TRANSACTIONS ON EMERGING SCIENCES
Ideas at the Edge of Discovery
ISSN 3136-2710

ORIGINAL RESEARCH ARTICLE

MACHINE LEARNING TECHNIQUES FOR SOFTWARE DEFECT PREDICTION: A SYSTEMATIC LITERATURE REVIEW

Umeza Riaz
Department of Computer Science,
University of Agriculture, Faisalabad, Pakistan
Corresponding Author Email: umezariaz@gmail.com

ABSTRACT

Developers and organisations are facing challenges to ensure software quality because software complexity is growing rapidly. Software defect prediction is used to predict defects early by using machine learning techniques on historical data in the development process. Early defect prediction improves the accuracy, quality, and reliability of software. It also reduces overall cost and time consumption. This paper includes a systematic literature review of machine learning methods used in software defect prediction. A total of 15 research articles published in the past 10 years were examined and selected from reliable resources, including IEEE Xplore, Science Direct, and Google Scholar. The research evaluates and compares the performance, pros, and cons of many machine learning techniques, such as Naive Bayes, Decision Trees, Artificial Neural Networks, Random Forest, and Support Vector Machine. Many studies claim that Random Forest has achieved the most significant prediction of accuracy among all studied techniques because it can efficiently handle noisy and imbalanced data. On high-dimensional datasets, Support Vector Machine works efficiently while Naive Bayes is effective for limited data resources. However, the model's reliability remains influenced by factors such as high computational power, data quality, and limited cross-project generalisation. Thus, machine learning plays an important role in improving the quality, reliability, and accuracy of software defect prediction.

Keywords: *Software Defect Prediction, Machine Learning, Defect Classification, Software Quality, and Predictive Modelling*

1. INTRODUCTION

Software complexity is increasing, and people are becoming more dependent on it over time. A major factor that can considerably affect software maintainability is software defects. Software maintainability predictions include predictions of errors and bugs that may occur in software during its functioning. Software developers are encouraged to avoid probable defects and bugs that might occur, and software predictions can help improve software design to increase its maintainability, since it is one of the biggest software development challenges.

Artificial intelligence (AI) and machine learning (ML) approaches have recently extended several prediction methods to improve software quality through several software testing approaches. On the other hand,

ISSN 3136-2710

developers still need to pay more attention to machine learning and deep learning to improve software quality. Insufficient datasets could be a major concern hindering the application of machine learning approaches. Many studies have indicated that several machine learning approaches have been developed to predict software maintainability using different algorithms (Albattah & Alzahrani, 2024).

A technique that is applied to early stages of the software development life cycle by using machine learning algorithms to predict past data is known as “software defect prediction” (Albattah et al., 2024). SDP's main objective is to build high-quality and dependable software while utilising accessible resources. Early fault detection helps in timely correction, which improves software maintainability (Batool et al., 2022). Basically, SDP serves as a systematic approach for developing software with reduced time and cost. As a result, software developers are able to efficiently prioritise utilisation of resources across each phase of the software development life cycle.

A variety of ML techniques have been studied until now to predict faults in software components to improve software quality and lower software testing expenditures. The most used approach for fault prediction is that a set of software parameters with labelled data is examined, related to several software parameters, and then machine learning techniques are applied to the datasets (Jorayeva et al., 2022). A lot of ML techniques are used in SDP, which include decision trees, Naïve Bayes (NB), support vector machines (SVM), and random forests (RF). Historical data retrieved from software libraries is evaluated to identify the quality and dependability of software components.

According to an estimate, the global technology outage in July 2024 was caused by a defect in a software update that resulted in economic loss and widespread service disruption across critical sectors, including aviation, finance, education and health care. Software maintenance consumes about 70% of software development costs. Minimising software defects and bugs can improve software maintainability and make it easier and less expensive. Thus, it is important to develop and maintain high-quality and efficient software at the lowest cost.

Several previous studies focused mainly on evaluating individual machine learning algorithms separately rather than providing a comparative analysis of multiple techniques under different dataset conditions. Another important research gap is that many prediction models are developed and tested using limited datasets, which reduces their applicability to real-world industrial software systems. Furthermore, there are limited studies on software defects prediction, Artificial Intelligence, and machine learning.

Therefore, to fill these gaps, I have conducted this study (1) to evaluate the impact of dataset characteristics, feature selection and class imbalance on prediction accuracy, (2) to examine the major challenges associated with machine learning-based software defect prediction models, (3) to compare the performance of different machine learning algorithms in predicting software faults, and (4) to identify the strengths and limitations of commonly used machine learning techniques in software fault prediction.

ISSN 3136-2710

2. METHODOLOGY

What role does machine learning play in software defect prediction, and how does it improve the dependability and accuracy of software? Which machine learning techniques are most effective at predicting bugs in software? What roles do machine learning techniques such as Naive Bayes, Random Forest, SVM, Decision Tree, and KNN play in detecting software defects?

Which machine learning technique is most effective for predicting the accuracy of software modules that are prone to defects? What are the benefits of applying machine learning techniques in software defect prediction? What main challenges and restrictions does machine learning usually face when applied to software defect prediction? What future enhancements can increase the performance of software defect prediction models?

As such, the research conducted is exploratory. Due to the nature of the study, which is qualitative-based, I wanted to find diverse kinds of links. I conducted this study to explore the significance of machine learning approaches in software defect prediction. Additionally, to determine the most effective machine learning model for detecting defect-prone software modules. As per the requirement to write ten questions, this report is based on different questions, which I have mentioned in the above paragraphs.

I conducted this secondary research with the assistance of several publicly available data sources. I worked on research projects based on existing literature. First, there needs to be more resources (time) available to do primary research due to space. I chose desk-based research/secondary research because I can answer the provided question in the introduction section about how this existing study is based on a different theme by gathering all the necessary data and exploring these questions. For this research, I used only academic research, not grey literature.

This secondary research relies on academic literature: I used IEEE Xplore, Google Scholar, Scopus, Science Direct, and Research Gate for my current report. I utilised the provided keywords to look for my issue in the available study, published from 2000 to 2026. I used the keywords “Software Defect Prediction, Machine Learning, Defect Classification, Software Quality, and Predictive Modelling”. I select all English language publications, store them in the library in Zotero, and then carefully review every detail of my relevant data. It should be noted that I have not used ChatGPT for this report; however, I have, of course, taken ideas from it. I have corrected all the reports and paragraphs using Grammarly.

After collecting relevant research papers, the selection of papers was performed using the inclusion and exclusion methods. All the papers, which were incomplete, irrelevant, and duplicate, were removed from review. After applying exclusion criteria, only required and refined papers were used for further analysis and processing. When selection is done, the papers that were selected go through the quality assessment process. In this process, each paper was processed following its research questions, methodology, datasets used, experimental evaluation, and results discussion.

There were limitations in this study, such as inadequate primary research and data collection resources. While secondary research has certain shortcomings, primary research is still far superior. Furthermore, there needs to be more time for an in-depth investigation of this research. Thirdly, I needed to improve in using the

ISSN 3136-2710

research instrument for understanding. Therefore, it is essential to carefully consider these limits while analysing the research's results and observations.

3. RESULTS

Software fault prediction has become more significant in research areas because the complexity and size of software systems are increasing day by day in software engineering. Researchers are applying machine learning approaches in software defect prediction because they easily learn patterns from old data, improve software quality, reduce maintenance costs, and provide accurate predictions (Lessmann et al., 2008).

3.1 SOFTWARE FAULT PREDICTION

Software fault prediction is a process used to identify fault-prone software modules. Prediction models are built by using various metrics like cohesion, coupling, inheritance, code complexity, and many more. Machine learning models learn patterns from previous projects, so historical data is also very significant (Hall et al., 2013).

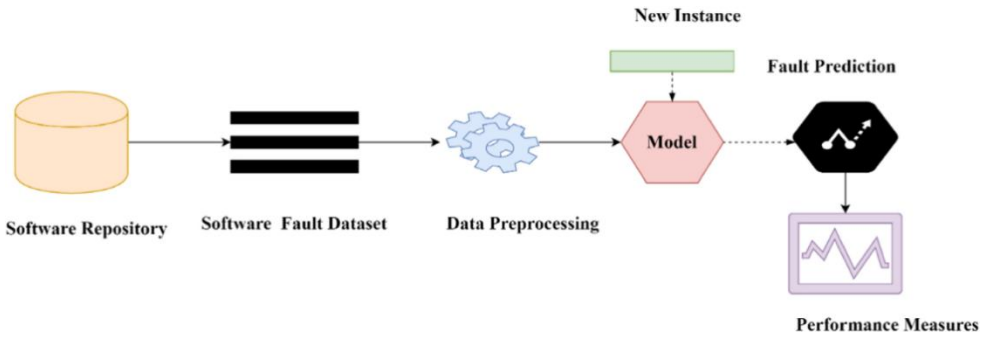

Fig. 1. An overview of the software fault prediction process

The early defect prediction helps improve software quality and ensure software dependability; it is observed by researchers. It becomes too complex to test the complete system thoroughly; it's better to test only risky areas, and it's done by prediction models (Catal & Diri, 2009).

3.2 MACHINE LEARNING TECHNIQUES

Software fault prediction uses machine learning algorithms so that it can easily categorise modules with defects and without defects automatically. There are various techniques that have been applied in software fault prediction, such as decision trees, Naïve Bayes, support vector machines (SVM), K-nearest neighbours (KNN), artificial neural networks (ANN), and random forests (Malhotra, 2015).

Researchers have found that machine learning approaches provide more quality compared to traditional methods. They also observed that improving software performance and accuracy feature selection methods are useful (Wahono et al., 2014).

Decision Trees

ISSN 3136-2710

One of the most useful machine learning approaches in software fault prediction is the decision tree. It is very useful because it is very easy and simple to understand and use. It creates a tree-like structure of software modules, which helps developers make decision rules easily.

These models are very easy to interpret and can handle numerical and categorical data, as observed by Koru and Liu (2005). However, the most common problem in this model, which is detected by researchers, is overfitting when a model is trained on large datasets.

Naive Bayes

The machine learning approach is based on Bayes' theorem. This technique uses probabilistic calculations for defects based on software measurements.

Menzies et al. (2007) found that this approach takes less processing power compared to other approaches, and they also observed that it works best with limited trained software datasets. The main drawback of this method is the reduction in prediction accuracy of complex datasets because of the assumption that every attribute is independent.

Support Vector Machine

Supervised machine learning algorithms, which are mostly used for classification issues in software fault prediction, are known as support vector machines (SVMs). Its working process is to find the best boundary between defective and non-defective software modules.

In 2008, SVM's performance was reviewed on NASA defect samples by Elish and Elish, and they found that sometimes SVM provides excellent and occasionally better predictions than other typical machine learning algorithms.

They also discovered that SVM performs well for high-dimensional datasets, but SVM requires high computer power and correct parameter adjustments for large datasets.

Random Forest

The machine learning algorithm, which is considered the most effective for software defect prediction, is Random Forest. Because it prevents overfitting and increases the performance of classification. Basically, it is a learning algorithm which combines multiple decision trees so that accurate predictions can be increased.

Random Forest usually performs better in accurate prediction than other individual classifiers. It was found by Lessmann et al. (2008) when he analyzed multiple classification models.

Random forest is also considered the most suitable real-world software application because it has the ability to manage uneven and messy datasets.

Artificial Neural Networks

It is observed by researchers that ANN performs well in software defect prediction with huge and complicated datasets. ANNs are modelled after the human brain foundation. They are mostly used for complicated prediction tasks. They can help in learning the relationship between software defects and measurements.

ISSN 3136-2710

The drawback is that it also requires more computational power and huge training datasets for defect prediction compared to many other simple machine learning algorithms (Ghotra et al., 2015).

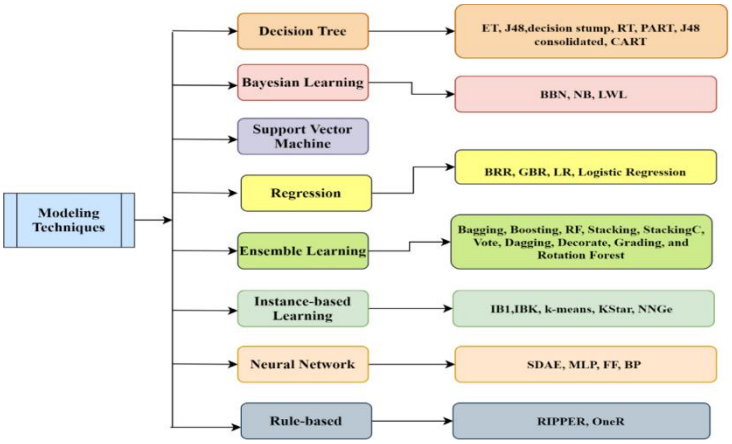

Fig. 2. Classification of modelling techniques. Abbreviations: ET (Extra Tree), RT (Random Tree), CART (Classification & Regression Tree), PART (Pruning rule-based), BBN (Bayesian Belief Network), LWL (Locally weighted learning), BRR (Bayesian Ridge), GBR (Gradient Boosting), LR (Linear Regression), RF (Random Forest), NNGe (Nearest neighbour with generalisation), SDAE (Stacked denoising auto-encoder), MLP (Multilayer Perceptron), FF (Feedforward), BP (Back Propagation), RIPPER (Repeated incremental pruning).

Technique	Key Strength	Key Limitation
Decision Tree	Simple, easy to interpret; handles numerical and categorical data	Prone to overfitting on large datasets
Naive Bayes	Low computational demand; effective with limited training data	Assumes attribute independence, reducing accuracy on complex data
Support Vector Machine	Strong performance on high-dimensional datasets	Requires high computing power and careful parameter tuning
Random Forest	Reduces overfitting; handles noisy and imbalanced data well	Less interpretable than a single decision tree
Artificial Neural Network	Captures complex, non-linear patterns in large datasets	Needs significant computational power and large training data

Table 1. Comparison of machine learning techniques used in software defect prediction, based on the reviewed studies.

3.3 CHALLENGES

As software defect prediction has become more accurate because of machine learning techniques, there are still many issues that are encountered by researchers, which result in lower accuracy and dependability of traditional prediction models. Because of such issues, it has become very challenging to build a system which functions effectively for all types of software projects.

ISSN 3136-2710

The most crucial challenge in software defect prediction is the quality of data. It becomes difficult for a machine learning algorithm to learn, which also affects the reliability and accuracy of prediction models. The problem is that datasets sometimes consist of incomplete, messy, and inaccurate data. Often, redundancy and inaccuracy of data values exist. Sometimes essential data is missing entirely (Bhandari et al., 2023).

Another major issue is class imbalance. In most real-world problems, the number of defective modules is smaller than the number of non-defective modules. As a result, machine learning models mostly focus on the majority class and cannot accurately detect actual defective modules. Because of this, the main purpose of defect prediction is not fully achieved. Hence, class imbalance must be carefully addressed to build accurate models (Batool & Khan, 2022).

For building prediction models, there are many software metrics available that include coupling, cohesion, complexity, and inheritance. However, selecting the most suitable and relevant features is not always easy. The model might become more complex and less effective due to the addition of irrelevant and unnecessary features. Hence, when dealing with huge and diverse datasets, it is crucial but challenging to select appropriate features (Ghotra et al., 2015).

Parameter tuning and high computational cost can also affect software fault prediction. Algorithms like Support Vector Machine and Artificial Neural Networks demand high computational power and fine-tuned parameters. Hence, training such models on large datasets is very time-consuming and can also increase processing costs. As a result, it is difficult to apply these models to environments where computing power and resources are limited (Elish & Elish, 2008).

Another major challenge is cross-project prediction in software defect prediction. Different projects use different technologies, development processes, and coding styles. Therefore, the model built for one software project may not be effective for another. Consequently, applying such prediction models across various software projects is challenging due to this limited generalisation (Jorayeva et al., 2022).

4. DISCUSSION

In this study, 15 papers are reviewed and analysed. The results show that different machine learning techniques have been used for software defect prediction based on varying performance levels. Each technique has different advantages and disadvantages based on the nature of datasets and software projects.

According to the reviewed studies, the most used model is the decision tree. It is very easy for developers to understand and evaluate the result because of its tree-like structure. Despite its simplicity, the most common drawback is overfitting, especially when the model is trained on large and complex datasets. As a result, the model becomes less accurate (Koru & Liu, 2005).

The most lightweight and effective technique found is Naive Bayes, which results in little computational demands and provides reasonable performance. It worked well for a small amount of training data. The main drawback is its assumption that all features are statistically independent, which is rarely valid in real-world software datasets, and this can reduce prediction accuracy (Menzies et al., 2007).

ISSN 3136-2710

Support Vector Machine can provide excellent prediction accuracy, specifically when working with high-dimensional datasets. Under certain circumstances, it can be more effective than other traditional machine learning techniques. However, its practical implementation on large datasets is limited because it requires careful parameter tuning and significant computational resources to achieve optimal results (Elish & Elish, 2008).

The technique that emerged as the most well-performing across the reviewed studies is Random Forest. It increases prediction accuracy because it reduces the risk of overfitting by combining multiple decision trees. It is significantly well-suited for real-world software defect prediction scenarios because it can also handle noisy, complicated, and imbalanced datasets (Lessmann et al., 2008).

It is demonstrated that Artificial Neural Networks have a strong ability to identify complex patterns in huge software datasets. They have been proven to be powerful prediction tools because they can model complex relationships between software metrics and defects. They require significant computational power and training datasets to achieve efficiency, and this limits their use in resource-constrained development environments (Ghotra et al., 2015).

The reviewed studies collectively indicate that no single machine learning technique is suitable for every situation of software defect prediction. It depends on various factors, including the quality of datasets, available computational resources, and the complexity of projects. However, Random Forest was shown to be the most reliable and accurate model overall, while Naive Bayes is considered most suitable for situations where data and resources are limited (Lessmann et al., 2008).

5. CONCLUSION

This study analyses the techniques used in software defect prediction by studying 15 research papers published in the past 10 years. This study explored the effectiveness of technology and which elements influence this. It also shows that Random Forest is the most effective among all because it assesses and compares machine learning techniques used in software fault prediction.

The result gives a detailed picture of each technique's performance in various conditions and talks about factors that affect its efficiency. It shows that random forest performs all the examined actions. Even when working with an unbalanced database, it still prevents excessive fitting of data while maintaining excellent prediction accuracy. However, SVM and ANN show required results but are not used widely because of their high processing needs. However, an inadequate fit is their major flaw, but random forest is still used because of its ease of use and clarity.

Major issues restricting the value of the fault prediction model are also studied. Missing data, high calculation cost, and limited inter-project extension are major problems. By addressing these problems through improved data preparation, smart data sampling, and standard learning strategies, we can build a more reliable learning system.

ISSN 3136-2710

REFERENCES

Albattah, W. and Alzahrani, M. (2024) 'Software defect prediction based on machine learning and deep learning techniques: An empirical approach', *AI*, 5(4), pp. 1743–1758. Available at: https://doi.org/10.3390/ai5040086

Bhandari, K., Kumar, K. and Sangal, A.L. (2023) 'Data quality issues in software fault prediction: A systematic literature review', *Artificial Intelligence Review*, 56, pp. 7839–7908. Available at: https://doi.org/10.1007/s10462-022-10371-6

Catal, C. and Diri, B. (2009) 'A systematic review of software fault prediction studies', *Expert Systems with Applications*, 36(4), pp. 7346–7354. Available at: https://doi.org/10.1016/j.eswa.2008.10.027

Elish, K.O. and Elish, M.O. (2008) 'Predicting defect-prone software modules using support vector machines', *Journal of Systems and Software*, 81(5), pp. 649–660. Available at: https://doi.org/10.1016/j.jss.2007.07.040

Ghotra, B., McIntosh, S. and Hassan, A.E. (2015) 'A large-scale study of the impact of feature selection techniques on defect classification models', *Empirical Software Engineering*, 20, pp. 1467–1498.

Hall, T., Beecham, S., Bowes, D., Grey, D. and Counsell, S. (2013) 'Software fault prediction metrics: A systematic literature review', *Information and Software Technology*, 55(8), pp. 1397–1418.

Jorayeva, M., Akbulut, A., Catal, C. and Mishra, A. (2022) 'Machine learning-based software defect prediction for mobile applications: A systematic literature review', *Sensors*, 22(7), Article 2551.

Khalid, A., Badshah, G., Ayub, N., Shiraz, M. and Ghouse, M. (2023) 'Software defect prediction analysis using machine learning techniques', *Sustainability*, 15(6), Article 5517.

Lessmann, S., Baesens, B., Mues, C. and Pietsch, S. (2008) 'Benchmarking classification models for software defect prediction: A proposed framework and novel findings', *IEEE Transactions on Software Engineering*, 34(4), pp. 485–496.

Malhotra, R. (2015) 'A systematic review of machine learning techniques for software fault prediction', *Applied Soft Computing*, 27, pp. 504–518.

Menzies, T., Greenwald, J. and Frank, A. (2007) 'Data mining static code attributes to learn defect predictors', *IEEE Transactions on Software Engineering*, 33(1), pp. 2–13.

Wahono, R.S., Herman, N.S. and Ahmad, S. (2014) 'A systematic literature review of software defect prediction', *Journal of Software*, 1(1), pp. 1–16.