International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-11 Issue-5, January 2023

Prediction of Software Defects using Ensemble Machine Learning Techniques

Sowjanya Jindam, Sai Teja Challa, Sai Jahnavi Chada, Navya Sree B, Srinidhi Malgireddy

Abstract: ***During software development and maintenance, predicting software bugs becomes critical. Defect prediction early in the software development life cycle is a crucial aspect of the quality assurance process that has garnered significant attention over the past two decades. Early detection of defective modules in software development can support the development team in efficiently and effectively utilising available resources to deliver high-quality software products in a short timeframe. The machine learning approach, which detects hidden patterns among software features, is an effective method for identifying problematic modules. The software flaws in NASA datasets MC1, MW1, KC3, and PC4 are predicted using multiple machine learning classification algorithms in this work. A new model was developed by adjusting the parameters of the previous XGBoost model, including N_estimators, learning rate, maximum depth, and subsample. The results were compared to those obtained by state-of-the-art models, and our model outperformed them across all datasets.***

Keywords: Machine Learning, Dataset, Supervised Learning, Random Forest, XgBoost, Ada Boost, Decision Tree.

I. INTRODUCTION

Completing a software project nowadays is a significant challenge. A project manager's primary concern is defects or problems—poor code design and implementation cause these issues to appear. Creating bug-free software is the most demanding job in the software industry. Even if they are focused on testing, software development companies will struggle to tackle this issue.

Manuscript received on 23 December 2022 | Revised Manuscript received on 30 December 2022 | Manuscript Accepted on 15 January 2023 | Manuscript published on 30 January 2023.

*Correspondence Author(s)
Sowjanya Jindam*, Assistant Professor, Department of Information Technology, Maturi Venkata Subba Rao (MVSR) Engineering College, Osmania University, Hyderabad (Telangana), India. Email: sowjanya_it@mvsrec.edu.in ORCID ID: https://orcid.org/0000-0001-6959-2110
Sai Teja Challa, Student, Department of Information Technology, Maturi Venkata Subba Rao (MVSR) Engineering College, Osmania University, Hyderabad (Telangana), India. Email: saitejachalla2001@gmail.com
Sai Jahnavi Chada, Student, Department of Information Technology, Maturi Venkata Subba Rao (MVSR) Engineering College, Osmania University, Hyderabad (Telangana), India. Email: jahnavisekharreddy02@gmail.com
Navya Sree B, Student, Department of Information Technology, Maturi Venkata Subba Rao (MVSR) Engineering College, Osmania University, Hyderabad (Telangana), India. Email: navyasree140301@gmail.com
Srinidhi Malgireddy, Student, Department of Information Technology, Maturi Venkata Subba Rao (MVSR) Engineering College, Osmania University, Hyderabad (Telangana), India. Email: malgireddysrinidhi51@gmail.com

Defects are a regular occurrence in any human-created application because it is not an automated process. Software development companies, on the other hand, place high importance on early fault detection through a range of inspection and testing methodologies. Predicting bugs has a significant impact in today's world, which is becoming increasingly dependent on software. We aim to compare the effectiveness of the XGBoost algorithm against classic machine learning methods, including logistic regression, decision trees, random forest, and AdaBoost, in detecting defects in software. In this project, Machine Learning algorithms such as logistic regression and decision trees have been used in past studies. They have lower accuracy and make numerous assumptions about datasets, which may lead to biases if the assumptions are not met. Later, for classification, ensemble techniques such as Random Forest were created, which outperform decision trees. However, we employed boosting techniques, such as AdaBoost for Classification, to mitigate bias introduced by the generation process. However, it, too, has poor accuracy.

II. RELATED WORK

In this methodology, machine learning models, including Logistic Regression, Decision Tree, Random Forest, AdaBoost, and XGBoost, were utilised as state-of-the-art models for four datasets from NASA-KC2, PC3, JM1, and CM1. Later on, a new model was proposed based on tuning the existing XGBoost model by changing its parameters, namely N_estimator, learning rate, max depth, and subsample[1]. Proposed a novel sparsity-aware algorithm for sparse data and a weighted quantile sketch for approximate tree learning. More importantly, we provide insights on cache access patterns, data compression, and sharding to build a scalable tree boosting system. By combining these insights, XGBoost scales beyond billions of examples using far fewer resources than existing systems[2]. Using a random selection of features to split each node yields error rates that compare favorably to Adaboost (Y. Freund & R. Schapire, Machine Learning: Proceedings of the Thirteenth International Conference, ***, 148–156), but are more robust concerning noise. Internal estimates monitor error, strength, and correlation, which are used to show the response to increasing the number of features used in the splitting. Internal estimates are also used to measure the importance of variables. These ideas are also applicable to regression[3].
An approach to the construction of classifiers from imbalanced datasets is described[4].They used three supervised machine learning algorithms to build the model and predict the occurrence of software bugs based on historical data by deploying the classifiers Logistic regression, Naïve Bayes, and Decision Tree.

Proposed bug detection as a binary classification problem, e.g., correct and wrong, and used the deep Bugs framework to train a classifier that can identify faulty code from correct code [5]. Proposed a defect detector system that works with a variety of compilers and languages, such as javac, gcc, and Visual Studio [6]. Using marginal R square values, proposed a strategy that uses the smallest and most accurate number of performing metrices at a time. The Eclipse JDT Core dataset [7] was chosen. Using One Class SVM, a one-class SFP (Software Fault Prediction) Model [8]. Machine learning was used to forecast vulnerability in web applications. This article generates input validation and sanitation properties. For each sink, it computes a static backwards slice. The control flow graph, control dependence graph, and system dependence graph of a web software [9] are used to analyse the programme. The suggested approach is to first prioritise the problems based on severity and component attributes. It employs the Xmean Clustering algorithm in conjunction with the Bayes Net Classifier [10].

III. METHODOLOGY

In our research and extensive literature review, we found that Random Forest works well for Software Defect Prediction with high accuracy, but that it can be further improved by another recently introduced algorithm called XGBoost. In this paper, we propose a method for building a predictive model with higher accuracy using Ensemble Machine Learning Techniques.

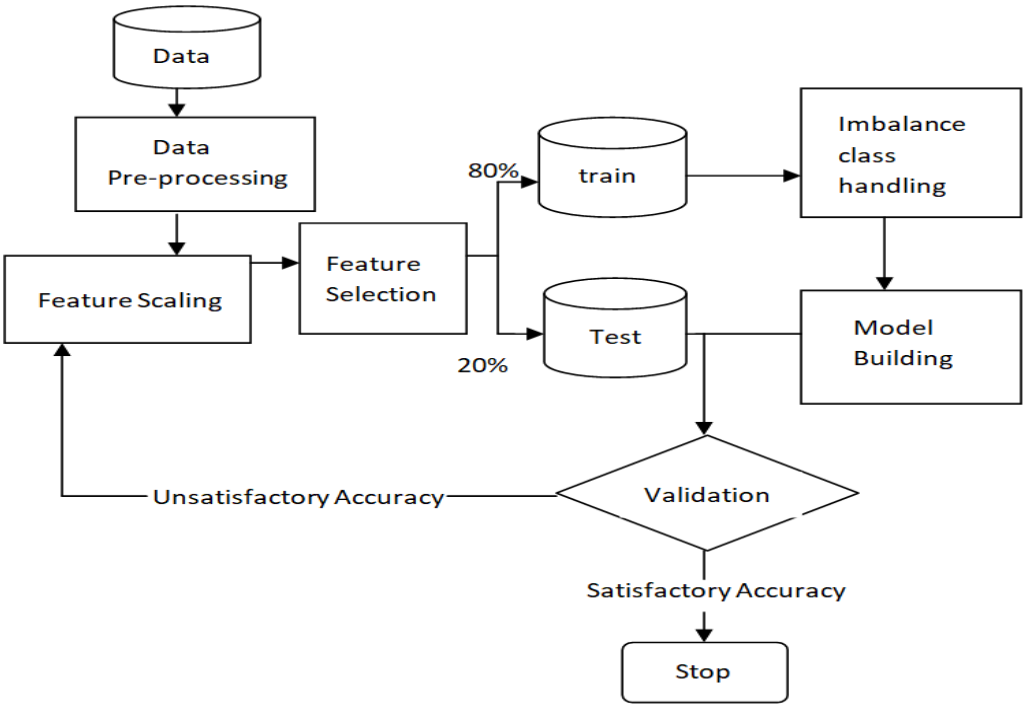

Fig-1: Architectural Diagram

The above figure illustrates that data preprocessing is performed in several phases. Firstly, the data is preprocessed, and then it is classified into training and testing datasets. After classification, the handling of imbalanced classes and model building is completed. The data is then validated to a satisfactory accuracy. If the satisfied accuracy is not met, it is fed back to the feature scaling again.

- Data Preprocessing

The most fascinating thing of our time is machine learning. For their organizations, everyone is starting to use machine learning models. Data is at the centre of the complex process. Our machine learning tools ensure that the data we process is of high quality. In that case, Data Pre-processing is an essential stage in creating a successful prediction model. The method of feature scaling is used to determine the range of independent variables or data attributes. In our study, we employed a normalisation approach to standardise our data to a standard scale.

Class Imbalance Handling: As we can observe from the distribution graph of our target variable. If our target variable is highly skewed, our prediction model will tend to ignore the minority class while being biased toward the majority. In this study, we employed SMOTE, a synthetic minority oversampling technique, to address this issue.

- Software Defect Dataset

We selected a standard NASA Promise Repository software bug dataset to evaluate the effectiveness of our technique. The datasets MC1, MW1, KC3 and PC4 are used. The public URL for this repository is https://www.kaggle.com/datasets/aczy156/nasa-software-defect-prediction. The following is a brief description of these datasets: NASA created the dataset as part of its Metrics Data Program. Shepperd et al. cleaned up the dataset in 2013 by removing duplicate and inconsistent data. The PROMISE database contains this enhanced information. As a result, for the cleaned data in our investigation, we used the NASA Dataset. NASA uses Halstead and McCabe metrics for each occurrence in their data.

There are around 40 features, such as the unique operator value (MU1), unique operator value (MU2), total number of operators (N1), total number of operands (N2), rows Code (LOC), and so on. Each NASA project has its own set of features. Table 1 lists the specific number of attributes available.

TABLE-1. Dataset Insights

Dataset	Project Name	#Instances	Features
NASA	MC1	1988	40
	MW1	253	39
	KC3	194	41
	PC4	1287	39

- Data Splitting

We can't use the dataset to train our model. If we train our model on all data points, we will encounter an issue with overfitting, and our model may make incorrect predictions for new statements. To test the effectiveness and reliability of our model, we decided to split our dataset into two sections with an 80:20 ratio. The model is trained using 80% of the dataset, while the remaining 20% is used to evaluate the model's performance through comparisons of projected and actual values.

IV. RESULT ANALYSIS AND DISCUSSION

Each dataset is trained on five different machine learning algorithms, and their respective accuracies are recorded. These accuracies are then used for comparison between the models.

A. Logistic Regression

Logistic regression is a statistical model that uses a logistic function to represent a binary dependent variable in its most basic form, though there are many more advanced variants. Logistic regression (or logit regression) is a technique for estimating the parameters of a logistic model in regression analysis (a form of binary regression).

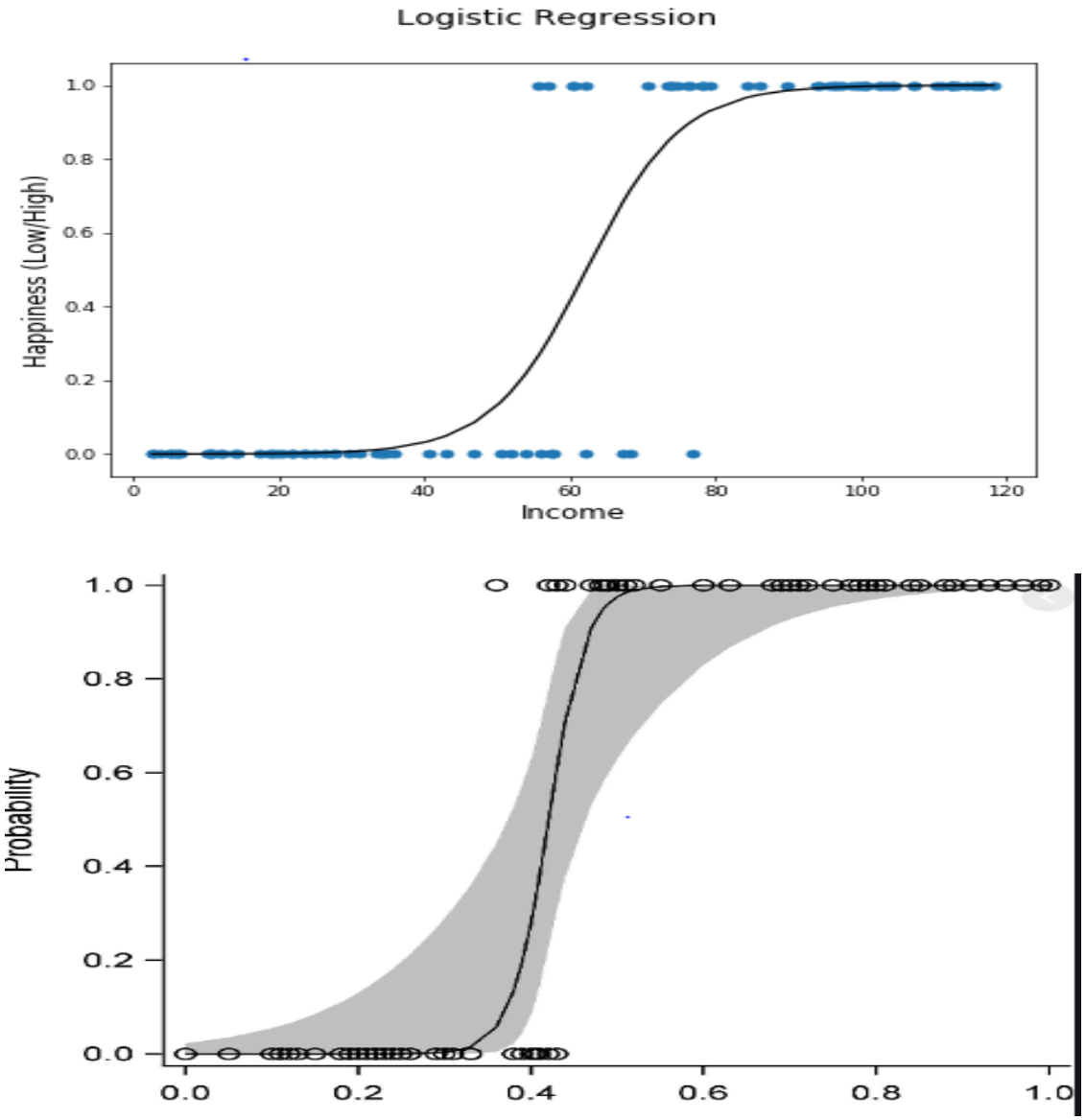

Fig-2: Logistic regression

This linear relationship can be written in the following mathematical form (where ℓ is the log-odds, is the base of the logarithm, and are parameters of the model):

$$\ell = \log_b \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \quad (1)$$

$$P = \frac{e^{a+bX}}{1+e^{a+bX}} \quad (2)$$

The variables x_1 and x_2 are explanatory variables, and β0, β1, and β_2 are coefficients. The probability odds ratio EB has a more straightforward interpretation in the case of a categorical explanatory variable with two categories; in this case, it is simply the odds ratio for one category compared with the other.

B. Decision Tree

For classification and regression, decision trees are a nonparametric supervised learning method. The goal is to learn simple decision rules using data attributes to build a model that predicts the value of a target variable.

The root of a decision tree is at the top of an upside-down tree. The bold writing in black in the figure on the left represents an internal node/condition that causes the tree to break into branches/edges. The decision/leaf, in this case, whether the passenger died or survived, is shown by red and green text, respectively, at the end of the branch that doesn't divide anymore.

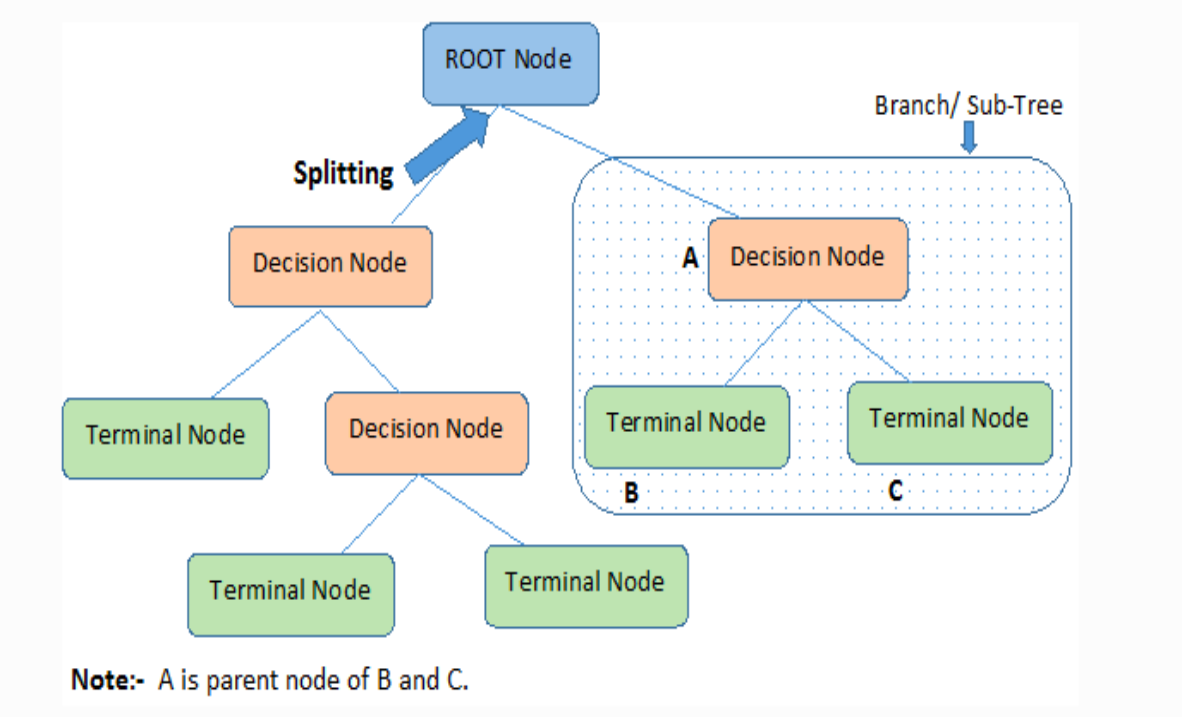

Fig-3: Block Diagram for Decision Tree Algorithm

C. Random Forest

As the name implies, a random forest is made up of a vast number of individual decision trees that work together as an ensemble. Each tree in the random forest produces a class prediction, and the class with the most votes becomes the prediction of our model.

Bootstrap aggregation is known as bagging. It combines the estimates of many learners to reduce variance.

The Random Forest algorithm is a bagging-based supervised classification system. The number of trees in the forest has a direct relationship with the accuracy of the results: the more trees there are, the more accurate the results.

The following are some of the advantages of Random Forest over Decision Trees:

- Overfitting is a serious issue that can damage results, but with the Random Forest technique, the classifier will not overfit the model if there are enough trees in the forest.
- The same random forest approach may be utilized for classification and regression tasks.
- Feature engineers may use the Random Forest technique to discover the most relevant characteristics from a training dataset.

D. AdaBoost

Ada-boost, or Adaptive Boosting, is an ensemble classifier that uses the boosting idea. It combines many classifiers to improve classifier accuracy.

AdaBoost is a technique for creating iterative ensembles. The AdaBoost classifier creates a robust classifier by merging several low-performing classifiers, resulting in a high-accuracy classifier.

Adaboost's core principle is to establish the weights of classifiers and train the data sample in each iteration, so that reliable predictions can be made for uncommon observations. Any machine learning method that accepts weights on the training set can be used as a basic classifier.

Adaboost must meet two requirements:

1. The classifier should be interactively trained using a variety of weighted training examples.
2. It seeks to offer an excellent fit for these instances in each of iteration by minimizing training error.

It operates in the following manner:

1. Adaboost picks a training subset at random at first.
2. It trains the AdaBoost machine learning model iteratively by selecting

the training set based on the accuracy of the previous training.

3. It gives incorrectly categorized observations a larger weight so that they have a higher chance of being classified in the next iteration. It also allocates weight to the trained classifier in each of iteration based on the classifier's accuracy. The more precise classifier will be given more weight.
4. This approach is repeated until all of the training data fits perfectly or until the maximum number of estimators is reached.
5. Perform a "vote" across all of the learning algorithms you created to categorize them.
6. It also allocates weight to the trained classifier in each of iteration based on the classifier's accuracy. The more precise classifier will be given more weight.
7. This approach is repeated until all of the training data fits perfectly or until the maximum number of estimators is reached.
8. Perform a "vote" across all of the learning algorithms you created to categorize them.

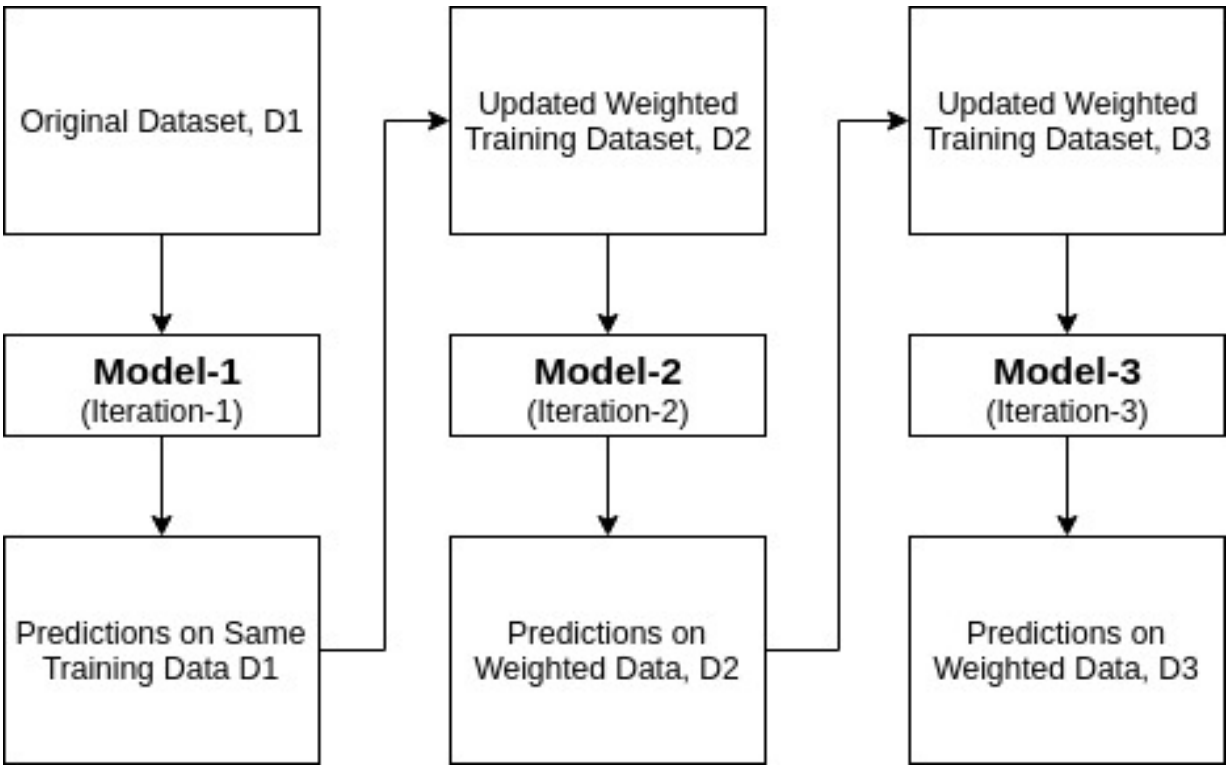

Fig. 4: Block Diagram of AdaBoost Algorithm

E. XG Boost

Gradient boosting also includes an ensemble method, similar to traditional boosting, that sequentially adds predictors and corrects previous models. Rather than providing changing weights to the classifiers after each iteration, this method fits the new model to fresh residuals from the prior prediction. Then it minimizes the loss when adding the most recent prediction.

So, in the end, you're utilizing gradient descent to update your model, which is why it's called gradient boosting. This supports both regression and classification issues. This approach is implemented for decision tree boosting in XGBoost with an additional custom regularisation term in the goal function.

Let's us understand the reason behind the good performance of XGboost -

Regularization: This is considered as the algorithm's most important aspect. The term "regularization" refers to a strategy for removing overfitting from a model.

Cross-Validation: We use cross-validation by importing the scikit-learn function; however, XGBoost has an inherent CV function.

Missing Value: It is built in such a way that it can cope with missing values. It identifies and apprehends trends in the missing values.

Flexibility: It enables the support of objective functions. They are the functions that are used to evaluate the model's performance and can also handle user-defined validation metrics.

Save and load: It allows you to save the data matrix and then reload it, conserving both resources and time.

The results achieved by state-of-the-art models are compared with our model in Figures 5 to 8.

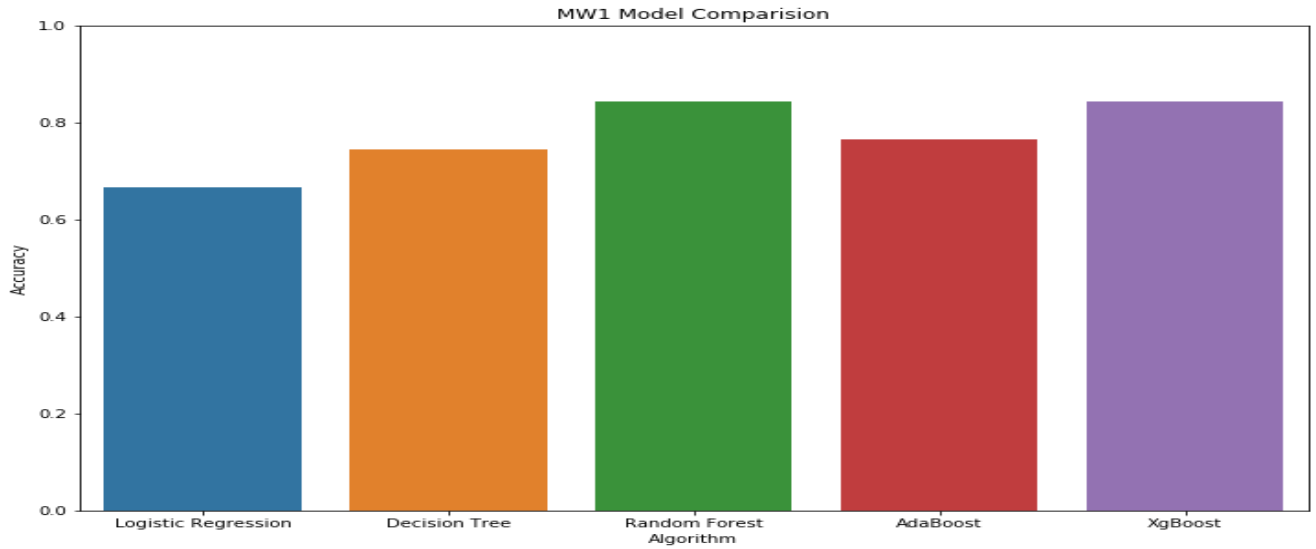

Fig-5: MW1 Model Comparison

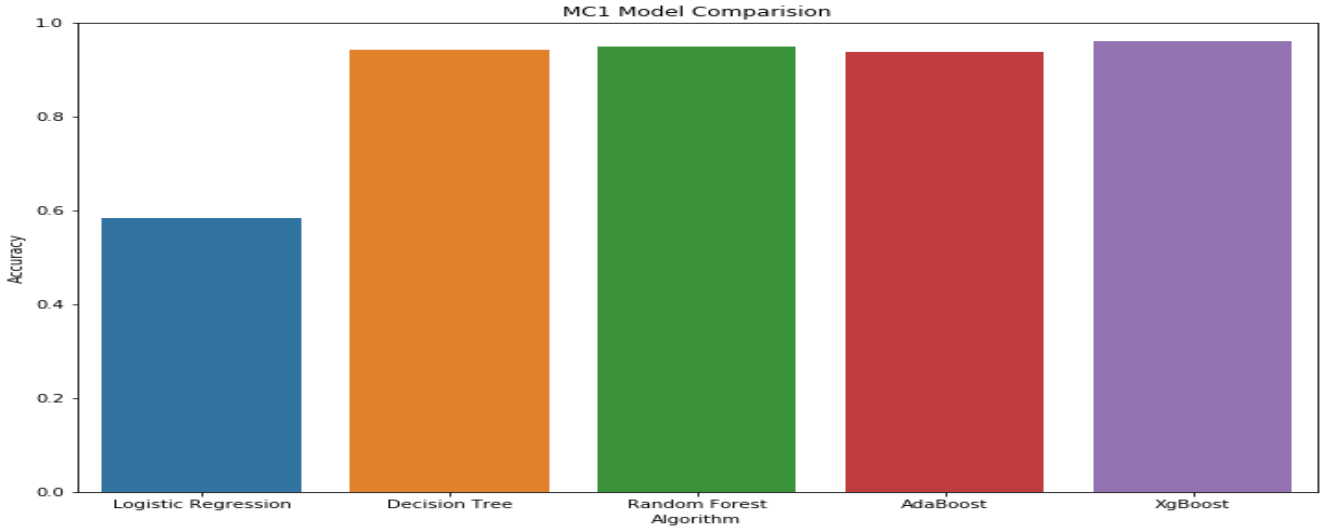

Fig-6: MC1 Model Comparison

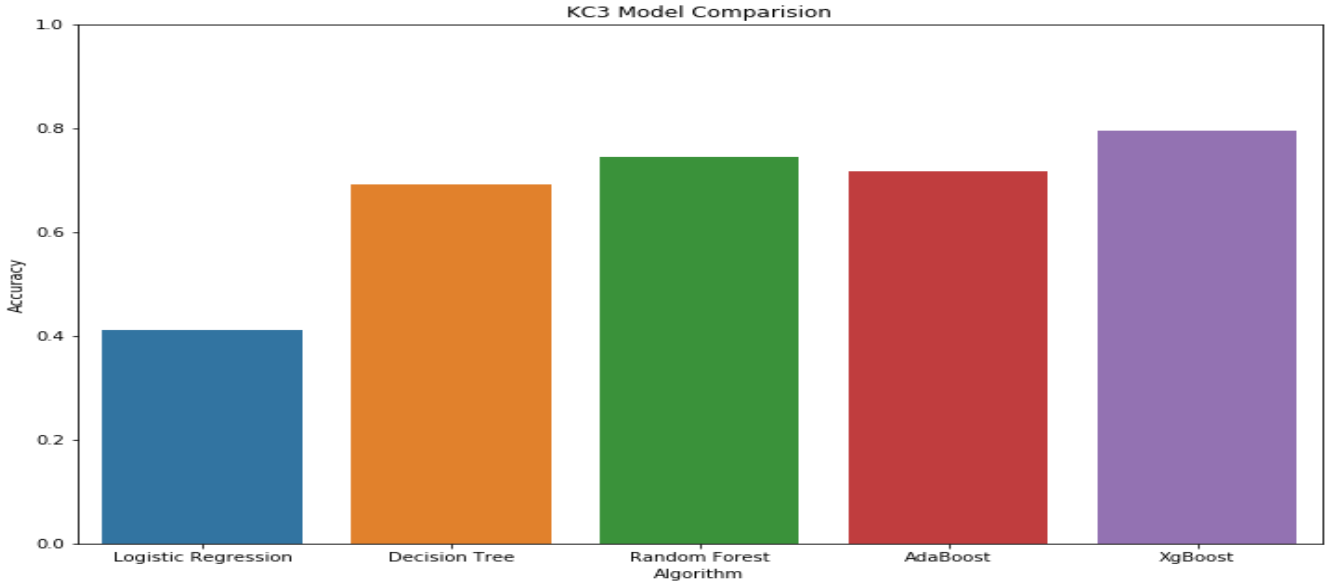

Fig-7: KC3 Model Comparison

Retrieval Number: 100.1/ijrte.E74210111523
DOI: 10.35940/ijrte.E7421.0111523
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

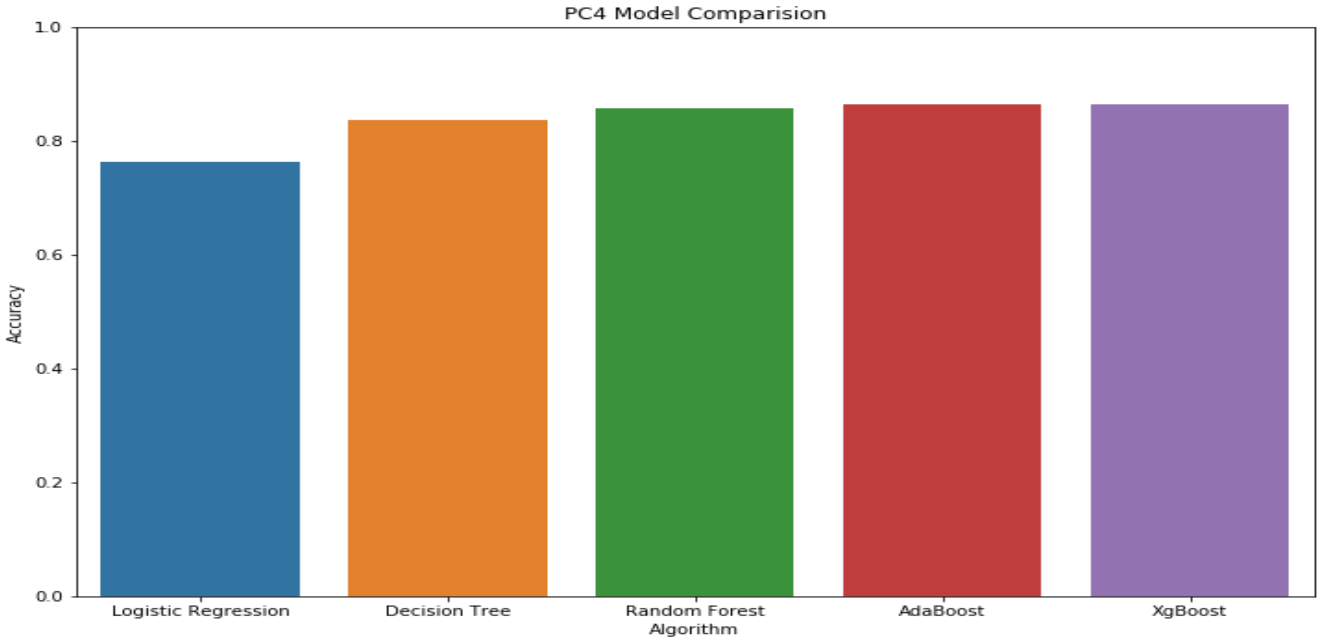

Fig-8: PC4 Model Comparison

The accuracies for all models and all datasets are shown below:

Table 2. Accuracies

	Logistic Regression	Decision Tree	Random Forest	Ada Boost	Xg Boost
MC1	0.41025	0.6923	0.717949	0.6923	0.794872
MW1	0.58291	0.9422	0.949749	0.9396	0.959799
KC3	0.66666	0.745	0.862745	0.7647	0.843137
PC4	0.76356	0.8333	0.856589	0.8643	0.864341

V. CONCLUSION

In this research, we employ feature scaling to preprocess the data for improved extraction and selection. Using the SMOTE technique, we successfully addressed the problem of class imbalance in the datasets. We then applied state-of-the-art machine learning models – including Logistic Regression, Decision Tree, Random Forest, AdaBoost, and XGBoost – to four NASA datasets: KC3, PC4, MC1, and MW1. Later, a new model was presented that was based on tweaking the old XGBoost model's parameters, such as the number of estimators (N_estimator), learning rate, maximum depth, and subsample. When the findings were compared against state-of-the-art models, we found that our model outperformed them across all datasets.

DECLARATION

Funding	Not funding.
Conflicts of Interest/ Competing Interests	No conflicts of interest are observed to the best of my knowledge.
Ethics Approval and Consent to Participate	The article does not require ethical approval or consent to participate.
Availability of Data and Materials	The dataset used in this project is available at https://www.kaggle.com/datasets/aczy156/nasa-software-defect-prediction
Authors Contributions	All authors have equal participation in this article.

REFERENCES

1. A. Gupta, S. Sharma, S. Goyal and M. Rashid, "Novel XGBoost Tuned Machine Learning Model for Software Bug Prediction," 2020 International Conference on Intelligent Engineering and Management (ICIEM). [CrossRef]
2. Chen, Tianqi & Guestrin, Carlos. (2016). XGBoost: A Scalable Tree Boosting System. 785-794. 10.1145/2939672.2939785. [CrossRef]
3. Leo Breiman. 2001. Random Forests. Mach. Learn. 45, 1 (October 1 2001), 5–32. DOI:https://doi.org/10.1023/A:1010933404324 [CrossRef]
4. Chawla, Nitesh & Bowyer, Kevin & Hall, Lawrence & Kegelmeyer, W. (2002). SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Intell. Res. (JAIR). 16. 321-357. 10.1613/jair. 953. [CrossRef]
5. Abdullah Alsaeedi, Mohammad, Zubair Khan "Software Defect Prediction Using Supervised Machine Learning and Ensemble Techniques: A Comparative Study" JSEA, 2019. [CrossRef]
6. Amod Kumar, Ashwni Bansal "Software Fault Proneness Prediction Using Genetic-Based Machine Learning Techniques" IEEE,2019. [CrossRef]
7. Meiliana, Syaeful Karim, Harco Leslie Hendric Spits Warnars, Ford Lumban Gaol, Edi Abdurachman, Benfano Soewito "Software Metrics for Fault Prediction Using Machine Learning Approaches" IEEE-2017.
8. Keita Mori and Osamu Mizuno "An Implementation of Just-In-Time Fault-Prone Prediction Technique Using Text Classifier" IEEE, 2015. [CrossRef]
9. Ali Ouni, Marwa Daagi, Marouane Kessentini, Salah Bouktif, Mohamed Mohsen Gammoudi. "A Machine Learning-Based Approach to Detect Web Service Design Defects", IEEE, 2017. [CrossRef]
10. Uma Subbiah, Muthu Ramachandran and Zaigham Mahmood "Software Engineering Approach to Bug Prediction Models using Machine Learning as a Service (MLaaS)" IEEE-2019. [CrossRef]

AUTHORS PROFILE

Sowjanya Jindam is an Assistant Professor from MVSR Engineering College with qualifications in MTech. Her research areas include Big Data Analytics, Machine Learning, and the Internet of Things. Her recent works include an Integrated Dynamic Approach for Predicting Rainfall, Classifying Authentic and Fraudulent Reviews. Using Supervised Machine Learning, a Social Distancing Detector using Deep Learning, and Recognition of

Traffic Signs Using CNN and Deep Learning.

Sai Teja Challa is a Bachelor of Engineering student, pursuing his final year in Information Technology at MVSR Engineering College, Osmania University. His research interests include the field of Data Science, Machine Learning, and Artificial Intelligence. Recently, he worked on the projects "Age & Gender Prediction Using Face Recognition," "Credit Card Fraud Detection," and "Pedestrian Detection Model Using Haar Cascade."

Sai Jahnavi Chada, a Bachelor of Engineering student, is pursuing her final year in Information Technology at MVSR Engineering College, Osmania University. Her research interests encompass the fields of Machine Learning and Data Science. Recently, she worked on the project "Heart Disease Prediction using Machine Learning."

Navya Sree B is a Bachelor of Engineering student, pursuing her final year in Information Technology at MVSR Engineering College, Osmania University. Her research interests encompass the field of Web development. Recently, she worked on the Farmers Portal project.

Srinidhi Malgireddy is a Bachelor of Engineering student, pursuing her final year in Information Technology at MVSR Engineering College, Osmania University. Her research interests encompass the field of Cloud Computing. Recently, she worked on the project titled "Cross Premises Connectivity Using Site-to-Site VPN."

Retrieval Number: 100.1/ijrte.E74210111523
DOI: 10.35940/ijrte.E7421.0111523
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)