Impact Factor-4.013 *e-ISSN: 2581-6667*
International Journal of Engineering and Creative Science, Vol. 5, No. 9, 2022
www.ijecs.net

Cross-Project Defect Prediction Using Deep Learning Techniques

Ms. Prachi Sasankar[1], Dr. Gopal Sakarkar[2]

Dept. of Computer Science, School of Science, G.H.Raisoni University,Saikheda, MP, India
Dept. of Artificial Intelligence, G. H. Raisoni College of Engineering, Nagpur, MH, India

Abstract – *Software testing is useful in increasing the quality of software, as high-quality software is devoid of errors, user-friendly, and provides client happiness [1]. The goal of fault-prediction approaches is to forecast which software modules are defective so that they can be used in later stages of software development. Initially, this paper focused on traditional types of Testing, Software defect management, Software Fault prediction and explored Machine Learning techniques. SFP models are examined and analyzed from many perspectives. The purpose of this work is to assist researchers in comprehending and exploring various facets of the fault prediction process as it relates to software fault prediction. The effort put into this article will help scholars gain a grasp on current best practices and certain well-known methodologies by searching through various libraries, websites, and research papers to discover all important publications released until 2020.*

Keywords- ***Machine Learning (ML), ML Techniques, Software Defect Management, Software Fault Prediction, Software Testing (ST), Cross Project Defect Prediction (CPDP).***

I- INTRODUCTION

Almost all industries, nowadays, are dependent on software. Software testing is the process of comparing a system to the real need in order to find any mistakes, gaps, or missing requirements. Software defect prediction models are useful for understanding, evaluating, and raising the bar for a software package. Software testing is a comprehensive and ongoing activity across the software development and maintenance process [3] [1]. Software vulnerability, software defects, software bugs lead to software security problems, further leading to resource loss, bad user experience, less client satisfaction, and increased development cost [4]. The reason behind this is defective software modules [5] [1]. To produce High-Quality Software we must thoroughly test it to check whether it is error-free, user-friendly, and with user-accepted features. These Bug-Prediction techniques tell us about faulty modules before the testing phase thus saving time and cost of production. Erroneous software has unbalanced data [1].

II- CLASSICAL TESTING TYPES

A. Types of Testing - Methods of software testing- Manual testing and Automated testing [8].

Manual Testing- This method is where software tester follows a test plan and therefore the written test plans lead them to a group of important test cases [9]. The early stages of SDLC use manual testing to detect errors and solve the errors compared to the expected output. SRS documents, design documents, source code, etc are referred to in this static testing [8] [11].

Automated Testing- Here the dynamic behaviour of code is analysed and validation of actual outputs is done. Skilled and domain knowledge experts verify input values and output values in testing [12]. Previous studies on software testing methods have been conducted, but no specific criteria have been established [12].

III-SOFTWARE FAULT PREDICTION

Software fault prediction (SFP) is a technique for increasing software quality through the use of software metrics (SQ). Statistical methods are used to forecast software errors in software fault prediction approaches [5]. Defect prediction at method-level needs attention which provides advanced information on defective

software modules [12]. Real-time defect prediction plays a vital role and is more useful in network development applications. Hybrid Learning and Deep Learning techniques modeled some methods which can improve prediction results, in cross-projects as well as within-project classification [13].

Investigators had formerly explored and suggested different classification approaches viz- Naïve Bayes, Logistic Regression, Decision Tree, Support Vector Machine, Ensemble Approaches, K-Means Clustering & Fuzzy Clustering [5] [14]. The utmost examination is done on Statistical Techniques and Supervised Techniques which uses labeled classified data. Unsupervised learning doesn't use former literal data [7].

IV-SFP MODELS

A. Prediction Model using Size and Complexity-

This paradigm overrides the programmers' and designers' unintended consequences. They are the human factors that actually commence the defects, so any attribution for flawed code depends on the individual(s) to a certain extent.

B. Machine Learning Basic Models

ML had proven results for resolving these defect prediction problems. When the domain of problems is not exactly defined and human intervention is not sufficient, ML algorithm comes into the picture. Machine learning includes different types of techniques [16] of learning like Artificial Neural Networks (ANN), Bayesian Belief Networks (BBN), Concept Learning (CL), Reinforcement Learning (RL), Genetic Algorithms (GA) and Genetic Programming (GP), Instance-Based Learning (IBL), Decision Trees (DT), Inductive Logic Programming (ILP), and Analytical Learning (AL).

1. The Probabilistic Model for Defect Prediction using Bayesian Belief Network –

BBN can be exploited to support effective decision making for SPI (Software Process Improvement), by executing the following steps –

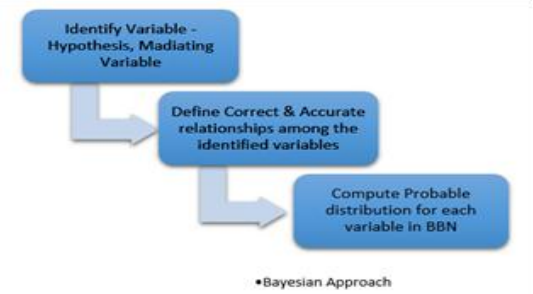

Fig. 1. Bayesian Approach [24]

2. The Fuzzy Logic Model

This approach works with an approximate value and is based on the concept of reasoning.

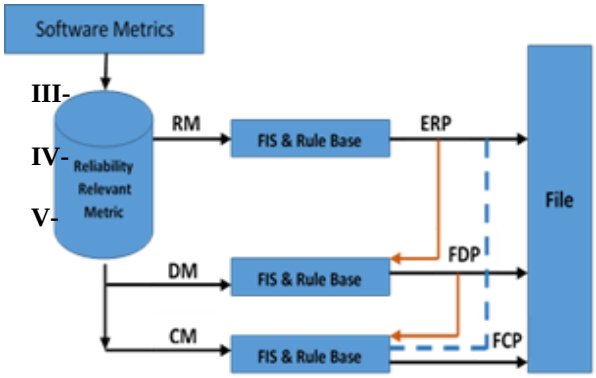

Fig 2. Fuzzy Logic Model[16]

The major advantage of Fuzzy logic, which is based on human intuition and behaviour, is that, unlike typical yes/no answers, it considers the degree of truth and hence allocates for more human-like responses [16].

3. Defect Prediction Models Based on Genetic Algorithms-

It's a form of Evolutionary Algorithm that develops solutions through natural processes including mutation, selection, and crossover [17].

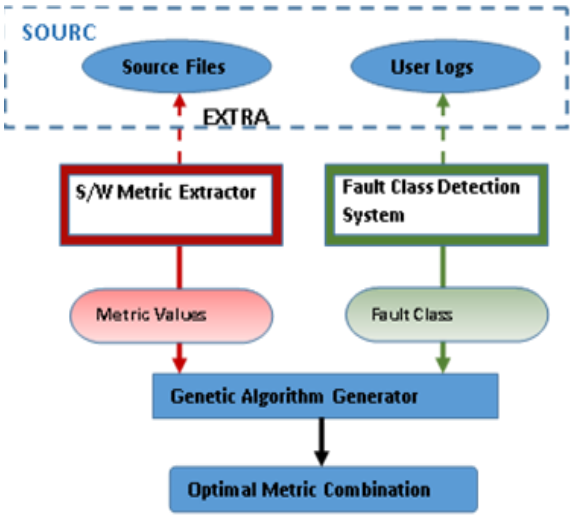

Fig 3. Genetic Algorithm Based Model [17-19]

The following are some things to think about. a) To establish whether a solution is possible, a fitness function must be present. b) A chromosome must be used to symbolize a solution once it has been discovered. c) It's time to figure out which genetic operators will be used. (Fig.3)

4. *Software Defect Prediction Models using Artificial Neural Network –*

The architecture and design of the artificial neural network are modelled by the human biological system.

C. Defect Density Prediction Model

The total verified flaws divided by the size of the software object being measured is referred to as defect consistency. The total number of problems associated with a specific software entity over a given time period is called the Number of Known Faults [18].

1. *Constructive Quality Modelling for Defect Density Prediction (COQUALMO*)

The COQUALMO model is usually applied to the first phases of the software lifecycle because of the activities of study and style.

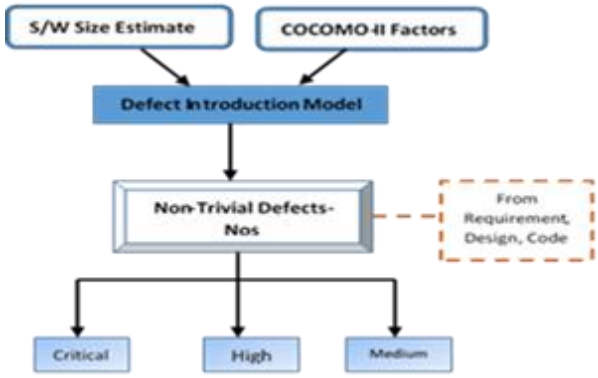

Fig 4. COQUALO Model *[26]*

2. *Defect Prediction Model based on Six Sigma Metrics*

The Six Sigma Metrics-based Model is a method for building a mathematical model for predicting functional failures in system testing that is both structured and methodical. It concentrates on V-Model-based software development projects. Prior to system testing, the Six Sigma method analyses essential parts in phases that have a direct impact on defect identification.

V- WHY USE THE CROSS-PROJECT DEFECT PREDICTION MODEL

Only when defect data is available can a good learning process be carried out. In practice, however, not all software businesses keep accurate records of past failure data or enough data from earlier projects [5]. External projects with known defect information can be used to build the training set in this study. Cross-project Defect Prediction (CPDP) [4] - From one viewpoint, such a methodology tends to the absence of chronicled imperfection information; then again, it presents heterogeneity in information, which might diminish the viability of deformity prediction models [6].

Cross-project defect prediction refers to the difficulty of effectively transferring source project knowledge to construct a defect prediction model for the target project (CPDP) [4]. There are three types of existing methods: supervised learning, unsupervised learning, and semi-supervised learning. The modules from the source project will be used to construct the model in supervised learning-based methods. Based on whether the source and target projects use the same metric set, these methods can be divided into two categories: homogeneous cross-project defect learning and heterogeneous cross-project defect prediction [1][4].

With limited historical software modules data, the concept "Within-Project-Defect–Prediction (WPDP)" also didn't come up with great results. Datasets of software projects with partial statistical measures used Cross-Project Defect Prediction (CPDP) as a countermeasure to overcome WPDP problems [13]. Research showed that the poor quality of training data used in WPDP gave substandard results. These inadequacies of properly trained historical data are a hindrance to the detection of upcoming faults in the modules, new version projects. CPDP is used to classify defect-prone modules obtained from the public datasets available [19]. Though CPDP was a major invention, the quality of defects in each section that were verified was not considered. CPDP is beneficial and effective for fault prediction; when research is done on external projects' historical data. CPDP uses separate learning and testing datasets, thus avoiding the dependency of data and giving unbiased results.

VI- PREVIOUS LITERATURE REVIEW

This section focuses on publications that used machine language techniques, neural network models, and deep learning techniques.

Sumit Mahapatra et al [7]. In this paper, Advanced machine learning capabilities like deep learning are studied. Software engineering tasks like code generation, fault prediction, defect analysis, code search and API sequence learning can use deep learning technology. In each category of vulnerability analysis, this study summarized prior studies as well as current state-of-the-art stages and approaches. And the time and effort put into it will aid scholars in understanding current best practices as well as some well- known strategies.

Ning et al [5] Software Fault Prediction techniques use unsupervised learning, which is studied in detail by the author. For this review, papers from 2000 to 2018 were evaluated. The author has completed 49 studies with in-depth analysis. In order to obtain more accurate findings, the author stressed the use of Unsupervised Learning approaches for fault prediction.

Han Cao et al [4] The challenges with standard machine learning, deep learning, and hybrid learning were discussed in this work. The author compared and contrasted all of the following strategies and methodologies. When it comes to network creation and implementation, Just-In-Time research is crucial. In the world of Just-In-Time research, real-time defect prediction is crucial. Deep Learning and Hybrid Learning have been shown to enhance prediction rates in both cross-project and intra-project scenarios, according to research.

Khuat T et al [20] Here author had used an ensemble-based bagging mechanism and SMOTE on imbalanced data for Software defect prediction. Metrics like Accuracy, Precision, Recall, and F1-score were calculated on datasets JC1, KC3, PC1, Ant-1.7, Synapse-1.2. Results of the experiment showed that combining ensemble learning with the sampling techniques increased prediction performance.

Alsawalqah H [20] et al employed a hybrid ensemble strategy with several classifiers on 12 failure datasets from PROMISE and NASA data repository in this study. The goal is to use heterogeneous ensemble approaches to study SFP. Precision, Recall, and G-means were used as base measures. The research revealed that bagging and Adaboost learning techniques gave lesser accuracy of predictions compared to ensemble methods proposed by the author.

Rehan Ullah Khan et al (2020) Cross-project defect prediction was employed in this study, which frequently reuses data from other projects. It works well when the data from the training models is more than enough to meet the project's requirements. The performance of various Machine Learning methods is determined on the dataset used to train the prediction model. Defect Induction Changes are also predicted using ML techniques. CM1, JM1, KC1, KC2, and PC1 are the five modules and repositories used in this work to model the outcomes using the PROMISE dataset. We used four distinct classifiers to build the dataset: Bayes network, Random forest, SVM, and Deep Learning based on F-measure, which made it more resilient and outperformed all other models. On performing various experiments, Random Forest and Deep learning work better than Bayes network and SVM (on all five datasets). In the proposed model of defect prediction, it requires some heterogeneous metric values that will give accurate predictions. This paper uses state-of-the-art deep learning and random forest to conduct a series of experiments on five different datasets. Using 10-fold cross-validation, the proposed model detects an error with 90 percent accuracy. On all five datasets, the obtained results show that Random Forest and Deep Learning provide more accurate predictions than Bayes network and SVM. For more accurate defect detection, the author created a Deep Learning classifier.

Jain M et al [21] Here the author experimented with AdaBoost, AdaBoost.M2, RUS Boost, SMOTE Booster, MSMOTE Boost, Data Boost using AUC, G-Mean and Balance parameters. Results had shown that RUSBoost had given best performance in handling imbalanced data. The second-best results were given by SMOTEBooste followed by SMOTEBoost Ensemble methods.

Pandey SK [22] and his colleagues' Deep feature representation was adopted by the researchers. On 12 NASA datasets, the article used stacked denoising auto-encoders for deep feature representation. The performance measurements MCC, AUC, precision-recall, and f-measure are used in the analysis of bug prediction utilizing deep representation and ensemble learning approaches. BPDET outperformed AdaBoost, Bagging, Random Forest, and Logistic Boost Author had built a Deep Learning classifier for more accurate defect identification.

Saifudin A et al [23] The author used Cross-project software prediction in this investigation. For performance validity on datasets, he defined accuracy, true positive rates, false-positive rates, and AUC. The bagging method applied which was combined with SMOTE had given the best results. Whereas Adaboost had not yielded up to the par performance as compared to other techniques applied. CM1, M W1, PC1, PC3, PC4 were used as experimental data sets from the NASA repository by the author.

Compos J et al [24] In his research, The results of this study showed that combining weak learners increased a method's overall prediction efficiency. In this work, the author used gradient boosting, stacking, and soft voting using Decision Tree, Bagging, and Random Forest. Analysis of heterogeneous ensembles for online failure

prediction was the goal of the study. The experiment was conducted on Windows XP with Service Pack 3 (SP3).

Zhigiang Li et al [25] In this paper, the author discussed how to predict heterogeneous software defects. The study took into account 30 software defect datasets from a variety of software repositories. The AUC, accuracy, Recall, and balance variables were used to compare the results of two- stage ensemble learning (TSEL), Ensemble Multiple Kernel Correlation Alignment (EMKCA), and RE Sample with replacement (RES). TSEL outperformed the baseline approaches for fault classification, according to the results of the output research.

Tong H et al [26] in this paper, He used Ensemble Multiple Kernel Correlation Alignment to align 30 software project datasets (EMKCA). In Heterogeneous Fault Prediction approaches, are Under Curve is the main differentiator. The findings of the study revealed that EMKCA outperformed alternative kernel learning and ensemble learning approaches.

Table1: Comparison of Ml Techniques

SN	Technique Used	Data Set Used	Advantages	Limitations
1	Artificial Neural Network	NASA, AR6, MDP	It has self-learning capability. The metric relationship doesn't matter	It can't handle Incorrect/Imprecise information
2	Support Vector Machine	NASA, AR1, AR6	Better prediction using a Kernel function	Large software metrics are not handled
3	Decision Tree	NASA, AR1, AR6	More accurate results are found	Decision tree construction is very complex
4	Association Rule	NASA, MDP	Historical data is used for rules generation and faultprediction	It requires all correct values of all metrics
5	Clustering	NASA, MDP	It is suitable for small datasets	The unlabeled dataset is used

VII-OUTCOME OF LITERATURE REVIEW

In SFP, no model is perfect. Applying one type of algorithm, ensemble methods, or like on any object oriented programming project dataset will not generalise the prediction strategy. Hence, more research is needed for accurate prediction percentages. Using different types of projects and more numbers of ML algorithms on datasets will give us more predictive fault prediction numbers.

- To obtain better results for SQ, deep learning analysis can be used for SFP
- Hybrid model (cross-project environment) with more than one classifier is beneficial for defect prediction.
- Cross-project SFP models will provide more accurate prediction rates.
- When compared to Bayes Network, Random Forest and Deep Learning provide more precise findings.
- Ensemble Learning could be used for combining different positive beneficial features of different models (Refer Table 1)

VIII- FUTURE SCOPE

As the software industry is growing day by day, the software's are becoming more and more complex. These complex software's are not only difficult to debug but also makes a person scratch his head when classifying the modules on the basis of faulty or else non-faulty.

Table2: Common Metrics used in Classification Models

Metric	Formula	Particular
Accuracy	$= \frac{TP+TN}{TP+TN+FP+FN}$	Overall Performance measure of model used
Precision	$= \frac{TP}{TP+FP}$	Positive Predictions accuracy
Recall	$= \frac{TP}{TP+FN}$	Positive Sample's Coverage
F1 Score	$= \frac{2TP}{2TP+FP+FN}$	Hybrid Metric UsefulnessScore

The process of classification was used in the early times of software development. Here, the aim of the research is to use fault prediction in a module rather than classifying them into categories.

We can used Ensemble Learning to forecast the number of errors in a dataset using different models, such as classifiers or experts. Linear Regression, random forest, SVM, Bagging, Boosting, MLP, and other fault prediction approaches have been employed. The major goal is to find the number of defects in certain modules with a high rate of prediction so that future debugging can be simplified [11]. To obtain better results for Software Quality, deep learning analysis is used for SFP to depict faults in the early stages of SDLC using Hybrid Model. (Fig.4)

IX- FURTHER DISCUSSION & CONCLUSION

In this paper, we reviewed various classical testing types, discussed SDM, SFP and models. We also looked into many parts of the software failure prediction process. We have observed that maximum review papers are based on Machine Learning Techniques, but the evidence is not included to support the result findings. Most recent research had used only one type of prediction model with reference to the object-oriented paradigm. Because of the variability in accuracy and false prediction results, it became vital to develop supervised, unsupervised, or deep learning methods that can close the prediction percentage gap and deliver more accurate, precise, and timely results. This is used as motivation for innovating a new hybrid model with cross- reference predictions.

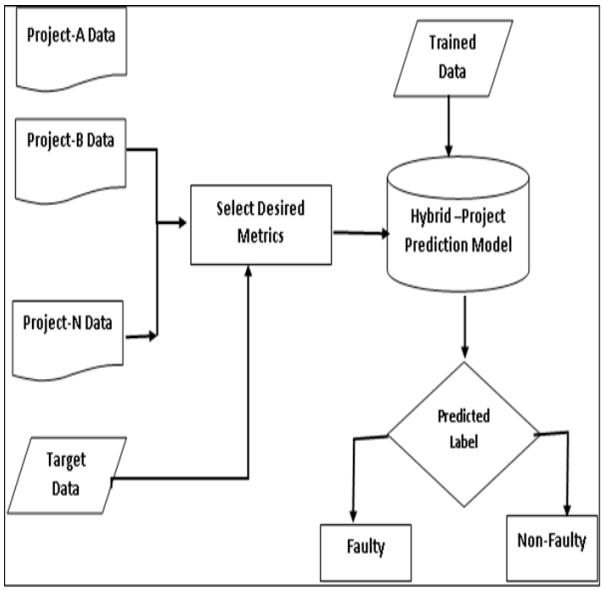

Fig.5 Proposed Methodology for software bug prediction

REFERENCES

[1] *L. K. S. R. Y. Suresh, "Statistical and Machine Learning Methods for Software Fault Prediction Using CK Metric Suite:A Comparative Analysis," ISRN Software Engineering, vol. 2014, p. 15, 2014.*

[2] *L. Z. Xiao-Yuan, "Heterogeneous defect prediction with two-stage ensemble learning," Springer Link, 2019.*

[3] *R. Wahono, "A Research Trends, Datasets, Methods and Frameworks,Systematic Literature Review of Software Defect Prediction," Journal of Software Engineering, vol. 1, no. 1.*

[4] *K. T. Tung and L. M. Hanh, "Evaluation of Sampling-Based Ensembles of Classifiers on Imbalanced Data for Software Defect Prediction Problems," Springer Nature, 2020.*

[5] *P. Sasankar, "Analysis of Test Management, Functional and Load Testing Tools," International Journal of Scientific Research in Computer Science, Engineering and Information Technology, vol. 1, no. 1, 2016.*

[6] *S.Saharudin, K.Wei and K.Na, "Machine Learning Techniques for Software Systematic Review," Journal of Computer Science, 2020.*

[7] *S.Mahapatra and S.Mishra, "Usage of Machine Learning in Software Testing," Automated Software Engineering: A Deep Learning-Based Approach. Learning and Analytics in Intelligent Systems-Springe, 2020.*

[8] *R.U.Khan, S.Albahli, W.Albattah and M.Khan, "Software Defect Prediction Via Deep Learning," International Journal of Innovative Technology and Exploring Engineering, 2020.*

[9] *R.Dabrowski and B.Wojcicki, "Applying Machine Learning to Software Fault Prediction," e-Informatica Software Engineering Journal, pp. 199-216, 2018.*

[10] *O.A.Qasem, M.Akour and M.Alenezi, "The Influence of Deep Learning Algorithms Factors in Software Fault Prediction," IEEE Access, vol. 8, 2020.*

[11] *O.A.Qasem and M.Akour, "Software Fault Prediction Using Deep Learning Algorithms," International Journal of Open Source Software and Processes,, 2019.*

[12] *N.Li, M.Sheperd and Y.Guo, "A systematic review of unsupervised learning techniques for software defect prediction," Information & Software Technology, 2020.*

[13] *S. P. a. R. Mishra, "BPDET: An effective software bug prediction model using deep representation and ensemble learning techniques," Science Direct, 2020.*

[14] *M.Jain, "Handling imbalanced data using ensemble*

learning in software defect prediction," in International Conference on Cloud Computing, 2020.

[15] *L.Perreault, S.Berardinelli, C.Izurieta and J.Sheppard, "Using Classifiers for Software Defect Detection," in International Conference on Software Engineering and Data Engineering, 2017.*

[16] *A. T. a. S. Kini, "Periodic Developer Metrics in Software Defect Prediction," in International Working Conference on Source Code Analysis and Manipulation, 2018.*

[17] *H.Cao, "A Systematic Study for Learning-Based Software Defect Prediction," IOP Conf. Series: Journal of Physics: Conf. Series, 2020.*

[18] *B. L. a. W. S. H. Tong, "Kernel Spectral Embeding Transfer Ensemble for heterogeneous defect prediction," in IEEE Transactions on Software Engineering, 2019.*

[19] *G.Tassey, "The Economic impacts of inadequate infrastructure for software testing," National Institute of Standards and Technology, 2020.*

[20] *F.Salfner and M.Malek, "A survey of Online Failure Prediction Methods," ACM Computing Surveys, 2010.*

[21] *C.Prabha and N.Dr.Shivkumar, "Software Defect Prediction Using Machine Learning Techniques," in International Conference on Trends in Electronics and Informatics, 2020.*

[22] *A.Saifudin, S.Hendric and B.Soewito, "Tackling imbalanced class on cross-project defect prediction using ensemble smote," 2019.*

[23] *A.Panichella, R.Oliveto and A.Lucia, "Cross-Project Defect Prediction Models-L'Union Fait la Force," CSMR-WCRE, pp. 164-174, 2014.*

[24] *N. Anwar and S. Kar, "Review Paper on Various Software Testing Techniques & Strategies," Global Journal of Computer Science and Technology, vol. 19, 2019.*

[25] *A. Abubakar, J. AlGhamdi and M. Ahmed, "Can Cohesion Predict Fault Density," in IEEE, 2006.*