I J C T A, 9(5), 2016, pp. 2173-2182
© International Science Press

Efficient Fault Prediction of Dissemblance Data Using Random Forest Algorithm

G. Saravana Kumar*, and B. Palanichelvam**

ABSTRACT

Software development always face a challenge in formulating mechanism for finding faults and developing effective methods and measures to remove these faults. Reducing the number of defects from 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 proper resource allocation and reducing 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 developing software fault prediction model that deals with imbalanced data. 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 a significant role in the quality of the software. Fault is a defect that may or may not causes failure while faults are the causes for internal errors. Because of human mistakes such as missing instruction, incorrect syntax etc errors are generated that may result in a fault in software. Software does not work according to the user needs in case of software failures. Bug is the non-conformance of software to its requirements. The main objective of all software industries is to reduce number of defects from software while developing it.

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 following basic methods: Integration testing, Peer review and Defect prediction. With the help of these methods, the unseen faults could be taken out and removed resulting a high quality software.

Fault prediction is to predict the probability that the software contains faults. To identify whether the software contains modules with the defect, 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 contains less number of faults and users are much satisfied. Software with less number of faults not only increase quality but its reliability also.

*,** Assistant Professors, Dept. Of CSE, Syed Ammal Engineering College, Ramanathapuram, Tamilnadu, India, Email: saravana.exe@gmail.com, palanichelvam@gmail.com

As software fault prediction systems are developed with the help of prior experiences. 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, tools are developed. After estimating quality, resources are 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 best among all these techniques. Random Forest algorithm is being successfully applied for solving both classification and regression problems. It is therefore important to investigate the capabilities of Random Forest algorithm in predicting software quality [16]. Random forest generally exhibits a substantial performance improvement over the single tree classifier such as CART and C4.5 (both techniques are decision tress) [5].

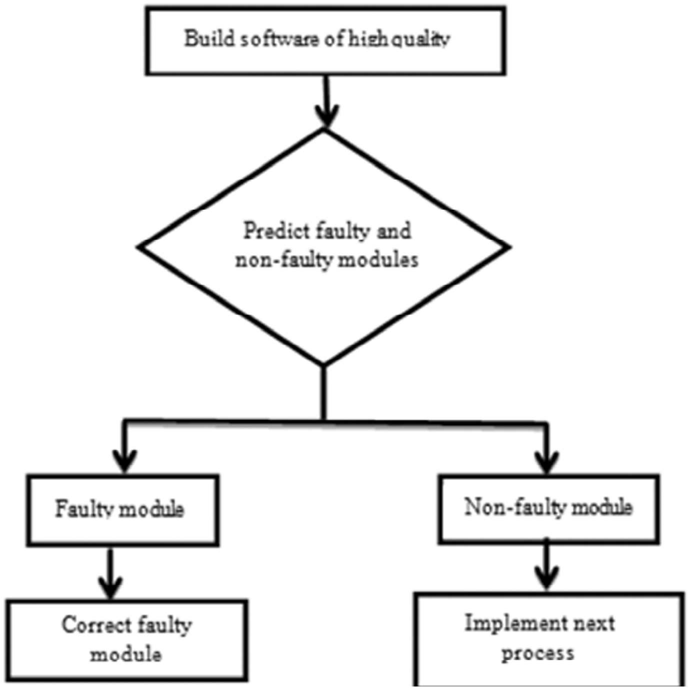

Figure 1: Software Fault Prediction

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 made from bootstrap samples of the training data, using random feature selection in the tree induction process. Prediction is made by aggregating (majority vote for classification or averaging for regression) the predictions of the ensemble.

The random forests algorithm is very much similar the bagging algorithm. Algorithm for random forest is given below:

- Let the number of training cases be *N* and number of variables in the classifier be *M*.
- Use number *m* of input variables to determine the decision at a node of a tree; *m* should be less than *M*.
- Training set for this tree is chosen by choosing *N* times with replacement form all *N* available training cases. Use the rest of the cases to estimate the error of the tree, by predicting their classes.

- For each node of the tree, randomly choose m variables. Calculate the best split based on these m variables in the training data set.
- Each tree is fully grown and not pruned.

The advantages of Random Forest are:

- It runs well on large data sets.
- It helps in resolving class imbalance problem.
- It helps in classification and prediction.
- It helps in estimating which variables are significant in the classification.
- It can handle thousands of variables without any variable deletion.
- It performs faster than boosting and bagging.

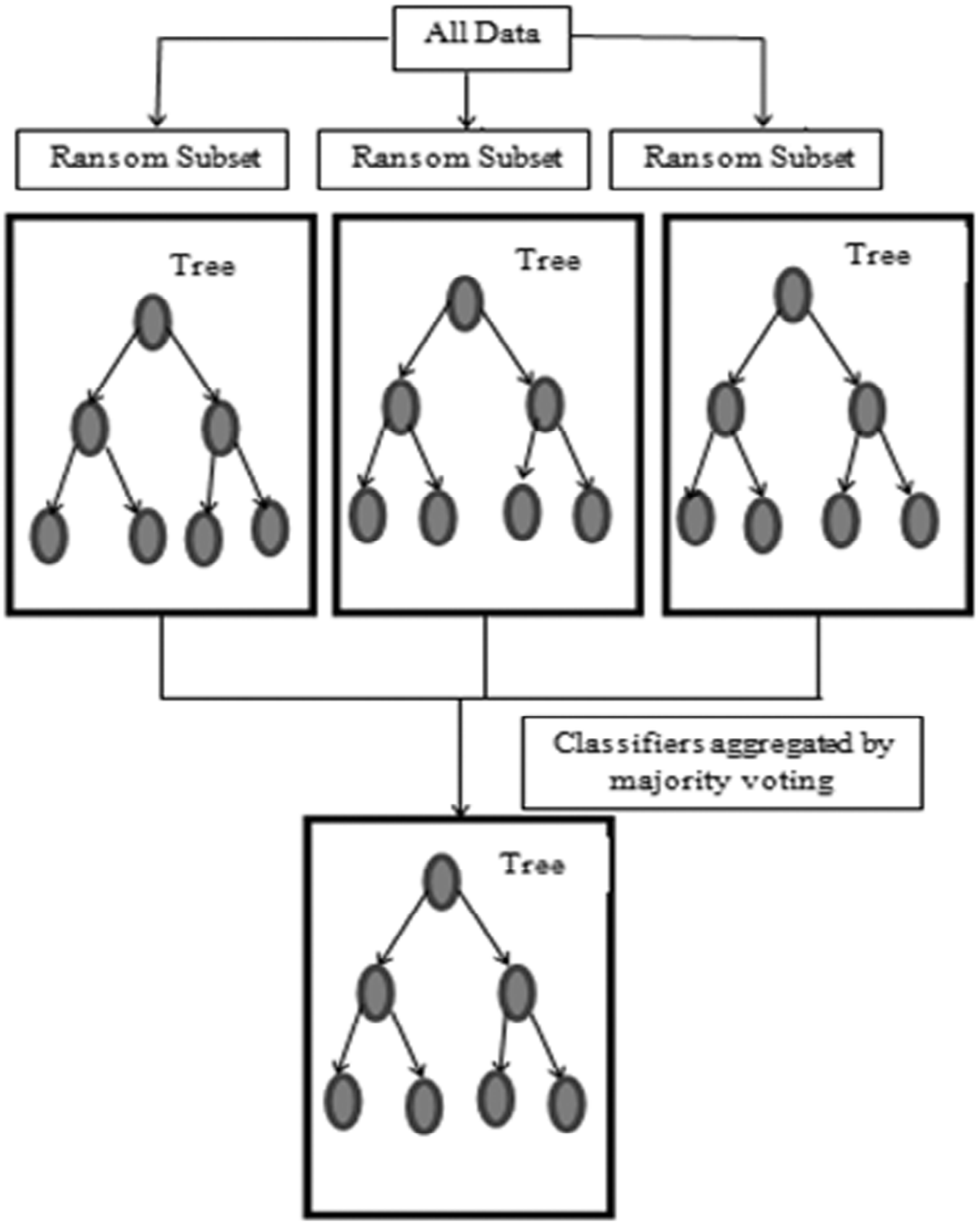

Figure 2: Random Forest

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 these two algorithms perform better than under-sampling because majority 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] addressed the class imbalance problems occurring when developing fault prediction model. He had used Roughly Balanced Bagging Algorithm that integrates both data sampling and bagging into one method that assists 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 software quality assurance systems. Each data set contains the information about number of faulty, non-fault modules 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 better 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 output 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 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:

- High dimensionality of software measurement data.
- Imbalanced distributions between the two types of modules.

These problems can be solved by using techniques such as feature selection and data sampling.

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 software metrics and defect data.

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:

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_COMPLEXITY, 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 run this technique on Weka and compare 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 the following table 1.

Where NFP is non-faulty modules

FP* is faulty Modules

Table 1
Confusion Matrix

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

Here,

- TN (True Negative): number of non-faulty modules that are correctly classified as non-faulty modules. Here 16943 are correctly classified as non-faulty.
- FN (False Negative): number of faulty modules that are incorrectly identified as non-faulty module. Here 42 faulty modules are incorrectly classified as non-faulty.
- TP (True Positive): number of faulty modules that are correctly identified as faulty module. Here 461 faulty modules are correctly classified as faulty.
- FP (False Positive): number of non-faulty modules 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 (*AC*) measures the chances that the fault proneness of module is correctly predicted. Its equation is as follows:

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

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

5.1.2. Precision

Precision (*P*) measures 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 faulty modules are correctly identified. Its equation is as follows:

$$TPR = ^{TP} /(FN + TP)$$

$$TPR = \frac{489}{14 + 489} = 97.2\%$$

√ **False Positive Rate (FPR):**

The false positive rate (*FPR*) is the proportion of the non-faulty modules that are 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 defined as the proportion of non-faulty modules 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 (*FNR*) is the proportion of faulty modules that are incorrectly identified as non-faulty. Its equation is as follows:

$$FNR = \frac{FN}{FN + TP}$$

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

5.1.3. 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:

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%.

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%.

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.

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

Table 2
Comparison of Results

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

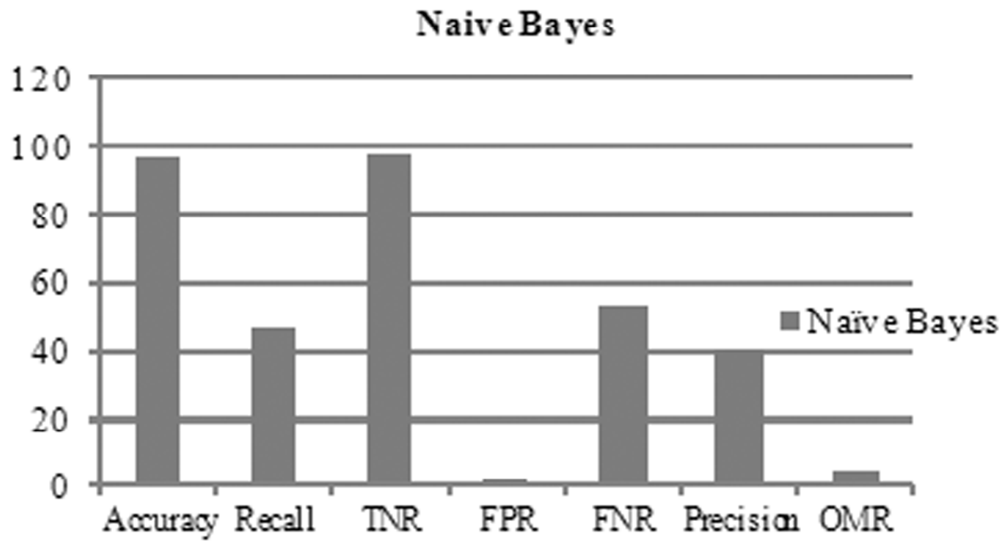

Figure 3: Graph for model based on Naïve Bayes

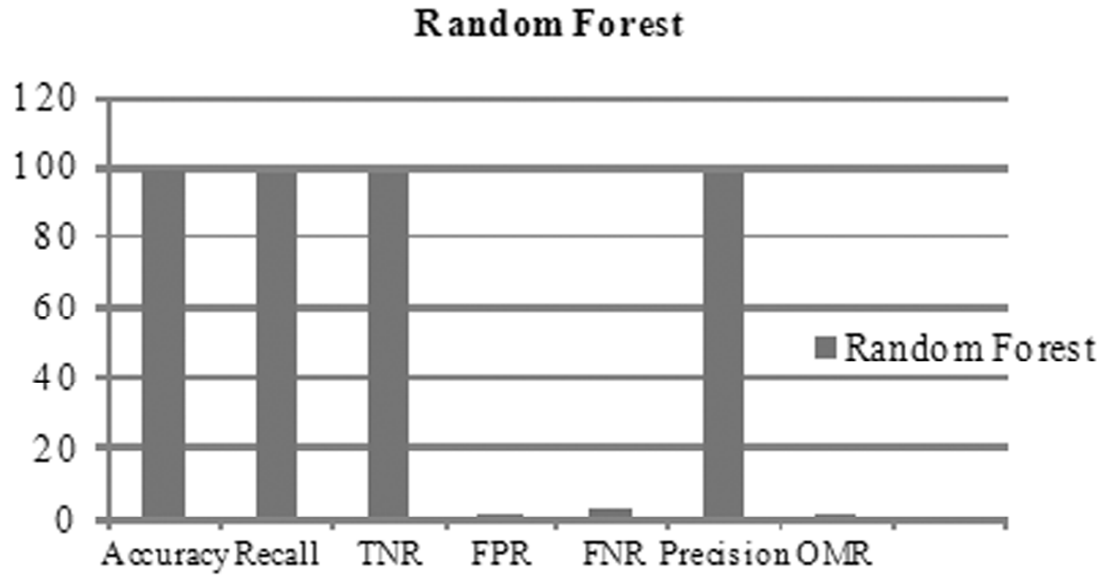

Figure 4: Graph for model based on Random Forest

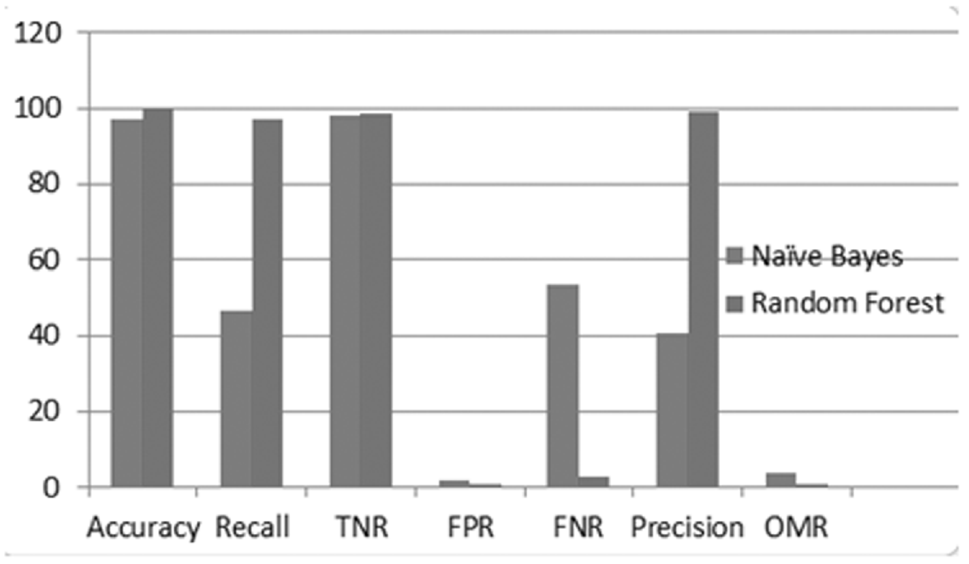

Figure 5: Graph for Comparison of Results

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, most important attribute can be found for fault prediction and this work can be extended to further programming languages. More algorithms can be evaluated and then we can find the best algorithm. Data from more programming languages can be taken.

REFERENCES

[1] A.A. Shahrjooj Haghighi, M. A. (2012). Appling Mining Schemes to Software Fault Prediction: A Proposed Approach Aimed at Test Cost Reduction. *Proceedings of the World Congress on Engineering* .

[2] Arashdeep, K. (2009). Early Software Fault Prediction using Real Time Defect Data. *Second International Conference on Machine Vision.*

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

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

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

[6] Cong Jin, E.-M. D.-N. (2010). Software Fault Prediction Model Based on Adaptive Dynamical and Median Particle Swarm Optimization. *Second International Conference on MultiMedia and Information Technology* (pp. 44-47). China: IEEE.

[7] Deepak Gupta, V. K. (2012). Analysis of Clustering Techniques for Software Quality Prediction. *Second International Conference on Advanced Computing & Communication Technologies*, (pp. 6-9).

[8] Hassan, A. E. (2009, May). Predicting faults using the complexity of code changes. 16-24. Vancouver, Canada: IEEE.

[9] Hemant Palivela, Y. H. (2013). 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.

[10] Kehan Gao, T. M. (2011). Impact of Data Sampling on Stability of Feature Selection for Software Measurement Data. *IEEE 23rd International Conference on Tools with Artificial Intelligence*, (pp. 7-9). Boca Raton, Florida USA.

[11] Kehan Gao, T. M. (2011). Software Defect Prediction for High-Dimensional and Class-Imbalanced Data. *23rd International Conference on Software Engineering & Knowledge Engineering* . Eden Roc Renaissance, Miami Beach, USA.

[12] Mikel Galar, A. F. (2012, July). A Review on Ensembles for the Class Imbalance Problem: Bagging, Boosting and Hybrid Based Approaches. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*.

[13] Partha Sarathi Bishnu, V. B. (June 2012). Software Fault Prediction using Quad Tree-based K-means Clustering Algorithm. *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, No. 6.

[14] Parvinder S. Sandhu, J. S. (2010). A K-Means Based Clustering Approach for Finding Faulty Modules in Open Source Software Systems. *World Academy of Science, Engineering and Technology 48*, 654-658.

[15] Ritika Sharma, N. B. (2012). Study of Predicting Fault Prone Software Modules. *International Journal of Advanced Research in Computer Science and Software Engineering*.

[16] Ruchika Malhotra, Arvinder Kaur (2008). Application of Random Forest in Predicting Fault-Prone Classes. *International Conference on Advanced Computer Theory and Engineering*. Delhi. (pp 37-43). IEEE

[17] S. G., A. K., M. A., & Kaur, D. (2010). A Clustering Algorithm for Software Fault Prediction. *IEEE international conference on computer and communication technology*, (pp. 603-607).

[18] Sandhu, D. P., & manpreet kaur and amandeep kaur. (2010). A Density Based Clustering Approach for Early Detection of Fault Prone Modules. *IEEE International Conference on Electronics and Information Engineering*, (pp. 525-530).

[19] Seliya, N., Tagi M. Khoshgoftaar, & Jason Van Hulse. (2010). Predicting Faults in High Assurance Software. *IEEE 12th International Symposium on High Assurance Systems Engineering*.

[20] Shatnawi, R. (2012). Improving Software Fault Prediction For Imbalanced Data. *IEEE International Conference in Information Technology (IIT)*, (pp. 54-59).

[21] Shuo Wang, X. Y. (2009). Diversity Analysis on Imbalanced Data sets by using Ensemble Models. IEEE.

[22] Xu-Ying Liu, J. W.-H. (2008). Exploratory Undersampling for Class-Imbalance Learning. *IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS – PART B*.

[23] Yue Jiang, J. L. (2009). Variance Analysis in Software Fault Prediction Models. *IEEE 20th International Symposium on Software Reliability Engineering*.

[24] Zhiwei Xu, T. M. (2000). Prediction of Software Fau.

[25] lts Using fuzzy Nonlinear Regression Modeling. 281-290. USA: IEEE.