Comparative Analysis of Software Defect Prediction Techniques

Mehreen Sirshar
faculty of software engineering
fatima jinnah women university
Rawalpindi, Pakistan
mehreensirshar@fjwu.edu.pk

Hira Mir
dept. of software engineering
fatima jinnah women university
Rawalpinidi, Pakistan
hiramir0987@gmail.com

Khadija Amir
dept. of software engineering
fatima jinnah women university
Rawalpindi, Pakistan
amirkhadija09@gmail.com

Laraib Zainab
dept.of software engineering
fatima jinnah women university
Rawalpindi, Pakistan
zainablaraib1@gmail.com

***Abstract*—Accurate prediction of defects in software components plays a vital role in administrating the quality of the quality and efficiency of the system to be developed. So we have written a systematic literature review in order to evaluate the four main defect prediction techniques. Defect prediction paves way for the testers to find bugs and modify them in order to achieve input to output conformance. In this paper we have discussed the open issues in predicting software defects and have provided with a detailed analyzation of different methods including Machine Learning, Integrated Approach, Cross-Project and Deep Forest algorithm in order to prevent these flaws. However, it is almost impossible to rule which method is better than the other so every technique can be analyzed separately and the best technique according to the problem at hand can be used or can be altered to create hybrid technique suitable for the cause.**

***Index Terms*—Software Defect Prediction, Machine Learning Approach, Integrated Approach, Deep Forest**

I. Introduction

Software quality assurance (SQA) is the process that includes tracking and managing the process of development life cycle to ensure the quality of software by reducing its cost. It may include structured code audits, code walkthroughs, software testing, and prediction of software faults. Software Defect Prediction (SDA) is the highly cost-effective and valuable operation in the field of software engineering. Such prediction approaches would concentrate on directing software testing activities by estimating potential defects in the code prior to release of a latest product before testing begins. Software practitioners see it as a critical step depending on which software quality is being produced. Significant resources were expended by both investors and companies on fixing the problems instigated by the defects contained in the software product. Ensurance of testing and debugging is done by the software prediction in a fast-track mode which is done by providing advanced data dependent on the errors that a new software would probably encounter. To find software components that are unstable as soon as possible would help specialists to concentrate and direct the efforts towards solving potential problems that may occur in a software system that is yet to be developed.

Software defect prediction is a method of creating machine learning classifiers to predict faulty code snippets, using historical information in computer repositories such as complexity of code and change in records to model defects of software metrics. Software's complex source code tends to produce software errors that may result in software failure. In the beginning of development process, when the designers fail to fix an issue in the software results lead to increase in complexity of the software along with the exponentially increase in the monetary terms of projects, predicted results can help developers to detect and repair possible errors, thus improving performance and stability of the software. In the project management, the metrics used early in the life cycle of software development helps in decreasing a software product's defect density; specifically, these metrics can be used to assess if additional performance control is required during process of development. Today, the prior defect prediction methods have been substituted by many data miners. Prediction of software defect has become one of the research avenues in the field of data mining to help developers and testers in identify timely defects of software.

A. Software Defect Dataset

A software defect prediction model understands the software defect data with the application system details (software metrics) being combined with the defect value to make the predictions. An essential requirement for successful estimation of software faults is the availability of reliable information on quality defects. In SDP, software defect database consists primarily of three elements: collection of software metrics, defect details such as defects-per-component and meta information. The software defect prediction is basically divided into three main components which are:

1) Project's defect information: The defect information explains how defect in software modules are identified? what is the extent of the defect etc.? To predict the software defect, three kinds of dataset repositories are accessible namely private repository, partially public/freeware or public repository. The defect data is collected during the software project's specifications, design, construction and test phases and registered in the database related to each element of the software.

- *Private Repository*: No defect database or source code is available in this form of repository. The organizational companies use and timely update this dataset. It may not be feasible to replicate the study relying on these datasets.
- *Partially public/freeware Repository*: The defect information or source code is available in this form of repository. Typically, the values of metric are not available. Users should be able to evaluate the standard values from source code and scale these values directly to the existing fault knowledge. The operator must therefore evaluate metric values and plot these values on the obtainable defect information. This kind of process often demand extra attention because it is a critical job to measure metric values and map their fault data.
- *Public Repository*: Both the metric values and defect information are available publicly is this from of repository. The research performed from these collections using data sets can be replicated.

2) Software Metrics: Developers want to predict the performance of the software objects for the expansion of an efficient and reliable SQA system. The software metrics are devised for this purpose. It is possible to quantitatively assess a software project and determine its performance using these metrics. These metrics are linked to the project's operational properties which includes inheritance, cohesion, coupling etc. Each software metric is linked to some of the software project's operational properties such as inheritance, coupling and cohesion etc. that are utilized to signify an internal performance features i.e, consistency, checking, or defect-proneness.

Software metrics are divided into two main groups: product metrics and process metric. Both groups are not mutually segregated and sometimes behave as both process and product metrics. Figure shows the taxonomy of software metrics.

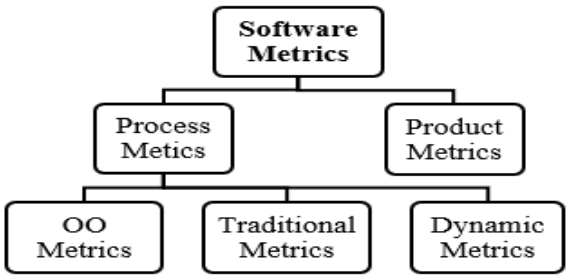

Fig. 1. Taxonomy of Software Metrics

a. Product Metrics: The type of metric that is computed with the help of different features of the product finally developed. These metrics are basically investigating whether the software product supports certain principles such as "ISO-9126". The Product measures is usually be categorized as object-oriented, traditional and dynamic metrics.
   - *Traditional metric*: Metrics that are developed earlier in the field of software engineering can be called traditional metrics such as source line of code, defects per SLOC, function points, kilo-SLOC etc.
   - *OO metrics*: The type of metric that is basically parameters determined through OO methodology from the software developed. In order to recognize the project's structural properties, several OO metrics suites have been suggested.
   - *Dynamic metrics*: The type of metric which focuses on a running program's collected features. Throughout the execution process These measurements disclose component behavior and used to evaluate the belongings of running program, modules and methods.

b. Process Metrics:The set of metrics that relies on characteristics gathered throughout the life cycle of software development. The strategic decision can be easily taken using these metrics. They aim to provide a collection of system steps that will improve the process of technology in the long term. We evaluate the feasibility of a process via extracting the measurements depend on the outcomes i.e, schedule conformance, bug-fix change modules, effort required for each activity etc.

3) Project Meta Information: Meta project information comprises of data on different software system characteristics. It consists of a variety of information includes domain of the developing software, amount of software revision, etc. and accurate evidence based on the reliable defect dataset used to create the defect predictive method. The meta information project attributes are illustrates in the figure.

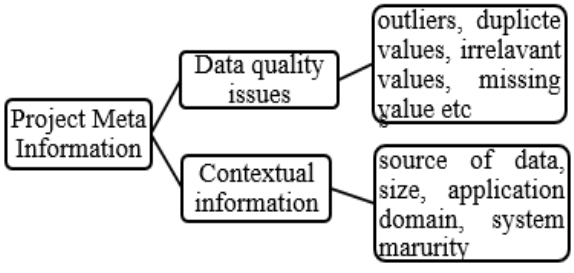

Fig. 2. Attributes of Meta Information

II. LITERATURE REVIEW

From the perspective of system testing, Shantanu et al. [1] tends to illustrate the schema of how to make accurate predic-

tion of the software. In this paper, by considering parameters from system testing matrices and a special parameter named "Component Dependency Score", he proposed that machine learning based approach with the main goal of finding probable areas of faults. The prediction model has been tested with data from only one version and the findings are very positive. By discovering more appropriate features, the model's reliability could be enhanced. As accurate prediction of such faults contributes to improve the quality of the continuously evolving software. The result of the test was an approximately 78 percent of the determination coefficient together with the acceptable absolute mean error.

Santosh S et al. [2] provides an overview of how to predict faults in the software system. The various aspects of the software fault prediction are investigated and discussed. The main purpose of this survey is to deliver the basic concept of different elements involved in the method of fault prediction and uncovered the problems associated with it. This survey undergoes classified into three main groups namely, issues of data quality, software metrics and processes of fault prediction. From this research, it is discovered that object-oriented metrics and process metrics are the one among which the maximum work is based. The results explored that the quality impacts on fault prediction techniques differs with respect to the use of multiple datasets.

Current software defect prediction approaches emphasis primarily on developing static code metrics that are fed into ML classifiers to predict code defect probabilities. Though, these metrics do not comprise of program logical structure and syntax structure. Guisheng Fan et al. [3] proposed a new framework for predicting defects that is "attention-based recurrent neural network (DP-ARNN)". The program logical structure and syntax structure is automatically discovered by DP-ARNN which can be more helpful for predicting accurate results of the model. The attention mechanism is used to find the critical features that contributes in enhancing the performance of the model. They performed an experiment using 7 open source projects which indicates that DR-ARNN progresses such as 14% of the F1 measure and 7% of the AUC.

Xian Zhang et al. [4] proposed a new metric called cross-entropy that contributes in improving the performance of defect prediction. Cross-entropy is a new code metric for the tasks of defect prediction and introduces a model for this process called Defect Learner. Cross-entropy is compared with twenty commonly used metrics on twelve open-source projects to estimate the discrimination of defect proneness. The experiential results reveal that the metric of cross-entropy is more prejudiced than 50 percent of the traditional metrics. The author used combination of both the traditional metrics with cross entropy for enhancing the defect prediction accuracy. By using this, the performance of F1-score predictive models is increased by an average of 2.8%. These results suggest that cross-entropy is effective in predicting code defects and in addition to the existing metric suits.

To improve the software performance, J. Li et al. [5] concentrate on predicting possible code defects in software implementation, thus reducing the software maintenance workload. They proposed a new framework known as "Defect Prediction via Convolutional Neural Network" (DP-CNN) to generate successful features. DP-CNN uses CNN to generate automated source code features with retained semantic and structural data. They also used word embedding and integrate the CNN-learned features with conventional handcrafted features to further increase performance of prediction of defects. By using seven open-source projects, results showed that DP-CNN improves the performance of defect prediction in terms of F-measure by 12% to 16% respectively.

The effective method for fault prediction is classification as the performance of algorithm is directly depending upon the dataset quality. W. Liu et al. [6] proposed a "Two stage data processing approach" that includes selection of features and reduction of unnecessary instances. Model consist of several steps that include feature ranking that comprises of relevance analysis, threshold-based cluster, feature selection, random sampling to differentiate between non-faulty and faulty instances and lastly the trained dataset. The model was carried out on the datasets obtained from NASA that showed cost provoked preprocessing.

Many traditional defect methodologies were used to identify change in software source code that causes defects at initial development stage by using training data. Due to limitation of training data at initial stage Yasutaka Kamei et al. [7] proposed a Cross project model that work with JIT (Just-in-time) model. The main purpose of the model was that it learns from the history of other related projects. The three main approaches of this model were select models that are related to testing projects, cartel many correlated projects to produce big training data and cartel models of many other related projects to produce a novel model.

Defect prediction at early stage may result in reduction of testing phase cost, saves delivery time of software, increase the reliability and quality of the software. Pradeep Kumar Singh et al. [8] suggested that identifying most prominent defects in software while development process removes dependent modules and their associated defects that indirectly results in reduction of testing time. Systematic review determined that produced, processed and object-oriented metrics were widely used as defect prediction methodologies.

In testing phase software defect prediction play an important role but at some stages it fails to predict the faulty module in software. Ishani Arora et al. [9] discussed issues related to defect prediction that includes discovering set of instances to be co-related with defects, lack of standard to measure software performance rate, finances of software defect prediction and lack of proper framework for defect prediction in software.

Fault prediction methodologies are highly helpful in increasing the performance of software. Tracy Hall et al. [10] examined 208 studies regarding the defect prediction from which 176 contained least information about the proposed model for software defect prediction due to which they became useless for other researchers to use them in their research. Only

36 studies contained brief description of proposed methodologies which was helpful in enhancing model performance and less defect prediction influence on cost and reliability of the software.

The paper [11] illustrates the important study to understand and promote the relationship between the metric defining the direction of the object and the principle of shift proneness in the subject area. The research study focuses on the identification and comparison of all learning techniques corresponding to the performance parameter with its statistical method and methodology, which can often lead to improved performance. By using machine learning approach, they suggested a reduction in the effort involved in software testing. The main aim is to reduce the period of testing using machine learning. The developer will have a known notion and idea in the vicinity about the sets of classes that present the evolving behavior and select them during testing time.

M. Kakkar et al. [12] perform a research which showed that the hybrid classifier model produced more accurate results than the single traditional approaches. The paper emphasizes on developing a framework using IBk, KStar, LWL, Random Tree, and Random Forest defect prediction attribute selection. Comparison of results is based on the values of accuracy and ROC. The proposed model enhances predictive reliability with fewer attributes. The findings and evaluation show that the system decreased the total number of parameters used in each dataset by an estimate of 6 folds, and that LWL operated much better than four other classifiers when evaluated with 10 Cross Validation (10CV) and 66% split. Thus, the time space complexity for predicting defect is reduced without affecting the prediction accuracy by using feature selection methods.

Four machine learning strategies were investigated with the proposed feature selection method to handle predictive tasks in the identification of code defects. Simulation studies carried out on public NASA data sets demonstrated the speed and performance of the diverse algorithms proposed in the paper. In most cases, we can increase the effectiveness of classified learning algorithms. The paper [13] shows that feature selection algorithms is used for software anticipation and makes the model function efficiently. Dealing with super copious characteristics is a significant challenge for classification algorithms. Thus, it is important to have a pre-defined set of features before performing classification algorithms.

In the evaluation of the commercial software product, the weighing component can be divided into two groups, static and dynamic measuring metrics. The static measurement component can be obtained from a static calculation of numerical software packages, including a few lines of code, magnitude function etc. The dynamic measuring aspect is that the anti-statistical measuring factor must be learned in the running path. Simultaneously, code evaluation must be done after the engineering work has been completed throughout the development process, making it tough to restrict the number of errors in software products. Nonetheless, code testing costs significantly and can be done in or just after the product development process if the design agencies do not have enough time to find out glitches. To address this issue in paper [14], scholars are promoting a computer defect prediction method. Software defect prediction strategies presume that multifaceted modules with a high error rate, statistic software product complexity, then foresee the software unit defect status.

Defected software modules have a huge impact of the software cost and development and often results is cost overruns, delay in project schedule and much higher budget and maintenance cost in general. The techniques of ”machine Learning Algorithms - ” such as ”Automated Neural Networks” (ANN), ”PSO” (P article Swarm Optimization”), ”Decision Trees”, ”Naïve Bayes” are analyzed using ”k-fold validation technique”. If the software faults are not eliminated prior to market testing then the bugged modules are taken back into account in the form of maintenance and the ripple effect is taken care of in the form of ”regression testing”. ”Neural networks” provide the lowest error rate in comparison to all other techniques [15].

Prediction of software defects provides engineering teams with actionable results when leading to operational performance. Empirical researches in paper [16] have been carried out on the revelation of code defects for cross-project and in-project defects. Predicting code abnormalities virtually guarantees that screening and debugging stays in a fast track mode by supplying real time data on the majority of faults that a new system is likely to encounter. Significant resources were expended by both investors and software companies on putting in place the defect caused by defects in the software product. Fault detection in faulty systems provides a base for program evaluation and also improves the efficacy of design activities. To ensure that software stakeholders make smart decisions about future programs and better allocate capital to software projects, managers must obtain a practical outcome from a predictive model.

Application development and testing focuses on the fault-prone system to increase reliability, statistical and machine learning approaches are the most commonly used or the combination of both methods is preferred. From the conclusion drawn from several work in this paper, code metric is shown to be an efficient source for providing a predictive model of fault. For all forms of metrics, class-level metrics show good predictive performance compared to method-level metrics. Machine predictive algorithm is incorporated by categorizing components into problematic and non-defective models to create predictive fault system instantly. Based on the performance such as sensitivity, precision and operating attribute analysis of the receiver shown from preceding experiments, the “SVM” and “Random Forest algorithm” provide the best model for two diverse data sets of prediction. Such two algorithms can be used as a suggestion for fault prediction machine learning algorithm. Most of these studies have demonstrated using some machine learning algorithms to use computer metric for fault prediction. [17]

Cross-project defect prediction (CPDP) has attracted a lot of attention for projects with partial historical statistics. Nonetheless, to the best of our knowledge, due to substandard

cross-project training data, the output of current approaches is usually poor. Many early researches thoroughly trained predictors (also known as prediction models) of historical structure inadequacies / bugs in more of the same software project and anticipated shortcomings in its upcoming multiplayer beta. This approach is referred as "Within-Project Defect Prediction (WPDP)". However, when a venture has limited historical defected data, WPDP has an obvious drawback. In order to address this particular issue "Cross-Project Defect Prediction (CPDP)" was introduced by He P. et al. [18]. The main purpose of "CPDP" is to classify fault-prone cases (such as classes) in a project based on defect data obtained from public software databases such as PROMISE from other programs. Although preceding "CPDP" studies considered various kinds of software metrics during the assortment process of applicable training samples, none considered the quantity of defects contained in each section (defects identified).

In the paper [19], there was built a four-stage cross-project defect prediction model, referred to as KMP, to combine data pre-processing and learning algorithm modeling at the same time. The method involves sorting instances, choosing software features, minimizing instances and predicting fault-prone class. The classic "k-means algorithm" is used to find the relatively nearest data of the software projects from historical cross-project data for the instance filtering stage.

Empirical studies have been carried out on the identification of code defects for "cross-project" and "in-project" defects by X. Chen et al. [20]. Nonetheless, a medium for the estimation of the number of defects in a forthcoming artifact release has yet to be demonstrated by existing studies. The aptitude to foresee the software defects would support software teams in software testing, planning and software standard maintenance. Therefore, considerable efforts are still required to develop an appropriate "prediction model" that can envisage the amount of software deficiencies in a potential software plan. A prediction model's primary purpose is to provide lucrative support and guidance during software testing. To ensure expense-effectiveness, it is important to try to predict the frequency of failures in a new software release.

III. Open Issues in Software Defect Prediction

This section presents the challenges faced in the prediction of software defects and the solutions proposed for these problems by researchers. It also addresses the issues that have not been resolved in this area.

A. No Generalized Framework Available

Major issue arising in software defect prediction is that different scholars use different techniques or methodologies on different data sets. Beside this very few frameworks are proposed on software defect prediction from which most are pending for future work.

Haijin Ji et al. [19] designed a new framework as KMP (k-Means and Maximum Correntropy) Criterion Based Predictor. The first step of model is instance filtering in which k-mean model is applied on the dataset obtained from cross project method and similar data is extracted by means of analog-based learning. The second step is to extract a feature from the cluster obtained after applying correlation techniques. The third step is to choose the training data after Appling random sampling based on false positive and false negative ratio. MCC (well known for handling noisy non gaussian data) was built to perform software defect prediction.

Chen et al. [21] came up with a new framework in software development. Almost nine projects were used and analyzed, and conclusion was drawn between proposed methodology and previously present methodologies. Different classification techniques were involved like NB (Naïve Byes), SVM (Support Vector Machine) and DT (Decision Trees) and for evaluating a new measure "process execution qualification rate" was developed with great advantages. Beside considering the advantages of various proposed frameworks there is no general framework proposed for predicting software defects.

B. Issues in Cross-Project Defect Prediction

It is preferred for a developed model to learn from the previous data available and can also be tested upon that data. Similar software projects can be used to obtain data by applying some programming languages. However, most of the times the scholars face problem of training data because they are unable to find similar projects related to their framework. Cross project was introduced by researchers to solve this problem. It aimed to develop defect prediction model on one software project and then the model is to be tested on another software project. Many researchers have worked to increase the efficiency of cross project prediction.

Menzies et al. [22] performed MORPH and CLIF methods on the datasets of ten different projects obtained from repository of PROMISE. He used three different classification algorithms that includes neural network, support vector machine and naïve byes in Weka tool. The proposed methodology was able to evaluate based on recall. The results showed that the methodology doesn't perform negotiation in software defect prediction on the contrary it conveyed the private data.

Zhang et al. [23] explained that by using only one transformation model cannot increase the efficiency of cross project defect prediction but using multiple transformation methodologies may result in increase in performance but above all selection of training data set can be done by using any unsupervised learning methodology.The proposed methods may have improved the reliability of cross-project models, but there was considerable scope for enhancement in this area.

C. Relationship between Various Attributes and Faults

Researchers are unable to predict attributes subsets which are non faulty or incorrect. Beside this issue the major problem is to detect which metrics level to be applied from requirement, source and design matrices for software analysis. It is also verified that software defect prediction can be done in early stages of software development life cycle. Many experiments were performed on developed software defect prediction methodologies from requirement metrics, sources

metrics and by using their combination. The results reveal that larger components with class level metrics can be used in software defect prediction.

A new metric is proposed called cross-entropy that contributes in improving the performance of defect prediction. Cross-entropy is compared with twenty commonly used metrics on twelve open-source projects to estimate the discrimination of defect proneness. The experiential results reveal that the metric of cross-entropy is more prejudiced than 50 percent of the traditional metrics. The author used combinations of both the traditional metrics with cross entropy for enhancing the defect prediction accuracy. By using this, the performance of F1-score predictive models is increased by an average of 2.8 percent. These results suggest that cross-entropy is effective in predicting code defects and in addition to the existing metric suits.

As explored from the above researches, none of them discovered a universal association between the attributes and the fault.

IV. Software Defect Prediction using Integrated Approach

In order to estimate the amount of errors in a given project, both simple and multiple linear regression structures are built. Multiple ”linear regression models” are designed for prediction based on two various independent variables, while only one independent predictive variable is considered to examine the specific effect of each predictor variable for each sample of ”linear regression model”. Each prediction method’s quality evaluation focuses on the ”data interpolation method” used. ”Cross-validation sampling” is used at times for testing in order to obtain reliable results but mostly because cross-validation sampling of information divided into 10 folds with a comparative training set volume of 8% and a test (validation) set size of 20% performs better than remain-one-out or randomized sampling for a huge data set. Model performance figures are used to show how well a certain model performs on unknown data.

Cross validation sampling technique was used for the assessment of their performance. Separate learning and testing data sets have been used in cross-validation to avoid dependency on each other, and this method is considered to be almost unbiased . However, that smaller sampler show signs of higher variance. It is thus claimed that when dealing with smaller samples, neither ”bootstrapping” nor ”cross-validation” is effective. Moreover, inconsistent evidence gathered by cross-validation can be balanced by repeating the validation process for several cycles, as in the 10-fold cross-validation used. Mechanisms of both distance and prediction error are regarded in the cross-validation sampling method to obtain a more accurate reflection of the quality of model prediction. To some degree, static metrics are effective, but their predictability declines over a certain time period. On the contrary, process metrics are better to use in order to assess a system’s quality because they record more information about defective code. System metrics show great potential for predicting the condition of launch of post-products and are effective for large amounts of data .

V. Software Defect Prediction using Cross-Project Model

Researchers and professionals have drawn keen interest in the prediction of defects on projects with minimal historical data. Use of cross-project model in prediction is very effective when the data related is not existing and analysis is performed by using the data of external project. ”Just-in-time (JIT) ” model recognizes defect-inducing modifications. JIT defect systems would provide programmers with quicker feedback when design decisions still are in mind. Unfortunately, JIT framework demands large amount of training data, similar to ”traditional defect models”, that is not accessible while programs are in initial development stages. Preliminary effort has suggested cross-project models, i.e. models trained from additional schemes through enough experience are used overcome this weakness in conventional defect estimation. JIT models seldom execute well within a cross-project context, they generally improve the efficiency when using strategies that:

1) Pick models trained and equipped with the help of other similar projects to the test project
2) Integrate data from a variety of other programs to create a broader selection of training data.
3) Incorporate multiple other project models to build an aggregate model.

It confirms therefore that for the projects with limited historical information, JIT models learned from many programs are a practical solution. Nonetheless, with the help of data that is thoroughly portrayed in the cross-project context, the JIT models operate significantly.

VI. Software Defect Prediction using Machine Learning Algorithm

The examination of ML expertise in predicting software defect is based on three supervised ML learning algorithms. The methods discussed below are ”Naïve Bayes (NB) classifier”, ”Decision Tree (DT) classifier” and ”Artificial Neural Networks (ANNs) classifier”. Following is a summarized description of the machine learning processes:

- *Naïve Bayes*: NB is basically a straightforward and effective deterministic algorithm depending upon the theorem of the Bayes with the presumption of autonomy among characteristics. NB isn’t just a single algorithm, however a set of algorithms related to a particular concept which presumes the presence and unavailability of specific class characteristics that does not relate to the participation or exclusion of another characteristics.
- *Decision Tree* Decision tree mentions a hierarchical and statistical model using perception of the object as sections to achieve the goal value of the object in the tree.

- *Artificial Neural Networks (ANNs)* Artificial Neural networks are non-linear classifiers that can model complex input-output relationships.

VII. Software Defect Prediction using Deep Forest Model

The software defect prediction can be effective for the implementation of software of good quality. In order to enhance the performance of defect prediction models, it will be necessary to develop more advanced techniques. The new model called deep forest is proposed for evaluating the accurate prediction of the software defect. This model also known as gcForest which an alternate approach of "deep neural network". It's helpful in reducing the overfitting problems and attaining high performance accuracy.

By using a new cascade strategy that basically converts RF classifier into the layered structure, deep forest can classify more crucial defect characteristics. The essential purpose of predicting defect using deep forest is to get benefit from deep learning by applying layered structure. After expanding a layer, the accuracy of the entire cascade is predicted on the test set and cascade is immediately terminated when no advancement in performance is observed. The use of RF classifier with this model promotes diversity and input discrepancies and increase the model's effectiveness to detect more crucial fault-prone features.

VIII. Conclusion

In this paper, we have provided with detailed information of all the defects prediction techniques our software development life cycle can adapt to in order to improve its quality and defer from all the change costs and maintenance costs it would have to spend in addition. To avoid these circumstances defect prediction techniques are the ultimate solution as they pave way to avoid errors prior to committing them. We have also provided with a detailed analysis of the open issues with these defect prediction techniques. Various methods of defect prediction such as integrated approach, cross project model, machine learning algorithms were the main highlight in the present paper. Based on analyzing these techniques and reading their limitations one can choose the best solution for their problem and analyze, predict and avoid all the errors.

References

[1] Sutar, S., Kumar, R., Pai, S., Shwetha, B. R. (2019, February). "Defect Prediction based on Machine Learning using System Test Parameters". In 2019 Amity International Conference on Artificial Intelligence (AICAI) (pp. 134-139). IEEE.

[2] Rathore, Santosh S., & Kumar, S. (2019). "A study on software fault prediction techniques" Artificial Intelligence Review, 51(2), 255-327.

[3] Guisheng Fan, Xuyang Diao, Huiqun Yu, Kang Yang, Liqiong Chen. (2019). "Software Defect Prediction via Attention-Based Recurrent Neural Network" Volume 2019, Article ID 6230953, Hindwai.

[4] Zhang, X., Ben, K., & Zeng, J. (2018, July). "Cross-entropy: A new metric for software defect prediction". In 2018 IEEE International Conference on Software Quality, Reliability and Security (QRS) (pp. 111-122). IEEE.

[5] J. Li, P. He, J. Zhu and M. R. Lyu, "Software Defect Prediction via Convolutional Neural Network," 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), Prague, 2017

[6] W. Liu, S. Liu, Q. Gu, J. Chen, X. Chen and D. Chen, "Studies of a Two-Stage Data Preprocessing Approach for Software Fault Prediction," in IEEE Transactions on Reliability, vol. 65, no. 1, pp. 38-53, March 2016.

[7] Kamei, Y., Fukushima, T., McIntosh, S., Yamashita, K., Ubayashi, N., & Hassan, A. E. (2016). Studying just-in-time defect prediction using cross-project models. Empirical Software Engineering, 21(5), 2072-2016.

[8] Singh, P. K., Panda, R., & Sangwan, O. P. (2015). A critical analysis on software fault prediction techniques. World applied sciences journal, 33(3), 371-379.

[9] Arora, I., Tetarwal, V., &; Saha, A. (2015). Open issues in software defect prediction. Procedia Computer Science, 46, 906-912.

[10] Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2014). A systematic literature review on fault prediction performance in software engineering. IEEE Transactions on Software Engineering, 38(6), 1276-1304.

[11] K. Chandra, G. Kapoor, R. Kohli and A. Gupta, "Improving software quality using machine learning," 2016 IEEE International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH), Noida,

[12] M. Kakkar and S. Jain, "Feature selection in software defect prediction: A comparative study," 2016 IEEE 6th International Conference - Cloud System and Big Data Engineering (Confluence), Noida, 2016, pp. 658-663.

[13] Rawat, Mrinal & Dubey, Sanjay. (2015). IEEE. Software Defect Prediction Models for Quality Improvement: A Literature Study. International Journal of Computer Science Issues. 9. 288-296.

[14] Z. Xiang and Z. Tang, "Research of Software Defect Prediction Model Based on Gray Theory," 2014 International Conference on Management and Service Science, Wuhan, 2009, pp. 1-4.

[15] P. Deep Singh and A. Chug, "Software defect prediction analysis using machine learning algorithms," 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, Noida, 2017, pp. 775-781

[16] Ebubeogu A. Felix, Sai Peck Lee. "Integrated Approach to Software Defect Prediction", VOLUME 5, 2017. IEEE Access

[17] Meiliana, S. Karim, H. L. H. S. Warnars, F. L. Gaol, E. Abdurachman and B. Soewito, "Software metrics for fault prediction using machine learning approaches: A literature review with PROMISE repository dataset," 2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), Phuket, 2017, pp. 19-23

[18] He P., He, Y., Yu, L., & Li, B. (2018). An Improved Method for Cross-Project Defect Prediction by Simplifying Training Data. Mathematical Problems in Engineering, 2018. Hindawi

[19] Ji, H., & Huang, S. (2018). A new framework consisted of data preprocessing and classifier modelling for software defect prediction. Mathematical Problems in Engineering, 2018. Hindawi

[20] C. Ni, W. Liu, Q. Gu, X. Chen and D. Chen, "FeSCH: A Feature Selection Method using Clusters of Hybrid-data for Cross-Project Defect Prediction," 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Turin, 2017, pp. 51-56.

[21] Chen N, Hoi SCH, Xiao X. Software process evaluation: A machine learning framework with application to defect management process. Empirical Software Engineering 2013; 19:1531-64

[22] Peters F, Menzies T, Gong L, Zhang H. Balancing privacy and utility in cross-company defect prediction. IEEE Transactions on Software Engineering 2013; 39:1054-68.

[23] Zhang, F., Keivanloo, I., Zou, Y. (2017). Data transformation in cross-project defect prediction. Empirical Software Engineering, 22(6), 3186-3218.

[24] Y. Zhang, D. Lo, X. Xia, and J. Sun, "An empirical study of classifer-combination for cross-project defect prediction", in Proc. IEEE 39th Annu.Comput. Softw. Appl. Conf. (COMPSAC), vol. 2. Jul. 2015

[25] D. G. Cavezza, R. Pietrantuono, and S. Russo,"Performance of defect-prediction in rapidly evolving software,"in Proc. IEEE/ACM 3rd Int. Workshop Release Eng. (RELENG), May 2015, pp. 8-11.

[26] Hammouri, Awni Hammad, Mustafa Alnabhan, Mohammad Al-sarayrah, Fatima. (2018). "Software Bug Prediction using Machine Learning Approach". International Journal of Advanced Computer Science and Applications.IJACSA.2018.

[27] Zhou, T., Sun, X., Xia, X., Li, B., Chen, X. (2019). Improving defect prediction with deep forest. Information and Software Technology, 114, 204-216.