A Study on Applying Machine Learning approach to Forecast a Software Defect

Manjunath R.
Professor,
Department of CSE,
R. R. Institute of Technology,
Bengaluru, Karnataka, India

Arpitha M. R.
UG Student,
R. R. Institute of Technology,
Visvesvaraya Technological University,
Bangalore, India

ABSTRACT

Defects are common in software systems and can potentially cause various problems to software users. Different methods have been developed to quickly predict the most likely locations of defects in large code bases. Most of them focus on designing features (e.g. complexity metrics) that correlate with potentially defective code. Those approaches however do not sufficiently capture the syntax and different levels of semantics of source code, an important capability for building accurate prediction models.In our approach, three supervised machine learning algorithms are considered to build the model and predict the occurrence of the software bugs based on historical data by deploying the classifiers Logistic regression, Naïve Bayes, and Decision Tree. Historical data has been used to predict the future software faults by deploying the classifier algorithms and make the models a better choice for predictions using random forest ensemble classifiers and validating the models with K-Fold cross validation technique which results in the model effectively working for all the scenarios.

Keywords: software metrics, bug, classifier, cross validation, machine learning.

INTRODUCTION:

Machine Learning algorithms sprawl their application in various fields relentlessly Software Engineering is not exempted from that. Software bug prediction at the initial stages of software development improves the important aspects such as software quality, reliability, and efficiency and minimizes the development cost. In majority of software projects which are becoming increasingly large and complex programs, bugs are serious challenge for system consistency and efficiency. In our approach, three supervised machine learning algorithms are considered to build the model and predict the occurrence of the software bugs based on historical data by deploying the classifiers Logistic regression, Naïve Bayes, and Decision Tree. Historical data has been used to predict the future software faults by deploying the classifier algorithms and make the models a better choice for predictions using random forest ensemble classifiers and validating the models with K-Fold cross validation technique which results in the model effectively working for all the scenarios. With the growing complexities of the software, the number of potential bugs is also increasing rapidly. These bugs hinder the rapid software development cycle Bugs, if left unresolved, might cause problems in the long run. Also, without any prior knowledge about the location and the number of bugs, managers may not be able to allocate resources in an efficient way. In order to overcome this problem, researchers have devised numerous bug prediction approaches so far.

RELATED WORK:

The following is the similar work identified in the field of bug prediction using machine learning algorithms. In a work done by G. Eason et. al [1] it has been found that, focused developers makes less error in software components. Less focused developers always make more errors according to the industry standards. They have constructed bug predictionmodel using three techniques for data set obtained from 26 projects. The important measure considered in their work is scattering changes performed by developers. The obtained results are more promising and high complement with respect to predictors commonly used in the literature. The results of a “hybrid” prediction model combining predictors with the existing ones are also shown. But the disadvantages we found that even though the Models have been evaluated across the models, the cross validation on various data sources was not done to understand the accuracy of the machine learning model, the number of factors selected are not exhaustive and to get right accuracy right number offactors needs to be selected. J. Clerk Maxwell [2] used a Semi-supervised structured dictionary learning (SSDL) approach. This approach is really giving very good results if there is lack of historical data for building accurate model. Semi-supervised defect prediction (SSDP) and Cross-project defect prediction (CPDP) are the possible effectivesolutions arrived by them. They proposed a semi-supervised structured dictionary learning (SSDL) approach for CSDP and WSDP. They used limited labeled defect data and huge unlabelled data. In their work it is concluded that SSDL performs better than related SSDP methods. This happens for two data sets in the CSDP scenario. The issue with this approach is cross project supervised learning algorithms generalizes the predictions. The choice of variables have to be satisfactory across the projects. Hence, it leads to get inaccurate predictions within the project level. Software testing is the process that verifies and validates quality of the software delivery. In SDLC this is an essential phase of software development life cycle. In the work [3], a set of various software metrics are used for predicting software fault and identified that main aspect of the classification techniques are base line models. When base line models are used, these are weak learners and will not have the right amount of accuracy required. One way to monitor software quality is to find software faults or defects and then correct those faults. Even though software metrics usage is meant for measuringsoftware quality, it can be used to identify the faulty modules in software. With this model construction we can analyze and able to predict the fault prone modules. The analysts work on this prediction models for long time, now it has become hot issue and given more significance. In Paper[4], Genetic algorithm based object oriented Unified Modeling Language (UML) approach is applied and they worked the detailed design of software fault proneness. SoftwareMetrics Information Extractor (SMIE), Fault Classes Detection System (FCDS) and Genetic Algorithm Generator (GAG) are used in their approach. Blank lines(bl), classes(c), code lines(cl), comment lines(cml), executable statements(es), files(f), functions(fn) and lines(ln) [5] are used as software metrics here. In software Engineering defect or bug prediction captured interest among analyst and developers over a period of time. The driving scenario is resource allocation. In order to develop quality software, more time and resources need to be allotted for the software system design with a higher probable quantity of bugs. A standard defect prediction model is presented in [6], which deals with a publicly available data set consisting of several software systems. Comparative study on well known techniques are explained in their approach. Though the good approach is devised, the number of factors considered for the analysis are very less. The simulated system is not representing the real world systems. Accuracy score was calculated on single data set. Comparison across multiple datasets could have been a better choice. In the approach [7], they tried to diminish the gap between two technical domain of knowledge such as software engineering and data mining. Maximize the reach through marketing and communication. Using integrated approaches bug predictions are advocated, instead of single classifier/ clustering. It has been shown that soft computing techniques such as genetic algorithm, fuzzy c-means clustering and random forest classifier produces better experimental results. The problem here is majority of the classification techniques are base line models. When base line models are used, these are week learners and will not have the right amount of accuracy required. In the work done by Gyimoet. al [8], they used the Object-oriented metrics given by Chidamber and Kemerer [11]. These metrics are calculated for the open source software Web and e-mail suite Mozilla. The results are compared with bug data base Bugzilla which uses regressionand learning techniques for validation. The downside of this approach is regression models required to follow linear relationship between independent variables. In real time data it`s unlikely to have linear relations. From the literature survey done, we found that it has been recommended to use more than one techniques combined to get better results.

MACHINE LEARNING ALGORITHMS:

This section deals with the concept of the algorithms used in our approach.
Logistic Regression: Logistic regression method solves classification problems. It is meant for predicting the likelihood of an entity belonging one class or another class. There are many such examples like marketing effort to know about whether customers purchase or non-purchase, creditworthiness for paying EMI is high or low, in insurance whether there is high or low risk of accident claim. The logistic regression follows a s-shaped curve to the predict the feature of the a binary response variables. In this approach complex optimized equation is obtained

by converting from the logistic equation to the OLS-type equation. The Eq. (1) which is derived from probabilistic approach is given below. P is the probability for Y=1 and 1-P is the probability of getting Y=0;

$$\ln (P/1-P) = a+bX \qquad (1)$$

P can also be obtained from the regression equation.

The regression equation given in Eq. (2) calculates the expected probability for a given value of X for Y=1.

$$P = \exp(a+bX) / 1+\exp(a+bX) = e^{(a+bX)} / 1 + e^{(a+bX)} \qquad (2)$$

In our model fitting we used Chi-square instead of R2 as the statistic. Chi-square is a measure of the observed and the expected values. It is the fit of the observed values (Y) to the expected values (Y'). If the deviance of the observed values is more from the expected values, the fitness of the model questionable. The optimization criteria is having chi square as small as possible. More variables added means the deviance will be smaller in turn there is an improvement in fit. Objective is to find the minimum deviance between the observed and predicted values to obtain the best fitting line using calculus. With the help of machine learning algorithms, the computer derives the best fit by tryingdifferent iterations with different methodologies to find the smallest difference between the actual and predicted values.

Naïve Bayes: Bayesian Decision Theory is predecessor to the concepts like Version Spaces, Decision Tree Learning and Neural Networks. The applications includes in the field of Statistical Theory and Pattern Recognition. This algorithm helps to understand Bayesian Belief Networks and the EM Algorithm. The fundamental assumptions the naïve Bayes theorem considers is the classes are independent with each other and there will not be any correlation between the variables. Equation (3) is used in this algorithm.

$$P(c/x) = P(x/c)\,P(c) / P(x) \qquad (3)$$

P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes). P(c) is the prior probability of class.P(x|c) is the likelihood which is the probability of predictor given class. P(x) is the prior probability of predictor.

Decision tree & Random Forest: Decision tree is a one of supervised learning algorithms that is widely used in classification problems with a fixed target variable. This algorithm can be applied to both categorical and continuous input and output variables. In this technique, the given data set is divided into two or more consistent sets based onmost significant input variables which act as differentiator. Random Forest is a ensemble classifier that is also meant for classification, and regression. It constructs the number of decision trees on different sub-samples of the dataset and takes average to get the predictive accuracy and also controls over fitting of the model. For classification its output is based on mode of the classes and for regression it uses mean of the individual trees.

- Consider the classifier with N number of rows and M number of column or factors.
- The task is to split M input variables to find the node of the tree and M should be smaller than M
- Select the training dataset for each decision tree by selecting the n occurrences with replacement from allavailable training rows and use the other rows to test the algorithm accuracy.
- For each child of the tree, randomly choose m variables on which we can make the decision of the node.
- Use Information gain or Gini index prioritize the feature importance of the columns.
- The final classifier from multiple trees will be constructed using voting techniques. The highest of group oftrees of a particular class will be prioritized.

METHODOLOGY:

Starts with solving the problem with identifying the right factors required for bug prediction. The prioritized factors are considered for creating analytical data set. Base line models and benchmarking models are implemented on top of analytical dataset. All algorithms are validated through k fold validation methodology to find the right accuracy. Any supervised machine learning algorithms will require a systematic flow of the below steps. These are generalized frameworks which can help in defining the problem better and executing the all the phases listed out below in structured manner. The phases of our system is given in the Figure 1.

a. Problem Solving: Defining the problem better.
b. Data pre-processing: Creating the base dataset required for the analysis.
c. Baseline Model: Creating the baseline model for predicting the bug occurrence.
d. Bench marking Model: validating the created model with bench marking models.
e. Model Validation : Validate the model performance on confusion matrix, or ROC-AUC Curve.
f. K Fold Validation: Test the model performance across the different datasets.

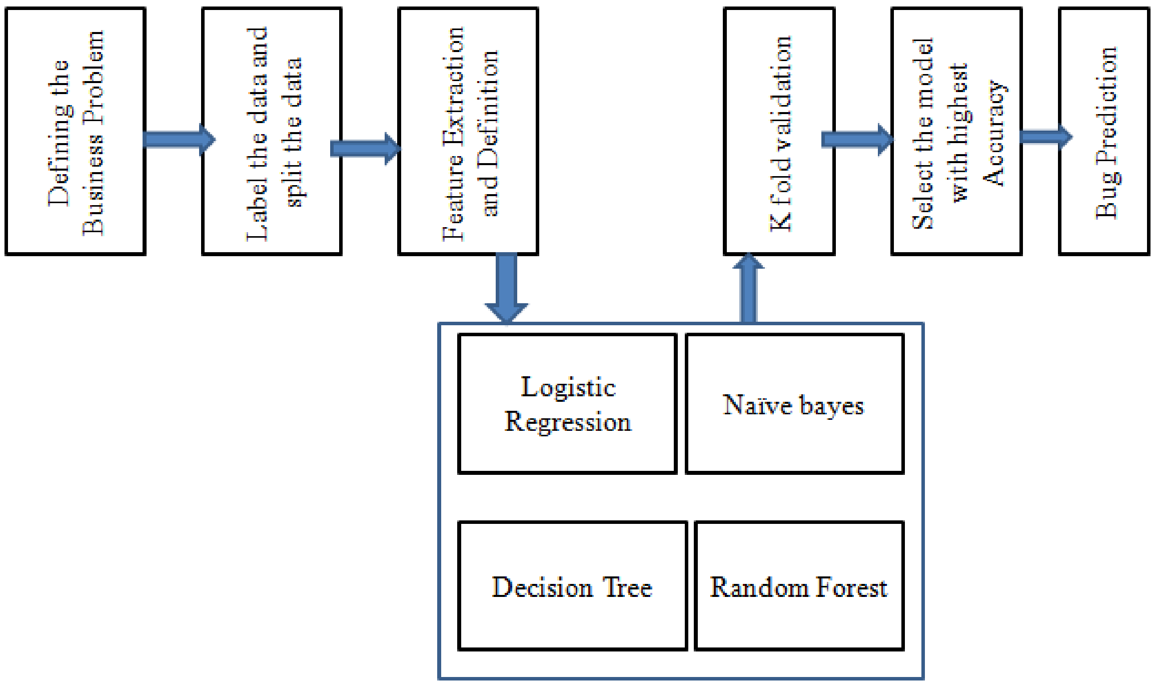
Defining the Business Problem
Label the data and split the data
Feature Extraction and Definition
K fold validation
Select the model with highest Accuracy
Bug Prediction
Logistic Regression
Naïve bayes
Decision Tree
Random Forest

Figure 1: Block Diagram of Software Bug Prediction System

Problem solving framework is one of the important phases of the any machine learning problem is how to define the problem statement clearly. The key factors required for the analysis are identified in the process. It will help in listing down the factors without bias. The factor map and Hypothesis are generated. Identify and prioritize the important factors based on actions and feasibility matrix.
Data pre-processing (exploratory data analysis) majority of the time the data collected for running machine learning algorithms is not available readymade. Hence, the data requires pre-processing which will help to get the right predictions. The fore most step in best model building is data pre-processing followed by Data Collection, Data Merging, Null Value Treatment, Outlier Treatment, Garbage Value Removal and creating the analytically ready dataset for the Effective Data Analysis. In our work, the above pre-processing steps are incorporated to create the analyticaldata set.
Baseline model logistic regression algorithm has been used as one of the base models in our work. As the bug prediction is a binary response variable, the bug behavior can be predicted by multiple algorithms. Logistic Regression is considered as base model and other algorithms will be considered as benchmarking models. Data set is divided in to train and test set. The performance of the models is validated by identify the accuracy score, confusion matrix, ROC- AUC curve designing, and Probability curve. With these measures it has been tried to improving upon the model performance and Defect Detection.
Bench Marking Model: Random Forest Classification Bagging or Boot strap aggregation is a method that minimizes the predictions variance by merging the outcome of more than one classifiers that work on various sub-samples of the same data set.
The steps followed in bagging are:
1. Create Multiple Datasets: New data sets are formed from the original data set as a part of columns and rows. Theseare called hyper parameters.
2. Build Multiple Classifiers: Multiple Classifiers are built on each data set. On different data set the same

classifier isapplied and results are obtained.

3. Combine Classifiers: Combining the predictions of all the classifiers are achieved by a mean, median or mode valuedepending on the problem statement considered. The obtained results are always more accurate than a single model.

Model validation Confusion Matrix Confusion matrix is the preferred choice for measuring the accuracy of random forest decision tree, Naive Bayes and majority of the classification algorithms. It is also known as error matrix is one ofthe important metrics to decide the accuracy of the classification algorithms. The confusion matrix for a binary classifier will be 2X2 Size array which will help in analyzing the properties of the classification algorithm like the true positive rate, true negative rate, false positive rate and false negative rate. The accuracy score of the model also will be determined by using confusion matrix. Accuracy matrix alone cannot be used in deciding the accuracy of the mode. Hence, We will be deriving multiple parameters like f1-score, precision, provenance and etc. from the confusion matrixto enhance the interpretation capabilities of the model . Type 1 and Type II errors are decided based on how represent the actual and predicted values in the matrix. If the response variable has more than 2 classes, the confusion matrix will be of the same size of the unique values in the predictor variable.

ROC – AUC Curve: A Receiver Operating Characteristic curve, i.e., ROC curve, is a visual representation that is meant for interpreting the true predictive power of the binary classifier system when its threshold is varied discriminately. The ROC curve takes true positive rate on x-axis and false positive rate on y axis at various threshold settings. True positive rate is a measure of how good the classifier was able to predict positive values out of the all actual positive values. On x axis and y axis for different intervals, the true positive rate and false positive rates are represented at different probability values and these measures are plotted. Higher the area under the curve signifies model has better accuracy.

Cross validation: It is an effective technique to interpret the model results. The datasets will be divided into multiple folds and in each iteration the algorithm will be executed to measure the accuracy. The average accuracy across the models will be computed to get the final accuracy of the models. If there is any outliers or skewness in the data , it will be course corrected by using cross validation techniques . The k-fold validation technique is efficient strategy to predictthe final accuracy of the model . The data set is created into multiple parts and the algorithm will be executed in multiple iterations and at each iteration the accuracy will be measured and finally the average of 10 accuracy scores will be treated as the final score for the models.

RESULTS AND DISCUSSION:

For our implementation, Python the multi-paradigm programming language with rich Data science packages has been selected. The data is collected from GitHub hosted projects. In this work, data model was built using Logistic regression, Decision Tree, Naïve Bayes and Random forest ensemble classifier. K-Fold cross validation is a recent and widely accepted type of cross validation technique in machine learning.
The observations made from figure is listed out below:

- The Base line model and one bench mark model produces the same accuracy. Random Forest produces thehigh accuracy rate.
- Overall feedback is that all the classifiers are producing promising results for the data set obtained.

The results of our bug prediction using four learning methods are given in Figure 2. The cell in figure gives accuracy ofthe techniques used (percentage) on test data set.

Method	Prediction	Condition Positive	Condition Negative	Accuracy
Random Forest	Positive	35037	102	97%
	Negative	1169	303	

Method	Prediction	Condition Positive	Condition Negative	Accuracy
Logistic Regression	Positive	34938	102	96%
	Negative	1469	84	
Decision Tree	Positive	31966	3074	96%
	Negative	0	1533	
Naive Bayes	Positive	31966	3074	92%
	Negative	0	1533	

Figure 2: Results of Bug Prediction

Identified some similar work and analyzing the accuracy obtained, we have found that in the work [12] using Extreme learning machines with three different types of Kernels, the accuracy achieved is around 87.5%. In the work [13] using Sigmoid model, 75% of accuracy has been achieved. Even in other models of them perform less than 85% accuracy except multiquadtric model. Our algorithms hows better results than these approaches.

CONCLUSION AND FUTURE WORK:

Bug prediction improves the software development process in terms of production cost, eases the maintenance phase and helps to increase reliability of the software. In our proposed work we try to use models for this process. Models constructed using individual classifiers are tend to have less accuracy. Models are not validated across different samples. Hence, if random samples are taken, there is high chance of getting less accuracy. Enough priorities were not given on variable selection (Prioritization of Factors), Feature engineering, variable transformations and Variable reductions which can improve the accuracy of the models to the great extent. There are multiple algorithms which can be used to predict the bug occurrence. Random forest algorithm is the preferred choice because of the ensemble nature. Instead of constructing the individual algorithms, multiple decision trees are used to predict the final outcome with the help of bagging logic, which leads to better accuracy. K fold validation is applied to eliminate the biasness in the dataset. In Future, the model can be trained in artificial neural networks to increase the accuracy of the predictions. A forecasting model can be built to predict the number of bugs which can be used as companion to get more accurate results. Our initial thought is to include Artificial Neural Network algorithm also. However, some of the algorithms were able to generate 100 % accuracy with train and test datasets. Hence, it is decided to settle down with Machine learning algorithms. If we increase the dataset size ANN will be a good option to proceed further.

REFERENCES:

Chidamber, S.R. and C.F. Kemerer, "A metrics suite for objectoriented design", in IEEE Transaction on Software Engineering., Vol. 20: pp.476-493, 1994

D'Ambros, M. Lanza, and R. Robbes, "An Extensive Comparison of Bug Prediction Approaches", In Proc. IEEE Seventh Working Conf. Mining Software Repositories, pp. 31-41, 2010

Dario Di Nucci, Fabio Palomba ,Giuseppe De Rosa Gabriele Bavota ,Rocco Oliveto, and Andrea De Lucia, “A developer centric bug prediction model", IEEE Transactions on Software Engineering, Vo.l 44, Issue 1, pp. 5-24, 2018

F. Wu et al., "Cross-Project and Within-Project Semi supervised Software Defect Prediction: A Unified approach", IEEE Transactions on Reliability, pp. 1-17, 2018

Gyimothy, T., Ferenc, R. and Siket, I., "Empirical validation of object-oriented metrics on open source software for fault prediction", IEEE Transactions on Software Engineering, 31(10), pp. 897-910, 2005.

John T. Pohlmann and Dennis w. Leitnera "Comparison of Ordinary Least Squares and Logistic Regression", The Ohio Journal of Science. vol. 103, number 5, pp. 118-125, Dec, 2003 SPSS, https://www.ibm.com/analytics/spss-statistics-software

Kumar, Lov, and AshishSureka. "Aging Related Bug Prediction using Extreme Learning Machines.", In Proc.

14thIEEE India Council International Conference (INDICON), pp.1-6, IEEE, 2017.

M. M. Rosli, N. H. I. Teo, N. S. M. Yusop, and N. S. Mohammad, "The design of a software fault prone application using evolutionary algorithm," in Proc. IEEE Conference on Open Systems (ICOS 2011). Los Alamitos, California: IEEE Computer Society, pp. 38-343. 2011

Meiliana, S. Karim, H. L. H. S. Warnars, F. L. Gaol, E. Abdurachman and B. Soewito, "Software metrics for fault prediction using machine learning approaches: A literature review with PROMISE repository dataset", In Proc. IEEE International Conference on Cybernetics and Computational Intelligence(CyberneticsCom), Phuket, pp.19- 23, 2017

Nigam, Ayan, et al. "Classifying the bugs using multi-class semi supervised support vector machine.", In Proc. International Conference, Pattern Recognition, Informatics and Medical Engineering (PRIME), pp.393-397, IEEE, 2012.

Pushphavathi T P, "An Approach for Software Defect Prediction by Combined Soft Computing", In Proc, International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS) pp.3003- 3006, 2017

S. Benlarbi, et al., "Issues in validating object-oriented metrics for early risk prediction," in International Symposium Software Reliability Eng.(ISSRE'99). , Boca Raton, Florida, 1999.
