Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

A NASA Dataset Based Automated Software Bug Prediction Model Using Feature Reduction Techniques

Amit Kumar Sahoo
Department of M.Sc.(CS)
NIIS Institute of Information Science and Management Bhubaneswar, India

Aditya Kumar Nayak
Department of M.Sc.(CS)
NIIS Institute of Information Science and Management Bhubaneswar, India

Ankita Nayak
Department of M.Sc.(CS)
NIIS Institute of Information Science and Management Bhubaneswar, India

Debasish Pradhan
Department of MCA
NIIS Institute of Business Administration Bhubaneswar, India

***Abstract*— Testing is an important step in the application creation process, aimed at identifying and correcting mistakes or defects introduced by the programmers. Tackling these problems later in the application development lifecycle might have a larger impact. To reduce this, early discovery of errors is critical, as is maximizing the use of testing resources. During fault estimation, software components are classified as fault-prone or non-defect-prone. This work aims to improve the efficiency and accuracy of detecting fractured software modules using a hybrid method. The approach includes preliminary processing, features reduction in dimensionality, and classifier. The model is evaluated using a publicly accessible dataset from NASA. PCA-LDA is a combination of PCA, or Principal Component Analysis, and LDA, or Linear Discriminant Analysis, that is used to reduce the dimensionality of vector features. The expected outcome rate is determined using the AdaBoost boosting approach with a random forest (ABR). Several performance measures are examined to verify the model that is suggested, include precision, sensitiveness, specificity, the F1 score, and MCC. The average accuracy of the PCA+ABR algorithm on the PC2 dataset is 0.9919. The experimental findings show that the proposed model outperforms current models in fault prediction accuracy.**

***Keywords*— Software Defect Prediction, Principal Component Analysis, Random Forest, AdaBoost, ABR, NASA Dataset**

I. INTRODUCTION

In order to anticipate and detect possible software flaws prior to deployment, sophisticated computational approaches are utilized in the introduction to machine learning-based software bug prediction. Machine learning models—that is, classification algorithms—are used to examine past project data and identify trends associated with the presence of defects. These models use key metrics as input characteristics, such code complexities, code churn, and development experience. By automating the detection process and giving developers insights into high-risk regions that could need more examination, machine learning is used to the prediction of bugs. Teams may improve overall software quality, allocate resources more efficiently, and proactively address possible problems by incorporating the use of machine learning for bug predictions into the application development method. Notwithstanding difficulties brought on by the unpredictable characteristics of the development of software, machine learning offers a promising approach to minimize post-release defects and streamline the debugging process.

utilizing cutting-edge computational approaches, software fault prediction utilizing machine learning aims to detect and predict any flaws or mistakes in software systems prior to deployment. The primary aim is to handle software dependability in a proactive manner by utilizing machine learning models, namely classification algorithms, to examine past project data and identify trends linked to the incidence of errors. These models usage a diversity of metrics as input characteristics, including code complexity, code updates, and previous fault occurrences. Software fault calculation uses machine learning to automate the recognition of trouble regions so that developers may concentrate on important components throughout the development process. Testing efficiency is increased when machine learning-based defect prediction is incorporated into the application creation process.

The application of machine learning (ML) in bug forecasting has become a proactive approach in software development to improve software quality. Conventional bug prediction is based on the examination of past data, whereas machine learning

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

techniques offer an evolving and automatic method. Based on measures like complexity of code and knowledge of developers, ML models, in particular classification algorithms, can detect possible bug-prone regions by looking for patterns and links in data. With this method, engineers may quickly resolve problems, manage resources effectively, and expedite the debugging process. A state-of-the-art development in software engineering, ML-driven bug prediction encourages a more proactive and economical use of resources in order to guarantee stable and dependable software systems.

The potential of machine learning to transform software development, guaranteeing improved dependability and fewer post-release problems, makes it an important tool for predicting software bugs. Machine learning models may discover possible bug-prone locations by utilizing complex algorithms and historical data. This enables developers to efficiently allocate resources and prioritize testing efforts. By taking a proactive stance, major errors are less likely to occur, which improves both software performance and customer satisfaction. In addition to automating the detection of possible problems, machine learning also adjusts to changing project conditions, offering a data-driven and dynamic approach to software bug management. By using machine learning into bug prediction, developers may anticipate problems before they arise, optimize development processes, and provide end users with software solutions that are more reliable and durable.

II. LITRETURE REVIEW

In the year of 2023 Das et al. [1] executed AB, RF, NB, J48, MLP and ADRF machine learning algorithm in JM1, PC5, PC4, MC1 and KC1 datasets to develop a vastly accurate method identified as PCA+ADRF for identifying software faults in specific modules and got nearly0. 985 accuracy in it. In the year of 2018, Hammouri et al. [2] published a paper on “Software Bug Prediction using Machine Learning Approach” . The experimental results showed that the ML approach outperformed other approaches, such as linear AR and POWM models, in terms of prediction model performance. Three different supervised machine learning algorithms were employed for forecasting future software defects by leveraging historical data. These algorithms include Naïve Bayes (NB), Decision Tree (DT) and Artificial Neural Networks (ANN). These algorithms were implements in the datasets DS1, DS2 and DS3 in 0.93 accuracy. In 2019, Tian et al. [3] published a paper on “Software Defect Prediction based on Machine Learning Algorithms”. The paper begins by outlining the concept of software defect prediction, with a subsequent emphasis on the machine. They use Naïve Bayes, Ensemble Learners, Neural Networks, SVM classifiers and implements in JM1 dataset. In the year 2018, Hammouri et al. [4] examined the application of machine learning algorithms for predicting software bugs. He used various machine learning methods like NB, DT, ANNs. He applied these classifiers in the datasets i.e. DS1, DS2 and DS3 having accuracy of 0.93. In 2019, Iqbal et al. [5] published a paper on “Performance Analysis of Machine Learning Techniques on Software Defect Prediction using NASA Datasets”. The research results can serve as a baseline for future studies, allowing for easy comparison and verification of proposed techniques, models, or frameworks. The classifiers includes: NB, MLP, RBF, SVM, KNN, kStar, OneR, DT, RF. These algorithms were implements in the dataset PC2 in 0.976959 accuracy. In the year 2021 Mustaqeem et al. [6] Present a novel approach that involves combining two highly promising algorithms for optimization and feature selection with the aim of achieving. He used classifiers, that are SVM, PCA which are implemented in datasets CM1 and KC1 and he got 0.9520 accuracy in that . In 2020, Rathore et al. [7] published a paper on “An empirical study of ensemble techniques for software fault prediction”. The findings presented in this paper may prove beneficial to the research community by aiding in the development of precise fault prediction models through the selection of suitable ensemble techniques. They used Naive Bayes, logistic regression, J48 algorithms, which were implemented in Ant, Camel, Jedit, Lucene, Poi, Prop, Tomcat, Xalan and Xerces datasets in accuracy of 0.8848. In 2019, Wang et al. [8] published a paper on “A cluster-based hybrid feature selection method for defect prediction”. In this research, the authors introduced a novel approach for feature selection, which combines filter and wrapper methods in a hybrid manner to address the issue. This method defines a feature quality coefficient using spectral clustering and utilizes sequential forward selection to derive the ultimate feature subset. They use K-Nearest Neighbor, Decision Tree and Random Forest classifiers and implements in Camel, Jedit, Lucene, Synapse, Xerces dataset. In 2017, Li et al. [9] published a paper on “Software Defect Prediction via Convolutional Neural Network”. The paper focuses on predicting code defects in software implementation to reduce the workload of software maintenance and improve reliability. They use CNN classifier and implements in xerces dataset in 0.845 accuracy. In 2019, Yalciner et al. [10] published a paper on “Software Defect Estimation Using Machine Learning Algorithms”. In this study, the authors assessed the performance of machine learning algorithms in predicting software defects and identified the top-performing category by evaluating seven different machine learning algorithms using four NASA datasets. They use Bayesian Learners, Ensemble Learners, SVM and Neural Networks classifiers and implements in PC1, CM1, KC1 and KC2 datasets in 0.94 accuracy.

III. PROPOSED WORK

Figure 1 shows the block figure for the future work, which includes six independent units: NASA dataset, data pre-processing, dimensionality reduction, random forest, AdaBoost, and performance evaluation.

Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

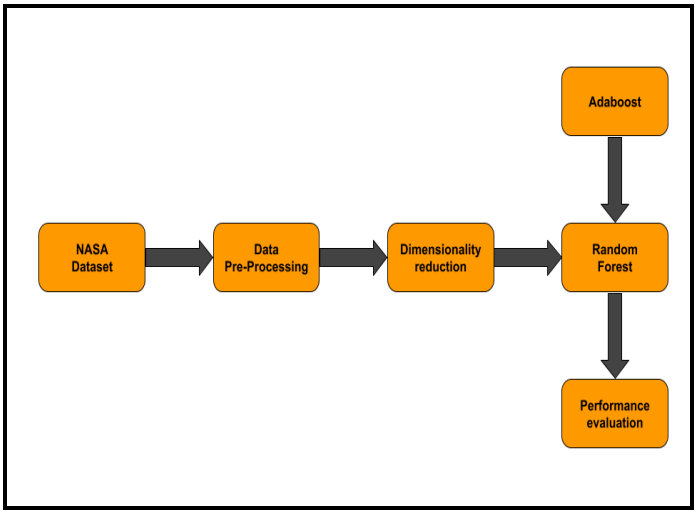

Fig 1. Block drawing for the proposed model

A. Dataset

We used the NASA Dataset, which is made up of fourteen distinct datasets, from the PROMISE repository for our research project[11].

For our experiment, we have utilized five of the fourteen datasets available. The following table provides the datasets' details.

TABLE I. DATASET DETAILED

Dataset	Total Instance	Non defective instance	Defective instance
KC1	1183	869	314
MC2	125	81	44
PC1	705	644	61
PC2	745	729	16
MW1	253	226	27

B. Feature normalization

Larger scale features might predominate during learning, which would skew model predictions. Regardless of their scales, normalization guarantees that every feature contributes equally to the model's learning process. When features are similar in size, many machine learning techniques, such optimization algorithms based on gradient descent, converge more quickly. Normalization makes model training faster and more smoothly convergent. Normalization facilitates the interpretation of the significance of the model's features. It could be difficult to compare the relative relevance of features that are scaled differently without normalization. By bringing all characteristics to a comparable scale, normalization can lessen the influence of outliers in the dataset and increase the model's resilience to extreme values.

$$B_{standardized}=\frac{B-\mu}{\alpha}$$

$B_{standardized}$ denotes the feature's lowest value in the dataset, where B is the initial feature value.

C. Dimensionality Reduction

The act of lowering the quantity of variables or structures while keeping relevant information is known as "dimensionality reduction" in the field of software error forecast 4using machine learning. In software engineering, where several measurements and characteristics are used to characterize software systems, maintaining high-dimensional information is commonplace, and this reduction is essential. Model interpretability, generalization performance, and efficiency of training are all improved when dimensionality is lowered since it lowers the difficulty of the statistics. Techniques for reducing the dimensions of a dataset seek to preserve important information linked to fault prediction while identifying the most useful characteristics and removing superfluous or unnecessary ones to provide a compact depiction of the original dataset.

1) PCA

Principal A dimensionality reduction technique entitled Principal Component Analysis (PCA) is frequently applied in machine learning to enhance model performance and lower the chance of over fitting. When it comes to software bug prediction, principal component analysis (PCA) may be utilized to pinpoint the key elements that influence the likelihood of defects occurring.

One research suggested employing several dimensionality approaches, such as PCA and Kernel PCA, to extract expressive features from an underlying arrangement of fundamental change measurements. Decision trees and artificial neural networks were then used to train a classifier based on the retrieved features. According to the study, the PCA approach helped to enhance the classifier's performance by identifying the key features that are responsible for the incidence of faults.

$$S=W\hat{}WT \quad (2)$$

PCA contributes significantly to software fault prediction and enhances software control and dependability. The most crucial elements influencing the likelihood of failure can be found by using PCA on characteristics taken from evaluation software . This decrease in size enhances the understanding of the underlying schema in addition to streamlining the dataset's complexity. Consequently, PCA aids in the development of a more effective and precise prediction model, enabling software makers to report any problems and raise the standard of their final output.

2) PCA-LDA

In the field of machine learning, methods for reducing dimensionality like Principal Component Analysis (PCA) along with Linear Discriminant Analysis (LDA) are frequently employed to enhance model performance and lower the likelihood of overfitting. When it comes to software fault estimation, PCA-LDA may be used to pinpoint the key elements that influence the likelihood of issues occurring.for

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

anticipating software bugs. According to the study, PCA-LDA was capable of to pinpoint the key elements that influence the likelihood of defects, which enhanced the models' functionality. To sum up, PCA-LDA is an effective method that may be applied to raise the accuracy of machine learning algorithms in software defect forecasting. It can assist in determining the key elements that influence the likelihood of glitches, which might improve the models' correctness.
This is how PCA-LDA is represented mathematically:
PCA-LDA = LDA(PCA(X))
while PCA stands for principal component analysis, LDA for linear discriminant analysis, and X is the input .

D. Machine learning Classification

A method for classifying data into different groups based on its properties is machine learning classification. Based on the properties of the program, machine learning classification may be used to anticipate the occurrence of problems in software.
After comparing several machine learning tactics for determining the degree and importance of software problem reports and recommend the use of random forests, or optimum decision trees, as a method for more research. In order to determine the severity and importance of new issues, the method sought to build many decision trees using subsets of the current bug dataset and characteristics. The most suitable decision trees were then chosen. The method may be used to anticipate and identify problems in distributed systems and big, intricate networks of communication.
One effective method for enhancing the efficiency of machine learning algorithms in software fault predictions is machine learning classification. It can assist in determining the key elements that influence the likelihood of bugs.

1) Adaboost

An integrated learning technique called AdaBoost is frequently used for machine learning to enhance performance of models and lower the chance of overfitting. AdaBoost may be used to forecast the likelihood of a software issue based on the characteristics of the software.
In the field of machine learning, it is 0recognized and useful for tasks requiring both regression and classification [21]. In order to provide predictions, it makes use of the combined strength of several decision trees that are based on mutual learning. Here, random samples from the data collection are used to develop several training techniques, giving the participants a choice of data. Different objects chosen from the characteristics of different training techniques should be used in this strategy to generate distinct decision trees. Combining the forecasts from each tree yields the final forecast. RF is superior to decision trees in many ways. For instance, it is less effective as it makes use of several trees as opposed to just one.
It is less susceptible to choosing the wrong hyperparameter and can also deal with outliers and missing data. Robust algorithms like Random Forests (RF) are helpful in image classification, fraud detection, bioinformatics, and other domains. It's crucial to realize, though, that RF may not be the ideal method. It is vital to thoroughly compare several models and select the unity that best fits the specific requirements of your issue.

Precision = TP / (TP + FP)

2) Random Forest

A machine learning procedure called Random Forest has become well-known for its efficiency in a number of fields, including the prediction of software bugs. Predicting and avoiding defects is crucial in software development to guarantee the dependability and quality of solutions for software. As a team-based learning approach, Random Forest has several benefits that make it a good choice for predicting software bugs.
The random forest's capacity to offer insightful information makes it special. Each characteristic is given a crucial score by the algorithm according to how well it contributes to prediction accuracy. Knowing which aspects are most important in the framework of software fault prevention can aid developers in setting priorities for their tasks through review and management.
Predicting software bugs frequently entails examining intricate connections and trends seen in software repositories. Because Random Forest can handle both numeric and categorical variables, it may be used to extract a wide range of information from databases used for issue tracking, versions management systems, and source code. In the environment of developing software, where different forms of data contribute to the emergence of bugs, this flexibility is essential.

Recall = TP / (TP + FN)

3) Adaboost + Random Forest

AdaBoost and random-forest approaches are combined in ABR, a novel method for software breakdown prediction (SFP), to increase accuracy, consistency, and address overfitting problems.. The AdaBoost algorithm is a cutting-edge boosting approach that is used to purposefully increase the accuracy of a certain learning system [22]. This method, which is based on ensemble learning, combines several weak hypotheses or classifiers with large error rates in order to create a final hypothesis that has a significantly reduced training error rate. The method is still straightforward, quick, and easy to use. Furthermore, it does not require any prior information of the poor learner and is nonparametric and skilled at recognizing outliers. As a result, this method has verified to be efficient in tackling a variety of prediction issues, especially when it comes to software failure prediction.
The AdaBoost method has been used in this particular implementation to forecast software faults. In terms of accuracy, Random Forests (RF), a well-known and powerful ensemble machine learning technique, perform better than current techniques. As a skilled bagging method, RF handles a large number of input variables without requiring variable removal. It performs well on large datasets, precisely estimating critical properties for prediction-making. Moreover, RF has a simple implementation, is easy to parallelize, and is excellent at predicting missing data. An ensemble of tree-based classifiers called RF is very resistant to noise and anomalies. Every tree in the forest is constructed from a randomly chosen vector with values that are evenly distributed and individually chosen for every tree.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

When utilizing the training dataset, each tree must meet the following characteristics in order to be created: • Samples of size S, where S is the number of samples, are randomly selected from the training set to construct trees. What are known as bootstrap samples are created by reintroducing these selected samples into the original set.

• A considerably lower number f (where f is much less than F) is specified in datasets with a total of F characteristics. A random subset of the F feature pool's features is chosen at random for every node in the tree structure. The best split, ascertained from the selected characteristics, is used to divide the nodes. During the whole process of creating a forest, the value of f remains constant.

F- Measure = (2* Recall * Precision)/(Recall + Precision)

IV. EXPERIMENTAL DISCUSSION

In order to confirm the suggested configuration, MATLAB 2017a was used to carry out a simulation on a Windows 11 OS-powered PC.
The PC had a 3.40 GHz Core-i7 CPU and 8 gigabytes of random access memory. The system's effectiveness was then assessed by comparing it to other capable methods while accounting for specificity, sensitivity, and accuracy. These measures, which are explained below, were used as standards for the evaluation.
"Sensitivity, also known as the true positive rate, is determined by correctly identifying the number of defective modules out of the total number of defective modules."

$$\text{Sensitivity} = \frac{A\_P}{A\_P+F\_N} \quad (5)$$

"Specificity, also known as the true negative rate, is calculated by dividing the count of accurately classified non-defective modules by the total number of non-defective modules."

$$\text{Specificity} = \frac{A\_P}{A\_P+F\_N} \quad (6)$$

Computes the precise identification of faulty modules in an all-encompassing way.

$$\text{Acc} = \frac{A\_P+A\_N}{A\_P+A\_N+F\_P+F\_N} \quad (7)$$

The precision (P) of machine learning-based software defect prediction may be calculated as follows: P is the ratio of appropriately projected useful observations to all expected positives.

$$\text{Precision} = \frac{A\_P}{A\_P+F\_P} \quad (8)$$

The following is the formula for a simple machine learning model that predicts software errors in Formula 1 (F1) scores, similar to a logistic reversion model: (9)

$$F1 = 2 \times \frac{(P \times R)}{(P+R)}$$

Six classifiers—NB, MLP, AB, RF, DT, J48, and ABR—as well as five distinct datasets—KC1, MC2, PC1, PC2, and MW1—are used in this instance. Three distinct phases comprise the experiment: in the first, we computed the accuracy with no feature selection; in the second, we used PCA; and in the third, we used PCA-LDA. Without FS, we used the ABR classifier with the PC2 dataset at a 5-fold magnification. Our accuracy of 0.9799 is the greatest of all the classifiers shown in Table II. The PCA+ABR classifier, which we used with the PC2 dataset and fivefold cross validation, yielded the greatest accuracy of any classifier at 0.9899, as shown in Table III.
With PCA-LDA, we used the PCA-LDA+ABR classifier in Table IV. Using the PC2 dataset and five fold validation, we obtained the greatest accuracy of any classifier—0.9919.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

TABLE II. Performance Evaluation without Feature selection

CLASSIFIER	DATASET	FOLD	SENSITIVITY	SPECIFICITY	PRECISION	F1SCORE	MCC	ACCURACY
NB	KC1	5	0.7797	0.4899	0.8838	0.8285	0.2279	0.7312
MLP	KC1	5	0.7694	0.6311	0.9563	0.8527	0.2557	0.7574
AB	KC1	5	0.7565	0.6364	0.9724	0.851	0.2042	0.7498
RF	KC1	5	0.7839	0.6268	0.939	0.8545	0.3022	0.7065
DT	KC1	5	0.7715	0.6087	0.9482	0.8508	0.2551	0.7557
J48	KC1	5	0.7781	0.5266	0.9079	0.838	0.2415	0.7422
ABR	KC1	5	0.784	0.6164	0.9356	0.8531	0.2983	0.7633
NB	MC2	5	0.7423	0.6786	0.8889	0.809	0.3674	0.7028
MLP	MC2	5	0.8	0.675	0.8395	0.8193	0.4639	0.7502
AB	MC2	5	0.7396	0.6552	0.8765	0.8023	0.3489	0.7012
RF	MC2	5	0.7551	0.6897	0.8916	0.8177	0.3923	0.7402
DT	MC2	5	0.7255	0.6957	0.9136	0.8087	0.3417	0.7103
J48	MC2	5	0.7586	0.6053	0.8148	0.7857	0.3505	0.7006
ABR	MC2	5	0.7551	0.7407	0.9136	0.8268	0.4272	0.8203
NB	PC1	5	0.9393	0.3548	0.9379	0.9386	0.2964	0.8879
MLP	PC1	5	0.9353	0.45	0.9658	0.9503	0.3171	0.9078
AB	PC1	5	0.9144	0.25	0.9953	0.9532	0.0439	0.9106
RF	PC1	5	0.9298	0.619	0.9876	0.9578	0.3319	0.9191
DT	PC1	5	0.9243	0.5	0.986	0.9542	0.2381	0.9135
J48	PC1	5	0.9352	0.439	0.9643	0.9495	0.3116	0.9064
ABR	PC1	5	0.9272	0.6111	0.9891	0.9572	0.302	0.9206
NB	PC2	5	0.9812	0.0556	0.93	0.9549	0.0657	0.9141
MLP	PC2	5	0.9811	0.4	0.9959	0.9884	0.2146	0.9772
AB	PC2	5	0.9798	1	1	0.9898	0.2475	0.9785
RF	PC2	5	0.9785	0	1	0.9891	0	0.9775
DT	PC2	5	0.9785	0.619	1	0.9891	0	0.9785
J48	PC2	5	0.9785	0.9352	0.9986	0.9885	0.9054	0.9772
ABR	**PC2**	**5**	**0.9785**	**0**	**1**	**0.9891**	**0**	**0.9799**
NB	MW1	5	0.9458	0.32	0.8496	0.8951	0.3428	0.8221
MLP	MW1	5	0.911	0.3529	0.9513	0.9307	0.214	0.8735
AB	MW1	5	0.9057	0.4444	0.9779	0.9404	0.2101	"0.8893
RF	MW1	5	0.917	0.5833	0.9779	0.9465	0.3445	0.9012
DT	MW1	5	0.9139	0.6667	0.9867	0.9489	0.3483	0.9011
J48	MW1	5	0.916	0.4667	0.9646	0.9397	0.2927	0.8893
ABR	MW1	5	0.917	0.5833	0.9779	0.9465	0.3445	0.9051

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

TABLE III. Performance Evaluation with PCA

CLASSIFIER	DATASET	CV	NO OF FEATURE	SENSITIVITY	SPECIFICITY	PRECISION	F1SCORE	MCC	ACCURACY
PCA+NB	KC1	5	30	0.8997	0.5099	0.8838	0.9485	0.3479	0.8512
PCA+MLP	KC1	5	30	0.8894	0.7511	0.9553	0.9727	0.3757	0.8774
PCA+AB	KC1	5	30	0.8755	0.7554	1	0.971	0.3242	0.8598
PCA+RF	KC1	5	30	0.9039	0.7458	0.939	0.9745	0.4222	0.8255
PCA+DT	KC1	5	30	0.8915	0.7287	0.9482	0.9708	0.3751	0.8757
PCA+J48	KC1	5	30	0.8981	0.5455	0.9079	0.958	0.3515	0.8522
PCA+ABR	KC1	5	30	0.904	0.7354	0.9355	0.9731	0.4183	0.8833
PCA+NB	MC2	5	30	0.7823	0.7185	0.9289	0.849	0.4074	0.7428
PCA+MLP	MC2	5	30	0.84	0.715	0.8795	0.8593	0.9301	0.7902
PCA+AB	MC2	5	30	0.7795	0.5952	0.9155	0.8423	0.3889	0.7412
PCA+RF	MC2	5	30	0.7951	0.7297	0.9315	0.8577	0.4323	0.7802
PCA+DT	MC2	5	30	0.7555	0.7357	0.9535	0.8487	0.3817	0.7503
PCA+J48	MC2	5	30	0.7985	0.5453	0.8548	0.8257	0.3905	0.7405
PCA+ABR	MC2	5	30	0.7951	0.7807	0.9535	0.8558	0.4572	0.8503
PCA+NB	PC1	5	30	0.9593	0.3848	0.9579	0.9585	0.3254	0.9179
PCA+MLP	PC1	5	30	0.9553	0.48	0.9958	0.9803	0.3471	0.9378
PCA+AB	PC1	5	30	0.9444	0.28	1	0.9832	0.9832	0.9405
PCA+RF	PC1	5	30	0.9598	0.549	0.9523	0.9878	0.3519	0.9491
PCA+DT	PC1	5	30	0.9543	0.53	1	0.9842	0.2581	0.9435
PCA+J48	PC1	5	30	0.9552	0.459	0.9943	0.9795	0.3415	0.9354
PCA+ABR	PC1	5	30	0.9572	0.5411	1	0.9872	0.332	0.9505
PCA+NB	PC2	5	30	0.9912	0.0555	0.94	0.9549	0.0757	0.9241
PCA+MLP	PC2	5	30	0.9911	0.41	0.99	0.9984	0.2245	0.9872
PCA+AB	PC2	5	30	0.9898	1	1	0.9998	0.2575	0.9885
PCA+RF	PC2	5	30	0.9885	0	0.8521	0.9991	0	0.9875
PCA+DT	PC2	5	30	0.9885	0.529	1	0.9991	0.8451	0.9885
PCA+J48	PC2	5	30	0.9885	0.9452	0	0.9985	0.9154	0.9872
PCA+ABR	**PC2**	**5**	**30**	**0.9885**	**0**	**1.01**	**0.9991**	**0.01**	**0.9899**
PCA+NB	MW1	5	30	0.9558	0.34	0.8595	0.9151	0.3528	0.8421
PCA+MLP	MW1	5	30	0.931	0.3729	0.9713	0.9507	0.234	0.8935
PCA+AB	MW1	5	30	0.9257	0.964	0.9979	0.9604	0.2301	0.9093
PCA+RF	MW1	5	30	0.937	0.6033	0.9979	0.9665	0.3645	0.9212
PCA+DT	MW1	5	30	0.9339	0.6867	1	0.9689	0.3683	0.9211
PCA+J48	MW1	5	30	0.936	0.4867	0.9846	0.9597	0.3127	0.9093
PCA+ABR	MW1	5	30	0.937	0.6033	0.9979	0.9665	0.3645	0.9251

Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

TABLE IV. Performance Evaluation with PCA-LDA

CLASSIFIER	DATASET	CV	NO OF FEATURE	SENSITIVITY	SPECIFICITY	PRECISION	F1SCORE	MCC	ACCURACY
PCA-LDA+NB	KC1	5	18	0.9017	0.6119	0.8858	0.9505	0.3499	0.8532
PCA-LDA+MLP	KC1	5	18	0.8914	0.7531	0.9583	0.9747	0.3777	0.8794
PCA-LDA+AB	KC1	5	18	0.8785	0.7584	1	0.973	0.3262	0.8718
PCA-LDA+RF	KC1	5	18	0.9059	0.7488	0.941	0.9765	0.4242	0.8285
PCA-LDA+DT	KC1	5	18	0.8935	0.7307	0.9502	0.9728	0.3771	0.8777
PCA-LDA+J48	KC1	5	18	0.9001	0.6486	0.9099	0.96	0.3635	0.8642
PCA-LDA+ABR	KC1	5	18	0.906	0.7384	0.9376	0.9751	0.4203	0.8853
PCA-LDA+NB	MC2	5	18	0.7873	0.7236	0.9339	0.854	0.4124	0.7478
PCA-LDA+MLP	MC2	5	18	0.845	0.72	0.8845	0.8643	0.9351	0.7952
PCA-LDA+AB	MC2	5	18	0.7846	0.7002	0.9215	0.8473	0.3939	0.7462
PCA-LDA+RF	MC2	5	18	0.8001	0.7347	0.9366	0.8627	0.4373	0.7852
PCA-LDA+DT	MC2	5	18	0.7705	0.7407	0.9586	0.8537	0.3867	0.7553
PCA-LDA+J48	MC2	5	18	0.8036	0.6503	0.8598	0.8307	0.3955	0.7456
PCA-LDA+ABR	MC2	5	18	0.8001	0.7857	0.9586	0.8718	0.4722	0.8653
PCA-LDA+NB	PC1	5	18	0.9703	0.3858	0.9689	0.9696	0.3274	0.9189
PCA-LDA+MLP	PC1	5	18	0.9663	0.481	0.9968	0.9813	0.3481	0.9388
PCA-LDA+AB	PC1	5	18	0.9454	0.281	1	0.9842	0.9842	0.9416
PCA-LDA+RF	PC1	5	18	0.9608	0.605	0.9533	0.9888	0.3629	0.9501
PCA-LDA+DT	PC1	5	18	0.9553	0.531	1	0.9852	0.2691	0.9445
PCA-LDA+J48	PC1	5	18	0.9662	0.407	0.9953	0.9805	0.3426	0.9374
PCA-LDA+ABR	PC1	5	18	0.9582	0.6421	1.001	0.9882	0.333	0.9516
PCA-LDA+NB	PC2	5	18	0.9932	0.0676	0.942	0.9669	0.0777	0.9261
PCA-LDA+MLP	PC2	5	18	0.9931	0.412	0.992	1	0.2266	0.9892
PCA-LDA+AB	PC2	5	18	0.9918	1	0.9885	0.9998	0.2595	0.9905
PCA-LDA+RF	PC2	5	18	0.9905	0.002	0.8541	1	0	0.9895
PCA-LDA+DT	PC2	5	18	0.9905	0.631	1.002	0.9891	0.8471	0.9905
PCA-LDA+J48	PC2	5	18	0.9905	0.9472	0.002	0	0.9174	0.9892
PCA-LDA+ABR	**PC2**	**5**	**18**	**0.9905**	**0.002**	**1**	**0.9918**	**0.912**	**0.9919**
PCA-LDA+NB	MW1	5	18	0.9668	0.341	0.8706	0.9161	0.3638	0.8431
PCA-LDA+MLP	MW1	5	18	0.932	0.3739	0.9723	0.9517	0.235	0.8945
PCA-LDA+AB	MW1	5	18	0.9267	0.965	0.9989	0.9614	0.2311	0.9103
PCA-LDA+RF	MW1	5	18	0.938	0.6043	0.9989	0.9675	0.3655	0.9222
PCA-LDA+DT	MW1	5	18	0.9349	0.6877	1	0.9699	0.3693	0.9221
PCA-LDA+J48	MW1	5	18	0.937	0.4877	0.9856	0.9607	0.3137	0.9103
PCA-LDA+ABR	MW1	5	18	0.938	0.6043	0.9989	0.9675	0.3655	0.9261

Figures 2 and 3 show the KC1 datasets without and with selecting features. In both figures, ABR and PCA-LDA+ABR provided the highest accuracy0.7633 and 0.8853 correspondingly.

Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

Fig 2. KC1 dataset accurateness without feature collection

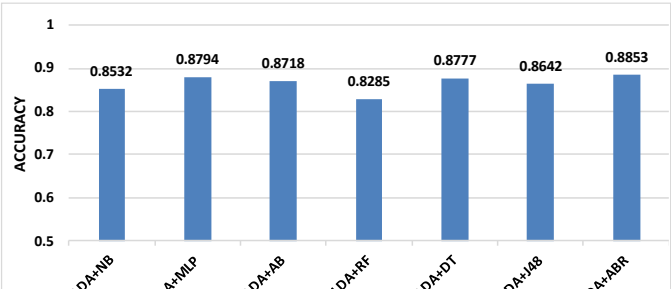

Fig 3. KC1 dataset correctness with feature collection

Figures 4 and 5 show the MC2 dataset both without and with selecting features. In both figures, ABR and PCA-LDA+ABR provided the greatest exactness 0.8203 and 0.8853 separately.

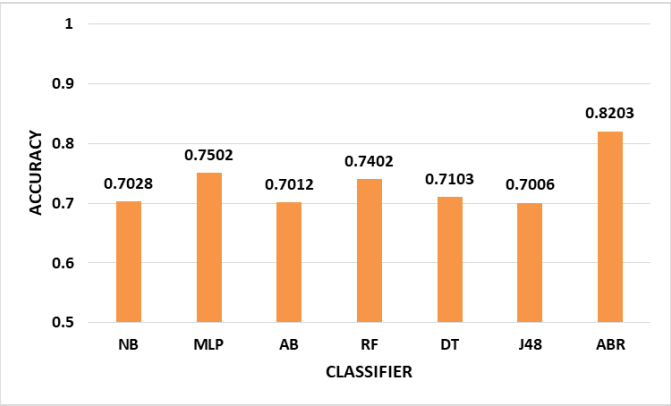

Fig 4. MC2 dataset accurateness without feature collection

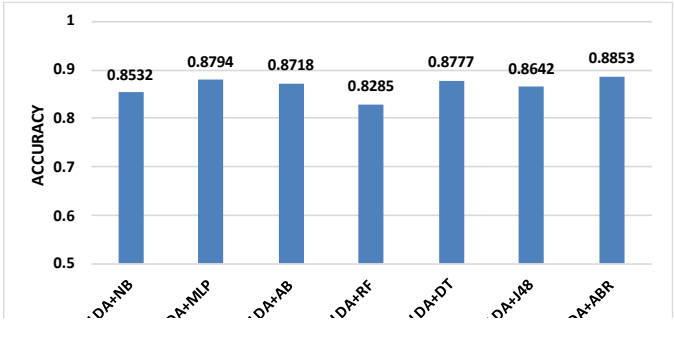

Fig 5. MC2 dataset correctness with feature collection

Figures 6 and 7 show the PC1 datasets both without and with selecting features. In both figures, ABR and PCA-LDA+ABR gave the most precision 0.9206 and 0.9516 correspondingly.

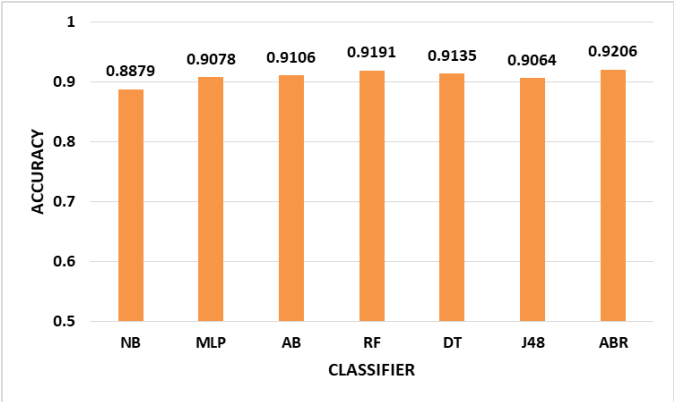

Fig 6. PC1 dataset correctness without feature collection

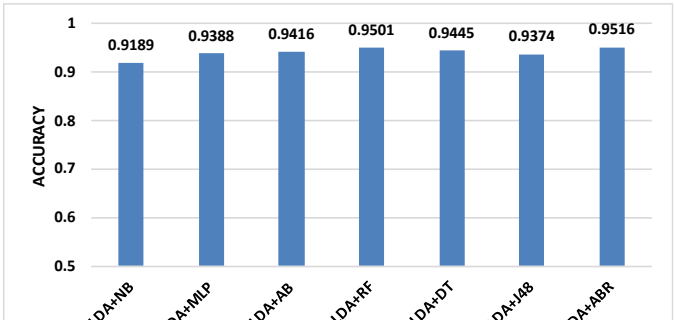

Fig 7. PC1 dataset correctness with feature collection

Figures 8 and 9 show the PC2 datasets without and with selecting features. In both the figures, ABR and PCA-LDA+ABR provided the most accurateness 0.9799 and 0.9919 correspondingly.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

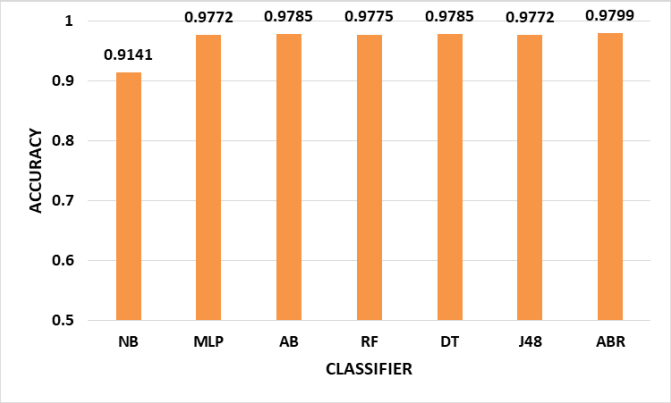

Fig 8. PC2 dataset correctness without feature collection

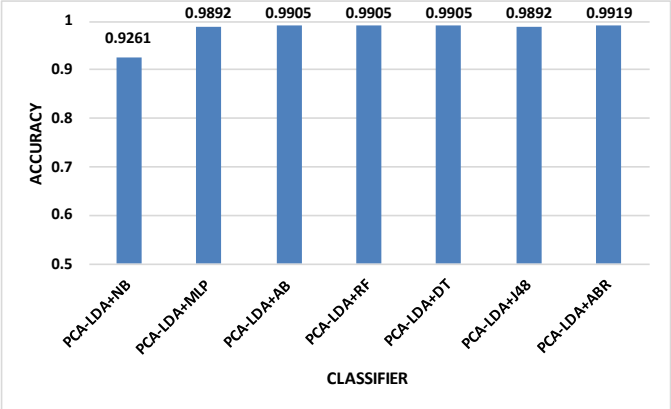

Fig 9. PC2 dataset correctness with feature collection

Figures 10 and 11 show the MW1 datasets both without and with selecting features. In both figures, ABR and PCA-LDA+ABR provided the most accurateness 0.9051 and 0.9261 correspondingly.

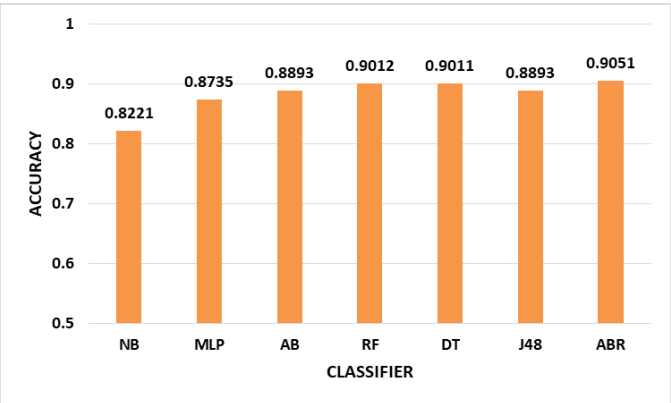

Fig 10. MW1 dataset correctness without feature collection

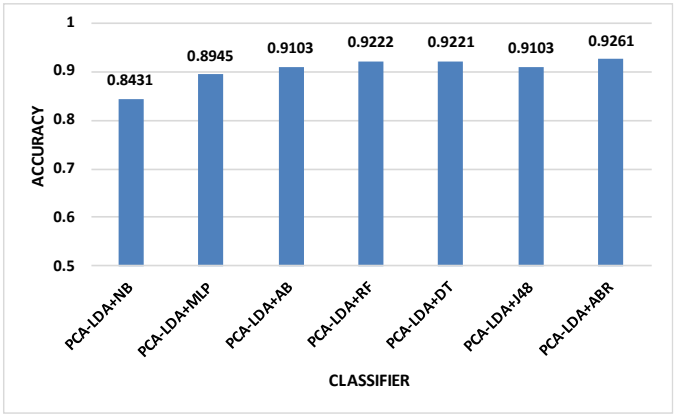

Fig 11. MW1 dataset correctness with feature collection

V. CONCLUSION & FUTURE WORK

We offer a novel technique for predicting software defects using PCA-LDA+ABR that combines dimensionality reduction with a hybrid classifier. Our solution employs PCA to minimize the amount of structures, followed by ABR, a grouping of AB and RF techniques, to categorize the software components as faulty or non-defective. We tested our strategy on six different datasets and found that it outperformed existing approaches. On the PC2 data set, our technique obtains an impressive accuracy of 0.9919. Our study contributes to current research into building more precise and effective software defect prediction systems, which can aid in early detection of software problems and reduce the cost of software development initiatives. We propose that future study investigate additional combination models to assist with software fault prediction.

REFERENCES

[1] Hammouri, Awni, et al. "Software bug prediction using machine learning approach." International journal of advanced computer science and applications 9.2 (2018).
[2] Immaculate, S. Delphine, M. Farida Begam, and M. Floramary. "Software bug prediction using supervised machine learning algorithms." 2019 International conference on data science and communication (IconDSC). IEEE, 2019.
[3] Khan, F., Kanwal, S., Alamri, S. and Mumtaz, B., 2020. Hyper-parameter optimization of classifiers, using an artificial immune network and its application to software bug prediction. Ieee Access, 8, pp.20954-20964.
[4] Rhmann, W., Pandey, B., Ansari, G., & Pandey, D. K. (2020). Software fault prediction based on change metrics using hybrid algorithms: An empirical study. Journal of King Saud University-Computer and Information Sciences, 32(4), 419-424.
[5] Chaturvedi, K. K., & Singh, V. B. (2012, September). Determining bug severity using machine learning techniques. In 2012 CSI sixth international conference on software engineering (CONSEG) (pp. 1-6). IEEE.
[6] Aleem, S., Capretz, L. F., & Ahmed, F. (2015). Benchmarking machine learning technologies for software defect detection. arXiv preprint arXiv:1506.07563.
[7] Ferenc, Rudolf, Péter Gyimesi, Gábor Gyimesi, Zoltán Tóth, and Tibor Gyimóthy. "An automatically created novel bug dataset and its validation in bug prediction." Journal of Systems and Software 169 (2020): 110691.
[8] Jayanthi, R., and Lilly Florence. "Software defect prediction techniques using metrics based on neural network classifier." Cluster Computing 22 (2019): 77-88.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

[9] Shivaji, S., Whitehead, E. J., Akella, R., & Kim, S. (2009, November). Reducing features to improve bug prediction. In 2009 IEEE/ACM International Conference on Automated Software Engineering (pp. 600-604). IEEE.

[10] Gupta, Aashish, et al. "Novel xgboost tuned machine learning model for software bug prediction." 2020 international conference on intelligent engineering and management (ICIEM). IEEE, 2020.

[11] Catal, Cagatay. "Software fault prediction: A literature review and current trends." Expert systems with applications 38.4 (2011): 4626-4636.

[12] Catal, Cagatay, Ugur Sevim, and Banu Diri. "Practical development of an Eclipse-based software fault prediction tool using Naive Bayes algorithm." Expert Systems with Applications 38.3 (2011): 2347-2353.

[13] Das M, Pradhan D, Mohapatra S, "A PCA BASED SOFTWARE FAULT PREDICTION MODEL USING ADRF", International Journal of Emerging Technologies and Innovative Research (www.jetir.org | UGC and issn Approved), ISSN:2349-5152, Vol.10, Issue 5, page no. ppj189-j199, June-2023.

[14] Rathore, Santosh S., and Sandeep Kumar. "An empirical study of ensemble techniques for software fault prediction." Applied Intelligence 51 (2021): 3515-3544.

[15] Erturk, Ezgi, and Ebru Akcapinar Sezer. "A comparison of some soft computing methods for software fault prediction." Expert systems with applications 42.4 (2015): 1872-1879.

[16] D'Ambros, Marco, Michele Lanza, and Romain Robbes. "An extensive comparison of bug prediction approaches." Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on. IEEE, 2010.

[17] Y. Tohman, K. Tokunaga, S. Nagase, and M. Y., "Structural approach to the estimation of the number of residual software faults based on the hyper-geometric districution model," IEEE Trans. on Software Engineering, pp. 345–355, 1989.

[18] T. Angel Thankachan1, K. Raimond2, "A Survey on Classification and Rule Extraction Techniques for Data mining",IOSR Journal of Computer Engineering ,vol. 8, no. 5,(2013), pp. 75-78.

[19] Okutan, Ahmet, and Olcay Taner Yıldız. "Software defect prediction using Bayesian networks." Empirical Software Engineering 19.1 (2014): 154-181.

[20] Ilsen, David L. and Delen, " Advanced Data Mining Techniques ", Springer, 1st edition, page 138, ISBN 3-540-76016-1, Feb 2008.

[21] D. Pradhan and D. Muduli, "Software Defect Prediction Model Using AdaBoost based Random Forest Technique," 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), Delhi, India, 2023, pp. 1-6, doi: 10.1109/ICCCNT56998.2023.10308208.

[22] Anwesh Kumar Mahanta, Smruti Rekha Pradhan, Biswajeet Sahoo, Debasish Pradhan, 2024, An Automated Pca-lda Based Software Fault Prediction Model Using Machine Learning Classifier, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 13, Issue 1 (January 2024).