Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Software Defect Prediction across Multiple Datasets using Classification Techniques

[1]Mrs. A. Priyadarshini, [2]Dr. V. Krishnapriya

[1]Research Scholar, Sri Ramakrishna College of Arts &Science and Assistant Professor ,PSGR Krishammal College for women, Coimbatore.

[2]Associate Professor &Head Department of Computer Science with Cognitive Systems, Sri Ramakrishna College of Arts &Science, Coimbatore.

[1]ndpriya7@gmail.com

ARTICLE INFO

Received: 17 Dec 2024

Revised: 14 Feb 2025

Accepted: 23 Feb 2025

ABSTRACT

Defect prediction in the early phases of the software development life cycle is a critical activity of the quality assurance process that has been extensively researched over the last two decades. Early detection of faulty modules in software development can assist the development team in making efficient and effective use of available resources to provide high-quality software products in a short period of time. The machine learning technique, which works by detecting hidden patterns among software features, is an excellent way to discover problematic modules. The results showed that Priority Based Fuzzy SVM yields 96% of accuracy while different defect datasets are taken into account compared to other techniques such as decision tree, PART, Random Forest, Naive Bayes and SVM. The mentioned existing techniques are used for prediction using same data sets which are taken for proposed work.

Keywords: Defect Prediction, Fault Modules, Hidden Patterns, Machine Learning

INTRODUCTION

Software defects undermine software quality and pose a serious threat to its reliability. While the size of the dataset is larger and more complicated as software development proceeds, making defects hide deeper and more difficult to uncover. As the software replicates, hidden defects might cause more server difficulties, ultimately in increased server implications when errors occur. Long-term research and practise have proven that the earlier software faults are discovered, the cheaper the cost of resolving them and the greater the amount of damage that may be restored.

The cost of identifying and resolving problems in the early coding phase of software development is one to two orders of magnitude cheaper than the cost of finding and fixing defects in the later testing or release phase of software development. As a result, both academics and business are interested in detecting software flaws early and repairing them at a cheaper cost. In this context, reliable prediction of faulty modules during the early phases of software development has emerged as a major technological issue that must be addressed. In general, there are two types of software defect prediction: dynamic defect prediction and static defect prediction.

A running system is usually required for static defect prediction. However, because the script is still not ready to implement, dynamically fault prediction is not applicable as during early coding stages of software development. Using approaches such as machine learning, static defect prediction may anticipate problematic modules in a software system without executing the programme. As a result, static defect prediction is perfect for detecting bugs in a programme at the early phases of software development.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Logistic regression, decision trees, Bayesian approaches, artificial neural networks, and support vector machine (SVM are examples of standard machine learning methods for static defect prediction. To develop a defect statistical method, these procedures require learning a large number of marker samples. However, marker samples must be made manually by studying the code, which takes time. Furthermore, obtaining numerous marker samples from a new project without a historical version is very hard when the amount of code in the early phases of development is already little. As a result, in the early phases of software development, defect prediction frequently encounters inadequate marker samples.

This paper is organized as follows. In Section II, a detailed description of machine learning techniques involved in predicting software defects are given. Sections III describe the proposed framework for software prediction with different datasets. Experimental results are presented in Section IV. Finally, section V concludes the paper.

LITERATURE SURVEY

During testing activities, software defect prediction (SDP) is critical in the early phases of defect-free software development. SDP that is effective can assist test managers in locating problems and defect-prone software modules. This allows for the most efficient and cost-effective utilisation of limited software quality assurance resources. The problem of feature selection (FS) is difficult and has a polynomial time complexity. The total search space for a dataset with N features includes 2N feature subsets, implying that the technique requires an exponential running time to traverse all of these feature subsets.

To address the issue of class imbalance, SMOTE-based oversampling techniques are frequently used in the field of SDP. Permutation is created when SMOTE-based oversampling methods are used [1]. However, because SMOTE-based interpolation techniques were popular baseline oversampling techniques, comparing the performance of SMOTE-based oversampling techniques to that of freshly presented approaches is suspicious and less compelling. They evaluated the performance of SMOTE-based oversampling strategies empirically in their work.

The author discovers that the performance of SMOTE-based oversampling algorithms is very unstable, and this instability has a detrimental influence on prediction model performance in terms of AUC, balance, and MCC on the KNN, SVM, RF, and DT classifiers. We present a series of robust SMOTE-based oversampling techniques to increase the stability of SMOTE-based oversampling techniques. Their methodologies were theoretically and experimentally confirmed to generate more stable and superior outcomes than SMOTE-based oversampling strategies. As a result, the authors advocate that stable SMOTE-based oversampling techniques be used as an effective substitute for SMOTE-based oversampling approaches.

Pandey, S.K. et colleagues [2] ran 864 tests over three public datasets, analysing the noise endure for well-known SDP models. They have manually added noise ranging from 0% to 80%. They employed four baseline SDP approaches and trained them on noisy datasets. To avoid the problem of class imbalance, the author employed random sampling. They also proposed a strategy that can handle high levels of noise while still outperforming baseline solutions. They discovered that the suggested strategy outperforms baseline technology with noisy cases and unbalanced data.

Software defect prediction using machine learning approaches is now regarded as one of the most promising research fields. Defect detection at an early stage of development can contribute to the delivery of high-quality software while utilising limited resources [3]. The purpose of this research is to conduct a comprehensive performance examination of several machine learning classification approaches on software defect prediction utilising 12 well used and publicly available NASA datasets. Nave Bayes (NB), Multi-Layer Perceptron (MLP), and other classification approaches are used (MLP).

Radial Basis Function (RBF), Support Vector Machine (SVM), K Nearest Neighbor (KNN), kStar (K*), One Rule (OneR), PART, Decision Tree (DT), and Random Forest are all examples of machine learning algorithms (RF).

Precision, Recall, F-Measure, Accuracy, MCC, and ROC Area are some of the metrics taken from the confusion matrix that are used to evaluate performance. The results show that neither the Accuracy nor the ROC can be utilised as an effective performance metric since they did not respond to the class imbalance issue.

Zhu, K et al [4] stated that when the data under examination contains a large number of characteristics, dimensionality reduction and deep learning can be advantageous. They want to enhance the neural network model in the future by adjusting different factors such as the number of hidden layers, neurons in each layer, optimizers, and the cost function. Various alternative strategies must be tried in order to reduce the dimensionality of the parameters, and according to current methodologies, altering the number of components used can also result in some improvement. They also plan to test other classifiers such as Naive Bayes, K-Nearest Neighbours, Kernel Support Vector Machines and ensemble techniques such as Random Forest and compare the results to those found by using Decision Tree classifier.

GHOST is a novel approach that combines deep learning with two helpful expansions, according to Yedida, R et al [5]. (a) Loss functions that are weighted (b) a unique fuzzy sampling strategy (see 3.3). SMOTE (c) They proved the effectiveness of our technique on 10 defect prediction datasets utilising four metrics, as well as within project and cross-project defect prediction data as analysed by Wang et al. The authors compared their results to three baselines: (a) a conventional deep learner, (b) a prior state-of-the-art result in defect prediction using non-deep learning methods, and (c) a prior state-of-the-art result in extracting features from code using deep learning.

Chen L et al [6] devised a unique technique for dealing with both class overlap and imbalance difficulties in SDP. To begin, using the neighbour cleaning procedure, the overlapping majority samples are deleted. After that, the number of classes is balanced using ensemble random under-sampling, and an effective prediction model is constructed. They picked nine extremely imbalanced datasets from a public SDP repository to evaluate their proposed model, and they ran a thorough empirical inquiry on current SDP models, including conventional classifiers, imbalance learning techniques, and data cleaning methods, in the trials.

Hammouri et al [7] created many methods that make use of various datasets, metrics, and performance measurements. There were three machine learning techniques used: NB, DT, and ANNs. In the assessment approach, three genuine testing/debugging datasets are employed. To collect experimental findings, the metrics accuracy, precision, recall, F-measure, and RMSE are employed. The findings demonstrate that ML approaches are useful tools for forecasting future software difficulties. The results of the comparison indicated that the DT classifier outperformed the others. Furthermore, experimental findings indicated that employing ML technique gives superior prediction model performance than other approaches, such as linear AR and POWM model.

Matloob F et al [8] utilised SLR to track the most current research advances in ensemble learning techniques for software fault prediction. Following a careful analysis of the most important research articles published in three well-known online libraries: ACM, IEEE, Springer Link, and Science Direct, this review is carried out. It is concluded that ensemble learning methodologies greatly outperformed individual classifiers. An assessment of the effects of feature selection procedures on ensemble learning is required. Furthermore, the variety of classifiers should be investigated while creating the ensemble model in order to increase the efficacy of the ensembles.

Iqbal, A., and Aftab et al [9] introduced a multi-filter feature selection based classification approach for software fault prediction. For defect prediction, the system makes use of Machine Learning Techniques (MLP). In addition, the framework leverages the oversampling approach to study the effect of class imbalance on classification performance. The experiment makes use of the following NASA MDPI cleaned datasets: "CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4, and PC5. The performance of the proposed framework is compared to that of ten well-known supervised classification approaches: "Nave Bayes (NB), Multi-Layer Perceptron (MLP), Radial Basis Function (RBF), Support Vector Machine (SVM), K Nearest Neighbor (KNN), kStar (K*), One Rule (OneR), PART, Decision Tree (DT), and Random Forest (RF)."

The above discussion showed different methodologies carried out for defect prediction in the software dataset. In our proposed work, priority based fuzzy SVM is used for predicting defects in various data sets taken into account. The following section elaborates the proposed methodology, and results obtained for performance measures.

METHODOLOGY

The purpose of this research is to investigate and evaluate Machine Learning algorithms: Naive Bayes (NB), Random Forest, PART, Support Vector Machine (SVM) and Decision Tree (DT). The study demonstrates the accuracy and capabilities of the ML techniques are used in the prediction of software defects and give a comparison of the selected ML algorithms. The supervised machine learning algorithms endeavour to create an inferring function based on relationships and relationships between the system's known inputs and outputs labelled training data, allowing us to predict output values for fresh input data based on the inferring function obtained

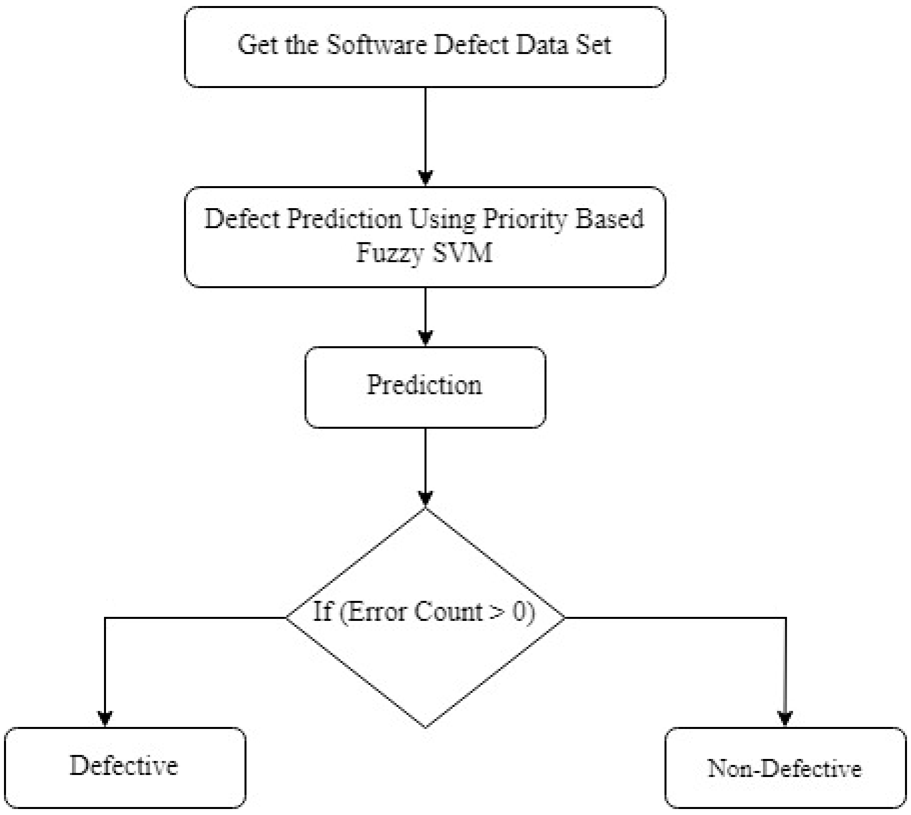

Get the Software Defect Data Set
Defect Prediction Using Priority Based Fuzzy SVM
Prediction
If (Error Count > 0)
Defective
Non-Defective

Fig 3.1 Flow Chart for Proposed Methodology

Fig 3.1 shows the flow of the proposed framework. It depicts that defected data set is taken as input. The data set having defections are predicted by priority based fuzzy SVM technique. If the error count is greater than zero the data set is defective otherwise non defective.

A. Data Set Pre Processing

The acquired data sets were pre-processed to prepare them for subsequent machine learning approaches.

B. Apply Naive Bayes, Random Forest and SVM

Naive Bayes (NB): NB is a simple and efficient probabilistic classifier based on the Bayes theorem and the independence assumption between features. NB is a family of algorithms based on a common premise, which asserts that the presence or lack of a certain characteristic of the class is unrelated to the presence or absence of any other features.

Decision Tree (DT): DT is a common learning method used in data mining. DT refers to a hierarchal and predictive model which uses the item's observation as branches to reach the item's target value in the leaf. DT is a tree with decision nodes, which have more than one branch and leaf nodes, which represent the decision.

The Naive Bayes Classifier is a basic and effective Classification method that aids in the development of rapid machine learning models capable of making quick predictions. It is a probabilistic classifier, which means it predicts based on an object's likelihood. A random forest algorithm is made up of several decision trees. The random forest algorithm generates a "forest" that is trained via bagging or bootstrap aggregation. Bagging is a meta-algorithm that increases the accuracy of machine learning algorithms using an ensemble approach. The outcome is determined by the (random forest) algorithm based on the predictions of the decision trees. It forecasts by averaging or averaging the output of several trees. The precision of the output improves as the number of trees grows.

SVM is a supervised machine learning technique that may be used to solve classification and regression problems. Individual observation coordinates are used to calculate support vectors. The SVM classifier is a frontier that best distinguishes between the two classes (hyper-plane/line). The pre-processed data sets are run through three machine learning algorithms: Naivy Bayes, Random Forest, and Support Vector Machine.

C. Proposed Priority Based Fuzzy SVM

The FSVM algorithm may assign fuzzy membership values to distinct samples to represent their relevance for their own class, with more significant samples receiving higher fuzzy membership values and less important samples receiving lower fuzzy membership values, such as outliers or noise.

D. Performance Evaluation

Classification accuracy, commonly known as the right classification rate, is one of the most important basic measures for evaluating the effectiveness of predictive models. It is used to calculate the proportion of correctly categorised cases to total occurrences.

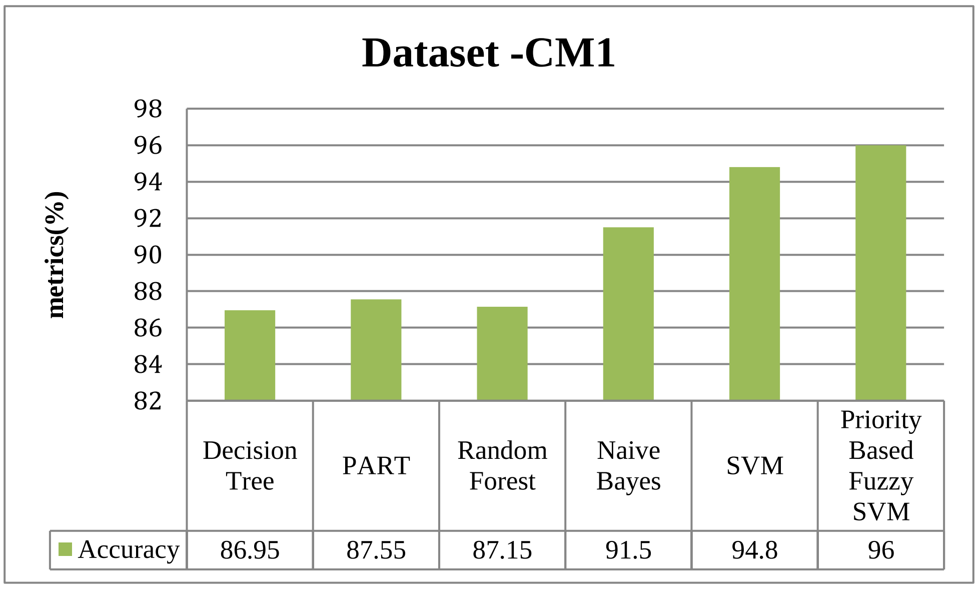
I. RESULTS AND DISCUSSIONS

The used datasets in this study are three different datasets, namely DS1, DS2 and DS3. Accuracy (ACC) is the proportion of true results (both TP and TN) among the total number of examined instances. The best accuracy is 1, whereas the worst accuracy is 0. ACC can be computed by using the following formula: ACC = (TP + TN) / (TP + TN+ FP + FN)

The following table shows the accuracy values obtained for various machine learning algorithms used and for the dataset CM1.

TABLE I ACCURACY FOR THE USED ML ALGORITHMS IN THE DATASET – CM1

Algorithm name	Accuracy
Decision Tree	86.95
PART	87.55
Random Forest	87.15
Naive Bayes	91.5
SVM	94.8
Priority Based Fuzzy SVM	96



	Decision Tree	PART	Random Forest	Naive Bayes	SVM	Priority Based Fuzzy SVM
🟩 Accuracy	86.95	87.55	87.15	91.5	94.8	96

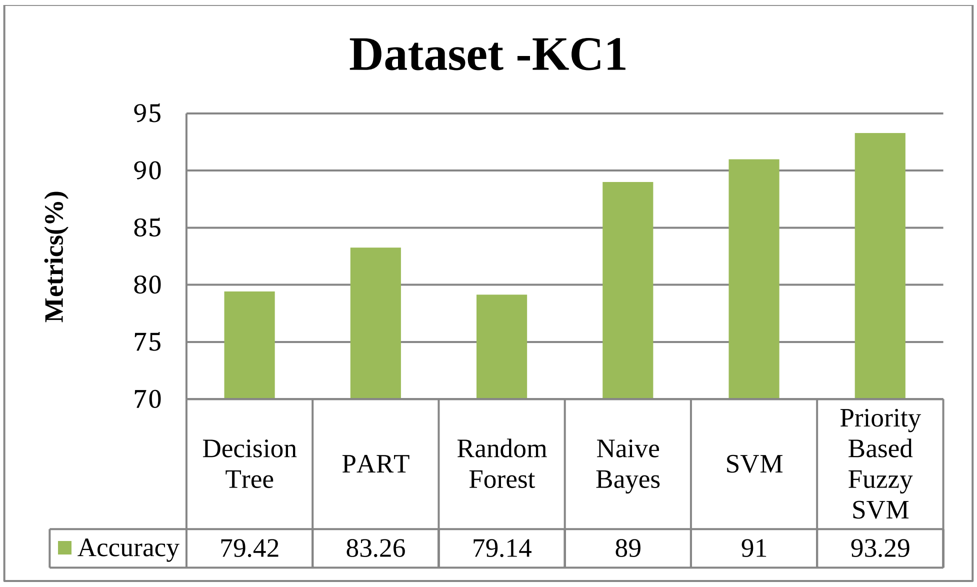
Fig 4.1 Accuracy for the used ML algorithms in the dataset – CM1

The following table shows the accuracy values obtained for various machine learning algorithms used and for the dataset KC1.

TABLE II ACCURACY FOR THE USED ML ALGORITHMS IN THE DATASET – KC1

Algorithm name	Accuracy
Decision Tree	79.42
PART	83.26
Random Forest	79.14

Naive Bayes	89.0
SVM	91.0
Priority Based Fuzzy SVM	93.29



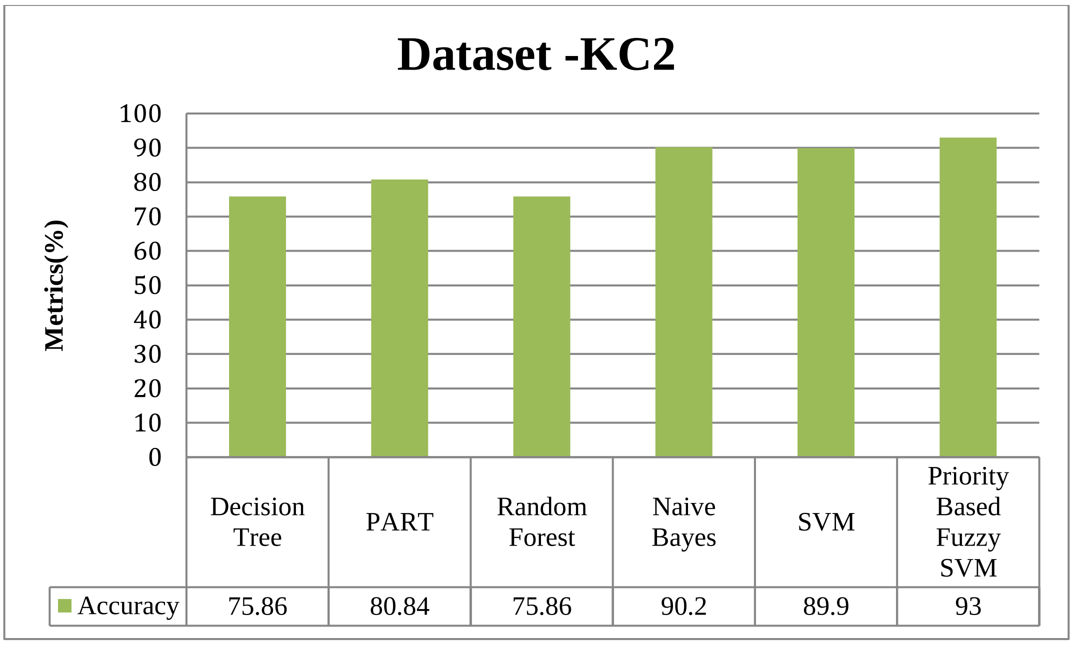
	Decision Tree	PART	Random Forest	Naive Bayes	SVM	Priority Based Fuzzy SVM
🟩 Accuracy	79.42	83.26	79.14	89	91	93.29

Fig 4.2 Accuracy for the used ML algorithms in the dataset – KC1

The following table shows the accuracy values obtained for various machine learning algorithms used and for the dataset KC2.

TABLE III ACCURACY FOR THE USED ML ALGORITHMS IN THE DATASET – KC2

Algorithm name	Accuracy
Decision Tree	75.86
PART	80.84
Random Forest	75.86
Naive Bayes	90.2
SVM	89.9
Priority Based Fuzzy SVM	93



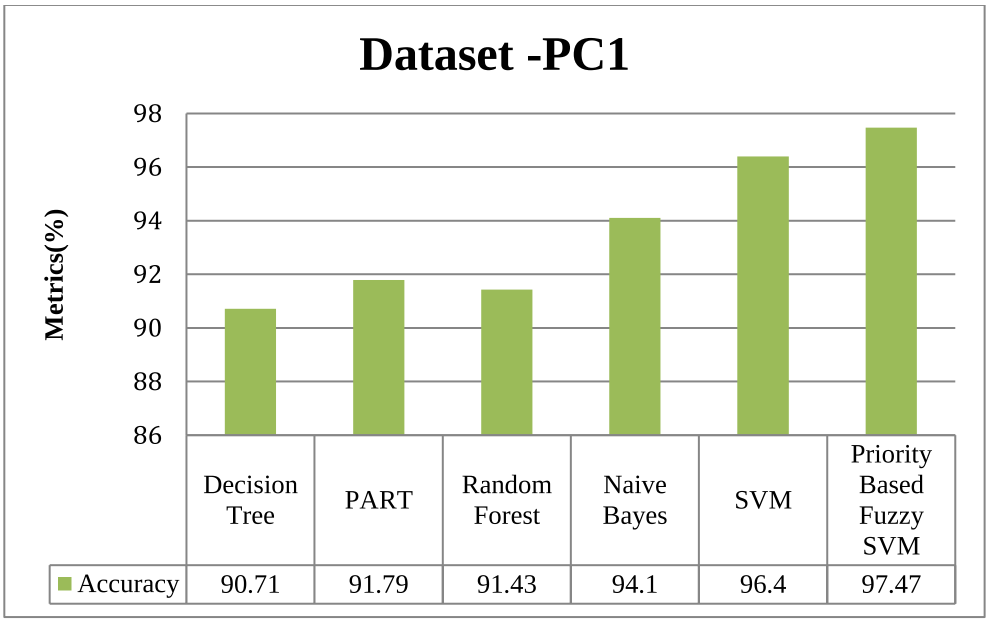
	Decision Tree	PART	Random Forest	Naive Bayes	SVM	Priority Based Fuzzy SVM
🟩 Accuracy	75.86	80.84	75.86	90.2	89.9	93

Fig 4.3 Accuracy for the used ML algorithms in the dataset – KC2

The following table shows the accuracy values obtained for various machine learning algorithms used and for the dataset KC2.

TABLE IV ACCURACY FOR THE USED ML ALGORITHMS IN THE DATASET – PC1

Algorithm name	Accuracy
Decision Tree	90.71
PART	91.79
Random Forest	91.43
Naive Bayes	94.1
SVM	96.4
Priority Based Fuzzy SVM	97.47



	Decision Tree	PART	Random Forest	Naive Bayes	SVM	Priority Based Fuzzy SVM
🟩 Accuracy	90.71	91.79	91.43	94.1	96.4	97.47

Fig 4.4 Accuracy for the used ML algorithms in the dataset – PC1

The Accuracy measures for applying DT, PART, Random Forest, NB, SVM and Priority Based Fuzzy SVM classifiers on CM1, KC1, KC2 and PC1 datasets are shown in Table I, II, III and IV. Results show that these ML algorithms can be used for defect prediction effectively with a good accuracy. The average accuracy values for all classifiers in the four datasets are more than 96%.

CONCLUSION

Software Defect prediction is a technique in which a prediction model is created in order to predict the future software faults based on data. Various approaches have been proposed using different datasets, different metrics and different performance measures. This paper evaluated the using of machine learning algorithms in software defect prediction problem. Machine learning techniques have been used, which are NB, DT, PART, Random Forest, SVM and Priority Based Fuzzy SVM. The evaluation process is implemented using three real testing/debugging datasets.

Experimental results are collected based on accuracy measures. Results reveal that the ML techniques are efficient approaches to predict the future software defects. The comparison results showed that the Priority Based Fuzzy SVM classifier has the best results over the others. As a future work, we may involve deep learning techniques and provide an extensive comparison among them. Furthermore, adding more software metrics in the learning process is one possible approach to increase the accuracy of the prediction model.

REFERENCES

[1]. Feng, S., Keung, J., Yu, X., Xiao, Y. and Zhang, M., 2021. Investigation on the stability of SMOTE-based oversampling techniques in software defect prediction. *Information and Software Technology*, *139*, p.106662.

[2]. Pandey, S.K. and Tripathi, A.K., 2021. An empirical study toward dealing with noise and class imbalance issues in software defect prediction. *Soft Computing*, *25*(21), pp.13465-13492.

[3]. Iqbal, A., Aftab, S., Ali, U., Nawaz, Z., Sana, L., Ahmad, M. and Husen, A., 2019. Performance analysis of machine learning techniques on software defect prediction using NASA datasets. *International Journal of Advanced Computer Science and Applications*, *10*(5).

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

[4]. Zhu, K., Ying, S., Zhang, N. and Zhu, D., 2021. Software defect prediction based on enhanced metaheuristic feature selection optimization and a hybrid deep neural network. *Journal of Systems and Software*, *180*, p.111026.

[5]. Yedida, R. and Menzies, T., 2021. On the value of oversampling for deep learning in software defect prediction. *IEEE Transactions on Software Engineering*.

[6]. Chen L, Fang B, Shang Z, Tang Y. Tackling class overlap and imbalance problems in software defect prediction. Software Quality Journal. 2018 Mar;26(1):97-125.

[7]. Hammouri A, Hammad M, Alnabhan M, Alsarayrah F. Software bug prediction using machine learning approach. International journal of advanced computer science and applications. 2018 Apr;9(2):78-83.

[8]. Matloob F, Ghazal TM, Taleb N, Aftab S, Ahmad M, Khan MA, Abbas S, Soomro TR. Software Defect Prediction Using Ensemble Learning: A Systematic Literature Review. IEEE Access. 2021 Jul 8.

[9]. Iqbal, A. and Aftab, S., 2020. A Classification Framework for Software Defect Prediction Using Multi-filter Feature Selection Technique and MLP. *International Journal of Modern Education & Computer Science*, *12*(1).

[10]. Ali, A. and Gravino, C., 2019. A systematic literature review of software effort prediction using machine learning methods. *Journal of software: evolution and process*, *31*(10), p.e2211.

[11]. Prabha, C.L. and Shivakumar, N., 2020, June. Software defect prediction using machine learning techniques. In *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)* (pp. 728-733). IEEE.

[12]. Malhotra, R., 2016. An empirical framework for defect prediction using machine learning techniques with Android software. *Applied Soft Computing*, *49*, pp.1034-1050.

[13]. Jaiswal, A. and Malhotra, R., 2018. Software reliability prediction using machine learning techniques. *International Journal of System Assurance Engineering and Management*, *9*(1), pp.230-244.

[14]. Sharma, D. and Chandra, P., 2018. Software fault prediction using machine-learning techniques. In *Smart computing and informatics* (pp. 541-549). Springer, Singapore.

[15]. Matloob, F., Ghazal, T.M., Taleb, N., Aftab, S., Ahmad, M., Khan, M.A., Abbas, S. and Soomro, T.R., 2021. Software defect prediction using ensemble learning: A systematic literature review. *IEEE Access*.

[16]. Prabha, C.L. and Shivakumar, N., 2020, June. Software defect prediction using machine learning techniques. In *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)* (pp. 728-733). IEEE.