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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

A Novel Developed Supervised Machine Learning System For Classification And Prediction of Software Faults Using NASA Dataset

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

Abstract— The software systems of modern computers are extremely complex and versatile. Therefore, it is essential to regularly detect and correct software design faults. In order to devote resources effectively towards the creation of trustworthy software, software companies are increasingly engaging in the practise of predicting fault-prone modules in advance of testing. These software fault prediction methods rely on the thoroughness with which prior software versions' fault as well as related code has been retrievedTime, energy, and money are all saved as a result. Increases the company's initial success and bottom line greatly by satisfying its clientele. Numerous academics have poured into this area throughout the years in an effort to raise the bar for all software. Nowadays, The most often used approaches in this field are those based on machine learning (ML). The field of ML seeks to perfect software capable of evolving as well as adapting in response to fresh data. This paper introduces a fresh approach for doing ML by bringing together a number of different expert systems. In order to reach agreement on which aspects of a software system need to be tested, the proposed multi-classifier model pools the strengths of the most effective classifiers. Several top-performing classifiers for defect prediction are put through their paces in an experiential evaluation. We test our method on 16 publicly available datasets from the NASA Metric Data Programme (MDP) repository at the promise repository. Parameters of confusion, recall, precision, recognition accuracy, etc., are evaluated and contrasted with existing schemes in a software analysis performed with the help of the python simulation tool with findings. The experimental outcomes demonstrate that by combining LGBM, XGBoost, and Voting classifiers, using a multi classifier approach, we are capable to significantly improve software fault prediction performance. The results of the investigation show that the suggested method will lead to better practical outcomes in the prediction of device failures.

Keywords- Software systems, Software defect, Software fault prediction, ML, Supervised ML, classification, LGBM, XGBoost , Voting, NASA dataset**.**

I. INTRODUCTION

The proliferation of software products is a side effect of software technology progress, and keeping up with them all has grown into a formidable challenge. Maintenance operations account for greater than 50% of the total cost of ownership of a software system. The likelihood of discovering flawed components in software systems grows in tandem with their increasing complexity[1]. It is crucial to anticipate and address problems before they is provided to users as-is since software quality assurance is time-consuming, and limited resources prevent thorough testing of the whole platform. Thus, finding a flawed piece of software can help us make better use of our timeand money. A software system flaw, or "bug," is another term for the same thing[2][3].

A software or product failure occurs when its results are not what the customer wants, we have a software defect. Such defects are examples of programming mistakes that manifest as failures, unpredictability, or unexpected consequences and might originate in either the source code or the requirements. Such defects have a negative impact on software quality as well as programme reliability and can result in wasted money, effort, and resources. Repairing failures requires more time and money spent on maintenance. This makes early defect prediction in software an important field of study [4].It is the objective of Software Fault Prediction (SFP) methods to help developers more efficiently allocate their time and resources between (i) testing, due to the increased likelihood of testing failure-prone components, and (ii) refactoring, with the objective of enhancing the design for these parts, to reduce the likelihood of incorporating new bugs while working on them[5].

SFP models [4] are typically built to determine fault severity, fault classification in binary terms, and the total amount of

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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

defects. As can be seen in Fig. 1, the classification model is first trained using data amassed from earlier versions of similar software projects to identify patterns of software faults. In software fault datasets, the independent variables (like the amount of attributes, methods, and lines of code) are software metrics, as well as the dependent variables (either the module is defective or not with regards to providing a true or false value) are faulty or non-faulty.

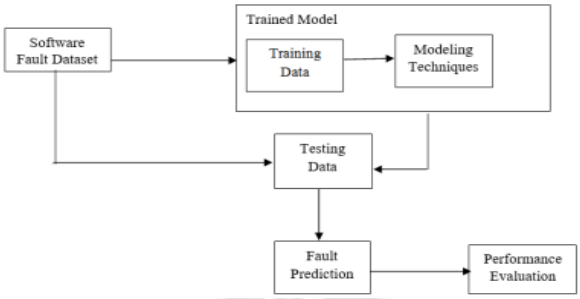

Figure 1. Software Fault Prediction Process

Since 1990, a lot of investigation has been conducted on metrics & efficient modelling strategies to improve SFP performance[6]. The creation of the PROMISE repository in 2005, nevertheless, caused a surge of activity in this sector. Investigators were capable of making their findings replicable and trustworthy because of the abundance of public datasets available in this repository. When describing the qualities of a software component, object-oriented metrics are frequently used. Amongst the various software defect prediction research that focus on ML and statistical approaches, random forest (RF) and naive Bayes (NB)have been shown to execute exceptionally well and consistently[7]. In addition to ML, ensemble learning also has a major effect on fault prediction outcomes. A broad range of evaluation metrics are utilised to examine the efficiency of classifiers, with the choice of a particular measure depending on the dataset and modelling method employed for prediction.

Various ML methods have been applied to the task of fault prediction. However, there is no universally superior ML method [9]. Therefore, the greatest outcomes can only be achieved by the use of an efficient method, and integrating the most effective classifier is one such method. Multiple classifiers working together might be able to make a more accurate forecast than any one of them could on their own. Such findings prompted investigators to pool their students for a collective classification verdict.It has been shown empirically that certain classifier combination schemes regularly outperform than a single best classifier. Integrating classifiers improves both productivity and precision. Researchers frequently employ majority voting as a classifier combination approach[8]. In, we provide the findings of a deep dive into the attribute space and classifier input range[7][9]. Research into software defect prediction employing a variety of ML methods is widespread. Such methods showed potential on some data sets but underwhelmed on others. One way is to use an ensemble method to construct a model that is reliable and effective across all data sets. The purpose of this research is to determine ML & ensemble-based modelling approaches for predicting software faults. Investigators have been making extensive usage of ML methods in recent years; the production and combination of weak learners for the final output has boosted the use of ensemble-based learning in nowadays. As a result, the model's accuracy improves.

The goal of this research is to improve software quality by employing supervised ML techniques for SFP and detection. Supervised learning refers to a subfield of ML in that systems are taught to make predictions based on data that has been explicitly tagged for use in training. We show off our algorithm on 16 datasets from NASA's Promise repository for predicting software bugs. The following are some of the major results of this study:

- NASA Metric Data Programme (MDP) programmes& Turkish software initiatives make up the 16 public datasets obtained from the PROMISE repository.
- To improve the precision of software defect prediction, supervised classifiers relying on ML will be implemented.
- To enhance fault prediction and produce continuously excellent outcomes across all datasets.
- We examine the recall, precision, accuracy, f1-score and Area Under the Curve (AUC) of different techniques to estimate fault prediction capabilities in order to provide an explanation for this experimental result.
- By contrasting the suggested classifiers with those already on the market, we can see that they function admirably.

The rest of this work is structured in four parts. Section 2 discusses software fault prediction-related works. There have been a significant number of research articles published in our chosen field of study. Section 3 describes the phases and strategies used in the proposed software failure prediction system in detail. Section 4 summarises the various experiments and offers the results. also conducts a comparative analytical investigation of the suggested SFP and other modern methods. Finally, Section 5 summarises the effort and results and suggests future research areas.

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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

II. RELATED WORK

The majority of the existing research on software defect prediction does not thoroughly compare and contrast all of the available ML techniques. While some have presented an approach that utilises preexisting ML methods by expanding them, some have utilised only a few methods & provided comparisons to those that have been explored thus far.

For this purpose[10], the NASA PROMISE repository's defective data set was used in conjunction with the K-Nearest Neighbour (KNN) and RF supervised ML algorithms to make predictions about the likelihood of future software defects. Multiple metrics, such as accuracy, precision, recall, and the f1 measure, were used to evaluate the models' performance. Highest and lowest accuracy of 99% and 88% respectively on MC1 as well as KC1 are demonstrated in this work, demonstrating the superior efficiency of the RF model over the KNN model.

In [11], use software metrics from the Promise repository dataset to carry out an experimental research comparing the efficacy of 7 well-known methods, such as Logistic Regression (LR), KNN, Support Vector Machine (SVM), DT, RF, NB, and Multilayer Perceptron (MLP). Both method-level as well as class-level datasets are used in our investigation. LR & MLP yield nearly identical AUCs (0.90 as well as 0.91, correspondingly). In addition, MLP and LR both yield 91% accuracy. MLP outperforms LR (0.45 F1), but just slightly. For datasets at the method level, MLP provides the most accurate error predictions.

In [12],Examine the propensity prediction abilities of the MLP, SVM as well as DT 3 ML algorithms. The experimental findings demonstrated that the use of the additional data sets enhanced the accuracy of error-type prediction by ML models.

In order to comprehend SFP, Improved defects[13], this research proposes a web-accessible healthcare defect diagnosis methodology. The effectiveness of the model is measured with the aid of ML technologies like RF, DT, and SVM, and particular metrics are built with feature extraction methods. Lastly, the relative merits of the various ML methods are reviewed and contrasted.

Considering their widespread use in fault prediction context,5 classifiers were implemented in this study [14], KNN, LR, multinomial NB, DT and NB. The model is evaluated using 4 datasets that may be found in the PROMISE database. The F-measure is employed to assess the efficacy of fault prediction models. The outcomes are cross validated employing k-fold (k=10) cross validation to remove the randomness as well as bias from the samples. As contrasted to other ensembles, the experimental findings favoured the model averaging strategy.

In [15], introduces a method for detecting software defects that can help fix a few of the most fundamental issues with current systems. Using a combination of fundamental noise removal, imbalanced class distribution, and software metrics selection methodologies, this study aims to enhance SFP. 10 SFP datasets were used to evaluate the method. The experimental findings demonstrate that the suggested approach improves fault prediction ability, with outcomes which are either greater than or comparable to a number of comparison models in terms of F-measure, recall, precision, accuracy, as well as ROC-AUC values. This demonstrates that our model is correct.

In [16], seeks to examine the amount of help SFP can get from inheritance metrics. The Chidamber&Kemerer (CK) metrics are chosen firstly because they are among the most widely used collection of indicators for forecasting software errors and inheritance. To assess the role of inheritance in SFP, we employ 65 freely available basis datasets including CK measurements and other inheritance metrics. For the purpose of making comparisons, we additionally divided every dataset into 2 subsets: inheritance with CK as well as CK without inheritance. Models are constructed using an ANN, with results evaluated in terms of F1 measure, recall, accuracy, precision, and the true negative rate (TNR). When compared, the outcomes demonstrate that inheritance metrics contribute reasonably well to SFP. Using inheritance metrics for defect prediction in software testing is completely safe. Furthermore, high inheritance is undesirable since it can cause software defects.

In [17] , In order to detect software defects, you need develop techniques to calculate Fourier coefficients and obtain the expected function. Also, we evaluate Fourier learning against standard ML techniques, like the RF method. Lastly, the experimental findings demonstrate that the Fourier learning method is superior to alternative methods in terms of both performance and stability.

In Table 1 of the literature review, the pros and cons of prior studies of the software failure prediction paradigm are outlined.

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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

TABLE I. ADVANTAGES AND DISADVANTAGES OF VARIOUS EXISTING STUDIE

Authors	Methods	Dataset	Benefits	Limitations
[16]	Artificial neural networks (ANN)	Inheritance with CK and CK without inheritance	offered improved outcomes for each of the four criteria, f1-measure, TNR, precision, and accuracy.	Some important issues are not addressed, such as the quantity, severity, and causes of faults.
[11]	LR, KNN, DT, RF, NB, SVM, as well as MLP	Promise repository dataset	The experiment was conducted on class-level datasets as well as method-level datasets.	Class imbalance issues are not addressed. Ignore many faults in favor of the simpler challenge of categorizing faults into two groups.
[17]	Fourier coefficient and Fourier learning algorithm	NASA dataset	The AUC of a model trained with Fourier methods is higher. It's more reliable than other SFP models out there.	They may have evaluated skewed data because they did not address data imbalance and noise issues. The number of errors ignored.
[14]	DT, LR, NB, multinomial NB and KNN	PROMISE repository	As contrasted to other ensembles, the experimental findings favoured the model averaging strategy.	All machine learning models are not achived similar accuracy for SFP
[10]	KNN and RF	NASA's PROMISE repository	99% and 88% accuracy that is highe detection rate of software fault	The KNN model obtain only 88% accuracy.

III. RESEARCH METHODOLOGY

Predicting defects and faults in software is what SFP is all about. This is a fundamental part of the procedure of ensuring the quality of software. Different kinds of software metrics are used for different purposes in fault prediction. The most common kinds of software metrics are code metrics and process measurements. The technique of calculating software metrics from a software then utilising them to anticipate software defects is computationally and time-intensive. In order to complete the fault prediction method effectively while utilising fewer resources, it is helpful to reduce the number of software metrics to utilise just to the essential metrics. Class imbalance is infrequently employed, and previous research shows that regression issues in SFP are not studied as thoroughly as classification issues. Our research solves this problem byclass imbalance and classification problems for prediction of number of software faults. SMOTE oversamplingrelied ondata balancing method is implemented on 16 public datasets that collected from the NASA PROMISE repository. Then, supervised ML classifiers like LGBM, XGBost, & Voting are applied to the training and testing set to evaluate and predict the number of software errors. These machine learning model enahcne the performance of software fault prediction. All methodology process discirbed below section with proposed flowchart.

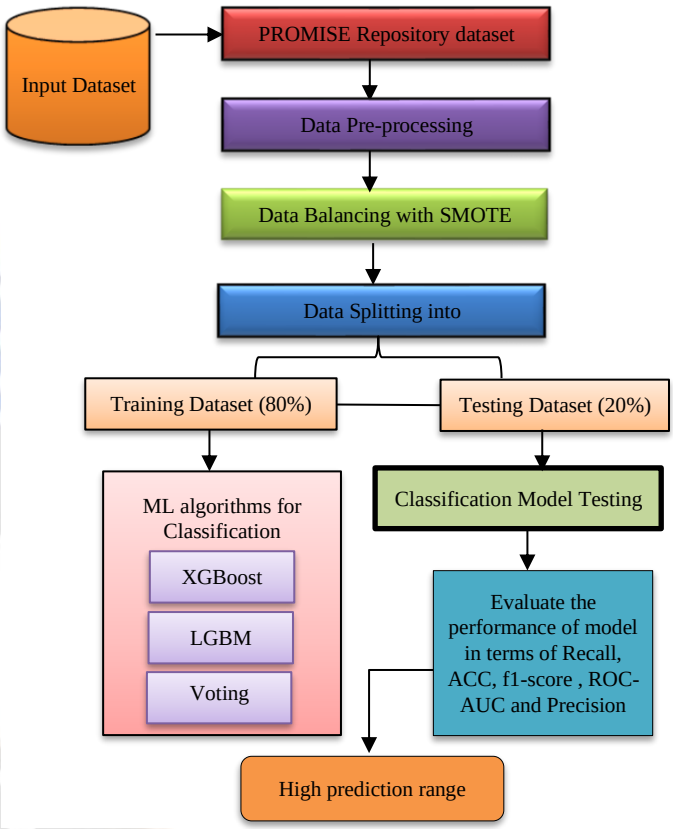

Figure 2. Flow chart of suggested model

A flowchart of the suggested method is illustrated in Fig 2. Initially, the 16 NASA PROMISE repository dataset is uploaded. This dataset has already been prepared & can be accessed online and through Kaggle. After the dataset has been pre-processed and standardised, it must be divided into a training set and a testing set. The results in regards to accuracy and AUC are then analysed using ML-based classification approaches. The all process of research methodology and Flow chart deeply described below:

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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

I. Data Collection

In the initial phase of SFP, data collection is carried out. The data used for the training in this study came from the tera - Promise data source. In this research, there are 16 different datasets has used for software fault prediction. The data has been collected from the PROMISE repository.

II. Data Pre-Processing

Fault prediction accuracy in supervised ML approaches is highly sensitive to data used for training and its quality. Preprocessing the data before training the approaches improves their precision. The goal of data preprocessing is to prepare raw data for use in subsequent ML steps. It is the first rung of the suggested study steps. Preprocessing entails activities like data scalling and data balancing across all sources by checking for missing data and null values.

1) Check Null Values

The process of deleting null values from the collection is an essential part of data integration. Any ML technique suffers from diminished precision and performance when dealing with such missing data. Thus, prior to using an ML method, it is essential to remove any missing or invalid values from the data. A null is not zero for numeric data types or an empty string for character or datetime data types.

2) Data scaling

When data is scaled, the range of its independent variables or features is standardisedIn the realm of data processing, it goes by various names, including "data normalisation" and "data standardisation."Data scaling is often done during the pre-processing phase of training models utilising ML approaches. StandardScaler was utilised for the data scaling.

Standard Scaler: StandardScaler is often used as a stage of preprocessing in a wide variety of ML models to normalise the functional range of the input dataset. StandardScaler is useful when the input dataset has features with widely varying ranges or when the features are evaluated using various units of measure. When analysing data, the standard deviation is used for the mean. However, outliers have an impact on both the empirical mean and the standard deviation, reducing the range of characteristic values. To fix this, we must initially input the regularly utilised data into the ML model after normalisation (μ= 0, σ= 1).The formula for Standardization is as follows:

$$X' = \frac{X - \mu}{\sigma} \ldots\ldots (1)$$

μ is the average of attribute values and

σ is the average disparity among feature values.

3) Data Balancing (SMOTE)

The term "data balancing" refers to the process of modifying a dataset's class distribution so that each class is displayed with an equal or proportional amount of data. Real-world applications like software fault prediction, fraud detection, medical diagnostics, and customer churn prediction often work with imbalanced datasets. In such cases, models are more likely to be prejudiced or incorrect due to an imbalance in the quantity of training data points used to represent each class. Data balancing is a crucial issue in Promise data source prediction since the dataset is frequently unbalanced, which means that the positive class (not faulty modules) has far fewer samples than the negative class (faulty modules).Unbalanced distribution of cases across several classes can interfere with reliable prediction of malfunctioning modules. In this case, we use SMOTE, an oversampling technique [18].

Class Imbalance Handling: The graph of our dependent variable's distribution shows the following. Our prediction model is more likely to side with the majority over the minority if our target variable is highly skewed. SMOTE, a synthetic minority oversampling approach, was implemented to address this issue.

The predictive power of the dataset for the underrepresented group improves if it is more evenly distributed. The issue of class disparity is addressed by employing the Synthetic Minority Oversampling Technique (SMOTE). SMOTE operates in feature space to produce synthetic samples from the underrepresented group. Every minority class sample as well as its KNN are presented as synthetic samples along the line that connects them. Initially a random number among 0 and 1 is multiplied by the difference in feature vectors among the minority class instance under investigation and its nearest neighbour. Multiplying the feature vector under examination by its product of multiplication factors yields a synthetic instance from the minority class (Chawla et al., 2002) [19]. For the purpose of this discussion, let's assume that fi is the feature vector of the minority class sample, and the vector $fnear$ is one of fi KNN. The resulting synthetic sample fnew can be expressed as the solution to Eq. (2).

$$f_{new} = f_i + (f_i - f_{near}) \ldots (2)$$

Here, *R* is a random number between 0 and 1.

III. Data Spiiting

Unfortunately, we cannot use the dataset to train our model. If we train our model with all of the available data points, it is possible that it will produce an inaccurate prediction of a new statement. We have opted to divide our dataset 80:20 so that we may assess the performance and consistency of our model. Using the remaining 20% of the data set, we compare the model's

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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

predictions to the actual values to determine how well it performs.

IV. *Classification Models*

Now, with the amount of data growing exponentially, the rational use of big data has become the focus of enterprises to serve the future and make better decisions. Using MLmethods for predicting the commodities and sales of products has grown a hotspot for investigators and companies in the past few years.Predicting sales with the use of ML and AI is becoming increasingly common. ML is a type of method that improves software's prediction powers without requiring new instructions. The goal of ML is to create methods that, given some input data, can utilise that data to make predictions about a target output, and then, as more data becomes available, update those predictions. Supervised learning and unsupervised learning are two distinct approaches to ML. The goal of unsupervised learning is to find concise summaries of data, while the goal of supervised learning is to make accurate predictions. Finding learning strategies which perform effectively on novel data is a goal in both supervised & unsupervised settings. The results of supervised learning can be classified further into 2 types: If the output is continuous, the issue is called a regression problem, and if it is discrete, it is called a classification problem[20].

Classification is a method for grouping things into groups that are the greatest fit for the way they are constructed. The training set consists of the qualities and the class labels associated with them, and this is what the classifier is initially trained on[21]. This is the stage that entails training or learning[22][23]. The second stage, "classification," involves evaluating the classifier's efficacy using a testing dataset. Once the rules' efficacy has been evaluated, they are used to make predictions about the classes of tuples of data about which more information is needed[24]. Classification's purpose is to place an unlabeled material into one of several predetermined categories. Classification can be represented mathematically as a function, as seen below [25]:

$$C = f(X, \theta), C \in L \ldots\ldots (3)$$

Class label of the new sample is denoted by C, features are represented by X, L is the set of class labels, f(.) is the classification function, and is the set of parameters for f(.).

In this research work, we used three classification techniques that are LGBM, XGBoost, and voting classifier. All classifiers are described below:

1) *LGBM Classifier*

The Light Gradient Boosting Method (abbreviated as "Light GBM") is a fast and effective tree-based gradient enhancement method. The name "light" comes from the fact that the classifier employs a tree-based method with vertical tree growth, making it more efficient than horizontal tree-based methods. The Light gradient boosting approach is advantageous for processing large datasets since it is both time- and resource-efficient[26].Light GBM differs from other methods in that it develops trees vertically, or leaf-wise, rather than horizontally, as is the case with most other methods. For agricultural purposes, the leaf with the highest delta loss will be selected. When cultivating the same leaf, a leaf-wise strategy can be more effective at minimising waste than a level-based one[27].

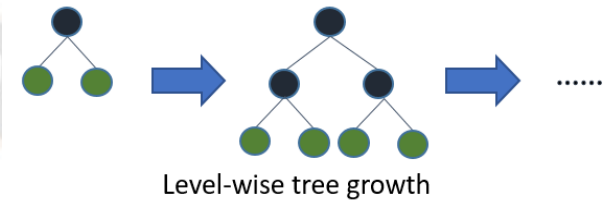

Figure 3. Level-wise tree growth in XGBOOST.

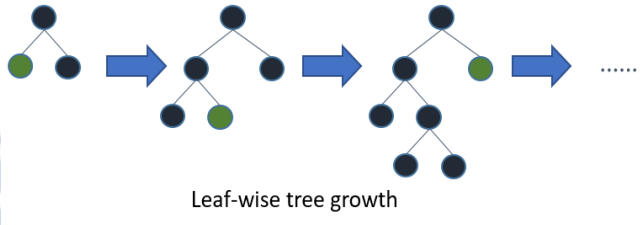

Figure 4. Leaf wise tree growth in Light GBM.

In order to prevent overfitting and the complexity explosion that results from leaf-wise splits, an additional parameter, max-depth, is specified.

2) *Xgboost Classifier*

The ML method XGBoost is quite effective. It's a brand new feature. Classified as Supervised Study. Gradient boosting serves as its conceptual backbone. XG Boost is based on a technique called parallel tree boosting, that produces accurate predictions by averaging the outputs of several relatively weak models[28].

To achieve a model with great computational speed and efficiency, we have employed XGBoost, also known as Extreme Gradient Boosting. The formula optimises its final predictions through the use of an ensemble approach which mimics the expected mistakes of some DT. The model output also includes a report on the relative importance of every feature's influence on the final performance score forecast for the building. Every characteristic's absolute value reflects the impact it has on predicting academic success.

Parallelization is supported in XGBoost through the use of a decentralised DT generator. This method's ability to analyse any huge and complex model using distributed computing is another

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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

one of its most notable features. Because it processes massive datasets with varying structures, it is performed outside of the typical computing core. This calculative model handles resource utilisation admirably. To lower the error, an additional model must be used at every stage.

XGBoost objective function at iteration t is[29]:

$$L(t)= \sum_{i=1}^{n} L(y\_out_i, y\_out1_i^{(t-1)} + f_t(x_i) + g(f_t) \quad \text{.....(4)}$$

where y_out is a known real value from the training data and the summing up part is f(x + dx) if and only if x = y_out1i (t-1).

The Taylor approximation is what we'll need to use. Let us approximate f(x) linearly by setting:

$$f(x) = f(b) + f`(b)(x-b)\,dx = ft(xi) \text{....(5)}$$

If L is the loss function, f(x) is the prediction at step t-1, b is the new learner we must absorb at step t, and dx is the x at which the prediction was made.

The Taylor approximation of the second order is:

$$f(x) = f(b) + f`(b)(x-b) + 0.5f``(b)(x-b)^2 \text{.....(6)}$$

$$L(t)= \sum_{i=1}^{n}[L(y\_out_i, y\_out1^{(t-1)}) + h_i f_t(x_i) + 0.5k_i f_t^2(x_i)] + g(f_t)$$

……(7)

Taking out the constant terms, we are left with the following goal to minimise at time step t,

$$L1(t)= \sum_{i=1}^{n}[\, h_i f_t(x_i) + 0.5k_i f_t^2(x_i)] + g(f_t) \quad \text{....(8)}$$

3) Voting Classifier

Classification models can also make use of the voting ensemble technique. Here, we take into account both the votes of the majority and the probabilities involved. There are two distinct voting systems:

- In hard voting, the outcomes of each technique are averaged based on a vote tally to improve prediction accuracy.
- Soft voting: Soft voting can be used in situations when different methods each produce a plausible outcome probability. This method optimises outcomes by averaging the odds of several different approaches. Soft voting is being used in this case. The voting system's architecture is depicted in Fig. 5.

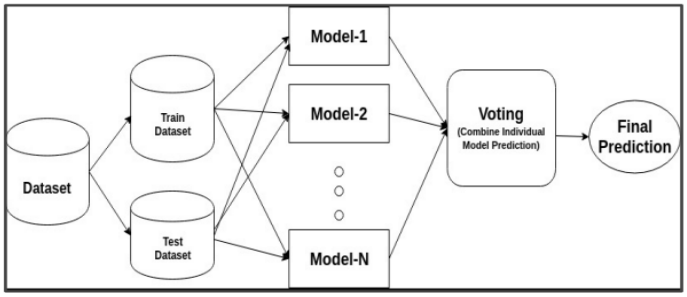

Figure 5. Architecture of voting method

The ultimate prediction of a new instance's class label is made using voting, which is based on the combined output of multiple ML classifiers. Voting could be strict or relaxed. Voting by simple majority is utilised in cases where there is a lot of opposition. In this instance, the most popular category will be chosen (guessed). When using soft voting, a forecast is formed by taking the mean of the class probabilities generated by the various classifiers. Predictions are made about the group with the highest mean likelihood. Soft voting was used in this study. Additionally, the VC's foundational estimators are tree-based ensemble classifiers[30].

IV. RESULTS AND DISCUSSION

Here, show the simulation results of predicting software faults with ML methods. For this application, we utilised the python simulation tool and a high-end HP computer equipped with an Intel Core i7 processor, 32GB of RAM, Windows 10, 24GB of Nvidia graphics memory, a 1TB hard drive, & so on. The effectiveness of an SFP ML classifier can be determined using the performance matrix. Using data from the NASA PROMISE Repository, the authors of this research suggest three ML classifiers: XGBOost, LGBM, as well as a voting classifier. Dataset description, simulation results, evaluation metrics, and the next section offers a discussion of the findings.

A. Dataset Discription

Open access to this dataset from the PROMISE Software Engineering Repository promotes reproducible, refutable, and/or upgradeable software engineering prediction models. We have employed CM1, KC1, KC2, KC3, PC1, PC2, PC3, PC4, MC2, MW1, JM1, AR1, AR3, AR4, AR5, and AR6 datasets obtained from PROMISE software engineering repository1[31], other data sets obtained from tera-PROMISE Repository2[32]. Figure 6 shows the correlation matrix of CM1 dataset. Similary other dataset features are strongly correlated, however these features were expected to be strongly correlated when they were chosen as features. Data sets contain collections of software components, every one of which has either been labelled as fp (fault prone) or nfp (not fault prone) to indicate whether or not it

[1] http://promise.site.uottawa.ca/SERepository/datasets-page.htm

[2] http://openscience.us/repo/defect/mccabehalsted

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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

is prone to errors. Table 2 details the fundamentals of each project.

TABLE II. DATASETS USED IN THIS STUDY.

s	Not Faulty Module	Faulty Module	Total no of Modules in Software	Features
CM1	449	49	498	21
KC1	1783	326	2109	21
KC2	415	107	522	21
KC3	415	43	458	39
PC1	1032	77	1109	21
PC2	5566	23	5589	36
PC3	1403	160	1563	37
PC4	1280	178	1458	37
MC2	109	52	161	39
MW1	372	31	403	37
JM1	8779	2106	10885	21
AR1	112	9	121	29
AR3	55	8	63	29
AR4	87	20	107	29
AR5	28	8	36	29
AR6	86	15	101	29

Figure 6. Correlation heatmap of dataset CM1

Our CM1 dataset feature correlation matrix is displayed in Fig. 6. A correlation heatmap is a visualisation technique that uses a color-coded matrix to show the degree to which different variables are linked to one another. A colour wheel, if you will. It reveals the degree of association between the variables. A correlation coefficient heat map that visualises the strength of associations among features.

B. Evaluation Measures for Software Fault Prediction

We will discuss the SFP sensitivity, specificity, positive predictive value, and false positive as well as negative predictive values. In software, TP indicates the sum of every situations that were correctly detected as flawed, while TN is the sum of all instances that were correctly identified as intact. One might think of FP as the number of good software instances that were wrongly identified as bad, and FN as the no. of bad software instances that were incorrectly labelled as good.

Classification accuracy, commonly known as the right classification rate, is one of the major simple measures used to evaluate the efficacy of predictive models. It's used to put a number on how many cases were correctly categorised out of the whole. To be more specific, precision is defined as the number of correctly identified defective events relative to the number of false positives. More specifically, recall measures how many instances were accurately tagged as defective (TP) relative to the overall number of defective (TP + FN) instances. F-scores, which are a harmonic mean of precision and recall, have been widely used in the academic literature. Target positive and false positive rates (TPR and FPR) are balanced to yield a ROC curve's area under the curve [4].

$$\text{Accuracy} = \frac{TP + TN}{\text{N}} \ldots\ldots (7)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \ldots\ldots (8)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FP}} \ldots\ldots (9)$$

$$\text{F}_1\text{score} = \frac{2 \times (\text{Precision} - \text{Recall})}{(\text{Precision} + \text{Recall})} \ldots\ldots (10)$$

Confusion Matrix:A classifier's propensity to favour specific classes can be revealed by examining its confusion matrix, which displays the quantity (or percentage, for normalised confusion matrices) of accurate and wrongly predicted labels for each class.

		Actual	
		Faulty Module	**Not Faulty Module**
Predicted	**Faulty Module**	TP(True positive)	FP (False positive)
	Not Faulty Module	FN (False negative)	TN (True Negative)

Figure 7. Confusion matrix

Many different metrics have been offered for two-class situations, and figure 7 shows that there are 4 distinct scenarios that can be expressed in the confusion matrix.

C. Simulation results of proposedModela

We provide the simulation results of three different machine learning boosting classifiers i.e., LGBM, XGBoost, and Voting classifier. Here we provide only CM1 datasset results visulisation, because similar kind of results obtain by the other 15 datasets. The CM1 dataset results provided below with using LGBM, XGBoost, and voting classifier. After implementation, the simulated results of these classifiers are given below as follows:

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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

1) Results of LGBM Classifier on CM1 Dataset

Here provide the simulation results of proposed LGBM classifier on CM1 dataset. Below provide the results in visualized with classification report, confusion matrix, ROC-AUC with Parameter performance.

	precision	recall	f1-score	support
0	0.96	0.85	0.90	92
1	0.86	0.97	0.91	88
accuracy			0.91	180
macro avg	0.91	0.91	0.91	180
weighted avg	0.91	0.91	0.91	180

Figure 8. Classification report of LGBM Classifier on CM1 Dataset

Classification results for the suggested LGBM classifier on the CM1 Dataset are displayed in Figure 8. In the field of ML, a classification report serves as an evaluation statistic. Our test classification model's accuracy, reliability, F1 Score, and acceptance rate can all be displayed here. Precision 96%, recall 85%, and f1-score 90% for class 0 in the input CM1 dataset; precision 86%, recall 97%, and f1-score 91% with support 92 and 88 for class 1. Classification accuracy for the suggested model LGBM classifier is 92% with 180 relevant supports.

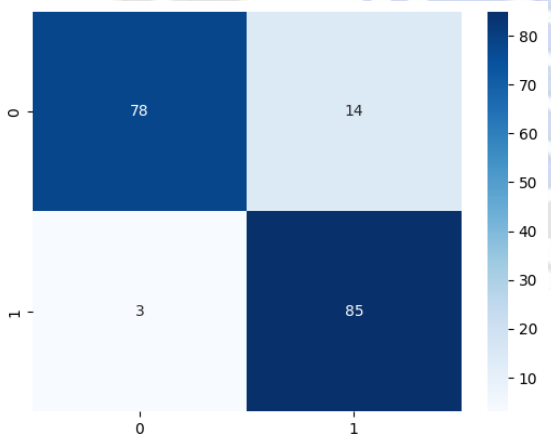

Figure 9. Confusion Matric of LGBM Classifier on CM1 Dataset

LGBM classifier's confusion matrix on CM1 dataset is depicted in figure 9. The performance of a classifier can be measured via a "confusion matrix," which is essentially a data table. Confusion matrices are useful for visualising and summarising a classification method's performance. The TP, FP, FN, and TN for the four possible examples may be found in the confusion matrix; these metrics have all been proposed for two-class problems. The modules that were appropriately recognised as defective (TP) and functional (TN) are indicated. When the outcome is projected to be yes when it is not, this is called a FP. Whenever the actual result turns out to be positive, but the forecaster wrongly assumes it will be negative, they have made a FN prediction. The matrix has the following values: false negative 3, false positive 14, true negative 78, and true positive 85.

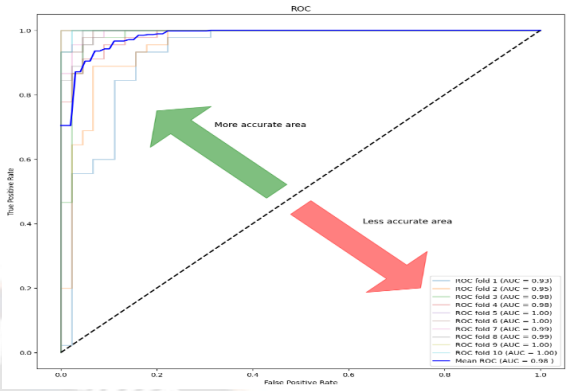

Figure 10. ROC Graph of LGBM Classifier on CM1 Dataset

Figure 10 illustrates a receiver operating characteristic (ROC) curve for the LGBM classifier on the CM1 dataset at k-fold. The ROC chart depicted above shows how the TPR and FPR change with respect to one another across a range of categorization cutoffs. The FPR is plotted along the x-axis and the TPR along the y-axis in this diagram. The True Positive Rate (TPR) measures how often the model correctly identifies positive events. It is also known as sensitivity or recall. FPR measures how often real-world negative examples are wrongly classified as positive by the model. The diagonal line on the ROC graph represents a random classifier or a model with no discrimination ability. This figure obtained the highest AUC values i.e., 0.93 on fold 1, 0.95 on fold 2, 0.98 on fold 3 and fold 4, 1.00 on fold 5, 99% AUC of fold 6 99%, fold 8 and fold 7 AUC is 98%, fold 9 and fold 10 is 100% AUC, and the mean ROC value is 0.98, respectively.

2) Results of XGBoost Classifier on CM1 Dataset

Here provide the simulation results of proposed XGBoost classifier on CM1 dataset. Below provide the results in visualized with classification report, confusion matrix, ROC-AUC with Parameter performance.

	precision	recall	f1-score	support
0	0.95	0.87	0.91	92
1	0.88	0.95	0.91	88
accuracy			0.91	180
macro avg	0.91	0.91	0.91	180
weighted avg	0.91	0.91	0.91	180

Figure 11. Classification report of XGBOOST Classifier on CM1 Dataset

Figure 11 displays the results of the suggested XGBOOST classifier's application on the CM1 dataset. The input CM1 dataset have two classes, for class 0 precision 96%, recall 87% and f1-score is 91% whereas for class 1 precision 88%, recall 95% and f1-score is 91% with support 92 and 88. The proposed

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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

model XGBOOST classifier classification accuracy is 92% with support 180 respectively.

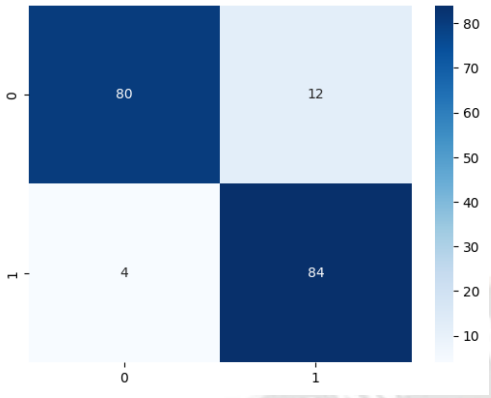

Figure 12. Confusion Matric of XGBoost Classifier on CM1 Dataset

The following figure 12 shows the confusion matrix of XGBOOST classifier on CM1 dataset. In this case, the matrix has true negative and positive values of 80 and 84, respectively, with false negative and false positive values of 4 and 12, correspondingly.

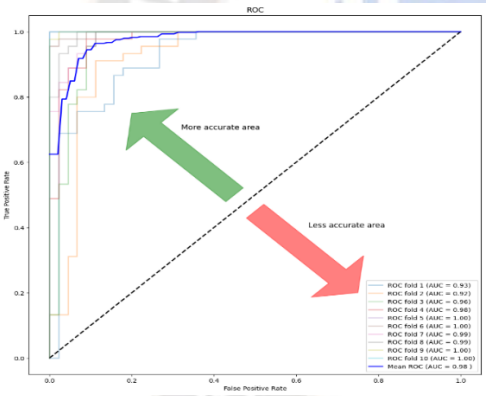

Figure 13. ROC Graph of XGBOOST Classifier on CM1 Dataset

The above figure 13 shows a ROC graph of XGBOOST classifier on CM1 dataset with k-fold (10). This figure obtained the highest AUC values i.e., 0.93 on fold 1, 0.92 on fold 2 AUC, 0.96 and 98% on fold 3 and fold 4, 1.00 on fold 5 and 6, 99% AUC of fold 7 and 8, fold 9 and fold 10 is 100% AUC, and the mean ROC value is 0.98, respectively.

3) Results of Voting Classifier on CM1 Dataset

Here provide the simulation results of proposed voting classifier on CM1 dataset. Below provide the results in visualized with classification report, confusion matrix, ROC-AUC with Parameter performance.

	precision	recall	f1-score	support
0	0.98	0.86	0.91	92
1	0.87	0.98	0.92	88
accuracy			0.92	180
macro avg	0.92	0.92	0.92	180
weighted avg	0.92	0.92	0.92	180

Figure 14. Classification report of Voting Classifier on CM1 Dataset

Classification results for the suggested vote classifier on the CM1 dataset are displayed in Figure 14 above. The input CM1 dataset have two classes, for class 0 precision 98%, recall 86% and f1-score is 91% whereas for class 1 precision 87%, recall 98% and f1-score is 91% with support 92 and 88. The proposed model voting classifier precision, recall, f1-score and accuracy is 92% with support 180 respectively.

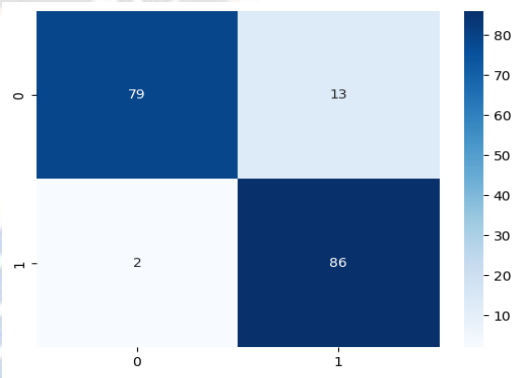

Figure 15. Confusion Matric of Voting Classifier on CM1 Dataset

The following figure 15 shows the confusion matrix of voting classifier on CM1 dataset. The matrix has the following values: 79 for true negative, 86 for true positive, 2 for false negative, and 13 for false positive.

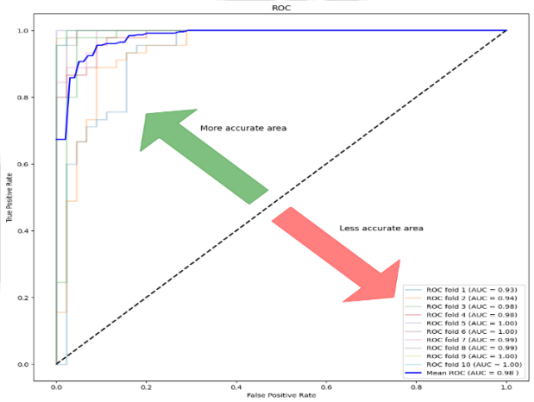

Figure 16. ROC Graph of Voting Classifier on CM1 Dataset

The above figure 16 shows a ROC graph of voting classifier on CM1 dataset with k-fold. This figure obtained the highest AUC values i.e., 0.90 on fold 1, 0.93 on fold 2, 0.98 on fold 3, and fold 4, 1.00 on fold 5, fold 6, fold 8, fold 9, and fold 10, 0.97 on fold 7 and the mean ROC value is 0.97, respectively.

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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

D. Parameter performance of ML proposed classifier with CM1 datasets

The following table 2 shows parameter performance of voting classifier using four datasets with five performance measure. This proposed models obtains 92% accuracy and 98% AUC with CM1 Dataset for SFD.

TABLE III. PARAMETER PERFORMANCE OF PROPOSED CLASSIFIERS WITH CM1 DATASET

Parameters	Voting	LGBM	XGBoost
Accuracy	92	91	91
Precision	92	91	91
Recall	92	90	91
F1-Score	92	90	91
AUC	98	98	98

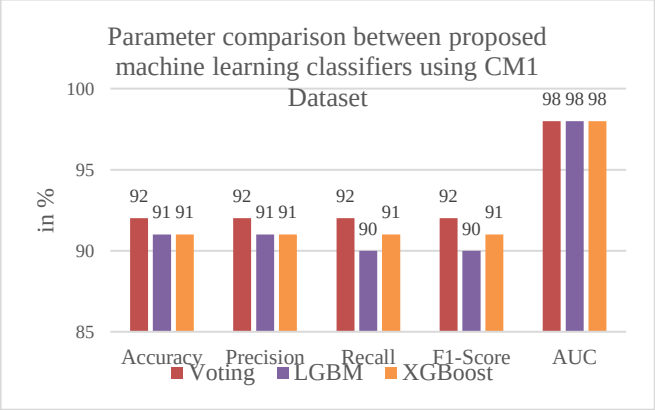

Figure 17. Bar graph of parameter comparison between threeproposed classifiers using CM1dataset

The above figure 17 shows the comparison between accuracy, precision, recal, f1-score and ROC parameter performance for SFP using XGBoost, LGBM and voting classifiers. The proposed XGBoost obtain 91%, 90% and 98% of accuracy, precision, recall, f1-score and AUC, while propose LGBM and voting classifier get 91% and 92% accuracy, precision, recall, and f1-score or 98% AUC respectively.

V. Compariosn between base and proposed classifiers using 16 NASA datasets

In this section gives the comparison between various machine learning datasets using 16 NASA software fault datasets. This work proposed three machine learning classifier that is LGBM, XGBoost and voting classifier. Here provide the implementation results comparison in terms of Accuracy and AUC parameters using python tool, also provide the comparison between base and purpose ML models for software fault prediction using PROMISE Software Engineering Repository datasets. The below table 4 and 5 provide the accuracy and AUC performance of each dataset with base (Base Voting, Naïve Bayes, SVM, and RF) and propose classifiers classifiers(LGBM, XGBoost, and propose Voting).

TABLE IV. ACCURACY (%) COMPARISON BETWEEN BASE AND PROPOSED MACHINE LEARNING CLASSIFIERS ON 16 DATASETS

Datasets	Base Classifiers				Proposed clasifier		
	Voting	NB	SVM	RF	LGBM	XGBoost	Voting
AR1	88	88	84	84	98	97	97
AR3	84	90	84	84	98	98	98
AR4	81	81	81	81	94	93	94
AR5	87	85	75	62	93	100	97
AR6	90	85	90	87	97	97	98
PC1	93	91	90	92	99	98	99
PC2	99	97	99	99	100	100	100
PC3	88	21	89	89	99	96	98
PC4	89	88	88	90	99	99	99
KC1	84	80	85	85	94	91	94
KC2	84	83	84	86	92	90	90
KC3	86	79	90	89	99	98	98
MW1	91	82	92	91	99	98	99
MC2	71	72	69	72	90	90	91
JM1	82	79	80	81	92	88	91
CM1	88	84	89	88	98	96	97

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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

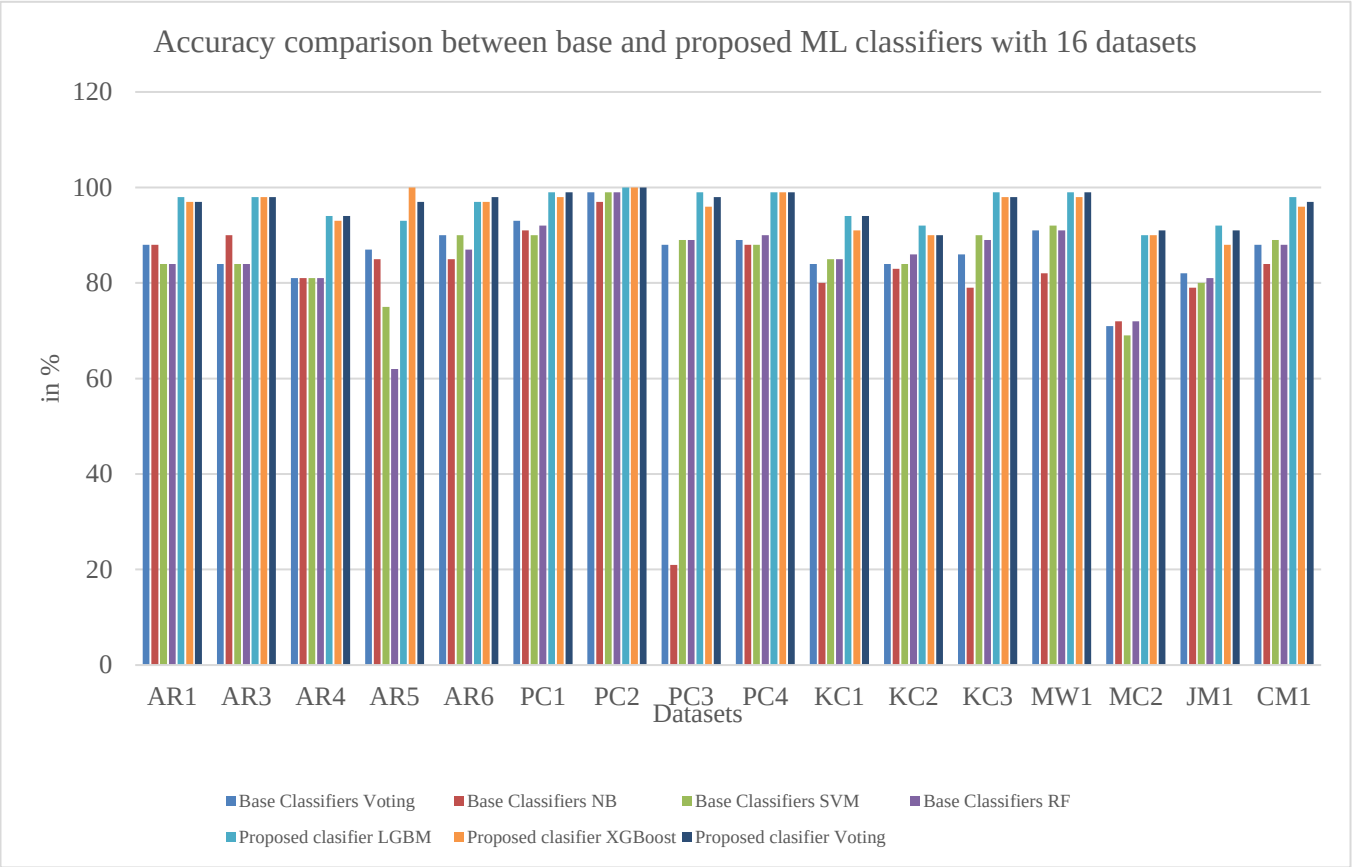

Figure 18. Bar graph of Accuracy comparison between base and propose machine learning classifiers using 16 datasets

In figure 18 shows the accuracy comparioson between base and propose classifiers with 16 datasets. The x-axis of the graph depicts the ML classifiers together with the dataset, while the y-axis depicts the corresponding accuracy percentage (%). For the MW1 dataset 93% and 90% accuracy, 89% and 83% accuracy of JM1 dataset, 91% and 92% accuracy given by the LGBM and Voting classifier with CM1 dataset, AR1 dataset get 91% and 93% accuracy, 86% and 88% with MC2 dataset, accuracy of 95% and 92% with PC1 dataset, PC3 obtain 93% and 94%, PC4 get 94% and 95%, PC2 achieved 98% and 99%, KC1 accuracy with 88% and 87%, AR6 get 94% and 98% accuracy, AR5 dataset 99% and 100% accuracy, KC2 accuracy 90% and 91%, KC3 dataset obtain 94% accuracy of voting classifier, 95% accuracy of LGBM classifier but XGBoost get only 93% accuracy, AR3 91%, 93% and 95% accuracy with three classifiers, and AR4 data get 82% and 85% accuracy with Voting , LGBM and XGBoost classifiers. The proposed methodology gets 100% accuracy with AR5 dataset using XGBoost dataset. While base base random forest only achieved 62% accuracy on AR5 dataset, base naïve Bayes only achieved 21% accuracy on PC3 dataset, base voting only achieved 79% accuracy on KC3 dataset, ans base SVM only achieved 69% accuracy on MC2 dataset, all base classifier shows very lower performance in comparison to propose classifiers on 16 datasets.

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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

TABLE V. AUC (%) COMPARISON BETWEEN BASE AND PROPOSED MACHINE LEARNING CLASSIFIERS ON 16 DATASETS

Datasets	Base Classifiers				Proposed classifier		
	Voting	NB	SVM	RF	LGBM	XGBoost	Voting
AR1	71	53	30	74	98	97	97
AR3	92	84	85	94	98	98	98
AR4	78	83	57	81	94	93	94
AR5	90	87	90	87	93	100	97
AR6	71	79	40	62	97	97	98
PC1	80	71	47	81	99	98	99
PC2	76	86	53	79	100	100	100
PC3	82	70	31	85	99	96	98
PC4	92	82	50	94	99	99	99
KC1	78	79	55	76	94	91	94
KC2	79	83	82	79	92	90	90
KC3	84	81	50	84	99	98	98
MW1	76	75	44	77	99	98	99
MC2	71	68	54	75	90	90	91
JM1	79	66	64	68	92	88	91
CM1	73	75	41	70	98	96	97

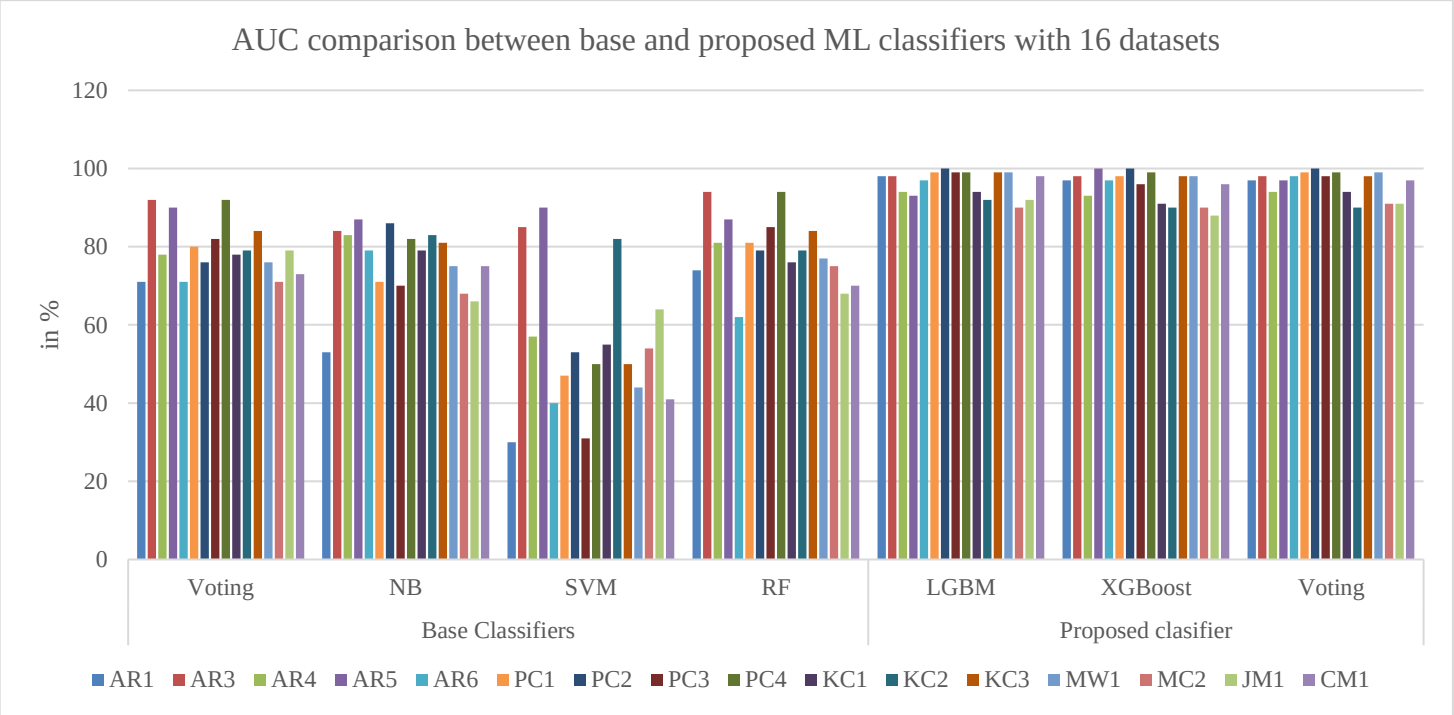

Figure 19. Bar graph of AUC comparison between base and propose machine learning classifiers using 16 datasets

The abovefigure19 provide the AUC performance of each dataset with proposed and base classifiers. Also, tabular representation provide table5. For the MW1 dataset 99% and 98% AUC, 91% and 92% AUC of JM1 dataset, 97% and 98% AUC given by the LGBM and Voting classifier with CM1 dataset, AR1 dataset get 97% and 98% AUC, 91% and 90% with MC2 dataset, similarly other dataset get highest AUC with proposed Voting, LGBM and XGBoost classifiers. The proposed methodology gets 100% AUC with PC2 dataset using all three methods datasets. Proposed LGBM classifier get 99% AUC on KC3 dataset that is higher than the other machine learning classifier while base SVM only achieved 50% AUC on KC3 dataset, proposed LGBM and Voting classifier obtain 99% accuracy on this dataset while base models only 76%, 75%, 45%

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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

and 77% accuracy that is very lower in our propose models. MW1 dataset obtain accuracy of LGBM 99%, XGBoost 98%, and Voting 99%, JM1 dataset obtain AUC of LGBM 92%, XGBoost 88%, and Voting 91%,and base models only 79%, 66%, 64% and 68% AUC that is very lower in our propose models. proposed LGBM classifier obtain 98% accuracy on this dataset while base models only 73%, 75%, 41% and 70% AUC that is very lower in our propose models. CM1 dataset obtain AUC of LGBM 98%, XGBoost 96%, Voting 97%, Stacking 98%, Gradient 97%, AdaBoost 93% and CatBoost 94% respectively. Show we can see that our proposed models AUC show very good performance in comparison to base classifiers using 16 datasets.

V. CONCLUSION AND FUTURE WORK

ML methods make use of example data or past algorithms to solve a given problem. In our study, we want to use ML to predict whether a specific module will be faulty or not based on the previous fault data collected and the metrics collected from the project. The goal of this research was to compare and contrast several well-known ML techniques for finding defects in computer programmes. In this thesis, we propose a novel approach utilizing ML for software defect prediction. The goal of the approach is to make use of label information for improving the performance of classification algorithms such as LGBM, XGBoost, and Voting classifiers. The performances of different algorithms were evaluated using classification accuracy, recall, precision, F-measure, and AUC metrics. The suggested ML classifiers on all 16 NASA datasets. ML models have reached 100% accuracy and AUC almost all datasets. In terms of area under the curve (AUC) and accuracy, the suggested technique performs exceptionally well across the board for fault data sets. Therefore, in the context of software failure prediction across a wide range of software projects, the suggested strategy is efficient, reliable, and consistent. Our evaluation on a widely used data set shows that our method significantly improves the performance of ML classifier. We provide research into many ML methods which have been shown to be effective in the prediction of software faults, and we analyse an examination of the different performance indicators utilised to make such predictions.

Interesting future extensions could include studying the impact of various metaheuristic feature selection approaches to select the optimal set of features for SFP. As data imbalance is still an issue that adversely impacts the performance of the existing SFP approaches, future research should look into DL methods and ensemble classifiers and contrast their results to those of other resampling methods. More research is needed to see if one of the software defect prediction models might be useful for quality assurance purposes in the setting of a software engineering organisation.

REFERENCES

[1] J. Xu, D. Ho, and L. F. Capretz, “An Empirical Study on the Procedure to Derive Software Quality Estimation Models,” Int. J. Comput. Sci. Inf. Technol., 2010, doi: 10.5121/ijcsit.2010.2401.

[2] S. Aleem, L. F. Capretz, and F. Ahmed, “Benchmarking Machine Learning Techniques for Software Defect Detection,” Int. J. Softw. Eng. Appl., 2015, doi: 10.5121/ijsea.2015.6302.

[3] 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.

[4] 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.

[5] M. Caulo and G. Scanniello, “A Taxonomy of Metrics for Software Fault Prediction,” in 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2020, pp. 429–436. doi: 10.1109/SEAA51224.2020.00075.

[6] C. Catal, “Software fault prediction: A literature review and current trends,” Expert Systems with Applications. 2011. doi: 10.1016/j.eswa.2010.10.024.

[7] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, “A systematic literature review on fault prediction performance in software engineering,” IEEE Transactions on Software Engineering. 2012. doi: 10.1109/TSE.2011.103.

[8] K. Tumer and J. Ghosh, “Analysis of decision boundaries in linearly combined neural classifiers,” Pattern Recognit., vol. 29, no. 2, pp. 341–348, 1996, doi: https://doi.org/10.1016/0031-3203(95)00085-2.

[9] P. Singh and S. Verma, “Multi-classifier model for software fault prediction,” Int. Arab J. Inf. Technol., 2018.

[10] M. Z. Mohammed and I. A. Saleh, “Predicted of Software Fault Based on Random Forest and K-Nearest Neighbor,” in 2022 4th International Conference on Advanced Science and Engineering (ICOASE), 2022, pp. 43–48. doi: 10.1109/ICOASE56293.2022.10075596.

[11] T. M. Phuong Ha, D. Hung Tran, L. E. T. My Hanh, and N. Thanh Binh, “Experimental Study on Software Fault Prediction Using Machine Learning Model,” in 2019 11th International Conference on Knowledge and Systems Engineering (KSE), 2019, pp. 1–5. doi: 10.1109/KSE.2019.8919429.

[12] K. Phung, E. Ogunshile, and M. Aydin, “A Novel Software Fault Prediction Approach To Predict Error-type Proneness in the Java Programs Using Stream X-Machine and Machine Learning,” in 2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT), 2021, pp. 168–179. doi: 10.1109/CONISOFT52520.2021.00032.

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.7710
***Article Received:** 09 June 2023 **Revised:** 28 July 2023 **Accepted:** 13 August 2023*

[13] C. Anjali, J. P. M. Dhas, and J. A. P. Singh, "A Study on Predicting Software Defects with Machine Learning Algorithms," in 2022 International Conference on Intelligent Innovations in Engineering and Technology (ICIIET), 2022, pp. 195–198. doi: 10.1109/ICIIET55458.2022.9967593.

[14] E. Elahi, S. Kanwal, and A. N. Asif, "A new Ensemble approach for Software Fault Prediction," in 2020 17th International Bhurban Conference on Applied Sciences and Technology (IBCAST), 2020, pp. 407–412. doi: 10.1109/IBCAST47879.2020.9044596.

[15] A. Joon, R. Kumar Tyagi, and K. Kumar, "Noise Filtering and Imbalance Class Distribution Removal for Optimizing Software Fault Prediction using Best Software Metrics Suite," in 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020, pp. 1381–1389. doi: 10.1109/ICCES48766.2020.9137899.

[16] S. R. Aziz, T. Khan, and A. Nadeem, "Experimental validation of inheritance metrics' impact on software fault prediction," IEEE Access, 2019, doi: 10.1109/ACCESS.2019.2924040.

[17] K. Yang, H. Yu, G. Fan, X. Yang, S. Zheng, and C. Leng, "Software Defect Prediction Based on Fourier Learning," 2018. doi: 10.1109/PIC.2018.8706304.

[18] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," 2008. doi: 10.1109/IJCNN.2008.4633969.

[19] M. S. Zulfiker, N. Kabir, A. A. Biswas, T. Nazneen, and M. S. Uddin, "An in-depth analysis of machine learning approaches to predict depression," Curr. Res. Behav. Sci., 2021, doi: 10.1016/j.crbeha.2021.100044.

[20] D. Barber, Bayesian Reasoning and Machine Learning. 2012. doi: 10.1017/cbo9780511804779.

[21] S. Geva and J. Sitte, "Adaptive Nearest Neighbor Pattern Classification," IEEE Trans. Neural Networks, 1991, doi: 10.1109/72.80344.

[22] S. J. Hong, "R-MINI: An iterative approach for generating minimal rules from examples," IEEE Trans. Knowl. Data Eng., 1997, doi: 10.1109/69.634750.

[23] E. W. T. Ngai, L. Xiu, and D. C. K. Chau, "Application of data mining techniques in customer relationship management: A literature review and classification," Expert Syst. Appl., vol. 36, no. 2, Part 2, pp. 2592–2602, 2009, doi: https://doi.org/10.1016/j.eswa.2008.02.021.

[24] J. R. Quinlan, "Generating production rules from decision trees," Proc. Tenth Int. Jt. Conf. Artif. Intell., 1987.

[25] Y. Ren, L. Zhang, and P. N. Suganthan, "Ensemble Classification and Regression-Recent Developments, Applications and Future Directions [Review Article]," IEEE Computational Intelligence Magazine. 2016. doi: 10.1109/MCI.2015.2471235.

[26] A. M. S. Sathya Bama, "Identification of Default Payments of Credit Card Clients using Boosting Techniques," Int. J. Recent Technol. Eng., 2020, doi: 10.35940/ijrte.f8897.038620.

[27] M. Fatima and M. Pasha, "Survey of Machine Learning Algorithms for Disease Diagnostic," J. Intell. Learn. Syst. Appl., 2017, doi: 10.4236/jilsa.2017.91001.

[28] A. Gupta, S. Sharma, S. Goyal, and M. Rashid, "Novel XGBoost Tuned Machine Learning Model for Software Bug Prediction," 2020. doi: 10.1109/ICIEM48762.2020.9160152.

[29] and R. R. Purvika Bajaj, "SALES PREDICTION USING MACHINE LEARNING ALGORITHMS," Int. Res. J. Eng. Technol., vol. 07, no. 06, pp. 1–7, 2020.

[30] E. K. Ampomah, Z. Qin, and G. Nyame, "Evaluation of tree-based ensemble machine learning models in predicting stock price direction of movement," Inf., 2020, doi: 10.3390/info11060332.

[31] S. Tong, Haonan; Liu, Bin; Wang, "Benchmark data sets," Mendeley Data, vol. V1, 2017, doi: 10.17632/923xvkk5mm.1.

[32] T. J. Sayyad Shirabad, J. and Menzies, "The PROMISE Repository of Software Engineering Databases," University of Ottawa, 2005.

[33] Bhawana Verma, S. K.A. (2019). Design & 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

[34] 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