International Journal of Electrical and Computer Engineering (IJECE)
Vol. 16, No. 1, February 2026, pp. 545~554
ISSN: 2088-8708, DOI: 10.11591/ijece.v16i1.pp545-554

A systematic review of software fault prediction techniques: models, classifiers, and data processing approaches

R. Kanesaraj Ramasamy[1], Venushini Rajendran[1], Parameswaran Subramanian[2]
[1]Faculty of Computing and Informatics, Multimedia University, Cyberjaya, Malaysia
[2]School of Business and Management, CHRIST University, Pune, India

Article Info

Article history:

Received Jun 12, 2025
Revised Sep 23, 2025
Accepted Nov 23, 2025

Keywords:

Classification algorithms
Data preprocessing methods
Machine learning techniques
Software fault prediction
Software reliability engineering

ABSTRACT

Software fault prediction (SFP) plays a critical role in improving software reliability by enabling early detection and correction of defects. This paper presents a comprehensive review of 25 recent and significant studies on SFP techniques, focusing on data preprocessing strategies, classification algorithms, and their effectiveness across various datasets. The review categorizes the approaches into traditional statistical models, machine learning methods, deep learning architectures, and hybrid techniques. Notably, wrapper-based feature selection, neural network classifiers, and support vector machines (SVM) are identified as the most effective in achieving high accuracy, particularly when dealing with imbalanced or noisy datasets. The paper also highlights advanced approaches such as variational autoencoders (VAE), Bayesian classifiers, and fuzzy clustering for fault prediction. Comparative analysis is provided to assess performance metrics such as accuracy, F-measure, and area under the curve (AUC). The findings suggest that no single method fits all scenarios, but a combination of appropriate preprocessing and robust classification yields optimal results. This review provides valuable insights for researchers and practitioners aiming to enhance software quality through predictive analytics. Future work should explore ensemble learning and real-time SFP systems for broader applicability.

Corresponding Author:

R. Kanesaraj Ramasamy
Faculty of Computing and Informatics, Multimedia University
Persiaran Multimedia, 63100 Cyberjaya, Selangor, Malaysia
Email: r.kanesaraj@mmu.edu.my

1. INTRODUCTION

Dealing with software faults and failures is an important task in the software quality and reliability topic. Raju *et al.* [1] defined software fault as a defect in software code resulting in software failure during execution. Whereas, Gupta *et al.* [2] mentioned that system failure is when the system is not behaving the way it should be. Failures in cloud systems could cause serious problems such as data damage, according to Lou *et al.* [3]. Software failures in software systems can lead to performance degradation and poor service, which need to be reduced to avoid customer dissatisfaction and increasing costs by Pitakrat *et al.* [4]. It is difficult for experienced programmers to avoid software faults in software projects. Commonly, Chatterjee and Maji [5] stated that the data failure is available during the testing or deployment phase. When a problem occurred, the system log would be the first place people would go to detect the cause of failure in Jauk *et al.* [6]. However, Sun *et al.* [7] stated that the amount of failure data is practically much less than non-failure data in software and resulting in an imbalanced data distribution, which may cause poor performance of software prediction.

2. LITERATURE REVIEW

2.1. Attribute selection, sampling technologies, and ensemble algorithm (ASRA) model

Ding *et al.* [8] propose an ASRA model that has a combination of attribute selection, sampling technologies and ensemble algorithms to solve the imbalanced data sets problems. The ASRA model borrowed the chi-square test of attribute selection. The selection algorithm works like this: first, the candidate's feature subset is generated, then the subsets are evaluated and filtered using a filter model. Then, the synthetic minority over-sampling technique (SMOTE) algorithm over-sampling technique and the resample algorithm under-sampling technique are performed. Lastly, the adaptive boosting (AdaBoost) algorithm is used. The ensemble algorithm function is to turn a weak classifier into a strong one. From the experiment, the ASRA algorithm has a high value of F-measures and area under the curve (AUC), which are more than 0.7 and 0.8, respectively, on all the datasets compared to the original and SRA design.

2.2. Spiral life cycle model-based Bayesian classification (SLMBC)

Dhanajayan and Pillai [9] propose a spiral life cycle model-based Bayesian classification technique where the spiral life cycle is integrated with the Bayesian classification with the help of the robust similarity clustering technique (RSC). This method has 4 phases. The first phase is to pinpoint the objective, functionality, alternatives and constraints of the software product. Then, the alternatives will be evaluated. The third phase is the development and testing phase and lastly is the planning phase for the next iteration. Then, the software reliability model is performed, followed by the Bayesian classification model. To predict the failure, we will look at the biggest posterior probability. After classifying the module, robust similar clustering (RSC) is carried out. Using the minimum distance measure of the similar features, a few clusters are generated. From the experiment, the SLMBC achieved 0.52 percent in the detection of faulty modules which are not faulty, 0.005 in false positive rate (FPR) and 0.02 in the overall error rate. The false negative rate (FNR), FPR and overall error rate are low using SLMBC compared to others.

2.3. Metric based on neural network classifier

Jayanthi and Florence [10] propose an integration of principal component analysis (PCA), a scheme of feature reduction with the application of a neural network-based classification technique. PCA works in such a way by adding second-order moment computation to any random vector's characteristics. The PCA data reconstruction might have errors; therefore, the PCA is improved by integrating maximum-likelihood estimation. Then, neural networks are implemented. The neural network has three layers and the software data will be processed by referring to its weights. At each layer, an input of neurons will be given after the weights have been adjusted following the requirement. To get the final result, all the inputs will be multiplied by their respective weights. The experiment used four datasets, which are KCI, JMI, PC3 and PC4. The accuracy of the proposed approach is as follows: KCI with 86.91%, JM1 with 83.03%, PC3 with 89% and lastly PC4 with 93.64%.

2.4. Grey system theory-based method

Mao [11] proposed a prediction framework based on a grey model. The grey model is combined together with interval prediction of software faults and prediction of fault number based on related factors. The grey theory is used to get the potential law of a dataset through mining. The process is called grey sequence generation. Then, through some operation, the randomness of the grey sequence can be reduced. A new data sequence can be obtained by applying a transformation operation (grey sequence generation) to the grey sequence. The new data is called a transform data sequence. A prediction can be employed when a relative level is reached through the smoothness of the sequence. In grey theory, grey modelling is used to express sequences using approximate differential equations. GM (1,1) is used in this method. Then, by using GM, the fault number can be predicted. To predict the interval of a fault number, proportional band-based and development-band methods are used. The approach is based on the minimum and maximum from the sequence, in addition to a few steps that need to be done. The method is proven to reduce the cost of maintenance and allow the organization to get better ideas on how to handle failures.

2.5. Fuzzy rules and data analysis-based method

Ding *et al.* [12] proposed new online prediction techniques using fuzzy rules and data analysis. This method has three phases: the first one is the online training, the requirement documentation and the running of the system. This is to get the log file corresponding to the sampling time. The second phase is the prediction model building, where a fuzzy rule is applied to get the variables' relationship and the evolutionary trend using the autoregressive integrated moving average (ARIMA) model. The last phase is the failure prediction, where both the values from the ARIMA model are compared with the fuzzy rule. If the difference between the values exceeds, then there would be an error. From the experiment that was conducted on

multiple monitored variables, it shows that the proposed method was able to get 26 TP from 28 numbers of failures injected, which is only 2 FP predicted wrongly.

2.6. Energy-based anomaly detection

Monni and Pezze [13] presented a new approach of energy-based models to predict failures based on the observation of analogies among complex software systems, physical systems, and networks. The feasibility of the approach is evaluated to reveal some preliminary results by measuring the precision of the restricted Boltzmann machine (RBM) used in using models to reveal failure-prone anomalies. The results suggested that revealing collective anomalies of key performance indicators (KPIs) values may predict failures in complex software systems. KPIs are the anomaly detectors found in many different parts of the software systems that are used to collect various metrics. The energy-based approach surmounts limitations of several different approaches, which are signature-based approaches as well as seeded and non-seeded data-driven approaches.

2.7. Deep learning technique-based model called VAE

Sun *et al.* [7] proposed a deep learning technique of a generative model called the variational autoencoder (VAE) method for predicting software faults. The proposed model is used to generate new samples of failure data to overcome the imbalanced data of software faults, in which there is more failure data to indicate the failure module (minority) than non-failure data (majority) to indicate the non-failure module. Thus, the VAE is designed to balance the datasets and to improve the accuracy of the classifier. In short, data processing, as well as VAE and Pass methods, have been used in the experiment, and several metrics have been chosen for the results evaluation. In conclusion, the results reported that the utilization of the VAE method improves the ability to predict failure data while the prediction of non-failure data is being performed.

2.8. Bayesian belief network-based model

A study by Chatterjee and Maji [5] explained a Bayesian-based model, developed to predict the net number of faults during the early development phase of software. First of all, the proposed Bayesian belief network is a directed acyclic graph consisting of nodes, each is associated with a node probability table or node probability table (NPT), which has the values of conditional probability and the expected fault index. Second, a type of fuzzy control system, called an interval type-2 fuzzy logic system, is used to calculate the probability values of the model. Besides, an artificial neural network (ANN) is applied to identify the output from the input of the data from similar or earlier projects. Third, six metrics are used to implement the proposed methodology. Software metrics have qualitative information about the software during its early phase. In sum, the proposed model can predict total faults in software.

2.9. Support vector machine classifier

A study by Raju *et al.* [1] proposed a work that uses a support vector machine (SVM) classifier to classify fault and non-faulty modules. Along with SVM, the authors also implemented feature extraction, or known as attribute selection, to find appropriate features for the classification model. Integrating a feature extraction method with a classifier (SVM) is also called the wrapper approach. Overall, the proposed methodology followed simple steps to predict the software faults: defected dataset ‘s’ as input; 1: Feature extraction method is used in pre-process of the data set; 2: 10-fold cross validation is applied by dividing the data set into training model and testing model; 3: SVM classifier is used for classification. The datasets used as the input to the proposed model are CM1, JM1, KC1, KC2, PC1, and DATATRIEVE. Each data set has a good number of instances indicating software modules and also has several software metrics, which can help to identify faults in the existing software modules. For the experimentation process, the CK-metrics are applied to the datasets KC1 and KC2 since both are related to the object-oriented approach. Then, an experiment is performed on all the data sets to evaluate the parameters, which are accuracy, sensitivity, completeness or specificity, cut-off or precision, and F-measure. In conclusion, the experimental results proved that the proposed SVM-based model as a binary classifier performed best in terms of classification accuracy.

2.10. Machine learning algorithms and techniques

Campos *et al.* [14] presented an analysis of several machine learning techniques and algorithms for supporting online failure prediction (OFP). In the experiments, multiple algorithms and different data processing methods are considered. A comparison is made with SVM, an algorithm used in OFP. In conclusion, the results show that SVMs can predict a single failure mode, but not when considering multi-class failures. For single and multiple failure modes, decision tree (DT), neural network (NN), and Bagging can predict them well.

2.11. Hierarchical online failure prediction approach (HORA)

Pitakrat *et al.* [4] proposed an approach called HORA, used for OFP, which consists of two kinds of architectural models and hierarchical online failure predictors. The objective is to predict the propagation of the failures if the failure of each component can be predicted and the dependencies among them are acknowledged. First of all, based on Bayesian networks, the failure propagation model (FPM) is architected, which comprises the prediction outputs and dependencies among the components obtained from the architectural models. Second, the architectural dependency model (ADM) is a middle model representing the dependencies among components of the architecture. Third, an automatic extraction of the ADM requires monitoring data that comprises system-level resource usage. The result is called the SLAstic model, which includes the relationships and components in the software, information about deployment, and the number of calls of each component. ADM is created by combining the information from the SLAstic model to obtain knowledge about the dependencies of the components. In conclusion, the proposed approach can predict component failures in the system and the probabilities of the propagation of those failures to other components.

2.12. Improving software fault prediction with threshold value

According to Shatnawi [15], threshold recognition and defect prediction are two methods that are used for analyzing statistical software. Those methods are combined in their research to include a new form of static analysis. Shatnawi [15] proposed a new dependent variable by using threshold values. The values of the threshold are used to identify the software systems that need focus in development, testing and maintenance. If they have no faults, these modules are labelled as a medium group. The faulty modules have been defined as strong, whereas non-faulty systems are defined as nothing at all. Afterwards, the latest identity has been used in pentuple algorithms. Five classifiers also evaluate the original variable, as well as the outcomes of the latter are evaluated through testing of statistics. Wilcoxon signed a rank test able to find three types of classifiers as of variables of two. When compared to the three classifiers, they noted that although JRip output diminishes, naive Bayes (NB), and logistic regression (LR) are notably greater in the recommended parameter. Conclusively, their findings indicate that in certain classifiers, the suggested parameter either enhances the performance or just zero impact on the performance of others.

2.13. Deep learning system software fault prediction

According to Qiao *et al.* [16], they developed the concept of a deep learning concept for predicting vulnerabilities in application systems. Their preferred method develops a comprehensive learning approach to estimate the serious flaw. The improvement of the suggested technique on the support vector regression (SVR), fuzzy support vector regression (FSVR), and decision tree regression (DTR) is essential for the collected data. Paled in comparison to such state-of-the-art methods on two well-known models, the method achieved a substantial reduction in the standard error and increased the compounded correlation coefficient.

2.14. Comprehensive model for software fault prediction

According to Singh [17], he has used the five developments from organic fields for testing and training. It is seen that in the case of a project, C4.5 executed well around a mean value of receiver-operator characteristic (ROC) than the proposed eight rule-based classification with the majority of ROC. The ability to deal with the program defect prediction problem-based learner rule also contributed to improved efficiency due to inadequate training data for a given class. Decision table-Naive Bayes hybrid classifier (DTNB) has surpassed several rule-based learners in cross-projects as well and the findings also become close to ROC 69 percent inside the programmed. Therefore, data from various activities relating to the very same area can be used to estimate inter-project failure, just in case of task data set is absent and can work similarly across tasks.

2.15. The software defect prediction concept relies on the AltaRica language

According to Song *et al.* [18], a method using the AltaRica algorithm is proposed in their research for software fault prediction. The proposed method comprises three components: one is AltaRica-based software requirement modeling, the two is line temporal logic (LTL)-based safety limitations, and the three is a model-based fault prediction algorithm. Eventually, they applied this model to the traditional study program for the aviation aid software system. The test results show that this latest design will boost the efficacy and validity of the defect prediction that can reliably characterize the operating characteristics of the aviation aid software system, and effectively recognize dynamic defects such as the state transition dispute, and the irregular feature series.

2.16. Research on software metric selection for software defect prediction

According to Wang and Khoshgoftaar [19], they have made a comparison with three classes of selection which is filter-based subset evaluators, wrapper-based subset selectors and filter-based feature ranking. They then built the model accordingly to resolve software failure predictions. Then, the reliability of its identification is evaluated according to the area under the receiver operating characteristic AUC efficiency metric. They have used data obtained from the four releases of the massive network system, specifically telecommunications. The predictive models are developed using five different classifiers, which are K-nearest neighbors, Bayes, multilayer perceptron, logistic regression and support vector machines. The test results reveal that, according to the AUC efficiency measure, the wrapper-based approach to selection of a subset performed better than the rest. The apps' ranking did worse. Furthermore, when compared to the five learners in our sample, logistic regression (LR) represents the highest output, then, multilayer perceptron (MLP).

2.17. Load-capacity model

The traditional load capacity model has two states, which are the normal state and the faulty state. Ji *et al.* [20] proposed a third state that is called state-congestion to meet the characteristics of communication networks. They improved the traditional model and made it able to be more reliable in terms of accuracy and precision of fault prediction and prediction effect. After comparison with the models, they found that the average accuracy of the proposed fault prediction model is around 66.61% and improved by 9.35% compared with non-linear load capacity model in Wang *et al.* [21]. The proposed model has a high accuracy rate but it is sensitive to the propagation prediction in the network's key node faults.

2.18. Artificial neural network and queuing theory

Using artificial neural networks (ANN) and queuing theory, Tripathi and Saraswat [22] were able to increase the reliability of software products to estimate failure rate, detection, correction, and allocation. The queuing theory model is mainly about simulating the customer arrival pattern in a queue that has been correlated with the error detection pattern in the process of software development. The paper reveals that less number of testers are required in the early phases of software development life cycle (SDLC), but the number increases as the software development phases increase. Team leaders of software companies can use this paper's results to predict the number of testers they should hire from the initial stage to the middle stage and at the end stage of software development. In the middle stage, the maximum number of software testers is needed, but it decreases in the initial phase and the last phase of the software development life cycle (SDLC).

2.19. Inheritance metrics and artificial neural network

Aziz *et al.* [23] selected Chidamber and Kemerer metrics (CK) to evaluate inheritance effects on Software fault prediction (SFP). They split the datasets into two sets, the first one is CK with inheritance and the other dataset is CK without inheritance for comparison. To build this model, they used an ANN. The results have shown that inheritance shows an acceptable contribution in SFP and it is safe, but high inheritance is not because it can lead to software faults. They suggested keeping the inheritance metrics minimum; the testing community can safely use inheritance metrics to predict faults, but high inheritance is not recommended because it can cause faults.

2.20. Residual errors: J-M and G-M

The hybrid model Jabeen *et al.* [24] proposes a combination of J-M and G-M based on residual errors; the model selects the most suitable predicted value from both models. G-M gives high prediction accuracy with small samples of data, but with low predicting accuracy with random fluctuating data. J-M is more suitable for forecasting vibrating and changing stochastic data sequences but it needs a large amount of data. The hybrid model combines the best-predicted values of the models and gives better forecasting results for failure data sequences. The model has good performance accuracy and applicability in Software fault prediction.

2.21. Wrapper-based selection method by particle swarm optimization-multi-Gaussian approach

Using datasets that consist of noisy and irrelevant records may result in unnecessary waste of resources and could lead to failures if using classification without selection methods. There are mainly three types for classification: filter-based feature selection, embedded feature selection and wrapper feature selection. Banga and Bansal [25] chose the wrapper-based selection method because it has the best accuracy among these approaches. The method is a hybrid of algorithms used to improve the accuracy the reliability of the software estimation using feature selection by particle swarm optimization (PSO)-multi-Gaussian approach (MGA) to reduce noise and select relevant attributes. Mean weighted least squares twin SVM (MW-LSTSVM) results show that the classification algorithm has reached the highest accuracy among all other classifiers with 91.3%, efficiency increased by 9.8% than the other existing methods in classifying defective and non-defective approaches.

2.22. Machine learning implementation

From the research, Wójcicki and Dąbrowski [26] found that they can automatically predict the possibly faulty fragment in the source code they coded and which allows developers to focus more on development, but not consume lots of time in failure finding. The techniques that they use improve the failure prediction by structural programming to restrict the control flow to a hierarchical structure. The next improvement is that the object-oriented paradigm will be the main programming concept. As a result, the research proves that fault-prediction methods can be applied successfully to Python, Java, and C/C++ programs, and get the result of 0.64 recall rate and 0.23 false positive rate. This research also supports applying the special logarithmic filter, since the fault predictor implemented without using this filter achieved a 0.328 of recall rate with a 0.108 of false positive rate. In conclusion, machine learning can be used for creating a fault predictor for Python, Java and C/C++ projects.

2.23. BR technique

In this research, Mahajan *et al.* [27] found Bayesian regularization (BR) technique helps with software faults predicted. The main function of the BR technique is to eliminate the math errors and also determine the suitable combination to produce an efficient network. The accuracy of the BR algorithm is compared with the Levenberg-Marquardt (LM) algorithm and the back propagation algorithm (BPA) for searching software faults. They also find accuracy with NN classifiers. For the result, the BR algorithm provides 92.44% accuracy, which is the highest accuracy compared to another algorithm that was tested in this experiment. In conclusion, the BR algorithm is the most reliable algorithm with the comparison to other algorithms.

2.24. Iterated feature selection algorithm and layer-recurrent neural network

From the research, Turabieh *et al.* [28] used binary genetic algorithm (BGA), binary particle swarm optimization (BPSO), and binary ant colony optimization (BACO) as wrapper FS algorithms. The result is compared with naïve Bayes (NB), ANN, logistic regression (LR), k-nearest neighbors (k-NN) and C4.5 decision trees. The proposed algorithm generates the testing dataset. They test the performance of the algorithms in amplifying the software fault and failure prediction system. They applied the proposed algorithms that they proposed with the layer-recurrent neural network (L-RNN) classifier. For the result, they found that the performance of L-RNN depends on the input data characteristics. Finding the important metric will enhance the training process. In conclusion, the proposed algorithm can choose the essential software metric by using distinct feature selection (FS) algorithms.

2.25. Semi-supervised deep fuzzy c-mean clustering method

From this review paper, Arshad *et al.* [29] deal with the supervised and unsupervised data to utilize the fuzzy information from labeled to unlabeled data to sustain the building of the current model pattern. They utilize deep fuzzy c-mean clustering (DFCM) clustering to replace human logic. To prove the method, they present a model for software fault prediction. They show the capability of their method with other methods and all results are performed by averaging 100 runs. They observe that the performance of class mass normalization (CMN) is worse than non-negative sparse graph based labeled propagation, NTC (NB), and DFCM. They also found that the performance of FTF is poor because FTF applies supervised data. It concludes that semi-supervised data for training models improves capability. In summary, the method can build a fine prediction system by generating good features and removing excessive features to decrease the unused data for classification.

3. ANALYSIS

This section will analyze all the available software prediction methods that have been researched. This includes the analysis of data processing techniques, classification techniques, metrics used, as well as how this will improve the accuracy of a software prediction. Table 1 shows a comparison table between different prediction methods with the corresponding advantages and disadvantages of each method.

3.1. Datasets

Datasets play a crucial role in the accuracy of the classifier. If the dataset is not chosen carefully, the classifier can become biased toward the non-fault-prone module, which leads to a decrease in the software prediction's classification accuracy. There are a lot of issues regarding the dataset, such as the imbalance of datasets or the selection of features. Below is the analysis for this particular problem.

Table 1. Comparison of software prediction methods

No	Prediction Technique	Advantage	Disadvantage
1.	Load-capacity model	- Enhancing the load capacity model. - Representative model. - Improvement in the traditional model. - High accuracy rate.	- Focused on large-scale apps. - Communication networks based.
2.	Inheritance metrics and ANN	- Uses CK metrics. - Uses ANN.	N/A
3.	Residual errors: JM and GM	- Combination of GM and JM models. - Selects the suitable predicted value from both models.	N/A
4.	ANN and queuing theory	- Estimate failure rate, detection, correction and allocation. - Simulate a queue.	N/A
5.	Imbalanced data processing model	- Lower limit of learning accuracy is not needed - Remove redundant attribute - Create a balanced dataset.	- Oversampling will increase training time. - Under-sampling can cause the loss of important information.
6.	Spiral life cycle model-based Bayesian Classification	- Can handle an uncertain dataset. - High accuracy, stability and consistency. - Computation time reduced. - The failure ratio is low.	- Can only detect 81% of faulty modules using the classification and clustering-based algorithm.
7.	Metric based on a neural network classifier	- Time and storage space can be reduced. - Removal of multicollinearity	N/A
8.	Grey system theory-based prediction	- Helps to reveal the fluctuating range of the fault.	- Handle small and uncertain datasets. - Limited to its intrinsic limitations.
9.	Online failure prediction based on fuzzy rule and data analysis	- No failure pattern or expected value is needed. - Can deal with a variable that is discrete, continuous and linguistic. - Avoid fake regression	- A fuzzy rule is not always accurate.
10	Energy-based anomaly detection	- Overcome limitations of signature-based as well as seeded and non-seeded data-driven approaches.	- The experimental results are far from being conclusive.
11	Deep learning technique-based model called VAE	- Improves the ability to predict failure data while the prediction of non-failure data is being performed.	N/A
12	Bayesian belief network-based model	- Assists developers in achieving a state of total software faults that has been targeted.	N/A
13	SVM classifier	- Improves classification accuracy.	N/A
14	Machine learning algorithms and techniques	- DT, NN, and Bagging can predict single and multiple failure classes.	- SVMs cannot predict multiple failures and their performance drops when predicting individual failures.
15	HORA	- Has higher modularity by reusing different techniques to predict failure among the components of the system.	- Can be used if monitoring of data is constantly collected.
16.	Improving software fault prediction with threshold values	- The result will help know the effect of the threshold for the technique of pre-processing to enhance the software defect prediction	- Focus only on CK metrics - There are several methods for testing software and certain metrics may have different meanings of variants. For example, the WMC metric has two main concepts, and there are several variants to the LCOM metric. But in this study, the metrics data are based on original concepts for possible comparisons.
17	Deep learning-based software defect prediction	- The count of fault prediction is very effective	- Various datasets that used different metrics could affect the performance of fault prediction - The approach was only evaluated based on two open datasets, so the outcome may not be the same for other commercially available software.
18	The software fault prediction model based on the AltaRica language	- Apt for state transition, data interaction, able to process complex systems precisely and use traversal search method to determine the software violation	NA
19	A study on software metric selection for software fault prediction	- LR performed the best	NA
20	Comprehensive model for software fault prediction	- Many rule-based learners have been used for comparison to get an appropriate result	N/A
21	Wrapper-based selection method by PSO-MGA	- Highest accuracy rate among other types of classification methods.	- Should reduce noise and remove irrelevant data first. - Could be improved by using AI.
22	ML method for software fault prediction	- Provide a satisfactory result in predicting faults	- The method is still in the experimental stage and future research is needed.
23	Bayesian regularization (BR) technique	- Fault prediction during the design phase - Best algorithm to apply compared to others	N/A
24	Iterated feature selection algorithm and L-RNN	- Able to select the most important software metrics - Able to receive a good classification rate	N/A
25	Semi-supervised DFCM method	- Multiple clusters can be amalgamated - Incorporate labeled data and unlabeled data - Removing excessive features	N/A

In the ASRA model, the chi-square method is used to discard or ignore irrelevant and redundant features. After filtering the subset, both over-sampling and under-sampling are used, but the disadvantage of this technique is that over-sampling could increase the training time and under-sampling could lead to loss of critical information by Zhou *et al.* [8]. Even though under-sampling can produce faster training processes, it can cause lower performance in Campos *et al.* [14].

Another method is the wrapper-based selection method, where the optimal parameter can be obtained by learning from the training of the datasets. The advantages are that it is more accurate compared to other feature selection methods, such as embedded feature selection and filter-based selection by Banga and Bansal [25]. In the paper by Turabieh *et al.* [28], wrapper-based selection is further improved by performing iteration. The advantages of this method are that it stops iterating if the optimal value is reached and allows you to find the most valuable metrics.

Other than that, in the metric-based neural network classifier proposed by Jayanthi and Florence [10], principal component analysis (PCA) is used, where they lower the proportion of the dataset as well as the dimensionality while reducing the information loss. The PCA is then further improved by adding maximum-likelihood estimation to reduce the reconstruction error. The advantages of this method are that it allows the dataset to be distributed equally without affecting other undistributed data. Based on the experiment, all the different datasets have an accuracy of more than 80%, which shows that PCA has an important role in the prediction accuracy. Lastly, in the paper from Mao [11], grey sequence generation is used to handle datasets, but the drawback is that it can only handle small data sizes compared to other methods that handle big datasets.

3.2. Classification

The second aspect is the classification process in predicting software faults. There are several classifiers used for faulty and non-faulty software components classification. One of the classifiers is SVM. In the study by Raju *et al.* [1], this classifier is used as a binary classifier for classifying several datasets input and experimental results showed that it has the accuracy above 97% for CM1 data set, 99.54% for JM1 data set, 96.58% for KC1 data set, 99.95% for KC2 data set, 99.57 for PC1 data set, and 93.89 for the DataTrieve data set. Besides, according to Banga and Bansal [25], a variation of the SVM classification algorithm known as least squares twin SVM (MW-LSTSVM) also has the highest accuracy, with 91.3%, than other classifiers like k-NN and LSTSVM, and the efficiency of the proposed approach (PSO-MGA) is 9.8% higher than other techniques in fault and non-faulty modules classification.

Although SVMs can predict one failure class, however, it cannot be used when considering multiple failure modes by Campos *et al.* [14]. The authors stated that when considering both one and multiple failures, DT, NN, and Bagging algorithms can predict them well. Next, other typical classifiers by Sun *et al.* [7], where five typical classifiers are used to predict software failure, including SVM, which are RF, DT, NB, and LR. In Dhanajayan and Pillai [9], Bayesian classification is used in the proposed technique called SLMBC to group or classify faulty and non-faulty modules using a probability distribution. As a result, FNR, FPR, and overall error rate can be greatly reduced using SLMBC than other techniques. After that, in Jayanthi and Florence [10] and Mahajan *et al.* [27], NN classifier is used while Turabieh *et al.* [28] proposed a classification technique called L-RNN used in the proposed algorithm called iterated feature selection algorithm and as a consequence, it can gain well rate of classification based on AUC results, outperforming NB, ANN, k-NN, and C4.5.

4. RECOMMENDATION

Dealing with software faults is a very important task when dealing with software reliability and quality assurance. In our review paper, we reviewed research papers and models that focus on Fault and Failure Prediction Techniques. our analysis is based on data processing techniques, classification techniques and metrics to come up with a recommendation on which SFP techniques are more convenient for some scenarios. Some of the traditional models in SFP can be improved and applied to software systems to ensure a better quality in predicting faults and failures.

All of the reviewed SFP techniques have their own advantages and limitations under different scenarios. Thus, the team has a few recommendations on how to improve the reliability of software products using software fault and failure prediction techniques. In terms of the used dataset, some methods cannot discard or ignore irrelevant and redundant data, and from our analysis, we suggest using the wrapper-based selection method because it is the most accurate selection method among the other major selection methods. Also, it can be improved by using iterations.

In terms of the classification aspect, we suggest using the neural network classifier proposed by Jayanthi and Florence [10]. The advantage of this method is that it allows the dataset to be distributed equally

without affecting other undistributed data. Based on the experiment, all the different datasets have an accuracy of more than 80%. In Banga and Bansal [25], a variation of the SVM classification algorithm known as MW-LSTSVM also has the highest accuracy with 91.3%, than other classifiers like k-NN and LSTSVM, and the efficiency of the proposed approach (PSO-MGA) is increased by 9.8% than other techniques. SVMs can predict one failure mode, while for single and multiple failure modes, DT and Bagging can predict them well.

5. CONCLUSION

In conclusion, we found that it has plenty of ways and techniques to perform software fault and failure prediction. Based on the total of 25 reviews in this research paper, we centralized our analysis and categorized our analysis into data processing techniques and classification techniques. Data processing techniques mainly control the accuracy of the classifier, and the classification technique is the way of categorizing input datasets. Through the analysis, we have shown several comparisons on data processing techniques as well as classification techniques. For the analysis result, we finalized our analysis and recommended several techniques, which are methods and NN classifier for data processing technique, and SVM for classification technique wrapper-based selection. These techniques have the best accuracy and also efficiency for fault and failure prediction based on our research. For future work, we may look at the research and discover more techniques as well as write a research paper that is more in-depth into the process of fault and failure prediction.

REFERENCES

[1] K. S. Raju, M. R. Murty, M. V. Rao, and S. C. Satapathy, "Support vector machine with k-fold cross validation model for software fault prediction," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 20, pp. 321–334, 2018.

[2] S. Gupta, A. Mishra, and M. Chawla, "Analysis and recommendation of common fault and failure in software development systems," *International Conference on Signal Processing, Communication, Power and Embedded System, SCOPES 2016 - Proceedings*, pp. 1730–1734, 2017, doi: 10.1109/SCOPES.2016.7955739.

[3] C. Lou, P. Huang, and S. Smith, "Understanding, detecting and localizing partial failures in large system software," in *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020*, 2020, pp. 559–574.

[4] T. Pitakrat, D. Okanović, A. van Hoorn, and L. Grunske, "HORA: Architecture-aware online failure prediction," *Journal of Systems and Software*, vol. 137, pp. 669–685, 2018, doi: 10.1016/j.jss.2017.02.041.

[5] S. Chatterjee and B. Maji, "A Bayesian belief network based model for predicting software faults in early phase of software development process," *Applied Intelligence*, vol. 48, no. 8, pp. 2214–2228, 2018, doi: 10.1007/s10489-017-1078-x.

[6] D. Jauk, D. Yang, and M. Schulz, "Predicting faults in high performance computing systems: An in-depth survey of the state-of-the-practice," *SC19: International Conference for High Performance Computing, Networking, Storage and Analysis*, Denver, CO, USA, 2019, pp. 1–13, doi: 10.1145/3295500.3356185.

[7] Y. Sun, L. Xu, Y. Li, L. Guo, Z. Ma, and Y. Wang, "Utilizing deep architecture networks of VAE in software fault prediction," in *Proceedings - 16th IEEE International Symposium on Parallel and Distributed Processing with Applications, 17th IEEE International Conference on Ubiquitous Computing and Communications, 8th IEEE International Conference on Big Data and Cloud Computing, 11th*, 2018, pp. 870–877, doi: 10.1109/BDCloud.2018.00129.

[8] L. Zhou, R. Li, S. Zhang, and H. Wang, "Imbalanced data processing model for software defect prediction," *Wireless Personal Communications*, vol. 102, no. 2, pp. 937–950, 2018, doi: 10.1007/s11277-017-5117-z.

[9] C. G. R. Dhanajayan and S. A. Pillai, "SLMBC: spiral life cycle model-based Bayesian classification technique for efficient software fault prediction and classification," *Soft Computing*, vol. 21, no. 2, pp. 403–415, 2017, doi: 10.1007/s00500-016-2316-6.

[10] R. Jayanthi and L. Florence, "Software defect prediction techniques using metrics based on neural network classifier," *Cluster Computing*, vol. 22, no. s1, pp. 77–88, 2019, doi: 10.1007/s10586-018-1730-1.

[11] C. Mao, "Software faults prediction based on grey system theory," *ACM SIGSOFT Software Engineering Notes*, vol. 34, no. 2, pp. 1–6, 2009, doi: 10.1145/1507195.1507206.

[12] Z. Ding, Y. Zhou, G. Pu, and M. Zhou, "Online failure prediction for railway transportation systems based on fuzzy rules and data analysis," *IEEE Transactions on Reliability*, vol. 67, no. 3, pp. 1143–1158, 2018, doi: 10.1109/TR.2018.2828113.

[13] C. Monni and M. Pezze, "Energy-based anomaly detection a new perspective for predicting software failures," in *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results, ICSE-NIER 2019*, 2019, pp. 69–72, doi: 10.1109/ICSE-NIER.2019.00026.

[14] J. R. Campos, M. Vieira, and E. Costa, "Exploratory study of machine learning techniques for supporting failure prediction," in *Proceedings - 2018 14th European Dependable Computing Conference, EDCC 2018*, 2018, pp. 9–16, doi: 10.1109/EDCC.2018.00014.

[15] R. Shatnawi, "Improving software fault prediction with threshold values," in *2018 26th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2018*, 2018, pp. 286–290, doi: 10.23919/SOFTCOM.2018.8555818.

[16] L. Qiao, X. Li, Q. Umer, and P. Guo, "Deep learning based software defect prediction," *Neurocomputing*, vol. 385, pp. 100–110, 2020, doi: 10.1016/j.neucom.2019.11.067.

[17] P. Singh, "Comprehensive model for software fault prediction," in *Proceedings of the International Conference on Inventive Computing and Informatics, ICICI 2017*, 2018, pp. 1103–1108, doi: 10.1109/ICICI.2017.8365311.

[18] J. Song, B. Chen, X. Li, Y. Yang, C. Liu, and H. Li, "The software fault prediction model based on the AltaRica language," in *Proceedings of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2019*, 2019, pp. 2549–2552, doi: 10.1109/ITNEC.2019.8729235.

[19] H. Wang and T. M. Khoshgoftaar, "A study on software metric selection for software fault prediction," in *Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019*, 2019, pp. 1045–1050, doi: 10.1109/ICMLA.2019.00176.

[20] X. Ji, X. Liang, Y. Zhou, Y. Huo, X. Shi, and Y. Yang, "A fault prediction method based on load-capacity model in the communication network," in *2019 20th Asia-Pacific Network Operations and Management Symposium: Management in a Cyber-Physical World, APNOMS 2019*, 2019, pp. 18–21, doi: 10.23919/APNOMS.2019.8893104.
[21] L. Wang, Y. Fu, M. Z. Q. Chen, and X. Yang, "Controllability robustness for scale-free networks based on nonlinear load-capacity," *Neurocomputing*, vol. 251, pp. 99–105, 2017, doi: 10.1016/j.neucom.2017.04.011.
[22] R. Tripathi and M. Saraswat, "Assessment & prediction of software reliability: ANN approach," in *SSRN Electronic Journal*, 2020, pp. 1–4, doi: 10.2139/ssrn.3565873.
[23] S. R. Aziz, T. Khan, and A. Nadeem, "Experimental validation of inheritance metrics' impact on software fault prediction," *IEEE Access*, vol. 7, pp. 85262–85275, 2019, doi: 10.1109/ACCESS.2019.2924040.
[24] G. Jabeen, X. Yang, L. Ping, S. Rahim, G. Sahar, and A. A. Shah, "Hybrid software reliability prediction model based on residual errors," in *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, 2017, pp. 479–482, doi: 10.1109/ICSESS.2017.8342959.
[25] M. Banga and A. Bansal, "Proposed software faults detection using hybrid approach," *Security and Privacy*, vol. 6, no. 4, pp. 1–14, Dec. 2023, doi: 10.1002/spy2.103.
[26] B. Wójcicki and R. Dąbrowski, "Applying machine learning to software fault prediction," *E-Informatica Software Engineering Journal*, vol. 12, no. 1, pp. 199–216, 2018, doi: 10.5277/e-Inf180108.
[27] R. Mahajan, S. K. Gupta, and R. K. Bedi, "Design of software fault prediction model using BR technique," *Procedia Computer Science*, vol. 46, pp. 849–858, 2015, doi: 10.1016/j.procs.2015.02.154.
[28] H. Turabieh, M. Mafarja, and X. Li, "Iterated feature selection algorithms with layered recurrent neural network for software fault prediction," *Expert Systems with Applications*, vol. 122, pp. 27–42, 2019, doi: 10.1016/j.eswa.2018.12.033.
[29] A. Arshad, S. Riaz, L. Jiao, and A. Murthy, "Semi-supervised deep fuzzy c-mean clustering for software fault prediction," *IEEE Access*, vol. 6, pp. 25675–25685, 2018, doi: 10.1109/ACCESS.2018.2835304.

BIOGRAPHIES OF AUTHORS

R. Kanesaraj Ramasamy is an associate professor at the Faculty of Computing and Informatics and Deputy Director of the Touchpoint Management Office, Multimedia University. With over 15 years of experience across academia, industry, and national initiatives, he specializes in AI, IoT, and predictive analytics, bridging research with real-world applications. A Ph.D. holder in IT, professional technologist (MBOT), and Senior IEEE Member, he has led research projects exceeding RM1.05 million and CSR initiatives worth RM137,000, producing award-winning innovations such as PulseWise and Smart Toilet. He has published widely in SCOPUS Q1/Q2 journals, holds five copyrights and one patent, and has advanced MMU's predictive analytics tNPS system across 38 departments. His contributions extend to Malaysia's National IoT Policy, while he actively supervises postgraduate research, delivers professional training to organizations such as TM, MCMC, and POS Malaysia, and champions ESG- and CSR-driven innovations in healthcare and sustainability. He can be contacted at email: r.kanesaraj@mmu.edu.my.

Venushini Rajendran is a lecturer at the Faculty of Computing and Informatics, Multimedia University, with a master of science specializing in web services and IoT. Formerly a research scholar with Telekom Malaysia R&D, she contributed to projects in web, mobile, and IoT, earning recognition as a Malaysian Young Digital Hero. Passionate about the intersection of technology and human behavior, she also holds multiple Microsoft certifications, showcasing her expertise in modern digital tools and positioning her as a promising talent in digital technology and innovation. She can be contacted at email: venushini@mmu.edu.my.

Parameswaran Subramanian is an educational administrator with over 22 years of experience in collegiate and university education. He holds a doctorate and specializes in taxation, supported by a master degree in accounting, finance, and human resource management. He has extensive experience handling accreditation processes for the Malaysian Qualification Agency (MQA), UK universities (QAA), and Singapore (SQC). Dr. Parameswaran has managed government-funded projects in taxation and blockchain solutions for business. A certified trainer and advocate for outcome-based education (OBE), Dr. Parameswaran has been widely recognized in his field. He holds seven patents and has published 27 research papers in Scopus, ISI, and peer-reviewed journals. He can be contacted at email: parameswaran.s@christuniversity.in.