Journal of Science, Computing and Engineering Research (JSCER)
Volume-8, Issue-7, July 2025.
DOI: *https://doi.org/10.46379/jscer.2025.080606*

Error Prediction in Software Using Machine Learning Algorithm

R. Anusuya, Sadhana Yadav

[1]Professor, Department of CSE, Modern Institute of Technology & Research Centre, Alwar, Rajasthan, India

[2]PG Student, Department of CSE, Modern Institute of Technology & Research Centre, Alwar, Rajasthan, India

Article Information

Received : 15 July 2025
Revised : 17 July 2025
Accepted : 21 July 2025
Published : 25 July 2025

Abstract— Software Engineering is a branch of computer science that enables tight communication between system software and training it as per the requirement of the user. We have selected seven distinct algorithms from machine learning techniques and are going to test them using the data sets acquired for NASA public promise repositories. The results of our project enable the users of this software to bag up the defects are selecting the most efficient of given algorithms in doing their further respective tasks, resulting in effective results.

Corresponding Author:
Sadhana Yadav

Keywords: ***Software quality metrics, Software defect prediction, Software fault prediction, Machine learning algorithms***

Citation: R. Anusuya, Sadhana Yadav, "Error Prediction in Software Using Machine Learning Algorithm", Journal of Science, Computing and Engineering Research, 8(07), July 2025.

I. INTRODUCTION

A) Problem Statement:

Now-a-days developing software system is a difficult process which involves planning, analyzing, designing, implementing, test, integrate and maintenance. A software engineer work is developing a system in time with limited budget which is done in planning phase. While doing the development process we can have few defects like not proper design, where the logic is poor, data handling is improper, etc. and these defects cause errors which lead to re-do the work, increasing in development and cost of maintenance. This all are responsible for the decrease in customer satisfaction. In this point of view, faults are grouped on the basis of sternness, corrective and advance Actions are taken as per the sternness defined. The selected machine learning algorithms for comparison are:

a. Multilayer Perceptron (MLP)
b. KNN classifier
c. Guassian Naïve Bayes
d. Decision tree
e. Support Vector Classifier (SVC).
f. Ensemble method

B) Objective:

The Objective of this project is to estimate the defect of software using machine learning algorithms. On training the various ML Algorithms we need to get good accuracy percentage so that the particular algorithm fits the best in order to estimate the defects Support Vector Classifier (SVC) supports both classification as well as regression. It is productive and straight-lined method which is used in classification. For classification it divides two groups by making boundaries between the group of data.

C) Proposed System:

The proposed system includes SVC, Multilayer perceptron, Naive Baye's algorithm, Decision Tree, KNN Classifer, Ensemble method, Functions to solve the class misbalancing problem which causes in the decreasing performance of defect prediction. The dataset has been trained and spitted according to the constraints and using the accuracies has been defined in order to measure the defect estimation capability of various algorithms proposed.

D) Advantages of proposed system:

1. Predicted model is used for evaluating the performance measures.

2. We can apply various datasets in this project. But we are using NASA datasets in our project.

3. Software defects are classified to the extent.

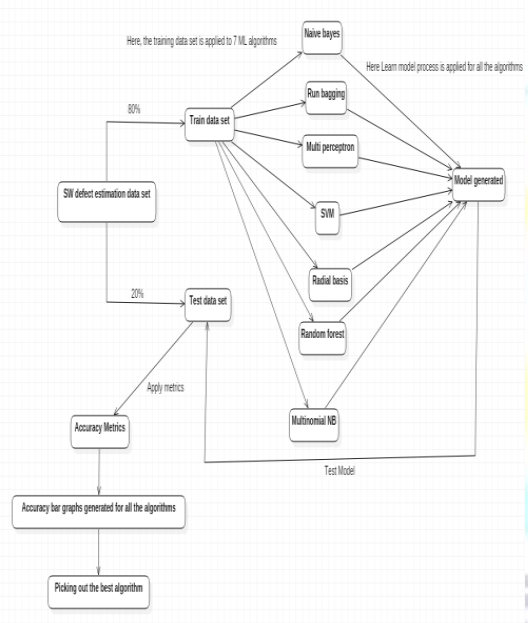
4. Advance measures can be taken on selection of algorithm

5. Provides Better results.

6. Identify defects in the early stage of the project which in turn results in Customer loyalty.

II. SYSTEM DESIGN

A) System Architecture:



III. RESEARCH METHODOLOGY

The proposed methodology for software failure prediction employs a systematic approach, encompassing critical stages essential for comprehensive data analysis and model performance assessment.

A) DATA INFORMATION

The JM1 dataset, a component of the PROMISE repository, is designed for software defect prediction and has been made publicly available by NASA and the NASA Metrics Data Program. Naive Baye's, derived from JM1, has exhibited promising performance, out performing J48 for defect detection, and the dataset has highlighted the nuanced relationship between accuracy and the effectiveness of defect detectors. The JM1 dataset comprises 10,885 instances, each characterized by 22 attributes.

Notably, there are no missing attributes in the dataset. The class distribution indicates that 19.35% of instances are labeled as "false" (modules without reported defects), while 80.65% are labeled as "true" (modules with one or more reported defects).

	loc	v(g)	ev(g)	iv(g)	n	v	l	d	i	e	...	lOCode	lOComment	lOBlank	locCodeAndComment
0	1.1	1.4	1.4	1.4	1.3	1.30	1.30	1.30	1.30	1.30	...	2	2	2	2
1	1.0	1.0	1.0	1.0	1.0	1.00	1.00	1.00	1.00	1.00	...	1	1	1	1
2	72.0	7.0	1.0	6.0	198.0	1134.13	0.05	20.31	55.85	23029.10	...	51	10	8	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10882	42.0	4.0	1.0	2.0	103.0	519.57	0.04	26.40	19.68	13716.72	...	29	1	10	0
10883	10.0	1.0	1.0	1.0	36.0	147.15	0.12	8.44	17.44	1241.57	...	6	0	2	0
10884	19.0	3.0	1.0	1.0	58.0	272.63	0.09	11.57	23.56	3154.67	...	13	0	2	1

10885 rows × 22 columns

Figure: data frame

B) DATA PRE-PROCESSING

In the preprocessing pipeline for software failure prediction using machine learning, several critical steps are undertaken to refine textual data and optimize it for analysis, the standard scaler technique is applied to normalize and scale the data, enhancing its numerical stability. Subsequently, the text is transformed to lowercase to ensure consistency and mitigate case- related variations. To facilitate readability and semantic analysis, punctuation signals are then systematically removed from the text. The final preprocessing step involves lemmatization, wherein words are transformed into their root forms. This not only improves consistency but also aids in further reducing dimensionality while retaining the essential semantic information. The overarching goal of this preprocessing approach is to enhance the quality of the textual data, minimizing noise and ensuring its readiness for subsequent analysis or utilization in machine learning algorithms tailored for software failure prediction

C) EDA

Exploratory Data Analysis (EDA) stands as a pivotal phase in the continuum of data analysis, marked by its systematic exploration, visualization, and comprehension of a dataset. It functions as a fundamental tool, empowering data analysts, researchers, and scientists to unearth pertinent insights, identify patterns, detect anomalies, and formulate hypotheses. EDA serves as a crucial precursor to in-depth studies and informed decision-making. Its significance lies in its dual capability: first, to unveil latent information inherent in the data set,and second, to provide a framework for conducting thorough and comprehensive research.

By cultivating a more comprehensive insight into the data, EDA plays a pivotal role in guiding subsequent stages of analysis, contributing to the formulation of hypotheses, model

selection, and the overall refinement of analytical methodologies.

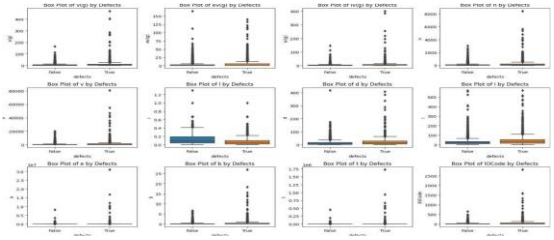


Figure: a box plot depicts the distribution of software defect metrics

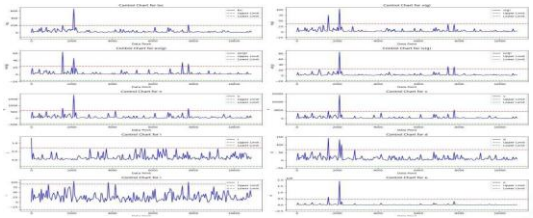


Figure: Informative control charts for the specified software Metrics

D) DATA SPLITTING

In the domain of machine learning for software failure prediction, a widely adopted methodology is the "80:20 data partitioning" strategy, which involves dividing a dataset into training and testing subsets, allocating 80% for training and reserving the remaining 20% for testing. This approach serves as a standard practice due to its effectiveness. By maintaining a clear distinction between training and testing data, this methodology establishes a framework for assessing the credibility of predictive and analytical outcomes. The evaluation process involves contrasting model-generated results with benchmarks derived from the reserved testing subset, ensuring a thorough assessment of the model's capabilities.

E) MODELS USED

The diagnostic approaches for identifying depression. The application of machine learning techniques to the detection of depression involves leveraging specialized algorithms, specifically the Multilayer Perceptron(MLP) ,the Decision Tree Classifier, K Neighbors Classifier, Gaussian Naive Bayes (Gaussian NB), and Support Vector Classification (SVC). These algorithms are adept at analyzing various characteristics and patterns to formulate effective

The Multi-Layer Perceptron (MLP) algorithm is a type of artificial neural network employed in machine learning, specifically for software failure prediction. Comprising an input layer, hidden layers, and an output layer, each layer consists of nodes connected by weighted edges.

MLP, with its versatility, is widely used for tasks like system health monitoring and predictive analytics in software reliability, offering a robust framework to discern intricate patterns and enhance predictive accuracy.

The Decision Tree Classifier is chosen for its ability to discern complex relationships within the data. This characteristic empowers the model to potentially capture nuanced features contributing to depression, enabling a more comprehensive understanding of the underlying factors

The K Neighbors Classifier, on the other hand, excels in identifying individuals with similar feature patterns. By relying on the similarity of examined cases, it becomes proficient in detecting shared characteristics among individuals experiencing depressive episodes. This capability facilitates a more personalized and precise approach to depression diagnosis.

The SVC algorithm is employed due to its efficiency in defining hyper planes within multi dimensional feature space. This attribute makes it a potent method for classifying individuals into depressive and non-depressive categories based on detailed feature patterns. The SVC's ability to create clear boundaries in feature space enhances its accuracy in distinguishing between individuals with and without depression.

Gaussian Naive Bayes is a classification algorithm in machine learning that uses Bayes' theorem and assumes features are normally distributed. It's particularly useful for handling continuous data and is known for its simplicity and effectiveness. This algorithm assumes that each feature is independent of the others, given the class label. Gaussian Naive Baye's assumes that each parameter, also called features or predictors, has an independent capacity of predicting the output variable.

F) Results

By scrutinizing the models' responses to different algorithms, this analysis aims to elucidate the strengths and limitations of each model, thereby informing the selection and optimization of models for enhanced predictive accuracy in the domain of software failure prediction using Machine Learning.

1) Accuracy

In the domain of classification, accuracy is quantitatively defined as the ratio of correctly classified instances to the total size of the dataset, expressed as a percentage. This fundamental metric serves as a rigorous measure of a classification model's effectiveness in accurately assigning data examples to their respective categories.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

2) Loss :

In instances where anticipated outcomes diverge from actual observations, the consequential outcome is often disappointment. This process aims to optimize the model's performance by mitigating discrepancies between predicted and actual outcomes, thereby enhancing its utility and reliability.

$$Loss = -\frac{1}{m}\sum_{i=1}^{m} fi.\log(fi)$$

3) Precision

Precision is a metric that assesses the model's performance in making accurate predictions, specifically measuring how often the model correctly anticipates a favorable outcome. This statistical measure addresses the question of how many times the model accurately predicts positive instances. Mathematically, precisionis expressed as the ratio of true positives to the sum of true positives and false positives

$$Precision = \frac{TP}{TP+FP}$$

4) Recall

The evaluation of model performance hinges on its ability to recall and correctly identify all pertinent data points. Specifically, when posed with the question, "Among all the genuine positive instances, how many did the model accurately predict as positive?" recall furnishes the relevant answer. In essence, while a high recall indicates the model's proficiency in capturing positive instances, a balanced consideration of other metrics is imperative for a comprehensive assessment of its overall performance.

$$Recall = \frac{TP}{TP+FN}$$

5) F1 - Score

The F1-score serves as a consolidated metric that integrates a classifier's recall and precision through the calculation of their harmonic mean. This metric is specifically designed for the comparative evaluation of two classifiers.

$$F1-score = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$

Performance Evaluation of Machine Learning Model

Models	Accuracy	Precision	Recall	F1-score
MLP model	0.93	0.70	0.92	0.79
Gaussian NB	0.98	0.92	0.96	0.94
Decision tree	0.99	1.0	1.0	1.0
SVC	0.97	0.97	0.80	0.87
KNN	0.99	0.98	0.96	0.93
Ensemble	0.99	1.0	1.0	1.0

The table provides a thorough assessment of machine learning models for software defect prediction, using important performance measures. The rows in the table represent individual models, while the columns provide information on several metrics, including accuracy, precision, recall, and F1-score.

The MLP model demonstrates a commendable accuracy of 0.93, suggesting that it consistently makes correct predictions.

Gaussian Naive Bayes (NB) demonstrates exceptional performance in defect prediction, with a high accuracyof0.98 and well-balanced precision, recall, and F1-score of 0.92, 0.96, and 0.94 respectively.

The Decision Tree algorithm demonstrates outstanding prediction capabilities, achieving perfect scores (1.0) in accuracy, precision, recall, and F1-score, across all measures.

The Support Vector Classifier (SVC) demonstrates a commendable accuracy of 0.97.

KNN has exceptional performance with a high level of accuracy(0.99) and well-balanced precision (0.98), recall (0.96), and F1-score (0.93).

The Ensemble method, which mirrors the Decision Tree, receives perfect scores (1.0) in all measures, confirming its effectiveness in predicting software defects.

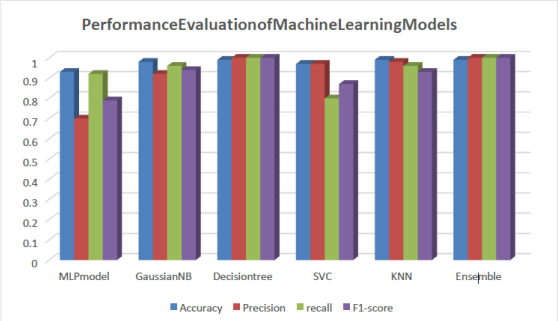


Figure: Performance evaluation graph of machine learning models

The Decision Tree and Ensemble models demonstrate exceptional predictive capabilities with perfect scores across all metrics, making them suitable for tasks where precision, recall ,and overall accuracy are crucial. Gaussian NB excels with a balanced approach, showcasing high accuracy and well-maintained precision, recall, and F1-score, making it a reliable choice for defect prediction. KNN, with its high accuracy and balanced precision and recall, offers a robust solution for identifying software defects.

IV. CONCLUSION

Software defects can have a severe impact on software quality, causing problems for customers and developers. With growing complexities in software designs and technology, manual software detection becomes a challenging and time-consuming task. Thus, automatic software detection has become a hotspot for industrial research in the past couple of years. In this paper, we try to apply machine learning and deep learning to solve this problem. We use datasets provided by the NASA Promise dataset repository and compare the state of the art machine learning algorithms' results. The strengths and weaknesses of each model are highlighted in these measures, helping choose the best model depending on specific goals and trade-offs between precision and recall in software defect prediction tasks. This field still has much scope for improvement. We can think of some novel approaches which use complex deep learning algorithms, and also researchers should focus on more data collection.

REFERENCES

[1]. Y.Cai,Softwarereliabilityengineeringfoundation,Tsinghua universitypress,1995.
[2]. J.Gao,L.Zhang,Z.FengrongandZ.Ye,"ResearchonSoftwareClassification,"inInformationTechnology,Networking,Electronic and Automation ControlConference,2019.
[3]. I.C.Society,"IEEE729-1983-EEEStandardGlossaryofSoftware Engineering Terminology," 1982.
[4]. W.Bi,"ResearchonSoftwareDefectClassificationandAnalysis," ComputerScience,2013.
[5]. X.YangandM.Duan,"ResearchofSoftwareDefectAnalysisTechnology,"Computer Engineering &Software,2018.
[6]. J.CollofelloandB.P.Gosalla,"Anapplicationofcausalanalysistothesoftwaremodificationprocess,"Software:Practiceand Experience, vol. 23, 1993.
[7]. J.W.Horch, Practica l Guide to Software Quality Management Artech House,2003.
[8]. R. Chillarege, I. Bhandari, J. Chaar, M. J. Halliday, D.S.Moebus, B.K.RayandM .-Y.Wong, "Orthogonal Defect Classification- A Concept for In-Process Measurements,"IEEETransactionsonsoftwareEngineering,vol. 18,pp.943-956,1992. S.&.S.E.S.Committee,"IEEE1044-1993-IEEE
[9]. StandardClassificationforSoftwareAnomalies,"IEEE,1993
[10]. X.HuangSoftwarere liability, safety and quality assurance, Electronic Industry Press, 2002.
[11]. Biffl, S. (2000). Using inspection data for defect estimation. IEEE Software, 17(6), 36-43.
[12]. Petersson, H., &Wohlin, C. (1999, November). An empirical study of experience-based software defect content estimation methods. In Proceedings 10th International Symposium on Software Reliability Engineering (Cat. No. PR00443) (pp. 126-135). IEEE.
[13]. Yalçıner, B., &Özdeş, M. (2019, September). 4th International Conference on Computer Science and Engineering IEEE.