Software Fault Prediction using Machine Learning
Vineetha M K1 Dr. G. N. Srinivasan2
1Student 2Professor
1,2Department of Information Science & Engineering
1,2R V College of Engineering, Bengaluru, India

Abstract—Software Fault Prediction assumes a critical job in the dynamic research regions of software building. A product fault is a blunder, bug, imperfection, blame, glitch or mix-up in programming that makes it make a wrong or startling result. The real hazard factors related with a product fault which isn't recognized amid the early period of programming improvement are time, quality, cost, exertion and wastage of assets. Faults may happen in any period of programming advancement. Booming programming organizations center fixation around programming quality, especially amid the early period of the product improvement. Thus, the key target of any association is to decide and address the imperfections in an early period of Software Development Life Cycle [SDLC]. To improve the nature of programming, machine learning strategies have been connected to assemble expectations regarding the disappointment of programming parts by misusing past information of programming segments and their deformities. This paper investigated the condition of craftsmanship in the field of programming imperfection the board and prediction and offered machine learning procedures in short.

Keywords: Software Fault Prediction, Machine Learning

I. INTRODUCTION

Software testing is an essential period of software development life cycle that confirm and approve programming quality. One approach to screen software quality is to discover programming deficiencies or flaws and after that right those flaws. In any case, testing process is a tedious action that makes it difficult to discover all flaws inside given assets. Along these lines, philosophies and procedures for foreseeing the testing exertion is essential procedure earlier the testing procedure to altogether expand effectiveness of time, exertion and cost use.

Software reliability is a vital trait of high-confirmation and mission-basic frameworks. Such perplexing frameworks are vigorously subject to unwavering quality and security of their hidden programming applications. The difficulties engaged with accomplishing high programming quality builds the significance in creating and evaluating measures for programming quality. Early fault prediction is a demonstrated strategy in accomplishing high programming quality and can be utilized to guide financially savvy quality improvement endeavour’s to modules that are probably going to have a high number of issues. A product flaw is a that causes programming failure in an executable software. Also, Mohan et al. demonstrate that 64% of s are produced by the prerequisite’s examination and configuration stages, while just 36 % of the are presented in other advancement stage [21], as appeared in Figure 1.1.

Fig. 1.1: Origin of software faults

A product fault is a blunder, disappointment, or blame in a PC program or framework that produces off base or unforeseen outcomes or makes it carry on in unintended manner. Programming fault forecast is the way toward finding faultier modules in programming. It improves programming standard and testing capability using structure insightful models from code ascribes engage an advantageous distinguishing proof of blame inclined modules, it additionally underpins us in coordinating, checking and control and anticipate fault thickness and to comprehend and control the product quality.

A product fault is a that causes programming failure in an executable product. In programming designing, the nonconformance of programming to its necessities is regularly called a bug. Programming Engineers perceive programming defects and programming bugs. On the off chance that there ought to emerge an event of a fault, the software does not know what the customer expects yet rather of course blame is a covered programming blunder that could truly appear as a fault and the non-conformance of programming to its prerequisites is normally called a bug.

To predict faults diverse measurement measures are accessible. Measurements accessible amid static code advancement, for example, Halstead multifaceted nature, McCabe unpredictability, can be utilized to check modules for fault inclination. Fenton offers a model where a similar program usefullness is accomplished utilizing diverse programming language builds bringing about various static measures for that module. Fenton utilizes this guide to contend the futility of static code properties. In this manner, utilizing static highlights alone can be uninformative.

A module right now a work in progress is fault inclined if a module with the comparable product or procedure measurements in a prior task created in a similar domain was fault inclined. In this manner, the data accessible ahead of schedule inside the present task or from the past undertaking can be utilized in making expectations. Even though necessities measurements alone are not best at finding the deficiencies, yet these measurements can build the execution forecast.
It is necessitated that fault inclined prediction models ought to be effective and precise. Hence, blends of static highlights extracted from prerequisites and code can be outstandingly great indicators for recognizing modules that contains issues.

To predict the fault in programming information an assortment of procedures has been proposed which incorporates measurable strategy, machine learning methods, neural system methods and clustering strategies. A significant part of the work has done on checking the quality with factual strategies and machine learning techniques utilizing either prerequisite or static code measurements. Be that as it may, not very many have taken a shot at finding the flawed and non-defective modules by utilizing clustering methods. Be that as it may, evaluating the quality utilizing clustering procedures on combination of necessity and code metric has not been proposed in the writing. Expectation of fault inclined modules gives one approach to help programming quality building through improved scheduling and project control. The rest of this paper is sorted out as pursue. Area 2 displays an exchange of the related work in SFP. A discussion of the chose ML calculations is introduced in Section 3.

II. RELATED WORK
Kalaivani et al. has reviewed that software defect management is used to enhance the quality of software by recognizing and fixing the faults in the early period of SDLC. It additionally improves the quality of software; datamining techniques have been connected to construct predictions regarding the failure of software components by using old information of programming segments and their faults [1].

Chalapathi et al. discussed that the software bug tracking is a fundamental methodology for bug discovery and finding substantial bugs as conceivable the same number of concerned faults. It can recognize more faults in software advancement life cycle the designers assess the outcomes amid the location procedure in software bug recognition. Bug reports, substantial bug accumulation and bug forecast are effectively included for the appraisal of different bugs [2].

Logan Perreault et al. applied classification algorithm, for example, naïve bayes, neural systems, support vector machine, linear regression, K-closest neighbor to recognize and anticipate faults. The creators utilized NASA and tera PROMISE datasets. At last, the writers presume that all datasets are comparative, and they are written in C or C++ and in future the work can be reached out by choosing the datasets that are written in Java and as opposed to utilizing weka instrument for execution some other device can likewise be utilized [3].

Eubbeogu et al. utilized indicator factors like density, velocity and presentation time which are gotten from acceleration and used to anticipate the all-out number of faults in a product. MAChine – Learning – Inspired (MACLI) approach is utilized for foreseeing faults. The proposed system for fault expectation has two stages that is data pre-handling stage and data investigation stage [4].

Yongli et al. applied data channels to datasets so as to expand the execution of CPDP. In this examination the creators proposed Hierarchical Select-Based Filter (HSBF) methodology. HSBF depends on progressive information determination from programming project level to programming module level. In this investigation, PROMISE datasets and Confusion framework are utilized to assess the execution measure. Consequently, the writers close from the investigations, Naïve Bayes [NB] calculation performs superior to Support Vector Machine [5].

Xiao Yu et al. build a prediction model for Cross Company Defect Prediction [CCDP] by applying six unevenness learning strategies, for example, under examining procedures (random under sampling and near miss), over sampling systems [SMOTE and ADASYN] and oversampling followed by under inspecting [SMOTE Limks, TOMEK, SMOTE ENN] [6]. PROMISE datasets and classification techniques, for example, NB, Random Forest [RF] and Linear Regression [LR] are connected. Likelihood of identification, likelihood of false alert and g-measure are utilized to quantify the execution. The writers infer that NB performs better in foreseeing imperfections and it has a high pf value. Under inspecting strategy works better with g-measure.

III. USED MACHINE LEARNING ALGORITHM
The investigation expects to examine and evaluate two regulated Machine Learning calculations, which are Random Forest (RB) and Naïve Bayes (NB). The investigation demonstrates the execution precision and capacity of the algorithms for programming fault expectation and gives a similar examination of the chose algorithm calculations. The oversaw AI estimations endeavor to develop a concluding limit by shutting associations and conditions between the realized data sources and yields of the checked getting ready data, with the ultimate objective that we can foresee the yield regards for new information data reliant on the decided interpreting limit. Following are abridged portrayal of the chose administered ML calculations:

A. Random Forest
Random forest is another machine learning type that comprise of numerous order trees. The characterization trees are choice trees that speaks to significant development in information disclosure and information mining. Random forest classifier offers expectation with abnormal state exactness. The grouping calculation arranges another item from an info vector. The info vector is put down to each tree in the woods, where each tree gives an arrangement result. The tree votes in favor of that class, at that point the forest will pick most votes of order.

B. Naïve Bayes
Naïve Bayesian is an arrangement strategy that very effective and simple to execute. Most utilized and surely understood classifier technique that need littler information amount to evaluate the parameters. The essential calculation is a probabilistic as per hypothesis of Bayes, that accept class include presence does not rely upon another element presence.
IV. DATASET AND EVALUATION METHODOLOGY

There are various open source datasets available online. The datasets were obtained from NASA Promise dataset repository. Three datasets namely CM1, PC1, KC1.

The datasets were pre-handled by a classification strategy. The classification method denotes the information with class marks. To assess the execution of utilizing ML calculations in software fault expectation, we utilized a lot of surely understood capacity dependent on the produced confusion frameworks. The accompanying subareas portray the disarray lattice and the utilized assessment capacity.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Language Used</th>
<th>No of Attribute</th>
<th>No of Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM1</td>
<td>C</td>
<td>22</td>
<td>345</td>
</tr>
<tr>
<td>JM1</td>
<td>C</td>
<td>22</td>
<td>641</td>
</tr>
<tr>
<td>PC1</td>
<td>NA</td>
<td>22</td>
<td>745</td>
</tr>
</tbody>
</table>

Table 1: Characteristics of Dataset Used

A. Confusion Matrix

The confusion matrix is a table that is utilized to gauge the execution of ML calculations. Table II demonstrates a case of a conventional confusion grid. Each line of the grid speaks to the occasions in a real class, while every segment speaks to the case in an anticipated class or the other way around. Confusion matrix condenses the aftereffects of the testing calculation and gives a report of the quantity of True Positive (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN).

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class X</td>
</tr>
<tr>
<td>Class A</td>
<td>True Positive</td>
</tr>
<tr>
<td>Class B</td>
<td>False Negative</td>
</tr>
</tbody>
</table>

Table 2: The Confusion Matrix

B. Accuracy

Accuracy is one measurement for assessing characterization models. Casually, accuracy is the portion of expectations a model got correct.

Accuracy = Number of correct predictions/ Total number of predictions

C. Precision

Precision is determined as the quantity of right positive forecasts partitioned by the all-out number of positive expectations. The best exactness is 1, while the most noticeably bad is 0 and it tends to be determined as:

Precision = True Positive / (True Positive + False Positive)

D. Recall

Recall is determined as the quantity of positive expectations isolated by the complete number of positives. The best review is 1, though the most noticeably bad is 0. For the most part, Recall is determined by the accompanying formula:

Recall = True Positive / (True Positive + False Negative)

V. EXPERIMENTAL RESULTS

This experiment used Jupyter notebook 3, a python open source IDE, to calculate two machine learning algorithms (NB and RB) in software fault prediction. The accuracy of NE and RF classifiers for three datasets are shown in Table III.

As shown in Table III, the two algorithms attained a higher rate of efficiency rate. The normal incentive for the accuracy rate in all datasets for the three classifiers is over 93% all things considered. Be that as it may, the most reduced esteem shows up for NB calculation in the CM1 dataset. We trust this is on the grounds that the dataset is little and NB calculation needs a greater dataset to accomplish a higher exactness esteem. Hence, NB got a higher exactness rate in JM1 and PC1 datasets, which they are moderately greater than the CM1 dataset.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Naive Bayes</th>
<th>Random Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM1</td>
<td>0.888</td>
<td>0.928</td>
</tr>
<tr>
<td>JM1</td>
<td>0.960</td>
<td>0.944</td>
</tr>
<tr>
<td>PC1</td>
<td>0.954</td>
<td>0.953</td>
</tr>
<tr>
<td>Average</td>
<td>0.934</td>
<td>0.942</td>
</tr>
</tbody>
</table>

Table 3: Accuracy rate of Two Algorithm over Datasets

The precision calculation for implementing Naïve Bayes and Random Forest classifiers on CM1, JM1 and PC1 datasets are shown in Table IV. Results displays that two algorithms might be used for fault prediction with a better precision measure. The average precision rate for all classifiers in the three datasets are more than 96%.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Naive Bayes</th>
<th>Random Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM1</td>
<td>0.946</td>
<td>1</td>
</tr>
<tr>
<td>JM1</td>
<td>0.979</td>
<td>0.980</td>
</tr>
<tr>
<td>PC1</td>
<td>0.980</td>
<td>1</td>
</tr>
<tr>
<td>Average</td>
<td>0.968</td>
<td>0.994</td>
</tr>
</tbody>
</table>

Table 4: Precision rate of Two Algorithm over Datasets

The third evaluation measure is the recall measure. Table V displays the recall rates for the two classifiers on the three datasets. Also, herein the ML algorithms produced a better recall rate. The best recall rate was produced by RF classifier, which is 100% in all datasets. On the other hand, the average recall rate for NB algorithm is 96%.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Naive Bayes</th>
<th>Random Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>JM1</td>
<td>0.904</td>
<td>1</td>
</tr>
<tr>
<td>PC1</td>
<td>0.962</td>
<td>1</td>
</tr>
<tr>
<td>Average</td>
<td>0.959</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5: Recall rate of Two Algorithm over Datasets

VI. CONCLUSION AND FUTURE WORK

Software Fault forecast is a strategy in which an expectation display is made to foresee the future programming fault dependent on chronicled information. Different methodologies have been proposed utilizing distinctive datasets, diverse measurements and diverse execution measures. This paper assessed the utilizing of AI calculations in programming fault expectation issue. Two AI methods have been utilized, which are NB and RF.

The assessment procedure is actualized utilizing three genuine testing/investigating datasets. Exploratory outcomes are gathered dependent on exactness, accuracy and recall measures. Results uncover that the ML procedures are productive ways to deal with anticipate the future programming faults. The examination results demonstrated that the RF classifier has the best outcomes over the others. In addition, exploratory outcomes demonstrated that utilizing ML approach gives a superior exhibition to the expectation.
show than different methodologies, for example, direct AR and POWM display.

As a future work, we may include other ML procedures and give a broad correlation among them. Moreover, including more programming measurements in the learning procedure is one conceivable way to deal with increment the exactness of the forecast model.

REFERENCES


