Software Bug Prediction using Machine Learning

M.Chandrika, Hemanth Sai, K.Krishnaveni, K.Sai Harshith

Under the guidance of Mrs. **Y. Swathi**, Associate Professor,

Department of ComputerScience and Engineering, Tirumala Engineering College, Narasaraopet, Andhra Pradesh

ABSTRACT

Software Bug Prediction (SBP) is an important issue in software development and maintenance processes, which concerns with the overall of software successes. This is because predicting the software faults in earlier phase improves the software quality, reliability, efficiency and reduces the software cost. However, developing robust bug prediction model is a challenging task and many techniques have been proposed in the literature. Supervised ML algorithms have been used to predict future software faults based on historical data.

Keywords—Software bug prediction; Linear Regression, Logistic Regression, Support Vector Classifier and Random Forests.

INTRODUCTION

The existence of software bugs affects dramatically on software reliability, quality and maintenance cost. Achieving bug-free software also is hard work, even the software applied carefully because most time there is hidden bugs. In addition to, developing software bug prediction model which couldpredict the faulty modules in the early phase is a real challengein software engineering.

Software bug prediction is an essential activity in software development. This is because predicting the buggy modules prior to software deployment achieves the user satisfaction, improves the overall software performance. Moreover, predicting the software bug early improves software adaptationto different environments and increases the resource utilization.

Various techniques have been proposed to tackle Software Bug Prediction (SBP) problem. The most known techniques are Machine Learning (ML) techniques. The ML techniques are used extensively in SBP to predict the buggy modules based on historical fault data.

With the prediction model, software engineers caneffectively allocate the available testing resources on the defective instances for improving software quality in the early phases of development life cycle.software can be deployed in the given time, resources and budget.

I. LITERATURE SURVEY

There are many studies about software bug prediction usingmachine learning techniques. For example, the study in [2] proposed a linear Auto-Regression (AR) approach to predict the faulty modules. The study predicts the software future faults depending on the historical data of the software accumulated faults. The study also evaluated and compared the AR model and with the Known power model (POWM) used Root Mean Square Error (RMSE) measure. In addition to, the study used three datasets for evaluation and the results were promising.

The studies in [3], [4] analyzed the applicability of various ML methods for fault prediction. Sharma and Chandra [3] added to their study the most important previous researches about each ML techniques and the current trends in software bug prediction using machine learning. This study can be used as ground or step to prepare for future work in software bug prediction.

R. Malhotra in [5] presented a good systematic review for software bug prediction techniques, which using Machine Learning (ML). The paper included a review of all the studies between the period of 1991 and 2013, analyzed the ML techniques for software bug prediction models, and assessed their performance, compared between ML and statistic techniques, compared between different ML techniques and summarized the strength and the weakness of the ML techniques

In [6], the paper provided a benchmark to allow for common and useful comparison between different bug prediction approaches. The study presented a comprehensive comparison between a well-known bug prediction approaches, also introduced new approach and evaluated its performance by building a good comparison with other approaches using thepresented benchmark.

D. L. Gupta and K. Saxena [7] developed a model for object-oriented Software Bug Prediction System (SBPS). The study combined similar types of defect datasets which are available at Promise Software Engineering Repository.The study evaluated the proposed model by using the performance.

Rosli et al. [8] presented an application using the genetic algorithm for fault proneness prediction. The application obtains its values, such as the object-oriented metrics and countmetrics values from an open source software project. The genetic algorithm uses the application's values as inputs togenerate rules which employed to categorize the software modules to defective and non-defective modules. Finally, visualize the outputs using genetic algorithm applet.

The study in [9] assessed various object-oriented metrics byused machine learning techniques (decision tree and neural networks) and statistical techniques (logical and linear regression). The results of the study showed that the Coupling Between Object (CBO) metric is the best metric to predict the bugs in the class and the Line Of Code (LOC) is fairly well, but the Depth of Inheritance Tree (DIT) and Number Of Children (NOC) are untrusted metrics.

Singh and Chug [10] discussed five popular ML algorithms used for software defect prediction i.e. Artificial Neural Networks (ANNs), Particle Swarm Optimization (PSO), Decision Tree (DT), Naïve Bayes (NB) and Linear Classifiers (LC). The study presented important results including that the ANN has lowest error rate followed by DT, but the linear classifier is better than other algorithms in term of defect prediction accuracy, the most popular methods used in software defect prediction are: DT, BL, ANN, SVM, RBL and EA, and the common metrics used in software defect prediction studies are: Line Of Code (LOC) metrics, object oriented metrics such as cohesion, coupling and inheritance, also other metrics called hybrid metrics which used both object oriented and procedural metrics, furthermore the results showed that most software defect prediction studied used NASA dataset and PROMISE dataset.

II. PROPOSED SYSTEM

Four supervised ML algorithms have been used to predict future software faults based on historical data. These classifiersare Linear Regression, Logistic Regression, Support VectorClassifier and Random Forests. Improve software quality and productivity. SDP can efficiently progress the effectiveness of software testing and direct the allocation of resources. To develop quality software, software flaws can be detected and corrected at early phase of SDLC.

Defect prediction is the method of designing models thatare utilized in the initial stages of the process to detect defective systems such as units or classes. This can be achieved by classifying the modules as defect prone or not. Different methods are used to identify the classification module,the most common of which is support vector classifier (SVC), random forest, naive bayes, decision trees (DT), neural networks (NN). The detected defect prone modules are given high priority in progress testing phases and the non-defect prone modules are examined as time and cost permits. The feature of classification, known as the relationship between the attributes and the training dataset class.

III. PROPOSED SOLUTION

We have used the following Algorithms :

1. Base Learners
2. Ensemble Methods

Base Learners

In order to set up an ensemble learning method, we first need to select our base models to beaggregated. Most of the time (including in the well known bagging and boosting methods) a single base learning algorithm is used so that we have homogeneous weak learners that are trained in different ways. The ensemble model we obtain is then said to be “homogeneous”. However, there also exist some methodsthat use different type of base learning algorithms: some heterogeneous weak learners are then combined into an “heterogeneous ensembles model”.

This brings us to the question of how to combine these models. We can mention three major kinds of meta-algorithms that aims at combining weak learners:

bagging, that often considers homogeneous weak learners, learns them independently from each other in parallel and combines them following some kind of deterministic averaging process

boosting, that often considers homogeneous weak learners, learns them sequentially in a very adaptative way (a base model depends on the previous ones) and combines them following a deterministic strategy

stacking, that often considers heterogeneous weak learners, learns them in parallel and combines them by training a meta-model to output a prediction based on the different weak models predictions.

Linear Regression

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.

Support Vector Classifer

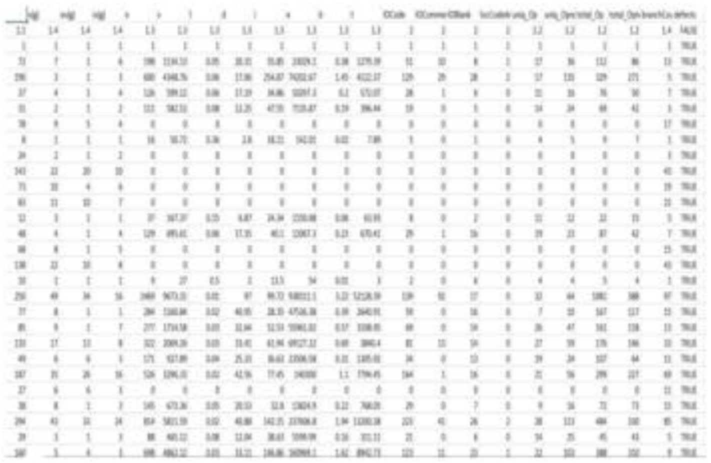
SVC, or Support Vector Classifier, is a supervised machine learning algorithm typically used for classification tasks. SVC works by mapping data points to a high- dimensional space and then finding the optimal hyperplane that divides the data into two classes. The goal of the SVM algorithm is to create the best line or decision boundary thatcan segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. If the hyperplane classifies the dataset linearly then the algorithm we call it as SVC and the algorithm that separates the dataset by non-linear approach then we call it as SVM.

Logistic Regression

Logistic regression is a statistical method used to predict the outcome of a dependent variable based on previous observations. It's a type of regression analysis and is a commonly used algorithm for solving binary classification problems.

IV. DATASET


[illegible]

V. OUTPUT

EVALUATING THE PERFORMANCE OF A MACHINE LEARNINGMODEL:

When performing classification predictions, there's four types of outcomes that could occur.

True positives are when you predict an observation belongs to a class and it actually doesbelong to that class.

True negatives are when you predict an observation does not belong to a class and itactually does not belong to that class.

False positives occur when you predict an observationbelongs to a class when in reality itdoes not.

False negatives occur when you predict an observation doesnot belong to a class when infact it does.

What we desire is **TRUE POSITIVE** and **TRUE NEGATIVE** but due to the misclassifications, wemay also end up in **FALSE POSITIVE** and **FALSE NEGATIVE.** This is because no machine learning algorithm is perfect

These four outcomes are often plotted on a confusion matrix.

A. Confusion Matrix

The confusion matrix is a specific table that is used to measure the performance of ML algorithms. Table V shows anexample of a generic confusion matrix. Each row of the matrix represents the instances in an actual class, while each column represents the instance in a predicted class or vice versa. Confusion matrix summarizes the results of the testingalgorithm and provides a report of the number of True Positive (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN).

B. Accuracy

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 byusing the following formula

ACC = (TP + TN) / (TP + TN+ FP + FN)

C. precision

Precision is calculated as the number of correct positive predictions divided by the total number of positive predictions.The best precision is 1, whereas the worst is 0 and it can be calculated as:

Precision = TP / (TP + FP)

D. Recall

Recall is calculated as the number of positive predictions divided by the total number of positives. The best recall iswhereas the worst is 0. Generally, Recall is calculated bythe following formula:

Recall = TP / (TP + FN)

E. F-measure

F-measure is defined as the weighted harmonic mean of precision and recall. Usually, it is used to combine the Recall and Precision measures in one measure in order to compare different ML algorithms with each other. F-measure formula is given by:

F- measure= (2* Recall * Precision)/(Recall + Precision)

F. Accuracy

Accuracy (ACC) is the proportion of true results (both TPand TN) among the total number of examined instances.The best accuracy is 1, whereas the worst accuracy is 0.ACC can be computedby using the following formula: Accuracy = correct predictions / all predictions ACC = (TP + TN) / (TP + TN+ FP + FN)

MEAN SQUARED ERROR

Mean squared error is simply defined as the average of squared differences between the predicted output and the true output. Squared error is commonly used because it is agnostic to whether the prediction was too high or too low, it just reports that the prediction was incorrect.

OUTPUT

Defects

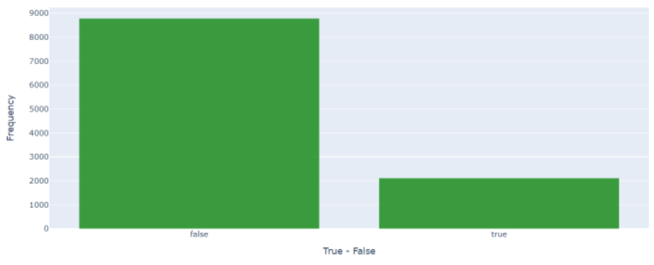


Fig: Defects

Volume – Bug

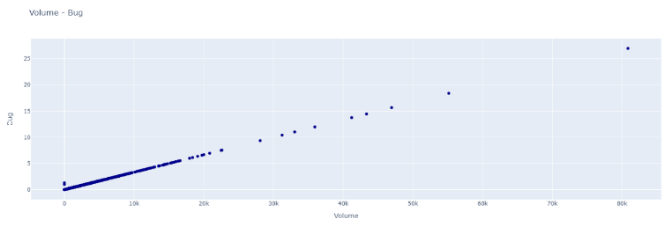


Fig: Volume – Bug

Complexity Evaluation

As a future work, we may involve other ML 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.

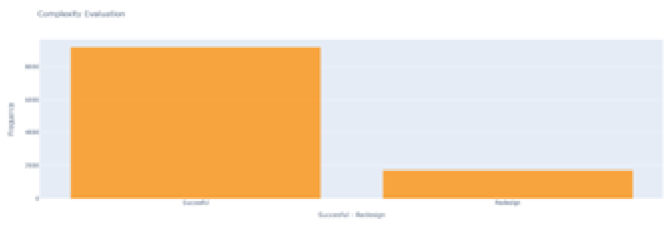


Fig: Complexity Evaluation

Line of code – Bug

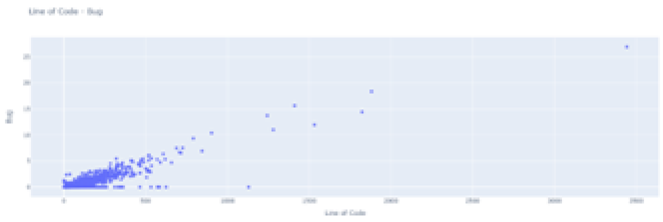
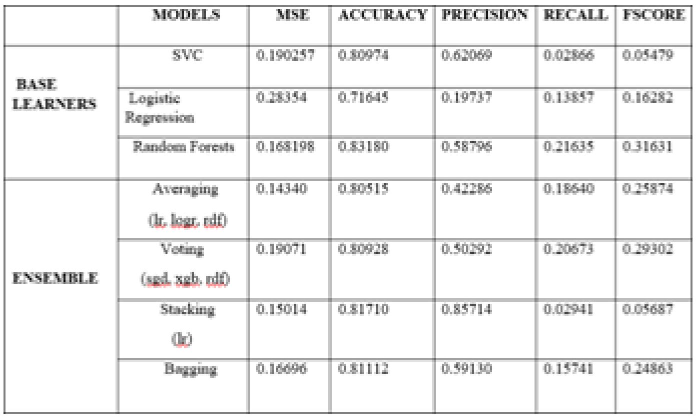


Fig: Line of code - Bug



	MODELS	MSE	ACCURACY	PRECISION	RECALL	FSCORE
BASE LEARNERS	SVC	0.190257	0.80974	0.62069	0.02866	0.05479
	Logistic Regression	0.28354	0.71645	0.19737	0.13857	0.16282
	Random Forests	0.168198	0.83180	0.58796	0.21635	0.31631
ENSEMBLE	Averaging (lr, logr, rdf)	0.14340	0.80515	0.42286	0.18640	0.25874
	Voting (sgd, xgb, rdf)	0.19071	0.80928	0.50292	0.20673	0.29302
	Stacking (lr)	0.15014	0.81710	0.85714	0.02941	0.05687
	Bagging	0.16696	0.81112	0.59130	0.15741	0.24863

VI. CONCLUSIONS AND FUTURE WORK

Software bug prediction is a technique in which aprediction model is created in order to predict the future software faults based on historical data. Various approaches have been proposed using differentdatasets, different metrics and different performance measures. This paper evaluated the using of machine learning algorithms in software bug prediction problem. Three machine learning techniques havebeen used, which are NB, DT and ANNs.

REFERENCES

[1] Y. Tohman, K. Tokunaga, S. Nagase, and M. Y., “Structural approach to the estimation of the number of residual software faults based on the hyper-geometric districution model,” IEEE Trans. on Software Engineering, pp. 345–355, 1989.

[2] A. Sheta and D. Rine, “Modeling Incremental Faults of Software Testing Process Using AR Models ”, the Proceeding of 4th International Multi-Conferences on Computer Science and Information Technology (CSIT 2006), Amman, Jordan. Vol. 3. 2006.

[3] D. Sharma and P. Chandra, "Software Fault Prediction Using Machine- Learning Techniques," Smart

Computing and Informatics. Springer, Singapore, 2018. 541-549.

[4] R. Malhotra, "Comparative analysis of statistical and machine learning methods for predicting faulty modules," Applied Soft Computing 21, (2014): 286-297

[5] Malhotra, Ruchika. "A systematic review of machine learning techniques for software fault prediction." Applied Soft Computing 27 (2015): 504-518.

[6] D'Ambros, Marco, Michele Lanza, and Romain Robbes. "An extensive comparison of bug prediction approaches." Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on. IEEE, 2010.

[7] Gupta, Dharmendra Lal, and Kavita Saxena. "Software bug prediction using object-oriented metrics." *Sādhanā* (2017): 1-15..

[8] M. M. Rosli, N. H. I. Teo, N. S. M. Yusop and N. S. Moham, "The Design of a Software Fault Prone Application Using Evolutionary Algorithm," IEEE Conference on Open Systems, 2011.

[9] T. Gyimothy, R. Ferenc and I. Siket, "Empirical Validation of Object- Oriented Metrics on Open Source Software for Fault Prediction," IEEE Transactions On Software Engineering, 2005.

[10] Singh, Praman Deep, and Anuradha Chug. "Software defect prediction analysis using machine learning algorithms." 7th International Conference on Cloud Computing, Data Science & Engineering- Confluence, IEEE, 2017.

[11] M. C. Prasad, L. Florence and A. Arya, "A Study on Software Metrics based Software Defect Prediction using Data Mining and Machine Learning Techniques," International Journal of Database Theory and Application, pp. 179-190, 2015.