Improved Framework for Bug Severity Classification using N-gram Features with Convolution Neural Network

Sarbjeet Kaur, Maitreyee Dutta

Abstract*: Foreseeing the seriousness/severity of bugs has been established in former research study in order to recover triaging and the process of bug resolution. Therefore, numerous prediction/classification methodologies were developed throughout the years to give an automated reasoning over the seriousness classes. Seriousness or severity is a significant trait of a bug that chooses how rapidly it ought to be measured. It causes designers to comprehend significant bugs on schedule. Though, manual evaluation of severity is a dreary activity and could be off base. This paper comprises of using the text/content mining together along with the use feature selection and bi-grams to improve the order of bugs in six classes. In the proposed methodology the features are refined by the use of convolution layers. Here, the process of convolution-based refining indicates mapping of the features utilizing non-linear methods of all the classes as compared to the existing methodologies.

***Keywords* : Bug Severity, Bug Prediction, Bi-grams**

I. INTRODUCTION

In software development improvement different types of resources are considered for bug reporting. A Software bug can be delegated error, flaw, failure or fault in any framework because of which framework carry on in an inappropriate way, may give outcomes about which are most certainly not expected or wrong outcomes. Different courses in which a bug can emerge are either because of flaws in source code, outlining of program or because of operating systems or additionally can be delivered by compiler errors [1]. The consequences of bugs finished up to be dangerous, from different occurrences in true.

Manuscript published on 30 September 2019
* Correspondence Author
Sarbjeet Kaur*, Department of Computer Science, NITTTR, Chandigarh, India. Email: sarbjeet.cse@nitttrchd.ac.in
Maitreyee Dutta, Department of Information Management and Co-ordination Unit, Email: d_maitreyee@yahoo.co.in

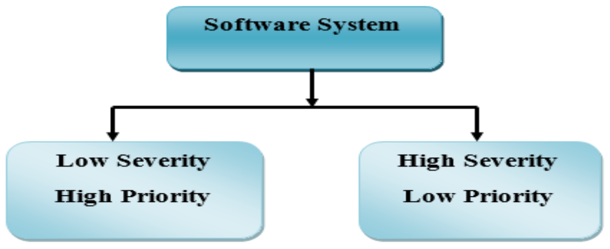

Fig. 1. Bug classifications on severity and priority basis [1]

A. Types of Bugs

For advancement of software quality it ought to be guaranteed that the bugs ought to be recognized and to be taken care in their beginning times amid software advancement [2, 11]. Software quality can be influenced because of following sorts of bugs:

1. *Arithmetic Bugs:* The bugs which are caused by infringement of arithmetic tenets. Case, isolate by zero.
2. *Syntax Bug:* The bugs which are caused by the infringement of the linguistic structure guidelines of programming dialect. Illustration, utilizing equivalent to administrator rather task administrator.
3. *Logic Bug:* The bugs which are caused by utilizing incorrectly logic and yield is not anticipated.
4. *Resource Bug:* The bugs which are caused by in suitable utilization of assets. Illustration, un instated variable.
5. *Multithreading Bug:* Example is Deadlock, for multithreading bug.

B. Bug Severity

It is the degree to which the error can influence the software. As such, it characterizes the effect that a given error has on the frame. For example: if an application or page fails when a remote connection is clicked, for this situation touching the remote connection of a client is uncommon, however, the effect of the destruction of the use is serious. Then, the severity is high but the need is low [1, 3, 4]. For the most part, the bug reporter does not know how to handle severity, so the standard is assigned in the severity value and discussed below.

1. *Critical:* The bug that outcomes in the end of the entire framework or at least one part of the framework and causes broad defilement of the information.

The fizzled work is unusable and there is no worthy option strategy to accomplish the required outcomes then the severity will be expressed as critical.

2. *Major:* The bug affects the main functionality or the main data. It has an answer, yet it isn't clear and it is troublesome. For instance, a component isn't useful from of a module yet the undertaking is possible if 10 indirect complicated steps are followed in another module/s.

3. *Moderate:* The bug that does not bring about the end, but rather makes the framework deliver mistaken, fragmented or conflicting outcomes then the severity will be expressed as moderate.

4. *Minor:* The bug that does not cause the end and does not harm the convenience of the frame and the coveted results can be acquired without effort when solving the bugs and then the severity is expressed as minor [5-7].

5. *Cosmetic:* The bug that is identified with the improvement of the framework where the progressions are identified with the look and feel of the application then the severity is expressed as cosmetic.

Severity is also denoted as at different levels shown in figure.2.

- If severity is **S1** that it is Critical
- If severity is **S2** that it is Major
- If severity is **S3** that it is Minor
- If severity is **S4** that it is Trivial

S1
Critical
S2
Major
S3
Moderate
S4
Trivial

Fig. 2. Bug Severity Level [2]

C. Bug Report Format

The below given table represents the contents of the bug reports. The fields and their description show their value. These content gives the information related to the bug in detail.

Table 1: Bug Report Format

Field	Description
Summ	Summary: This field contains a short description and contains only few words.
Desc	Description: This field contains a detailed explaination of the bug and information like how bug is reproduce and error in the output when the bug will take place.
Prod	Product: It is the product i.e. bug affected.
Comp	Component: It is the component affected by bug.
Sev	Severity: It is basically the level which is assigned to the bug according to its impact on the software. The severity level for the bugs are following: • Critical • Major • Moderate • Trivial

Severity of bug is an important factor in deciding the priority of the bug because the number of bugs is usually high. Bug is basically a description in which software engineers mention the position of fault in the software system [8-10]. Nowadays different types of the bug tracking system are used to encounter the bug by using Jira and Bugzilla bug tracker. While reporting the bug user also mention the description related to the bug by filling the form, it helps the development team resolve the reported bug. The below given figure represents the main function of severity shown in figure.3.

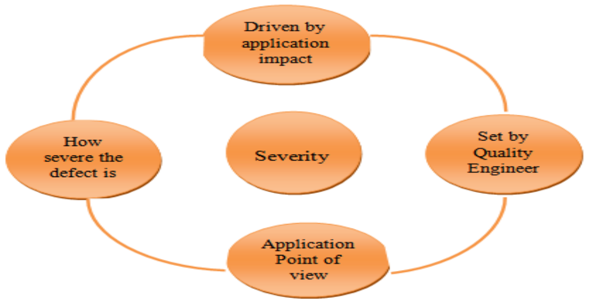

Fig. 3. Severity and its functions [4]

D. Different Bug Prediction Algorithm

The huge number of techniques and algorithms are developed for the bug severity prediction by the researchers but there are also some issues related to them that are focused for the future research. Here some to techniques are discussed in brief that are mostly used by the researchers in traditional approaches according to their data and bug reports.

- ***Naïve Bayes Classification***

It is a classifier which gauges the likelihood of contingent class by utilizing presumption that every one of the qualities are restrictively autonomous. In such type of classifier, contingent likelihood is determined for each term of the class. After the estimation of probabilities another classification report is produced for each class of each term happened in the archive.

- ***K-nearest neighbour***

The algorithm based on k-NN contrast each report in dataset placed within the provided report and after that discover the comparability in the current report. The report-based similarity is estimated by utilizing any distance or separation measure. The classification of data point depends on the class labels and neighbors. Regardless, information point have more than one mark new report will be allocated to most of the holdup class. In this type of algorithm, k is usually presents the neighboring points for instance k=4 and then it focusses over the fact that k has four closest points and it discover most of the class.

- ***Naïve Bayes Multinomial***

The multinomial Naïve Bayes algorithm is enhanced version of nave Bayes and it considers the weight during the calculation of conditional probability. This algorithm also provides the length of document and terms occurrence in the different documents. For this purpose TF-IDF measure is used that is term frequency-inverse document frequency.

- ***Support Vector Machine***

SVM is a statistical and mathematical technique and gives the effective result on the field of recognition and classification of objects.

This techniques also gives effective result in the field of text mining and avoid the problem of dimensionality because it works on high dimensions of data [12, 13]. The report sets are separated in the given categories which depends upon the hyperplane having maximum margin. The distance measured among the hyper planes are maximized to separate the testimony established into binary kind of classes.

E. Pros and Cons of Bug Severity Prediction

1 Pros

- Helps to find the bugs and helps to fixed them
- It allow repository of documents which helps the trouble shooter later in related issue.
- Helps in early detection of bugs which reduces the failure rate of the software

2 Cons

- Some associations utilize numerous apparatuses to track deformities of various kinds, and those instruments regularly don't coordinate well with each other.
- Bug opening procedure is a convoluted assignment and devours additional time.
- Complications can emerge out of perplexity over depictions, absence of data, devices that are excessively awkward and require compulsory fields for which the client doesn't have the appropriate responses, and trouble in revealing.

II. RELATED WORK

Jindal, Rajni et al. [1] presented a defect severity prediction model by minig the software reports. This model has ability to predict the severity level of the bug in the bug reports. The defects reports used in this are from the PITS dataset. This data set is popular because it is used by the NASA. Data mining approach is involved to extract the data from the reports and then used by the prediction model. The concept of logistic regression, multi-layer perceptron and decision tree is used in this model. The outcomes of the decision tree is best among all when it is compared in analysis process.

Heena Singla et al [2]: proposed the bug tracking system which identifies the bugs. Physical method is different from software which inputs are accepted and outputs are generated. Any change in the software is usually requested while developing any software system. Mainly, the request is related to software maintenance. The cost of the software is usually spent on software maintenance. The error tracking system (BTS) is manually assigned to a respected developer for fixation. The main reason for errors is to identify errors that require instantaneous concentration. The user reports the errors and assigns their priority to the importance of the error. However, the field level may not be assigned correctly by the user because its assumption regarding the importance of an error may vary from another user.

Kwanghue Jin et al [3]: proposed a new programming frameworks which has been produced constantly, and utilized as a part of a multiplicity of fields. Along these lines, bugs ought to be settled accurately to their seriousness levels since they have isolate seriousness levels. Subsequently, rectify expectation of bug seriousness is required for effective programming improvement and support that designers can know which bug requires to be settled promptly. Prior investigations utilized just content data of the bug report for their forecast systems. In this manner, so as to build up the bug settling, initialize a solid procedure of bug seriousness expectation that additional the Description and Reporter fields since bug columnists record content qualities of the report.

Zhou, Yu, et al. [4] proposed a multi-association approach by consolidating content extraction and data mining techniques to modernize the determining procedure. The primary stage uses content-based extraction procedures to disect the principle bug report parts and request them as indicated by three degrees of likelihood. Features and some different features of the bug reports are then consolidated into the understudy machine during the subsequent stage. Data Joining Strategies are utilized to interface the two stages. Relative testing with past surveys of comparable data - three enormous scale open source ventures - dependably accomplish a decent redesign, have accomplished their best outcomes in terms of its execution. Extra observational audits of seven other understood open source systems affirm the discoveries.

Hans Hansson et al [5]: presented multicore and other parallel designs, there is an expanded necessity for adequate and viable treatment of programming executing on such structures. A huge angle in this setting is to comprehend the bugs that happen because of parallel and simultaneous execution of programming. Testing and troubleshooting simultaneous programming are looked with the different difficulties. Creating simultaneous programming need designers to monitor all the plausible correspondence designs, which create from an extensive number of likely interleaving or simultaneously covering executions that can happen among particular execution strings through using the common memory.

Fairuz Amakina Narudin et.al. [6] proposed an alternative response for surveying bug discovery using the irregularity based approach with machine learning classifiers. Among the various framework action incorporates, the four courses of action picked are principal information, content-based, affiliation based, and time-based. The methodology utilizes two datasets: private (self-aggregated) and open (MalGenome). In light of the assessment happens as expected, both the Bayes framework and unpredictable forest classifiers passed on progressively exact readings, with a 99.97% authentic positive rate rather than the multi-layer perceptron with just 93.03% on the dataset named MalGenome dataset.

Mohd Faizal Ab Razab et.al. [7] intended to fill in that hole by exhibiting a thorough assessment of bug explore hones. It starts by taking a gander at a pool of more than 4000 articles that are distributed in the vicinity of 2005 and 2015 in the ISI Web of Science database. Utilizing bibliometric analysis, this paper examines the examination exercises done in North America, Asia, and different landmasses. This paper played out a point by point analysis by taking a gander at the quantity of articles distributed, references, inquire about the region, catchphrases, establishments, terms, and creators.

A synopsis of the examination exercises proceeds by posting the terms into an order of bug location framework which underlines the essential range of bug explore. From the analysis, it was inferred that there are a few huge effects of research exercises in Asia, in contrast with different main lands. Specifically, this paper talks about the quantity of papers distributed by Asian nations.

Kwanghue Jin et al [8]: created new programming frameworks ceaselessly, and are utilized as a part of a multiplicity of fields. Along these lines, bugs ought to be settled effectively to their seriousness levels since they have isolate seriousness levels. Henceforth, revise forecast of bug seriousness is required for effective programming advancement and support that engineers can know which bug requires to be settled instantly. Prior examinations utilized just content data of the bug report for their expectation systems. Consequently, to build up the bug settling, they initiate a dependable procedure of bug seriousness forecast that additional the Description and Reporter fields since bug correspondents record content qualities (Summary and Description) of the report.

Fabio Palomba et.al. [9] discovered about the propensity to bug to construct a particular bug prediction that is demonstrated for range classes. In particular, they evaluate the commitment of a measure of the severity of the odors of the code (ie, the intensity of the code warning) by adding it to existing bug prediction models and observing the consequences of the new model against the pattern shown. The results show that the accuracy of an error prediction display increases by including the code warning intensity as an indicator. They also evaluate the actual pick-up given by the intensity list in terms of alternative measures in the model, counting those used to record the intensity of the code warning. They see that the intensity list is much more vital when compared to the different measurements used to anticipate the buggies of the doubtful classes.

Zhou, Yu, et al. [10] proposed a multi-organize approach by joining both content mining and information mining procedures to computerize the forecast procedure. The main stage use content mining systems to dissect the outline parts of bug reports and arranges them into three levels of likelihood. The extricated highlights and some other organized highlights of bug reports are then sustained into the machine student in the 2nd stage. Information joining methods are utilized to connect the two phases. Relative tests with past investigations on similar information—three substantial scale open-source ventures—reliably accomplish a sensible improvement finished their best outcomes as far as by and large execution. Extra near observational analyses on other seven mainstream open-source frameworks affirm the discoveries. Besides, in view of the information acquired, they likewise observationally examined the effect connections between the fundamental classifiers and different properties of the joined model. A prototype recommender framework has been produced to show the appropriateness of their approach.

Yuan Tian et al [11]: presented the concept of multi-factor analysis for bug report prediction. This work is based on the machine learning and factors like temporal, textual, author, and bug reports. These factors are considered as feature and used for the training of discriminative model using the classification algorithm. This classification algorithm handles the imbalanced data and ordinal class. The proposed model improves the f-measure in the outcomes. *Xiaohu Yang et al [12]:* Software bugs are regular in all phases of the product advancement and upkeep lifecycle. To dealing with the report of programming bugs, every one of the engineers utilized following framework in programming bugs. In light of the most vital of programming bugs, countless systems have been intended to oversee and decrease the effect of programming bugs. These systems incorporate bug triaging and designer suggestion bugs need. In regular bug settling strategy, an analyzer or a client distinguishes a bug and presents a bug answer to clarify the bug in bug following frameworks. At that point, the bug is allocated to a comparing designer to settle. On the off chance that the bug is settled once, the second engineer would affirm the fixes, and finally shut the bug report. Consequently, in specific cases, the whole settling technique is slowed down because of the presence of a blocking bug task, copy bug report discovery, bug settling time forecast, and revived bug expectation.

Gitika Sharma et al [13]: Software is affecting gigantic human exercises and its utilization is ascending at an exceptional rate. On account of increment popular and diminished in conveyance time giving the surety of value while diminishing conveyance time is ends up basic. Subsequently, to guarantee the nature of programming, different testing strategies are utilized. The product bugs that are recognized after the sending of programming influence consistency and nature of the product. Bug following frameworks (BTS) enable clients and in addition designers to report these bugs to programming. The detailed bugs in BTS are analyzed by Triager to process their legitimacy, exactness, importance, seriousness and furthermore to affirm its trickery and subsequently are doled out to the pertinent designer to determine it. Triager is the individual who utilizes his insight and experience to assess and refine the bugs that are accounted for.

Jacek Ratzinger et al. [14] Solved the impact of advanced exercises such as refactoring on software bugs. For a situation investigation of five open source ventures, we utilized characteristics of software advancement to foresee bugs in eras of a half year. They utilize forming and issue tracking systems to separate 110 information mining highlights, which are isolated into refactoring and non-refactoring related components. These components are utilized as a contribution to arrangement calculations that make prediction models for software bugs.

III. THE PROPOSED METHOD

A. Proposed Framework

Step 1: Input bug text. This bug test are the bug reports that are used as input in this methodology. These reports contains the list of bugs are their cause of occurrence in the previous system.

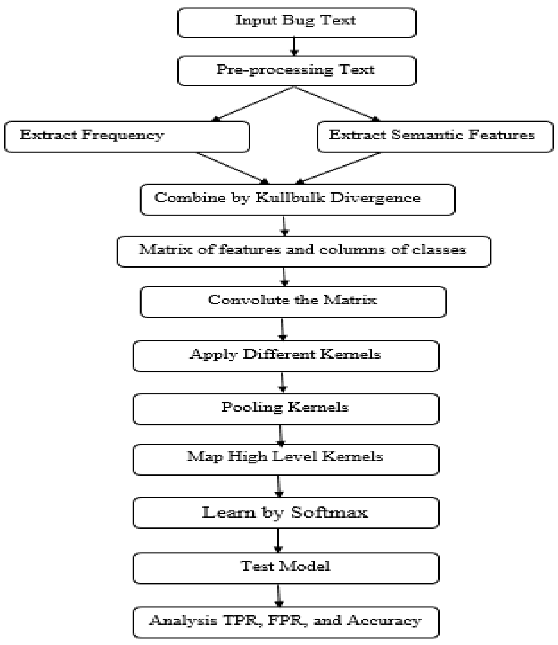

Fig. 4. Proposed Framework

Step 2: Pre-processing of bug reports. Pre-processing is a process in which duplicate data is removed by using the process of stemming and tokenization. In these processes stop words removal, has tag removal, and repeated word removal are performed.

Step 3: The third step followed in this methodology is extraction of features in two types that are Frequency Features and Semantic features.

Step 4: Combines the features by Kullbulk Divergence. Basically this probability function is used to measure the probability of feature set one is different from the second one.

B. Proposed Methodology

Implementation is the process that turn strategies and plans into actions in order to accomplish objective and goals. Following are steps which are used in the implementation bug severity prediction.

Step1: In the first step, the Bug severity dataset is divided into two parts. One is the testing phase and the other is the phase of training. Further, the training Bugs performs the operation of preprocessing and changes into gray scaled bug in order to finding the values of pixel.

Step 2: In the second step, the process begins with initialization of the convolutional part with convolution window size 3*3 convolution window that mixes the pixels continuously and further uses stride base convolution padded by a zero in case of non-availability of any pixel.

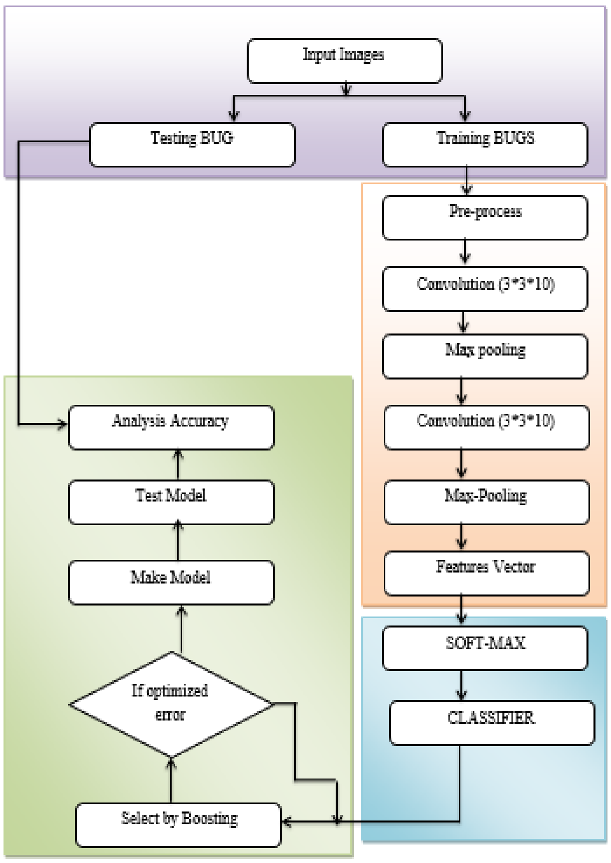

Fig. 4. Proposed Flowchart

Step3: In case of third step, pooling of the kernel function is done as defined in a convolution layer. Out of possible 512 functions, 10 functions are defined in convolution layer. These 10 type of function results in 10 different answers out of which the maximum value is selected by using max-pooling layer.

Step4: After the process of max-pooling, resultant matrix further applied with convolution-based window size 3*3 and 10 kernel function. Thereby, this process results in a feature vector. Feature vector and class of Bugs basically learn the process by using Softmax.

IV. RESULT ANALYSIS

In the experimental setup, a dataset of (Zhou, Yu, Yanxiang Tong, Ruihang Gu & Harald Gall, 2016) was used with various classes shown in Table 2. The dataset consists of bug reports from five different open recourse's that are Eclipse, Mozilla, JBoss, Firefox and OpenFOAM.

Table2.Comparison of Proposed and Existing

Projects	Existing SVM Technique[4]				Proposed Work (CNN)			
	Precision (%)	Recall (%)	F-measure (%)	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)	Accuracy (%)
Mozilla	82.6	82.4	81.7	83.45	98.48333	98.51667	98.11667	97.687
Eclipse	81.8	82.1	81.6	80.56	99.38333	99.48333	99.11667	98.8883333
JBoss	93.7	93.7	93.7	82.2	98.88333	98.95	98.41667	99.155625
OpenFOAM	85.3	85.3	84.7	81.34	95.25	95.35	94.55	97.857619
Firefox	80.3	80.5	79.5	80.34	92.75	92.95	91.95	96.234

The results were evaluated by using existing SVM and proposed feature weighting CNN approach with bi-gram and TF-IDF applications using K-L Divergence. These bug reports were cut across into the testing and training dataset based on the 10-cross-validation mechanism. Based on the above methodology the series of experiments were performed to analyze the approach using evaluation metrics like Recall, Precision, F-measure and Accuracy are used. The different values were tested for CNN parameters. The result demonstrates that the best execution is accomplished by setting the parameters to values appeared in (Table 2).

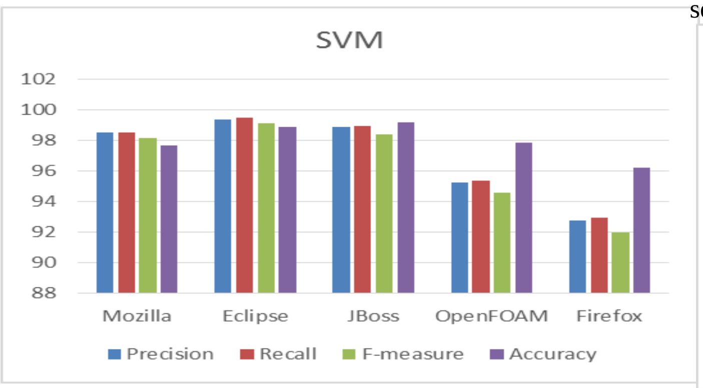

Fig 5: Comparison of Proposed (CNN) and Existing(SVM) in different dataset

Initially, the proposed CNN was applied on bug reports of five projects shown in (Table 2). The basic aim of the proposed algorithm is to improve the performance of existing SVM by Convolution the features of a dataset.

Figure 4.6 shows the comarision of precision on the proposed and the exisiting work on different opensource datasets. It can be noticed that the CNN approach give us less loss than the SVM .

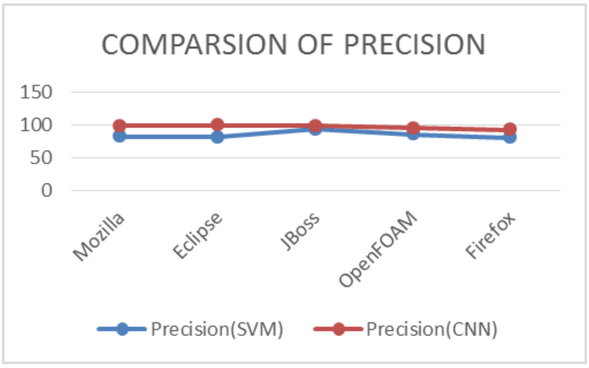

Fig 6: Comparison of Proposed (CNN) and Existing (SVM) in different dataset of precision

Fig 7: shows the comparison of Recall on the proposed work and the existing work. In case of Recall CNN shows the high results. JBoss using CNN show again high performance but others also increase there bug prediction.

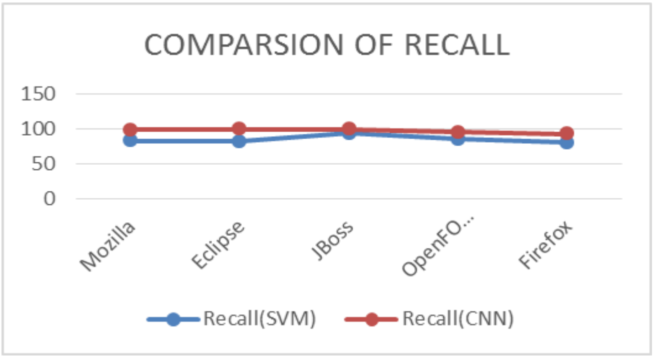

Fig 7: Comparison of Proposed (CNN) and Existing (SVM) in different dataset of Recall

In this experiment, the existing SVM was applied on bug reports of five projects as shown in (Table 2). The NB was directly applied to the whole feature space. The results of CNN and SVM were compared. The proposed approach was performed very well in classifying the bugs of seven severity classes of Mozilla and Firefox dataset

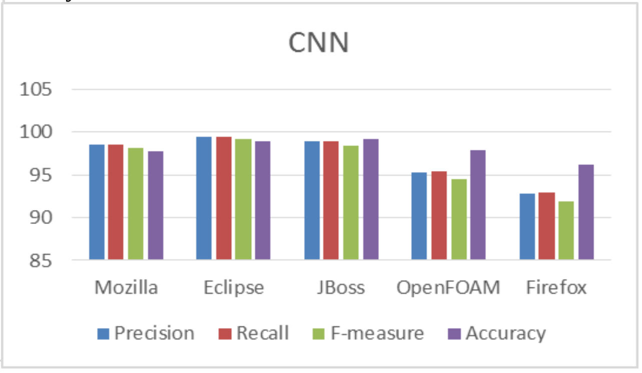

Fig 8: Comparison of Proposed (CNN) different dataset

It can be seen the accuracy of proposed CNN an existing SVM on the Blocker, Critical, Enhancement, Major, Normal, Minor and Trivial classes vary between 96% to 99% ,92% to 98%for accuracy and precision Firefox dataset.

The descriptive result shows that the CNN approach is more accurate and more effective than the SVM. According to the purposed objective, this experiment has been performed using the multiclass classifier, which accurate bug prediction. This experiment is doing by combining semantic features and frequency features. At the end in this experiment comparative analysis is done between the purposed model and existing model, purposed model accuracy better than the existing model.

V. CONCLUSION

Lastly, it can be concluded the Bug Report Severity Classification is an important task in testing and maintenance phase of the software development lifecycle.

It is a challenging task because of multiclass classification. If bugs are classified incorrectly, then it will induce a delay in the system as bugs with high priority will not be dealt at the right time. This task done manually is prone to errors, thus there is a need for automatic classification of bugs to help the triager. This paper proposed an automatic classifier of bugs using bi-gram and TF-IDF approach to extract the features. Later, the FW method was used to assign relative weights which were optimized using ACO. The ACO was used to identify important features for reducing the overlapping. Furthermore, the classification is done by the generative ML model NB and discriminative ML model Support Vector Machine. The experiments were conducted on five datasets which are Eclipse, Mozilla, JBoss, Firefox and Open FOAM. The results were later compared with the existing CNN and SVM. The existing CNN and SVM approach has an accuracy ranging from 90 to 96% and 92 to 99%, while the proposed methodology, i.e., CNN and SVM with ACO varied from 92 to 98% and 90 to 96%. The accuracy of the proposed model was higher than the existing model. This proves the proposed automatic classifier performed better than existing ones by optimizing the weights of features.

REFERENCES

1. Jindal, Rajni, RuchikaMalhotra, and Abha Jain. "Prediction of defect severity by mining software project reports." *International Journal of System Assurance Engineering and Management* 8.2 (2017): 334-351.
2. Singla, Heena, Gitika Sharma, and Sumit Sharma. "Domain Specific Automated Triaging System for Bug Classification." *Indian Journal of Science and Technology* 9.33 (2016).
3. Jin, Kwanghue, et al. "Bug Severity Prediction by Classifying Normal Bugs with Text and Meta-Field Information." (2016).
4. Zhou, Yu, et al. "Combining text mining and data mining for bug report classification." *Journal of Software: Evolution and Process* (2016).
5. AbbaspourAsadollah, Sara, Daniel Sundmark, Sigrid Eldh, Hans Hansson, and Eduard Paul Enoiu. "A Study on Concurrency Bugs in an Open Source Software." In *The 12th International Conference on Open Source Systems (OSS), 30 May-2 June 2016, Gothenburg, Sweden*. 2016.
6. Narudin, FairuzAmalina, et al. "Evaluation of machine learning classifiers for mobile bug detection." *Soft Computing* 20.1 (2016): 343-357.
7. AbRazak, MohdFaizal, et al. "The rise of "bug": Bibliometric analysis of bug study." *Journal of Network and Computer Applications* 75 (2016): 58-76.
8. Jin, Kwanghue, et al. "Bug Severity Prediction by Classifying Normal Bugs with Text and Meta-Field Information." *Advanced Science and Technology Letters* 129 (2016): 19-24.
9. Palomba, Fabio, et al. "Smells like teen spirit: Improving bug prediction performance using the intensity of code smells." *Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on*.IEEE, 2016.
10. Zhou, Yu, et al. "Combining text mining and data mining for bug report classification." *Journal of Software: Evolution and Process* 28.3 (2016): 150-176.
11. Tian, Yuan, et al. "Automated prediction of bug report priority using multi-factor analysis." *Empirical Software Engineering* 20.5 (2015).
12. Xia, Xin, et al. "Elblocker: Predicting blocking bugs with ensemble imbalance learning." *Information and Software Technology* 61 (2015)
13. Sharma, Gitika, Sumit Sharma, and ShrutiGujral. "A Novel Way of Assessing Software Bug Severity Using Dictionary of Critical Terms." *Procedia Computer Science* 70 (2015): 632-639.
14. Ratzinger, Jacek, Thomas Sigmund, and Harald C. Gall. "On the relation of refactorings and software bug prediction." *International working conference on Mining software repositories*. ACM, 2008.

AUTHORS PROFILE

Sarbjeet Kaur, M.E Scholar, Department of Computer Science, NITTTR Chandigarh, B.E from Sant Longowal Institute of Engineering and Technology, Sangrur, Punjab

Dr. Maitreyee Dutta Ph.D. (Engg. & Tech.) from Panjab University, M. Tech(ECE) from Panjab university, B.E(ECE) from Guwahati University.