A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning

G Raghu Vamshi[1] , K M Kabilan[2], Praharsh Sai[3] ,T. Lavanya[4], Dr. S. Shiva Prasad[5]
[1,2,3]*Studentss, Department of CSE (Data Science), Malla Reddy Engineering College, Secunderabad, Telangana,500100*
[4]*Assistant Professor, Department of CSe(Data Scince), Malla Reddy Engineering College, Secunderabad, Telangana,500100*
[5]*Professor, Department of CSe(Data Scince), Malla Reddy Engineering College, Secunderabad, Telangana,500100*

***Abstract*—Defect prediction is an important area of research in software engineering. Identifying defects early can significantly improve software quality and lower development costs. The goal of software defect prediction is to spot potential faults in source code before the testing phase. This is done by using data mining and machine learning techniques. This research connects software engineering and data analysis to improve defect prediction performance, focusing on feature selection and effective machine learning models. Five publicly available NASA datasets, CM1, JM1, KC2, KC1, and PC1, are analyzed using various classifiers, including Random Forest, Logistic Regression, Support Vector Machine, Decision Tree, and ensemble voting methods. Feature selection is applied to optimize the datasets and increase predictive accuracy. Besides basic defect classification, the proposed system offers useful predictive insights like defect risk scores, severity levels, and criticality status. This enables more informed and actionable decision-making. The WEKA machine-learning workbench is used for data preprocessing, model training, and evaluation. Minitab is used for statistical validation. Experimental results show a significant improvement in defect prediction accuracy when feature selection is used. The ensemble voting classifier performs the best among all the models tested. Additionally, the system features a modern user interface that improves visualization, clarity, and interpretability of results, making the approach practical and effective for assessing software quality in the real world.**

***Index Terms*—Software Defect Prediction, Feature Selection, Machine Learning, NASA Software Metrics Datasets, Ensemble Voting Classifier, Defect Risk Assessment, Severity Analysis, Predictive Analytics, WEKA, Software Quality Assurance, Modern User Interface**

I. INTRODUCTION

In software systems, any unexpected behavior or performance that doesn't quite meet what a client is looking for is known as a software defect. These defects usually come to light during the software testing phase, where testers spot unusual or abnormal behavior in the system. You might also hear the term software fault, which refers to inconsistencies that occur during the software development journey, often leading to system failures that don't align with user expectations [3]. In this case, an error is seen as a human action that results in an incorrect outcome, while a defect is how that error appears in the software product, which can ultimately lead to system failure.

Predicting software defects aims to catch faulty software modules early on in the software development life cycle (SDLC). Traditional techniques for detecting defects like code reviews, unit testing, integration testing, system testing, and beta testing prove to be effective but become harder to manage as software systems expand in size, complexity, and code volume [4][5].

Recently, software defect prediction has attracted a lot of attention due to its direct effects on software quality, development costs, and delivery timelines. Defective modules can hamper software reliability, causing budget overruns, project delays, and increased maintenance efforts. Detecting and preventing defects are crucial parts of software quality assurance, with defect prevention concentrating on spotting and

eliminating potential faults before the software is launched. Defect prediction plays a key role in this process by anticipating faults, errors, or defects in advance, which helps in efficiently allocating resources and enhancing overall software quality [6].

By predicting software defects early on, developers and project managers can concentrate their testing and maintenance efforts on high-risk modules. This proactive strategy reduces development costs, limits rework, and boosts user satisfaction. Consequently, defect prediction models have become an invaluable resource for assessing software quality and ensuring reliable software delivery. When defect prediction techniques are integrated into the development process, it significantly enhances the effectiveness of software testing, enabling teams to prioritize troubled modules and refine testing strategies [7]. There are various methods for predicting software defects proposed in the literature; however, no single approach excels across all datasets because of differences in data characteristics. Among these methods, machine learning stands out as one of the most effective techniques for defect prediction. Machine learning algorithms can learn from historical software metrics data, identify complex relationships, and make precise predictions with little human involvement. Techniques like classification, clustering, and regression are widely utilized to boost software quality and efficiency. Machine learning-based defect prediction carries numerous advantages, such as enhanced prediction accuracy, effective prioritization of testing resources, and improved decision-making in software quality assurance. By pinpointing high-risk and critical software modules early, developers can tackle potential problems before the software is deployed. This research aims to enrich the testing phase by utilizing machine learning techniques to elevate defect prediction accuracy, providing more insightful predictive insights and facilitating the creation of high-quality, reliable, and user-satisfying software systems.

II. LITERATURE SURVEY

Software defect prediction has become a hot topic in software engineering, mixing machine learning, software metrics, and data-driven methods. A bunch of studies from 1990 to 2022 have tackled various angles of fault prediction, from how to develop models to the best ways to evaluate them. Early on, researchers pointed out that the size and complexity of software played a big role in defect occurrences, showing that as the lines of code increase, so do the defects this makes tracking measurable software qualities super important for spotting bug-prone parts[11].

As machine learning progressed, experts started using smart algorithms to make defect predictions more accurate. Neural network models were some of the first machine learning methods applied in this area, and research showed they often beat out traditional statistical approaches. Many studies relied on benchmark datasets like PROMISE and NASA repositories, zeroing in on metrics such as lines of code and code quality indicators as key factors. They also looked into ensemble learning and hybrid machine learning methods to boost predictive performance[12].

With software systems getting bigger and more complex, researchers sought out ways to pinpoint defect-prone components without having to test each module thoroughly. They found that static code metrics and conformity indicators were worth a look due to their cost-effectiveness and consistency across projects. These findings revealed that combining reliable measurements with machine learning techniques could highlight quality indicators early in the development process, ultimately slashing development and maintenance costs[13][14].

Machine learning classification techniques have also found applications in other areas, like medical diagnosis and pattern recognition. Studies using tools like WEKA tested classifiers including Naive Bayes, K-Nearest Neighbors, Support Vector Machines, decision trees, and neural networks, evaluating them on metrics like accuracy and precision. The results consistently showed that machine learning models could perform well when the right features and datasets were in play.

Many researchers have pointed out that data quality seriously impacts defect prediction models; noisy or incomplete datasets can really hurt how well predictions perform. They stressed the importance of data preprocessing and cleansing in creating reliable models. Some studies even proposed hybrid strategies that combine evolutionary algorithms, artificial neural networks, and support vector machines, showing better prediction accuracy with NASA software metrics datasets[15-17].

Reviews looking at machine learning classification techniques grouped defect prediction methods into

supervised, unsupervised, and semi-supervised categories. They concluded that there's no one-size-fits-all algorithm that works best for all datasets, which calls for tailored modeling strategies based on the dataset at hand. Techniques like classification, clustering, and association rule mining were also explored to uncover hidden defect patterns without too much testing effort.

Recent work has underscored the value of catching defects early with advanced techniques like fuzzy logic systems and ensemble learning models. It's been shown that selecting the right software attributes can dramatically improve prediction efficiency and accuracy, while poor feature selection can lead to worse outcomes. Because of this, feature selection has become a crucial part of modern defect prediction research[9].

Meta-analyses and large empirical studies have reinforced the success of ensemble learning and stacking techniques but also pointed out shortcomings like inconsistent reporting and simplistic benchmark datasets. New developments in deep learning, time-series analysis, and effort-aware defect prediction show promising results by factoring in things like defect-fix effort and the temporal behavior of software systems.

Despite lots of research in this area, many methods still either skip over feature selection or use outdated datasets. Plus, several studies focus mainly on classification accuracy instead of the bigger picture of defect prediction effectiveness. This current research sets itself apart by using updated datasets, incorporating feature selection techniques, and applying multiple machine learning classifiers to boost defect prediction accuracy. This approach aims to offer a more solid and informative prediction framework that addresses the key issues found in earlier studies.

III. PROPOSED METHODOLOGY

Software defect prediction comes with its fair share of challenges, mainly due to the variety of datasets, the differences in software metrics, and the unique characteristics of each project. While there are many defect prediction methods discussed in existing literature, there isn't one approach that shows consistent success across all datasets. The success of a prediction model is largely influenced by the type, quality, and distribution of the dataset being analyzed. As software systems grow larger and more complex, it becomes trickier to choose the right prediction technique. That’s why adopting a flexible and data-driven approach is so important for creating reliable defect prediction models.

Machine learning has become one of the most trusted and effective methods for predicting software defects, thanks to its ability to learn patterns from past data and adjust to the unique characteristics of various datasets. In this research, we use machine learning algorithms to pinpoint software modules that are likely to have defects by analyzing different software metrics. A crucial aspect of our approach is featuring selection, which focuses on identifying the most important features that strongly relate to the input variables and the target class. By filtering out irrelevant and redundant features, our method not only simplifies the model but also boosts learning efficiency and enhances prediction accuracy. For this study, we utilized the WEKA machine learning workbench to carry out data preprocessing, feature selection, and classifier implementation. We trained and evaluated various machine learning models using the selected features to measure their defect prediction performance. To confirm the effectiveness of our approach, we conducted statistical analysis using a two-tailed t-test via the Minitab statistical tool. This analysis ensures that the performance improvements we observed are meaningful and not just random variations, thus boosting the reliability and credibility of our experimental results.

3.1 Feature Selection

Feature selection is really important when it comes to tackling problems in predictive modeling since it helps remove unnecessary or duplicate data. The main aim here is to boost how well prediction models perform, which means getting faster, more accurate, and cost-effective predictions. Choosing the right variables for a predictive model can be tricky because it depends on the specific domain and the characteristics of the dataset. It’s often hard to figure out manually which features will actually help improve the model’s accuracy.

By using feature selection, you can automatically identify relevant variables that are closely linked to the target outcome. This process not only helps get rid of irrelevant or only somewhat relevant features that might drag down model performance but also streamlines things overall. Good feature selection can enhance training efficiency, lower model complexity, cut down on overfitting by filtering out noisy data, and ultimately boost prediction accuracy. Plus, having a smaller set of

relevant features makes training quicker and makes machine learning models simpler to interpret. Figure 1 gives a general overview of feature selection, showcasing how to pinpoint key variables while discarding redundant or irrelevant ones.

3.2 Datasets

This research looks at five public benchmark datasets from the PROMISE repository: JM1, CM1, KC1, PC1, and KC2.

Table 1. Dataset

Fid	Defect id	Creation Date	Software Name	Status Category	Defect fix time	Resolution Category
172.217.12.173-10.42.0.151-443-33895-6	GCC-49282	04-06-2019	middle-end	resolved	125	fixed
10.42.0.211-31.13.80.12-48416-443-6	GCC-36574	19-06-2018	middle-end	resolved	29	fixed
172.217.9.238-10.42.0.151-443-44152-6	GCC-77269	16-08-2016	middle-end	resolved	6	fixed
10.42.0.211-23.23.129.231-51242-443-6	GCC-78479	22-11-2016	fortran	resolved	0	fixed
10.42.0.211-77.234.44.199-36959-80-6	GCC-632	12-10-2019	c++	resolved	115	fixed
172.217.11.46-10.42.0.151-443-53998-6	GCC-4071	21-08-2021	libstdc++	resolved	50	fixed
172.217.11.42-10.42.0.42-443-44392-6	GCC-67037	27-07-2015	rtl-optimization	resolved	373	fixed
10.42.0.211-47.89.69.237-35895-443-6	GCC-83112	22-11-2017	libgcc	resolved	8	fixed
172.217.12.206-10.42.0.42-443-46233-6	GCC-7111	24-06-2022	libstdc++	resolved	10	fixed

You can find detailed info about each dataset in above mentioned table, which covers the total number of instances, number of attributes, and the percentage of defective instances. These datasets form the basis for training and testing the machine learning models, allowing us to evaluate feature selection and predictive performance across different software projects.

3.2 Algorithms and Mechanisms

This research applies a range of machine learning algorithms to predict software defects with greater accuracy. Each algorithm offers unique strengths, making it suitable for various data types and pattern recognition tasks. The following sections outline the principal algorithms used in this study and their specific functions in the context of software defect prediction.

3.2.1 Logistic Regression

Logistic Regression is a statistical method designed to estimate the probability of a binary outcome, such as determining whether a software module is defective or non-defective. The model uses the logistic function to generate probabilities between 0 and 1, ensuring that the probabilities across all classes sum to one. Logistic Regression is particularly effective when the relationship between input features and the output variable is approximately linear. It produces a transparent and interpretable model, enabling researchers to identify which features most significantly influence defect prediction. In this study, Logistic Regression serves as a baseline for evaluating more complex algorithms.

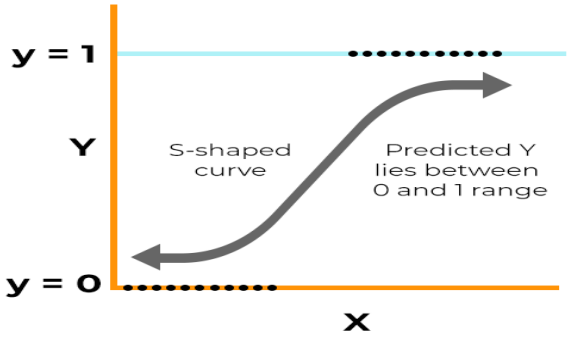

Figure 1. Logistic Regression

3.4.2 Bayesian Network

Bayesian Networks are probabilistic graphical models that represent conditional dependencies among variables using directed acyclic graphs. These models combine multiple probability distributions to efficiently capture complex relationships between features. By leveraging conditional independence, Bayesian Networks reduce computational demands and enhance prediction accuracy. In the context of software defect prediction, they are valuable for modeling the interplay between various software metrics and the likelihood of faults. This approach supports a clearer understanding of feature

interdependencies and informs decisions aimed at defect prevention.

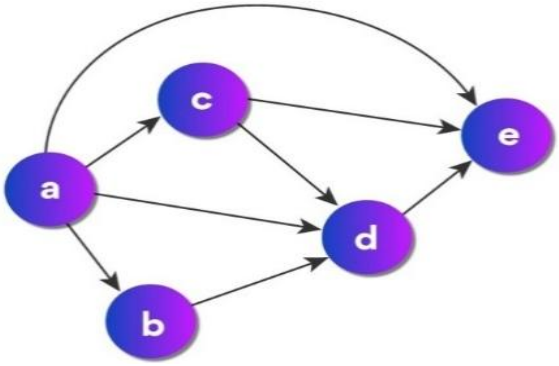

Figure.2 Bayesian Network

3.4.3 Multilayer Perceptron (MLP)

The Multilayer Perceptron is a feedforward artificial neural network consisting of an input layer, one or more hidden layers, and an output layer. Each layer is composed of multiple neurons connected by weighted edges, which are adjusted during training through backpropagation. MLPs are adept at capturing complex, non-linear relationships between input features and target variables. This capability makes them well-suited for software defect prediction, where data patterns can be intricate. By learning from historical data, MLPs can generalize to new modules, thereby improving the accuracy of defect prediction and overall software quality.

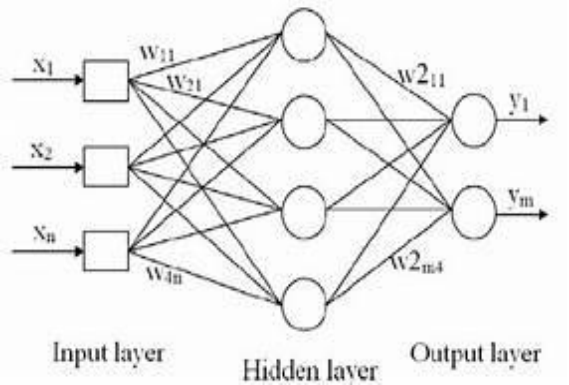

Figure 3. Multilayer Perceptron (MLP)

3.4.4 J48

J48 is an implementation of the C4.5 decision tree algorithm, commonly used for classification tasks. It constructs a decision tree by splitting the dataset based on attribute values. Internal nodes represent feature tests, branches correspond to possible outcomes, and leaf nodes indicate class labels. J48 is capable of handling both continuous and discrete features and includes automatic pruning to mitigate overfitting. For defect prediction, J48 offers interpretable results, clearly illustrating which features are associated with faulty software modules. Its straightforward design and robustness make it sui This research looks at five public benchmark datasets from the PROMISE repository: JM1, CM1, KC1, PC1, and KC2. These datasets have software metrics and defect labels, making them a good fit for testing defect prediction models. learning models, allowing us to evaluate feature selection and predictive performance across different software projects.

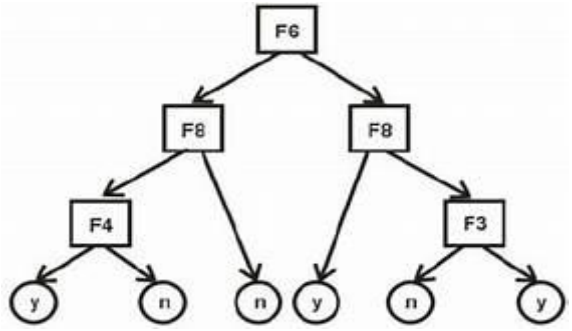

Figure 4. J48

3.4.5 Decision Stump

A Decision Stump is a simple decision tree with only one level, making predictions based on a single input feature. It operates using basic if-then rules and is often employed as a weak learner in ensemble methods such as boosting. Despite its simplicity, a Decision Stump is useful for assessing feature importance and establishing a baseline for more advanced models. Its low computational requirements and rapid training process make it appropriate for initial evaluations of large datasets before deploying more complex algorithms.

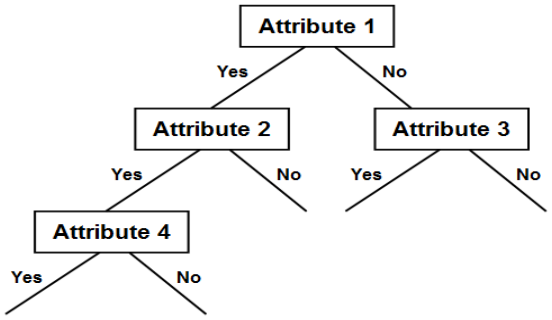

Figure 5. Decision Stump

3.4.7 Random Forest

Random Forest is an ensemble learning technique that constructs multiple decision trees using bootstrapped samples from the training data. Each tree generates an independent prediction, and the final result is determined by majority voting. Random Forest enhances prediction accuracy by reducing overfitting and capturing the diversity present in the data. It is capable of managing large datasets with numerous features, making it well-suited for software defect prediction involving multiple metrics. Additionally, Random Forest provides feature importance scores, which assist in identifying the most significant metrics for defect detection.

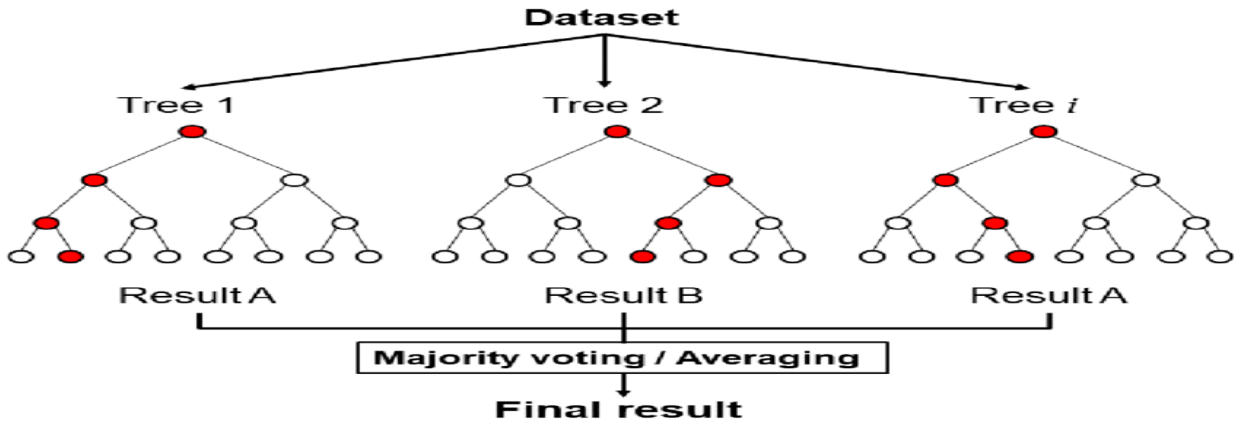

Figure 7. Random Forest

3.4.6 Support Vector Machine (SVM)

Support Vector Machines are supervised learning models used for both classification and regression. SVMs map input data into a high-dimensional space and identify an optimal hyperplanene that separates classes with the maximum possible margin. In software defect prediction, SVMs are instrumental in distinguishing between defective and non-defective modules, even when the data is complex or contains noise. Their strong generalization capability ensures consistent performance on previously unseen data.

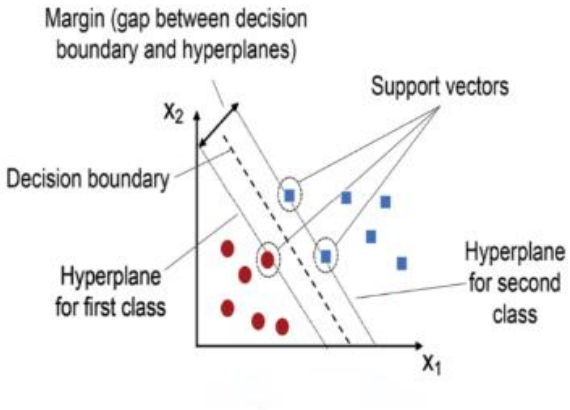

Figure 6. Support Vector Machine (SVM)

IV. RESULTS

To examine the defect prediction performance with feature selection for better accuracy, this research uses publicly available benchmark datasets from the NASA Promise repository. The datasets include CM1, PC1, JM1, KC1, and KC2. We also use five classifiers: Lazy IBK, Bayes Net, Rule ZeroR, Multilayer Perceptron, and J48. The main goal of this evaluation is to assess how well the proposed feature-selection-based approach improves defect prediction accuracy while also lowering computational complexity. We conducted experiments on five widely used publicly available NASA software defect datasets: JM1, CM1, KC1, PC1, and KC2. These datasets are commonly used in defect prediction research because of their diverse characteristics and different levels of complexity. We used WEKA as the main machine learning platform for implementing classifiers due to its strong support for data preprocessing, feature selection methods, and classification algorithms. We also employed MiniTab for statistical analysis and validation of the results. We considered several performance metrics, including accuracy, precision, recall, F-measure, and training time, to ensure a thorough and fair evaluation of model performance.

The results from the experiments clearly show that the proposed framework significantly improves defect prediction performance compared to traditional approaches that use the complete feature set without preprocessing.

Figure 8. Home Screen

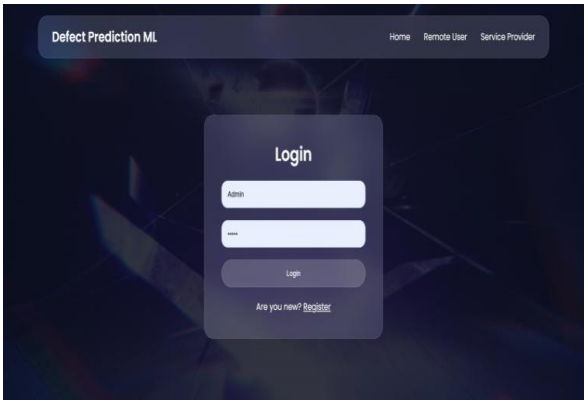

Figure 9. Login Page

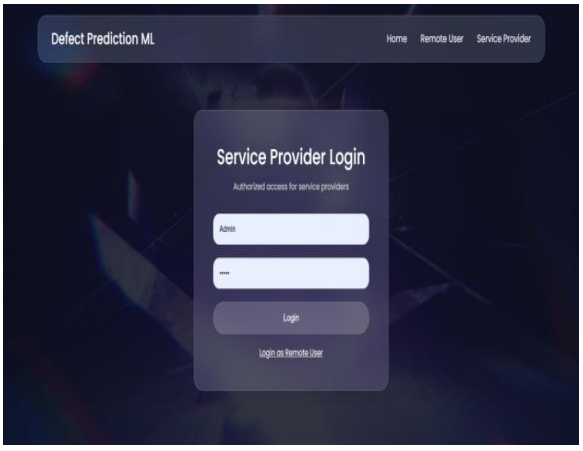

Figure 10. Admin/Service Provider Login Page

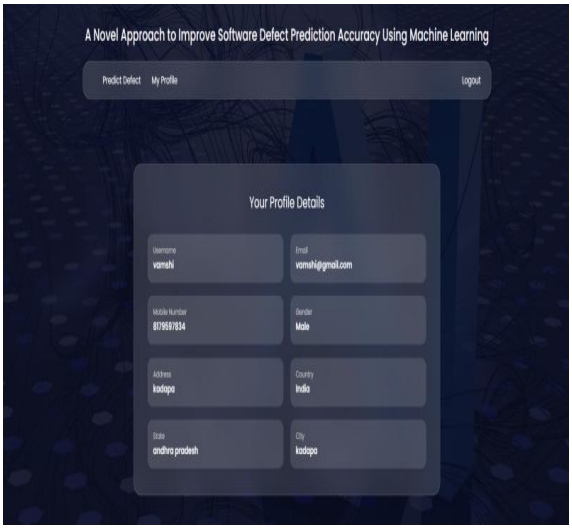

Figure 11. Remote User Profile Page

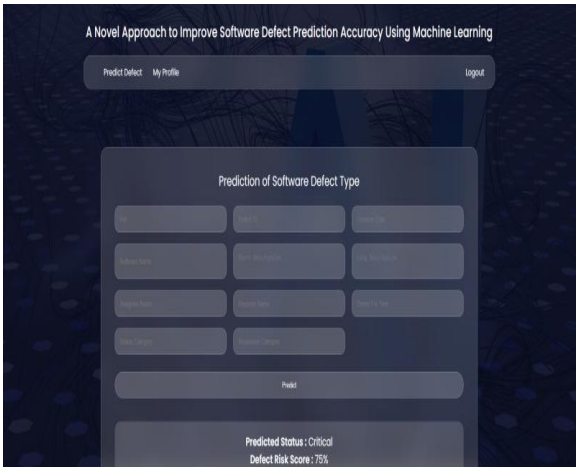

Figure 12. Software Defect Prediction Page

Figure 13. Outcomes of Predictione

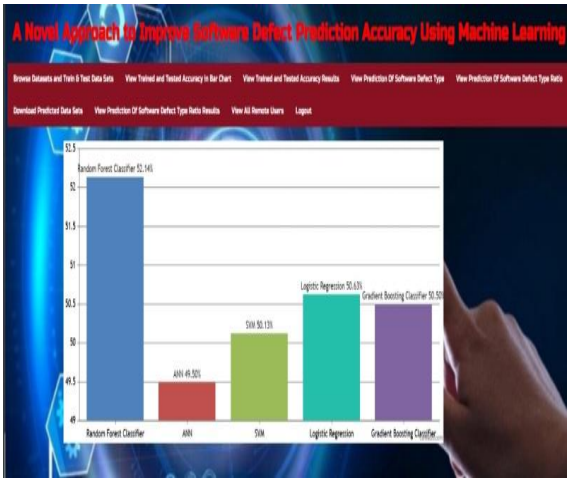

Figure.14 Performance comparison

Figure 15. Line Chart Showing Critical and normal Model Percentages

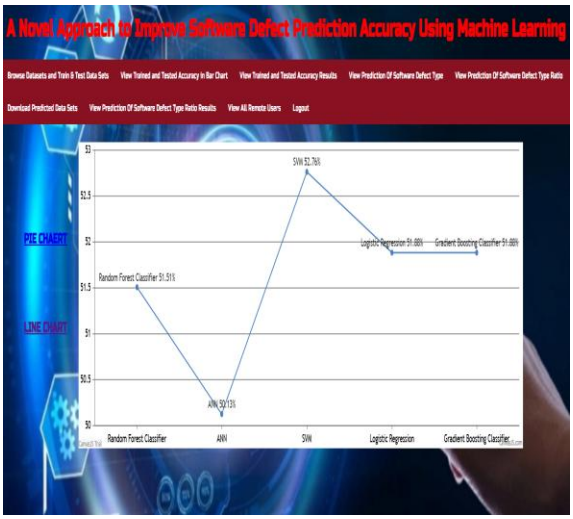

Figure 16. Line Chart Showing Performance of all Model's

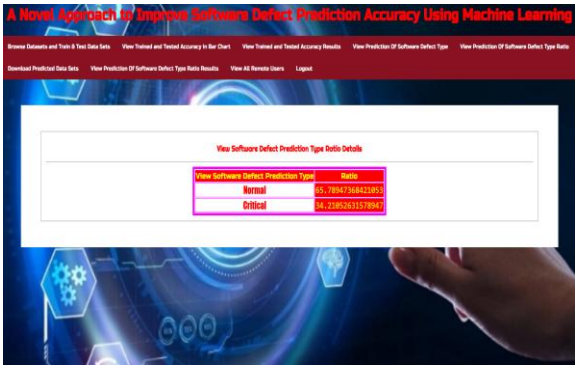

Figure 17. Software Defect Prediction Type Ratio

4.1 Dataset Overview

The datasets in this study include software modules labeled as faulty or non-faulty. Each module is defined by software metrics like code complexity, size, and structural attributes. These metrics give important insights into the software's quality and are often used as predictors in defect prediction models. The datasets differ significantly in the number of instances, attributes, and the proportion of defective modules, making them useful for testing how well the proposed method works in various situations.

Before training the classification models, a feature selection process was used to find and remove redundant, noisy, or irrelevant attributes. This preprocessing is crucial for lowering dataset dimensionality, reducing overfitting, and improving computational efficiency. By concentrating on the most relevant features, the proposed framework promotes better learning and improved predictive performance across different datasets.

4.2 Comparison with Existing Approaches

To assess the effectiveness of the proposed method, a comparative analysis was done against current defect prediction methods found in the literature. Various machine learning classifiers were used, including Random Forest, Support Vector Machine (SVM), Multilayer Perceptron (MLP), Logistic Regression, Bayesian Networks, Decision Stump, and J48 decision tree. These classifiers were chosen because they are widely used and have proven effective in software defect prediction tasks.

4.2.1 Accuracy Improvement:

The experimental results show a consistent increase in prediction accuracy across all datasets when feature

selection was applied. On average, the proposed method achieved an accuracy improvement of 5% to 15% compared to models trained on the complete feature set. This improvement emphasizes the value of removing irrelevant attributes, which often introduce noise and harm model performance. The increased accuracy indicates that the selected features provide more useful information for distinguishing between faulty and non-faulty modules.

4.2.2 Precision and Recall:

Precision and recall values improved significantly across all classifiers, showing better fault detection ability. Higher precision means fewer false positive predictions, which helps cut down on unnecessary debugging efforts. Improved recall also indicates that the proposed method correctly identifies more actual defective modules, reducing the chance of missing faults. The balanced improvement in both precision and recall underscores the effectiveness of the proposed framework.

Another key benefit of this approach is the shorter training time. By using feature selection, the number of input attributes is reduced, which lowers the computational load during model training. This is especially crucial for large-scale software projects, where quick model training and deployment matter. Overall, the comparative analysis indicates that this feature-selection-based framework is more effective than traditional defect prediction methods, providing both higher predictive accuracy and better computational efficiency.

4.3 Analysis of Proposed Approach

A careful analysis was conducted to assess the impact of feature selection on individual classifiers and to gain a better understanding of how the proposed method works.

Figure 18. Datasets Trained and Tested Results

4.4 Key observations

1. Feature selection reduces dataset complexity, shortens training time, and improves prediction reliability.
2. Ensemble methods, especially Random Forest and the Voting Classifier, consistently outperform single classifiers in defect prediction tasks.
3. Neural network models, like MLP, handle non-linear relationships well but gain significantly from feature selection.
4. The Voting Classifier shows the highest overall robustness, combining multiple algorithms to address the weaknesses of individual models.

Overall, the results confirm that the proposed approach, which leverages feature selection, machine learning classifiers, and ensemble voting, offers an accurate, efficient, and practical solution for early detection of software defects. This can significantly enhance software quality, lower maintenance costs, and ensure timely delivery of high-quality software products.

V. CONCLUSION

Software defect prediction is one of the important applications in improving the quality and cost-effectiveness of software development by pinpointing defective components at earlier stages of software development. This paper presented a novel hybrid approach combining feature selection and machine learning classifiers to boost the accuracy of defective component detection. A set of experiments was carried out on five publicly available datasets of the NASA PROMISE repository, namely, CM1, JM1, KC1, KC2, and PC1, with various machine learning classifiers, namely, Logistic, Bayesian Network, Multilayer Perceptron, J48, Random Forest, Support Vector Machine, Lazy IBK, Rule ZeroR, and Decision Stump. From the experimental result, it has been observed that models with feature selection (WFS) perform better than models without feature selection (WOFS) for most of the data sets and classifiers. An average improvement of about 5-8% in accuracy has been observed, and Logistic Regression has attained more than 93% accuracy, and Bayesian Network has shown a remarkable improvement with feature selection. The improvement in accuracy using feature selection has been proved to be statistically significant using a paired two-tailed t-test. In the current investigation, the findings indicate the significance of feature

selection in software defect predictions. It helps minimize unnecessary attributes while also promoting the generalization capability of the model. The method has proven useful in effective testing resource allocation, reduced maintenance costs, along with the development of quality software. The future study could focus on metaheuristic techniques for feature selection, deep learning approaches, and ensemble classifiers along with resampling methods to overcome the issues of data imbalance and to gain higher accuracy in the prediction process.

REFERENCES

[1] M. A. Memon, M.-U.-R. Magsi, M. Memon, and S. Hyder, “Defects prediction and prevention approaches for quality software development,” International Journal of Advanced Computer Science and Applications, vol. 9, no. 8, pp. 451-457

[2] M. Gayathri and A. Sudha, “Software defect prediction system using multilayer perceptron neural network with data mining,” International Journal of Recent Technology and Engineering, vol. 3, no. 2, pp. 2277–3878, 2014.

[3] R. Malhotra, L. Bahl, S. Sehgal, and P. Priya, "Empirical comparison of machine learning algorithms for bug prediction in open-source software," in Proc. International Conference on Big Data Analytics and Computational Intelligence (ICBDAC), pp. 40-45, 2017.

[4] M. S. Rawat and S. K. Dubey, “Software defect prediction models for quality improvement: A literature study,” International Journal of Computer Science Issues, vol. 9, 2012, pp. 288–296.

[5] Sachdev I. Singh, “A survey on data mining techniques in software engineering,” International Journal of Research in IT, Management and Engineering, vol.6, no.3, pp.30-34,2016.

[6] N. Kalaivani & R. Beena, "Overview of software defect prediction using machine learning algorithms," International Journal of Pure and Applied Mathematics, vol. 118, pp. 3863-3873, 2018

[7] M. Dhiauddin and S. Ibrahim, “A prediction model for system testing defects using regression analysis,” International Journal of Soft Computing and Software Engineering, vol. 2, no. 7, pp. 55-68, 2012.

[8] R. Malhotra and A. Jain, “Fault prediction using statistical and machine learning methods for improving software quality,” Journal of Information Processing Systems, vol. 8, no. 2, pp. 241–262, 2012.

[9] A. Hammouri, M. Hammad, M. Alnabhan, F. Alsarayrah, “Software bug prediction using machine learning approach,", International Journal of Advanced Computer Science and Applications, Volume 9, issue 2, 2018, pp 78-83

[10] M. M. A. Abdallah & M. M. Alrifaee, “Towards a new framework of program quality measurement based on programming language standards,” *International Journal of Engineering & Technology*, vol. 7, pp. 1-3, 2018.

[11] I. H. Sarkar, “Machine learning: Algorithms, real-world applications and research directions,” SN Computer Science, vol. 2, p. 160,

[12] P. Langley and J. G. Carbonell, “Approaches to machine learning,” Journal of the American Society for Information Science, vol. 35, no. 5, pp. 306–316, 198

[13] T. G. Dietterich, “Machine learning in ecosystem informatics and sustainability,” in Proc. 21st International Joint Conference on Artificial Intelligence, 2009.

[14] A. Chug and S. Dhall, “Software defect prediction using supervised and unsupervised learning algorithms,” in Proc. 4th International Conference on Confluence, 2013, pp. 173–179

[15] [15] D. Gray et al., "Reflections on the NASA MDP data sets," IET Software, vol. 6, no. 6, pp. 549-558

[16] M. M. Askari and V. K. Bardsiri, “Software defect prediction using a high-performance neural network,” International Journal of Software Engineering & Applications, vol. 8, pp. 177–188, 2014.

[17] P. Mandal and A. S. Ami, "Selecting best attributes for software defect prediction," in Proc. IEEE International WIE Conference on Electrical and Computer Engineering, 2015, pp. 110-113.

[18] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," Neurocomputing, vol. 300, pp. 70

[19] J. Petrić et al., "The jinx on the NASA software defect data sets," in Proc. 20th International Conference on Evaluation and Assessment in Software Engineering, 2016.

[20] A. Alsaeedi and M. Z. Khan, "Software defect prediction using supervised machine learning and ensemble techniques: A comparative study," Journal of Software Engineering and Applications, vol. 12, no. 5, pp. 85-100, 2019.

[21] M. M. Askari and V. K. Bardsiri, "Software defect prediction using a high performance neural network," International Journal of Software Engineering and Applications, vol. 8, pp. 177-188, Jan. 2014.

[22] M. C. M. Prasad, L. F. Florence, and A. Arya, "A study on software metrics based software defect prediction using data mining and machine learning techniques," International Journal of Database Theory and Application, vol. 8, no. 3, pp. 179-190, Jun. 2015.

[23] D. ChandraYadav and S. Pal, "Software bug detection using data mining," International Journal of Computer Applications, vol. 115, no. 15, pp. 21-25, Apr. 2015.

[24] D. Kumar and V. H. S. Shukla, "A defect prediction model for software product based on ANFIS," International Journal of Scientific Research and Devices, vol. 3, no. 10, pp. 1024-1028, 2016.

[25] P. Mandal and A. S. Ami, "Selecting best attributes for software defect prediction," in Proc. IEEE International WIE Conference on Electrical and Computer Engineering, Dec. 2015, pp. 110-113.

[26] N. Li, M. Shepperd, and Y. Guo, "A systematic review of unsupervised learning techniques for software defect prediction," Information and Software Technology, vol. 122, Jun. 2020, Art. no. 106287.

[27] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," Neurocomputing, vol. 300, pp. 70-79, Jul. 2018.

[28] K. P. Singh, N. Basant, and S. Gupta, "Support vector machines in water quality management," Analytica Chimica Acta, vol. 703, no. 2, pp. 152-162, Oct. 2011.

[29] K. L. Chiew, C. L. Tan, K. Wong, K. S. C. Yong, and W. K. Tiong, "A new hybrid ensemble feature selection framework for machine learning-based phishing detection system," Information Sciences, vol. 484, pp. 153-166, May 2019.

[30] S. P. Potharaju and M. Sreedevi, "A novel subset feature selection framework for increasing the classification performance of SONAR targets," Procedia Computer Science, vol. 125, pp. 902-909, Jan. 2018.

[31] J. Petrić, D. Bowes, T. Hall, B. Christianson, and N. Baddoo, "The jinx on the NASA software defect data sets," in Proc. 20th International Conference on Evaluation and Assessment in Software Engineering, Jun. 2016, pp. 1–12.

[32] J. A. Wass, "WEKA machine learning workbench," Scientific Computing, vol. 24, pp. 1–4, Jan. 2007.

[33] A. Ahmad, "Use of minitab statistical analysis software in engineering technology," in Proc. ASEE Annual Conference and Exposition, 2019, pp. 1–10.

[34] L. Niu, "A review of the application of logistic regression in educational research: Common issues, implications, and suggestions," Educational Review, vol. 72, no. 1, pp. 1–27, 2018.

[35] R. R. Bouckaert, "Bayesian network classifiers in WEKA for version 3-5-7," Artificial Intelligence Tools, vol. 11, no. 3, pp. 369–387, 2008.

[36] G. Kaur and A. Chhabra, "Improved J48 classification algorithm for the prediction of diabetes," International Journal of Computer Applications, vol. 98, no. 22, pp. 13–17,

[37] Sterbenz et al, "Redundancy, diversity, and connectivity to achieve multilevel network resilience, survivability, and disruption tolerance," Telecommunication Systems 56 (2014) 17–31, doi: 10.1007/s11235-

[38] P. W. Wang and C. J. Lin, "Support vector machines," IEEE Intelligent Systems and Applications, vol. 13, no. 4, pp. 18–28, Jul. 2014

[39] L. Breiman, "Random forests," _Machine Learning_, vol. 45, pp. 5–32,

[40] S. Mwanjelemwagha, M. Muthoni, and P. Ochieng, "Comparison of nearest neighbor (IBK), regression by discretization and isotonic regression classification algorithms for precipitation classes prediction," International

Journal of Computer Applications, vol. 96, no. 21, pp. 44–48, Jun.

[41] J. Novakovic and A. S. S. V. Ilic, "Experimental study of using the K-nearest neighbour classifier with filter methods," in Proc. Conference on Computer Science and Technology, 2016, pp. 91-99.

[42] L. Devasena, "Effectiveness analysis of ZeroR, RIDOR and PART classifiers for credit risk appraisal," International Journal of Advanced Computer Science and Technology, vol. 3, pp. 6-11, May 2014.

[43] A. Kaur, P. S. Sandhu, and A. S. Bra, "Early software fault prediction using real time defect data," in Proc. 2nd International Conference on Machine Vision, 2009, pp. 242-245.

[44] R. Jayanthi and L. Florence, "Software defect prediction techniques using metrics based on neural network classifier," Cluster Computing, vol. 22, no. S1, pp. 77-88, Jan. 2019.

[45] S. Aleem, L. F. Capretz, and F. Ahmed, "Benchmarking machine learning techniques for software defect detection," International Journal of Software Engineering and Applications, vol. 6, no. 3, pp. 11-23, May 2015.

[46] M. Khanum, T. Mahboob, W. Imtiaz, H. A. Ghafoor, and R. Sehar, "A survey on unsupervised machine learning algorithms for automation, classification and maintenance," International Journal of Computer Applications, vol. 119, no. 13, pp. 34-39, Jun. 2015.

[47] C. Manjula and L. Florence, "Deep neural network-based hybrid approach for software defect prediction using software metrics," Cluster Computing, vol. 22, no. S4, pp. 9847-9863, Jul. 2019.

[48] A. Al-Nusirat, F. Hanandeh, M. K. Kharabsheh, M. Al-Ayyoub, and N. Al-Dhfairi, "Dynamic detection of software defects using supervised learning techniques," International Journal of Communication Networks and Information Security, vol. 11, no. 1, pp. 185-191, Apr. 2022.

[49] E. Naresh and S. P. Shankar, "Comparative analysis of the various data mining techniques for defect prediction using the NASA MDP datasets for better quality of the software product," Advances in Computer Science and Technology, vol. 10, no. 7, pp. 2005-2017, 2017.

[50] A. Alsaeedi and M. Z. Khan, "Software defect prediction using supervised machine learning and ensemble techniques: A comparative study," Journal of Software Engineering and Applications, vol. 12, no. 5, pp. 85-100, 2019.

[51] A. Potochnik, M. Colombo, and C. Wright, "Statistics and probability," Recipes for Science, vol. 10, pp. 167–206, Aug. 2018, doi: 10.4324/9781315686875-6.

[52] M. Liu et al., "DIG: A turnkey library for diving into graph deep learning research," Journal of Machine Learning Research, vol. 22, no. 1, pp. 10873–10881, 2021.

[53] J. Isabona et al., "Development of a multilayer perceptron neural network for optimal predictive modeling in urban microcellular radio environments," Applied Sciences, vol. 12, no. 11, p. 5713, Jun. 2022, doi: 10.3390/app12115713.

[54] J. M. Álvarez-Alvarado et al., "Hybrid techniques to predict solar radiation using support vector machine and search optimization algorithms: A review," Applied Sciences, vol. 11, no. 3, p. 1044, Jan. 2021.

[55] Z. Zhong, S. Lv, and K. Shi, "A new method of time-series event prediction based on sequence labeling," Applied Sciences, vol. 13, no. 9, p. 5329, Apr. 2023.

[56] M. R. Camana, C. E. Garcia, T. Hwang, and I. Koo, "A REM update methodology based on clustering and random forest," Applied Sciences, vol. 13, no. 9, p. 5362, Apr. 2023.

[57] M. Pavana, L. Pushpa, and A. Parkavi, "Software fault prediction using machine learning algorithms," in Proc. International Conference on Advances in Electrical and Computer Technology, 2022, pp. 185–197, doi: 10.1007/978-981-19-1111-8_16.

[58] K. Sekaran and L. Annabel, "A deep learning-based model for defect prediction in intra-project software," in Proc. 7th International Conference on Trends in Electronics and Informatics, 2023, pp. 1148–1155, doi: 10.1109/ICOEI56765.2023.10126014.

[59] R. Malhotra and Y. Singh, "On the applicability of machine learning techniques for object-oriented software fault prediction," Software Engineering: An International Journal, vol. 1, no. 1, pp. 24–37, 2011.

[60] L.-Q. Chen, C. Wang, and S.-L. Song, "Software defect prediction based on nested-stacking and heterogeneous feature selection,"

[61] R. Bahaweres, F. Agustian, I. Hermadi, A. Suroso, and Y. Arkeman, "Software defect prediction using neural network based SMOTE," in Proc. 7th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), 2021.

[62] Y. Qiu, X. Chen, and J. Zheng, "Automatic feature exploration for software defect prediction," Journal of Systems and Software, vol. 171, 2021.

[63] Y. Qiao and J. Wang, "Effort-aware defect prediction using feedforward neural networks," Journal of Systems and Software, vol. 162, 2020.

[64] C. Ding et al., "Minimum redundancy feature selection from microarray gene expression data," Journal of Bioinformatics and Computational Biology, vol. 3, no. 2, pp. 185–205, 2005.

[65] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, University of Waikato, 1999.

[66] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," Journal of Machine Learning Research, vol. 5, pp. 1205–1224, 2004.

[67] I. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of ReliefF and RReliefF," Machine Learning, vol. 53, pp. 23-69, 2003.

[68] E. Karabulut, S. S. Işık, and A. A. Karabulut, "A comparative study on the effect of feature selection on classification accuracy," Expert Systems with Applications, vol. 39, no. 6, pp. 5932-5943, 2012.

[69] Q. M. Al-Tashi et al., "Approaches to multi-objective feature selection: A systematic literature review," IEEE Access, vol. 8, pp. 126076-126096, 2020.

[70] I. Fakhouri, D. Kocadag, O. Ozkan, and R. Ravishanker, "Feature selection approaches for machine learning classifiers on yearly credit scoring data," Procedia Computer Science, vol. 163, pp. 441-448, 2019.

[71] S. S. Ahmadi et al., "Efficient feature selection for intrusion detection systems," Journal of Information Security and Applications, vol. 47, pp. 1-12, 2019.

[72] H. Malhotra et al., "Intrusion detection using machine learning and feature selection," Procedia Computer Science, vol. 132, pp. 130-138, 2018.

[73] E. Jaw et al., "Feature selection and ensemble-based intrusion detection system: An efficient and comprehensive approach," Computers & Security, vol. 102, 2021.

[74] S. A. A. Shah Hafiz Muhammad Shabbir, "A comparative study of feature selection approaches: 2016-2020," IEEE Access, vol. 8, pp. 188874-188889, 2020.

[75] D. R. Patil and T. M. Pattewar, "Majority voting and feature selection-based network.