Software Engineering, vol. 46, no. 12, pp. 1267-1293, 1 Dec. 2020.

[8] T. Hoang, H. Khanh Dam, Y. Kamei, D. Lo, and N. Ubayashi, "DeepJIT: An end-to-end deep learning framework for just-in-time defect prediction," in Proceedings of the 16th International Conference on Mining Softw. Repositories (MSR), May 2019, pp. 34–45.

[9] D. Chen, X. Chen, H. Li, J. Xie, and Y. Mu, "DeepCPDP: Deep learning based cross-project defect prediction," IEEE Access, 7 (2019), 184832–184848.

[10] G. Zhao and J. Huang, "DeepSim: Deep learning code functional similarity," in the Proceedings of the 26th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng., October 2018, pp. 141–151.