Performance Gauging of Discrete Classifiers by Prognostication of a Reported Bug

Mary Tannia Padua, Amitha Joy, G Deepa

Abstract*: Bugs are the frequently transpiring drawback in software's, to prevent these problems; an in-depth study of bugs is required. Bug repositories are a significant supply of information. The bug repository helps the software team to have a better study about bugs and its related parameters. Often arising bugs helps the developers to get rid of them in upcoming releases. There is a huge variety of algorithms that facilitate in finding bugs. This paper intends to measure the performance of mining algorithms in predicting the bug severity by scrutinizing their capabilities. We propose a study of assorted algorithms in Lazy (IBK and KStar) and Tree (Random Forest) Classifiers with regard to different parameters. The case inspected here is Mozilla_Thunderbird, which is implemented in WEKA.

***Keywords:* IBK, KStar, Random Forest, Resampling**

I. INTRODUCTION

Bugs also known as defects are reportable by engineers, standard analysts and users of the merchandise [4]. The reported bugs are kept within the bug repositories like Mozilla, Google Chromium, Android, etc. First and foremost, finding the often occurring bugs is the right answer for the software malfunctioning. There are many algorithms that facilitate finding the proper solution. Bug mining is the process of discovering bugs, determining its useful patterns and associated relationship among them. These patterns help in the classification of bugs into different class labels based on their severity. These class labels help the developers in determining the relevance of various bugs present in software and it is resolved and fixed by them. Fault detection is tracked by different tools like Bugzilla, JIRA, etc [3] [4].

The vital part of a bug report is the bug summary data; the data chosen here for analysis is Mozilla_Thunderbird. The study is finished on the premise of accuracy, precision, recall, root mean squared error, etc. from an inventory of classifiers in weka, we have examined Tree Classifier - Random Forest and Lazy Classifier - IBK and KStar.

Revised Manuscript Received on April 30, 2020.
* Correspondence Author
Mary Tannia Padua*, PG Student, Department of Computer Science & IT, Amrita School of Arts and Sciences, Kochi, Amrita Vishwa Vidyapeetham, India. Email: tannia.padua@gmail.com
Amitha Joy, PG Student, Department of Computer Science & IT, Amrita School of Arts and Sciences, Kochi, Amrita Vishwa Vidyapeetham, India. Email: amithajoy246@gmail.com
G Deepa, Assistant Professor, Department of Computer Science and IT, Amrita School of Arts and Sciences, Kochi ,Amrita Vishwa Vidyapeetham, India .Email: deepsgopi@gmail.com

II. LITERATURE REVIEW

In [1], the authors applied J48, ID3 and Naive Bayes data mining algorithms for classification and detection of software bugs by comparing their accuracy and time taken to build the model. From the finding they concluded that j48 is best among the three in terms of accuracy. In [2], comparing four text mining algorithms: Naive Bayes Multinomial, K-Nearest Neighbor and Support Vector Machines with respect to accuracy and training set size. They classified bug reports into two categories as severe or non severe. Eclipse and GNOME are the dataset taken for analysis. The result showed Naive Bayes Multinomial as the best among them, with respect to accuracy.

In [3], performs the mining of bugs from the bug summary data to find the frequently occurring pattern. Text mining techniques like tokenization and stop-words removal are performed before applying FP-Growth. A better frequent pattern was generated. [4] Text mining techniques are applied on the bug data to find the frequent pattern by using FP Growth algorithm. In [5], an algorithm is formulated using the data mining techniques to detect the software bug. The Clementine data mining tool is used for modeling.

II. METHODOLOGY

In this section, we briefly discuss bug mining, the different mining techniques and associated algorithms, which we used for our proposed work, done in WEKA. The following techniques are performed on the data.

A. Bug Summary

The summary of a bug report is to elucidate the bug to the developer. It contains several attributes like Bug ID, Title, Description, Status etc. Using these, developers will check the importance of bugs and conjointly checks for duplication [3]. Bug Summary chosen for analysis is Mozilla_Thunderbird.

B. Resampling

It is an efficient method which produces a random subsample of the dataset. It either uses sampling with replacement or without replacement according to the nature of the dataset. Resampling within the supervised version works just for a dataset having nominal category attribute. In WEKA, it's found underneath the instance of the supervised filter

C. Random Forest

It consists of an oversized variety of individual trees that operate altogether.

Every individual tree within the random forest spits out a category prediction and therefore, the class with the foremost votes becomes our models prediction. The basic idea behind random forest is the low interrelationship between models.

D. IBK

In WEKA, k-Nearest Neighbor algorithmic program is termed as IBK (Instance Based Learner). It generates a prediction for a test instance instead of building a model. The algorithm finds k "close" instances within the training data for every test instance and uses those to form a prediction.

E. KStar

KStar (K*) is a Heuristic Search Algorithm for finding the k shortest ways. It is an instance based classifier which categorizes the training data based on the test data by using a similarity function.

III. PROPOSED WORK

Here, we propose a model which helps in the prediction of bugs and classifying them into two categories labeling Resolved and Verified.

Bug repositories are the main supply of knowledge associated with bugs in software. Common attributes in a bug repository include: Bug ID, Priority, Component, Title, Description, Status etc. The bug summary extracted from bug repository is preprocessed using Resample filter and further classified. Preprocessing technique is applied in order to transform the unprocessed data into proper information. It cleans the data by removing noises, transforms data to proper formats and reduces the amount of data in use.

We have divided our data into 60 and 40 percentages for training and test set respectively before Resampling filter [6]. This method produces better results than other filters when applied with classifiers to our data. Later, the dataset is trained using Tree and Lazy classifiers and then saved and reloaded the models and reevaluated each model with the supplied test set. Classification helps in flawlessly finding the class label for each data. The method concerned in our work is illustrated in Fig. 1.

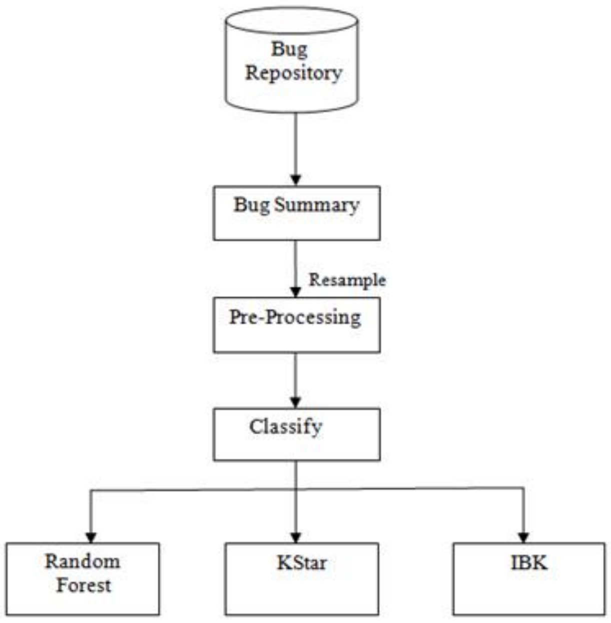

Fig. 1. Proposed Method

IV. EXPERIMENTAL RESULTS

The dataset considered for inspection contains the bugs in Mozilla, which consists of 10000 instances, from which 6000 is taken as train data and 3000 as test data. After training the data, performance of the models are assessed in terms of Accuracy. Accuracy is determined by calculating the percentage of correctly classified instances.

The dataset is evaluated for Tree and Lazy classifiers and thereafter the following results are obtained as shown in TABLE I. The factors considered for comparison are: accuracy, precision, recall, RMSE and F-Measure [1].

From Table 1, it is clear that Random Forest shows better results in terms of accuracy compared to other classifiers. It also has the highest precision and lowest RMSE. Lower RMSE worth indicates higher match. Although F-Measure values of both Random Forest and KStar provide the same output, however other parameters of Random Forest have got higher values. These are graphically represented in Fig. 2, and Fig. 3.

Table- I: Detailed test result

Classifier	Accuracy	Precision	Recall	RMSE	F-Measure
Random Forest	90.952	0.879	0.91	0.286	0.868
KStar	90.6	0.851	0.906	0.2972	0.868
IBK	84.179	0.844	0.842	0.3982	0.843

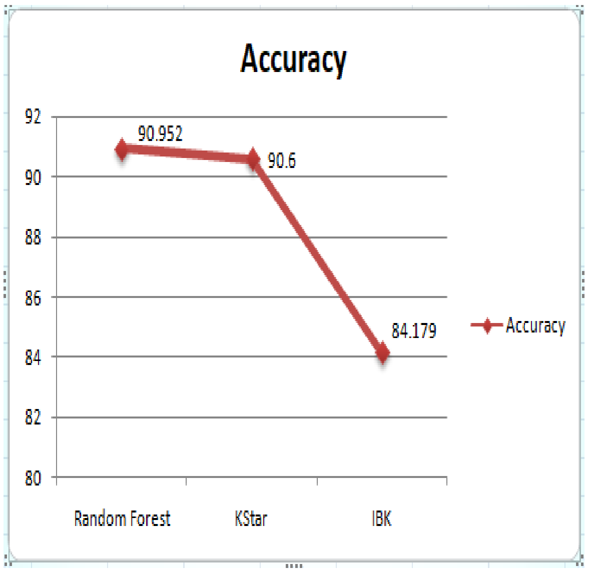

Fig. 2. Classifiers Accuracy Values

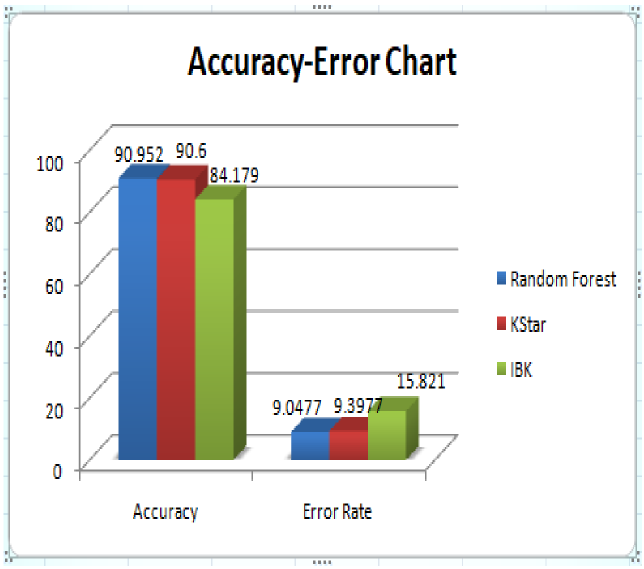

Fig. 3. Graphical Representation of Accuracy by Class

V. CONCLUSION

Most of the bugs in software are due to human intervention. When an error occurs in software, the cost and effort required to endure its terribly high [3]. These errors can be reported in a bug repository. The bug summary data from the repository helps the developers to take preventive measures. This paper puts forward a much better technique for predicting the severity of a bug. Applying these classifiers to different datasets produces different results .The results are also affected by the quantity of data used. To find which algorithm will give the best accuracy the data collected is analyzed and compared. From our study we have concluded that Random Forest shows better results for our dataset regarding high accuracy which is 90.952% and low RMSE which is 0.286. Our future work will focus on the mixture of various classification techniques that can be used to improve the performance in prediction.

REFERENCES

1. Dhyan Chandra Yadav, Saurabh Pal "Software Bug Detection using Data Mining" IJCA Vol. 115 - No. 15, April 2015
2. Ahmed Lamkanfi, Serge Demeyer, Quinten David Soetens, Tim Verdonck "Comparing Mining Algorithms for Predicting the Severity of a Reported Bug" March 2011
3. K Divya Varma, M Remya, G Deepa "An Enhanced Bug Mining for Identifying Frequent Bug Pattern using Word Tokenizer and FP-Growth" March 2017
4. Kiran Kumar B, Jayadev Gyani, Narsimha G "Mining Frequent Patterns from Bug Repositories" Vol. 4, Issue 5, May 2014
5. Safia Yasmeen "Software Bug Detection Algorithm using Data mining Techniques" IJIRAE, Vol. 1, Issue 5, June 2014
6. A.Hency Juliet, R. Padmajavalli "Comparative Analysis of Classification Algorithms on Endometrial Cancer Data" IJST, Vol. 9, July 2016
7. Tomas Pranckevicius, Virginijus Marcinkevicius "Comparison of Naïve Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification" BJMC January2017
8. C. Arjunchandra, C.J. Sandhya and G. Deepa "A Method for Improving the Accuracy of Bug Mining by Replacing Stemming with Lemmatization" IJPAM, Vol. 119, 2018
9. Sunil Joy Dommati, Ruchi Agarwal "Bug Classification: Feature Extraction and Comparison of Event Model using Naïve Bayes Approach" ICRTCIE April 2012
10. Smita Mishra, Somesh Kumar "Survey on Types of Bug Reports and General Classification Techniques in Data Mining" IJCSIT, Vol. 6, 2015
11. Lakshmi Devasena C "Comparative Analysis of Random Forest, REP Tree and J48 Classifiers for Credit Risk Prediction" IJCA, 2014
12. Fartash. Haghanikhameneh, Payam. Hassany Shariat Panahy, Nasim. Khanahmadliravi, Seyed Ahmad. Mousavi "A Comparison Study between Data Mining Algorithms over Classification Techniques in Squid Dataset" IJAI, Vol. 9, 2012
13. Promila Devi, Rajiv Ranjan "Enhanced Bug Detection by Data Mining Techniques" IJCER 2014, Vol. 04, Issue 7
14. Rajeshwari M.R, Dr. Kavitha K S "Reduction of Duplicate Bugs and Classification of Bugs using a Text Mining and Contingency Approach" IJAER, Vol. 12, 2017
15. Er. Amandeep, Er. Saurabh Mittal "A Review on Bug Tracking System Using Naïve Byes in Data Mining", IJARSE, Vol. 3, Issue 9, September 2014

AUTHORS PROFILE

Mary Tannia Padua, Integrated MCA final year student in Amrita Vishwa Vidyapeetham University. Presented a paper on "Health and Technology" in two day National Conference on Data Science-Applications, Analytics and Visualization, organized by the department of Computer Science and IT, Amrita School of Arts and Sciences, Kochi.

Amitha Joy, Integrated MCA final year student in Amrita School of Arts and Sciences, Amrita Vishwa Vidyapeetham University.

G Deepa, Master Degree in Computer Applications, almost 17 Years of teaching experiences. Currently working as Asst.Professor in Amrita Vishwa Vidyapeetham.