Utilizing Supervised Learning Techniques for Program Error Detection: A Comprehensive Approach

Kuruva Renuka

Department of Computer Science Sri Venkateswara University, Tirupati

Abstract— *Detecting software errors is a critical task for large software development organizations, especially as the complexity of software projects continues to increase rapidly. Accurately assessing the severity of defects encountered in large and intricate software endeavors can aid software professionals in allocating resources effectively and planning subsequent defect resolution activities. This study aims to evaluate the performance of machine learning systems in software error prediction, employing Support Vector Machine, Naive Bayes, Multilayer Perceptron, Random Forest, and K-Nearest Neighbor algorithms. The effectiveness of these algorithms is assessed based on key performance metrics: accuracy, precision, and recall. Among the five algorithms, the Support Vector Machine model achieved the highest overall accuracy rate of 93%. The proposed model is evaluated using datasets from the PROMISE Software Engineering Repository. Results indicate that the model performs exceptionally well in predicting high severity software defects compared to defects of other severity levels.*

I. INTRODUCTION

Predicting software defects is a crucial aspect of software development; anticipating bugs early enhances software adaptability to various conditions and optimizes resource utilization. Software defect prediction identifies areas in the source code where deficiencies are likely to occur. Predicting defects in computer programs involves identifying segments of a software structure that may contain errors. Early identification of a defect leads to efficient allocation of resources, reduces time and cost in software development, and ensures high-quality software. Numerous artificial intelligence techniques have been proposed for software defect prediction.

AI methods are widely employed in software defect prediction to forecast defective modules based on sound defect data, empirical assessments, and sophisticated software engineering techniques. Predicting defects at an early stage improves software quality, reliability, and efficiency, while reducing development costs. Software defect prediction models are developed using two approaches: utilizing statistical properties of the software structure, known as software metrics, and leveraging defect data from previous software projects. Once established, the defect prediction model can be applied to future software projects, enabling developers to identify defect-prone areas of a software structure.

Software defect prediction, a process of anticipating code sections that may contain defects, enables developers to prioritize their testing efforts by first inspecting potentially faulty code. Predicting defective units is essential to ensure the reliability of large-scale software development projects. Identifying defects in software code becomes increasingly challenging due to the vast size, numerous lines of code, and intricate design of typical software projects. The significance and challenges of software defect prediction have made it a prominent research area in software engineering. With software systems becoming more complex and sophisticated, traditional software development approaches may encounter difficulties in meeting these evolving requirements.[1][3][7][8][9][10][11][12].

II. SUPERVISED LEARNING TECHNIQUES

AI (ML) is a piece of man-made awareness that gets data from planning data subject to deep rooted real factors. ML is described as an assessment that licenses PCs to learn data without being changed. There are a couple of ML techniques embraced to expect the attacks in the Test datasets which was used to set up the structure. These estimations were used to bunch the attacks in other to become familiar with a successful technique in expecting and requesting attacks. ML techniques are gathered into three general classes like directed learning and solo learning. Coordinated computations gains for expecting the thing class from pre-stamped (gathered) objects. Nevertheless, the independent estimation finds the ordinary social affair of articles given as unlabeled data. In this work, the premium is with the going with directed learning estimations like Multilayer Perceptron, Support Vector Machine, Random Forest, K-Nearest Neighbor and Naïve Bayes procedures are evaluated.

III. METHODOLOGY

At the present time clarified about directed learning procedures like Multilayer Perceptron, Support Vector Machine, Random Forest, K-Nearest Neighbor and Naïve Bayes system models for our Software Defect grouping issue.

3.1 Multilayer Perceptron (MLP)

Multi-facet Perceptron (MLP) is a champion among the most broadly perceived Neural Network plan that has been used for various applications. The MLP organize is commonly made out of different center points or dealing with units, and it is figured out into a movement of somewhere around two layers. The essential layer (or the most diminished layer) is named as an information layer where it gets the external information while the last layer (or the most surprising layer) is a yield layer where the response for the issue is gotten. The disguised layer is the widely appealing layer in the information layer and the yield layer, and may frame with something like one layer. The arrangement of MLP could be communicated as a nonlinear improvement issue. The objective of MLP learning is to find the best loads that limit the differentiation between the information and the yield [5]. The most pervasive getting ready computation used in NN is Back inducing (BP), and it has been used in dealing with various issues in model affirmation and portrayal. This estimation depends on a couple of boundaries, for instance, different covered centers at the hid layers learning rate, energy rate, actuation work and the quantity of preparing to happen. Moreover, these boundaries could change the presentation on the gaining from awful to great precision [10].

3.2 Naive Bayes

The Naive Bayes is a smart technique for creation of quantifiable perceptive models [6]. NB relies upon the Bayesian speculation. This portrayal system examinations the association between every trademark and the class for every guide to surmise an unexpected probability for the associations between the quality characteristics and the class [5]. During setting up, the probability of each class is figured by counting how oftentimes it occurs in the planning dataset. This is known as the "prior probability" P(C=c). Despite the previous probability, the estimation also enlists the probability for the event x given c with the doubt that the characteristics are free. This probability transforms into the consequence of the probabilities of each single characteristic. The probabilities would then have the option to be assessed from the frequencies of the events in the readiness set.

3.3 Support Vector Machines (SVM)

SVMs are a ton of related directed learning procedures that take apart data and see plans, used for request and backslide assessment. SVM is an estimation that undertakings to find a straight separator (hyper-plane) between the data reasons for two classes in multidimensional space. SVM addresses a learning methodology which adheres to principles of quantifiable learning speculation [5][6]. All around, the rule considered SVM begins from combined portrayal, to be explicit to find a hyperplane as a division of the two classes to restrict the gathering bungle. The SVM finds the hyperplane using support vectors and edges.

3.4 K-Nearest Neighbor (KNN)

The K-Nearest-Neighbors (KNN) is a straightforward however compelling strategy for arrangement. The KNN calculation is a strategy for grouping objects dependent on nearest preparing models in the component space. KNN is a sort of occasion based learning, or apathetic realizing where the capacity is just approximated locally and all calculation is conceded until grouping [6]

For an information record D to be ordered, its K closest neighbors are recovered, and these structures a neighborhood of D. Larger part casting a ballot among the information records in the area is generally used to choose the order for D with or without thought of distance-based weighting. In any case, to apply KNN we need to pick a suitable incentive for K, and the achievement of grouping is a lot of wards on this worth. The significant disadvantages regarding KNN are (1) its low productivity - being a languid learning strategy denies it in numerous applications, for example, dynamic web digging for a huge vault, and (2) its reliance on the choice of a "great worth" for K.

3.5 Random Forest

Discretionary forest area is a gathering learning system dependent upon depiction and break faith trees. Each tree is prepared on a bootstrap test, and ideal segments at each split are seen from a self-self-assured subset thing being what they are. Notwithstanding suspicion, self-confident trees can be utilized to evaluate variable significance measures to rank components by sensible significance. The sporadic forest area is utilized to get the fragment arranging qualities, and these attributes are applied to pick which features are disposed of in every emphasis of the assessment [2]. The structure consolidates the

progression of a gigantic number of decision trees and inside unusual trees; haphazardness is utilized in the going with ways: first thing, every decision tree is created utilizing another bootstrap test. Also, during the improvement of every choice tree, each middle split consolidates the inconsistent affirmation of a subset of k segments, of which the best split is settled. It is particularly useful for monstrous datasets with a couple of data features since it reduces the commotion, diverse nature and running period of the assessment.

IV. EXPERIMENTAL RESULTS

The objective of this space is to survey five AI computations with respect to execution systems. A total report has been coordinated to evaluate estimate execution of five ML computations using Software Defects dataset gotten from PROMISE Software Engineering Repository dataset which is obtained from NASA's Metrics Data Program data vault [4]. This is KC1 which is a C++ program that is included sound get-togethers of PC programming sections inside a gigantic ground system. The dataset subtleties are displayed in table-1 and the Statistical outline of the dataset as displayed in the figure-1.

Table-1
PROMISE Software Engineering Repository dataset

S. No	Dataset	No. of Features	No. of Instances	Class Division
1	KC1/software defect prediction	22	2109	Defective_326 Non-defective_1783

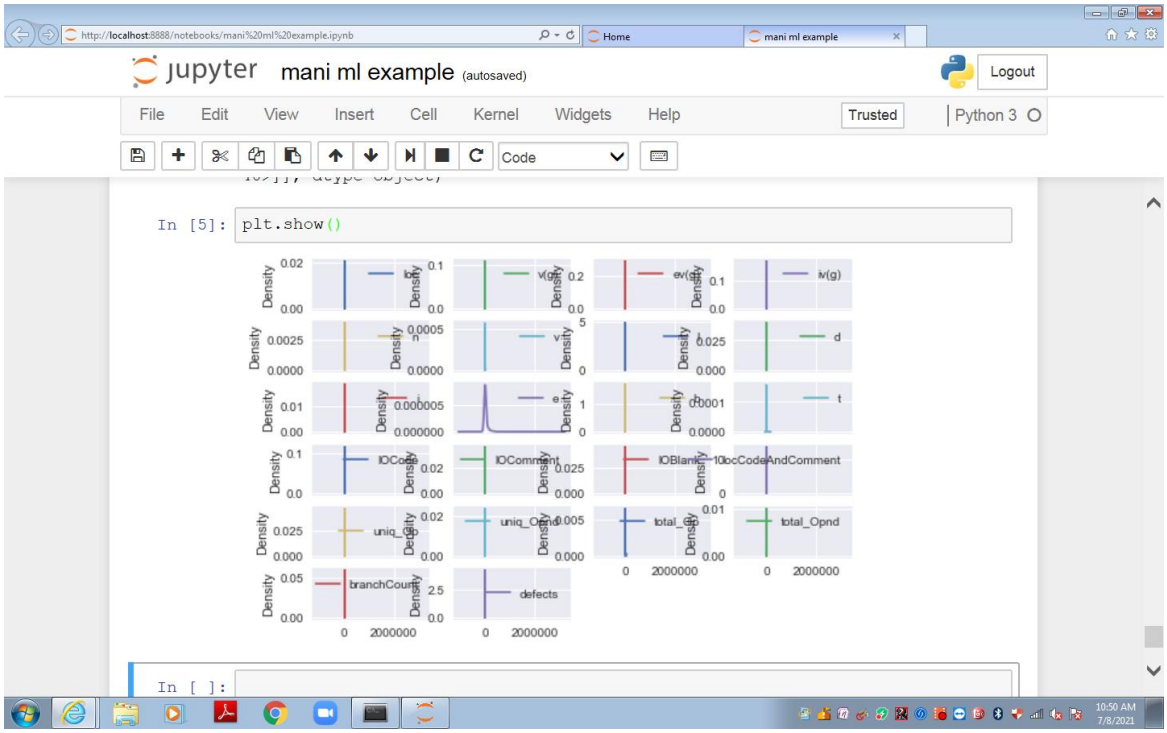

Figure-1: Statistical summary of the dataset

We have utilized Python programming to analysis of proposed structure. To support the gauge delayed consequences of the assessment of the proposed procedure, the 10-cover cross endorsement is used. The k-cover cross endorsement is regularly used to decrease the error came about due to unpredictable reviewing in the connection of the accuracy of different assumption models. The current appraisal isolated the data into ten folds where 1 wrinkle was for trying and nine folds were getting ready for the 10-overlay cross endorsement.

The dataset is separated in two sets. The planning set is 70% and the remaining 30% are used for testing. We have used the Python Programming to investigate five ML course of action estimations. We survey our five models using assorted execution estimations like Accuracy, Precision and Recall, the Experimental results are showed up in the table-2 and same showed up in the Figure-2.

Table-2
Performance of ML Classifiers

Algorithm	Accuracy	Precision	Recall
KNN	82	81	82
Naïve Bayes	85	92	85
Random Forest	89	81	89
MLP	87	76	87
SVM	93	94	93

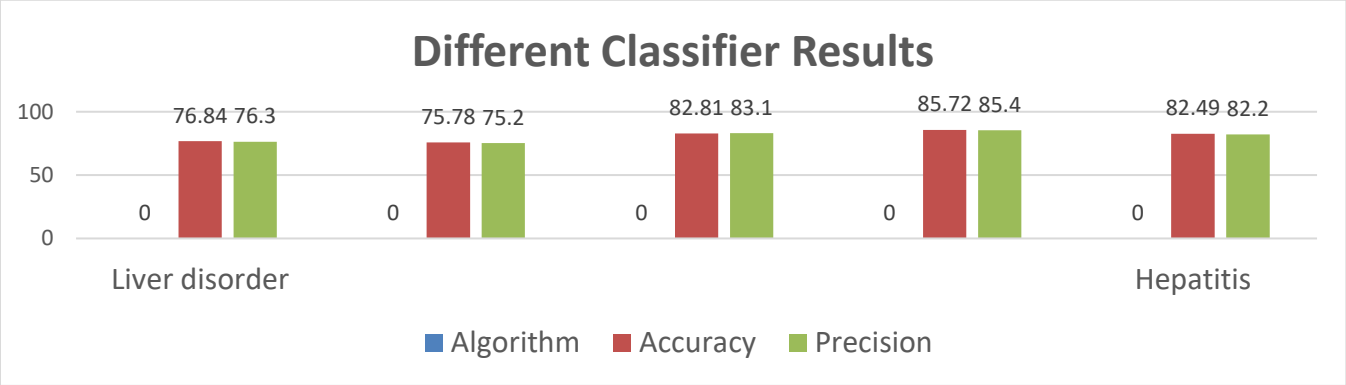

Figure-2: Results of different Classifiers

We find in the Figure-2, the introduction of the KNN estimation has accomplished 82% precision, Naïve Bayes has achieved 85%, Random Forest model has achieved 89%, MLP has 87% and SVM model has accomplished 93%. As the result from assessment among the five computations, we find that most vital precision of Classification model is SVM (93%). Precisely when veered from exactness and survey are in addition higher in the SVM model when appeared differently in relation to other four models.

The Experimental Results of SVM and KNN models screen shots are displayed in the figure-3 and figure-4.

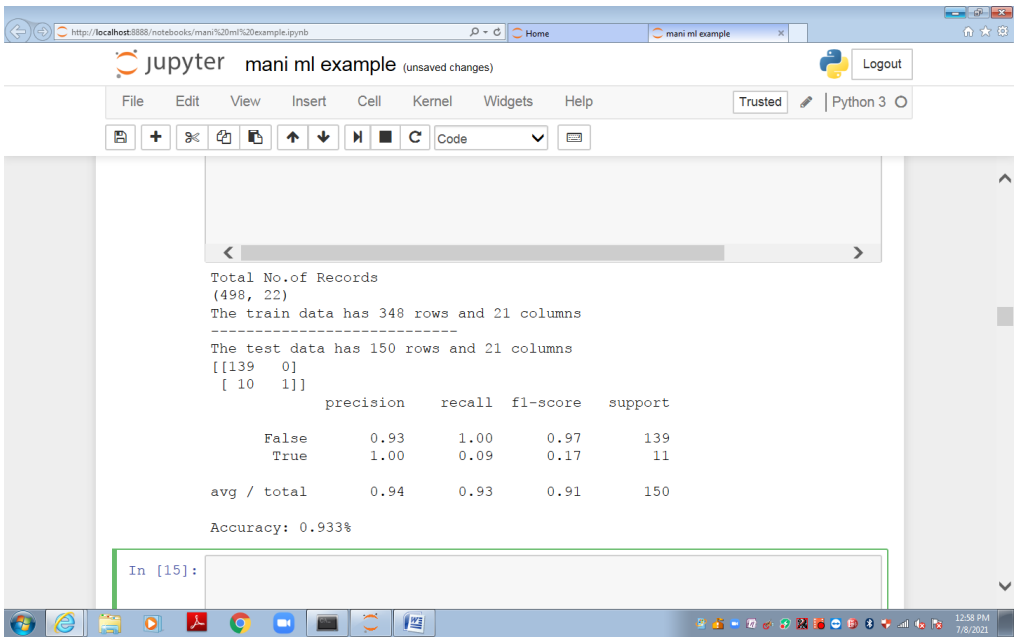

Figure-3: Experimental Results Screen Shot of SVM

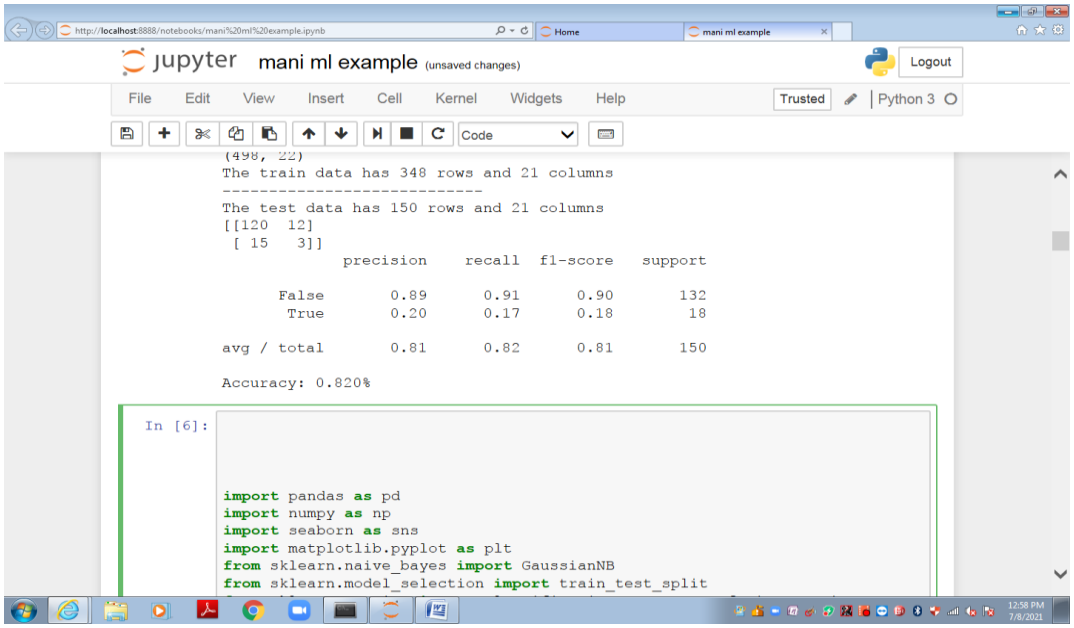

Figure-4: Experimental Results Screen Shot of KNN

V. CONCLUSION

Identifying and addressing defective software components in their early stages allows software developers to efficiently utilize time and resources, enhance project reliability, and improve software quality control processes. In this paper, we propose a novel approach to enhance the accuracy of predicting faulty software components. The proposed framework aims to develop a robust classifier to predict whether a Software Project will be defective or non-defective. Based on the evaluation of results, Support Vector Machine (SVM) achieved the highest prediction accuracy of 93%.

REFERENCES

[1] B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," in Proceedings of the 37th International Conference on Software Engineering-Volume 1, pp. 789-800, 2015, IEEE Press

[2] Breiman L. Random forests. Mach Learn 2001; 45:5–32

[3] Goseva-Popstojanova, Katerina, Mohammad Ahmad, and Yasser Alshehri. &Quot; Software fault proneness prediction with group lasso regression: on factors that affect classification performance. In 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), vol. 2, pp. 336-343. IEEE, 2019

[4] http://promise.site.uottawa.ca/SERepository

[5] Ian H. Witten and Eibe Frank. Data Mining: Practical machine learning tools and techniques. 2nd ed. San Francisco: Morgan Kaufmann, 2005.

[6] J. Han and M. Kamber," Data Mining concepts and Techniques", the Morgan Kaufmann series in Data Management Systems, 2nd ed. San Mateo, CA; Morgan Kaufmann, 2006.

[7] Kitchenham, B.A. Guidelines for Performing Systematic Literature Review in Software Engineering; Technical Report EBSE-2007-001; Keele University and Durham University: Staffordshire, UK, 2007

[8] M. Shepperd, D. Bowes, and T. Hall, "Researcher Bias: the use of machine learning in software defect prediction," IEEE Transactions on Software Engineering, vol. 40, no. 6, pp. 603-616, 2014.

[9] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and W. Meding, "Analyzing defect inflow distribution and applying Bayesian inference method for software defect prediction in large software projects," Journal of Systems and Software, vol. 117, pp. 229-244, 2016

[10] S. Haykin, Neural Networks – A Comprehensive Foundation, 2nd Edition, Pearson Education, Inc., Upper Saddle River, New Jersey 07458, 2000

[11] T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang, and A. Bener, "Defect prediction from static code features: current results, limitations, new approaches," Automated Software Engineering, vol. 17, no. 4, pp. 375–407, 2010.

[12] Y. Zhou, B. Xu, and H. Leung, "On the ability of complexity metrics to predict fault-prone classes in object-oriented systems," Journal of Systems and Software, vol. 83, no. 4, pp. 660-674, 2010.