Software Bug Prediction Using Supervised Machine Learning Algorithms

S. Delphine Immaculate
Department of Computer Science & Engineering
Mookambigai College of Engineering
Tiruchirappalli, India
delphine.rajesh@yahoo.co.in

M. Farida Begam
Department of Information Science and Engineering
CMR Institute of Technology
Bengaluru, India
farida.b@cmrit.ac.in

M. Floramary
Department of Computer Science & Engineering
Mookambigai College of Engineering
Tiruchirappalli, India
floramary363@gmail.com

Abstract—**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.**

Keywords —bug; classifier; cross validation; defect; machine learning; software metrics.

I. INTRODUCTION

A. Machine Learning

Machine learning provides ability predict the occurrence of an event based on historical data without being explicitly programmed. Machine learning focuses on the output variable and try to look for patterns within the data and predicts the final outcome. There are two kinds of machine learning algorithms. Supervised, Unsupervised machine learning algorithms. In supervised algorithms, we will have an output variable decided upfront and we try to look for pattern similarities between the dependent and independent variables. In un supervised machine learning algorithms, one can pass the data and predict the final outcome variable. In case of un supervised machine learning algorithms, the concept of dependent variable is not there. The algorithms uses the complete data and creates the final outputs. Machine learning algorithms help to track real time behaviors that can be used as inputs for companies to reduce operational cost and gain better result. Fault prone development of software components will definitely leads to loosing of business. With the help of machine learning algorithms, uncertainty can be predicted with certain confidence up front which leads to trimming of chaos or disorganization and increases the momentum of automation in SDLC.

B. Defect Prediction Using Machine Learning Algorithms

Predicting the bug occurrence and understanding the process flow of SDLC are the key factors which leads to success criteria. Defect prediction using Machine Learning algorithm can be applied for any stage of SDLC such as , identification of current problems, plan, design, build, test, deploy and maintain and also any model type of SDLC such as Waterfall Model, Agile Model, Iterative Model, V-Shaped Model, Big Bang Model and Spiral Model. Machine learning algorithms and statistical analysis actually helps to guess that a code is buggy. Improved and better approaches in the development will increase the quality of the software.

C. Defect Life Cycle

Predicting the occurrence of bug and understanding the defect life cycle are very essential and mandatory which signifies the importance of predicting the bug earlier in the SDLC. It is helpful to avoid the time, effort and cost spent in detecting and fixing the defects. One of the key issues the modern software industry is facing is the number of bugs raised during the development cycle. This leads to an addition time to final delivery of the product and total increase in the operation cost . It is estimated to be 10% average operation cost increase across the projects because of the bugs that happens during code development and production of the software. As this is the hour of the need, we try to address the defect predictions using advanced machine learning algorithms. The model uses parameters from multiple dimensions and try to predict the occurrence of the bug. The model can be used as input for the occurrence of the bug before a tester identifies. Hence, the developer can retrospect the code and fix the issues with in short duration of the time. The model is well suited for most famous software development frameworks like waterfall, agile, V-shape, spiral and etc. By keeping the different kinds of software development modeling frameworks, we have constructed this final machine learning models. This acts as a universal model for all kind of defect prediction algorithms. Understanding the defect management life cycle is one of the key aspects. Defect life cycle or Bug Life cycle shown in Fig. 1 gives the insight of the states of the defect it pass through in its valid tenure.

New: If the defect is raised by the testers or anybody as a part of the software development life cycle for the first time, we call this as a new defect.
Assigned: Every new bug raised gets assigned to one of the team members for immediate resolution based on the seniority and priority of the bugs then, the corresponding developer will proceed with the action.
Open: if it is really bug after preliminary state of the testing by the developer, then it will be pushed to open state. This can be generally carried out by a developer team or product research team.
Fixed: All the bugs that are open state needs to be fixed based on severity and priority. It is all the stakeholders together decides the priority and severity of the bugs within the software development life cycle.
Retest: If the bug is not tested or the description is not given properly, this creates the situation of more round of testing to make sure the raised bug is valid for all the scenarios.
Reopen: the reopened bugs are critical, as the operation time increase with each number of the reopen bugs. reopen bugs takes more time to fix.
Rejected: Rejected bugs are not actually bugs but it was raised by the tester. These are primarily due to the skill set gap and miscommunication in the requirements
Deferred: Due to the priorities, some of the bugs will pushed to the next releases. These bugs are been classified as differed bugs.
Duplicated: If the same bug is repeated twice or more than that we call this as a repeated bug.
Closed: If the bug is fixed properly, this will be treated as fixed bugs.

D. Research Gap, Open Issues, Contribution for Justification

Existing research works on bug prediction do not focus on validating algorithm across the datasets. To build a good machine learning model with good accuracy, the data needs to be tested by using cross validation techniques (K-Fold validation-10 fold validation). We have used this technique for validating the model. In our approach, imbalanced data was converted to balanced data by using oversampling method called SMOTE (Synthetic Minority Over-sampling Technique) and obtained better accuracies.

We approached the problem of predicting the bugs or defects in software in their earlier life cycle states using of the machine learning algorithms. In our work, we tried to predict the occurrence of the software bugs based on historical data by deploying or running the classifiers such as Logistic regression[9], Naïve Bayes, and Decision and Random Forest. These Supervised machine learning algorithms are utilized and ensemble to predict the future software faults deploying the classifier algorithms and make the models a better choice for predictions. We try to validate our models with K-Fold cross validation technique which results in the effective working model for all the scenarios.

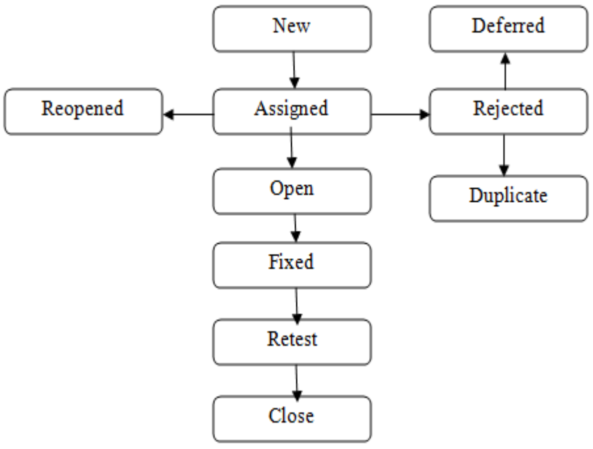

Fig 1. Defect Life Cycle

In Section 2 we present similar work, and a brief discussion about our machine learning algorithms used, and in Section 3 we deal with System Design and Methodology elaborately and in section 4, we deal with our approach of bug detection with machine learning algorithms. The results obtained are given in this section. Section 5 ends the paper with future work prospects.

II. LITERATURE SURVEY

A. 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 prediction model 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 of factors 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 effective solutions 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 measuring software 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. Software Metrics 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 Gyimo et. 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 regression and 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.

B. Machine Learning Algorithms used

This subsection deals with the concept of the algorithms used in our approach.

1. 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;

$$\ln\left(\frac{P}{1-P}\right) = a + bX \qquad (1)$$

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.

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

In our model fitting we used Chi-square instead of R^2 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 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 trying different iterations with different methodologies to find the smallest difference between the actual and predicted values.

2. 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 on most 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 all available 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 of trees of a particular class will be prioritized.

3. 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.

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)} \tag{3}$$

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.

III. SYSTEM DESIGN AND METHODOLOGY

Solving the problem starts 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 design of our system is given in the Fig 2.

1. **Problem Solving**: Defining the problem better.
2. **Data pre-processing**: Creating the base dataset required for the analysis.
3. **Baseline Model**: Creating the baseline model for predicting the bug occurrence.
4. **Bench marking Model**: validating the created model with bench marking models.
5. **Model Validation** : Validate the model performance on confusion matrix, or ROC-AUC Curve.
6. **K Fold Validation**: Test the model performance across the different datasets.

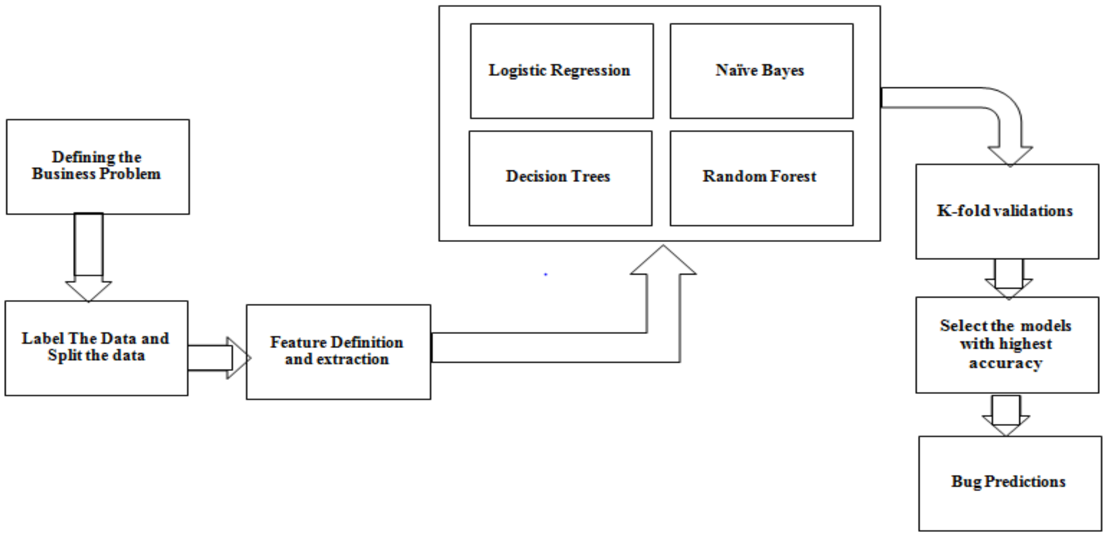
Defining the Business Problem
Label The Data and Split the data
Feature Definition and extraction
Logistic Regression
Naïve Bayes
Decision Trees
Random Forest
K-fold validations
Select the models with highest accuracy
Bug Predictions

Fig 2. System Design

A. Problem solving framework

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.

B. 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 data set for the Effective Data Analysis. In our work, the above pre-processing steps are incorporated to create the analytical data set.

C. Baseline model logistic regression

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.

D. 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. These are called hyper-parameters.
2. **Build Multiple Classifiers:**
   Multiple Classifiers are built on each data set. On different data set the same classifier is applied and results are obtained.
3. **Combine Classifiers:**
   Combining the predictions of all the classifiers are achieved by a mean, median or mode value depending on the problem statement considered. The obtained results are always more accurate than a single model.

More number of models produces consistent performance. In this scenario, based on some assumptions, it can be proved that variance of the combined predictions is reduced to 1/n (n: number of classifiers).

E. 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 of the 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 matrix to 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.

F. 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 predict the 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

IV. Experiments and Discussion

A. Data Sets

For our implementation, Python the multi-paradigm programming language with rich Data science packages has been selected. The alternate could be SPSS [10] developed by IBM. In our work, in order to construct bug data set, fifteen Java and Python projects were selected from GitHub. From the already known and fixed bugs, matching was done with the corresponding source code elements (classes and files) in order to calculate a wide set of product metrics on these elements. There are various kinds of software metrics identified such as Dimensional Metrics, Complexity Metrics, Object-Oriented Metrics and Android-oriented Metrics. Dimensional metrics provides a quantitative measures that related to the software sizes such as code size and modularity. In this category, Number of Byte-code Instructions (NBI), Number of Classes (NOC), Number of Methods (NOM), and Instructions per Method (IPM) are the metrics used for our analysis. Complexity metrics estimate the complexity of applications. Since the program comprehension is closely related to program complexity, these indicators help in understanding how expensive is the comprehension of a product when testing activities have to be done. A more complex program has more possible paths of execution, so it is difficult to test fully, understand and maintain. The mobile apps are implemented with Java, C++, and C#. These metrics suite given by Chidamber and Kemerer [11] are used to test the quality of the applications. This suite measures the code's complexity, cohesion and coupling. Cyclomatic complexity(CC), Weighted Methods per Class (WMC) in this analysis. Number of Children (NOCH), Depth of Inheritance Tree (DIT), Lack of Cohesion in Methods (LCOM), Coupling Between Objects (CBO), Percent Public Instance Variables (PPIV), Access to Public Data (APD) are list of metrics considered in Object-Oriented Metrics. Various Android errors like Bad Smell Method Calls (BSMC), WakeLocks with no timeout (WKL), Number of Location Listeners (LOCL), Number of GPS Uses (GPS), XML Parsers (XML), Network Timeouts (NTO). Data set for bug prediction has been created and four machine learning algorithms are used to obtain very high and promising bug coverage values.

B. Data Pre-processing

The data is collected from GitHub hosted projects. The preprocessing on the text data was done by using the following methods. 1. Stop words removal 2. Stemming 3. Lemmatization 4. Tokenization 5. POS (Part of Speech) tagging. After treating the data through with these methods, modeling was performed on the processed words. Apart from the key performance metrics these words are also used as input for the model. Appending all the datasets, we have created the final base dataset. The data set created has the shape of 182962 rows and 113 columns. Continuous and categorical variables are defined. Followed by this Null value treatment, Outliers treatment, and Garbage value treatment have been done to create analytical dataset. The imbalanced data set is converted to balanced data set. Univariate and Bivariate data analysis have been done to find out what variables influence the training of the model.

C. Data Modeling and Comparison

In our work, data model was built using Logistic regression, Decision Tree, Naïve Bayes and Random forest ensemble classifier.

TABLE I. RESULTS OF BUG PREDICTION

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	
Random Forest	Positive	35037	102	97%
	Negative	1169	303	

The results of our bug prediction using four learning methods are given in Table I. The cell in Table gives accuracy of the techniques used (percentage) on test data set. The observations made from Table I is listed out below:

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

D. Cross validation

K-Fold Cross Validation: This is a recent and widely accepted type of cross validation technique in machine learning. The general procedure is as follows:

- Shuffle the dataset randomly and Split the dataset into k groups.
- For each unique group:
- Consider the group as a test data set Remaining groups are considered as a training data set
- Build the model on the training set and evaluate it on the test set Analyze the keep the evaluation score and discard the model
- Summarize the fitness of the model using the sample model evaluation scores

We applied 10-Fold cross validation on the algorithms and the results obtained are consolidated in the Table II and the corresponding graphs in Fig 3 (x axis: folds, ; y axis: Accuracy) are obtained. The observations obtained form from Table II are listed out below.

- There is a proper consistency in getting accuracy of all the algorithms. Random forest produces the highest accuracy
- Our prediction model produces Less Variance and Biasness. Better choice of variables through feature engineering and prioritization matrix has been obtained.

E. Performance Comparison

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 shows better results than these approaches.

V. 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 settled down with Machine learning algorithms. If we increase the dataset size ANN will be a good option to proceed further.

REFERENCES

[1] 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

[2] 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

[3] 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

[4] 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

[5] 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.

[6] 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

[7] 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.

[8] 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.

[9] 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

[10] SPSS,https://www.ibm.com/analytics/spss-statistics-software

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

[12] Kumar, Lov, and Ashish Sureka. "Aging Related Bug Prediction using Extreme Learning Machines.", In Proc. 14th IEEE India Council International Conference (INDICON), pp.1-6, IEEE, 2017.

[13] 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.

TABLE II. RESULTS OF 10-FOLD CROSS VALIDAT ION

Method	**10 Fold Validation(Accuracy Score)**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Random Forest	0.99990892	0.99845159	0.99781381	0.9989069	0.99963563	0.99954	1.00000	0.99863	0.99909	0.99991
Naive Bayes	0.91665908	0.91556608	0.91382766	0.91373656	0.91246129	0.90999	0.92129	0.91719	0.90936	0.91273
LogisticRegression	0.95746425	0.95473176	0.95500091	0.95554746	0.95572964	0.95955	0.95500	0.95737	0.95554	0.95573

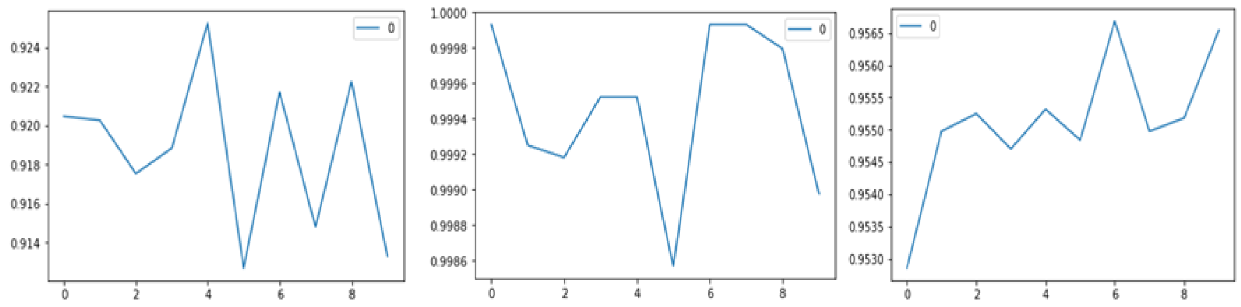

Fig 3. 10-Fold Cross Validation for Naive Bayes, Logistic Regression and Random Forest Classifier(x axis: folds, ; y axis: Accuracy)