International Journal of Research and Technology (IJRT)

International Open-Access, Peer-Reviewed, Refereed, Online Journal

ISSN (Print): 2321-7510 | ISSN (Online): 2321-7529

| An ISO 9001:2015 Certified Journal |

Optimization Accurcay of Predicting Software Defects using Ensemble Machine Learning

Ashwini Sunil[1], Prof. Sugandh Singh[2], Prof. Arjun Rajput[3], Prof. Saurabh Karsoliya[4], Dr. Surabhi Karsoliya[5]

Department of Computer Science and Engineering

Technocrats Institute of Technology, Bhopal, India

Abstract

Software defect prediction plays a vital role in improving software reliability and reducing development costs by identifying fault-prone modules before deployment. Traditional machine learning techniques such as Decision Trees, Support Vector Machines (SVM), and Naïve Bayes have been widely employed for defect prediction; however, their performance often varies due to the imbalance and complexity of software datasets. This study proposes an ensemble machine learning-based approach to enhance the prediction accuracy and robustness of software defect detection. Ensemble techniques such as Random Forest, Gradient Boosting, AdaBoost, and Voting Classifier are analyzed and optimized using hyperparameter tuning strategies to achieve the best predictive performance. Feature selection techniques are also integrated to minimize noise and computational complexity. Publicly available datasets such as NASA and PROMISE repositories are used for experimental validation. Performance metrics such as accuracy, precision, recall, F1-score, and AUC are used to evaluate model efficiency. The results demonstrate that ensemble learning models significantly outperform individual classifiers in terms of both accuracy and generalization. This research highlights the importance of hybrid ensemble approaches in achieving optimized defect prediction accuracy, contributing to reliable and cost-effective software development processes.

Keywords: Software Defect Prediction, Ensemble Machine Learning, Optimization, Random Forest, Gradient Boosting, Software Quality Assurance.

1. INTRODUCTION

Software Defect Prediction (SDP) plays an important role in reducing the costs of software development and maintaining the high quality of software systems [1]. When there repeatedly exists a software failure in system through time it automatically leads to software defect. Software defect are an error that are introduced by software developer and stakeholders. The main objective of software defect prediction is to improve the quality, minimized cost and time of software products [2]. For a high-performance defect predictor, researchers have been working on the choice of static attributes and effective learning algorithms since the 1990s.

SDP employs software metrics (also referred to as software features or software attributes, such as lines of code (LOC), and change information to predict their defect-proneness to support software testing activities [3]. Software defects have become one of the main reasons for the failure of large engineering projects, leading to huge economic losses in the 21st century. In software quality, various defect prediction techniques have been proposed.

Essentially, such techniques rely on different predictors, including source code metrics (e.g., coupling, cohesion, size). Software defect prediction can help to allocate testing resources efficiently through ranking software modules according to their defects. Existing software defect prediction models that are optimized to predict explicitly the number of defects in a software module might fail to give an accurate order because it is very difficult to predict the exact number of defects in a software module due to noisy data [4].
Each software defect is generated under different conditions and environments, and hence differs in its specific characteristics. A software defect may have an enormous negative effect upon software quality. Therefore, defect prediction is extremely essential in the field of software quality and software reliability. Defect prediction is comparatively a novel research area of software quality engineering [5]. Current defect prediction work focuses on

1) Estimating the number of defects remaining in software systems,
2) Discovering defect associations, and
3) Classifying the defect-proneness of software components, typically into two classes, defect-prone and not defect-prone.

The prediction result can be used as an important measure for the software developer and can be used to control the software process. SDP empirical studies are highly biased with the quality of data and widely suffer from limited generalizations [6, 7]. Defect predictors are expected to help to improve software quality and reduce the costs of delivering those software systems. There is a rapid growth of SDP research after the promise repository was created in 2005. It includes a collection of defect prediction data sets from real-world projects for public use, and allows researchers to build repeatable, comparable models across studies. So far, great research efforts have been devoted to metrics describing code modules and learning algorithms to build predictive models for SDP [8].

2. MACHINE LEARNING

ML proves invaluable, offering reproducible outcomes and the ability to learn from previous computations.

3.1 Supervised Learning

Supervised learning utilizes labeled data for classifying and solving problems, with regression and classification techniques as its two main branches. The regression analysis determines relationships among variables, indicating whether changes in explanatory variables are linked to changes in the dependent variable. In contrast, classification techniques assign objects to specific classes based on predefined criteria. Supervised learning methods represent the predominant approach in ML for predicting HD. These algorithms undergo training using a dataset comprising historical patient information, with each patient possessing a known label indicating the presence or absence of HD.

3.2 Unsupervised Learning

On the other hand, unsupervised learning lacks labeled data and introduces biases about the input's structure. When addressing CVD risk, regression techniques are essential to calculate an individual's risk based on actual numerical values associated with various risk factors (El). In contrast, unsupervised learning algorithms analyze HD data without predefined labels, enabling them to uncover inherent patterns and relationships autonomously.

3.3 Reinforcement Learning

Within this framework, an agent is tasked with performing actions, and its effectiveness is contingent on its ability to comprehend the environment in which these actions occur. The agent maintains an internal state and interacts with the environment to achieve this understanding. A crucial aspect of this learning process involves using a reward function. The agent acquires knowledge about its environment by receiving positive or negative rewards based on its actions. The objective is to maximize positive rewards and minimize negative ones, encouraging the agent to learn and adapt over time. It is noteworthy that in reinforcement learning, there is no obligatory reliance on human experts possessing domain-specific knowledge. Applying this concept to healthcare, particularly in the context of HD management, reinforcement learning could prove valuable. For instance, an intelligent system could adapt its decision-making processes to optimize patient care by continuously learning from the patient's health data and treatment outcomes.

3. SOFTWARE QUALITY AND SD PREDICTION

3.1 Software Quality

In computer science, software pertains to the processing of data by hardware, programs and other data to obtain information. Software has been described as a program developed in segments with source code. It is a set of agents handling complexity, testability, visibility, changeability, conformity, reliability and traceability [9]. Software has become the foundation of modern technology; it constitutes or controls the products and services that human beings rely on for a wide variety of daily activities, from the crucial to the trivial. It therefore refers to measurable characteristics. Juran's defines quality as a product feature that satisfies customer needs, providing customer satisfaction. The American Society for Quality (ASQ) defines quality as the foremost characteristics of the product and it must endure the quantified and inferred needs and the software distributed to the user community should be free of defects and scarcities. Software characteristics are measured with respect to its complexity, cohesion, streaks of code, amount of function points and other factors. Quality needs to be measured which imply to deliver a product designed in a stated way. When we design a product, the quality of the software designed must pertains to the features specified for that product by the designer [10].

Waman believes that a successful project gives rise to customer satisfaction. It was advocated that meeting customer expectations leads to quality. Software quality assurance is achieved through designing test cases and using them as a control measure to ensure desired quality. Quality Assurance of the Software is defined as a list of events intended to appraise the product that will also help us to develop and maintain the software [11, 12].

3.2 Software Defect Prediction

The error in the algorithm or in the software program will not permit the software product to satisfy the user requirements and because of that, the user expectations and the software requirement standards are not maintained by the product and is also called as software defect. The error in the software sometimes produce unexpected outcome and cause software malfunctioning too. There are several ways to define the defects produced by the software: -

- The defect or the bug in the application may be caused or created due to some mistake of the programmer.

International Journal of Research and Technology (IJRT)

International Open-Access, Peer-Reviewed, Refereed, Online Journal

ISSN (Print): 2321-7510 | ISSN (Online): 2321-7529

| An ISO 9001:2015 Certified Journal |

- If there is a deviation in the expected result produced by the software and it is not producing the result specified or defined in Specification document, then it is considered as a defect.
- Failing to satisfy the end user expectations is considered as defect in the software and it is mainly due to the bugs arising in the program or the methods used when the product is developed.

The foremost thing when developing a piece of software is to produce it with great eminence and with high excellence. Superiority of the software is measured by the degree to which the piece of code meets the requirements specified in the requirement specification document. [13].

4. SIMULATION RESULTS

Simulation Parameter

The accuracy of each fold determines how well the model has learned from the training data and how accurately it can predict new data. If the accuracy of a fold is high, it indicates that the model has successfully learned the underlying patterns in the data and can make accurate prediction. So, the accuracy can be measured according to Eq. 1

$$Accurancy = \frac{TN + TP}{TN + TP + FN + FP} \tag{1}$$

For a diabetes classification problem, its measures include Precision-Recall and accuracy. The formula to derive these measures is given in Eq. 2 and Eq. 3.

$$\Pr ecision = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Re}call = \frac{TP}{TP + FN} \tag{3}$$

In this test case, we considered other standard classification scheme such as VC. JM1 dataset are used and defective and non-defective classes.

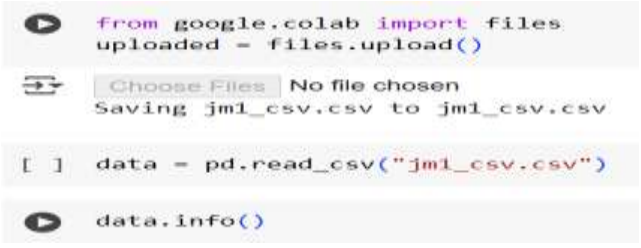

Figure 1: Dataset

International Journal of Research and Technology (IJRT)

International Open-Access, Peer-Reviewed, Refereed, Online Journal

ISSN (Print): 2321-7510 | ISSN (Online): 2321-7529

| An ISO 9001:2015 Certified Journal |

```
Data columns (total 22 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   loc                 11353 non-null  float64
 1   v(g)                11353 non-null  float64
 2   ev(g)               11353 non-null  float64
 3   iv(g)               11353 non-null  float64
 4   n                   11353 non-null  float64
 5   v                   11353 non-null  float64
 6   l                   11353 non-null  float64
 7   d                   11353 non-null  float64
 8   i                   11353 non-null  float64
 9   e                   11353 non-null  float64
 10  b                   11353 non-null  float64
 11  t                   11353 non-null  float64
 12  lOCode              11353 non-null  int64
 13  lOComment           11353 non-null  int64
 14  lOBlank             11353 non-null  int64
 15  locCodeAndComment   11353 non-null  int64
 16  uniq_Op             11346 non-null  float64
 17  uniq_Opnd           11346 non-null  float64
 18  total_Op            11346 non-null  float64
 19  total_Opnd          11346 non-null  float64
 20  branchCount         11346 non-null  float64
 21  defects             11353 non-null  bool
dtypes: bool(1), float64(17), int64(4)
memory usage: 1.8 MB
```

Fig. 2: Data Import

	loc	v(g)	ev(g)	iv(g)	n	v	l	d	i	e	...	lOCode	lOComment
0	1.1	1.4	1.4	1.4	1.3	1.30	1.30	1.30	1.30	1.30	...	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
2	72.0	7.0	1.0	6.0	198.0	1134.13	0.05	20.31	55.85	23029.10	...	51	10
3	190.0	3.0	1.0	3.0	600.0	4348.76	0.06	17.06	254.87	74202.67	...	129	29
4	37.0	4.0	1.0	4.0	126.0	599.12	0.06	17.19	34.86	10297.30	...	28	1

	loc	v(g)	ev(g)	iv(g)	n	v	l	d	i	e	...	lOCode	lOComment
10880	18.0	4.0	1.0	4.0	52.0	241.48	0.14	7.33	32.93	1770.86	...	13	0
10881	9.0	2.0	1.0	2.0	30.0	129.66	0.12	8.25	15.72	1069.68	...	5	0
10882	42.0	4.0	1.0	2.0	103.0	519.57	0.04	26.40	19.68	13716.72	...	29	1
10883	10.0	1.0	1.0	1.0	36.0	147.15	0.12	8.44	17.44	1241.57	...	6	0
10884	19.0	3.0	1.0	1.0	58.0	272.63	0.09	11.57	23.56	3154.67	...	13	0

Fig. 3: JM1 Dataset

International Journal of Research and Technology (IJRT)

International Open-Access, Peer-Reviewed, Refereed, Online Journal

ISSN (Print): 2321-7510 | ISSN (Online): 2321-7529

| An ISO 9001:2015 Certified Journal |

Finding a better way to divide two groups is really difficult. It is possible for the test findings to overlap with each other. The aforementioned research demonstrates that when the threshold is low at a certain moment, both sensitivity and the true positive percentage will increase. Occasionally, FP will also increase, resulting in a fall in specificity and TN.

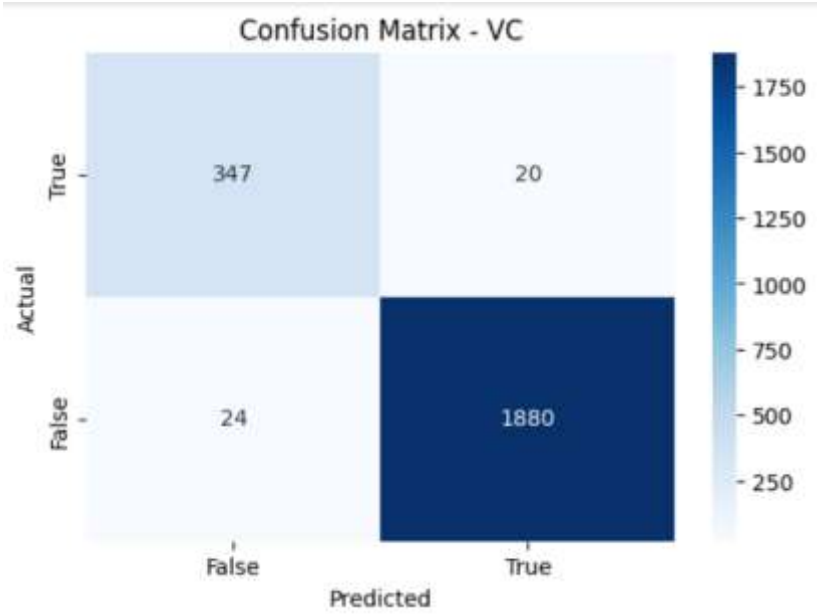

Fig. 4: CM

```
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
#Accuracy score
from sklearn.metrics import accuracy_score
print("ACC: ",accuracy_score(y_pred,y_test))
```

```
              precision    recall  f1-score   support

    Redesign       0.94      0.95      0.94       367
   Succesful       0.99      0.99      0.99      1904

    accuracy                           0.98      2271
   macro avg       0.96      0.97      0.96      2271
weighted avg       0.98      0.98      0.98      2271

[[ 347   20]
 [  24 1880]]
ACC:  0.980625275209159
```

Fig. 5: Simulation Report

5. CONCLUSIONS

The study demonstrated that ensemble machine learning models provide a powerful and reliable solution for software defect prediction. By combining multiple classifiers, ensemble techniques such as Random Forest, Gradient Boosting, AdaBoost, and Voting Classifier effectively capture complex data patterns and minimize the risk of overfitting associated with

individual models. The optimization of hyperparameters and the integration of feature selection methods further improved model accuracy and robustness. Experimental results on benchmark datasets like NASA and PROMISE confirmed that ensemble-based models consistently outperform traditional single classifiers in terms of accuracy, precision, recall, and F1-score.

The findings highlight that ensemble learning not only enhances prediction performance but also contributes to better software quality assurance by enabling early detection of defect-prone modules. This leads to reduced testing effort, lower maintenance costs, and more reliable software products. Future research can focus on hybrid ensemble frameworks that combine deep learning with traditional ensemble models, as well as the application of explainable AI techniques to interpret model decisions. Overall, the optimization accuracy achieved through ensemble machine learning establishes it as an efficient and scalable approach for intelligent software defect prediction systems.

REFERENCES

[1] Iqra Mehmood, Sidra Shahid, Hameed Hussain, Inayat Khan, Shafiq Ahmad, Shahid Rahman, Najeeb Ullah and Shamsul Huda, "A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning", IEEE Access 2023.

[2] L.-Q. Chen, C. Wang, and S.-L. Song, "Software defect prediction based on nested-stacking and heterogeneous feature selection," Complex Intell. Syst., vol. 8, no. 4, pp. 3333–3348, Aug. 2022

[3] M. Pavana, L. Pushpa, and A. Parkavi, "Software fault prediction using machine learning algorithms," in Proc. Int. Conf. Adv. Elect. Comput. Technol., 2022, pp. 185–197

[4] A. Al-Nusirat, F. Hanandeh, M. K. Kharabsheh, M. Al-Ayyoub, and N. Al-Dhfairi, "Dynamic detection of software defects using supervised learning techniques," Int. J. Commun. Netw. Inf. Secur., vol. 11, no. 1, pp. 185–191, Apr. 2022.

[5] R. Bahaweres, F. Agustian, I. Hermadi, A. Suroso, and Y. Arkeman, "Software defect prediction using neural network basedSMOTE," in Proc. 7th Int. Conf. Electr. Eng., Comput. Sci. Informat. (EECSI), Oct. 2020, pp. 71–76

[6] N. Li, M. Shepperd, and Y. Guo, "A systematic review of unsupervised learning techniques for software defect prediction," Inf. Softw. Technol., vol. 122, Jun. 2020, Art. no. 106287.

[7] Y. Qiu, Y. Liu, A. Liu, J. Zhu, and J. Xu, "Automatic feature exploration and an application in defect prediction," IEEE Access, vol. 7, pp. 112097–112112, 2019.

[8] A. Alsaeedi and M. Z. Khan, "Software defect prediction using supervised machine learning and ensemble techniques: A comparative study," J. Softw. Eng. Appl., vol. 12, no. 5, pp. 85–100, 2019.

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

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

[11] A. Hammouri, M. Hammad, M. Alnabhan, and F. Alsarayrah, "Software bug prediction using machine learning approach," Int. J. Adv. Comput. Sci. Appl., vol. 9, no. 2, pp. 78–83, 2018.

[12] 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.

[13] 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.

[14] 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,’’ Adv. Comput. Sci. Technol., vol. 10, no. 7, pp. 2005–2017, 2017.

[15] D. Kumar and V. H. S. Shukla, ‘‘A defect prediction model for software product based on ANFIS,’’ Int. J. Sci. Res. Devices vol. 3, no. 10, pp. 1024–1028, 2016.

[16] 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.