International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7659
***Article Received:** 05 June 2023 **Revised:** 30 July 2023 **Accepted:** 15 August 2023*

A New Improved Prediction of Software Defects Using Machine Learning-based Boosting Techniques with NASA Dataset

Jayanti Goyal[1], Ripu Ranjan Sinha[2]
[1]Research Scholar, Computer Science Department
Rajasthan Technical University (RTU), Kota
goyal.jayanti@gmail.com
[2]Professor, Computer Science, S. S. Jain Subodh P.G. College
Rajasthan Technical University, Kota
drsinhacs@gmail.com

Abstract— Predicting when and where bugs will appear in software may assist improve quality and save on software testing expenses. Predicting bugs in individual modules of software by utilizing machine learning methods. There are, however, two major problems with the software defect prediction dataset: Social stratification (there are many fewer faulty modules than non-defective ones), and noisy characteristics (a result of irrelevant features) that make accurate predictions difficult. The performance of the machine learning model will suffer greatly if these two issues arise. Overfitting will occur, and biassed classification findings will be the end consequence. In this research, we suggest using machine learning approaches to enhance the usefulness of the CatBoost and Gradient Boost classifiers while predicting software flaws. Both the Random Over Sampler and Mutual info classification methods address the class imbalance and feature selection issues inherent in software fault prediction. Eleven datasets from NASA's data repository, "Promise," were utilised in this study. Using 10-fold cross-validation, we classified these 11 datasets and found that our suggested technique outperformed the baseline by a significant margin. The proposed methods have been evaluated based on their abilities to anticipate software defects using the most important indices available: Accuracy, Precision, Recall, F1 score, ROC values, RMSE, MSE, and MAE parameters. For all 11 datasets evaluated, the suggested methods outperform baseline classifiers by a significant margin. We tested our model to other methods of flaw identification and found that it outperformed them all. The computational detection rate of the suggested model is higher than that of conventional models, as shown by the experiments..

Keywords-oftware Defect Prediction, Machine Learning, Class Imbalance, Feature Selection, NASA Promise Dataset, Catboost, Gradient Boost, Random Over Sampler, Cross Validation.

I. INTRODUCTION

The use of software has permeated every aspect of modern life. Software systems have had a significant impact on the economies of today's established and emerging nations, and software products are used in almost every industry and sector[1], from retail to transportation to banking to healthcare to government. Algorithms, procedures, and active modules make up the programme. Allocation of resources and planning [2] including Time, human expertise, computer resources, tools, and infrastructure are all necessities in the design and development of a software system. As long as software plays a significant role, developers will need to think about how often bugs occur. Software failure rates are sometimes rising, even at companies with extensive development expertise.[3][4][5].

When the outcomes of a software programme or product do not correspond to the needs of the end user, we have a software fault. The failures[6][7], unpredictability, or unexpected outcomes induced by these faults are the consequence of either source code or requirement problems. These issues negatively affect software quality and programme reliability and can lead to unnecessary expenditures of time, energy, and money. When problems occur, it takes more time and money to do maintenance. This makes early fault prediction in software a topic of interest for study. Over the course of the past two decades, academics have proposed several prediction models employing various machine learning classifiers.[8][9]. Inappropriately[10][11], Uneven data distribution presents a significant difficulty for the SDP procedure, lowering the quality of the learning model as a result. Due to the asymmetry of the situation, there are fewer malfunctioning modules than there are functional ones. The distribution of classes among them is not an issue if they can be divided into the target classes utilising the available characteristics. Problems arise only when this attribute hinders the efficacy of the algorithms or the resulting models.

The number of successful and unsuccessful stages of a project are used to compile a data set for defect prediction.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7659
***Article Received:** 05 June 2023 **Revised:** 30 July 2023 **Accepted:** 15 August 2023*

Numerous even and unbalanced data sets have been utilised to anticipate the flaw by researchers. Software testing is profoundly affected by the results on data used for training defect prediction models symmetrical and asymmetrical data sets. During the data collecting and processing processes, we considered the wide range of metrics. There are data sets of 19, 40, 42, 63, and 209 metrics. [12],[13]. Public data sets for software defect prediction are abundant and may be found in the PROMISE repository [14], where they have been utilised by several studies. Data for predicting defects have been collected from the past. Many powerful There are now machine learning approaches available and used to the problem of software fault prediction throughout the years.[15]. The defect prediction method splits the data into a "train" and "test" section. The classifier is trained with sample data and then used to make predictions about test-data defects.[12]. The learning approaches, such as supervised learning, have been included into a number of ML techniques [16], [17], and imbalance learning [15], [18][13]. These techniques have historically been the gold standard in software fault prediction.

The primary objective of this study is to compare two approaches to evaluating ML algorithms in the context of improved NASA datasets provided from the open PROMISE repository, with the hope of identifying the most effective class for software fault prediction. When it comes to defect prediction, the results from the two machine learning learners, CatBoost and GredientBoost, are very comparable.

The purpose of this research was to use machine learning-based best classification algorithms to improve the precision of software default prediction on the NASA dataset. The following is a brief overview of our contributions:

- The goal is to conduct a literature review on the subject of Software defect prediction. We describe current Software defect prediction detection research tools, methodologies, and datasets by addressing certain research issues.
- To solove the data imbalacing problem using the Random Over Sampler balancing approach.
- To select the important feature of eleven NASA datasets using Mutual_info_classif techniques.
- To proposes a machine learning approach using CatBoost and Gradient Boost classifiers to predict Software defect.
- Using machine learning approaches, we can improve the precision with which we can forecast Software defects and so lessen the likelihood that we'll make a mistake.
- The goal is to simplify the training and testing processes while improving the accuracy of fault prediction and categorization in software.
- To evaluate the suggested models' efficacy in terms of RMSE, MSE/MAE, ROC-AUC, f1-score, and recall for SDP.

This section presents in-depth domain knowledge, and the remainder of the article is organized as follows: Section 2 is the literature review, Section 3 is the study methodology, Section 4 is the results and analysis, and Section 5 is the conclusion and future work.

II. RELATED WORK

The most popular ML methods used in SDP. Many SDP models employing In-depth domain knowledge are presented in this section, while the rest of the article is structured as follows: The literature review is in Section 2, the study methodology is in Section 3, the findings and analysis are in Section 4, and the conclusion and next steps are in Section 5[19], used K-means clustering to organise the classes into meaningful groups. In addition, they used classification models on some of the characteristics. Optimizing ML models using Particle Swarm Optimization. The models' efficacy was evaluated using a variety of criteria, including measures of performance errors and a confusion matrix; precision, accuracy, recall, the f-measure. All of the ML models and their optimised versions provide optimal outcomes, however, the SVM models and their optimised versions show the greatest performance with accuracy of 99.0% and 99.80%, respectively. Ensemble techniques have an accuracy of 97.60 percent, whereas individual methods have accuracies of 93.7 percent, 93.8 percent, 98.5 percent, 99.50 percent, and 98.80 percent. Our ultimate goal was to increase the precision of previous studies, and we believe that we have accomplished this.

In Deep Singh and Chug, (2017), Due to their frequency and significance in Machine Learning, ANN, PSO, DT, NB, and Linear have all been investigated. KEEL employed k-fold cross-validation to assess the validity of the five methodologies it considered. Datasets were obtained from NASA's openly accessible Promise dataset repository. In this research on defect prediction, we analysed data from seven distinct sources. The accuracy of our classifications utilising 10fold cross-validation on these 7 datasets. In terms of accuracy in defect prediction, Linear Classifier was shown to be the most effective technique [20].

Another, Assim, Obeidat and Hammad, (2020), A wide variety of ML techniques are employed in studies, including The terms "artificial neural networks," "random forest," "random tree," "decision table," "linear regression," and "Gaussian processes" are all terms used in artificial intelligence. sensitivity analysis using mean and variance (SMOreg), and maximum entropy (M5P). A revolutionary defect prediction model is created for the goal of seeing into the future and predicting the likelihood of software bugs. The projected flaws are based on historical data. Results confirmed the practicality of combining multiple ML algorithms for software fault prediction. The SMOreg classifier did better than the ANN classifier in terms of performance [21].

Also, Tadapaneni et al., (2022) In the proposed study, two Several machines learning algorithms, including Naive Bayes, LSTM, and DNN, were presented. The PROMISE dataset is taken into account because of its potential application in binary prediction. It was decided to utilise a classification model for this inquiry because of the binary nature of software problem prediction. Because of this, we use the NB on ML model to test out DNN and LSTM and see how well they do. The DNN algorithm outperformed other approaches in a randomized test aimed to detect software flaws [22].

However, Shen et al., (2022) While machine learning techniques have become increasingly popular for constructing defect prediction models, finding the optimal values for the relevant parameters remains challenging. The problem is addressed by proposing a random forest for software fault prediction that is optimized using Bayesian theory. After the data has been cleaned and organized, The hyperparameters of the random forest model are optimised via a Bayesian technique. Lastly, simulations are validated using the NASA MDP datasets. The experimental findings validate the superior performance of our approach in predicting software defects [23].

According to the research done thus far, many ML approaches have been used, but their accuracy varies between datasets and is generally low. As a result, we aim to boost precision by studying a wide range of ML methods, including feature selection and data balance. The goal of study is to enhance precision in literary analysis.

III. RESEARCH METHODOLOGY

This section provides the research methodology for deepfakes detection. Also, the section first discusses the problem statement and then solves the deepfakes problems.

A. Problem Statement

Improving software quality and dependability is difficult, and one of the biggest obstacles is defect prediction. The challenge in this field is pinpointing the faulty code with precision. There have been several attempts to solve the difficult challenge of fault prediction model development. New developments in Because of developments in machine learning technology, several issues have been resolved. Based on the early failure prediction model, we can state that there is an issue with the dataset's imbalanced class distribution, which can be addressed by machine learning. This project aims to employ machine learning techniques to increase the number of datasets for the underrepresented group in the Promise Data Repository (which is used as a reference for many studies of software defect prediction models). The usage of the Promise Data Repository provided by NASA allows for comparing the results of different research in the same domain. The data thus generated will be used for modeling the ML algorithm to predict faults in the system and then compared.

B. Proposed Methodology

Eleven popular NASA datasets are used to provide predictions about software problems using a variety of machine-learning classification approaches. A Python programme was used to examine ML algorithms. This research used datasets found in NASA's open-access Promise dataset repository. The input dataset is highly imbalance so apply Random Over Sampler. Also, select the important feature with the mutual_info_classif method. The train and test sets of this dataset are evenly matched. The outcomes of classifying eleven datasets were evaluated using 10-fold cross-validation. Two examples of classification strategies are Gradient Boost and CatBoost. The success of deployed categorization techniques may be measured in a variety of ways, including Precision, Recall, F-Measure, Accuracy, and ROC Area. This comprehensive dataset allows for direct comparison and evaluation of any claims made regarding the superiority of a certain method, model, or framework for making predictions. The all-proposed process described in below subsections also shows the whole methodology process in Figure 3.1.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7659
***Article Received:** 05 June 2023 **Revised:** 30 July 2023 **Accepted:** 15 August 2023*

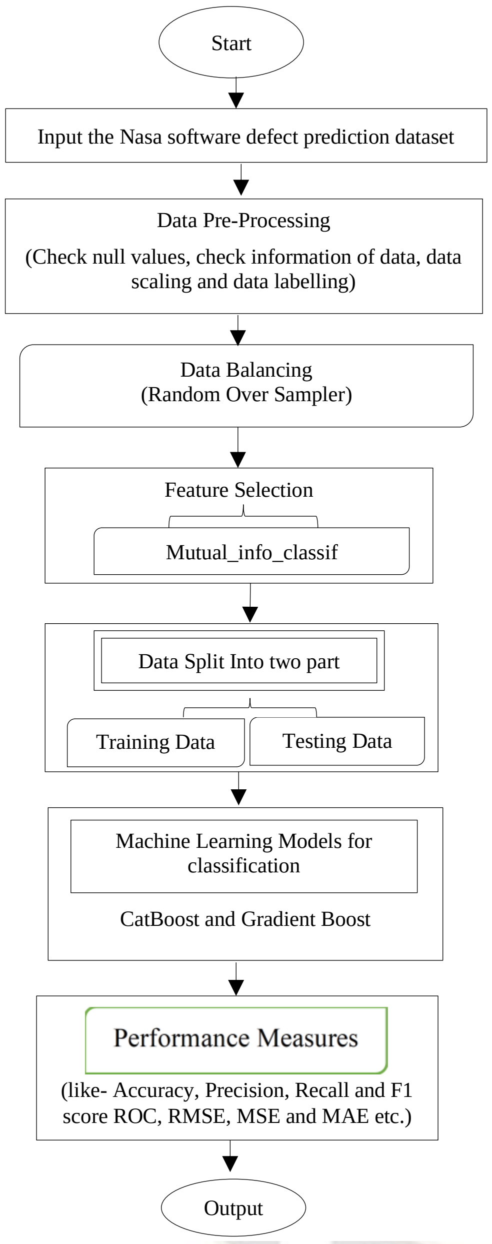

Start
Input the Nasa software defect prediction dataset
Data Pre-Processing
(Check null values, check information of data, data scaling and data labelling)
Data Balancing
(Random Over Sampler)
Feature Selection
Mutual_info_classif
Data Split Into two part
Training Data
Testing Data
Machine Learning Models for classification
CatBoost and Gradient Boost
Performance Measures
(like- Accuracy, Precision, Recall and F1 score ROC, RMSE, MSE and MAE etc.)
Output

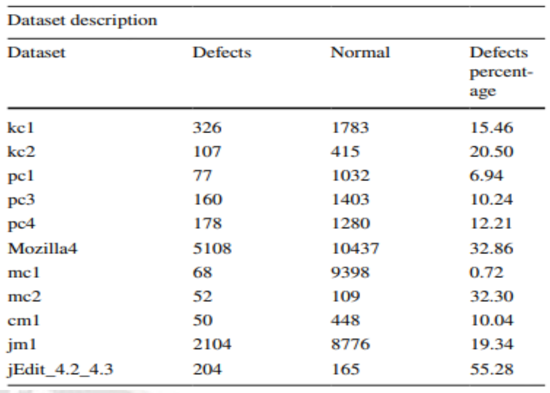
Figure 1. Flow Diagram of the Suggested Methodology

The suggested methodology contains five key phases for the software defects predication including the primary machine learning classification techniques, data collecting, preprocessing, balancing, feature selection, data splitting, and so on. These phases are described below:

1) Data Collection

As you can see in Table 2, we employ eleven datasets from the PROMISE repository (created by Shirabad and Menzies (2005)) to conduct our tests. Table 2 shows that the majority of datasets are very skewed. Since this data is extremely skewed, our goal is to make accurate predictions about the flaws.

TABLE I. DATASETS DESCRIPTION AND LEVEL OF CLASS IMBALANCE



Dataset description

Dataset	Defects	Normal	Defects percent-age
kc1	326	1783	15.46
kc2	107	415	20.50
pc1	77	1032	6.94
pc3	160	1403	10.24
pc4	178	1280	12.21
Mozilla4	5108	10437	32.86
mc1	68	9398	0.72
mc2	52	109	32.30
cm1	50	448	10.04
jm1	2104	8776	19.34
jEdit_4.2_4.3	204	165	55.28

2) Data Preprocessing

The dataset was preprocessed after I had collected it. Data preprocessing [24]is the process of getting ready to feed raw data into a machine-learning model. One aspect of data cleansing is getting the data ready for use. This is a crucial first stage in the data mining process and has been recognised as such for some time. the data was preprocessed in the following manner: null values were checked; information data was checked also applied label encoding was. In this, we applied data preprocessing for the:

Check Null values: In a relational database, if the value of a column is unknown or missing, it is given the null value. When dealing with character and DateTime data types, null is not equivalent to zero or an empty string.

Data scaling: Data scaling[25] is often done before training models using machine learning techniques, during the data pre-processing step. Here we applied Robust Scaler for given data scaling:

- **Robust Scaler:** We may preprocess the data with either RobustScaler() or StandardScaler by removing the outliers. Scikit-learn, a Python machine learning package, provides the RobustScaler class, which implements the robust scaler transform.

3) Feature Selection

After completing the preprocessing phase, I applied feature selection techniques[26]. ML algorithms perform better when their input data is simplified. The most popular method for achieving this goal is to use a feature selection algorithm, which evaluates each feature's quality and ranks them appropriately. The feature selection procedure relies heavily on a feature selection algorithm, which eliminates attributes that are unlikely to improve classification accuracy. One technique to calculate the complexity between dataset features is applying Mutual_info_classif. This technique is an approximation of values to compute the importance of each software feature.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7659
***Article Received:** 05 June 2023 **Revised:** 30 July 2023 **Accepted:** 15 August 2023*

Mutual_info_classif: The mutual information-based feature selection approach is used to identify the most important aspects of a dataset while eliminating those that aren't necessary. The mutual information feature selection algorithm is a filtering technique. mutual info classify. Calculate the mutual information measure between a set of discrete variables. Mutual information (MI) is a positive number that may be used to measure the degree of reliance between two random variables.

4) Data Balancing

The pre-processing stage in the sampling strategy was implemented to achieve data set parity. Using a classifier constructed from previously observed data, Classification is a supervised machine learning technique used to infer an unknown data set's category. In most cases, while developing a classifier, it is assumed that the training data is uniformly distributed. But many data sets are very unbalanced.[27][28]. Datasets with an unbalanced number of samples in one class compared to others are a typical issue in machine learning classification. In the case of supervised ML, when two or more classes are involved, this issue becomes very important. The dataset is skewed since there is an uneven number of samples from each identified class. For the imbalanced software defect dataset applied oversampling techniques[29]. Here we used Random over sampler technique:

Random Over Sampler: The simplest kind of oversampling is creating a duplicate set of training data examples from the minority class at random. Using a random number to enhance the proportion of underrepresented cases in a training set is the purpose of random oversampling.

5) Data Splitting

In this experiment, the dataset was divided randomly such that 80% of the photos and just 20% were used for actual testing. In order to do the sorting task efficiently. we guarantee that the data used for testing are never utilised in training.

6) Classification using the ML model:

The suggested methodology makes use of boosting algorithms grounded on machine learning techniques like CatBoost and Gradient Boost. In this part, we take a quick look at these algorithms.

a) CatBoost Classifier

CatBoost is a training algorithm with fast speed and good classification accuracy. It was the newest classification system that could learn a lot. CatBoost additionally provides a method for converting input attributes to numeric values. However, tiny sample sets of experts, especially samples with multiple inputs' attributes, are not a good fit for the CatBoost discussion approach. Although the combined CatBoost method shows promise as a learning technique for the expert categorization problem, it has to be reshaped for the input function dialogue. [30]. The predictive time for CatBoost, a decision tree gradient boosting method, is negligible. Weather forecasting, autonomous vehicles, digital assistants, and recommendation engines are just some of the many uses for this technology. One-hot encoding is used by default in most configurations of CatBoost for categorical features with a small number of possible values. The term "symmetric tree" is used in CatBoost to denote trees in which the splitting requirement is satisfied by all nodes on the same level of the tree. Instead of catBoost, these trees rely on decision tree categorization, which makes advantage of the scaling feature.

IV. GRADIENT BOOST CLASSIFIER

When it comes to prediction and classification, GB is the most powerful ensemble method. It takes several rather feeble learners and combines them into one powerful predictive model. A decision tree structure is put into effect. It's a tried and true method of filing away all sorts of information. The GB model becomes more and more effective as iterations go. [31]. The GB model uses a stack of relatively weak forecasters, decision trees in this case, to arrive at a final prediction. After the original tree has been fitted to the data, a new version of the data is created that places more weight on predicting observations that were poorly predicted by the tree. In the deployed version, least squares regression was used as the loss function and as the function to be optimized using gradient descent. In order to get a more precise forecast, it is thought that a large number of these rather weak trees should be combined. There are as many estimators as there are trees (number of boosting stages completed) [32].

Algorithm GBM:

- Put a P next to some of the target numbers.
- Find the margin of error for the set targets.
- Error M may be reduced if the weights are revised and updated.
- P(x) = P(x) + Alpha M[x]
- Model learners are analysed and calculated using the loss function F.
- Maintain till you reach the goal of P.

C. Proposed Algorithm

Input: Nasa software defect prediction Dataset	
Output: High prediction accuracy.	

Start:
Install Python simulation tool and Jupyter Notebook as a simulation platform
Import Python Libraries (NumPy, Pandas, Scikit-Learn, Matplotlib, Scrapy, Keras, Seaborn, and sklearn, etc.):

Step 1 Upload the Nasa software defect prediction Dataset from the Kaggle

Step 2	Apply Data Pre-Processing
	• Check null values • Check information of data • Label Encoder
Step 3	Data Balancing
	• Random Over Sampler
Step 4	Feature Selection
	• Mutual_info_classif
Step 5	Dataset Split
	• The data will then be split into training sets (80 percent) & testing sets (20 percent).
Step 6	Classification Models (Machine learning classifiers)
	• CatBoost classifier and • Gradient Boost classifiers focus on high precision of outcomes.
Step 7	By utilizing multiple performance metrics, including precision, accuracy, f1-score, and recall, ROC of the model's performance was assessed.
Step 8	Comparative analysis was done utilizing different machine-learning techniques.
Step 9	Get high accuracy for SDF
Step 10	Finish!!!

V. RESULTS ANALYSIS

In this part, we show the results of our simulation experiments on predicting software artefacts using machine learning. We ran the simulation using a Windows 10 machine equipped with an Intel Core i7-9750H CPU running at 2.6 GHz and 16 GB of RAM. Also used Python [33], was the programming language for software development technologies. This module was once responsible for data analysis and implementation of machine learning models. The Jupyter Notebook [34] included a simulation system as well as several Python libraries[35], such as NumPy, Pandas, Matplotlib, Keras, TensorFlow, and Seaborn, among others. Various specialised performance matrices were employed (described further below). A variety of graphs, metrics, and tables are used to summarise the experiment's outcomes.

D. Dataset Description

As can be seen in Table 2, we use eleven datasets from the PROMISE repository (created by Shirabad and Menzies (2005)) to conduct our tests. The feature selection procedure aims to pick an appropriate feature subset for precise defect prediction by removing redundant and unnecessary characteristics from the dataset. As was previously shown, the most significant or best qualities can have a disproportionate impact on performance, allowing for improvements to be made with less training data. Our findings show that in certain situations, we may minimise the feature set by as much as 60%. (In the pc3 dataset, for instance, there are a total of 37 characteristics, but only 15 make it into the reduced set). To make the classifier more scalable and to lower its processing cost, the features are cut by around 50% on average across all 11 datasets.

Visulisation results of feature impotence-based mutual_info_classif Method: Visualization is the use of computer-supported, visual representation of data. Here provide the visulisation results of feature impotence based mutual_info_classif Method using the 11 Nasa SDP dataset. Below figure 4 shows each dataset feature importance graph using mutual_info_classif method.

Although the mutual info classify function is used to describe and mock-up the feature selection process only on one dataset (pc3), the same procedure is carried out for all the datasets. The steps involved in selecting the most effective discriminatory features from the pc3 dataset are outlined below.

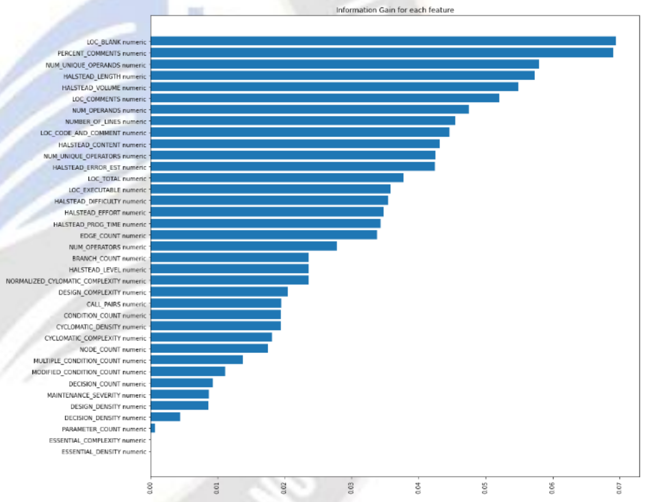


Figure 2. Feature impotence of PC3 Dataset Drop those column which have zero value and lower value by mutual_info_classif()

The above figure 2 shows the feature importance to get by the mutual_info_classif() method by the PC3 Dataset that contain almost 37 attributes. We can see that the graph LOC_BLANK feature shows highest importance in comparison to other attributes. Mutual information gain is largest between the LOC BLANK, as seen by the graph (0.07). In this scenario, LOC BLANK provides 70% of the data necessary to understand the wine variable of interest. We utilise a function from the mutual info classif() python package to find the best K features. The most popular features might be selected as an alternative. Only the top ten characteristics are included in this list.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7659
***Article Received:** 05 June 2023 **Revised:** 30 July 2023 **Accepted:** 15 August 2023*

E. Performance Measures

Standard metrics obtained from the resulting confusion matrices were used to assess the performance of applying ML algorithms to software defact prediction [36]. The confusion matrix and its accompanying assessment metrics are discussed below.

Confusion Matrix: One table used to evaluate ML algorithms' efficacy is called a confusion matrix. The rows of the matrix reflect the examples that belong to a certain actual class, while the columns represent the instances that belong to a given anticipated class, or vice versa. In the summary confusion matrix (FN), the testing algorithm reports the number of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN).

- $\boldsymbol{Tp}$ = represents true positive cases number
- $\boldsymbol{Fp}$ = represents false-negative cases number
- $\boldsymbol{FN}$ = represents false-positive cases number
- $\boldsymbol{TN}$ = represents true negative cases number

Accuracy, precision, recall, ROC, and F1 score were the primary metrics we utilised to assess the performance of our model.

- **The accuracy**

True findings (TP and TN) as a percentage of all evaluated cases are accuracy (ACC). The highest possible accuracy is 1, while the lowest is 0. The following formula may be used to determine ACC:

$$\text{Accuracy} = \frac{TP+TN}{\text{N}} \dots (1)$$

- **The Precision**

Number of accurate positive forecasts as a percentage of total positive predictions represents precision. Maximum accuracy is 1, minimum is 0, and both extremes may be determined using the following formulas.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \dots. (2)$$

- **Recall**

The percentage of correct predictions made is what is used to determine recall. A recall of 1 is the best and a recall of 0 is the worst. The formula for determining Recall is as follows.

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FP}} \dots (3)$$

- **The F1 score**

The F-measure equals the arithmetic mean of the recall and precision values. When comparing several ML algorithms, it is common practice. to use a single metric that includes both Recall and Precision. The formula for the F-measure is:

$$\text{F}_1\text{score} = \frac{2\times(\text{Precision}-\text{Recall})}{(\text{Precision}+\text{Recall})} \dots. (4)$$

- **Mean Absolute Error (MAE)**

The Mean Absolute Error (MAE) is a common measure of effectiveness in regression analysis. Averaging the absolute differences between observed and predicted values of the dependent variable over all observations yields the average absolute difference. Here's the formula:

$$MAE = \frac{1}{n}\sum|y - y'| \quad \dots..(5)$$

where y represents the observed value, n is the sum of observations and y' the foreseen value. The MAE value decreases as performance improves.

- **Mean Squared Error (MSE)**

The average squared deviation from the predicted value of a target variable is known as the mean squared error (MSE), and it is used to measure the precision of a forecasting method. The formula is as follows:

$$MSE = \frac{1}{n}\sum(y - y')^2 \quad \dots..(6)$$

Number of observations (n), value (y), and anticipated value (y') are defined as follows. When measuring performance, a lower MSE is preferable.

- **Root-Mean-Square Error (RMSE)**

Using the root-mean-square error (RMSE), scientists may assess how well a prediction model does its job. Here, we propose a method for quantifying the deviation between expected and observed results. To determine RMSE, we compare the actual value X to the anticipated value XP.

$$RMSE = \sqrt{\frac{1}{n} * \sum_{i=1}^{n}(Xi - XPi)^2} \quad \dots. (7)$$

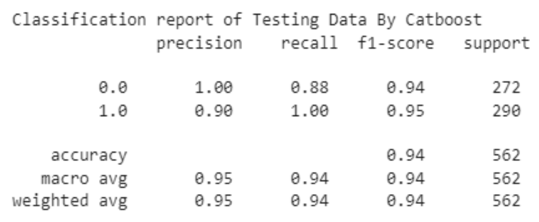
F. Simulation Results of proposed methods

Here, we present the simulation results of the suggested model, a machine learning-based CatBoost, and a Gradient Boost, all applied to a NASA dataset for software defact prediction. This experimental result shows in form of confusion matrix, classification repost, and ROC curve also calculate the accuracy, precision, recall AUC and f1-score or MSE, RMSE, and MAE performance parameters also using 10 K fold cross-validation. Here we provide the results of only one datasets out of eleven because the same results are repeated for other datasets. Below the experimented results are explained to the PC3 dataset. The following bar graph shows results of proposed CatBoost, and Gradient Boost machine learning models for software default predication.

Results of CatBoost and GradientBoost Classifiers with PC3 Dataset

Here in this section provide the experimented results of suggested CatBoost and GradientBoost classifier using PC3 datasets. Following figure 5 to 14 shows the results of these techniques.

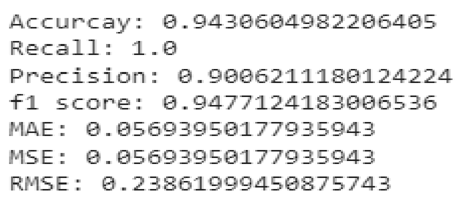
International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7659
***Article Received:** 05 June 2023 **Revised:** 30 July 2023 **Accepted:** 15 August 2023*



```
Classification report of Testing Data By Catboost
                  precision    recall  f1-score   support

         0.0       1.00      0.88      0.94       272
         1.0       0.90      1.00      0.95       290

    accuracy                           0.94       562
   macro avg       0.95      0.94      0.94       562
weighted avg       0.95      0.94      0.94       562
```

Figure 3. Classification report of CatBoost classifier using PC3 Dataset

The results of the suggested CatBoost classifier on the PC3 Dataset are displayed in Figure 3 above. In machine learning, one way to measure success is using a categorization report. It demonstrates the test classification model's accuracy, recall, F1 Score, and support. The input PC3 dataset has two classes, for class 0 precision is 100%, recall 88% and f1-score is 94% whereas for class 1 90%, recall 100% and f1-score is 95% with support 272 and 290. The proposed model CatBoost classifier precision, recall, f1-score, and accuracy is 94% with support 562 respectively.



```
Accurcay: 0.9430604982206405
Recall: 1.0
Precision: 0.9006211180124224
f1 score: 0.9477124183006536
MAE: 0.05693950177935943
MSE: 0.05693950177935943
RMSE: 0.23861999450875743
```

Figure 4. Parameter performance of CatBoost classifier using PC3 Dataset

The above figure 4 shows the Parameter performance of proposed CatBoost classifier using PC3 Dataset. The model gets 94% classification accuracy, precision 90%, recall 100% and f1-score 94% while MAE and MSE are 0.05 or RMSE 0.23 respectively.

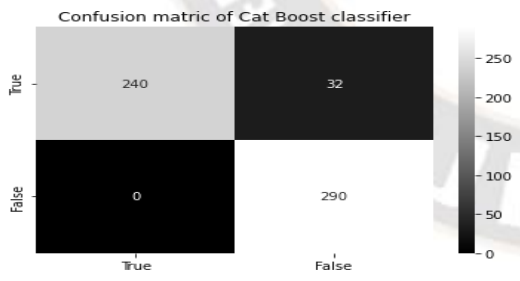
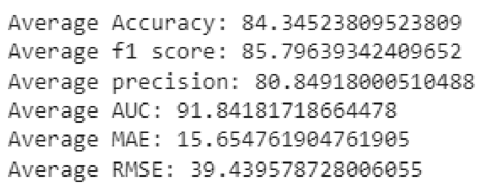


Figure 5. Confusion matrix of CatBoost classifier using PC3 Dataset

The above figure 5 shows the Confusion matrix of proposed CatBoost classifier using PC3 Dataset. The model gets 94% classification accuracy. The false positive rate is 0 and false negative value is 32 while true positive predicated data is 240 and true negative predicated data is 290 respectively.



```
Average Accuracy: 84.34523809523809
Average f1 score: 85.79639342409652
Average precision: 80.84918000510488
Average AUC: 91.84181718664478
Average MAE: 15.654761904761905
Average RMSE: 39.439578728006055
```

Figure 6. Parameter performance of CatBoost classifier with 10K-fold Cross Validation technique using PC3 Dataset

The above figure 6 shows the Parameter performance of proposed CatBoost classifier with 10K-fold Cross Validation technique using PC3 Dataset. The model gets 84% average accuracy with CV, average precision 80%, and average f1-score is 85%, average AUC is 91%, while average MAE 15.65 and average RMSE 39.43 respectively.

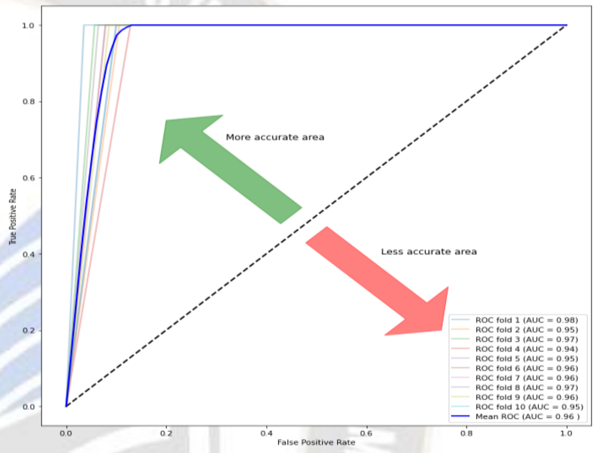
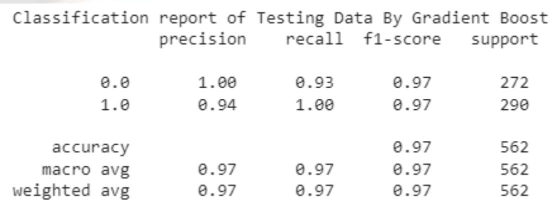


Figure 7. ROC-AUC curve of CatBoost classifier with 10K-fold Cross Validation technique using PC3 Dataset

The above figure 7 shows the ROC-AUC curve of CatBoost classifier with 10K-fold Cross Validation technique using PC3 Dataset. On the ROC fold 1 (AUC=98%), ROC fold 2 (AUC=95%), ROC fold 3 (AUC=97%), ROC fold 4 (AUC=94%), ROC fold 5 (AUC=95%), ROC fold 6 (AUC=96%), ROC fold 7 (AUC=96%), ROC fold 8 (AUC=97%), ROC fold 9 (AUC=96%) and ROC fold 10 (AUC=95%) respectively. The proposed model shows the 96% Mean AUC performance on PC3 dataset.



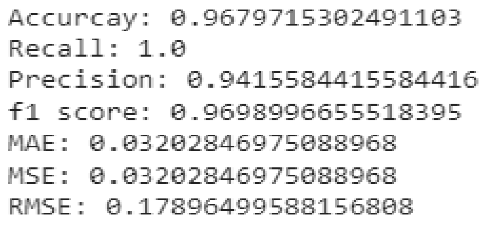
```
Classification report of Testing Data By Gradient Boost
                  precision    recall  f1-score   support

         0.0       1.00      0.93      0.97       272
         1.0       0.94      1.00      0.97       290

    accuracy                           0.97       562
   macro avg       0.97      0.97      0.97       562
weighted avg       0.97      0.97      0.97       562
```

Figure 8. Classification report of GradientBoost classifier using PC3 Dataset

The results of the proposed Gradient Boost classifier on the PC3 Dataset are displayed in Figure 8 above. A classification report is a performance evaluation metric. The input PC3

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7659
***Article Received:** 05 June 2023 **Revised:** 30 July 2023 **Accepted:** 15 August 2023*

dataset has two classes, for class 1 precision 94%, recall 100% and f1-score is 97% whereas for class 0 90%, recall 100% and f1-score is 97% with support 272 and 290. The proposed model GradientBoost classifier precision, recall, f1-score, and accuracy is 97% with support 562 respectively.



```
Accurcay: 0.9679715302491103
Recall: 1.0
Precision: 0.9415584415584416
f1 score: 0.9698996655518395
MAE: 0.03202846975088968
MSE: 0.03202846975088968
RMSE: 0.17896499588156808
```

Figure 9. Parameter performance of GradientBoost classifier using PC3 Dataset

The above figure 9 shows the Parameter performance of proposed GradientBoost classifier using PC3 Dataset. The model gets 96% classification accuracy, precision 94%, recall 100% and f1-score is 96% while MAE and MSE are 0.032 or RMSE 0.17 respectively.

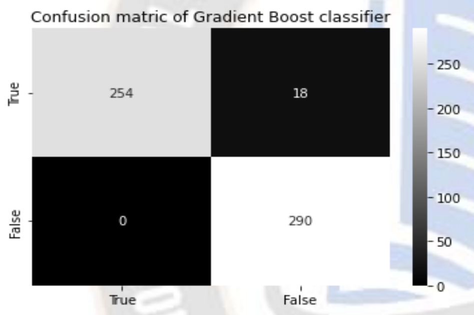
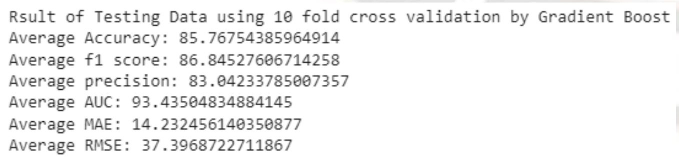


Figure 10. Confusion matrix of GradientBoost classifier using PC3 Dataset

The above figure 10 shows the Confusion matrix of proposed GradientBoost classifier using PC3 Dataset. The model gets 97% classification accuracy. The false positive rate is 0 and false negative value is 18 while true positive predicated data is 254 and true negative predicated data is 290 respectively.



```
Rsult of Testing Data using 10 fold cross validation by Gradient Boost
Average Accuracy: 85.76754385964914
Average f1 score: 86.84527606714258
Average precision: 83.04233785007357
Average AUC: 93.43504834884145
Average MAE: 14.232456140350877
Average RMSE: 37.3968722711867
```

Figure 11. Parameter performance of GradientBoost classifier with 10K-fold Cross Validation technique using PC3 Dataset

The above figure 11 shows the Parameter performance of proposed GradientBoost classifier with 10K-fold Cross Validation technique using PC3 Dataset. The model gets 85% average accuracy with CV, average precision 83%, and average f1-score is 86%, average AUC is 93%, while average MAE 14.23 and average RMSE 37.39 respectively.

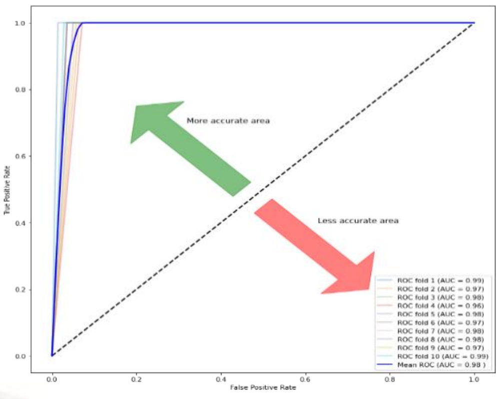


Figure 12. ROC-AUC curve of GradientBoost classifier with 10K-fold Cross Validation technique using PC3 Dataset

The above figure 12 shows the ROC-AUC curve of GradientBoost classifier with 10K-fold Cross Validation technique using PC3 Dataset. On the ROC fold 1 (AUC=99%), ROC fold 2 (AUC=97%), ROC fold 3 (AUC=98%), ROC fold 4 (AUC=96%), ROC fold 5 (AUC=98%), ROC fold 6 (AUC=98%), ROC fold 7 (AUC=97%), ROC fold 8 (AUC=98%), ROC fold 9 (AUC=98%) and ROC fold 10 (AUC=99%) respectively. The proposed model shows the 98% Mean AUC performance on PC3 dataset.

G. Comparative analysis and Discussion

Eleven datasets are used to evaluate several metrics, including mean absolute error, mean square error, root mean square error, and f1-score. In this study, we examine and compare the various ML methods used for defect prediction. Here, we examine the PC4, PC3, CM1, and MC1 datasets and compare the most discriminating experimental outcomes. The following graph is a comparison of many machine learning models used for predicting software failure. Tables 2–6 compare a variety of ML methods utilised in past research. Logistic Regression (LR) and Decision Tree (DT) with 10-dold cross validation were two of the ML methods we looked at since they have been used as cutting-edge approaches in previous studies. using 10-fold cross-validation on the PC4, PC3, CM1, and MC1datasets we chose, as well as our proposed CatBoost and Gradient Boost machine learning models.

TABLE II. ACCURACY PARAMETER COMPARISON OF BASED AND PROPOSE CLASSIFIERS

Datasets	**Base Classifiers**		**Propose Classifiers**	
	LR	**DT**	**CatBoost**	**GradientBoost**
PC4	71	87	89	91
PC3	73	84	84	85
CM1	90	88	73	75
MC1	73	85	89	99

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7659
***Article Received:** 05 June 2023 **Revised:** 30 July 2023 **Accepted:** 15 August 2023*

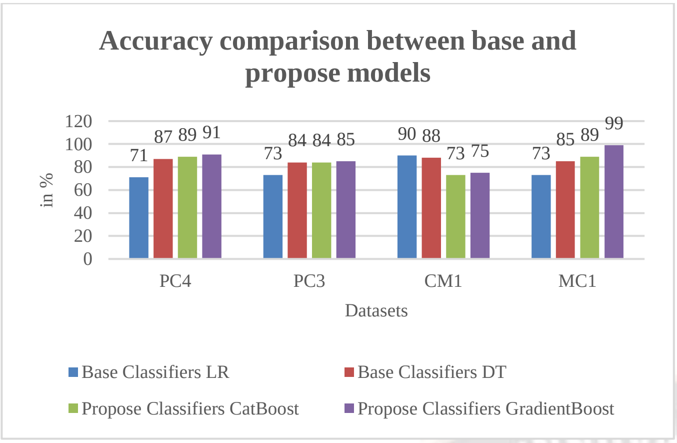


Figure 13. Bar graph of accuracy comparison between base and propose classifier

The above figure 13, shows the machine learning classifiers performance for software default prediction using PC4, PC3, CM1, and MC1 datasets. In bar graph shows the comparison between base and propose classifiers in terms of accuracy. For the PC4 dataset, 71% accuracy of base logistic regression and 87% accuracy of decision tree, PC3 dataset obtain 73% accuracy of base logistic regression and 84% accuracy of decision tree, CM1 got 90% and 88% accuracy for both base classifiers and MCI dataset achieved 73% and 85% accuracy of logistics regression and decision tree, While propose CatBoost 89% accuracy, and GradientBoost 91% accuracy on PC4 dataset, PC3 dataset obtain 84% and 85% accuracy, 73% and 75% accuracy on CM1 and 98% and 99% accuracy on MC1 dataset. We can the proposed CatBoost, and GradientBoost classifiers achieved high accuracy with 99% in comparison to base classifiers for the software default prediction

TABLE III. PRECISION PARAMETER COMPARISON OF BASED AND PROPOSE CLASSIFIERS

Datasets	**Base Classifiers**		**Propose Classifiers**	
	LR	**DT**	**CatBoost**	**GradientBoost**
PC4	32	62	83	87
PC3	50	24	80	83
CM1	13	20	68	72
MC1	50	58	83	98

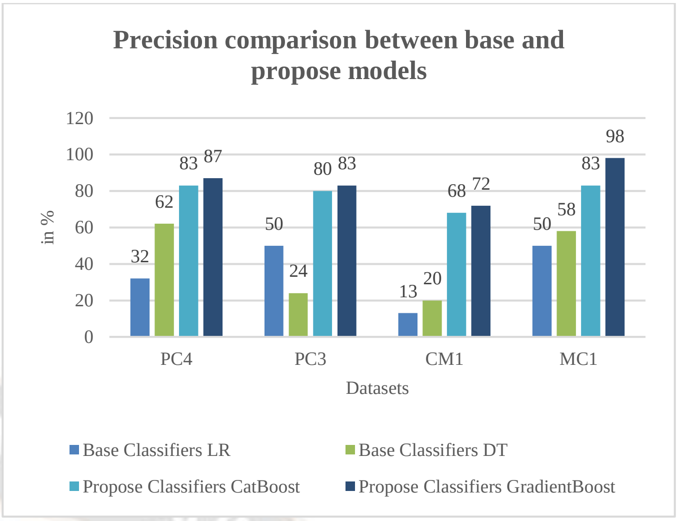


Figure 14. Bar graph of precision comparison between base and propose classifier

The above figure 14 shows the comparison between base and propose classifiers in terms of precision. For the PC4 dataset, 32% precision of base logistic regression and 62% precision of decision tree, PC3 dataset obtain 50% precision of base logistic regression and 24% precision of decision tree, CM1 get 13% and 20% precision for both base classifiers and MCI dataset achieved 50% and 58% precision of logistics regression and decision tree, While propose obtain CatBoost 83%, and GradientBoost 87% on PC4 dataset, PC3 dataset obtain precision 80%, and 83%, precision of proposed CatBoost and GradientBoost classifiers. On the CM1 dataset CatBoost precision 68%, and GradientBoost precision 72%. At last 83% and 98% precision obtain by the proposed classifier on the MC1 dataset respectively. We can the proposed achieved high precision with 98% in comparison to base classifiers for the software default prediction.

TABLE IV. F1-SCORE PARAMETER COMPARISON OF BASED AND PROPOSE CLASSIFIERS

Datasets	**Base Classifiers**		**Propose Classifiers**	
	LR	**DT**	**CatBoost**	**GradientBoost**
PC4	45	52	83	91
PC3	66	27	85	85
CM1	18	20	73	75
MC1	66	57	99	90

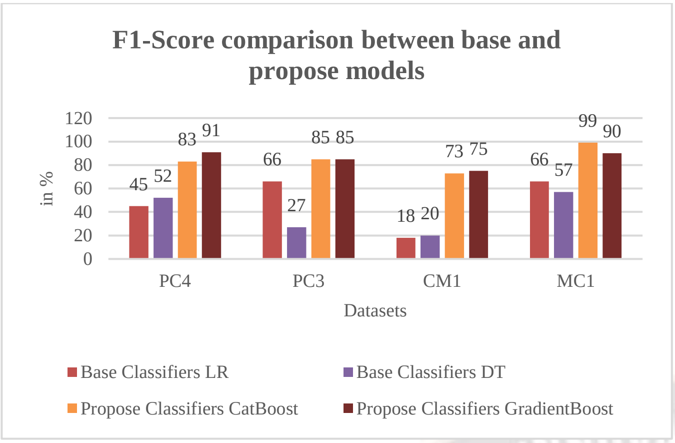


Figure 15. Bar graph of f1-score comparison between base and propose classifier

The above figure 15 shows the comparison between base and propose classifiers in terms of f1-score. For the PC4 dataset, 45% f1-score of base logistic regression and 52% f1-score of decision tree, PC3 dataset obtain 66% f1-score of base logistic regression and 27% f1-score of decision tree, CM1 get 18% and 20% f1-score for both base classifiers and MCI dataset achieved 66% and 57% f1-score of logistics regression and decision tree, While propose CatBoost 83%, and GradientBoost get 91% f1-score on PC4 dataset, PC3 dataset obtain 84%, and 85% f1-score of proposed classifiers. On the CM1 dataset CatBoost f1-score 73%, GradientBoost f1-score 75%. At last 90% and 99%, f1-score obtain by the proposed classifiers on the MC1 dataset respectively. We can the proposed classifiers achieved high f1-score with 99% in comparison to base classifiers for the software default prediction.

TABLE V. RMSE PARAMETER COMPARISON OF BASED AND PROPOSE CLASSIFIERS

Datasets	Base Classifiers		Propose Classifiers	
	LR	DT	CatBoost	GradientBoost
PC4	52.53	33.45	32.42	29.34
PC3	31.11	37.27	39.43	37.39
CM1	24.24	22.33	50.42	48.93
MC1	31.11	21.35	32.42	8.20

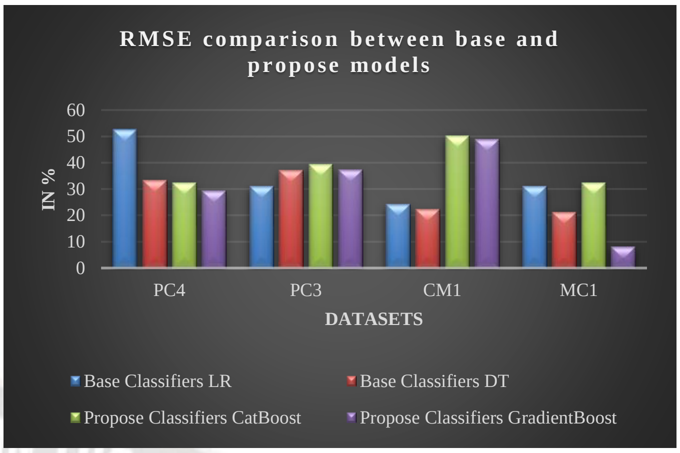


Figure 16. Bar graph of RMSE comparison between base and propose classifier

The above figure 16 shows the comparison between base and propose classifiers in terms of RMSE. For the PC4 dataset, 52.53% RMSE of base logistic regression and 33.45% RMSE of decision tree, PC3 dataset obtain 31.11% RMSE of base logistic regression and 37.27% RMSE of decision tree, CM1 get 24.24% and 22.33% RMSE for both base classifiers and MCI dataset decrease 31.11% and 21.35% RMSE of logistics regression and decision tree, While propose CatBoost 32%, and GradientBoost 29% RMSE on PC4 dataset, PC3 dataset obtain 39%, and 34% RMSE of proposed classifiers. On the CM1 dataset CatBoost RMSE 50%, and GradientBoost RMSE 48%. At last 32% and 8%, RMSE obtain by the proposed classifiers on the MC1 dataset respectively. We can the proposed CatBoost, and GradientBoost classifiers decrease high RMSE with 8% in comparison to base classifiers for the software default prediction.

TABLE VI. MAE PARAMETER COMPARISON OF BASED AND PROPOSE CLASSIFIERS

Datasets	Base Classifiers		Propose Classifiers	
	LR	DT	CatBoost	GradientBoost
PC4	28.15	11.26	10.74	8.98
PC3	9.68	15.93	15.65	14.23
CM1	10	10.24	26.21	24.46
MC1	9.68	15.93	10.74	0.79

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7659
***Article Received:** 05 June 2023 **Revised:** 30 July 2023 **Accepted:** 15 August 2023*

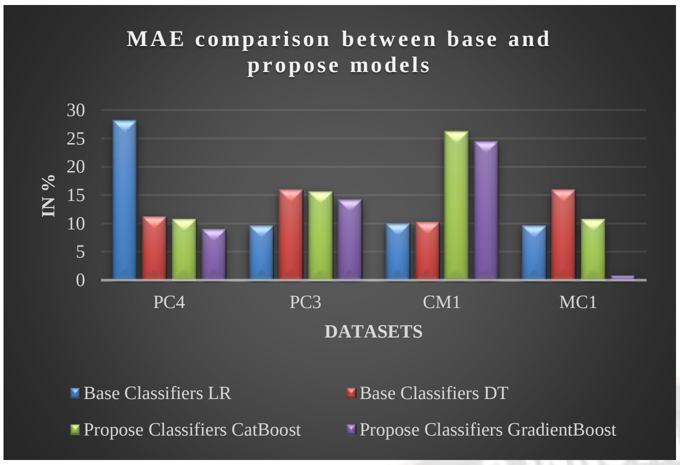


Figure 17. Bar graph of MAE comparison between base and propose classifier

The above figure 17 shows the comparison between base and propose classifiers in terms of MAE. For the PC4 dataset, 28.15% MAE of base logistic regression and 11.26% MAE of decision tree, PC3 dataset obtain 9.68% MAE of base logistic regression and 15.93% MAE of decision tree, CM1 get 10% and 10.24% MAE for both base classifiers and MCI dataset decrease 9.68% and 15.93% MAE of logistics regression and decision tree, While propose CatBoost 10%, and GradientBoost 8.98%, MAE on PC4 dataset, PC3 dataset obtain 15.65%, and 14.23%, MAE of proposed CatBoost, and GradientBoost. On the CM1 dataset CatBoost MAE 26.21%, and GradientBoost MAE 24.46%. At last 10.74 and 0.79MAE obtain by the proposed classifiers on the MC1 dataset respectively. We can the proposed classifiers decrease high MAE with 10.74 % in comparison to base classifiers for the software default prediction.

VI. CONCLUSION AND FUTURE SCOPE

Today, machine learning research is prevalent across the whole IT and software landscape. The subfield of machine learning known as software defect prediction is making significant strides in recent years. One step in that direction is employing ML methods for fault prediction in software. When used early in the development process, this method enhances programme performance while decreasing software maintenance costs. In this paper, we provide a system for predicting software flaws that are both effective and trustworthy. Our work using NASA's JM1, CM1, PC1, PC3, PC4, MC1, KC1, MOZILLA4, and JEDIT defect prediction data sets utilising the PYTHON programming language and recommended machine learning classification approaches like CatBoost and GredientBoost classifiers is available in the PROMISE repository. Algorithms are evaluated based on metrics like MSE, MAE, and RMSE, all of which are calculated using data. Measurements of the efficacy of machine learning algorithms reveal that they fare admirably on each of the eleven datasets considered. The data obtained overwhelmingly supported the superior performance of the ML method. The proposed CatBoost and GredientBoost classifiers obtain highest 90% to 99% accuracy, recall, precision, and f1-score on MC1 dataset or better MAE and RMSE with 8.20 and 0.79 obtained by the MC1 dataset. This dataset reduces the error of the software default prediction.

In a possible follow-up, we want to include more ML techniques and compare them extensively. Adding more software measurements throughout the learning process may also increase the prediction model's accuracy.

REFERENCES

[1] N. Babu, Himagiri, V. Vamshi Krishna, A. Anil Kumar, and M. Ravi, “Software defect prediction analysis by using machine learning algorithms.,” Int. J. Recent Technol. Eng., 2019, doi: 10.35940/ijrte.B1438.0982S1119.

[2] M. C. M. Prasad, L. F. Florence, and A. Arya3, “A Study on Software Metrics based Software Defect Prediction using Data Mining and Machine Learning Techniques,” Int. J. Database Theory Appl., 2015, doi: 10.14257/ijdta.2015.8.3.15.

[3] S. Huda et al., “A Framework for Software Defect Prediction and Metric Selection,” IEEE Access, 2017, doi: 10.1109/ACCESS.2017.2785445.

[4] P. Paramshetti and D. A. Phalke, “Software Defect Prediction for Quality Improvement Using Hybrid Approach,” Int. J. Appl. or Innov. Eng. Manag., 2015.

[5] M. W. Thant and N. T. T. Aung, “Software Defect Prediction using Hybrid Approach,” in 2019 International Conference on Advanced Information Technologies (ICAIT), 2019, pp. 262–267. doi: 10.1109/AITC.2019.8921374.

[6] K. Tanaka, A. Monden, and Z. Yucel, “Prediction of Software Defects Using Automated Machine Learning,” 2019. doi: 10.1109/SNPD.2019.8935839.

[7] Meiliana, S. Karim, H. L. H. S. Warnars, F. L. Gaol, E. Abdurachman, and B. Soewito, “Software metrics for fault prediction using machine learning approaches: A literature review with PROMISE repository dataset,” in 2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), 2017, pp. 19–23. doi: 10.1109/CYBERNETICSCOM.2017.8311708.

[8] A. Alsaeedi and M. Z. Khan, “Software Defect Prediction Using Supervised Machine Learning and Ensemble Techniques: A Comparative Study,” J. Softw. Eng. Appl., 2019, doi: 10.4236/jsea.2019.125007.

[9] P. Paramshetti and D. A. Phalke, “Survey on Software Defect Prediction Using Machine Learning Techniques,” Int. J. Sci. Res., 2014.

[10] M. F. Sohan, M. A. Kabir, M. I. Jabiullah, and S. S. M. M. Rahman, “Revisiting the Class Imbalance Issue in Software Defect Prediction,” 2019. doi:

10.1109/ECACE.2019.8679382.
[11] S. K. Pandey and A. K. Tripathi, "Class Imbalance Issue in Software Defect Prediction Models by various Machine Learning Techniques: An Empirical Study," 2021. doi: 10.1109/ICSCC51209.2021.9528170.
[12] Z. W. Zhang, X. Y. Jing, and T. J. Wang, "Label propagation based semi-supervised learning for software defect prediction," Autom. Softw. Eng., 2017, doi: 10.1007/s10515-016-0194-x.
[13] H. Wang, T. M. Khoshgoftaar, and A. Napolitano, "A comparative study of ensemble feature selection techniques for software defect prediction," 2010. doi: 10.1109/ICMLA.2010.27.
[14] R. S. Wahono, "A Systematic Literature Review of Software Defect Prediction: Research Trends, Datasets, Methods and Frameworks," J. Softw. Eng., 2015.
[15] Q. Song, Y. Guo, and M. Shepperd, "A Comprehensive Investigation of the Role of Imbalanced Learning for Software Defect Prediction," IEEE Trans. Softw. Eng., 2019, doi: 10.1109/TSE.2018.2836442.
[16] K. Bashir, T. Li, C. W. Yohannese, and Y. Mahama, "Enhancing software defect prediction using supervised-learning based framework," 2017. doi: 10.1109/ISKE.2017.8258790.
[17] J. Ge, J. Liu, and W. Liu, "Comparative study on defect prediction algorithms of supervised learning software based on imbalanced classification data sets," 2018. doi: 10.1109/SNPD.2018.8441143.
[18] K. E. Bennin, J. Keung, P. Phannachitta, A. Monden, and S. Mensah, "MAHAKIL: Diversity Based Oversampling Approach to Alleviate the Class Imbalance Issue in Software Defect Prediction," IEEE Trans. Softw. Eng., 2018, doi: 10.1109/TSE.2017.2731766.
[19] A. Khalid, G. Badshah, N. Ayub, M. Shiraz, and M. Ghouse, "Software Defect Prediction Analysis Using Machine Learning Techniques," Sustainability, vol. 15, no. 6, 2023, doi: 10.3390/su15065517.
[20] P. Deep Singh and A. Chug, "Software defect prediction analysis using machine learning algorithms," 2017. doi: 10.1109/CONFLUENCE.2017.7943255.
[21] M. Assim, Q. Obeidat, and M. Hammad, "Software Defects Prediction using Machine Learning Algorithms," 2020 Int. Conf. Data Anal. Bus. Ind. W. Towar. a Sustain. Econ. ICDABI 2020, 2020, doi: 10.1109/ICDABI51230.2020.9325677.
[22] P. Tadapaneni, N. C. Nadella, M. Divyanjali, and Y. Sangeetha, "Software Defect Prediction based on Machine Learning and Deep Learning," in 2022 International Conference on Inventive Computation Technologies (ICICT), 2022, pp. 116–122. doi: 10.1109/ICICT54344.2022.9850643.
[23] Y. Shen, S. Hu, S. Cai, and M. Chen, "Software Defect Prediction based on Bayesian Optimization Random Forest," 2022. doi: 10.1109/DSA56465.2022.00149.
[24] S. A. Alasadi and W. S. Bhaya, "Review of data preprocessing techniques in data mining," J. Eng. Appl. Sci., 2017, doi: 10.3923/jeasci.2017.4102.4107.
[25] M. M. Ahsan, M. A. P. Mahmud, P. K. Saha, K. D. Gupta, and Z. Siddique, "Effect of Data Scaling Methods on Machine Learning Algorithms and Model Performance," Technologies, 2021, doi: 10.3390/technologies9030052.
[26] J. Li et al., "Feature selection: A data perspective," ACM Computing Surveys. 2017. doi: 10.1145/3136625.
[27] A. Jadhav, S. M. Mostafa, H. Elmannai, and F. K. Karim, "An Empirical Assessment of Performance of Data Balancing Techniques in Classification Task," Appl. Sci., vol. 12, no. 8, 2022, doi: 10.3390/app12083928.
[28] S. A. Alsaif and A. Hidri, "Impact of data balancing during training for best predictions," Inform., 2021, doi: 10.31449/inf.v45i2.3479.
[29] R. Ghorbani and R. Ghousi, "Comparing Different Resampling Methods in Predicting Students' Performance Using Machine Learning Techniques," IEEE Access, 2020, doi: 10.1109/ACCESS.2020.2986809.
[30] X. Y. Wang, Y. Yang, Y. T. Bai, J. Bin Yu, Z. Y. Zhao, and X. B. Jin, "Fuzzy Boost Classifier of Decision Experts for Multicriteria Group Decision-Making," Complexity, 2020, doi: 10.1155/2020/8147617.
[31] A. K. Jaggi, A. Sharma, N. Sharma, R. Singh, and P. S. Chakraborty, "Diabetes Prediction Using Machine Learning," Lect. Notes Networks Syst., vol. 185 LNNS, no. 09, pp. 383–392, 2021, doi: 10.1007/978-981-33-6081-5_34.
[32] J. H. Friedman, "Stochastic gradient boosting," Comput. Stat. Data Anal., 2002, doi: 10.1016/S0167-9473(01)00065-2.
[33] S. Nagar, Introduction to Python for Engineers and Scientists. 2018. doi: 10.1007/978-1-4842-3204-0.
[34] M. Källén, "Jupyter Notebooks on GitHub: Characteristics and Code Clones," Uppsala University, 2020.
[35] S. Gupta, "Best Python Libraries for Machine and Deep Learning," 2022.
[36] M. Gong, "A Novel Performance Measure for Machine Learning Classification," Int. J. Manag. Inf. Technol., vol. 13, no. 1, pp. 11–19, 2021, doi: 10.5121/ijmit.2021.13101.
[37] Bhawana Verma, S. K.A. (2019). Design &amp; Analysis of Cost Estimation for New Mobile-COCOMO Tool for Mobile Application. International Journal on Recent and Innovation Trends in Computing and Communication, 7(1), 27–34. https://doi.org/10.17762/ijritcc.v7i1.5222
[38] S. K.A., Raj, A. ., Sharma, V., & Kumar, V. (2022). Simulation and Analysis of Hand Gesture Recognition for Indian Sign Language using CNN. International Journal on Recent and Innovation Trends in Computing and Communication, 10(4), 10–14. https://doi.org/10.17762/ijritcc.v10i4.5556.