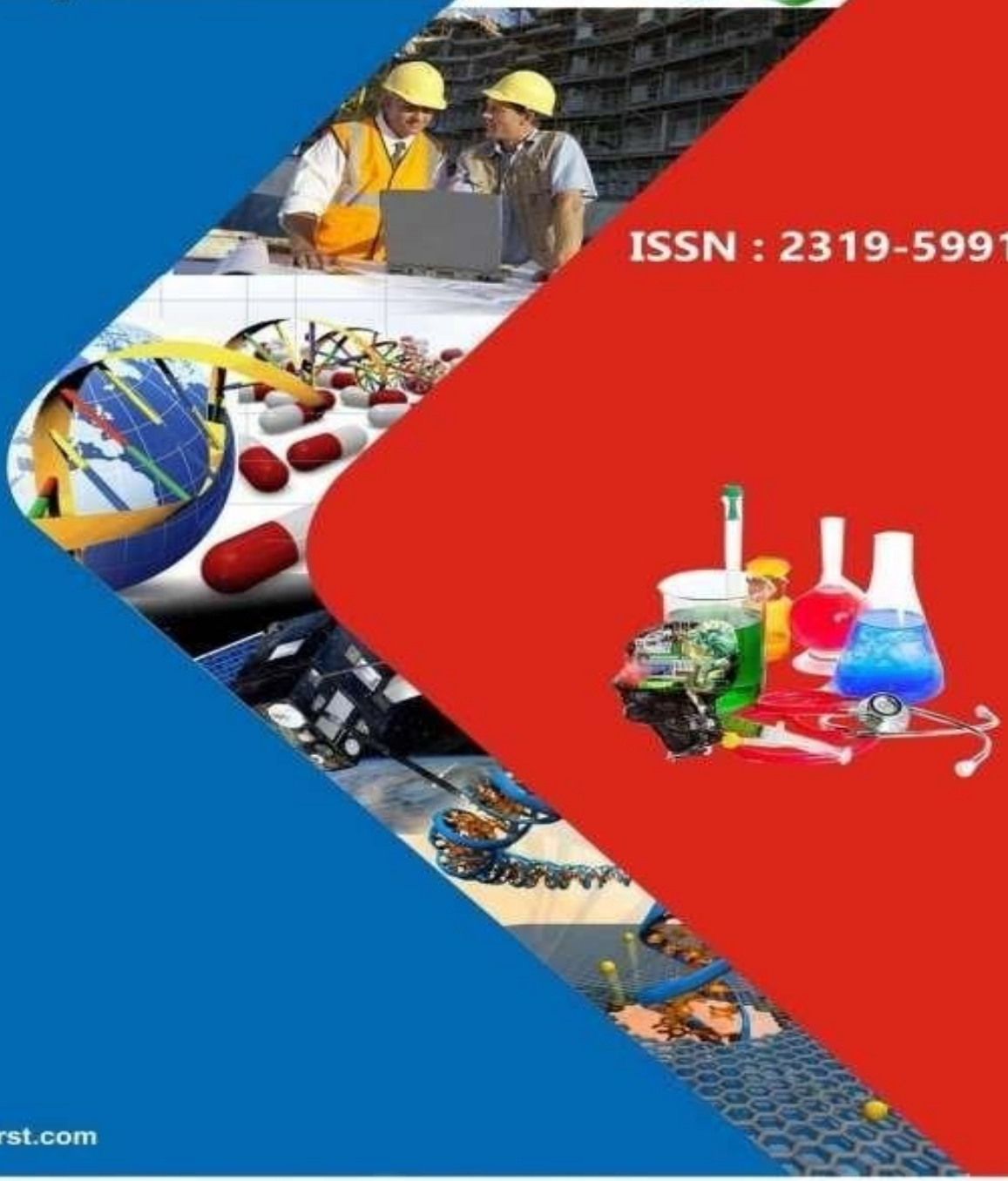
International Journal of Engineering Research and Science & Technology

IJERST

ISSN : 2319-5991



www.ijerst.com

Email: editor@ijerst.com or editor.ijerst@gmail.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

https://doi.org/10.62643/ijerst.2025.v21.i2.pp1050-1061 Vol. 21, Issue 2, 2025

A MACHINE LEARNING-BASED FRAMEWORK FOR IMPROVED SOFTWARE DEFECT PREDICTION

[1]*Lokineni Tejeshwar*, *MCA Student, Department of MCA*

[2] *K Samson Paul, M.Tech, (Ph.D), Assistant Professor, Department of MCA*

[12]*Dr KV Subba Reddy Institute of Technology, Dupadu, Kurnool*

ABSTRACT

Defect prediction is one of the active areas in the software engineering field. Closing the gap between data mining and software engineering is crucial to the product's success. Prior to testing, software defects prediction predicts source code faults. The impact area in software is often explored using techniques for predicting software flaws, including clustering, statistical approaches, mixed algorithms, metrics based on neural networks, black box testing, white box testing, and machine learning. This study's primary contribution is the first use of feature selection to improve machine learning classifiers' performance in predicting faults. This project aims to increase the accuracy of defect prediction in five NASA data sets: CM1, JM1, KC2, KC1, and PC1. The public may access these NASA data sets. In order to achieve a higher defect prediction accuracy than without feature selection (WOFS), this study combines the feature selection technique with machine-learning techniques, including Random Forest, Logistic Regression, Multilayer Perceptron, Bayesian Net, Rule ZeroR, J48, Lazy IBK, Support Vector Machine, Neural Networks, and Decision Stump. The aforementioned classifiers are used, da-ta is refined, and data is preprocessed using the research workbench, a machine-learning program named WEKA (Waikato Environment for Knowledge Analysis). A mini tab statistical tool is used to evaluate statistical studies. The study's findings show that, when compared to WOFS, the accuracy of defects prediction using feature selection (WFS) is improved.

I. INTRODUCTION

A fault in a software system occurs when it performs unexpectedly in response to a client's request. This odd behaviour in software is usually seen by software testers. Errors in the software testing process are detected by software testers. "Irregularities in the software development process that frequently result in software failure and fall short of user expectations" are another word for "software faults." [1]. Any lack of imperfection brought on by a mistake, flaw, or failure in the software development process or final result is called a defect. The paradigm defines "error" as human behaviour that results in improper consequences and "defect" as a judgement that, when used to address a problem, produces erroneous results.

Defective module identification and a range of testing criteria are part of the software defect prediction process. In software engineering, creating a good defect prediction model that can forecast software flaws or malfunctioning modules at an early stage of the software development life cycle is quite challenging. The conventional method of identifying software problems includes re-examining the source code, doing beta testing, integration testing, system testing, and unit testing. Consequently, when software grows in size, complexity, and source code size, these tests become more difficult to perform [2].

Predicting software defects has grown in popularity in recent years. Software quality is

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025
https://doi.org/10.62643/ijerst.2025.v21.i2.pp1050-1061

directly impacted by software defect prediction. The quality of the final product is greatly impacted by defective software modules, which leads to price overruns, delays in the program's completion schedule, and higher maintenance expenses [3].

Defect identification and prevention are the first and second essential techniques of software quality assurance. Defect prevention aims to avoid probable flaws as soon as they arise. Current faults are addressed via defect prediction. Our study attempts to improve software quality by anticipating errors, which is the process of improving software quality via defect prevention (Memoren et al., 1). Algorithm design, algorithm execution evaluation, and software requirement planning mistakes are examples of defect prevention actions [4]. Predicting defects, faults, or flaws in software products in order to estimate the delivered maintenance effort and quality is the fundamental objective of defect prediction prior to the software product's deployment process [5]. Software quality is increased by using the defect prevention strategy [1]. One of the most important steps in developing quality software is error prediction. Because in order to improve overall system performance and guarantee user happiness, software deployment comes before fault prediction. Early mistake or problem detection leads to appropriate resource allocation, which saves money and time while generating high-quality output. Consequently, software defect prediction algorithms actively contribute to the education of individuals on how to assess software and enhance its quality [6].

The defect-prediction technique, which finds defective software modules, makes the software-testing phase more efficient. A number of methods and approaches have yielded exceptional outcomes by using effective defect prediction methodologies or models. Combining a good measuring method with an effective defect prediction model is essential [7]. Predicting software errors enables the deployment of high-quality, user-satisfying software. Code reviews and other software-quality assurance procedures are often used to find software defects [8]. To address problems or concerns with software defect prediction, a variety of techniques have been used. The literature review mentions a variety of defect prediction techniques, but no one approach works for all datasets. as it is dependent upon the features of the dataset. Selecting the most effective fault prediction technique may be challenging. The best method for predicting defects is machine learning [9]. Throughout the SDLC, defect prediction methods (DPT) are used to avert these kinds of software product defects [10].

Machine learning has enabled IT systems to identify various patterns and provide effective solutions based on certain machine learning algorithms and data sets. Furthermore, machine learning results are predicated on past understanding of pertinent content [11]. Based on historical performance, systems may now potentially learn on their own. According to machine learning, computers will be able to identify patterns in data, learn from them, and make decisions with little assistance from humans. The ability to build on past knowledge to learn useful business rule logics and much more makes it an appealing area. What is special about this? The machine learning process is not simple, however. The ability to continuously learn from data and make predictions about the future is what makes machine learning valuable in the twenty-first century. This robust set of models and algorithms is used in many different sectors to improve software performance and find anomalies and trends in data [12].

An individual learning strategy and machine learning work similarly. Machine learning relies on information to make judgements, much as people do [13]. It is defined as the process of estimating the hidden structures

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025
https://doi.org/10.62643/ijerst.2025.v21.i2.pp1050-1061

of a system with the least amount of previous knowledge. Machine learning tasks include classification, grouping, and regression [14]. Different machine-learning techniques may improve the quality and efficiency of software by using distinct machine learning patterns [5]. Additionally, the practice of anticipating software issues or defects early to improve software quality has a significant role in decreasing re-work [9].

There are several benefits of utilising machine learning techniques for software fault prediction. It helps businesses to make well-informed choices regarding software quality, efficiently manage resources, and prioritise testing efforts. Early detection of high-risk regions enables developers to fix any problems before they affect end users, increasing customer satisfaction and lowering maintenance requirements. The authors of this study participate in the testing process to improve the machine learning algorithm's accuracy and better anticipate user errors.

II. LITERATURE SURVEY

"Approaches to defect prediction and prevention for high-quality software development,"
M. A. Memon, S. Hyder, M.-U.-R. Magsi, and M. Memon
Error-free and high-quality application systems are necessary to meet the enterprise's requirement for dispersed and complicated business applications. Regretfully, the majority of written software has flaws that lead to system failure. The development of sensitive or critical applications cannot tolerate such failures. Because of this, creating software that is both high-quality and devoid of defects is crucial. To effectively forecast and eliminate software defects, reduce failures, and enhance software quality, it is critical to better understand and calculate the relationship between these flaws and software failures. In order to ensure high-quality software development, this article reviews methods for predicting and preventing software errors. It also examines the limitations and possibilities of such systems in the creation and upkeep of high-quality products.

"A multilayer perceptron neural network with data mining for software defect prediction,"
A. Sudha and M. Gayathri,
For every software company to provide dependable and high-quality software, fault prediction in software systems is essential. Early in the software life cycle, faults (defects) or fault-proneness of software modules should be anticipated so that further testing may be done on problematic modules. Numerous software measures, such as cyclomatic complexity and lines of code, have been computed and successfully used to failure prediction. Predicting software flaws has also been done using approaches including mixed algorithms, data mining, statistical methods, and machine learning that were based on software measurements. Numerous studies have been conducted using a variety of methodologies to forecast software system problems and fault-proneness. In this study, a comparative analysis is conducted for the modelling of fault-proneness prediction in software systems using an improved machine learning approach based on Multilayer Perceptron Neural Networks. NASA's Metrics Data Program (MDP) provided the software metrics data set utilised in this study.

"Empirical comparison of machine learning algorithms for open source software bug prediction,"
L. Bahl, S. Sehgal, P. Priya, and R. Malhotra,
One of the most active fields of software engineering research is still bug monitoring and analysis. Software practitioners working on big software projects may successfully use bug tracking findings. From requirement analysis until the maintenance phase, when flaws may possibly result in fatalities, the cost of finding and fixing the flaw increases rapidly. Defect data and software metrics may be used as the foundation for predictive model development. Millions of individuals have contributed to open source

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025
https://doi.org/10.62643/ijerst.2025.v21.i2.pp1050-1061

projects, which provide a large dataset for testing. In order to analyse intricate connections and extract valuable information from issues while using the best possible time and resources, a variety of machine learning algorithms have been offered in the literature. To prove that one strategy is better than another, more thorough study comparing various approaches is required. The purpose of this research is to compare 14 machine learning strategies for creating efficient defect prediction models. 1) Developing an automated Java tool to gather OO, inheritance, and other metrics and identify errors in classes taken from open source repositories are the concerns addressed. 2) Using pertinent performance metrics to assess how well predictive models identify problems in classes; 3) Comparing the predictive power of various machine learning approaches using statistical tests; and 4) Validating defect prediction models. The study's findings indicate that, out of all the methods used in the creation of defect prediction models, the Single Layer Perceptron is the most effective. The study's insights may be used in real-world scenarios by software professionals to identify the most effective defect prediction approach and, as a result, allocate resources efficiently.

"A literature study on software defect prediction models for quality improvement,"

S. K. Dubey and M. S. Rawat,

Even with careful planning, thorough documentation, and appropriate process control, certain faults may always emerge throughout software development. These software flaws have the potential to cause quality deterioration, which might be the root cause of failure. Controlling and minimising software engineering flaws is essential in today's cutting-edge competitiveness. But these endeavours come at a cost in terms of money, time, and resources. This study pinpoints the causes and then offers solutions to raise the calibre and productivity of software. The study also demonstrates how the use of different defect prediction algorithms lowers the severity of problems.

"A survey? methods for data mining in software engineering,

Singh, I.

Every step in a typical software development process has a purpose and is dependent upon the others. Every step produces a diverse range of data and is often intricate. We can quantify the influence of each step on the others, find hidden patterns in this data, and get valuable knowledge to enhance the software development process by using data mining tools. Software engineers may improve future software development efforts by using the insights gleaned from the extracted knowledge patterns to anticipate, plan, and understand the project's many complexities. Every step of the development process has a certain objective, thus choosing the finest data mining methods is essential to effectively achieving these objectives. We examine the various data mining approaches in this study and suggest the best ones for every phase of the development process. The benefits of data mining for software development in terms of time, money, resources, dependability, and maintainability are also covered.

"Overview of machine learning algorithms for predicting software defects,"

R. Beena and N. Kalaivani,

The maintenance life cycle and software development are drawn-out procedures. Nonetheless, there may be a significant chance of software flaws. Software performance and dependability are crucial success factors that impact software cost and user happiness. One way to move in this direction is to use machine learning (ML) methods to predict software faults. By putting this strategy into practice early on in the software development process, software performance quality is enhanced and maintenance costs are decreased. Numerous research have used various models and methods to forecast software flaws. Artificial neural networks (ANNs), random forest (RF), random tree (RT), decision table (DT), linear regression (LR), gaussian processes

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025
https://doi.org/10.62643/ijerst.2025.v21.i2.pp1050-1061

(GP), SMOreg, and M5P are among the machine learning (ML) techniques used in this study. A novel approach for predicting software defects in the future is put forward. Historical data serves as the foundation for the fault prediction. The findings demonstrated that software problems might be accurately predicted by using a variety of machine learning methods. The ANN classifier produced the poorest performance results, while the SMOreg classifier produced the best.

III. SYSTEM ANALYSIS &DESIGN

EXISTING SYSTEM

Defect prediction using machine learning, data metrics, and other techniques is the most sought-after research area. Various methods have produced a range of models and interpretations. From 1990 to 2022, a large number of works on the study of software failure prediction were published [15], [16]. Benton and Neil (1999) established size and complexity measures for defect prediction. In the defect prediction process, metrics related to software size and complexity were discussed. Defect prediction is done using multivariate algorithms, software measurements, processing high-quality data, and an evaluation of current approaches. They estimate that there are around 23 errors per thousand lines of code (KLOC).

Vanmali et al. [17] used machine learning (ML) techniques for fault prediction. They employed neural networks to forecast the malfunction. The results of similar comparisons between Neural Network and other techniques showed that Neural Network performed better than other approaches in terms of error detection. They also spoke about using different machine learning approaches. They hypothesised that reactions to classes, LOC, and the lack of decent code are the best markers of programming using the PROMISE data set. They are also doing comparative studies on ensemble techniques for best practices in software.

Biçer et al. [18] looked at the scenario when it is not feasible to fully examine each element of complex frameworks. They discussed the characteristics of effective conformance indicators and looked at a variety of strategies for their development. After putting their assessments of static code measures together, they found that these fault identifiers provide reliable findings across a variety of applications, are affordable to utilise, and can be tailored to the specific needs of the company at hand. They concluded that a better assessment allows for tenfold more financial returns and considered actual criteria for assessing programming expenditures. They show how ML techniques and trustworthy measurements may be used to identify quality indicators early in the product improvement process.

Ramana et al. [19] looked at a few chosen machine learning classification methods to classify different datasets pertaining to liver patients. The effectiveness of a few machine learning classification techniques was evaluated using two datasets. Records for 751 liver patients from Andhra Pradesh, India, with 12 characteristics were included in the initial dataset. The second dataset, which had 345 entries with five characteristics, was made available by the University of California, Irvine (UCI) Machine Learning Repository. The experiments used the WEKA data mining open source machine learning tool or workbench, which has a corei7 CPU and 4 GB of RAM. In this case, machine learning classification methods such as the Naive Bayes classifier, K-Nearest Neighbour Algorithm, Back Propagation Neural Network Algorithm, C4.5, and Support Vector Machines were examined. These algorithms evaluated the results based on four criteria: accuracy, specificity, sensitivity, and precision. When using a selected dataset, K-Nearest Neighbour Algorithm, Support Vector Machines, and Back Propagation all provide better results with all feature set combinations.

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025
https://doi.org/10.62643/ijerst.2025.v21.i2.pp1050-1061

Grey et al. [20] developed models that included the quality of the datasets, which were noisy and had missing values that may have an impact on the results, in order to forecast software problems. The study came to the conclusion that data purification may play a significant role in determining the impact of quality while developing a model and forecasting flaws.

Artificial neural networks were used by Askari and Bardsiri [21] to anticipate defects. Evolutionary methods were combined with SVM learning for prediction in order to create efficient extension machine learning algorithms. NASA Datasets were utilised to evaluate the support vector technique utilising machine learning models. They came to the conclusion that SVM learning performed better than other approaches in terms of accuracy and precision when combined.

Prasad et al. [22] discussed a number of machine learning (ML) classification techniques, such as semi-supervised (Low-density separation, SVM, expectation maximisation, and class mass normalisation), unsupervised (j48, Random forest, Naive Bayes classifier, k-mean clustering, Hierarchical Clustering), and super-supervised (Bayesian Network, Ensemble Method/Random Forests, SVM, Decision Tree) methods. These techniques are the best classification strategy to forecast the defect for high-quality software.

The defect detection method using clustering, association rule, and classification techniques was introduced by Chandra Yadav et al. [23]. The authors describe the Knowledge Discovery in Database (KDD) method and how to apply patterns and extract flaws or defects to identify possible outcomes. The authors came to the conclusion that little testing equipment could be able to detect errors or flaws. In Kumar and Shukla [24], the fuzzy logic information system was used to discover flaws early. The fuzzy inference system of the suggested model makes use of metrics data and additional error data at the function level. Five input variables are used in one input layer and one output layer of this model to check for flaws in the input layer data.

A study on the flaws prediction model was conducted by Hammouri et al. [9]. Supervised machine learning methods, including Naive Bayes, Artificial Neural Networks, confusion matrices, and decision tree algorithms, were used to a variety of datasets in order to predict errors. The experiment used three debugging datasets. Recall, accuracy, RMSE measurements, precision, and F-measure were the components of the experimental results. Furthermore, these experimental findings show that machine learning outperforms other strategies, like the POWM and AR models, in terms of outcomes and prediction model performance.

The techniques for anticipating software mistakes and reducing their impact on the creation of high-quality software are assessed by Memon et al. [1]. The necessity of minimising the reasons of system failure utilising the latest technology is discussed, along with a number of defect prediction techniques (based on pattern, graph mining ASA employing Classifier) and preventative strategies via defect detection and analysis. They also outline the benefits and drawbacks of certain strategies for producing high-quality goods.

Disadvantages

Several solutions were planned to solve the difficulties or issues in software defect prediction. The literature mentions a variety of defect prediction techniques, however no one approach can be used for all datasets. as it is dependent upon the features of the dataset. Selecting a technique to utilise for fault prediction is challenging.

PROPOSED SYSTEM

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

https://doi.org/10.62643/ijerst.2025.v21.i2.pp1050-1061

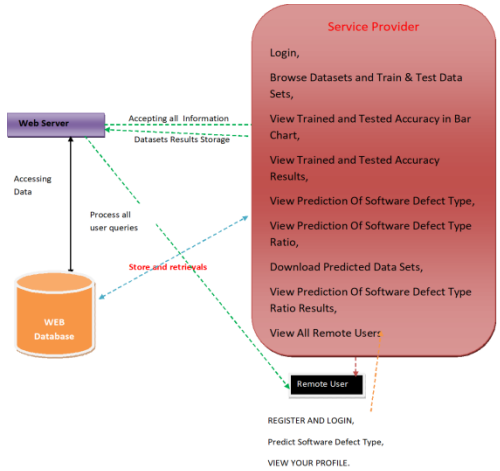
Vol. 21, Issue 2, 2025

This study's primary contribution is the first use of feature selection to improve machine learning classifiers' performance in predicting faults. Improving the accuracy of defect prediction in five data sets is the aim of this work. In order to obtain high defect prediction accuracy in comparison to WOFS, the following machine-learning algorithms were utilised in this study: Random Forest, Logistic Regression, Multi-layer Perceptron, Bayesian Net, Rule ZeroR, J48, Lazy IBK, Support Vector Machine, Neural Networks, and Decision Stump.

Advantages

- There are several benefits of use machine learning techniques for software fault prediction. It helps businesses to make well-informed choices regarding software quality, efficiently manage resources, and prioritise testing efforts.
- Early detection of high-risk regions allows developers to resolve any problems before they affect end users, improving customer happiness and requiring less maintenance.

SYSTEM ARCHITECTURE



IV. IMPLEMENTATION

Modules

Service Provider

The Service Provider must use a working user name and password to log in to this module. Following a successful login, he may do various tasks including browsing datasets and training and testing datasets. View Software Defect Type Prediction, View Software Defect Type Ratio Prediction, Download Predicted Data Sets, Review Trained and Tested Accuracy in Bar Chart, View Trained and Tested Accuracy Results, and View All Remote Users.

View and Authorize Users

The administrator may see a list of all registered users in this module. Here, the administrator may see the user's information, like name, email, and address, and they can also grant the user permissions.

Remote User

A total of n users are present in this module. Before beginning any actions, the user needs register. Following registration, the user's information will be entered into the database. Following a successful registration, he must use his password and authorised user name to log in. The user will do many tasks after successfully logging in, including registering and logging in, predicting the kind of software defect, and seeing their profile.

ALGORITHMS

Logistic regression Classifiers

The relationship between a collection of independent (explanatory) factors and a categorical dependent variable is examined using logistic regression analysis. When the dependent variable simply has two values, like 0 and 1 or Yes and No, the term logistic regression is used. When the dependent variable contains three or more distinct values, such as married, single, divorced, or widowed, the technique is sometimes referred to as multinomial logistic regression. While the dependent variable's data type differs from multiple regression's, the procedure's practical application is comparable.

When it comes to categorical-response variable analysis, logistic regression and discriminant analysis are competitors. Compared to discriminant analysis, many statisticians believe

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025
https://doi.org/10.62643/ijerst.2025.v21.i2.pp1050-1061

that logistic regression is more flexible and appropriate for modelling the majority of scenarios. This is due to the fact that, unlike discriminant analysis, logistic regression does not presume that the independent variables are regularly distributed.

Both binary and multinomial logistic regression are calculated by this software for both category and numerical independent variables. Along with the regression equation, it provides information on likelihood, deviance, odds ratios, confidence limits, and quality of fit. It does a thorough residual analysis that includes diagnostic residual plots and reports. In order to find the optimal regression model with the fewest independent variables, it might conduct an independent variable subset selection search. It offers ROC curves and confidence intervals on expected values to assist in identifying the optimal classification cutoff point. By automatically identifying rows that are not utilised throughout the study, it enables you to confirm your findings.

Naïve Bayes

The supervised learning technique known as the "naive bayes approach" is predicated on the straightforward premise that the existence or lack of a certain class characteristic has no bearing on the existence or nonexistence of any other feature. However, it seems sturdy and effective in spite of this. It performs similarly to other methods of guided learning. Numerous explanations have been put forward in the literature. We emphasise a representation bias-based explanation in this lesson. Along with logistic regression, linear discriminant analysis, and linear SVM (support vector machine), the naive bayes classifier is a linear classifier. The technique used to estimate the classifier's parameters (the learning bias) makes a difference.

Although the Naive Bayes classifier is commonly used in research, practitioners who want to get findings that are useful do not utilise it as often. On the one hand, the researchers discovered that it is very simple to build and apply, that estimating its parameters is simple, that learning occurs quickly even on extremely big datasets, and that, when compared to other methods, its accuracy is rather excellent. The end users, however, do not comprehend the value of such a strategy and do not get a model that is simple to read and implement.

As a consequence, we display the learning process's outcomes in a fresh way. Both the deployment and comprehension of the classifier are simplified. We discuss several theoretical facets of the naive bayes classifier in the first section of this lesson. Next, we use Tanagra to apply the method on a dataset. We contrast the outcomes (the model's parameters) with those from other linear techniques including logistic regression, linear discriminant analysis, and linear support vector machines. We see that the outcomes are quite reliable. This helps to explain why the strategy performs well when compared to others. We employ a variety of tools (Weka 3.6.0, R 2.9.2, Knime 2.1.1, Orange 2.0b, and RapidMiner 4.6.0) on the same dataset in the second section. Above all, we make an effort to comprehend the outcomes.

Random Forest

Random forests, also known as random decision forests, are ensemble learning techniques that build a large number of decision trees during training for tasks like regression and classification. The class chosen by the majority of trees is the random forest's output for classification problems. The mean or average forecast of each individual tree is given back for regression tasks. The tendency of decision trees to overfit to their training set is compensated for by random decision forests. Although random forests are less accurate than gradient enhanced trees, they often perform better than choice trees.

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Vol. 21, Issue 2, 2025

https://doi.org/10.62643/ijerst.2025.v21.i2.pp1050-1061

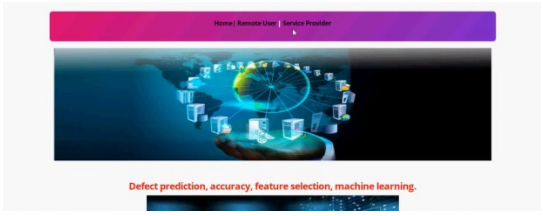
However, their performance may be impacted by data peculiarities.

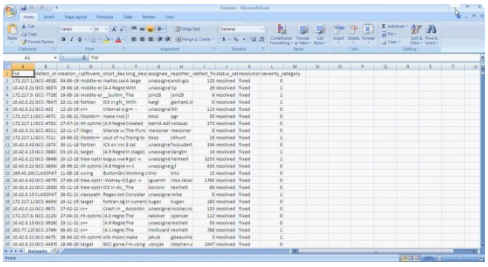
Tin Kam Ho[1] developed the first algorithm for random decision forests in 1995 by using the random subspace technique, which in Ho's definition is a means of putting Eugene Kleinberg's "stochastic discrimination" approach to classification into practice.

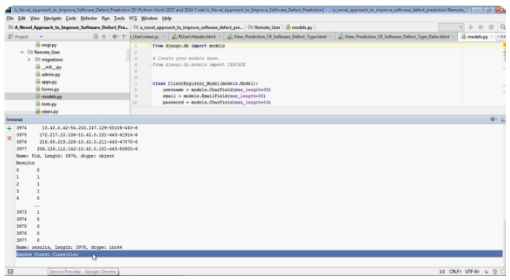
Leo Breiman and Adele Cutler created an algorithm extension and filed for a trademark in 2006 for "Random Forests" (owned by Minitab, Inc. as of 2019).The extension builds a set of decision trees with controlled variance by combining Breiman's "bagging" concept with random feature selection, which was initially proposed by Ho[1] and then separately by Amit and Geman[13].

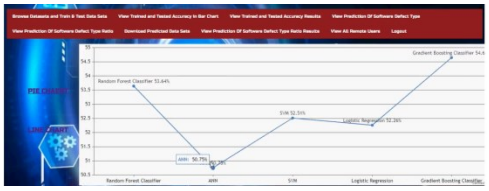
Businesses often employ random forests as "blackbox" models since they need minimal setup and provide accurate forecasts across a variety of inputs.

V. SCREEN SHOTS







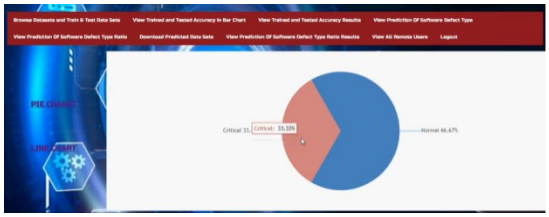


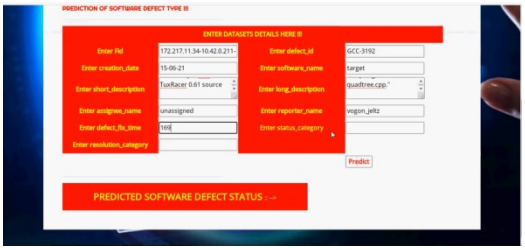
International Journal of Engineering Research and Science & Technology

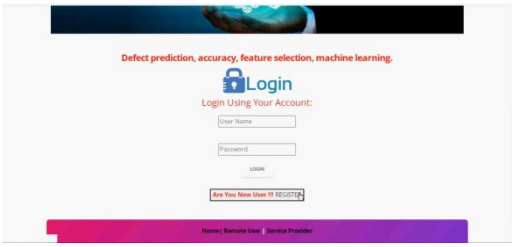
ISSN 2319-5991 www.ijerst.com

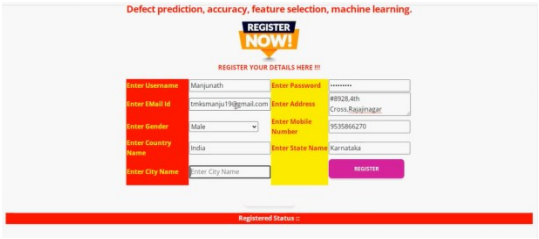
Vol. 21, Issue 2, 2025

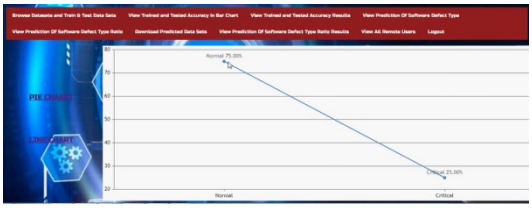
https://doi.org/10.62643/ijerst.2025.v21.i2.pp1050-1061

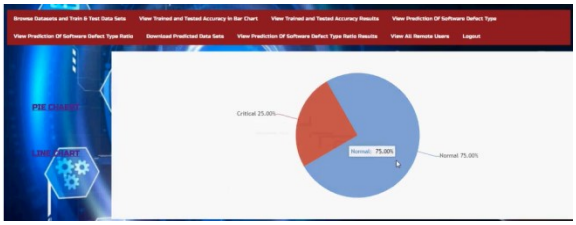


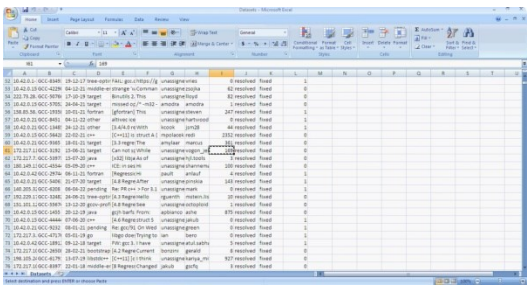












VI. CONCLUSION

One of the areas of software engineering research that is now underway is the prediction of software defects. Software defect prediction forecasts errors in source codes before testing. The conventional techniques for identifying defects include box testing, system testing, and unit testing. Because of this, it becomes difficult to do these tests as a project grows. To investigate software flaws, a variety of approaches are often used, including classification, clustering, mixed algorithms, data mining statistical methods,

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025
https://doi.org/10.62643/ijerst.2025.v21.i2.pp1050-1061

neural networks, and machine learning. Several research approaches have been planned to solve issues or obstacles with software fault prediction. Software flaws may be predicted using a variety of techniques, although no one technique is effective for all datasets. In light of this, it depends on the dataset's primary data. Selecting the best approach for software prediction may be challenging. Software defect prediction is the technique of identifying mistakes in the source code. White in size and complexity of source code, code review, beta testing, Black Box testing, and integrated testing. It is harder and harder to find and remedy defects. Software defect models are useful for these types of problems.

The complexity of software systems is rising in this century of technological progress. Consequently, it is essential to search for defects. If the right technique for identifying software defects is not used, a product may be released with subpar quality. Defect prediction is a critical measure of software quality and dependability, two of its most important characteristics. Five NASA data sets—JM1, CM1, KC1, KC2, and PC1—are analysed in this work to increase the forecast accuracy of software problems. Machine-learning methods such as Bayesian Net, Logistic Regression, Multilayer Perception, Ruler Zero R, J48, Lazy IBK, Support Vector Machine, Neural Networks, Random Forest, and Decision Stump are used to execute feature selection and get the highest degree of accuracy. feature selection method applied to the previously described datasets using the WEKA machine-learning workbench. While the greatest accuracy of the Logistic Regression method is around 93%, the accuracy rate of the Bayesian net algorithm rises by an average of 8% with feature selection. Future studies might uncover other ways to achieve high accuracy, and more datasets could be tested to boost the precision rate. It could be interesting to do further study on how different met heuristic feature selection methods affect the selection of the optimal collection of attributes. One future objective is to examine and evaluate the performance of deep learning algorithms and ensemble classifiers with different resemblance techniques, even if data imbalance is still an issue that negatively impacts performance.

REFERENCES

[1] M. A. Memon, M.-U.-R. Magsi, M. Memon, and S. Hyder, ‘‘Defects prediction and prevention approaches for quality software development,’’*Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 8, pp. 451–457, 2018.

[2] M. Gayathri and A. Sudha, ‘‘Software defect prediction system using multilayer perceptron neural network with data mining,’’ *Int. J. Recent Technol. Eng.*, 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. Int. Conf. Big Data Anal. Comput. Intell. (ICBDAC)*, Andhra Pradesh, India, 2017, pp. 40–45, doi: 10.1109/ICBDACI.2017.8070806.

[4] M. S. Rawat and S. K. Dubey, ‘‘Software defect prediction models for quality improvement: A literature study,’’ *Int. J. Comput. Sci. Issues*, vol. 9,pp. 288–296, Jan. 2012.

[5] I. Singh, ‘‘A survey? Data mining techniques in software engineering,’’ *Int. J. Res. IT, Manage. Eng.*, vol. 6, no. 3, pp. 30–34, Mar. 2016.

[6] N. Kalaivani and R. Beena, ‘‘Overview of software defect prediction using machine learning algorithms,’’ *Int. J. Pure Appl. Math.*, vol. 118,pp. 3863–3873, Feb. 2018.

[7] M. Dhiauddin and S. Ibrahim, ‘‘A prediction model for system testing defects using regression analysis,’’ *Int. J. Soft Comput. Softw. Eng.*, vol. 2,no. 7, pp. 55–68, Jul. 2012.

[8] R. Malhotra and A. Jain, ‘‘Fault prediction using statistical and machine learning methods for improving software quality,’’ *J. Inf. Process. Syst.*,vol. 8, no. 2, pp. 241–262, Jun. 2012.

[9] A. Hammouri, M. Hammad, M. Alnabhan, and F. Alsarayrah, ‘‘Software bug prediction

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025
https://doi.org/10.62643/ijerst.2025.v21.i2.pp1050-1061

using machine learning approach,'' *Int. J. Adv. Comput. Sci.Appl.*, vol. 9, no. 2, pp. 78–83, 2018.
[10] M. M. A. Abdallah and M. M. Alrifaee, ''Towards a new framework of program quality measurement based on programming language standards,''*Int. J. Eng. Technol.*, vol. 7, pp. 1–3, Oct. 2018.
[11] I. H. Sarkar, ''Machine learning: Algorithms, real-world applications and research directions,'' *SN Comput. Sci.*, vol. 2, p. 160, 2021, doi:10.1007/s42979-021-00592-x.
[12] P. Langley and J. G. Carbonell, ''Approaches to machine learning,'' *J. Amer. Soc. Inf. Sci.*, vol. 35, no. 5, pp. 306–316, 1984. [13] T. G. Dietterich, ''Machine learning in ecosystem informatics and sustainability,''in *Proc. 21st Int. Joint Conf. Artif. Intell.*, 2009, pp. 1–5.
[14] S. E. E. Profile, ''Machine learning of hybrid classification models,''in *Proc. Impact Internet Bus. Activities Serbia Worldwide*, 2014,pp. 318–323.
[15] A. Chug and S. Dhall, ''Software defect prediction using supervised learning
algorithm and unsupervised learning algorithm,'' in *Proc. 4th Int. Conf.Confluence Next Gener. Inf. Technol. Summit*, Noida, 2013, pp. 173–179,doi: 10.1049/cp.2013.2313.
[16] A. Shanthini, ''Applying machine learning for fault prediction using software metrics,'' *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 2,pp. 274–278, Jan. 2012.
[17] M. Vanmali, M. Last, and A. Kandel, ''Using a neural network in the software testing process,'' *Int. J. Intell. Syst.*, vol. 17, no. 1, pp. 45–62,2002.
[18] S. Biçer, A. B. Bener, and B. Çağlayan, ''Defect prediction using social network analysis on issue repositories,'' in *Proc. Int. Conf. Softw. Syst.Process*, May 2011, pp. 63–71.
[19] B. Venkata Ramana, M. S. P. Babu, and N. B. Venkateswarlu, ''A critical study of selected classification algorithms for liver disease diagnosis,'' *Int.J. Database Manage. Syst.*, vol. 3, no. 2, pp. 101–114, May 2011.
[20] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, ''Reflections on the NASA MDP data sets,'' *IET Softw.*, vol. 6, no. 6, pp. 549–558,Dec. 2012.
[21] M. M. Askari and V. K. Bardsiri, ''Software defect prediction using a high performance neural network,'' *Int. J. Softw. Eng. Appl.*, 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 learningtechniques,'' *Int. J. Database Theory Appl.*, vol. 8, no. 3, pp. 179–190, Jun. 2015.
[23] D. ChandraYadav and S. Pal, ''Software bug detection using data mining,''*Int. J. Comput. Appl.*, vol. 115, no. 15, pp. 21–25, Apr. 2015.
[24] D. Kumar and V. H. S. Shukla, ''A defect prediction model for softwareproduct based on ANFIS,'' *Int. J. Sci. Res. 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 Int. WIE Conf. Electr. Comput. Eng.*, Dec. 2015,pp. 110–113.