THINK INDIA JOURNAL

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

Random Forest Based Fault Prediction For Imbalanced Data

Balraj Singh
School of Computer Science & Engineering
Lovely Professional University, Phagwara, Punjab, India
balraj.13075@lpu.co.in

Aditi Sanyal
School of Computer Science & Engineering
Lovely Professional University, Phagwara, Punjab, India
balraj.13075@lpu.co.in

Chandni
School of Computer Science & Engineering
Lovely Professional University, Phagwara, Punjab, India
chandni.24893@lpu.co.in

Abstract

Software development always faces a challenge in formulating mechanism for finding faults and developing effective methods and measures to remove these faults. Lowering the amount of defects from a software is a primary goal of software quality control. Software metrics and defect data is available to predict faults in software. This prediction helps in improving the software quality, proper resource allocation and reducing the testing effort. But sometimes software metrics and defect data is highly skewed towards the distribution of the non-faulty modules i.e. faulty modules are less in number than non-faulty modules. This is a class imbalance problem and data is imbalanced data. This study investigates the Random Forest algorithm for proposing software fault forecast model that deals with datasets that are not balanced. We used the dataset from NASA promise repository. PC5 dataset is used in our study to investigate Random forest. The result demonstrates that the accuracy of the proposed fault prediction model based on Random Forest is good and overall misclassification error is also less than the model based on Naïve Bayes i.e. the proposed model outperforms model based on Naïve Bayes.

Keywords: Random Forest, Fault Prediction, Class Imbalance Problem, Software metrics.

1. INTRODUCTION

The major challenge that software industry faces is to find the faults and to resolve them. The terms failure, fault and bug have an important liability in the quality of the software. Fault is a defect that may or may not cause failure. Because of human incorrectness such as missing instructions, incorrect syntax, syntactic etc, a bug may be introduced in the software that may result in a fault in software causing a failure. Software fails to respond as per the user needs in case of software failures. Bug is the non or poor compliance of software to its requirements. The main objective of the testing is to capture such errors and assist in eradicating them.

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

Highly reliable software is important to both developers who develop software and users who use these software, as well as society in general, since software failures may result in major business shut downs. During process of software development, to assure the quality of software, we use the white box and black box testing techniques such as Integration testing, Peer review etc. Defect prediction can be used along with these techniques to predict the occurrence of the defects. With the help of these methods, the unseen faults could be taken out and removed ,resulting a high quality software. The utility of fault prediction approach is to predict the probability of presence of faults in the software. To verify existence of faulty in the software modules, we analyzed the software measurement data and defect data. Software quality and reliability are checked by predicting software faults at a particular period of the time.

If number of faults are predicted accurately in the software, it saves testing time, testing effort, cost, and increases the reliability of a software. Good quality signifies that software have lesser number of faults and most of the critical issues are either resolved or does not exist. Software with less number of faults increases quality of the software in terms of reliability and user satisfaction. Majority of the software fault prediction models are based upon the prior experiences and historical data. These systems are trained to learn from software measurement and erroneous data to predict error prone units in system and then the system is validated. To check the quality and estimate the quality in terms of numbers of defect prone modules and non-defect prone modules, advance tools and prediction algorithms are developed. After estimating quality, resources and re structuring is assigned to the modules that are of poor quality [19].A fault predictor trained on skewed dataset will not be practical as the dataset contains large number of non-faulty modules, yields the model that is poor to predict faulty modules. This is very important issue to address when developing fault prediction models [19]. Number of techniques has been developed that deals with class imbalance problem but Random Forest is observed to be better among all these techniques. Classification and regression problems have successfully used the Random Forest algorithm for solving the problems. The capabilities of Random Forest can be further analyzed and investigated in predicting the faults in a software and evaluate the quality of the software [16]. Random forest in general demonstrate a considerable performance enhancement over the single tree classifier like CART and C4.5 which are decision tress techniques [5].

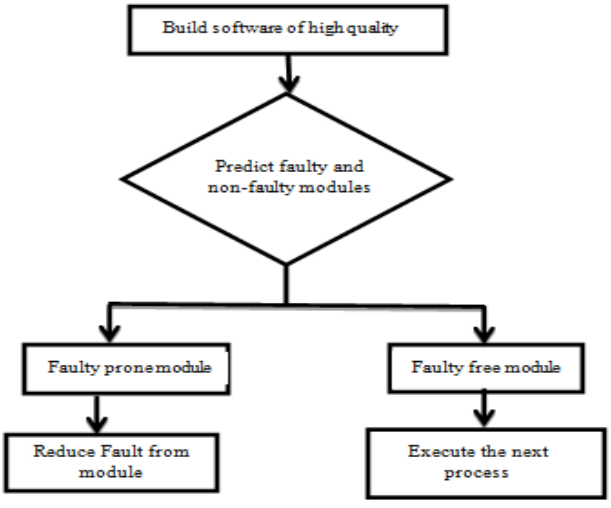

Fig1. Fault Prediction in Software

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

2. RANDOM FOREST

The main principle of ensemble methods is to group a "weak learners" to "strong learner". It can be used for classification and regression. Random forest is an ensemble classification and integration of both bagging and decision tree learning. Random forest is developed using the bootstrap samples of the data trained, utilizing random feature collection in the tree induction method. The required forecasting is made using the aggregated (collective vote for the classification or using averaging of the regression) forecast of the ensemble.
Random forests algorithm is very much similar to the bagging algorithm. Algorithm for random forest is given below:

1. Consider N be the value of the training cases and M be the value of variables in the classifier.
2. To find the decision on the node of the tree, m amount of variables should be used; value of variable m have to be smaller than M.
3. Training dataset for the given tree is chosen by selecting N times with alternate from all N present training cases. Remaining cases are utilized to estimate the error of the tree, by forecasting their classes.
4. For every node, randomly select m number of variables. Using these m variables, estimate the best split in the training data set.
5. Growth of each tree is full and not pruned.

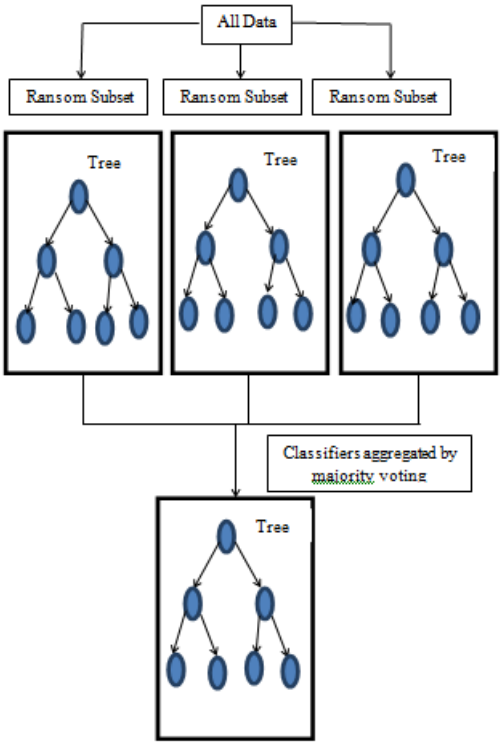

Fig 2. Random Forest

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

The advantages of Random Forest are:

1) It runs efficiently on large data sets.
2) It helps in resolving class imbalance problem.
3) It helps in classification and prediction.
4) It helps in estimating which variables are significant in the classification.
5) It can handle thousands of variables without any variable deletion.
6) It performs faster than boosting and bagging.

3. RELATED WORK

Xu-Ying Liu et al. [22] introduced two techniques for addressing class imbalance problem. The main problem that occurs in class imbalance problem is majority instances are not considered. To overcome from this problem, author had investigated two techniques i.e. Easy Ensemble and Balance Cascade. These two techniques utilizes the majority class instances but not by under-sampling. The results showed that given algorithms perform improved than the under sampling because most of the instances are used and also the performance of these two algorithms is higher than the other class imbalance learning algorithms.

Naeem Seliya et al. [19] deal with the class imbalance problems occurring when creating the fault prediction models. He had used Roughly Balanced Bagging Algorithm that integrates both data sampling and bagging in single method that aid in constructing software fault prediction model. Naïve Bayes and C4.5 decision tree (both are classification algorithms) are used by the author. In this paper, data set containing software measurement and defect data is collected from the repositories of software quality assurance systems. Each data set contains the information about number of modules having fault, modules with no faults and metrics. The result showed that system developed using Roughly Balanced Bagging performs better than the system without bagging or data sampling and Naïve Bayes performs better than C4.5 but when integrated with Roughly Balanced Bagging, the performance of the c4.5 is improved than that Naïve Bayes.

Deepak Gupta et al. [7] presented different clustering techniques and examined their performances. Clustering helps in classifying patterns into groups. He had discussed these clustering techniques clustering techniques can be used to develop software fault prediction systems in order to predict fault prone and non-fault prone modules. Early software fault prediction can also be possible with these prediction models. The results showed that author analyzed fuzzy c means is more effective to use for development of software fault prediction models. Fuzzy C-means clustering can be implemented in MATLAB.

Zhiwei Xu et al. [24] proposed the Fuzzy Non-linear Regression (FNR) technique for prediction of faults in software modules. The linear regression technique is different from FNR modeling technique because the yield of FNR model is a fuzzy number/ The usefulness of FNR model is describes with the help of case study of industrial software system. Because of limited information known in the early life cycle of

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

software development, this model performs better in this case. In this, FNR is the combination of fuzzy logic and neural networks. The results demonstrated that FNR modeling technique gives good results. FNR is very good technique for early-stage prediction.

Cagatay Catal et al [3] examined CK (chidamber-kemerer) metrics and other metrics that are module based for Artificial Immune System (AIRS) algorithm for software fault prediction problem. NASA data set is used. The main objective of author was to improve the performance of the prediction model. The object-oriented metrics and module metrics helped to have better prediction performance. The results demonstrated that integration of both CK metrics and module metrics gives good prediction output for fault prediction system to predict faults in software.

Cong Jin et al. [6] proposed the adaptive dynamic and median particle swarm optimization (ADMPSO) based on the PSO classification technique for software fault prediction. It is good technique to predict faulty modules from software. NASA data set is used. This technique helps in predicting faulty modules and shows higher performance. The accuracy of software prediction model developed using this technique is also higher.

Shuo Wang et al. [21] discussed about the quality prediction. The software metrics obtained during the software development phases may comprise of information that can help in predicting software quality. The software fault prediction classification problem is the main problem. Two challenges that arise with the modeling process:

- Increases dimensionality of software measurement data.
- Imbalanced allotment among two modules.

These problems can be solved by using techniques such as feature selection and data sampling. [25] Kumar has given the observation in his study about fault prediction that highest category of measures used for software evaluation are accuracy and precision , amounting towards 46 % of all metrics. Further , it is given in their study that supervised learning and statistical methods are highly utilized in prediction and mostly the public datasets are used in their research work. [26] Has developed a framework to conduct co clustering of multimodal data. It has a higher convergence and lower complexity. [27] Shows the importance of weighed method for classifying the potential software faults and shows that neural networks under perform as compared to the hybrid method of radial base function.

4. EMPIRICAL EVALUATION

4.1. **Goal:**The goal of this study is to evaluate the accuracy of proposed model and overall misclassification error. The accuracy is evaluated using dataset containing information about metrics and defects information.

4.2. Dataset:

The software measurement and defect data are taken from NASA Promise Repository. PC5 dataset is used regarding defect problems. It contains imbalanced data consisting of 39 attributes and 17001 modules where 16498 (97.04%) are non-faulty modules and 503(2.96%) are faulty modules. Different attributes (software metrics) present in this dataset are:

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

LOC_BLANK, BRANCH_COUNT,
CALL_PAIRS,
LOC_CODE_AND_COMMENT,
LOC_COMMENTS, CONDITION_COUNT,
CYCLOMATIC_COMPLEXITY,
CYCLOMATIC_DENSITY,
DECISION_COUNT,
DESIGN_COMPLEXITY, DESIGN_DENSITY,
EDGE_COUNT, ESSENTIAL_COMPLEXITY,
ESSENTIAL_DENSITY, LOC_EXECUTABLE,
PARAMETER_COUNT,
GLOBAL_DATA_COMPLEXITY,
HALSTEAD_CONTENT,
HALSTEAD_DIFFICULTY ,
HALSTEAD_EFFORT ,
HALSTEAD_ERROR_EST,
HALSTEAD_LENGTH, HALSTEAD_LEVEL,
HALSTEAD_PROG_TIME,
HALSTEAD_VOLUME,
MAINTENANCE_SEVERITY,
MODIFIED_CONDITION_COUNT,
MULTIPLE_CONDITION_COUNT,
NODE_COUNT,
NORMALIZED_CYLOMATIC_COMPLEXIT
Y, NUM_OPERANDS, NUM_OPERATORS,
NUM_UNIQUE_OPERANDS,
NUM_UNIQUE_OPERATORS,
NUMBER_OF_LINES,
PERCENT_COMMENTS, LOC_TOTAL,
c {FALSE,TRUE}.

4.3 Independent and Dependent Variables:

Independent variables are the attributes (software metrics) of dataset and dependent variable is Random forest.

5. RESEARCH METHODOLOGY

Random Forest is used for training the model. We implemented this technique on Weka and compared the results with simple classification algorithm, i.e. Naïve Bayes.

5.1. Performance Evaluation:

The random classifier is evaluated using confusion metrics. Confusion metrics after running Random Forest on classifier is given by:

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

Actual Class		Predicted Class	
		NFP	FP*
	NFP	16493 (TN)	5 (FP)
	FP*	14 (FN)	489(TP)

Table.1 Confusion Matrix

Where NFP is non-faulty modules

FP* is f ' Modules

Here,

1. TN is true negative i.e the number of modules with no fault that are correctly classified as no faulty modules. Here 16943 are correctly classified as non-faulty.
2. FN is false negative i.e the set of modules having fault that are incorrectly identified as non-faulty module. Here 42 faulty modules are incorrectly classified as non faulty.
3. TP is true positive i.e the set of the modules having faulty those are correctly identified as faulty module. Here 461 faulty modules are correctly classified as faulty.
4. FP false positive i.e the set of modules with no fault that are incorrectly classified as faulty modules. Here 5 non-faulty modules that are incorrectly classified as non-faulty.

5.1.1. Accuracy:

The accuracy denoted by AC, measures the chances that the fault proneness of module is correctly forecasted. Its equation is as follows:

$$AC = \frac{TN + TP}{TN + FN + TP + FP}$$

$$AC = \frac{16943 + 489}{16943 + 5 + 14 + 489} = 99.8\%$$

Precision (*P*) evaluates the chance of correctly predicting the faulty modules among the modules classified as fault prone. Its equation is as follows:

$$P = \frac{TP}{TP + FP}$$

$$P = \frac{489}{489 + 5} = 98.9\%$$

The recall or true positive rate (*TPR*) is the proportion that the modules with fault are correctly identified. Its equation is as follows:

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

$$TPR = {}^{TP}/_{(FN+TP)}$$ $$TPR=\frac{489}{14+489} = 97.2\%$$

5.1.2. False Positive Rate (FPR):

The false positive rate denoted by *FPR*, is the proportion of the modules with no faulty but incorrectly identified as faulty. Its equation is as follows:

$$FPR = \frac{FP}{TN+FP}$$ $$FPR= \frac{5}{5+16943} = 0.03\%$$

True Negative Rate (TNR):

The true negative rate (*TNR*) is given as the amount of modules with no faults that are correctly identified as non-faulty. Its equation is as follows:

$$TNR = \frac{TN}{TN+FP}$$ $$TNR= \frac{16943}{16943+5}= 98.4\%$$

False Negative Rate (FNR):

The false negative rate denoted by *FNR* is the amount of the faulty modules that are incorrectly identified as non faulty. Its equation is as follows:

$$FNR = \frac{FN}{FN+TP}$$ $$FNR=\frac{14}{14+489}=2.78\%$$

5.1.4. Overall Misclassification Rate (OMR):

$$OMR = \frac{FN+FP}{TP+FP+FN+TN}$$

$$OMR=\frac{14+5}{16943+14+5+489} =0.11\%$$

We have compared the result of proposed model with other model based on Naïve Bayes. The result shows that proposed model has high accuracy than the model used for comparison. The comparison results are given below:

Model	Naïve Bayes (%)	Random Forest (%)
Accuracy	96.88	99.80
Recall	46.70	97.20
TNR	97.90	98.40
FNR	53.28	2.78
FPR	2.09	0.03
OMR	3.61	0.11
P	40.44	98.90

Table.2 Comparison of Results

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

The results of model based on Naïve Bayes are shown in below graph. The accuracy is 96.88%. FPR and FNR is 2.09 % and 53.28% respectively. Its OMR is 3.61%.

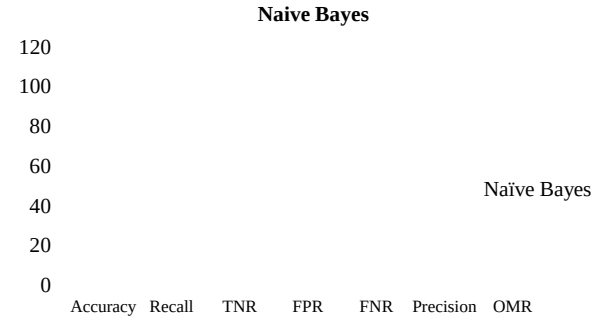

Fig3. Graph for model based on Naïve Bayes

The results of model based on Random Forest are shown in below graph. The accuracy is 99.80%. FPR and FNR is 0.03% and 2.78% respectively. Its OMR is also less i.e. 0.11%.

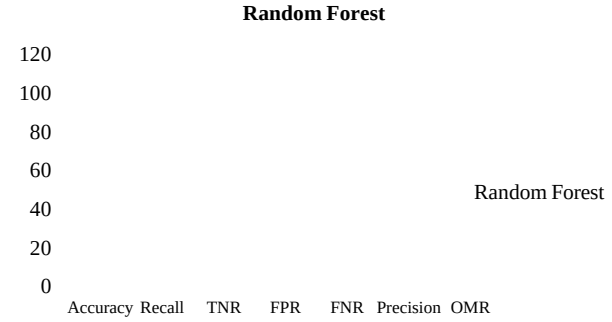

Fig 4. Graph for model based on Random Forest

The comparison graph for both models is given below. The graph shows that the proposed model has high accuracy and less overall misclassification error. The percentage rate of both FPR and FNR is also less for proposed model. The model based on Random Forest is more practical to use than model based on Naïve Bayes.

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

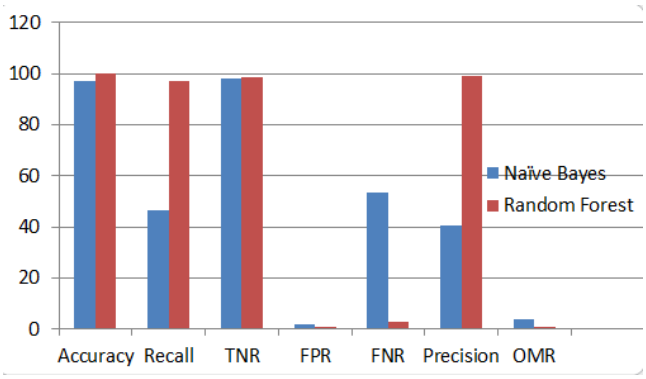

Fig 5. Graph for Comparison of Results

Random Forest predicts faults in imbalanced data more accurately than other classification techniques.

6. CONCLUSION

The result demonstrates that the Random Forest can be used to predict faults in software containing imbalanced data. Random Forest achieves higher accuracy and overall misclassification error is also less which indicates that the percentage of misclassification of faulty modules as non-faulty or vice versa is very less.

The Accuracy is 99.80%, its Recall is 97.20%, its precision is 99.80% and its overall misclassification error is 0.11%. This results show that Random Forest Algorithm can be used in constructing models to predict faults in software.

In future, the other vital characteristic can be determined for fault prediction by further implementing and comparing the proposed model with the models. Datasets from different repositories and software applications can be used for fault prediction with the proposed model.

REFERENCES

[1] A.A. Shahrjooi Haghighi, M. Abbasi Dezfuli, S.M. Fakhrahmad,"Appling Mining Schemes to Software Fault Prediction: A Proposed Approach Aimed at Test Cost Reduction," Proceedings of the World Congress on Engineering, 2012.

[2] A Kaur, P S Sandhu and A S Brar, "Early Software Fault Prediction using Real Time Defect Data," Second International Conference on Machine Vision, 2009.

[3] Cagatay Catal, B. D, "An Artificial Immune System Approach for Fault Prediction in Object-Oriented Software," 2nd International Conference on Dependability of Computer System . Turkey: IEEE, 2007.

[4] C Catal, "Performance Evaluation Metrics for Software Fault Prediction Studies," Acta Polytechnica Hungarica, 2012.

[5] Chao Chen, Andy Liaw, Leo Breiman, "Using Random Forest to Learn Imbalanced Data," pp. 1-12, July 2004.

[6] Cong Jin, E.-M. D.-N, "Software Fault Prediction Model Based on Adaptive Dynamical and Median Particle Swarm Optimization," Second International Conference on MultiMedia and Information Technology. China: IEEE. pp. 44-47.2010

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

[7] Deepak Gupta, V. K, "Analysis of Clustering Techniques for Software Quality Prediction," Second International Conference on Advanced Computing & Communication Technologies, pp. 6-9, 2012
[8] Hassan, A. E. "Predicting faults using the complexity of code changes," Vancouver, Canada: IEEE, pp.16-24 , May 2009.
[9] Hemant Palivela, Y. H, "A Study of Mining Algorithms for Finding Accurate Results and Marking Irregularities in Software Fault Prediction," International Conference on nformation Communication and Embedded Systems (ICICES). Banglore: IEEE, 2016
[10] Kehan Gao, T. M, "Impact of Data Sampling on Stability of Feature Selection for Software Measurement Data," IEEE 23rd International Conference on Tools with Artificial Intelligence, Boca Raton, Florida, pp. 7-9. USA, 2011.
[11] Kehan Gao, T. M. "Software Defect Prediction for High-Dimensional and Class-Imbalanced Data," 23rd International Conference on Software Engineering & Knowledge Engineering. Eden Roc Renaissance, Miami Beach, USA, 2011
[12] Mikel Galar, A. F, "A Review on Ensembles for the Class Imbalance Problem: Bagging, Boosting and Hybrid Based Approaches," IEEE transactions on systems, man and cybernetics, July 2012
[13] Partha Sarathi Bishnu, V. B., "Software Fault Prediction using Quad Tree-based K-means Clustering Algorithm," IEEE Transactions on Knowledge and Data Engineering, vol. 24 No. 6, June 2012.
[14] Parvinder S. Sandhu, J. S, "A K-Means Based Clustering Approach for Finding Faulty Modules in Open Source Software Systems," World Academy of Science, Engineering and Technology 48, p.p. 654-658, 2010.
[15] Ritika Sharma, N. B," Study of Predicting Fault Prone Software Modules," International Journal of Advanced Research in Computer Science and Software Engineering, 2012
[16] R Malhotra, A Kaur," Application of Random Forest in Predicting Fault-Prone Classes," International Conference on Advanced Computer Theory and Engineering. Delhi. IEEE, pp. 37-43, 2012.
[17] S. G., A. K., M. A., & D kaur, "A Clustering Algorithm for Software Fault Prediction," IEEE international conference on computer and communication technology, pp. 603-607, 2010.
[18] D P Sandhu, M kaur and A kaur, "A Density Based Clustering Approach for Early Detection of Fault Prone Modules," IEEE International Conference on Electronics and Information Engineering, pp. 525-530, 2010.
[19] N Seliya, M Tagi. Khoshgoftaar and J V Hulse, "Predicting Faults in High Assurance Software," IEEE 12th International Symposium on High Assurance Systems Engineering, 2012.
[20] R Shatnawi, "Improving Software Fault Prediction For Imbalanced Data," IEEE International Conference in Information Technology (IIT), pp. 54-59, 2012.
[21] Wang, Shuo & Yao, Xin, "Diversity Analysis on Imbalanced Data Sets by Using Ensemble Models," IEEE Symposium on Computational Intelligence and Data Mining, CIDM - Proceedings. 324-331, 2009
[22] X Y Liu, J. W.-H, "Exploratory Undersampling for Class-Imbalance Learning," IEEE Transactions on system, man and cybernetics – PART B, 2009.
[23] Yue Jiang, J. L, "Variance Analysis in Software Fault Prediction Models," IEEE 20th International Symposium on Software Reliability Engineering, 2009.
[24] Zhiwei Xu, T. M, "Prediction of Software Faults Using fuzzy Nonlinear Regression Modeling, " 281-290. USA: IEEE, 2009.

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

[25] Rathore, S.S. & Kumar, "A study on software fault prediction techniques,"Artif Intell Rev, Springer, 2019.

[26] Zhao H., Wei Z. and Yan H, "Multimodal Co-clustering Analysis of Big Data Based on Matrix and Tensor Decomposition," In: Seng K., Ang L., Liew AC., Gao J. (eds) Multimodal Analytics for Next-Generation Big Data Technologies and Applications, Springer, Cham, 2019.

[27] Y Suresh, L Kumar and S Ku. Rath,"Statistical and Machine Learning Methods for Software Fault Prediction Using CK Metric Suite: A Comparative Analysis," ISRN Software Engineering, vol. 2014, Article ID 251083, 15 pages, 2014.