"AUTOMATED BUG CLASSIFICATION AND TRAIGE SYSTEM USING HYBRID ALGORITHM BASED ON DATA MINING"

[1]RAHUL V. BAMBODKAR
M. Tech Student, Department of Computer Science & Engineering, REC, Bhopal, India
rahulbambodkar1@gmail.com

[2]PROF. SHIVENDRA DUBEY
Assistant Professor, Department of Computer Science & Engineering, REC, Bhopal, India
shivendradubey5@gmail.com

ABSTRACT: *Flaw is only a blunder or bug which produces unforeseen and mistaken result. In Companies all product tasks are influenced by programming Flaws (bug).Every day new bugs are created and engineer needs to alter that bug or imperfection. Programming Company spends heaps of cash to alter them. Settling bug is hard so we will diminish this by utilizing some technique. Every time when bug is created we have to arrange that bug and for that reason we require classifier. Classifier is the procedure by which we can arrange the bug with the goal that we decide at which class that bug is have a place. In this paper we are utilizing two procedures NB (naive bayes) and KNN (k-nearest neighbour) for arrangement. NB depends on recurrence and KNN depends on word check. After the arrangement the bug is grouped and administrator can allocate them to the designer to alter. In this paper we likewise present element choice and example choice for diminishing database. Bug storehouse is the database which is utilized to store bug points of interest. In this paper mix of NB and KNN classifier is utilized which is more effective and take less time to arrange the bug so that administrator can allocate a legitimate bug of specific class to the ideal designer AND the bug will settle effortlessly. In the past paper manual triaging framework is utilized which is not effective and taking an excessive amount of time. In this paper we enhancing imperfection triage furthermore decreasing the database by utilizing these two systems.*

Keywords: bug triage, bug data reduction, bug classification technique.

1. INTRODUCTION

The vast majority of the organizations burn through 45% of expense in managing the product bugs. This bug squander the season of engineer who build up the venture. In this paper we enhance the imperfection triage by utilizing order strategy furthermore lessening the database .the information which is not valuable for altering the bug we will expel it. This all procedure is goes under the preprocessing where all the undesirable information is expelled and administrator get legitimate information which portray the bug subtle element. For putting away this bug point of interest we require database this is called bug archive.

Open source programming advancements fuse an open bug store that permits both designers and clients to post issues experienced with the product, propose conceivable improvements, and remark after existing bug reports. One potential point of interest of an open bug vault is that it might permit more bugs to be recognized and fathomed, enhancing the nature of the product created [12].

For overseeing programming bugs bug storehouse or bug settling assumes an essential part. Vast of programming which are open source ventures have an open bug archive which permits designers and also clients to submit issues or surrenders in the product that propose conceivable arrangements and remark on existing bug reports. The quantity of customary happening bugs for open source huge scale programming tasks is so much expansive that makes the triaging procedure exceptionally troublesome and testing .For altering programming bugs a large portion of programming organizations pays a great deal . The substantial scale and the low quality are primary two difficulties which are connected with bug information that may influence the successful utilization of bug archives in programming advancement tasks. Bug is kept up as a bug report in a bug store that records the recreating bug in literary frame and upgrades as per the status of bug altering [1].

This paper presents Preprocessing and Classification. Preprocessing is the procedure where undesirable information will evacuate and we get the helpful information for grouping and For Classification this two strategies NB (gullible base) and KNN (k nearest neighbor) are use which characterize the bug. Subsequent to applying the classifier, bug will characterized.

2. NB CLASSIFIER

The Naive Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.

To demonstrate the concept of Naïve Bayes Classification, consider the example displayed in the illustration above. As indicated, the objects can be classified as either GREEN or RED. Our task is to classify new cases as they arrive, i.e., decide to which class label they belong, based on the currently exiting objects.

Since there are twice as many GREEN objects as RED, it is reasonable to believe that a new case (which hasn't been observed yet) is twice as likely to have membership GREEN rather than RED. In the Bayesian analysis, this belief is known as the prior probability. Prior probabilities are based on previous experience, in this case the percentage of GREEN and RED objects, and often used to predict outcomes before they actually happen.

Thus, we can write:

$$Prior\ probability\ for\ GREEN \propto \frac{Number\ of\ GREEN\ objects}{Total\ number\ of\ objects}$$

$$Prior\ probability\ for\ RED \propto \frac{Number\ of\ RED\ objects}{Total\ number\ of\ objects}$$

2.1 Plan (Rule)- based Classifiers:

In standard based classifiers we choose the word plans which are well while in transit to be related to the particular classes. We construct a course of action of standards, in which the left-hand side identifies with a word plan, and the right-hand side looks at to a class name. These rules are used for the inspirations driving portrayal.

2.2 SVM Classifiers:

SVM Classifiers try to distribute data space with the usage of straight or non-direct frameworks between the particular classes. The key in such classifiers is to choose as far as possible between the unmistakable classes and use them for the purposes behind request.

2.3 Neural Network Classifiers:

Neural frameworks are used as a part of a wide combination of regions for the inspirations driving gathering. As to substance data, the principal contrast for neural framework classifiers is to modify these classifiers with the usage of word parts. We observe that neural framework classifiers are related to SVM classifiers; without a doubt, they both are in the order of discriminative classifiers, which are on the other hand with the generative classifiers [102]. Bayesian (Generative) Classifiers: In Bayesian classifiers (moreover called generative classifiers), we attempt to amass a probabilistic classifier in perspective of showing the essential word highlights in different classes. The thinking is then to request content considering the back probability of the reports having a spot with the particular classes on the reason of the word closeness in the records.

2.4 Different Classifiers:

All classifiers can be conformed to the example of substance data. A part of substitute classifiers consolidate nearest neighbor classifiers, and genetic figuring based classifiers. We will look at some of these particular classifiers in some unobtrusive component and their use for the occurrence of substance data. The area of substance grouping is inconceivable to the point that it is hard to cover all the various counts in inconspicuous component in a lone segment. In this way, we will likely give the peruser an audit of the most basic frameworks, moreover the pointers to the differing assortments of these procedures. Highlight decision is a basic issue for substance course of action. In highlight determination, we try to choose the components which are most huge to the request method. This is in light of the fact that a part of the words are significantly more obligated to be identified with the class movement than others. Hence, a wide grouping of procedures have been proposed in the written work with a particular finished objective to choose the most fundamental components with the final objective of plan. These consolidate measures, for instance, the gini-list or the entropy, which choose the level of which the proximity of a particular component skews the class allotment in the basic data. We will moreover inspect the particular segment determination methods which are routinely used for substance request.

3. LITERATURE SURVEY

***1.* Towards Effective Bug Triage with Software Data Reduction Techniques.[1]**

In this paper a bug document (a typical programming storage facility, for securing purposes of enthusiasm of bugs), expect a fundamental part in supervising programming bugs. Programming bugs are unavoidable and changing bugs is exorbitant in programming change. Programming associations spend more than 45 percent of cost in settling bugs. Broad programming wanders pass on bug vaults (furthermore called bug taking after structures) to support information collection and to help creators to handle bugs,. In a bug storage facility, a bug is kept up as a bug report, which records the printed delineation of impersonating the bug and updates as demonstrated by the status of bug adjusting. A bug store gives a data stage to reinforce various sorts of endeavors on bugs, e.g., inadequacy desire, bug restriction, and resuscitated bug examination. In this paper, bug reports in a bug storage facility are called bug data.

.

2 "Who should fix this bug?"[5]

In this paper they propose open bug store to which both creators and customers can report bugs. The reports that appear in this storage facility must be triaged to make sense of whether the report is one which requires thought and if it is, which architect will be consigned the commitment of deciding the report. Endless open source change sare agitated by the rate at which new bug reports appear in the bug document. In this paper, we display a semi-robotized approach proposed to straightforwardness one a player in this methodology, the errand of reports to an originator. Our philosophy applies a machine learning figuring to the open bug vault to take in the sorts of reports each designer decides. Right when another report arrives, the classifier made by the machine learning technique suggests somewhat number of fashioners appropriate to decide the report. With this philosophy, we have accomplished precision levels of 57% and 64% on the Eclipse and Firefox headway expands separately

***3.* Finding bugs in web applications using dynamic test generation and explicit-state model checking**.[3]

In this paper they propose DYNAMIC test time instruments, for instance, DART , Cute, and EXE , produce tests by executing an application on strong data qualities, and a short time later making additional information qualities by comprehending run of the mill objectives got from honed control stream ways. To date, such strategies have not been practical in the space of Web applications, which pose novel challenges as a result of the dynamism of the programming

tongues, the use of comprehended information parameters, their use of relentless state, and their flighty case of customer association. This paper extends component test period to the space of web applications that dynamically make web (HTML) pages in the midst of execution, which are normally shown to the customer in a project.

***4*. Towards graphical models for content preparing. [9]**

In this paper, they propose the concept of *distance graph representations* of text data. Such representations preserve information about the relative ordering and distance between the words in the graphs, and provide a much richer representation in terms of sentence structure of the underlying data. Recent advances in graph mining and hardware capabilities of modern computers enable us to process more complex representations of text. We will see that such an approach has clear advantages from a qualitative perspective. This approach enables knowledge discovery from text which is not possible with the use of a pure vector-space representation, because it loses much less information about the ordering of the underlying words. Furthermore, this representation does not require the development of new mining and management techniques**.**

5. Bug Tracking and Reliability Assessment System (BTRAS).[10]

In this paper they propose comprehensive classification criteria to review the available tools and propose a new tool named Bug Tracking and Reliability Assessment System (BTRAS) for the bug tracking/reporting and reliability assessment. BTRAS helps in reporting the bug, assigning the bug to the developer for fixing, monitoring the progress of bug fixing by various graphical/charting facility and status updates, providing reliability bug prediction and bug complexity measurements, and distributing fixes to users/developers.

6. Reducing the effort of bug report triage [11]

In this paper they propose cooperation amidst architect and customer. In open-source wanders, bug taking after systems are a basic bit of how gatherings, (for instance, the ECLIPSE and MOZILLA bunches) interface with their customer bunches. As a result, customers can be incorporated into the bug adjusting process: they show the primary bug reports and additionally share in talks of how to settle bugs. Subsequently they settle on decisions about the future heading of a thing. To a sweeping degree, bug taking after systems serve as the medium through which originators and customers associate and grant. In any case, grinding develops when settling bugs: engineers get disturbed and energetic over divided bug reports and customers are frustrated when their bugs are not instantly changed.

7. CLUBAS: An Algorithm and Java Based Tool for Software Bug Classification Using Bug Attributes Similarities [6]

In this paper, a product bug characterization calculation, CLUBAS (Classification of Software Bugs Using Bug Attribute Similarity) is exhibited. CLUBAS is a half breed calculation, and is planned by utilizing content bunching, continuous term figuring's and taxonomic terms mapping methods. The calculation CLUBAS is a case of arrangement utilizing grouping strategy. The proposed calculation works in three noteworthy strides, in the initial step content bunches are made utilizing programming bug literary qualities information and took after by the second step in which group marks are produced utilizing name incitement for every bunch, and in the third step, the group names are mapped against the bug taxonomic terms to recognize the proper classes of the bug groups. The group names are created utilizing successive and significant terms present in the bug characteristics, for the bugs having a place with the bug bunches. The outlined calculation is assessed utilizing the execution parameters F-measures and precision. These parameters are contrasted and the standard order procedures like Naïve Bayes, Naïve Bayes Multinomial, J48, Support Vector Machine and Wake's characterization utilizing bunching calculations. A GUI (Graphical User Interface) based instrument is additionally created in java for the execution of CLUBAS calculation.

8. Bug Triage with Bug Data Reduction.[8]

The paper is altogether dedicated to taking after the bugs that are hereby rise. The director keeps up the master bits of knowledge regarding the bugs id, bugs sort, bugs depiction, bugs earnestness, bugs status, customer purposes of premium. The head too has the ability to update the master purposes of enthusiasm of earnestness level, status level, etc. The supervisor incorporates the customers and doles out them commitment of completing the paper. Finally on examining the paper doled out to the particular customer, the executive can track the bugs, and it is actually added to the tables containing the bugs, in response to popular demand of reality and status. The supervisor can know the information in judgment the diverse paper's consigned to various customers, their bug taking after status, and their delineation et cetera as reports every so often. The paper completely uses the secured strategy for taking after the structure by realizing and joining the Server side scripting. The chief can now incorporate the endeavor modules, wander delineations et cetera. He too incorporates the earnestness level, its status et cetera.

9. Characterization and prediction of bug report [18]

The late improvements in variable and highlight determination have tended to the issue from the down to earth perspective of enhancing the execution of indicators. They have met the test of working on info spaces of a few thousand variables. Refined wrapper or implanted techniques enhance indicator execution contrasted with less difficult variable positioning strategies like connection strategies, however the changes are not generally noteworthy: spaces with expansive quantities of information variables experience the ill effects of the scourge of dimensionality and multivariate strategies may over fit the information. For a few areas, applying initial a technique for programmed

highlight development yields enhanced execution and a more minimal arrangement of elements. The strategies proposed in this unique issue have been tried on a wide assortment of information sets (see Table 1), which restrains the likelihood of making correlations crosswise over papers. Further work incorporates the association of a benchmark. The methodologies are exceptionally assorted and persuaded by different hypothetical contentions, yet a binding together hypothetical system is deficient. Due to these deficiencies, it is critical when beginning with another issue to have a couple gauge execution values. To that end, we prescribe utilizing a direct indicator of your decision (e.g. a straight SVM) and select variables in two substitute courses: (1) with a variable positioning technique utilizing a connection coefficient or shared data; (2) with a settled subset choice strategy performing forward or in reverse determination or with multiplicative upgrades. Further not far off, associations should be made between the issues of variable and highlight choice and those of test configuration and dynamic learning, with an end goal to move far from observational information toward exploratory information, and to address issues of causality derivation.

4. PROBLEM STATEMENT

1. In existing system, a Bug sorting System is planned within which 2 major knowledge set square measure used and data reduction techniques square measure planned with the assistance of instance choice and feature choice.
2. Instance choice and feature choice square measure used for knowledge reduction and higher quality of Bug. In existing system, no text classification rule is planned to avoid manual classification that is incredibly time overwhelming.
3. Thus to avoid such state of affairs we have a tendency to extend our base paper practicality with automatic classification of Bugs mistreatment hybrid combination of KNN & NB classifier system.
4. In existing system nothing is incredibly abundant mentioned concerning automatic classification which may really scale back time in bug sorting system.
5. The potency of the present system is any extended by mistreatment Hybrid Algorithms.

6. PROPOSED WORK

The main aim of proposed system is to classify & enlist the bugs efficiently into different categories using hybrid combination of KNN & Naïve bayes algorithm so that the detected bugs can be efficiently solved by concerned user. Also, an approach for efficient automatic bug triage & classification is undertaken in proposed system.

Generic Strategy for Classifying a Text Document

The main steps involved are
i) Document pre-processing,
ii) Feature extraction / selection
iii) Model selection
iv) Training and testing the classifier

We present the problem of data reduction for bug triage. This problem aims to augment the data set of bug triage in two aspects, namely
a) To simultaneously reduce the scales of the bug dimension and the word dimension.
b) To improve the accuracy of bug triage.

We propose a combination approach to addressing the problem of data reduction. This can be viewed as an application of instance selection and feature selection in bug repositories. We build a combination of NB and KNN classifier to predict the class of the bug. These two techniques are never use in combine form. So, we are using this combination for increasing the efficiency & accuracy.

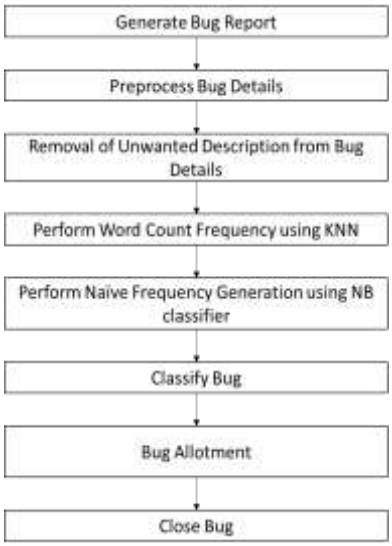

Figure 1: Bug Preprocessing & Classification

Figure 1 shows flowchart of Bug Preprocessing & Classification where bug reports taken from open sources are feed into the system for bug preprocessing. Bug preprocessing removes unwanted symbols, stop words, digits, etc from the bug reports through the proposed algorithm. After Preprocessing, analysis of preprocessed bug report using two different algorithms like KNN & Naïve Bayes Classifier. And according to this analysis, efficient bug classification & allotment is done & status of concern bugs submitted.

Algorithm:
Algorithm Preprocess (Data D)

Step 1: Read Data into Array
Step 2: Remove All Stop words
$$\Sigma\, i = 0 \mid \phi\, n \neq stop(i)$$
Step 3: Remove Redundancy from Array
$$\Sigma\, i = 0 \mid \phi\, n \neq repeat(i)$$
Step 4: Remove all Special Symbol and digits.
Step 5: Write back

Algorithm Hybrid Classification (Data D)
Step 1: Read Data into Array
Step 2: Call Preprocess (D)

Step 3: Calculate Word count

$\Sigma i = 0 \mid \phi\, i = i + 1$ if $a[i] \in final[i]$

Step 4: Calculate Frequency

TF = number of occurrences / total words

Step 5: Calculate Normalized TF

NTF = sum of TF / number of classes

Step 6: Generate Decision Matrix

Step 7: Calculate Final max class value and classify.

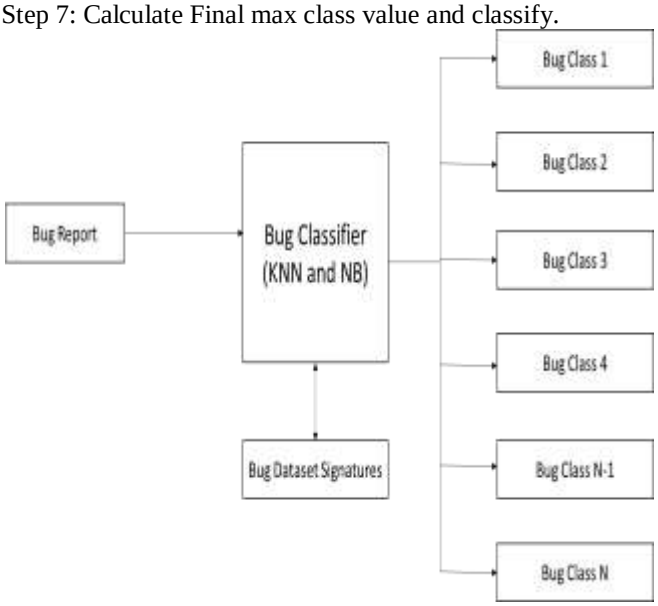

Figure 2: System Architecture

Figure 2 shows system architecture of proposed system where bug reports are feed to the hybrid combination of KNN & Naïve bayes Classifier. And according to the classification results concern bug is categorized & enlisted into the appropriate bug classes. Hybrid combination of KNN & Naïve bayes Classifier also uses bug dataset signature at the time of bug classification.

7. OVERVIEW OF DATASET

We have used FINDBUGS Categories as our bug dataset for unstructured bug categories.

Some of Bug categories are:

7.1 Correctness bug

Probable bug - an apparent coding mistake resulting in code that was probably not what the developer intended. We strive for a low false positive rate.

7.2 Bad Practice

It is the code which has violations of recommended and essential coding practice. Examples include hash code and equals problems, clone able idiom, dropped exceptions, serializable problems, and misuse of finalize. We strive to make this analysis accurate, although some groups may not care about some of the bad practices.

7.3 Dodgy Code

It is the code that is confusing, anomalous, or written in a way that leads itself to errors. Examples include dead local stores, switch fall through, unconfirmed casts, and redundant null check of value known to be null. More false positives accepted. In previous versions of FindBugs, this category was known as Style.

7.4 Malicious code vulnerability

It is the code that is vulnerable to malicious code like Trojan or which can send data to another application. Such code comes under this class.

7.5 Performance

It is the code that degrades the performance of the system by some looping or function calling. It mainly includes boxing and unboxing primitives of a program.

7.6 Security

Code that is vulnerable to security attacks. For Ex: Hardcoded password that is always constant in database or constant OTP for all users.

7.7 Multithreaded correctness

Code related to concurrency control and read write permissions to user in the system. It mainly includes multithreading program ambiguity. For Ex: In JAVA multiple run methods cause this kind of issue.

8. EXPERIMENTAL RESULTS

The proposed system is implemented in Java and MySQL. The dataset is provided with 9 classes from FindBugs 2.0 that is openly available on GitHub and SourceForge. The algorithm used for classification is hybrid combination of KNN and Naïve Bayes and we also provide comparative study for both this algorithms in terms of Bug Triage System.Here, different screenshots of GUI are shown along with their detail descriptions and navigations from one page to another.

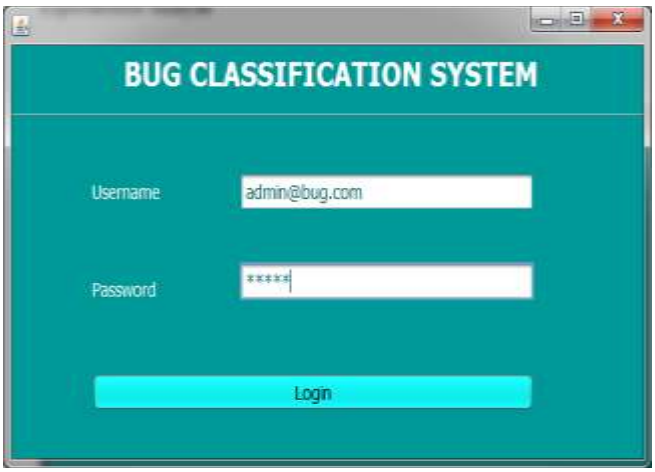

Figure 3: Login Page

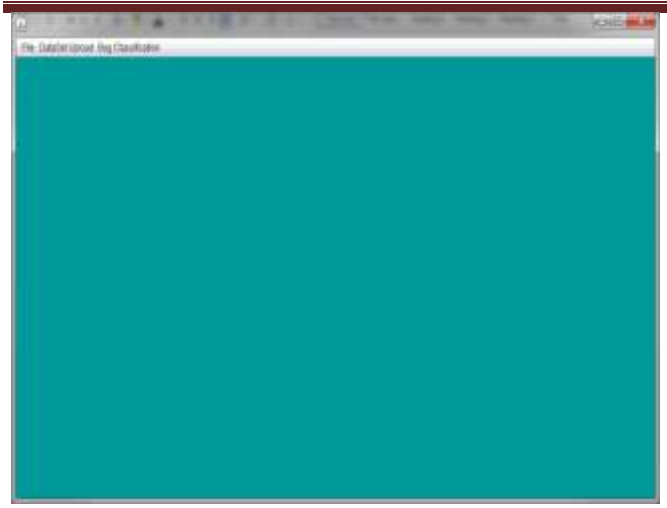

Figure 4: Main Page

Figure 5: User Registration

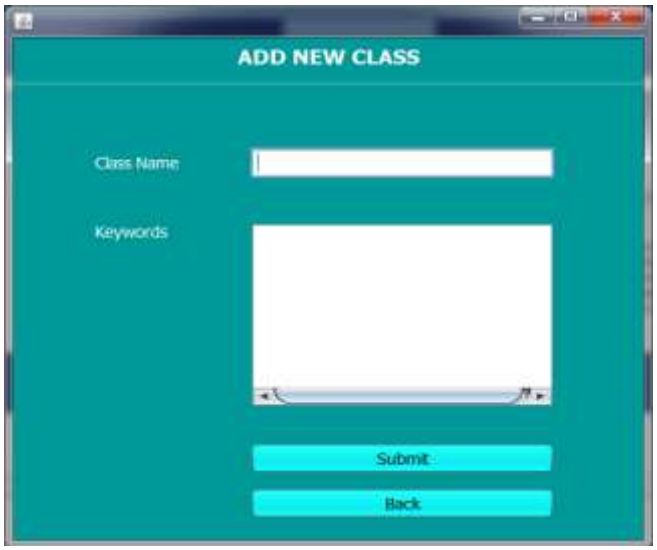

Figure 6: Class Insertion

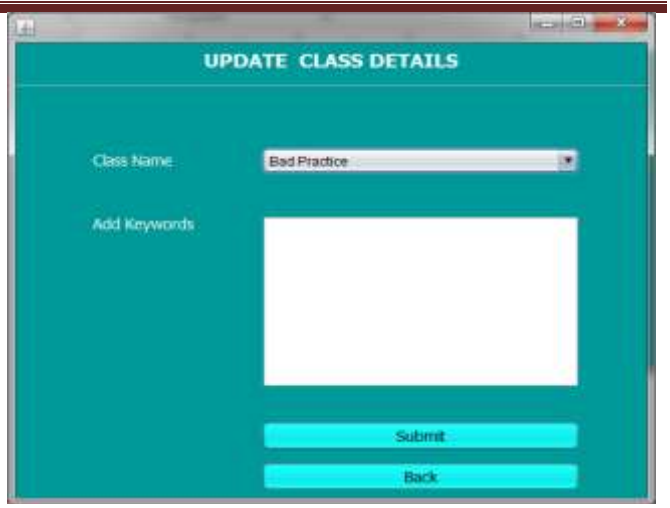

Figure 7: Update Class Detail

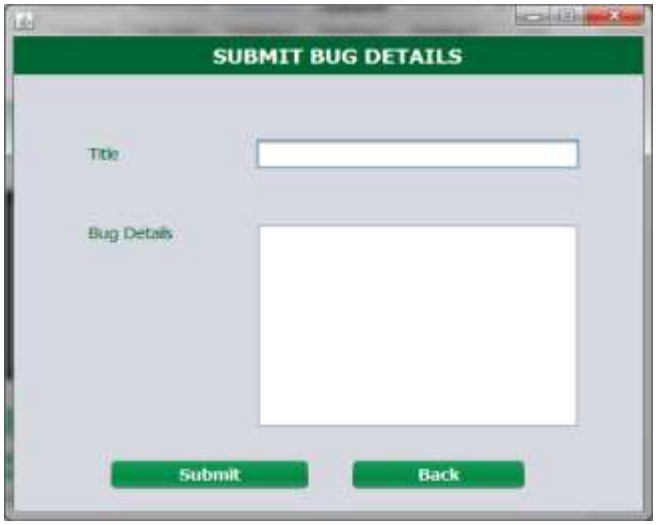

Figure 8: Bug Report Submission

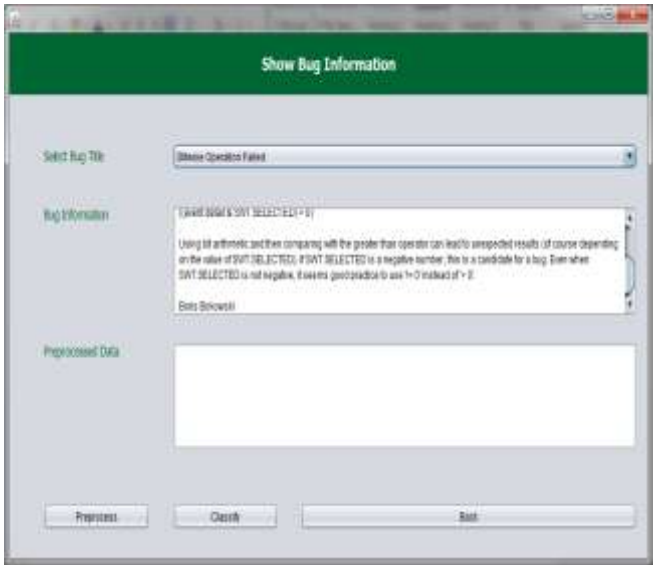

Figure 9: Bug Information Page

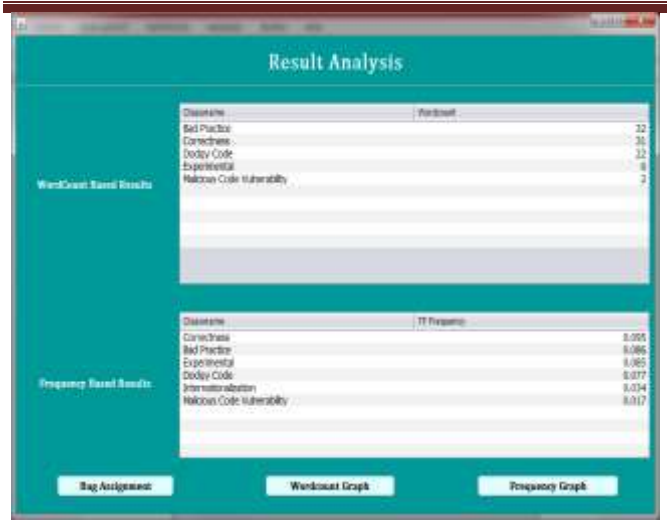

Figure 10: Result Analysis

Figure 11: Bug Allotment

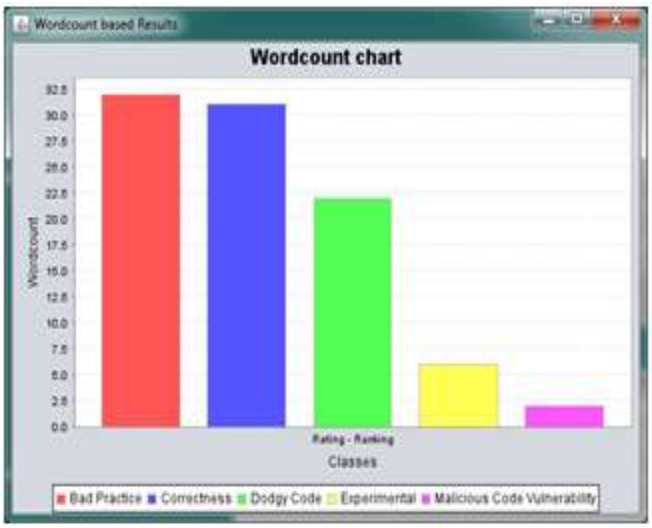

Figure 12: Word count Chart

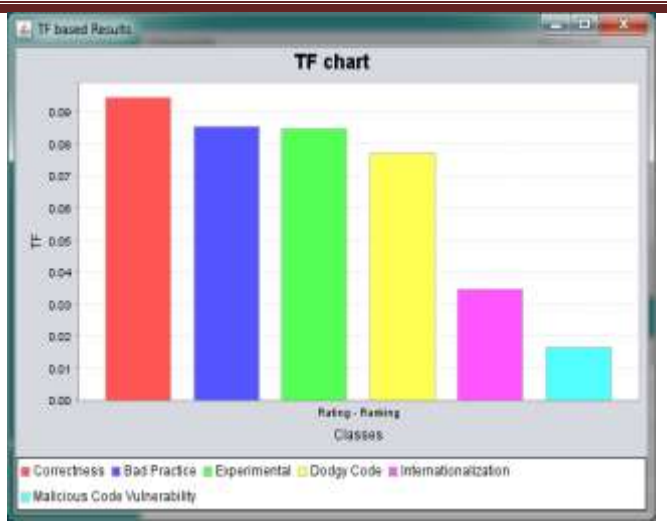

Figure 13: Term Frequency (TF) Chart

For the result analysis, parameter based comparative analysis of existing system with proposes system is done. For this purpose, three versions of C4.5 algorithms are considered & their observations are taken from previous work and then compared with the proposed algorithms.

Table of analysis for proposed system

Dataset	Algorithm	Precision	Recall	F1	Accuracy
Eclipse	C4.5	84.9	94.9	89.62 %	81%
	AdaBoost C4.5 resampling	85.0	88.6	86.76 %	77%
	AdaBoost C4.5 reweighting	85.3	88.3	85.8%	75%
FindBugs	KNN	72	84	77.5%	73%
	NB	80	88	83.8%	84%
	Hybrid	92	96	93.9%	95%

.

Table 1: Parameter based comparative result analysis for proposed system

Bug triage and classification systems can be effectively analyzed by using precision, recall & F1 measures parameters which are follows.

Precision = # of appropriate recommendations / # of recommendations made (1)

Recall = # of appropriate recommendations / # of possibly relevant developers (2)

F1 measure = 2*Recall*Precision / Recall + Precision (3)

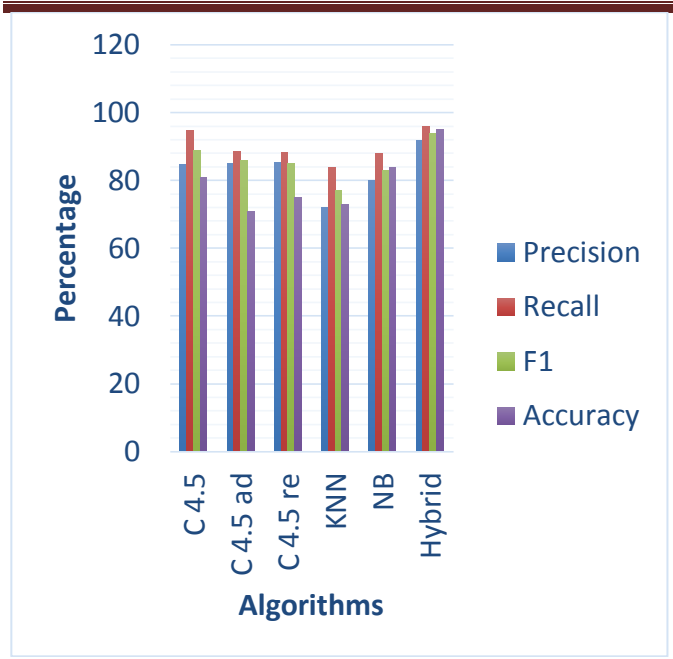

Figure 14: Comparative result analysis of different algorithm on bug triage system

Figure 14 and table 1 shows parameter based Comparative analysis of 3 different algorithms from previous system against 3 proposed algorithms on bug triage system. The table contains the results of experimental execution of system designed by hybrid combination of KNN & NB Classifier. The parameters considered for concern result analysis are precision, recall, F1 and accuracy. ECLIPSE and Find bugs are the dataset of previous work and proposed work, respectively. From the Figure 6.14 and table 6.1, it is concluded that the proposed algorithms have better results as compared to the existing algorithms of previous work.

9. CONCLUSION

Existing bug trailing frameworks don't successfully aggregate the greater part of the learning required by engineers. While not this data engineers can't resolve bugs in an exceedingly convenient manner and afterward we tend to trust that upgrades to the technique issue trailing frameworks gather information are required. We condensed criteria that are utilized in electronic gear bug trailing frameworks. Such criteria for the most part doesn't give adequate winds up in depicting bug. In this way, we tend to anticipate an enhanced arrangement of criteria which will give significantly all the more fulfilling determination to the present framework. This work are regularly crucial to the planners of the more extended term bug and abscond trailing frameworks. They should get a handle on significance of decision criteria for depicting bug, as an aftereffect of a well outline bug are simpler to be follow and illuminated.

In Future, the potency of the projected system is tested on completely different completely different dataset for checking the potency of the algorithms on different systems. We will conjointly merge some additional rules to improvise the potency of the projected Hybrid algorithm.

10. REFERENCES

[1] JifengXuan, He Jiang, Yan Hu, ZhileiRen, WeiqinZou, ZhongxuanLuo, and Xindong Wu, ―Towards Effective Bug Triage with Software Data Reduction Techniques,‖ IEEE Transactions, Volume 27, NO. 1, JANUARY 2015.

[2] Anjali, Sandeep Kumar Singh, ―Bug Triaging: Profile Oriented Developer Recommendation‖, International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2163, Volume 2, Issue 1, January 2015.

[3] S. Artzi, A. Kie_zun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M. D.Ernst, "Finding bugs in web applications using dynamic test generation and explicit-state model checking," IEEE Softw., vol. 36, no. 4, pp. 474–494, Jul./Aug. 2010.

[4] PankajRana, Asst. Prof. Saurabh Sharma "Review of Bug Serverity Prediction Techniques Using Data Mining"Volume 5, Issue 6,June2015 .

[5] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in Proc. 28th Int. Conf. Softw. Eng., May 2006, pp. 361–370.

[6] Naresh Kumar Nagwani, ShrishVerma, "CLUBAS: An Algorithm and Java Based Tool for Software Bug Classification Using Bug Attributes Similarities," Journal of Software Engineering and Applications, 2012, 5, 436-447,May 10th, 2012.

[7] P. S. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree-based k-means clustering algorithm," IEEE Trans. Knowl. Data Eng., vol. 24, no. 6, pp. 1146–1150, Jun. 2012.

[8] PankajGakare, YogitaDhole,SaraAnjum, ―Bug Triage with Bug Data Reduction‖, International Research Journal ofEngineering and Technology (IRJET) e-ISSN: 2395 - 0056,Volume 02 Issue 04, July 2015.

[9] C. C. Aggarwal and P. Zhao, "Towards graphical models for text processing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.

[10] D. Cubranic and G. C. Murphy, "Automatic bug triage using text categorization," in Proc. 16th Int. Conf. Softw. Eng. Knowl. Eng., Jun. 2004, pp. 92–97.

[11] J. Anvik and G. C. Murphy, ―Reducing the effort of bug report triage: Recommenders for development-oriented decisions,‖ ACM Trans. Softw. Eng. Methodol., vol. 20, no. 3, pp. 10:1–10:35, Aug. 2011.

[12] A. K. Farahat, A. Ghodsi, M. S. Kamel, "Efficient greedy feature selection for unsupervised learning," Knowl. Inform. Syst., vol. 35, no. 2, pp. 285–310, May 2013.

[13] K. Gao, T. M. Khoshgoftaar, and A. Napolitano, "Impact of data sampling on stability of feature selection for software measurement data," in Proc. 23rd IEEE Int. Conf. Tools Artif. Intell., Nov. 2011, pp. 1004–1011.

[14] J. A. Olvera-Lopez, J. A.Carrasco-Ochoa, J. F. Martınez-Trinidad, and J. Kittler, "A review of instance selection methods," Artif. Intell. Rev., vol. 34, no. 2, pp. 133–143, 2010.

[15] C. Sun, D. Lo, S. C. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports," in Proc. 26th IEEE/ACM Int. Conf. Automated Softw. Eng., 2011, pp. 253–262.

[16] G.Jeong S. Kim, and T. Zimmermann, ―Improving bug triage with bug tossing graphs‖, in Proceedings of Seventh

joint meeting ofEuropean Software Engineering Conference & ACM SIGSOFT symposium on Foundations of software engineering, ser. ESEC/FSE '09. New York, NY, USA: ACM, pp. 111–120, 2009.

[17] R. S. Pressman, ―Software engineering: A Practitioner's Approach‖, 7th ed. New York, NY, USA: McGraw-Hill, 2010.

[18] S. Shivaji, E. J. Whitehead, Jr., R. Akella, and S. Kim, "Reducing features to improve code change based bug prediction," IEEE Trans. Soft. Eng., vol. 39, no. 4, pp. 552–569, Apr. 2013

[19] T. Zimmermann, N. Nagappan, P. J. Guo, and B. Murphy, "Characterizing and predicting which bugs get reopened," in Proc. 34th Int. Conf. Softw. Eng., Jun. 2012, pp. 1074–1083.

[20] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schr€oter, and C. Weiss, "What makes a good bug report?" IEEE Trans. Softw. Eng., vol. 36, no. 5, pp. 618–643, Oct. 2010.

[21] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," Knowl. Inform. Syst., vol. 35, no. 2, pp. 249–283, 2013.

[22] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," J. Mach. Learn. Res., vol. 3, pp. 1157–1182, 2003.

[23] A. Srisawat, T. Phienthrakul, and B. Kijsirikul, "SV-kNNC: An algorithm for improving the efficiency of k-nearest neighbor," in Proc. 9th Pacific Rim Int. Conf. Artif. Intell., Aug. 2006, pp. 975–979.

[24] J. Tang, J. Zhang, R. Jin, Z. Yang, K. Cai, L. Zhang, and Z. Su, "Topic level expertise search over heterogeneous networks," Mach. Learn., vol. 82, no. 2, pp. 211–237, Feb. 2011.

[25] I. H. Witten, E. Frank, and M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd ed. Burlington, MA, USA: Morgan Kaufmann, 2011.

[26] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," Mach. Learn., vol. 38,pp. 257–286, 2000.

[27] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and execution information," in Proc. 30th Int. Conf. Softw. Eng., May 2008,
pp. 461–470.

[28] J. Xuan, H. Jiang, Z. Ren, and Z. Luo, "Solving the large scale next release problem with a backbone based multilevel algorithm," IEEE Trans. Softw. Eng., vol. 38, no. 5, pp. 1195–1212, Sept./Oct.
2012.

[29] J. Xuan, H. Jiang, Z. Ren, J. Yan, and Z. Luo, "Automatic bug triage using semi-supervised text classification," in Proc. 22nd Int. Conf. Softw. Eng. Knowl. Eng., Jul. 2010, pp. 209–214.

[30] J. Xuan, H. Jiang, Z. Ren, and W. Zou, "Developer prioritization in bug repositories," in Proc. 34th Int. Conf. Softw. Eng., 2012, pp. 25–35.