IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

APPROVAL PREDICTION FROM IMPROVEMENT REQUESTS BASED ON SENTIMENT ANALYSIS

Kavita Goura, Shivakumar Dasari

Assistant Professor, Student(M.Tech)
Computer Science and Engineering,
Chaitanya Bharathi Institute of Technology, Hyderabad, India

Abstract: Even after delivering the product by implementing all the features as per customers requirements, the improvement or enhancement requests are frequently proposed by the users. For a product if more numbers of stakeholders are there, then the improvement requests are more in number and also frequency of receiving requests is high. Though the received requests are high in number, all are not suitable to implement. It is very time consuming for a developer to manually go through each and every request and to differentiate them between implementable and un-implementable requests. So, to overcome this problem, sentiment-based analysis can be used to predict the most likely requests which will be approved for implementation. This approach makes the software applications competent in the market by helping in implementing the features at a rapid rate and thus meeting the deadlines. In this approach, the first step is to pre-process improvement requests using natural language pre-processing techniques. Second is to calculate the sentiment of each improvement request by identifying the words having positive and negative sentiments in the summary attribute of the data collected. Third is to train the machine learning based classifier to predict whether a given improvement request would be approved. The proposed approach is evaluated with the sample history data from real software applications by using the algorithms SVM (Support Vector Machine), Naïve Bayes, Logistic Regression and Random Forest. A comparison among these algorithms is done to decide on the best algorithm for the kind of the data considered.

Index Terms - **Improvement requests, Sentiment Analysis, SVM, Random Forest, Naïve Bayes, Logistic Regression**.

1. INTRODUCTION

All the software applications related to desktop, laptop, mobiles and other electronic gadgets have become very dynamic in nature. Many requests are received to upgrade each of these applications. Unless and until they are upgraded, their existence in the market is difficult. The number of improvement requests may be large for software applications for which the number of users are high. The management of these requests and dividing them into implementable and unimplementable is a challenging task. It is a very time consuming process for the Developers to go through each of these requests manually and resolve the requests into add on features for the applications. Among these received requests, many of them are rejected due to different reasons like for eg. Few requests are not feasible, improper or incomplete description of the request, few requests are very time consuming to implement etc. Studies have shown that up to 75% of the requests are rejected. But though, majority of the requests are rejected, developers have to go through each of these requests, to assure not missing on any of the implementable request. This manual procedure causes unnecessary wastage of time. This wastage of time can be avoided by automating the approval prediction of each of the requests. With the automated and accurate approval prediction of requests, developers may first concentrate on implementing the most likely approved requests instead of wasting time with the most likely to be rejected requests. Two advantages are achieved with this. One is, a rank based inspection may help to find large number of useful suggestions from the user requests. Another one is that developers may quickly implement the useful requests thus increasing the efficiency and reputation of the software application in the market.

In this paper, we propose Approval Prediction from improvement requests based on sentiment analysis. We used the data related to Mozilla ecosystem which was extracted by nizamani (2017)[11] were used. Each request was pre-processed. Pre-processing was done in several steps like Tokenization, spelling correction, lower case conversion, removal of punctuation marks, stop words removal, white space removal, Parts-of-speech tagging, lemmatization. In tokenization, a sequence of strings, here requests are broken into pieces of words. In spelling correction, the misspelled words are corrected. In lower case conversion, the text is converted to lower case which helps with consistency of the expected output. In removal of punctuation marks, all the punctuation marks like comma, full stop, exclamation marks are removed. Stop words are the words which are used very frequently

and they are so frequent that they lose their semantic meaning. Words like of, are, the, it, is are the examples of stop words. In stop words removal, stop words are removed. In white space, extra spaces like tabs, etc are removed. Parts-of-speech tagging is a process of converting a sentence to the forms of tag. The tag signifies whether the word is a noun, adjective, verb etc. The words having positive and negative sentiments in the summary attribute of improvement requests are identified and its sentiment is calculated. The Features as well as the summary attributes are used for training and testing.

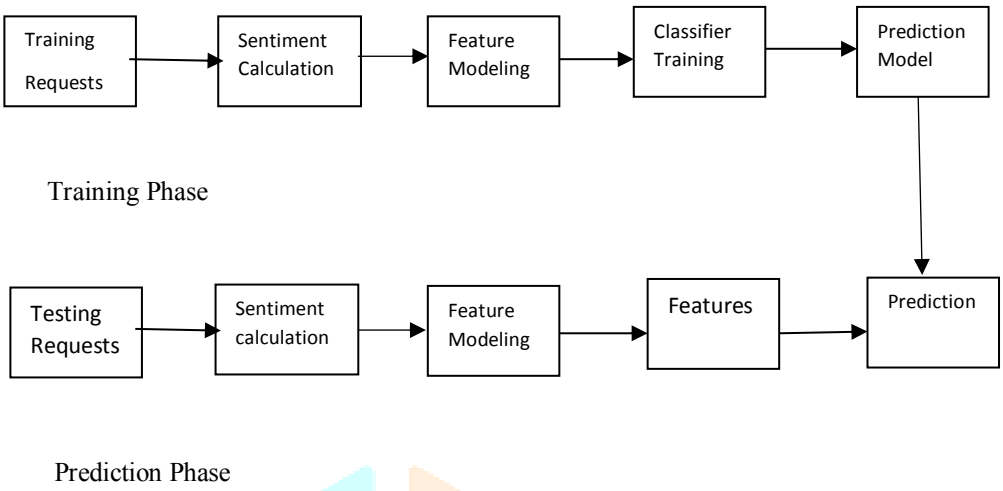

Figure 1: Process flow of Proposed Approach

Sentiment analysis helps in predicting how the improvement requests are approved for implementation or rejection, based on the following reasons. People usually respond in a positive manner for positive and constructive feedback. Here also same hold true. Requests having positive words and with suggested solutions are more likely to be implemented than the ones with negative words and without any suggestions. Example: A particular software application is not user friendly and can be improved if suitable popup's at appropriate places are given, is a constructive feedback which can be implemented. Instead, if a request is of the kind that a particular software is not attractive and not user friendly, is more of a complaining kind. So this is more likely to be rejected. Statistically also it was proved that the negatively oriented requests were rejected and positively oriented requests were accepted. We train a classifier and test the trained classifier on different sets of the requests. The ideology of the paper is organized as follows: Section 1 is the Introduction that highlights the time consuming task of manually going through the improvement requests and the need to overcome this problem. Section 2 speaks about related works. Section 3 explains about the proposed approach in detail. Section 4 shows the experimental evaluation to show the effectiveness of the proposed work. Section 5 concludes the work by igniting with future directions.

2. LITERATURE SURVEY

Sentiment analysis is the interpretation and classification of emotions (positive, negative, neutral) within text data using text analysis techniques. Sentiment analysis tools allow businesses to identify customer sentiment towards products, brands or services in online feedback. The benefits of sentiment analysis include sorting data efficiently at a very huge scale thus reducing the wastage of time, for identifying critical issues in real time. The main types of algorithms used include a) Rule–based systems that perform sentiment analysis based on a set of manually crafted rules b) Automatic systems that rely on machine learning techniques to learn from data c) Hybrid systems that combine both rule-based and automatic approaches. [1] discusses about the overview of sentimental analysis approach of Opinion Mining with the techniques and tools. [2] Provides the background information on the semantic analysis and its development over time. The approach and the steps involved in developing a software system for opinion mining is presented in [3]. [4] Experiments for sentence level and review level categorization was performed using Naïve Bayesian, Random Forest and Support Vector Machine algorithms. [5] The authors have presented about how the sentiment analysis has shifted from analysing online product reviews to social media texts from Twitter and Facebook. Now a days sentiment analysis is used in many fields like stock markets, elections, disasters, medicine, software engineering, cyberbullying. [6] In this paper, a study was conducted on the happenings of being happy and unhappy during the development of software. It was proved that there are damaging effects of unhappiness and beneficial effects of fostering happiness in the job environment. [7] A tool was developed for improved sentiment analysis in text especially designed for application in the software engineering domain. [8] shows the advantage of using KNN and how it outperforms other traditional cost estimation models such as linear regression and COCOMO (Constructive Cost Model), when the dataset is small.[10] It was proved that Manual prioritization is time-consuming and cumbersome. To overcome this drawback, an automated state-of-the art approach is used which recommends the priority level information of the bug reports. [11] shows the usage of multinomial naïve Bayes classifier for binary classification of the reports into approved or rejected.

3. PROPOSED APPROACH

In the proposed Method the improvement requests are classified into two categories of approved and rejected. Figure 1 represent the system model to carry out the proposed work.

Step1: In the first phase that is Training phase we pre-process the improvement requests of the real software applications using natural language processing techniques. Sentiment of each request is calculated in this phase. Each pre-processed request and its sentiment is converted into a feature vector.

Step2: In the second phase, that is prediction phase, feature vectors of the requests and the corresponding category label are used for the training of the classifier.

Step3: The trained classifier is applied to the vectors of testing requests for their approval prediction. The approval prediction of the new request re into a defined category c could be defined as a function f,

C=f(ar) where C belongs to {approved, rejected} , ar belongs to AR (1)

Here C represents the classification result: approved or rejected ,

f represents the classification function of approval prediction ,

ar represent a request which is an input of the function and,

AR represents the set of the improvement requests.

The data of improvement requests from an issue tracking system extracted by Nizamani et al. (2017) is used . First each request is preprocessed and sentiment for each request is calculated to extract its features. Then we train the machine learning classifier to predict whether the given request would be approved or rejected. Each of the steps in detail are explained in the next section.

4. SYSTEM MODEL AND METHODOLOGIES

In figure 2, the architecture of the system model is shown.

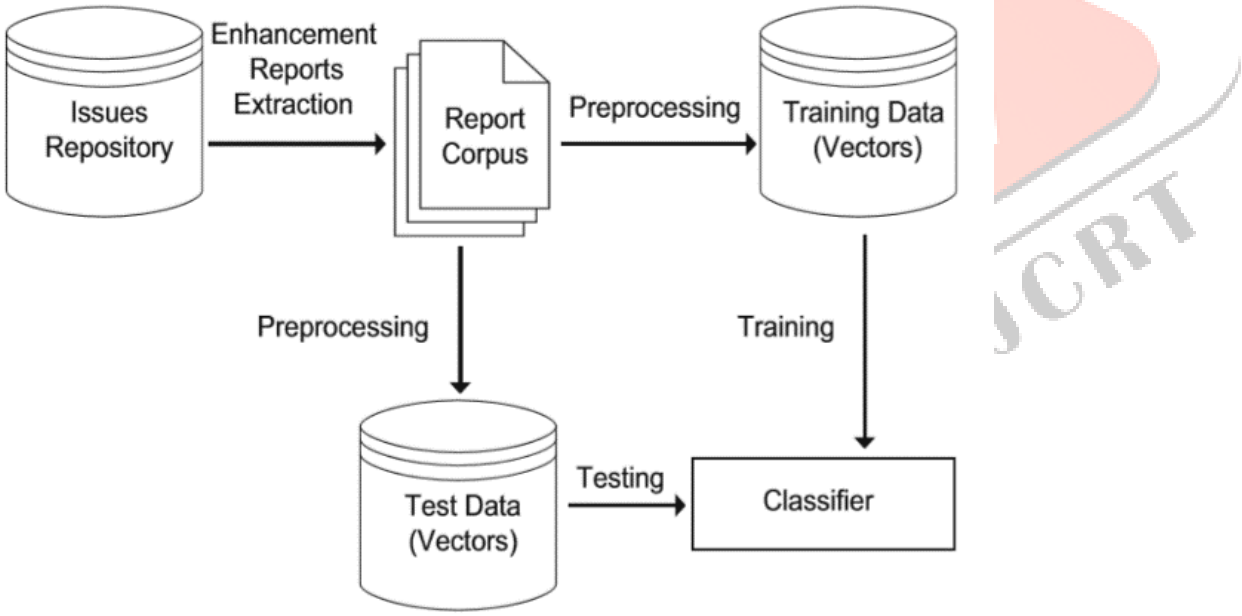

Figure 2. Architecture of the proposed Approach

An architecture is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system. Each structure comprises software elements, relations among them, and properties of both elements and relations.

The Architecture of the proposed approach is shown in figure 2. Issues repository contain all kinds of the requests including bugs etc. From Issues Repository, the extraction of Enhancement Reports are done. These requests or reports are pre-processed by using techniques like tokenization, spelling correction, lower case conversion, white space removal, Parts-of-speech tagging , Lemmatization etc. Then, the data is separated into two parts, training and testing data. SVM classifier, Logistic Regression classifier and Naïve Bayes classifiers are trained using training data. Then each of these Classifier are used to test the data from the testing data. Each of the classifier's performance is evaluated by using the metrics like accuracy, precision, recall, and F-measure. From the performance of each of the classifier's the best one is selected.

4.1 Data Acquisition

The software application issues are reported into an issue tracking system which allows users to report the problems , keeps track of registered problems and provides a platform to the developer to resolve the registered problems and maintain a good quality product. We have used the enhancement reports of open-source software applications extracted from Bugzilla. A report ar is formalized as

ar= <d,r> (2)

where d represents the textual description of a report and r represents its resolution attribute.

4.2 Sentiment Calculation

Sentiment analysis is the interpretation and classification of emotions with in the text data using text analysis techniques. Sentiment analysis allows to identify users sentiment towards products, brands or services, with online conversations and feedback reports. There are various Sentiment analysis tools to analyse the sentiment. Sentiment analysis tools use Natural Language Processing (NLP) to analyse online conversations and determine deeper context- positive, negative, neutral. These tools mimic our brains, to a greater or lesser extent, allowing us to monitor the sentiment behind online content. Some of the popular sentiment analysis tools are as below. 1) Quick search gives the instant overview of the brand online. It's is social media search engine that offers extensive coverage of social networks-including news, sites, blogs and forums.2) Hootsuite Insights automatically analyses all your social media platforms, news sites, forums and blogs to reveal insights that include influencers, stories, trends and sentiment. 3) RapidMiner provides text mining to help brands perform sentiment analysis and unites data preparation, machine learning and predictive model deployment. 4) NCSU Tweet Visualizer is for Twitter sentiment analysis. It has the choices of views, where in use can select either sentiment view or Topics view or Heatmap view. 5) Meaning Cloud, the sentiment analysis API implements a detailed, multilingual analysis of content from several sources. 6) Social Mention is a real-time search platform that monitors more than 100 social platforms and aggregates user generated content.7) Sentiment Analyzer uses computational linguistics and text mining to identify the sentiment behind text. It judges the overall sentiment score, regardless of its length it considers the entire sample. 8) SentiStrength is a free sentiment analysis tool for academic research. 9) Sentigem is a analysis tool for English language documents or blocks of text. 10) Social searcher is a Real-time search engine for Twitter and Google+. For this paper we have used Sentiment Analyzer, sentiment analysis tool. This is free and easy to use tool.

We pass the textual representation of each request ar to the sentiment Analyzer that computes the sentiment of the given request. The sentiments are stored with the corresponding reports. A report with its sentiment is formalized as ,

ar=<d,s,r> (3)

where d represents the pre-processed textual description of a request ar, s represents the sentiment of a report ar and r represents the resolution attribute.

4.3 Pre-processing

We have used R tool for pre-processing, which includes Tokenization, spelling correction, lower case conversion, removal of punctuation marks, stop words removal, white space removal, Parts-of-speech tagging, lemmatization. Each request has gone through the phases of pre-processing as shown in fig 2 and stored for the feature extraction.

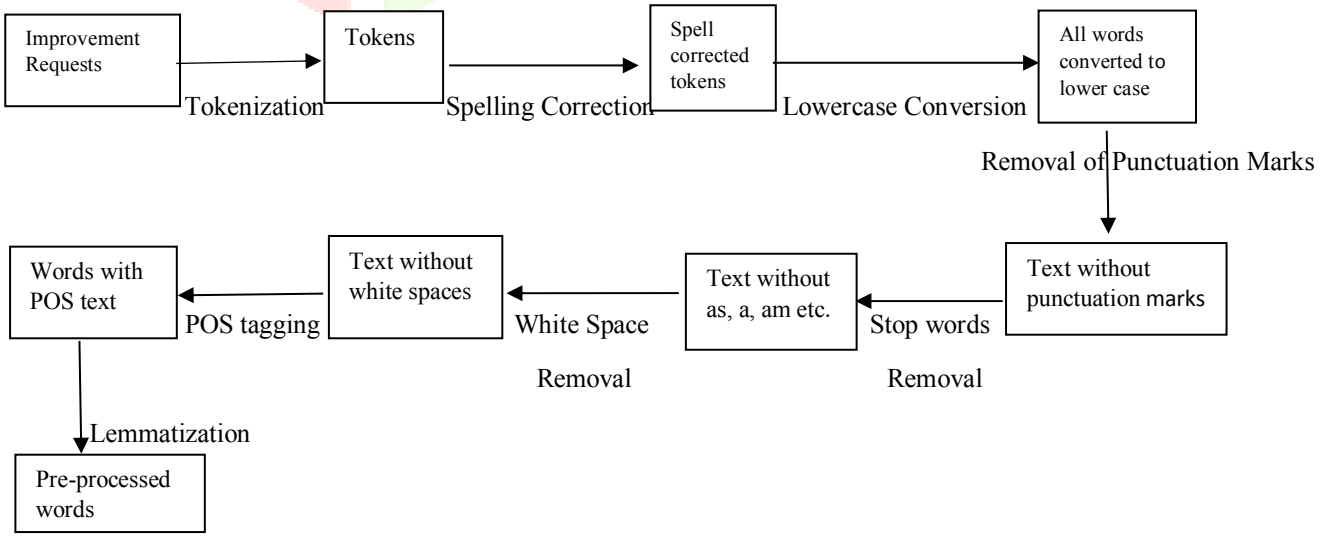

Figure 3: Steps followed in Pre-processing of data that is improvement requests.

First we tokenize each improvement requests using white spaces involved in the summary attribute. Then these tokens go through spelling correction. For example, in a sentence where there should be their , if there is written, it will be corrected. In the next step, all the letters are converted to lower case. Next step is the removal of punctuation marks. Then the stop words like as, a , and , but ,

how, or, what , am, about etc are removed and also the special characters like hypen,@ etc are removed. From this text, the white spaces are removed. Then each token is tagged with the POS tag. At last, Lemmatization is performed which links words with similar meaning to one word. It transforms the superlative and comparative words into their root words because both have same meanings. Example of lemmatization is words like work, worked, working are replaced with work. The preprocessed report is formalized as

ar=<d′, s, r> where (4)

d′= <t1,t2… tn> (5)

where d′ represents the preprocessed textual description of a request ar, s represents the sentiment of ar, r represents the resolution attribute and t1,t2, ….tn represents the n tokens involved in d′.

4.4 Feature Modeling

Matrix is created for the dataset of improvement requests considered. The matrix contains the sentiment and features of reports where the features are the list of words from all the requests. Each row, in the matrix comprehensivly represents a preprocessed request and columns represent the sentiment and features in it. A request in the matrix is defined by

ar=<s, f1, f2, … fn> (6)

where ar represents a report, s and f1,f2 … fn represent the sentiment and the features respectively.

Equation (6) is used to model each request. We count the frequency (N) of each feature to populate the matrix. In this regard, the following conditions are applied to each request to assign the value of each feature.

$f_i(ar) = 0$, if $t_i \notin d'$ (7)

$f_i(ar) = 1$, if $t_i \in d'$

where f_i is a set of features of pre-processed textual description d′ of a report ar .

4.5 Training :

The Support Vector Machine (SVM) algorithm is a popular machine learning tool that offers solutions for both classification and regression problems. SVM was developed at AT&T Bell Laboratories by Vapnik with colleagues. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. We train the support vector classifier, Naïve bayes classifier, and logistic regression classifier using the feature sets of each request which are tagged while feature modelling using equation (6).

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

Naïve Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables.

The trained classifier is tested to predict the approval of requests. To construct a classification model that assigns an unlabeled request to a class c as defined in Eq. (1) , a set of reports AR = (ar 1 , ar 2 , ..., ar n) is given. Each request from AR contains its classification category c as mentioned in Eq. (1) , and sentiment s and set of features f_i as mentioned in Eq. (6) . We train support vector clas- sifier by using the proposed approach. We use the approved and rejected requests to seek the decision surface for the approval prediction of reports. The decision surface is a hyperplane which is used for the training of proposed approach. Given the requests ar i labeled into specified binary categories y_i , find a weight vector w such that discriminant function separates the categories for i = 1, 2, ... n and can be formalized as

$f(ar_i) = w^T er\ i + b$

where f is the discriminant function which calculates the decision surface for the training data ar of enhancement reports, w is the normal to decision surface which is known as weight vector and b is bias.

4.6 Prediction

Once the vector is defined with the training set, each report from the testing dataset may be approved if $w^T er + b > 1$ rejected otherwise

5. EVALUATION

Four algorithms namely SVM, Logistic Regression, Naïve Bayes and Random Forest have shown the comparison between these algorithms, by following the proposed approach. we have defined the metrics and evaluation process. Finally, the findings and results are shown in which it is found that SVM outperforms Logistic Regression and Naïve Bayes in terms of improvement approval prediction. The performance of the proposed approach by considering accuracy, precision, recall.

5.1 Dataset

The data extracted by Nizamani et al. (2017) from Bugzilla using Bugzilla Native REST API. This history data of requests is available online related to Mozilla ecosystem. Each request is having the attributes summary, status, Resolution, Id, Product, Severity. We used the Severity attribute to access the requests from Bugzilla. The value of severity attribute varies from blocker, critical, major, normal, minor, trivial and enhancement. Therefore, the value of severity attribute is specified as enhancement in the URL of Bugzilla REST API to filter the reports from bugs reports.

5.2 Process:

The proposed approach is evaluated by following the steps as given below. First, the obtained requests or reports are pre-processed by using the steps shown in figure 2. We select few requests AR for training and consider it as training data D and other requests AR as S for testing.

First we train SVM classifier, second we train Logistic Regression Classifier, Third we train Naïve Bayes classifier, Fourth we train Random Forest Classifier on data D. Fifth, for each report from S, we predict whether it could be approved using SVM, Logistic Regression, Naïve Bayes and Random Forest.

Finally we have calculated the accuracy, precision, recall, F-measure, Matthews correlation coefficient and odds ratio for each classifier.

The below given well known metrics of accuracy, precision, recall, and F-measure are used to evaluate the performance of classification algorithms. We compute the approval specific accuracy, precision, recall, and F-measure of the approaches that can be formalized as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (9)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (10)$$

The metrics Accuracy, Precision, Recall are the considered to calculate accuracy and precision, recall and F-measure are considered in predicting approval of the requests.

TP is the number of requests that are correctly predicted as approved,

TN is the number of requests that are correctly predicted as rejected,

FP is the number of requests that are incorrectly predicted as approved, and

FN is the number of requests that are incorrectly predicted as rejected.

We also compute the Mathews correlation coefficient (MCC) and odds ratio (OR) to measure the quality and effectiveness of the classifier, respectively.

$$MCC = \frac{TP * TN - FP * FN}{(TP + FP)(TP + FN)(TN + FP)(TN + FN)} \quad (12)$$

$$OR = \frac{TP/FP}{FN/TN} \quad (13)$$

6. RESULTS

The evaluation results of the proposed approach is shown in the below figures.

Algorithm	Accuracy	Precision	Recall
Proposed Algorithm (Support Vector Machine)	0.92	0.92	0.92
Naïve Bayes	0.87	0.87	0.92
Logistic Regression	0.82	0.92	0.82
Random Forest	0.87	0.92	0.77

Table 1: Algorithms metrics

Table 1 shows that Proposed algorithm gives more accuracy comparing to naïve bayes, Logistic Regression, Random Forest

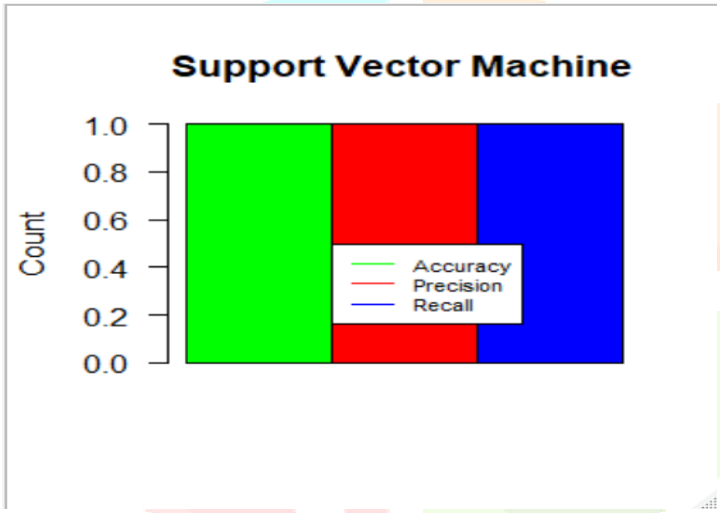

Figure 4: SVM classifier accuracy , precision and recall

From the above figure , it is observed that SVM is giving 90% accuracy.

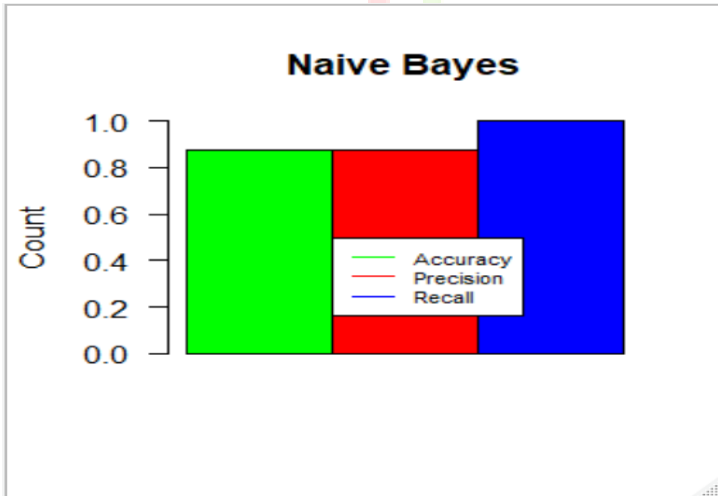

Figure 5: Naïve bayes classifier accuracy, precision and recall

From the above figure, it is observed that accuracy of 87.6, precision of 87.6 and Recall of 1 is achieved with Naïve bayes classifer.

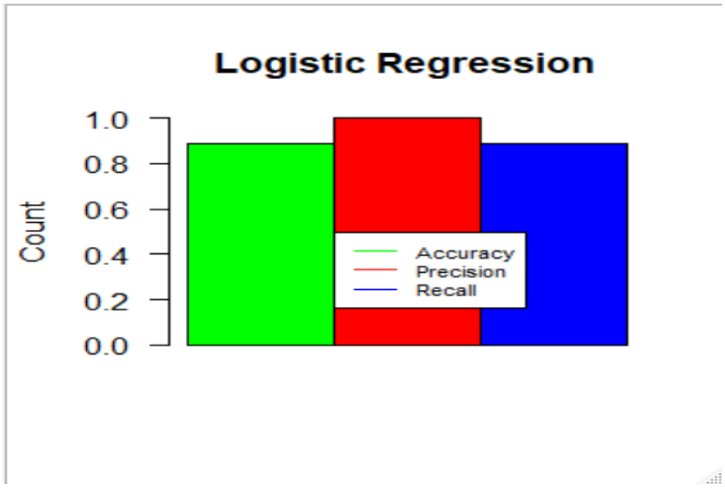

Figure 6: Logistic Regression classifier accuracy, precision and Recall

From the above figure it is observed that Logistic Regression classifier is giving an accuracy of 88%, recall of 88% and recall of 88.9%.

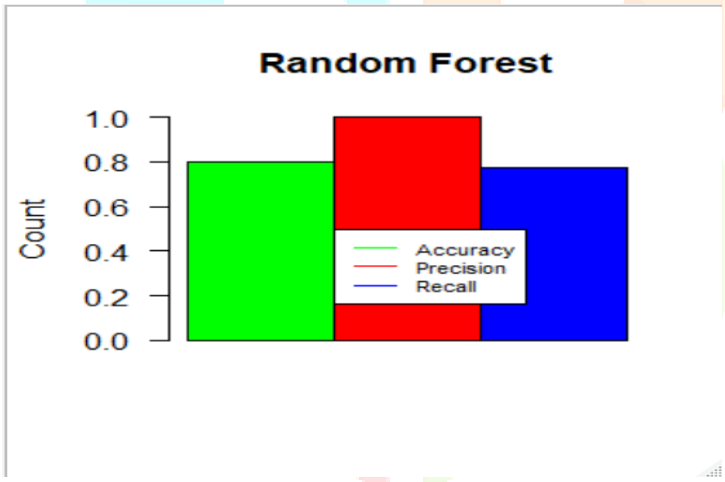

Figure 7: Random Forest classifier accuracy, precision and Recall

It was also observed that Random Forest classifier is giving the Accuracy and precision of 90% and recall of 1, but taking lot of time for processing compared to SVM, Logistic Regression and Naïve Bayes. So for this data, we can conclude that SVM out performs other algorithms.

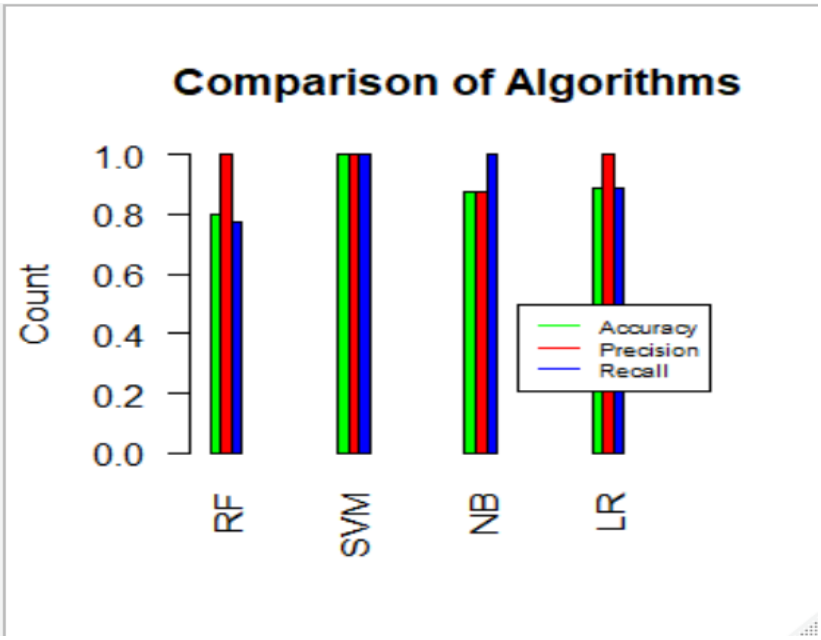

Figure 8: Comparison of three classifiers, SVM, Naïve Bayes and Logistic Regression for metrics accuracy, precision and recall is shown.

7. CONCLUSION AND FUTURE WORK:

Huge numbers of improvement requests are received frequently for the software applications which have high number of stake holders. But most of the improvement requests are rejected due to various reasons like negative feedback, without any suggestions, incomplete requests etc. Developers going through each of these requests, to categorise them into likely to be implemented and likely to be rejected is a very tedious task. To overcome this problem , suitable approach is presented in this paper with which the wastage of time of the developers is avoided and the improvement requests can be categorized in an efficient manner. The evaluation of the proposed approach was done using Naïve bayes classification, and SVM classification techniques. Compared to Naive bayes, SVM is giving better accuracy of 92%. It was also shown that the Improvement requests which are positive and have constructive suggestions are implemented, in contrast to the ones which are negative and without any suggestions.

Work can be extended by accommodating the software specific code in the improvement requests as part of the suggestions in the improvement requests and thus helping the developers to readily implement the solution. Special pre-processing techniques would be required to handle these type of the improvement requests. Furthermore, we would like to explore various swarm intelligence algorithms to reduce the time taken and improve the efficiency.

8. ACKNOWLEDMENT

The authors would like to say thanks to the associate editor and the anonymous reviewers for their insightful comments and constructive suggestions. The work is supported by the International Journal of Creative Research Thoughts (IJCRT).

REFERENCES:

[1]. Abhishek kaushik, Anchal Kaushik,Sudhanshu Naithani . A study on Sentiment Analysis – Methods and Tools. International Journal of Science and Research (IJSR) ISSN :2319-7064 Volume 4 Issue 12, December 2015.

[2]. Oskar Ahlgren Sch. of Econ., Dept of Inf. and Econ. , AaltoUniv, Aalto , Finland, Research on Sentiment Analysis : The First Decade, IEEE 16th International Conference on Data Mining Workshops (ICDMW) 2016 **ISSN:** 2375-9259

[3]. Dongjoo Lee, Ok-Ran Jeong , Sang-goo Lee , Opinion Mining of Customer Feedback Data on the Web, ICUIMC '08: Proceedings of the 2nd international conference on Ubiquitous information management and communicationJanuary 2008 Pages 230–235 https://doi.org/10.1145/1352793.1352842

[4]. Xing Fang and Justin Zhan, sentiment analysis using product review data , Fang and Zhan Journal of Big Data (2015) 2:5 DOI 10.1186/s40537-015-0015-2.

[5]. Mika V. Mäntylä, Daniel Graziotin, Miikka Kuutila, The evolution of sentiment analysis—A review of research topics, venues, and top cited papers, Computer Science Review, Volume 27,February 2018, Pages 16-32, ISSN 1574-0137, https://doi.org/10.1016/j.cosrev.2017.10.002.

[6]. Graziotin, D., Fagerholm, F., Wang, X., Abrahamsson, P., 2018. What happens when software developers are (un)happy. J. Syst. Softw. 140, 32–47. doi: 10.1016/j.jss. 2018.02.041 .

[7]. Md Rakibul Islam, Minhax.F Zibran, Sentistrength-SE: exploiting domain specificity for improved sentiment analysis in software engineering text. J. Syst. Softw. 145, 125–146. doi: 10.1016/j.jss.2018.08.030 .

[8]. https://github.com/shanniz/Bugzilla .

[9]. https://bugzilla.mozilla.org/rest/bug?severity=enhancement

[10]. Umer, Q., Liu, H., Sultan, Y., 2018. Emotion based automated priority prediction for bug reports. IEEE Access 6

[11]. Nizamani, Z.A., Liu, H., Chen, D.M., Niu, Z., 2017. Automatic approval prediction for software enhancement requests. Autom. Softw. Eng. doi:10.1007/ s10515-017-0229-y.

[12]. Jongeling, R., Sarkar, P., Datta, S., Serebrenik, A., 2017. On negative results when using sentiment analysis tools for software engineering research. Empirical Softw. Eng. 22 (5), 2543–2584. doi:10.1007/s10664-016-9493-x

[13]. Sohrawardi, S.J., Azam, I., Hosain, S., 2014. A comparative study of text classification algorithms on user submitted bug reports. In: Ninth International Conference on Digital Information Management (ICDIM 2014), pp. 242–247. doi:10.1109/ICDIM. 2014.6991434.

[14]. Sharma, G., Sharma, S., Gujral, S., 2015. A novel way of assessing software bug severity using dictionary of critical terms. Procedia Comput. Sci. 70 (Supplement C), 632–639. doi:10.1016/j.procs.2015.10.059.

[15]. Alipour, A., Hindle, A., Stroulia, E., 2013. A contextual approach towards more accurate duplicate bug report detection. In: Proceedings of the 10th Working Conference on Mining Software Repositories. IEEE Press, Piscataway, NJ, USA, pp. 183–192. Antoniol, G., Ayari.