

# Software Defect Sentry using Ensemble Learning

Darshana Gajbhiye<sup>1</sup>, Ritesh Shete<sup>2</sup>, Param Shah<sup>3</sup>, Pratik Debory<sup>4</sup> and Bhavya Vora<sup>5</sup>

<sup>1-5</sup>Department of Information Technology VPPCOE&VA, University of Mumbai

Email: Darshana.tambe@pvppcoe.ac.in, Vu4f2021012@pvppcoe.ac.in, Vu4f2021119@pvppcoe.ac.in,  
Vu4f2021029@pvppcoe.ac.in, Vu4f20211121@pvppcoe.ac.in

**Abstract**— These days, software is ingrained in every part of our life, so it's highly likely to be flawed. Software Defect Prediction (SDP) predicts both front-end and back-end flaws, among other technical problems. To predict bugs, all we have to do is use various Ensemble Learning techniques. Software Defect Prediction (SDP) is one of the most helpful techniques in the Testing Phase of the Software Development Life Cycle (SDLC). It identifies the modules that require testing because of their error-prone nature. This allows for efficient use of the testing resources without going above permitted limits. In the rapidly evolving world of software development, making sure a software system is of high quality is essential. Program testing is one of the most important techniques for guaranteeing program quality. It was found that the cost of testing accounted for almost half of the project's overall expenses. Effective and efficient software testing makes use of the least amount of software resources. Designing a procedure that may effectively carry out testing while also utilizing the fewest resources feasible for the project is therefore essential. Software Defect Sentry was thus developed. The goal of our Sentry is to find as many software bugs as it can. The world is always moving toward making important judgments based on facts. Therefore, we will do machine learning analysis on datasets that are made publicly available in this project in order to achieve the best level of accuracy feasible. The primary objective of the study is to apply several machine learning algorithms to the datasets and identify which methods produce the best outcomes. To be more precise, we demonstrated an ensemble learning model and compared the performance of KNN, Decision tree, SVM, and Naive Bayes on multiple datasets with the Ensemble technique. We also measured F1-score, accuracy, precision, and recall.

**Index Terms**—Quality assurance, Software testing, Accuracy, Ensemble learning, KNN,SVM, Naive Bayes, Comparative analysis, Machine Learning

## I. INTRODUCTION

Without software, the modern world could not run; it powers everything from cellphones and web apps to essential systems in the banking, transportation, and healthcare industries. Software has undoubtedly changed our lives, but it is not perfect. Software defects, often known as faults or bugs, are frequent during the development process and can have detrimental effects if disregarded. This introduction explores the concept of software defects and highlights the significance of finding and resolving them. Software flaws cover a wide range of irregularities in a computer programme, such as improper coding, bad designs, and strange system behaviour. These might show up as little inconveniences like typos in the user interface or as more significant mistakes like software flaws that

Jeopardise confidential information or crash the system. Since these errors can occur at any stage of the software development lifecycle, from the initial design and coding to testing and deployment, they are a continual source of frustration for software engineers and developers. There are several reasons to look for software bugs. First and foremost, flaws may affect the functioning and performance of the product. In critical applications such as driverless automobiles or medical equipment, defects can lead to fatalities and major risks to public safety. In less severe situations, software defects may lead to increased costs for customer service, decreased user satisfaction, and reputational damage to a company. Consequently, in order to guarantee the dependability and efficiency of software systems, flaws must be found and fixed.

Machine learning has been more practical in real-world scenarios during the last few decades because to the expansion in labelled data. Machine learning describes a computer's ability to learn from data. In machine learning, there are three categories. 1. Training Under Guidance 2) Instruction Without Monitoring And finally, instruction under some supervision. In semi-supervised learning, a sizable amount of unlabeled data coexists with a tiny amount of labelled data; in supervised learning, both features and labels are present; in unsupervised learning, features are the only thing present. Beyond unsupervised and semi-supervised learning, supervised learning finds application in other domains as well. One other typical use of unsupervised learning is the detection of software defects. Two further fields where supervised learning can be used are regression and classification. Classification variables are discrete, but regression labels are continuous. As a result of datasets having discrete variable labels, we address classification issues in this work. For increased fault prediction accuracy, this study recommended applying the ensemble learning technique. Datasets from the NASA archive, including CM1, KC1, and PC1, were made available to the public for use in this study. In comparison with other popular machine learning methods, such as decision trees (DT), support vector machines (SVM), and K nearest neighbors (KNN), the results of the proposed ensemble learning approach are found to be less reliable and less effective than the others. We evaluate the results using the assessment metrics, which include the F1-score, ROC curve, precision, recall.

## II. LITERATURE REVIEW

Over the years, many methods for predicting bugs have been offered by scholars. These approaches vary in terms of deployment costs, processing times, application coverage, computational complexity, and other aspects. [1] The file goes over the variables including classifier selection, model development parameters, and data pre-processing that have an impact on defect prediction systems' accuracy. The study emphasises how important model building factors are to predicting the results of the software defect classificatory. A collection of popular machine learning techniques for software defect prediction can be found in the [1] file. Among these techniques are logistic regression, support vector machines, decision trees, random forests, and artificial neural networks. To develop knowledge and offer criteria for predicting the class label for a new data set, these approaches are applied to a specified set of training data. The publication also describes the study's hybridised methodology, which enhances fault prediction accuracy by combining the best features of many machine learning techniques. The SVM software framework, PCA, naive Bayes, and random forest are all incorporated into the hybridised approach. The study looks at software utilising five datasets (PC3, MW1, KC1, PC4, and CM1) and the Weka simulation tool. A unique method for predicting software issues with Fourier learning algorithms is given in [2]. The authors state that a Boolean function that connects system settings to performance indicators is software defect prediction. A thorough explanation of the mathematical foundations of the Fourier technique is provided, along with a review of pertinent software defect prediction research, before using Fourier expressions to find the coefficients of this function, which can be used to predict potential vulnerabilities in software systems. In addition, the authors present experimental data that contrast their method with popular prediction algorithms including Naive Bayes, KNN, Random Forest, and Logistic Regression. According to the data, the Fourier learning algorithm performs better than these conventional techniques in terms of the AUC value, which is a metric used to assess these algorithms. [3] Bagogun and collaborators. To enhance the performance of defect prediction models, a method utilising SMOTE (synthetic minority oversampling technique) and homogeneous ensemble approaches (bagging and boosting) was described. Decision trees (DT) and Bayes networks (BN) served as the model's baseline classifiers. Their testing showed that the suggested strategy performed significantly better than baseline classifiers. This is how the performance of the suggested defect prediction model is evaluated: by using the AUC, Accuracy, and F measure. Five datasets were created using NASA software. On five projects with a homogeneous set of measurements, the effectiveness of DT and BN was contrasted with a homogenous ensemble technique based on SMOTE. In addition to bagging and boosting ensembles, DT and BN were utilized as basic classifiers. 8 Alazba and associates presented a method for defect prediction that contrasted the precision of seven ensembles based on trees. Ada boost, Gradient Boosting,

Hist Gradient Boosting, XGBoost, and Cat Boost were used as boosting ensembles, while Random Forest and Extra Trees were used as bagging ensembles. Tree-based bagging ensembles outperformed Tree-based boosting ensembles, according to the empirical data. In terms of prediction performance, none of the tree-based ensembles performed any worse than individual decision trees. Moreover, out of all the tree-based ensembles, the Ada boost ensemble had the lowest performance.

In order to compare the prediction performance of individual decision tree classifiers with tree-based ensembles employing Extra Trees (ET), XG-Boost, Cat Boost, Gradient Boosting, Hist Gradient Boosting, Ada-Boost, and RF, eleven NASA software project datasets were employed. They discovered that whereas tree-based ensembles typically outperformed the decision tree classifier, it underperformed for all defect datasets, with the exception of the Ada ensemble. In terms of prediction accuracy, the bagging ensembles consisting of RF and ET performed better than the others. A heterogeneous layered technique utilizing "2- levels" was employed by Tong et al. [5] to foresee software difficulties. Base classifiers are first tested and trained. Ten-fold Cross-Validation is used to validate the level 1 learners. The meta learner, or level-2 classifier, is used with the level-1 prediction outputs from this model as training data. NASA SDP datasets, including CM1, KC1, KC2, PC1, and JM1 SDP. The accuracy, F- measure, and AUC models are used to evaluate the performance of this proposed defect prediction model. The accuracy and AUC of the suggested classifier are contrasted with those of traditional classifiers. The most popular classifiers in the experiments were Support Vector Machine, Naive Bayes, K-nearest neighbor, and Artificial Neural Network Decision Tree. Confirmed are the AUC values of every conventional classifier and recommended stacking ensemble. The results show that the stacked Ensemble performs better on all datasets than any other traditional classifier. On average, heterogeneous stacking outperforms the traditional approach by 90%. [6] The machine learning-based software failure prediction approach is based on features extraction and data preparation. Feature extraction determines the metric properties of the software module by reflecting the original abstract software data and using relevant metric elements. Cleaning and making necessary modifications to the data collection are part of preparing it for data extraction. Using software defect prediction approaches like cross-company defect prediction, a model is trained to forecast issues within a certain firm using labeled samples from other organizations. This method is used in the early stages of software testing when there is insufficient data to train a model. After training on a project with abundant historical data for cross-company defect prediction, the model is utilized to anticipate faults in the target project. It's also known as the cross-project defect prediction technique. The goal of cross-company defect prediction is to minimize training data gaps by utilizing data from other businesses. Cross-company defect prediction has garnered significant attention and has been widely applied in recent times. To improve the efficacy of cross- company defect prediction, researchers have proposed two approaches: transfer defect learning and heterogeneous defect prediction. Cross-company defect prediction has many advantages, such as lower data collection costs and time requirements and higher defect prediction accuracy. Its disadvantages, however, are numerous and include the need for cautious source data selection and the potential for bias in the training set. All things considered, one useful method for software defect prediction that can improve software system reliability is cross-company defect prediction. In [7], one may get an overview of the most recent advancements in machine learning methods for software defect prediction. The study looks at how successfully a number of algorithms, such as Logistic Regression, Random Forest, and Regression Tree, uncover vulnerabilities in software. The authors propose numerous more models as well. One such model is the Software Fault Prediction System (SBPS), which forecasts when a fault will arise in a class that is tested with a new feature and whether or not a class has problems. A number of metrics for software defect prediction, such as S/W size and delivered source instruction, are also included, along with statistics for the data set, such as min, max, mean, median, skewness, and kurtosis. The authors suggest that feature. reduction and classification can improve the accuracy of massive dataset classification by comparing a number of methodologies, such as PSO, Neural Network Classification, and Change Metrics. [8] The paper provides several important prediction models, including decision trees, naïve Bayes, and support vector machines (SVMs). It also offers answers for the main problems associated with software defect prediction, such as unbalanced, noisy, and high-dimensional feature spaces. To address these problems, the study suggests using feature selection, ensemble learning, and data pretreatment methods. The authors conduct a thorough comparison of the performance of the different prediction models on a variety of datasets. [9] The study identifies six primary components: data sets, performance metrics, years, publications, classifiers, and participants. The review procedure included selecting relevant keywords, looking for publications on Google Scholar by hand, and performing statistical analysis on the collected articles. The study provides useful information about using machine learning to improve software quality. However, the report also highlights several challenges and limitations when using machine learning to anticipate software defects. Taking everything into account, this study is a useful resource for professionals and academics interested

in integrating machine learning into software engineering.[10] the importance of foreseeing software defects and the techniques that machine learning systems may use to accomplish so. The authors describe a unique approach for software defect predicting based on historical data. In order to determine how well various machine learning algorithms anticipate software problems, they also assess these methods. Among the evaluation metrics used are correlation coefficient (R2), mean absolute error (MAE), root mean square error (RMSE), relative absolute error (RAE), and root-relative square error (RRSE). It is believed that the machine learning algorithm with the lowest error rate is the most suitable for use in the prediction process. The proposed model and machine learning techniques can improve software performance quality and reduce maintenance expenses. [11] the use of the automated machine learning method auto-sklearn to forecast software defects in open- source software projects. The authors evaluate auto-sklearn's performance against other prediction models using data from six open-source software projects. In terms of prediction accuracy and F1 score, they found that auto-sklearn outperformed the other models. The performance of auto- sklearn was further examined by the authors in relation to several hyperparameters. Overall, the study demonstrates how software quality may be raised and development expenses can be reduced using automated machine learning. Ensemble learning for software fault prediction using unbalanced data [12]. The authors combine sampling techniques with popular categorization algorithms to develop a workable ensemble model for the software defect prediction problem. They conducted their analysis using seven significantly skewed binary datasets from the PROMISE database of software defects. The empirical results show that the concept performs promisingly when compared to independent classifiers. The article provides software engineers with a useful plan for dividing up their limited time and resources so that they may put in place an efficient software testing process. An essential part of the software development life cycle is software testing [13]. The success of software as a whole depends on the testing phase. Various machine learning techniques are proposed for efficient software defect detection. C. Manjula et al. [14] introduced a mixed machine learning technique. A genetic algorithm is included to enhance the fitness function, and a decision tree (DT) is employed to evaluate the optimized features for more optimization. It is demonstrated that the hybrid approach offered outperforms an ID3-based decision tree in terms of performance. The performance concern is effectively addressed by the suggested approach. A feature-engineering- focused ensemble learning strategy was proposed by I Laradji et al. [15]. The robustness issue with earlier defect prediction methods is addressed by the suggested approach, which combines efficient feature selection with ensemble learning. The average probability ensemble (APE), an ensemble of seven machine learning models, is another component of the suggested approach. It is shown to outperform other machine learning models, including weighted SVM and random forests. It is discovered that for the PC2, PC4, and MC1 datasets, the APE in conjunction with greedy forward selection yields superior results. A hybrid approach to defect prediction proposed by O Arar et al. [16] uses an artificial neural network (ANN) to make predictions and an artificial bee colony (ABC) method to optimize the weight of the ANN. An study of the proposed technique's performance against the Naive Bayes, Random Forest, C4.5, Immunos, and AIRSand algorithms reveals that it performs similarly to Random Forest on KC1 datasets and yields good accuracy on KC2 and CM1 datasets. Results on the PC1 and JM1 datasets show a decline in performance compared to previous methods. According to the author, greater outcomes might come from concentrating more on feature engineering.

### III. PROPOSED METHODOLOGY

Fig. 2 displays the suggested ensemble learning model that was learned on CM1. This demonstrates that ensemble learning—an amalgam of Classification by Regression and KNN model is used following basic feature preprocessing. KNN is used once more after the ensemble approach to improve classification accuracy. Regression model classification involves a number of sub-processes, each of which has an operator trained on the regression model after the operator in the sub-processes is trained on the classification model. K closest neighbor, or KNN, is a well- known classification model. After calculating the distance between each point in the training set and the test point, it locates the k closest neighbor between the points.

### IV. ENSEMBLE LEARNING

Ensemble learning is a powerful and well-liked machine learning technique that aims to increase the expected accuracy and durability of models by combining multiple distinct models. Instead of relying solely on one model to produce predictions, ensemble techniques incorporate the expertise of several models, usually producing better results. Ensemble learning's basic tenet is that, when merged, several models can offer a more complete and accurate understanding of complex data patterns. This diversity in models can be achieved in a number of ways, such as by using unique features, data subsets, or algorithms. Their combined predictions could lead to better

overall performance. Ensemble learning comes in various forms, including 1) Bagging, 2) Boosting, 3) Stacking, 4) Max Voting, and 5) Averaging. Figure 3 compares various ensemble learning techniques.

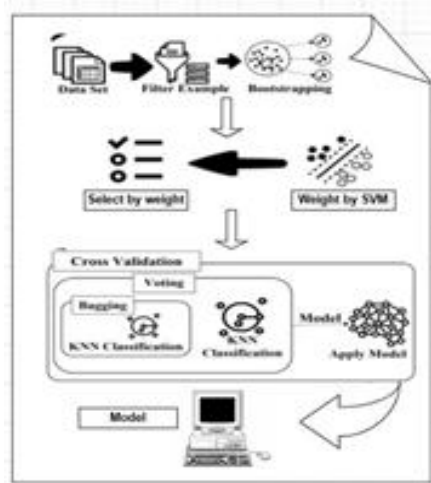


Fig. 2. Proposed Architecture

| Ensemble Technique | Concept                                       | How It Works                                 | Advantages                          | Disadvantages                                              |
|--------------------|-----------------------------------------------|----------------------------------------------|-------------------------------------|------------------------------------------------------------|
| Bagging            | Bootstrap Aggregating                         | - Resamples training data subsets.           | - Reduces variance and overfitting. | - Limited improvement for strong base models.              |
|                    |                                               | - Trains separate models on subsets.         | - Robust to noisy data.             | - Reduced model interpretability.                          |
|                    |                                               | - Combines predictions (e.g., averaging).    | - Parallel processing possible.     | - Inefficient for imbalanced datasets.                     |
| Boosting           | Iterative Model Training with Weighted Errors | - Sequentially trains models.                | - Significantly improves accuracy.  | - Sensitive to noisy labels.                               |
|                    |                                               | - Assigns more weight to misclassifications. | - Handles noisy data and outliers.  | - Requires substantial computational resources.            |
|                    |                                               | - Aims to reduce bias and increase accuracy. | - Suitable for weak learners.       | - Potential for overfitting.                               |
| Stacking           | Combining Predictions with a Meta-Model       | - Combines predictions from base models.     | - Enhances predictive accuracy.     | - Increases model complexity.                              |
|                    |                                               | - Uses a meta-model to learn how to combine. | - Flexible in choosing base models. | - Requires higher computational resources.                 |
|                    |                                               | - Effective for complex tasks.               | - Reduces overfitting.              | - Complex parameter tuning for base models and meta-model. |
|                    |                                               | - Provides some interpretability.            |                                     | - Risk of data leakage between models.                     |
|                    |                                               |                                              |                                     | - Potential for overfitting.                               |

Fig. 3. Comparison of Techniques models

## V. CONCLUSION

It is determined that the most crucial approach to guarantee software system quality is through software testing. Consequently, we suggested an ensemble learning approach in this research study to forecast software flaws. We looked into a variety of ensemble learning techniques. A possible method for predicting software defects is the ensemble learning model that has been suggested. To improve accuracy, it makes use of a variety of strategies, feature preprocessing, and an iterative framework. This model offers a comprehensive perspective on software defect prediction by taking into consideration both the classification and regression parts of the problem by integrating Classification by Regression and KNN. The model's interconnected sub processes present a sophisticated, but potentially effective, framework for finding patterns in defect data. The focus on similarity-based classification and data quality is in line with best practices for software defect prediction, which can eventually lead to more accurate and consistent defect prediction models.

## FUTURE WORK

With much anticipation, we will be implementing our ensemble learning project in software defect prediction, but first we will be exploring feature engineering and selection methods in more detail. Our objective is to find the most relevant characteristics by experimenting with various feature extraction techniques, with the ultimate goal of achieving an impressive accuracy rate of approximately 99%.

## REFERENCES

- [1] Matloob, F., Ghazal, T. M., Taleb, N., Aftab, S., Ahmad, M., Khan, M. A., ... & Soomro, T. R. (2021). Software defect prediction using ensemble learning: A systematic literature review. *IEEE Access*, 9, 98754-98771.
- [2] Singh, P. K., Agarwal, D., & Gupta, A. (2015, March). A systematic review on software defect prediction. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 1793-1797). IEEE.
- [3] Catal, C., & Diri, B. (2009). A systematic review of software fault prediction studies. *Expert systems with applications*, 36(4), 7346-7354.
- [4] Wang, S., & Yao, X. (2013). Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability*, 62(2), 434- 443.
- [5] Yang, X., Tang, K., & Yao, X. (2014). A learning-to-rank approach to software defect prediction. *IEEE Transactions on Reliability*, 64(1), 234- 246.
- [6] 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.
- [7] Pal, Sourabh. (2021). Generative Adversarial Network-based Cross- Project Fault Prediction.
- [8] Z. Tian, J. Xiang, S. Zhenxiao, Z. Yi and Y. Yunqiang, "Software Defect Prediction Based on Machine Learning

Algorithms”

- [9] Khan, Shahzad Ali, and Zeeshan Ali Rana. “Evaluating Performance of Software Defect Prediction Models Using Area Under PrecisionRecall Curve (AUC-PR)” In 2019 2nd International Conference on Advancements in Computational Sciences (ICACS), pp. 1-6. IEEE,
- [10] Gong, Lina, Shujuan Jiang, Rongcun Wang, and Li Jiang. “Empirical
- [11] Evaluation of the Impact of Class Overlap on Software Defect Prediction” In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 698 -709. IEEE, 2019.
- [12] Nawaz, Ali, Attique Ur Rehman, and Muhammad Abbas. ”A Novel Multiple Ensemble Learning Models Based on Different Datasets for Software Defect Prediction.” arXiv preprint arXiv:2008.13114 (2020).
- [13] Shepperd, Martin, David Bowes, and Tracy Hall. “Researcher bias: The Use of Machine Learning in Software Defect Prediction” IEEE Transactions on Software Engineering 40, no. 6 (2014): 603-616.
- [14] Liang, H., Yu, Y., Jiang, L., & Xie, Z. (2019). Seml: A semantic LSTM model for software defect prediction. IEEE Access, 7, 83812-83824.
- [15] Laradji, I. H., Alshayeb, M., & Ghouti, L. (2015). Software defect prediction using ensemble learning on selected features. Information and Software Technology, 58, 388-402.
- [16] Prabha, C. L., & Shivakumar, N. (2020, June). Software defect prediction using machine learning techniques. In 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184) (pp. 728-733). IEEE.
- [17] Q. Liu, J. Xiang, B. Xu, D. Zhao, W. Hu and J. Wang, "Aging-Related Bugs Prediction Via Convolutional Neural Network," 2020 7th International Conference on Dependable Systems and Their Applications (DSA), 2020, pp. 90-98.