Int J Syst Assur Eng Manag (Oct-Dec 2014) 5(4):651–663
DOI 10.1007/s13198-014-0219-4

ORIGINAL ARTICLE

An empirical evaluation of cross project priority prediction

Meera Sharma · Punam Bedi · V. B. Singh

Received: 24 November 2013 / Published online: 11 February 2014

Abstract Prioritization of bugs decides the bug fix sequence. Incorrect prioritization of bugs results in delay of resolving the important bugs, which leads delay in release of the software. Prediction of bug priority needs historical data on which we can train the classifiers. However, such historical data is not always available in practice. In this situation, building classifiers based on cross project is the solution. In the available literature, we found very few papers for bug priority prediction and none of them dealt in cross project context. In this paper, we have evaluated the performance of different machine learning techniques namely Support Vector Machine, Naive Bayes (NB), K-Nearest Neighbors and Neural Network in predicting the priority of the newly coming reports in intra and cross project context. To evaluate the performance of these machine learning techniques for priority prediction in cross project context, we have considered three scenarios: (i) 10 fold cross-validation for intra project (ii) cross project validation for inter projects and (iii) inter project cross validation with different combination of training datasets. We performed experiments for each scenario on five datasets. Results from these experiments conclude that the accuracy performance for all machine learning techniques except NB is above 70, 72 and 73 % in respective scenarios. The experimental results also show that the combination of different project datasets for training candidates does not provide a significant improvement in performance measures.

Keywords Bug repositories · Bug priority · Classifiers · 10 fold cross validation · SVM · K-NN · Naive Bayes · Neural network · Cross project validation

M. Sharma (✉) · P. Bedi
Department of Computer Science, University of Delhi, Delhi, India
e-mail: meera_sharma44@yahoo.com

V. B. Singh
Delhi College of Arts & Commerce, University of Delhi, Delhi, India
e-mail: vbsinghdcacdu@gmail.com

1 Introduction

With the frequent changes in the software source code to meet the changing and enormous requirements of the users, a large number of bugs are being reported on daily basis. A bug may be reported by a user, a developer, or by any staff member. The reported bugs should be analyzed carefully to find it out whether the bug is correct or incorrect, valid or invalid, important or unimportant, new or duplicate. We assign priority to a bug so that an important bug should not be left untreated for a long time. Correct prioritization is again a problem. The reporter may not have the complete knowledge of the project which may result in incorrect prioritization. Triager (a person who analyzes, expands and overall refines the bugs) with his knowledge and experience assign the priority to the bug. Manually doing this is a cumbersome task. We need to automate the process of bug priority prediction.

Cross project study is getting an edge in software engineering to predict bugs, cost, bug fix time, severity and priority and some other attributes or project property on the basis of historical data of other projects. In the available literature, very few attempts have been made in cross project study and very few authors have attempted to work on cross project validation for priority prediction. Recently, Sharma et al. (2012) have made an effort to predict the priority of a reported bug in cross project context but it was for a limited number of cases and combination of datasets for developing training candidate has not been considered which is proven to be very successful. In this paper we have made an attempt to predict the priority of a reported bug using different machine learning techniques and investigated the priority predictions in the cross project context by considering the combination of datasets for training candidates. To study and investigate the priority prediction in intra and cross project context, we have set the following research questions:

Research Question 1:	Which machine learning technique is most appropriate for priority prediction?
Research Question 2:	How performances of different machine learning techniques vary with number of terms?
Research Question 3:	Does training data from other projects provide acceptable priority prediction results?
Research Question 4:	Does the combination of training data sets provide better performance than single training data set?

We have visualized the empirical evaluation and experimental setup by creating three scenarios.

Scenario 1	10 fold cross-validation within same project dataset
Scenario 2	Cross project validation across different project datasets
Scenario 3	Cross project validation by combining different project datasets as training candidate

Scenario 1 answers the research questions 1 and 2. Scenario 2 answers the research question 3 and scenario 3 answers research question 4.

We have applied Support Vector Machine (SVM), Naïve Bayes (NB), K-Nearest Neighbors and Neural Network classification techniques on bug repositories of open source projects namely Eclipse and OpenOffice. The performance measures namely accuracy, precision, recall and F-measure have been calculated using 10 fold cross-validation with stratified sampling. Rest of the paper is organized as follows: Section 2 provides related research work. Section 3 discusses about the pre-processing and representation of textual data (summary) of bug reports. Section 4 describes the dataset and features selected. Section 5 deals with the experimental setup. Section 6 discusses about the results. Section 7 mentions the threats to validity of results and finally, the paper is concluded in Section 8 with future research directions.

2 Related work

Attributes of a bug can be used to predict the priority, severity, fixer and status of the bug. Summary field represents the information about the bug (what this bug is). By mining the summary field and using machine learning techniques we can predict various attributes for new bug report. Canfora and Cerulo (2006) have proposed a study on how change requests have been assigned to developers involved in an open source project (Mozilla and KDE) and a method to suggest the set of best candidate developers to resolve a new change request. An approach consisting of constructing a recommender for bug assignments has been proposed by Anvik (2006). Anvik et al. (2006) applied SVM, Naive Bayes and Decision Trees on Eclipse, Firefox and GCC projects for automatic assignment of developer to a new bug report. Tamrawi et al. (2011) proposed a fuzzy set-based approach for automatic assignment of developers to bug reports. Weib et al. (2007) presented an approach to predict the fixing effort for an issue allowing early effort estimation, helping in assigning issues and scheduling stable releases. Kim and Whitehead (2006) computed the bug-fix time of files in ArgoUML and PostgreSQL by identifying when bugs are introduced and when the bugs are fixed. Lamkanfi et al. (2010) investigated whether we can accurately predict the severity of a reported bug by analyzing its textual description using text mining algorithms. Authors concluded that it is possible to predict the severity with a reasonable accuracy (both precision and recall vary between 0.65–0.75 with Mozilla and Eclipse; 0.70–0.85 in the case of GNOME). Chaturvedi and Singh (2012) have made an attempt to demonstrate the applicability of machine learning algorithms namely NB, K-Nearest Neighbor, NB Multinomial, SVM, J48 and RIPPER in determining the severity class of bug report data of NASA from PROMISE repository. Menzies and Marcus (2008) presented a new and automated method named SEVERIS (SEVERity ISsue assessment), to assist the test engineer in assigning severity levels of defect reports of NASA's Project and Issue Tracking System (PITS). Anvik and Murphy (2011) proposed a recommender for development oriented decisions like assignment of developer to a new bug report, prediction of the component for new bug report. Marks et al. (2011) have done the analysis for different features of bug reports to find the characteristics of bug fix-time using Mozilla and Eclipse bug repositories. Sharma et al. (2013) predicted the cc-list count of a bug by using different regression techniques. Yu et al. (2010) predicted the priority of a bug during software testing process using artificial neural network and NB classifier. Kanwal and Maqbool (2010, 2012) used a classification based approach to compare and evaluate the

SVM and NB classifiers to automate the prioritization of new bug report by using the categorical and textual attributes of bug report to train the model. They have shown that SVM performance was better than the NB with textual attributes and NB performance was better than SVM for categorical attributes. But this analysis has been carried out on a limited data and techniques. Cross project study is a new and challenging task.

Our study on cross project priority prediction is motivated and based on the following papers:

Recently, Sharma et al. (2012) conducted a study to predict the bug priority within project using SVM, K-NN, NB and Neural Net, and found that SVM and Neural Network are better than K-NN and K-NN is better than NB. He et al. (2012) have done an investigation on the feasibility of cross-project defect prediction and found that cross-project defect prediction is better than prediction when training data set is from the same project. Zimmermann et al. (2009) performed a large scale experiment on Data versus Domain versus Process cross-project defect prediction. Turhan et al. (2009) have done an empirical study on the relative value of cross-company and within-company data for defect prediction.

3 Preprocessing and representation of data

We have predicted the priority of a bug report based on its summary attribute entered by user at the time of bug filling. We pre-processed the bug summary in Rapid Miner tool containing the following steps:

3.1 Tokenization

Tokenization is the process of breaking a stream of text into words, phrases, symbols, or other meaningful elements called tokens. In this paper a word or a term has been considered as token.

3.2 Stop word removal

In bug summary prepositions, conjunctions, articles, verbs, nouns, pronouns, adverbs, adjectives, etc. are stop words and have been removed.

3.3 Stemming to base stem

The process of converting derived words to their base word is known as stemming. In this paper, we have used Standard Porter stemming algorithm (Porter 2008) for stemming.

3.4 Feature reduction

Tokens of minimum length 3 and maximum length 50 have been considered because most of the data mining algorithm may not be able to handle large feature sets.

3.5 Weight by information gain or infogain

Information gain tells the importance or relevance of the term or token.

To make textual data structured for analysis it is represented as document*term matrix where the rows are considered as documents or files and columns are considered as terms or tokens. The frequency of a term in the document will be counted and stored in the matrix form. $\mathrm{TF_i}$ is the occurrences of a term in the document. To normalize $\mathrm{TF_i}$ it is divided by total number of terms in the document, n is the total number of terms in the document.

TF × IDF represents "term frequency (TF) times inverse document frequency (IDF)". It determines the importance of a term in the complete dataset or document set. Importance of a term increases with the frequency count of the term in the document but is offset by the frequency of the word in the dataset. The inverse document frequency is obtained by dividing the number of all documents by the number of documents containing the term, and then taking the logarithm of that quotient. This representation is used to rank the terms and selecting top few terms.

$$\mathrm{Wi} = \mathrm{TFi} \times \mathrm{IDFi} \text{ where } \mathrm{IDFi} = \log(N/\mathrm{DFi})$$

$\mathrm{DF_i}$ is the document frequency which shows the appearance of a particular term in the number of documents and N is the total number of documents.

Different accuracy measures namely accuracy, precision, recall and F-measure can be calculated to measure the performance of a classifier.

Accuracy of a classifier is defined as proportions of classifications, over all the N examples that were correct. It is the percentage of correct classification.

Precision of a class A is defined as number of instances correctly classified as class A divided by the total number of instances classified as class A. It measures the percentages of correct predictions related to the predictions made by the classifier.

$$\text{Precision} = \frac{\text{No. of instances correctly classified as class A}}{\text{Total number of instances classified as class A}}$$

Recall of a class A is the number of instances correctly classified as class A divided by the total number of instances in the dataset having class label A. It measures the percentage of correct predictions related to actual class.

$$\text{Recall} = \frac{\text{No. of instances correctly classified as class A}}{\text{Total no. of instances in dataset having class label A}}$$

F-measure is calculated to measure the average performance of the classifier to avoid the bias towards precision or recall. It is twice of the harmonic mean of precision and recall.

$$\text{F - Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{(\text{Precision} + \text{Recall})}$$

4 Dataset and feature selection

In this paper, we have collected bug reports from Eclipse and OpenOffice projects. These projects are very popular in terms of their usage. Eclipse projects provide a development platform with extensible frameworks, tools and runtimes for building and managing software throughout its life time. In the Eclipse project, we have considered 6 products and 49 components. We have used only bug reports for platform product which has 21 components because other components do not have sufficient number of bug reports. Bug reports were collected up to 13th July 2012 from http://bugs.eclipse.org/bugs/. For the OpenOffice, we have collected Database Access, Presentation and Spreadsheet products bug reports taken up to 17th July 2012 from http://issues.apache.org/ooo/. We have used summary feature of bug report to predict the priority of newly coming bug. Only resolved bug reports with status value “resolved”, “closed” or “confirmed” with “fixed” and “duplicate” resolution have been taken because only these types of bug reports contain the meaningful information for building and training the models. Table 1 shows priority wise bug reports of different Eclipse and OpenOffice products.

5 Experimental setup

To conduct the experiment, we made an automated workflow in Rapid Miner (Mierswa et al. 2006) containing steps for preprocessing of bug reports, model building, cross validation and model testing.

A graphical presentation for generating training datasets for Database Access (DB) product of OpenOffice project in scenario 2 and 3 has been shown in Fig. 1.

6 Results and discussion

Scenario 1 we have applied different machine learning techniques namely SVM, Naive Bayes, K-Nearest Neighbors and Neural Network for predicting the priority of reported bug using 10 fold cross-validation and varying number of terms from 25 to 200 (Sharma et al. 2012). We have tried different kernels and other parameters and then we used the appropriate parameters values to get the significant amount of performance. For SVM we have taken polynomial kernel with degree 3.

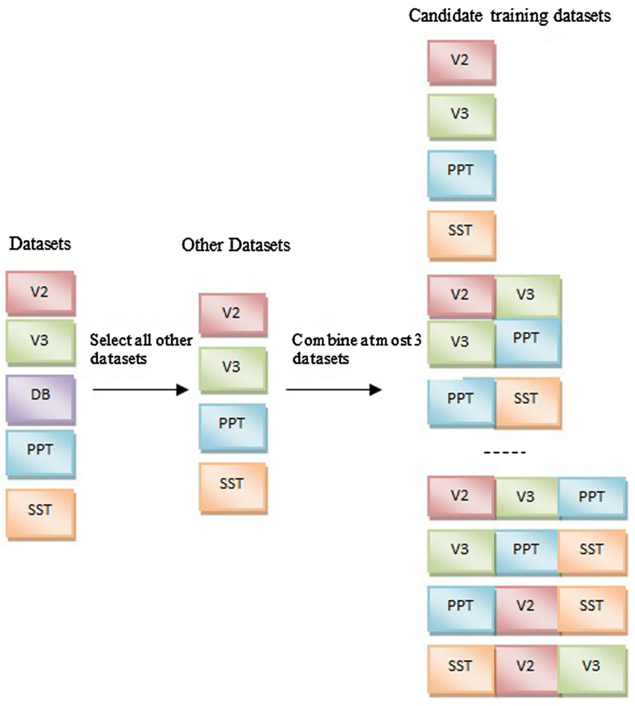
Candidate training datasets
Datasets
Other Datasets
V2
V3
DB
PPT
SST
Select all other datasets
Combine atmost 3 datasets

Fig. 1 Procedure of generating training sets for DB

Table 1 Number of bug reports in different projects

Projects	Product	Priority level					
		P1	P2	P3	P4	P5	Total
Eclipse	Platform version 2(V2)	923	1,416	8,609	370	229	11,547
Eclipse	Platform version 3(V3)	361	963	26,667	320	136	28,447
OpenOffice	Database Access (DB)	76	472	2,834	243	38	3,663
OpenOffice	Spreadsheet (SST)	82	518	4,210	316	114	5,240
OpenOffice	Presentation (PPT)	62	553	2,688	90	37	3,430

Table 2 Accuracy measures of SVM and NB classifiers

Project	Product	Classifier	No. of terms/features/attributes							
			Accuracy (in %)							
			25	50	75	100	125	150	175	200
Eclipse	Platform version 2 & subversions	SVM	74.66	74.66	74.76	74.67	74.62	74.44	74.49	74.48
		Naïve Bayes	5.93	6.77	8.33	8.52	8.97	9.28	9.27	9.38
Eclipse	Platform version 3 & subversions	SVM	93.73	93.70	93.68	93.71	93.72	93.69	93.69	93.70
		Naïve Bayes	3.48	3.85	5.86	5.855	6.25	6.56	7.21	7.54
OpenOffice	Database Access	SVM	77.94	78.19	78.21	78.27	78.02	78.11	77.94	78.13
		Naïve Bayes	9.17	10.16	9.61	12.99	14.88	16.22	15.86	16.65
OpenOffice	Spreadsheet	SVM	81.07	80.99	80.84	80.84	80.88	80.90	81.05	81.09
		Naïve Bayes	4.75	5.32	7.00	7.92	8.55	9.43	9.90	11.01
OpenOffice	Presentation	SVM	79.74	79.74	80.09	80.20	80.12	80.23	80.26	80.17
		Naïve Bayes	4.72	9.21	13.35	15.95	15.95	17.35	17.78	17.93

Accuracy of SVM and NB has been shown in Table 2.

Above table shows that the accuracy of SVM is more than 74 % for all datasets. It slightly improves or just fluctuates by increasing the number of terms from 25 to 200. But NB accuracy increases as we increase the number of terms from 25 to 200. For Eclipse project version 2 and subversions, SVM accuracy slightly increased from 74.66 to 74.76 % for variation in number of terms from 25 to 75. After this it keeps on fluctuating with a small fraction. For Eclipse project version 3 and subversions, SVM gives highest accuracy i.e., 93.73 %. In OpenOffice Database access product, accuracy slightly increased from 77.94 to 78.27 % for 25 to 100 terms and then it keeps on fluctuating with the increase in number of terms. For OpenOffice Spreadsheet product, accuracy slightly increased from 81.07 to 81.17 % for 25 to 200 terms. In case of OpenOffice Presentation product, accuracy slightly increased from 79.74 to 80.26 % for 25 to 175 terms. These results show that there is no significant increase in accuracy with the increase in number of terms after 100 terms.

Accuracy of NB classifier is not significant. The reason behind this is the large degree of class overlapping in our datasets as features (terms) we used for priority prediction belongs to more than one priority class. Denil and Trappenberg (2010) shown that SVM is not sensitive to the overlapping problem. It is able to achieve performance comparable to the optimal classifier in the presence of overlapping classes.

Accuracy of K-NN for K = 1 to 5 has been shown in Table 3.

Accuracy of K-NN increases slightly with the increase in value of K from 1 to 5 and decreases slightly with increase in number of terms from 25 to 200. For Eclipse platform product version 2 accuracy for K = 1, is 69.48 % which goes on decreasing to 64.25 % with increase in number of terms from 25 to 200. On the other hand for increasing value of K, the accuracy increased from 69.48 to 73.98 %. For version 3 accuracy for K = 1, is 91.70 % which goes on decreasing to 89.61 % with increase in number of terms from 25 to 200. On the other hand for increasing value of K, the accuracy increased up to 93.70 %. In all cases for all datasets accuracy is above 64 %. The maximum accuracy of the classifier with present datasets is 93.70 % for 25 and top 200 terms with K = 5. Same trend is followed for other datasets. This shows that there is no effect of number of terms on the accuracy in a great extent.

Accuracy of NNET has been shown in Table 4.

Accuracy of Neural Network decreases with increase in number of training cycles from 100 to 300 in almost all datasets. We get maximum number of highest accuracy at 100 training cycles. Accuracy in case of Eclipse version 2 and 3 are 74.75 and 93.77 %for 100 training cycles and 100 terms. For OpenOffice project highest accuracy is 79.06, 81.31 and 81.16 %.

In the answer of research question 1 we concluded that the performance of Neural Network in terms of accuracy is better than SVM which is better than K-NN. Accuracy of Naive bayes is the lowest of all.

Graphical presentation of performance measure accuracy for all the classifier with varying number of terms i.e. from 25 to 200 across all datasets has been shown in Fig. 2, 3, 4, 5, 6.

Figure 2 shows that SVM and NNET accuracy remain invariant with increase in number of terms whereas, K-NN accuracy goes down slightly with increase in number of terms. Figure 3 shows that all the three classifiers: SVM, NNET and K-NN show no significant effect of increase in number of terms on accuracy. From Fig. 4 it's clear that

Table 3 Accuracy measure (in %) of K-NN classifier with K = 1–5

Projects and products	K-NN accuracy (in %)					
	No. of terms	K = 1	K = 2	K = 3	K = 4	K = 5
Eclipse platform product version 2 & subversions	25	69.48	73.14	73.10	73.51	73.98
	50	67.75	75.59	72.34	73.27	73.54
	75	66.33	72.44	72.24	73.03	73.40
	100	65.24	72.37	72.32	73.01	73.45
	125	64.74	72.30	71.85	72.75	73.28
	150	64.34	72.37	71.76	72.87	73.29
	175	64.16	72.25	71.71	72.88	73.23
	200	64.25	72.49	71.71	72.98	73.29
Eclipse platform product version 3 & subversions	25	91.70	93.48	93.47	93.66	93.70
	50	90.43	93.41	93.31	93.60	93.64
	75	90.03	93.34	93.29	93.58	93.64
	100	89.93	93.37	93.25	93.61	93.70
	125	89.72	93.38	93.25	93.63	93.68
	150	89.69	93.43	93.26	93.61	93.68
	175	89.70	93.37	93.36	93.62	93.67
	200	89.61	93.41	93.56	93.69	93.70
OpenOffice Database Access	25	72.37	77.64	77.15	78.02	78.63
	50	71.77	77.51	77.12	77.89	78.00
	75	72.18	77.64	76.85	77.48	78.02
	100	72.32	76.82	76.39	77.12	77.10
	125	71.44	76.99	76.36	76.50	77.07
	150	70.90	76.41	75.76	76.96	77.04
	175	70.79	76.25	75.46	76.66	77.31
	200	71.06	76.58	76.06	77.15	77.29
OpenOffice Presentation	25	76.76	79.62	79.18	79.68	79.56
	50	75.83	79.56	78.98	79.62	79.94
	75	75.60	79.33	78.75	79.01	80.09
	100	74.43	78.86	78.16	78.69	79.39
	125	75.34	79.10	78.25	79.33	79.33
	150	74.61	79.15	78.63	78.92	79.88
	175	74.66	79.18	78.89	78.89	79.71
	200	75.04	79.27	79.07	79.30	79.88
OpenOffice Spreadsheet	25	77.82	80.29	80.40	80.76	81.03
	50	73.32	79.79	79.69	80.44	81.01
	75	71.97	79.47	79.37	79.90	80.31
	100	72.73	79.43	79.33	80.08	80.17
	125	72.61	79.52	79.10	80.29	80.36
	150	72.50	79.56	78.85	80.32	80.52
	175	71.79	79.05	78.23	80.11	80.23
	200	71.98	79.24	78.72	80.32	80.23

Table 4 Accuracy measure (in %) of Neural network for 100–300 training cycles

Projects and products	Neural network training cycles			
	No. of terms	100	200	300
Eclipse platform product version 2 & subversions	25	74.72	74.69	74.66
	50	74.63	74.56	74.54
	75	74.69	74.47	74.37
	100	74.75	74.67	74.73
	125	74.38	74.20	74.03
	150	74.67	74.55	74.28
	175	74.62	74.33	68.18
	200	74.67	74..22	74.27
Eclipse platform product version 3 & subversions	25	93.77	93.75	93.71
	50	93.74	93.73	93.73
	75	93.74	93.65	93.65
	100	93.76	93.73	93.69
	125	93.75	93.74	93.69
	150	93.77	93.71	93.67
	175	93.78	93.63	93.65
	200	93.74	93.69	93.65
OpenOffice Database Access	25	79.03	78.79	78.71
	50	79.06	78.90	78.41
	75	78.79	78.84	78.52
	100	78.71	77.75	71.08
	125	78.54	67.90	77.83
	150	78.46	77.86	73.22
	175	78.41	78.00	77.45
	200	78.27	77.61	77.97
OpenOffice Presentation	25	81.20	81.08	81.28
	50	81.20	81.31	80.85
	75	80.85	80.47	79.94
	100	80.55	78.92	73.79
	125	80.17	79.39	78.39
	150	81.17	79.27	79.15
	175	80.29	79.65	72.94
	200	80.52	79.30	72.62
OpenOffice Spreadsheet	25	81.09	81.03	81.13
	50	81.16	81.13	80.74
	75	80.84	80.48	80.36
	100	80.46	80.53	80.32
	125	80.86	80.38	80.90
	150	80.48	70.11	79.89
	175	80.27	75.44	80.53
	200	80.73	80.63	80.71

accuracy of SVM and NNET remain invariant with increase in number of terms whereas, K-NN accuracy goes down slightly with increase in number of terms. Figure 5 shows that accuracy of NNET goes down with increase in number of terms from 25 to 125. After that it increases for 150 terms and then again decreases with increase in number of terms. K-NN accuracy goes up for 75 terms and then shows a fluctuating behavior with increase in number

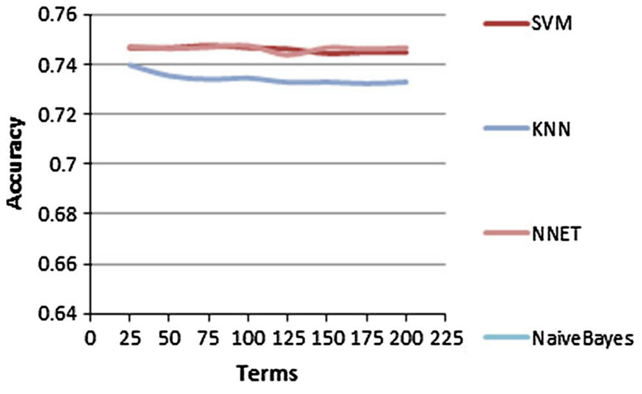

Fig. 2 Accuracy measure for Eclipse Ver2

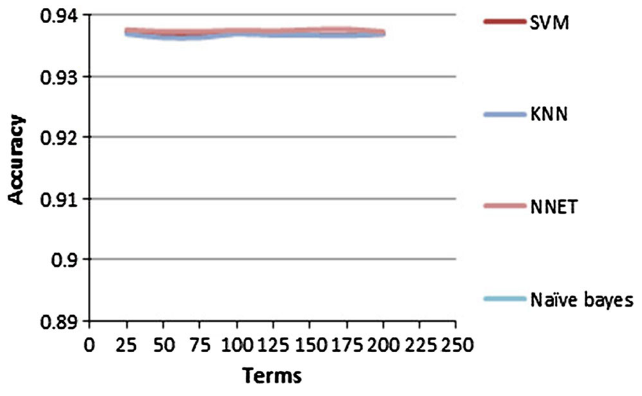

Fig. 3 Accuracy measure for Eclipse Ver3

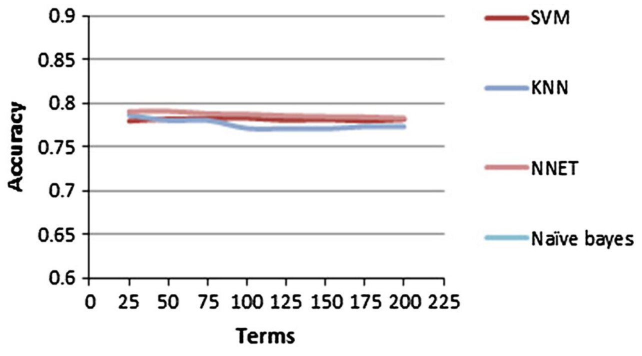

Fig. 4 Accuracy measure for Database access

of terms. SVM accuracy shows a very slow increment with increase in number of terms. From Fig. 6 we see that K-NN and NNET accuracy show a fluctuating behavior with increase in number of terms. SVM accuracy shows an invariant behavior with increase in number of terms.

The accuracy trends with varying number of terms show that the globally distributed users of open source software

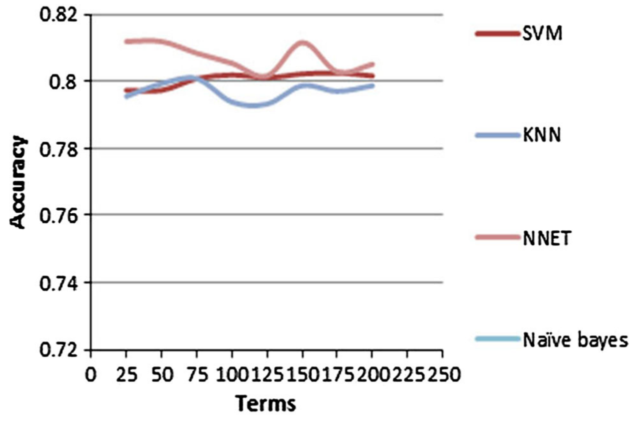

Fig. 5 Accuracy measure for Presentation

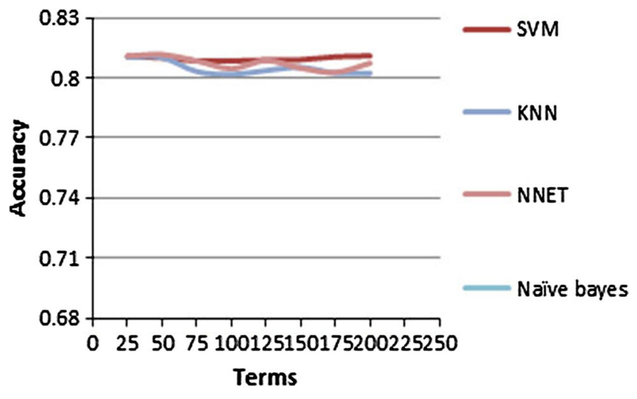

Fig. 6 Accuracy measure for Spreadsheet

use a set of technical terms (words) related with software to report a bug. This set of prominent terms is fixed and is not very large in open source software. For some software it's of 25 terms only and adding more terms to this will not show improvement in accuracy. This shows that bug reporting follows a systematic approach in open source software.

For SVM classifier in Eclipse version 2 dataset, P1 class precision increased from 34.62 to 46.88 % for 25 to 75 terms and after that it starts decreasing for 100 to 200 terms. For P2 class, precision increases from 14.29 to 26.79 % for 25 to 75 terms, then it starts decreasing and again increases to maximum value i.e., 27.03 % at 175 terms. Precision of P3 class increases from 74.94 to 75.24 % till 100 terms and then starts decreasing with increase in number of terms. Precision increases from 65.45 to 72.55 % till 75 terms in case of P4 class. P5 class precision increases from 0.00 to 54.55 % for 25 to 75 terms, after that it decreases. In case of Eclipse version 3 dataset, P1 class precision increased from 0.00 to 73.33 % for 25 to 125 terms and after that it starts decreasing for

150 to 200 terms. For P2 class, precision increases to 17.65 % at 100 terms and then fluctuates and reaches to 18.75 % for 200 terms. Precision of P3 class increases from 93.81 to 93.90 % till 100 terms and then starts fluctuating with increase in number of terms. Precision increases from 55.88 to 58.33 % till 100 terms in case of P4 class. P5 class precision, increases from 0.00 to 50.00 % for 25 to 75 terms, then it fluctuates.

In case of OpenOffice Database Access dataset, P1 class precision increased from 75.00 to 87.50 % for 75 terms and after that it starts fluctuating. For P2 class, maximum precision is 61.04 % at 75 terms. Precision of P3 class increases to 78.88 % till 75 terms. Precision increases to 57.14 % till 200 terms in case of P4 class. P5 class precision, increases from 0.00 to 20.00 % for 50 terms. In case of OpenOffice Spreadsheet dataset, P1 class precision increased to 62.50 % for 75 terms and after that it fluctuates. For P2 class, maximum precision is 73.44 % at 25 terms. Precision of P3 class increases to 81.51 % till 200 terms. Precision increases to 83.33 % till 200 terms in case of P4 class. P5 class precision, increases to 57.14 % for 175 terms. In case of OpenOffice Presentation dataset, P1 class maximum precision is 83.33 % for 25 terms and after that it fluctuates. For P2 class, maximum precision is 69.84 % at 150 terms. Precision of P3 class increases to 80.88 % till 50 terms. Precision increases to 66.67 % till 50 terms in case of P4 class. P5 class precision remains 0.00 % for all number of terms.

For 5 different priority classes and 5 datasets we have 25 maximum precision values. Out of which, for P5 class, in one case precision value remains 0 for all the number of terms. 18 cases of maximum precision we get for range of 25 to 100 terms and 6 cases for range of 125 to 200 terms. This concludes that 25 to 100 terms are sufficient to get the maximum precision for a class.

In case of Eclipse version 2 dataset, P1 class f-measure, increased from 1.91 to 6.63 % for 25 to 100 terms and after that it starts decreasing for 125 to 200 terms. For P2 class, f-measure increases from 0.41 to 2.68 % for 25 to 175 terms. F-measure of P3 class increases from 85.52 to 85.55 % till 75 terms and then starts decreasing with increase in number of terms. F-measure increases from 16.94 to 17.58 % till 75 terms in case of P4 class. P5 class f-measure increases from 0.00 to 5.74 % for 25 to 100 terms, after that it decreases. In case of Eclipse version 3 dataset, P1 class f-measure increased from 0.00 to 6.29 % for 25 to 100 terms and after that it starts decreasing for 125 to 200 terms. For P2 class, f-measure increases from 0.00 to 1.20 % at 100 terms. F-measure of P3 class is 96.76 % at 25 terms and then starts fluctuating with increase in number of terms. F-measure increases from 10.74 to 14.08 % till 200 terms in case of P4 class. P5 class f-measure, increases from 0.00 to 9.34 % for 25 to 100 terms, after that it fluctuates. In case of OpenOffice Database Access dataset, P1 class f-measure is 31.25 % at 25 terms and after that it starts fluctuating. For P2 class, maximum f-measure is 17.13 % at 25 terms. F-measure of P3 class increases to 87.72 % till 100 terms. F-measure increases to 6.22 % till 200 terms in case of P4 class. P5 class f-measure is 4.65 % for 25 terms. In case of OpenOffice Spreadsheet dataset, P1 class f-measure increased to 11.12 % for 75 terms and after that it fluctuates. For P2 class, maximum f-measure is 17.96 % at 100 terms. F-measure of P3 class increases to 89.48 % till 200 terms. F-measure increases to 9.35 % till 200 terms in case of P4 class. P5 class f-measure, increases to 6.61 % for 175 terms. In case of OpenOffice Presentation dataset, P1 class maximum f-measure is 24.66 % for 100 terms and after that it fluctuates. For P2 class, maximum f-measure is 28.61 % at 50 terms. F-measure of P3 class increases to 88.91 % till 175 terms. F-measure increases to 4.30 % till 50 terms in case of P4 class. P5 class f-measure remains 0.00 % for all number of terms.

18 cases of maximum f-measure we get for range of 25 to 100 terms and 6 cases for range of 125 to 200 terms. This concludes that 25 to 100 terms are sufficient to get the maximum f-measure for a class.

For K-NN classifier precision increases with increase in value of K from 1 to 5 and decreases with increase in number of terms from 25 to 200 across all datasets. F-measure shows fluctuating behavior with increase in number of terms, but shows an increasing trend for increase in value of K from 1 to 5. Maximum F-measure for P3 class is 85.16 for $K = 5$ and 25 terms. For value of K equal to 4 and 5, we got maximum highest values of precision and F-measures with less number of terms.

For Neural Network classifier, precision increases with increase in number of terms in most of the cases. Highest precision for Eclipse version 2 is 39.29, 36.36, 75.26, 73.47 and 50 % for P1 to P5 class. Highest precision for Eclipse version 3 is 40.00, 16.67, 93.86, 62.50 and 0 % for P1 to P5 class. Precision for OpenOffice Database Access is 100.00, 64.18, 82.12, 100 and 0 % for P1 to P5 class. Precision for OpenOffice Spreadsheet is 63.64, 53.33, 82.60, 100 and 20 % for P1 to P5 priority level. Highest precision for OpenOffice Presentation is 100.00, 60.80, 85.52, 50 and 0 % for P1 to P5 priority level. We get highest precision, recall and f-measure for P3 class.

For NB classifier in eclipse version 2 dataset, P1 class maximum precision is 20.40 % at 50 terms. For P2 class, precision is 26.68 % for 50 terms. Precision of P3 class is 89.33 % at 75 terms. Precision is 8.21 % at 25 terms in case of P4 class. P5 class precision increases from 2.35 to

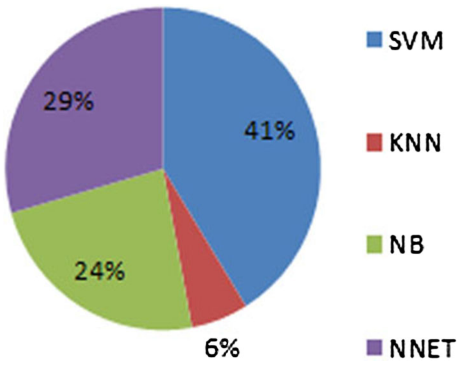

Fig. 7 Classifiers performance using f-measure

2.72 % for 25 to 100 terms, after that it decreases. In case of Eclipse version 3 dataset, P1 class maximum precision is 2.56 % at 200 terms. For P2 class, precision is 12.48 % for 150 terms. Precision of P3 class is 98.33 % at 75 terms. Precision is 8.91 % at 25 terms in case of P4 class. P5 class precision increases from 0.52 to 0.66 % for 25 to 200 terms. In case of OpenOffice Database Access dataset, P1 class maximum precision is 19.39 % at 25 terms. For P2 class, precision is 36.94 % for 25 terms. Precision of P3 class is 93.36 % at 100 terms. Precision is 18.46 % at 100 terms in case of P4 class. P5 class precision increases from 1.11 to 1.25 % for 25 to 125 terms. In case of OpenOffice Spreadsheet dataset, P1 class maximum precision is 5.15 % at 25 terms. For P2 class, precision is 34.08 % for 50 terms. Precision of P3 class is 89.66 % at 150 terms. Precision is 34.04 % at 25 terms in case of P4 class. P5 class precision increases from 2.35 to 3.33 % for 25 to 200 terms. In case of OpenOffice Presentation dataset, P1 class maximum precision is 7.07 % at 50 terms. For P2 class, precision is 50.00 % for 25 terms. Precision of P3 class is 91.51 % at 75 terms. Precision is 4.71 % at 75 terms in case of P4 class. P5 class precision increases from 1.15 to 1.41 % for 25 to 100 terms.

19 cases of maximum precision we get for range of 25 to 100 terms and 6 cases for range of 125 to 200 terms. This concludes that 25 to 100 terms are sufficient to get the maximum precision for a class.

For NB, in case of Eclipse version 2 dataset, class P1 f-measure, is 24.90 % at 75 terms. For P2 class, f-measure is 8.48 % for 100 terms. F-measure of P3 class is 7.74 % at 200 terms. F-measure is 12.72 % for 75 terms in case of P4 class. P5 class f-measure increases from 2.35 to 5.25 % for 25 to 100 terms, after that it decreases. In case of Eclipse version 3 dataset, P1 class f-measure, is 4.73 % at 200 terms. For P2 class, f-measure is 11.48 % for 175 terms. F-measure of P3 class is 11.82 % at 200 terms. F-measure is 13.77 % for 25 terms in case of P4 class. P5 class f-measure increases from 0.52 to 1.31 % for 25 to 100 terms, after that it decreases. In case of OpenOffice Database Access dataset, P1 class f-measure, is 26.55 % at 25 terms. For P2 class, f-measure is 24.75 % for 150 terms. F-measure of P3 class is 19.74 % at 200 terms. F-measure is 23.55 % for 100 terms in case of P4 class. P5 class f-measure increases from 1.11 to 2.44 % for 25 to 125 terms, after that it decreases. In case of OpenOffice Spreadsheet dataset, P1 class f-measure, is 8.58 % at 25 terms. For P2 class, f-measure is 31.26 % for 200 terms. F-measure of P3 class is 9.94 % at 200 terms. F-measure is 19.59 % for 150 terms in case of P4 class. P5 class f-measure increases from 2.35 to 6.08 % for 25 to 200 terms. In case of OpenOffice Presentation dataset, P1 class f-measure is 12.10 % at 75 terms. For P2 class, f-measure is 31.89 % for 100 terms. F-measure of P3 class is 7.74 % at 200 terms. F-measure is 7.26 % for 100 terms in case of P4 class. P5 class f-measure increases from 1.15 to 2.76 % for 25 to 100 terms, after that it decreases.

Table 5 Accuracy (in %) of cross validated projects

Training vs. testing dataset	SVM	K-NN	NNET	Naive Bayes
V2 vs. V3	92.05	90.38	93.60	3.66
V2 vs. DB	77.07	76.03	77.23	3.55
V2 vs. SST	79.77	79.27	80.23	4.94
V2 vs. PPT	77.84	77.14	78.02	4.29
DB vs. V2	74.30	73.00	74.05	8.00
SST vs. V2	74.47	72.92	74.51	9.57
PPT vs. V2	74.25	73.28	74.43	8.07
V3 vs. DB	77.26	77.37	77.37	3.52
V3 vs. SST	80.32	80.32	80.34	4.85
V3 vs. PPT	78.28	78.34	80.23	5.51
DB vs. V3	93.44	90.94	93.37	6.28
SST vs. V3	93.55	91.32	93.73	4.79
PPT vs. V3	93.45	91.70	93.56	7.13
DB vs. SST	79.64	78.83	79.87	10.15
SST vs. DB	77.04	77.04	77.15	6.42
PPT vs. DB	77.01	77.10	76.90	13.43
DB vs. PPT	77.96	78.31	75.23	9.18
SST vs. PPT	77.96	78.51	78.28	9.18
PPT vs. SST	79.52	79.50	79.69	11.24

12 cases of maximum f-measure we get for range of 25 to 100 terms and 13 cases for range of 125 to 200 terms. This concludes that 125 to 200 terms are sufficient to get the maximum f-measure.

A Graphical presentation for performance of the different classifiers across all datasets on the basis of f-measure has been shown in Fig. 7.

We have counted maximum F-measures for each priority class for each technique. After adding all these values

Table 6 Accuracy (in %) of SVM model in different datasets

Training dataset	Testing dataset				
	V2	V3	DB	PPT	SST
V2	–	92.05	77.07	77.84	79.77
V3	74.32	–	77.26	78.28	80.32
DB	74.30	93.44	–	77.96	79.64
PPT	74.25	93.45	77.01	–	79.52
SST	74.48	93.55	77.04	77.96	–
V2 + V3	–	–	77.37	77.35	80.04
V2 + DB	–	93.12	–	78.08	80.02
V2 + PPT	–	93.26	77.07	–	79.77
V2 + SST	–	93.13	76.74	78.05	–
V3 + DB	74.44	–	–	78.05	80.23
V3 + PPT	74.37	–	77.26	–	79.96
V3 + SST	74.44	–	77.31	77.99	–
DB + PPT	73.95	93.38	–	–	79.68
DB + SST	74.13	93.58	–	77.61	–
PPT + SST	74.05	93.32	76.49	–	–
V2 + V3 + DB	–	–	–	78.22	80.23
V2 + V3 + PPT	–	–	77.20	–	80.19
V2 + V3 + SST	–	–	77.29	77.90	–
V2 + DB + PPT	–	93.19	–	–	80.06
V2 + DB + SST	–	93.36	–	78.10	–
V2 + PPT + SST	–	93.19	77.04	–	–
V3 + DB + PPT	74.45	–	–	–	80.06
V3 + DB + SST	74.32	–	–	78.25	–
V3 + PPT + SST	74.37	–	76.82	–	–
DB + PPT + SST	73.89	93.44	–	–	–

Table 7 Accuracy (in %) of K-NN model in different datasets

Training dataset	Testing dataset				
	V2	V3	DB	PPT	SST
V2	–	90.38	76.03	77.14	79.27
V3	74.53	–	77.37	78.31	80.32
DB	73.00	90.94	–	78.31	78.83
PPT	73.28	91.70	77.10	–	79.50
SST	72.92	91.32	77.04	78.51	–
V2 + V3	–	–	77.23	78.31	80.34
V2 + DB	–	93.22	–	77.67	79.98
V2 + PPT	–	93.24	77.07	–	80.00
V2 + SST	–	93.35	77.10	77.93	–
V3 + DB	74.52	–	–	78.22	80.34
V3 + PPT	74.52	–	77.29	–	80.06
V3 + SST	74.50	–	77.18	78.40	–
DB + PPT	73.77	93.33	–	–	79.66
DB + SST	73.63	93.41	–	77.35	–
PPT + SST	73.95	93.60	77.20	–	–
V2 + V3 + DB	–	–	–	78.31	80.42
V2 + V3 + PPT	–	–	77.34	–	80.04
V2 + V3 + SST	–	–	77.37	78.37	–
V2 + DB + PPT	–	93.15	–	–	80.13
V2 + DB + SST	–	93.29	–	78.10	–
V2 + PPT + SST	–	93.37	77.37	–	–
V3 + DB + PPT	74.56	–	–	–	80.23
V3 + DB + SST	74.53	–	–	78.25	–
V3 + PPT + SST	74.53	–	77.23	–	–
DB + PPT + SST	73.93	93.50	–	–	–

for each technique, we plotted the graph for f-measure. Result shows that performance in terms of f-measure is 41, 29, 24 and 6 % for SVM, NNET, NB and K-NN respectively.

In answer of research question 2, we have concluded that 100 top terms are sufficient to get optimum performance in terms of accuracy, precision and f-measure across all machine learning techniques. We found that for K = 5 for K-NN and 100 training cycles for Neural Network we get optimum performance.

Scenario 2 From scenario 1 we concluded that K-NN and NNET give optimum results for K = 5 and training cycles = 100. So, we ran Scenario 2 for these values. Table 5 summarizes accuracy for cross project priority prediction for 4 different techniques.

Cross project validation for Eclipse version 2 as training set and version 3 as testing set gives 90.38, 92.05 and 93.60 % accuracy for K-NN (K = 5), SVM and Neural Network with 100 terms. It is clear from the empirical evidence that cross validation works well within same domain. We also found that cross project validation is bidirectional with accuracy above 72 %.

In answer of the research question 3, we concluded that cross project validation across domain gives accuracy above 72 % for SVM, K-NN and Neural Network. We found that cross projects validation working well with significant accuracy in priority prediction.

The value of f-measure for machine learning techniques namely SVM, K-NN, Neural Network and NB varies in the range of 85.25 to 96.67, 84.49 to 95.69, 84.97 to 96.76 and 0.83 to 5.21 % across all datasets for priority level 3. Due to insufficient number of reports in case of priority level 1, 2, 4 and 5 we are not getting desired performance of different machine learning techniques. This is one of the problems in multi-class prediction.

Scenario 3 Table 6, 7, 8, 9 summarize the accuracy for different training candidates for the studied learning techniques.

From these tables, the accuracy of V3 dataset has been found more than 90 % in all the techniques except NB classifier irrespective of the training datasets.

Table 8 Accuracy (in %) of NB Model in different datasets

Training dataset	Testing dataset				
	V2	V3	DB	PPT	SST
V2	–	3.66	3.55	4.29	4.94
V3	2.11	–	3.52	5.51	4.85
DB	8.00	6.28	–	9.18	10.15
PPT	8.07	7.13	13.43	–	11.24
SST	9.57	4.79	6.42	9.18	–
V2 + V3	–	–	1.20	1.08	2.27
V2 + DB	–	0.51	–	1.14	2.44
V2 + PPT	–	0.51	1.17	–	2.35
V2 + SST	–	0.49	1.45	1.14	–
V3 + DB	2.65	–	–	1.22	2.40
V3 + PPT	3.25	–	6.61	–	6.18
V3 + SST	2.10	–	6.50	1.22	–
DB + PPT	2.23	0.51	–	–	3.26
DB + SST	7.92	1.30	–	2.39	–
PPT + SST	2.24	0.51	2.10		–
V2 + V3 + DB	–	–	–	1.52	2.39
V2 + V3 + PPT	–	–	1.56	–	2.33
V2 + V3 + SST	–	–	2.32	1.49	–
V2 + DB + PPT	–	0.59	–	–	6.11
V2 + DB + SST	–	0.50	–	2.36	–
V2 + PPT + SST	–	0.51	1.39	–	–
V3 + DB + PPT	2.07	–	–	–	6.26
V3 + DB + SST	7.99	–	–	1.72	–
V3 + PPT + SST	2.07	–	1.50	–	–
DB + PPT + SST	2.31	0.62	–	–	–

In answer of the research question 4, we conclude that for combined training datasets we get accuracy which is above 73 % for all cases which is not a significant improvement over single training dataset. In NB we are getting very low accuracy.

The value of f-measure for machine learning techniques namely SVM, K-NN, Neural Network and NB varies in the range of 85.08 to 96.68, 84.92 to 96.70, 85.18 to 96.77 and 0.00 to 1.27 % across all data sets for priority level 3. Due to insufficient number of reports in case of priority level 1, 2, 4 and 5 we are not getting desired performance of different machine learning techniques. This is one of the problems in multi-class prediction where we have imbalance data sets.

7 Threats to validity

Following are the factors that affect the validity of our approach:

Table 9 Accuracy (in %) of NNET model in different datasets

Training dataset	Testing dataset				
	V2	V3	DB	PPT	SST
V2	–	93.60	77.23	78.02	80.23
V3	74.56	–	77.37	80.23	80.34
DB	74.05	93.37	–	75.23	79.87
PPT	74.43	93.56	76.90	–	79.69
SST	74.51	93.73	77.15	78.28	–
V2 + V3	–	–	77.37	78.37	80.34
V2 + DB	–	93.74	–	78.37	80.34
V2 + PPT	–	93.74	77.37	–	80.34
V2 + SST	–	93.73	77.37	78.37	–
V3 + DB	74.43	–	–	78.16	80.29
V3 + PPT	74.54	–	77.29	–	80.34
V3 + SST	74.50	–	77.37	78.37	–
DB + PPT	74.09	93.53	–	–	79.71
DB + SST	74.44	93.66	–	77.90	–
PPT + SST	74.54	93.73	77.29	–	–
V2 + V3 + DB	–	–	–	78.37	80.34
V2 + V3 + PPT	–	–	77.37	–	80.34
V2 + V3 + SST	–	–	77.37	78.37	–
V2 + DB + PPT	–	93.74	–	–	80.34
V2 + DB + SST	–	93.71	–	78.37	–
V2 + PPT + SST	–	93.74	77.37	–	–
V3 + DB + PPT	74.54	–	–	–	80.32
V3 + DB + SST	74.56	–	–	78.37	–
V3 + PPT + SST	74.54	–	77.37	–	–
DB + PPT + SST	74.52	93.71	–	–	–

– shows that no validation has been done because testing dataset is already a part of training dataset and it will lead a biased result

7.1 Construct validity

The accuracy of classifier depends on the summary text; if summary does not contain appropriate terms to be learned then the result will be in wrongly predicted class. Number of bug reports in P3 class is more than number of bug reports in other classes. This makes classifier biased towards P3 class.

7.2 Internal validity

We have only taken summary feature of bug report. Other features can also be considered for prediction.

7.3 External validity

We have considered Eclipse and OpenOffice projects which are open source. We can consider closed source software also.

7.4 Reliability

Rapid Miner (http://www.rapid-i.com/) tool has been used in this paper for data pre-processing, model building and tenfold cross validation. The increasing use of Rapid Miner tool in data mining community confirms the reliability of the tool.

8 Conclusion

In response to the Scenario 1 and subsequent experimental setup, following conclusions have been drawn with the variation of number of terms from 25 to 200:

- SVM, Neural Network and K-NN techniques are applicable to predict the priority level of reported bug in open source projects.
- SVM and Neural Network give overall higher accuracy in comparison of K-NN and Naive Bayes for all datasets.
- SVM performance in terms of accuracy had no significant improvement with increase in number of terms.
- SVM performance in terms of precision and f-measure slightly improved by increasing the number of terms from 25 to 100.We found fewer cases of increase in range of 125 to 200 terms.
- K-NN performance in terms of accuracy, precision and f-measure slightly decreased by increasing the number of terms from 25 to 200 and increased by increasing the value of K from 1 to 5.
- NB performance in terms of accuracy increases with increase in number of terms from 25 to 200. Its precision increases in range of 25 to 100 terms. We found fewer cases of increase in range of 125 to 200 terms.
- Neural Network gives higher accuracy for 100 training cycles, which decreases with increase in training cycles in almost all cases. We get highest precision, recall and f-measure for P3 priority level.
- Recall of SVM is high for class having large number of reports and low for class having less number of reports. Whereas in Naive Bayes, recall is high for class having less number of reports and low for class having large number of reports.
- Machine learning techniques performed well in terms of precision only in case of priority level 3 due the fact that it has sufficient number of bug reports.
- The value of performance measure, precision has shown significant improvement with increase in number of reports across all the techniques. The value of precision increases from 25 to 100 terms, after that it starts decreasing with increase in number of terms.
- Automation of bug triage by using priority prediction will save time and resources. It will help in solving higher priority bugs within given time period.
- We found that the SVM and Neural Network are better than K-NN and K-NN is better than NB.

In response to the Scenario 2 and subsequent experimental setup, following conclusions have been drawn:

- The accuracy in cross project context is better than within project.
- Cross project validation for different cases are working with accuracy level more than 72 % except for NB learner.

Finally we concluded that historical data of other projects developed in open source environment is better priority predictor and priority prediction in cross project context is working well.

As a result of scenario 3, we found that combined training datasets from other projects for training working well but does not show significant improvement over single training data set. In the non-availability of historical data, combined training datasets from other projects provide an acceptable performance. In our case we get accuracy above 73 % for all cases except NB learner.

In future, we will work on the following agendas:

- The current empirical study can be carried out on more open source and closed source projects to validate priority predictions in cross project context.
- The study can be extended to determine the optimum number of bug reports as well as optimum number of features/terms required to get the best performance.
- Impact of imbalance data on performance of the classifier can be considered.
- Training data selection method used in this paper gives an exponential increase in training datasets to be considered to find best one. We will try to find a similarity measure between training and testing projects using fuzzy logic.

References

Anvik J (2006) Automating bug report assignment. In: Proceedings of the 28th International Conference on Software Engineering, Shanghai, China, pp 937–940

Anvik J, Murphy GC (2011) Reducing the effort of bug report triage: Recommenders for development-oriented decisions. ACM Transact Softw Eng Methodol 20(3):10

Anvik J, Hiew L, Murphy GC (2006) Who should fix this bug? In: Proceedings of the 28th International conference on Software Engineering, Shanghai, China, pp 361–370

Canfora G, Cerulo L (2006) Supporting change request assignment in open source development. In: Proceedings of the ACM Symposium on Applied Computing, Dijon, France, pp 1767–1772

Chaturvedi KK, Singh VB (2012) Determining bug severity using machine learning techniques. In: Proceedings of the International Conference on Software Engineering (CONSEG), Indore, India, pp 378–387

Denil M, Trappenberg T (2010) Overlap versus Imbalance. In: Proceedings of the 23rd Canadian Conference on Advances in Artificial Intelligence, Springer, Verlag Berlin, Heidelberg, pp 220–231

He Z, Shu F, Yang Y, Li M, Wang Q (2012) An investigation on the feasibility of cross-project defect prediction. In: Automated Software Engineering, pp 167–199

Kanwal J, Maqbool O (2010) Managing open bug repositories through bug report prioritization using SVMs. In: Proceedings of the International Conference on Open-Source Systems and Technologies, Lahore, Pakistan

Kanwal J, Maqbool O (2012) Bug prioritization to facilitate bug report triage. J Comput Sci Technol 27(2):397–412

Kim S, Whitehead J (2006) How long did it take to fix bugs? In: Proceedings of the International Workshop on Mining Software Repositories, Shanghai, China, pp 173–174

Lamkanfi A, Demeyer S, Gigery E, Goethals B (2010) Predicting the severity of a reported bug. In: Proceedings of the 7th Working Conference on Mining Software Repositories, Cape Town, South Africa, pp 1–10

Marks L, Zou YA, Hassan E (2011) Studying the fix-time for bugs in large open source projects. In: Proceedings of the 7th International Conference on Predictive Models in Software Engineering, Banff, Article No. 11

Menzies T, Marcus A (2008) Automated severity assessment of software defect reports. In: Proceedings of the International Conference on Software Maintenance, IEEE, pp 346–355

Mierswa I, Wurst M, Klinkenberg R, Scholz M, Euler T (2006) YALE: Rapid Prototyping for Complex Data Mining Tasks. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06). http://www.rapid-i.com

Porter M (2008) An algorithm for suffix stripping. Program 14(3):130–137

Sharma M, Bedi P, Chaturvedi KK, Singh VB (2012) Predicting the priority of a reported bug using machine learning techniques and cross project validation. In: Proceedings of the 12th International Conference on Intelligent Systems Design and Applications (ISDA) Kochi, India, pp 539–545

Sharma M, Kumari M, Singh VB (2013) Understanding the meaning of bug attributes and prediction models. In: Proceedings of the 5th IBM Collaborative Academia Research Exchange Workshop, I-CARE 2013, Article No. 15, ACM, New York, USA

Tamrawi A, Nguyen T, Al-Kofahi J, Nguyen TN (2011) Fuzzy set based automatic bug triaging. In: Proceedings of the 33rd International conference on Software Engineering (NIER Track), Miami, USA, pp 884–887

Turhan B, Menzies T, Bener AB, Stefano JD (2009) On the relative value of cross-company and within-company data for defect prediction. Empir Softw Eng. doi:10.1007/s10664-008-9103-7

Weib C, Premraj R, Zimmermann T, Zeller A (2007) Predicting effort to fix software bugs. In: Proceedings of the Workshop on Software Reengineering, Bad Honnef, Germany

Yu L, Tsai W, Zhao W, Wu F (2010) Predicting defect priority based on neural networks. In: Proceedings of the 6th International Conference on Advanced Data Mining and Applications, Wuhan, China, pp 356–367

Zimmermann T, Nagappan N, Gall H (2009) Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In: Proceedings of the 7th Joint Meeting of the European Software Enginnering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, pp 91–100