**Proposed Performance Evaluation Of Some Selected**

**Machine Learning Algorithms For Software Defect**

**Prediction System Using Stochastic Gradient Decent Algorithm**

**ASIFAT, Shuaib Akintunde**

*Department of Computer Science,*

*Federal College of Education (Special), Oyo*

*asifat.shuaib1503@fcesoyo.edu.ng*

***GIWA, Mary Funmilola***

*Department of Computer Science,*

*Federal College of Education (Special), Oyo*

*maryfunmilolagiwa@gmail.com*

***OLUOKUN, John Oluwole***

*Department of Computer Science,*

*Federal College of Education, Iwo*

*Osun State*

*oluokunjo@fceiwo.edu.ng*

**Abstract**

*Software defects are critical issues that can compromise the quality, increase costs, and delay the development of software systems. This research evaluates the performance of various machine learning algorithms—Naive Bayes (NB), Support Vector Machine (SVM), Artificial Neural Network (ANN), K-Nearest Neighbor (KNN), and Decision Tree (C4.5)—for software defect prediction. The study employs the Stochastic Gradient Descent Algorithm (SGDA) to extract relevant features from publicly available datasets, ensuring the elimination of overfitting. These extracted features are then classified using the selected machine learning algorithms, with their performance evaluated based on accuracy, precision, sensitivity, and F1-score. The findings aim to identify the most effective classifier, thereby guiding the development of more reliable software defect prediction systems.*

**Keywords:** Software, Prediction, Machine learning, Classification

**Introduction**

This is technology era that the backbone of every individual endeavour is a reliable software system (Rao & Patra, 2020). The reason for this is that, software as become part and parcel of human life. Nowadays, there is no field of study whether private small scale business or large scale business and government parastatals that do not depend on reliable software. In this case, it is therefore very necessary to make real time software that will be reliable, trustworthy and economical available at any point in time. Moreover, society as increasingly relied on advanced software systems, as software is intertwined with all aspects of human live (Singh & Salaria, 2013). The software that will serve this purpose needed to be well monitored before started using it for the task it is meant for. This leads to software defect prediction, with this, the software developer will be able to find the likely faults on the software before the implementation.

The performance of machine learning algorithms depends majorly on the features used for the classification phase (Varghese & Verghese, 2012). If features are not effectively obtained, it can result in misclassification. Feature reduction is a significant phase in the machine learning algorithm, this technique consists of feature extraction and selection (Telgaonkar & Deshmukh, 2015). Feature selection is the process of selecting a subset of relevant features for model construction, thus reducing training times, simplifying the models (to make interpretation easier), and improving the chances of generalization, avoiding over-fitting (Novaković,*et..al,* 2011; Patra & Dash, 2016) The basic difference between feature selection and extraction is that feature selection retains a subset of the original features while feature extraction produces new ones (Kumar & Bhatia, 2014). With these aforementioned merits attributed to feature selection, this study will develop a software prediction system applying a new approach by the introduction of a stochastic gradient descent algorithm to select the most relevant features before the classification using learning algorithms.

**Statement of the Problem**

 Software defects are bugs, errors, malfunctions, flaws, faults, or inaccuracies in software that lead to an unpredicted or erroneous output (Cao, 2020). Faults are indispensable properties that exist in a system. They occur from scheme, manufacture or external environment. The software flaws are errors in programming which result in diverse performances compared with anticipation (Rana, *et..al.* 2014). Most faults recorded are from source code or design, some errors are from the improper or wrong code creating from compilers (Babu, *et..al*., 2019). For software developers and clients, software faults are a serious threat. Software defects not only reduce software quality, it increase cost and also delay the development schedule (Yang, 2017). With this problem on ground, several researchers have developed software defect prediction system using various machine learning algorithms as classifier on different programming languages platform to solve the problem. For instance Perraut, *et..al* ( 2017), compared Naïve Bayes (NB), Support Vector Machine (SVM), Artificial Neural Network (ANN), Linear regression (LR) and K-Nearest Neighbourhood (KNN) on five datasets. The outcome of their experiment did not show a superior classifier at the identifying defects. Hence, this research is evaluating the performance of NB, SVM, ANN, KNN and Decision tree C4.5 as classifiers on ECLIPSE JDT CORE, ECLIPSE PDE UI, EQUINOX FRAMEWORK, LUCENE and MYLN datasets to know which one is going to perform best among them, using Stochastic Gradient Descent Algorithm for feature extraction and Python Programming language for implementation.

**Aim and Objective**

To evaluate the performance of selected machine learning algorithms for software defect prediction and identify the most effective classifier. The objectives are to:

1. select the relevant features from the acquired data using stochastic gradient descent algorithm
2. classify the obtained relevant features using NB, SVM, ANN, Decision tree (C4.5) and KNN
3. evaluate the performance of the classifiers in term of accuracy, sensitivity, precision and F1-score,

**Methodology**

**The Research Approach**

The proposed architecture for the performance evaluation of selected machine learning algorithms is presented in Figure 1. The research adopted the following stages:

**Data Acquisition**

This initial stage involved gathering necessary datasets for the research. The datasets were acquired from http://bug.inf.usi.ch/download.php, which is publicly available for use. For this research, the bug prediction dataset that has been acquired contain data about the following software systems: Eclipse JDT Core, PDE UI, Equinox Framework, Lucene and Myln. Each of these software contain different piece of information but the one with relevant parameter that suite the research will be used during the cause of implementation.

**Extraction of Features**

In this stage, an optimization algorithm called stochastic gradient descent was used to extract the relevant features from the acquired datasets. This process helped to identify the most important attributes for software defect prediction. The data acquired from source are in buggy form, therefore the task here will be the extraction of the features that will be relevant for the prediction and elimination of those that will not contribute to the effective performance of the prediction. Feature extraction identified and extracts the most useful features of the dataset for learning, and these features will be applied as valuable features for prediction.

**Classification**

The third stage involved the classification of the extracted features. This was done using five machine learning algorithms: Support Vector Machine (SVM), Artificial Neural Network (ANN), Naïve Bayes (NB), Decision Tree (C4.5), and K-Nearest Neighbourhood (KNN).

**System Evaluation:**

 In the final stage, the performance of each classifier was evaluated based on four metrics: accuracy, precision, sensitivity, and F1-score. This comprehensive evaluation allowed for a thorough comparison of the different algorithms' effectiveness in software defect prediction.

In evaluating the performance of selected machine learning algorithms for software defect prediction systems, various measures were used to assess effectiveness. The performance evaluation was based on the following classification evaluation parameters:

1. **Accuracy** - It shows the rate of software defects that are predicted correctly over the total number of software defects. It is given as:

$Accuracy=\frac{TP+TN}{TP+TN+FP+FN}$

Where:

TP (True Positive) = Number of defects correctly identified as defects

TN (True Negative) = Number of non-defects correctly identified as non-defects

FP (False Positive) = Number of non-defects incorrectly identified as defects

FN (False Negative) = Number of defects incorrectly identified as non-defects

1. **Precision** - It denotes the percentage of software that were predicted to be software

defects by the classification algorithm. It shows the exact correctness. It is given as:

$Precision=\frac{TP}{TP+FP}$

Where:

TP (True Positive) = Number of defects correctly identified as defects

FP (False Positive) = Number of non-defects incorrectly identified as defects

This formula calculates the ratio of correctly identified defects to the total number of instances predicted as defective. It measures the classifier's ability to avoid labeling non-defective software as defective.

To express precision as a percentage, the result can be multiplied by 100

1. **Sensitivity** - It denotes the percentage of software that were predicted to be having defects and classified as defects. It is given as:

$Recall=\frac{TP}{TP+FN}$

1. **F1-score** - It is defined as the harmonic mean of Precision and Recall. It is given as:

$F-measure=\frac{2\*Precion\*Recall}{Precision+Recall}$

Where

 TP = True positive

 TN = True negative

 FP = False positive

 FN = False negative.



Figure 1: The Proposed Architecture of Software Defect Prediction System

The proposed model utilizes various machine learning algorithms, including Naive Bayes (NB), Support Vector Machine (SVM), Artificial Neural Network (ANN), K-Nearest Neighbourhood (KNN), and Decision Tree (C4.5), to classify software defects. The Stochastic Gradient Descent Algorithm (SDGA) plays a crucial role in feature selection, which significantly impacts the classification's accuracy, precision, sensitivity, and F1-score. The interpretation of the model's performance will involve a detailed analysis of how each algorithm processes the selected features, its strengths in identifying software defects, and its potential limitations.

**Conclusion**

This study aims to evaluate the performance of selected machine learning algorithms in predicting software defects. The use of SDGA for feature selection is expected to enhance the model's accuracy and reliability. Through this research, we hope to identify the most effective algorithm for software defect prediction, which will aid developers in creating more reliable and efficient software systems.

**Recommendations**

1. Adopt the Best-Performing Algorithm: Based on the evaluation results, it is recommended that software developers utilize the algorithm that demonstrates the highest accuracy and reliability in defect prediction.
2. Continuous Monitoring and Updates: Regularly update the predictive model with new data and retrain the algorithms to ensure they adapt to new software environments and emerging defects.
3. Incorporate AI-Based Tools: Integrate the best-performing machine learning models into AI-based tools that can be easily accessed by developers to streamline the defect prediction process.

**References**

Babu, N., Himagiri, Vamshi Krishna, V., Anil Kumar, A., & Ravi, M. (2019). Software defect prediction analysis by using machine learning algorithms. *International journal of recent technology and engineering*, *8*( 3544–3546. https://doi.org/10.35940/ijrte.B1438.0982S1119

Cao, H. (2020). A Systematic Study for Learning-Based software defect prediction. *Journal of physics: Conference series*, *1487*(1), 1–10. https://doi.org/10.1088/1742-6596/1487/1/012017

Kumar, G., & Bhatia, P. K. (2014). A detailed review of feature extraction in image processing Systems. *Internationa conference on advanced computing & Communication*, 5–12. https://doi.org/10.1109/ACCT.2014.74

Novaković, J., Strbac, P., & Bulatović, D. (2011). Toward optimal feature selection using ranking methods and classification algorithms. *Yugoslav journal of operations research*, *21*(1), 119–135. https://doi.org/10.2298/YJOR1101119N

Perreault, L., Berardinelli, S., Izurieta, C. & Sheppard, J. (2017). Using classifiers for software aefect

Rana, R., Staron, M. Berger, C., Hansson, J., Nilsson, M., & Meding, W. (2014). The adoption of machine learning sechniques for software defect prediction: An initial industrial validation. *Communications in computer and information science*, *466 CCIS*, 270–285. https://doi.org/10.1007/978-3-319-11854-3\_23

Rao, S. V. A., & Patra, P. S. K. (2020). An artificial neural network genetic algorithm with shuffled frog leap algorithm for software defect prediction. *International journal of innovative technology and exploring engineering (IJITEE)*, *9*(2), 831–836. https://doi.org/10.35940/ijitee.B6706.129219

Singh, M. & Salaria, D. S. (2013). Software defect prediction tool based on neural network. *international journal of computer applications*, *70*(22), 22–28.

Telgaonkar, A. H & Deshmukh, S. (2015). Dimensionality reduction and classification through PCA and LDA. *International journal of computer applications*, *122*(17), 4–8.

Varghese, N., & Verghese, V. (2012). A SURVEY OF DIMENSIONALITY REDUCTION AND CLASSIFICATION. *International journal of computer ecience & engineering survey*, *3*(3), 45–54.

Yang, Y. (2017). Software defect prediction model research for Network and cloud software development. *5th International conference on mechatronics, materials, chemistry and computer engineering (ICMMCCE 2017)*, *141*, 717–723. https://doi.org/10.2991/icmmcce-17.2017.131