

Entropy Based Machine Learning Models for Software Bug Severity Assessment in Cross Project Context

Madhu Kumari[1], Ujjawal Kumar Singh[2], and Meera Sharma[3(✉)]

[1] Delhi College of Arts & Commerce, University of Delhi, Delhi, India
mesra.madhu@gmail.com
[2] Amity School of Engineering and Technology, GGSIP University, Delhi, India
ujjawalkumarsingh99@gmail.com
[3] Swami Shraddhanand College, University of Delhi, Delhi, India
meerakaushik@gmail.com

Abstract. There can be noise and uncertainty in the bug reports data as the bugs are reported by a heterogeneous group of users working across different countries. Bug description is an essential attribute that helps to predict other bug attributes, such as severity, priority, and time fixes. We need to consider the noise and confusion present in the text of the bug report, as it can impact the output of different machine learning techniques. Shannon entropy has been used in this paper to calculate summary uncertainty about the bug. Bug severity attribute tells about the type of impact the bug has on the functionality of the software. Correct bug severity estimation allows scheduling and repair bugs and hence help in resource and effort utilization. To predict the severity of the bug we need software project historical data to train the classifier. These training data are not always available in particular for new software projects. The solution which is called cross project prediction is to use the training data from other projects. Using bug priority, summary weight and summary entropy, we have proposed cross project bug severity assessment models. Results for proposed summary entropy based approach for bug severity prediction in cross project context show improved performance of the Accuracy and F-measure up to 70.23% and 93.72% respectively across all the machine learning techniques over existing work.

1 Introduction

In software development life cycle, bug reporting and fixing is a continuous and iterative activity [1]. A large number of bugs are reported on bug tracking systems by different users, developers and staff members located at different geographical locations in a distributed environment. Bug severity is one of the most important bug attributes which tells about its extent of impact on the functionality of the software. Bug severity is labeled in seven classes from 1 to 7, namely "Blocker", "Critical", "Major", "Normal", "Minor", "Trivial" and "Enhancement". The automated bug severity prediction is useful in resource allocation and bug fix scheduling. It also assists the priority assignment for the bug. Bug severity prediction needs training data, i.e. the history of

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12254, pp. 939–953, 2020.
https://doi.org/10.1007/978-3-030-58817-5_66

the software to train the classifier. But it is not easy to get such data always as some projects may be new with very less of no history of bug data. In such situation, we can use history of bug data from other software projects for training purpose [2, 4–6]. Bugs are reported by users with different levels of understanding and knowledge about the software working which may result in noise and uncertainty in different bug attributes entered. This noise and uncertainty present in training data may degrade the performance of automated bug severity assessment and hence need to be considered during prediction process. Bug summary attribute (the brief description of the bug) has been used for bug severity prediction in this paper. No attempt has been made in literature to consider uncertainty in bug summary in cross project context for bug severity prediction. The contribution of this paper is cross project severity prediction models based on summary entropy in addition to priority and summary weight using "k-Nearest Neighbors (k-NN)", "Support Vector Machine (SVM)", and "Naïve Bayes (NB)". The proposed models result in improved performance when compared with summary based cross project bug severity assessment models [6].

The remaining paper is structured as follows: Sect. 2 describes the review of related work. Section 3 contains the brief of bug reports and its pre-processing. Section 4 deals with data collection and model building required to perform the analysis. Results have been documented in Sect. 5. The conclusion of the paper has given in Sect. 6.

2 Related Work

Bug severity prediction helps in assigning bug priority, fix time prediction and resources allocation. Many bug summary based severity assessment models have been proposed in literature [7–12]. Different authors compared the performance of different machine learning techniques for bug severity assessment [19–21].

An attempt has been made to propose bug summary based cross project severity prediction models using "SVM", "NB" and "k-NN" [6]. Authors also identified the best training candidates for a project. Bug summary based cross project priority prediction models have been proposed by [2, 4] using "SVM", "NB", "k-NN" and "NNET".

Entropy based measure has been used to predict the bugs lying dormant in the software [14, 15]. Recently entropy based measures have been used to handle the uncertainty during the prediction of priority and severity of the reported bug [3, 13].

To our knowledge, no work has been done for considering the uncertainty and noise present in bug summary data that can affect the performance of prediction models in cross project context. In this paper, we have measured the uncertainty in bug summary by using entropy based measures for cross project severity prediction. In addition to summary entropy, we have considered bug priority and summary weight to assess bug severity in cross project context. We have compared our proposed summary entropy based cross project bug severity assessment models with [6] and found improvement in the performance of the classifiers.

3 Bug Reports and Pre-processing

A bug report contains the information about bug in the form of different attributes reported by the users and the developers use this information to fix the bug. In this section we have discussed different bug attributes and two derived attributes summary weight and summary entropy used in bug severity prediction.

We have taken bug priority and two derived bug attributes: summary weight [4] and summary entropy to predict severity in cross project context.

Bug priority and severity are categorical attributes, whereas summary weight and summary entropy are continuous attributes. Bug priority determines the importance of a bug in the presence of others. Bugs are prioritized by P1 level, i.e. the most important to P5 level, i.e. the least important.

Bug severity tells about the extent of bug's impact on software functionality. Eclipse project define the seven levels of severity, namely "Blocker", "Critical", "Major", "Normal", "Minor", "Trivial" and "Enhancement". Throughout this analysis, we have not included bugs with "Normal" and "Enhancement" severity levels because "Normal" is the default standard stated in the reports submitted, and "Enhancement" does not reflect actual bug reports. The severity weights and levels as mentioned in Table 1 (IEEE std 92, 1989) have been defined by IEEE Standard Classification Levels [16]. "Blocker" and "Critical" are most severe severity levels, "Major" is medium severity level and "Minor", "Trivial" are minor severity levels.

Table 1. Severity levels categories [16]

From the IEEE Standard Severity Classification Levels	Severity Weight	Severity Level
Blocker, Critical	10	Most Severe
Major	3	Medium
Minor, Trivial	1	Minor

Summary weight attribute is extracted from the bug summary provided by the numerous users. We pre-processed the bug summary in RapidMiner tool [18] to compute the summary weight of a reported bug, with the steps of text mining: "Tokenization", "Stop Word Removal", "Stemming to base stem", "Feature Reduction" and "Info Gain" [6].

We assume that the bug reports, i.e. different bug attributes, reported in software bug repositories are trustworthy during bug triaging process. In reality, the bug reports data is not trustworthy in terms of various aspects like integrity, authenticity and trusted origin as the bugs are reported by users who may or may not have proper knowledge of the software. It may result in uncertainty in reported bug data. Without proper handling of these uncertainties in different bug attributes, the performance of learning strategies used for different bug attributes prediction can be significantly reduced.

The validation of cross project is a key concern in empirical software engineering where we train the classifiers with historical data of projects other than the testing projects. In literature, researchers have made attempts for cross project bug summary based severity assessment [6]. But no attempt has been made to handle uncertainty in bug summary in cross project context for bug severity assessment.

We have proposed summary entropy based measure to build the classifier for bug severity prediction to handle uncertainty in cross project context. We have calculated the summary entropy for model building using Shannon's entropy [17]. Shannon's entropy, S is defined as:

$$S = -p_i \log_2 p_i$$

In the case of summary entropy, p is calculated as:

$$p_i = \frac{total\ number\ of\ occurences\ of\ terms\ in\ i^{th}\ bug\ report}{total\ number\ of\ terms}$$

To rationalize the effect of the severity, we multiplied entropy with 10 for "Blocker" and "Critical" severity level bugs, 3 for "Major" severity level bugs and 1 for "Minor" and "Trivial" severity level bugs as given in Table 1 [16].

The cross project bug severity model has been shown in Fig. 1.

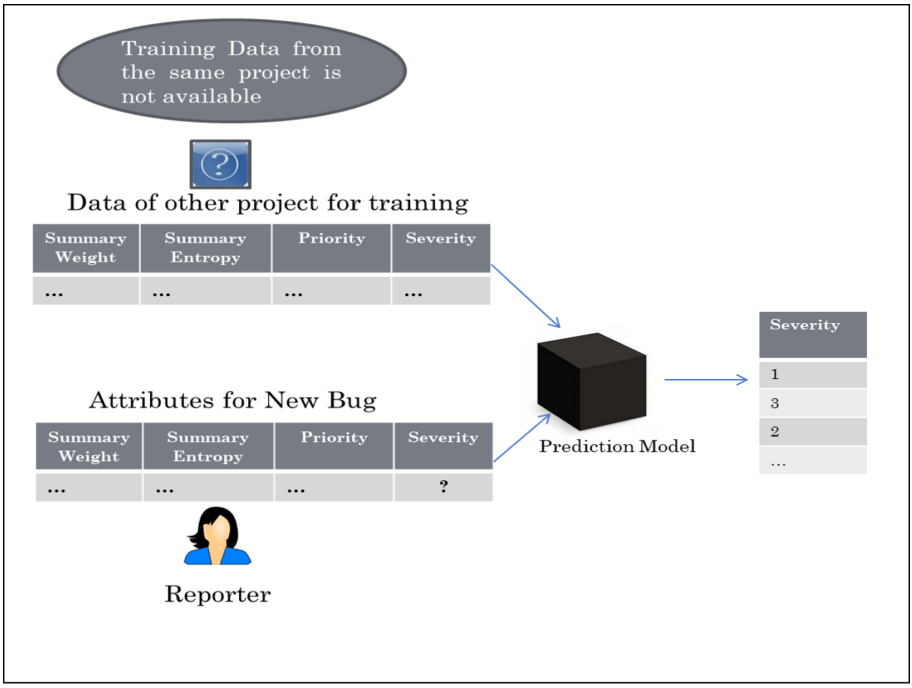

Fig. 1. Cross project bug severity prediction

4 Methodology

In this section, we briefly described the data collection and model building for summary entropy based cross project bug severity assessment.

4.1 Data Collection

The empirical validation has been conducted on different products, namely "CDTDebug (CD)", "EclipseDebug (Deb)", "EclipseJDTUI (TUI)", "EclipseSWT (SWT)", "EclipseUI (UI)", "IDEPlatform (IDE)", and "JDTUI (TUI2)" of Eclipse project (http://bugs.eclipse.org/bugs/) to assess cross project bug severity. Table 2 shows the severity level wise number of bug reports across different products.

Table 2. Severity wise Bug Reports in Eclipse Projects [6]

Projects	1 (Blocker)	2 (Critical)	3 (Major)	4 (Minor)	5 (Trivial)	Total
CD	25	25	122	53	8	233
Deb	23	97	213	72	39	444
TUI	23	81	282	281	81	748
SWT	71	161	298	64	36	630
UI	28	124	401	327	109	989
IDE	23	75	267	148	85	598
TUI2	1	24	118	204	56	403

4.2 Model Building and Experimental Setup

We have developed summary entropy based models using different classifiers, namely "k-NN", "SVM" and "NB" for cross project bug severity assessment by taking priority and summary weight. The empirical evaluation has been validated on 7 products of the Eclipse project. Number of cross fold validations is taken as 10 with stratified sampling for different classification techniques. We have validated our proposed approach and compared it with state of art [6] using performance measures, namely Accuracy and F-measure.

The experimental setup of severity prediction in cross project context developed in RapidMiner tool [18] has been shown in Fig. 2.

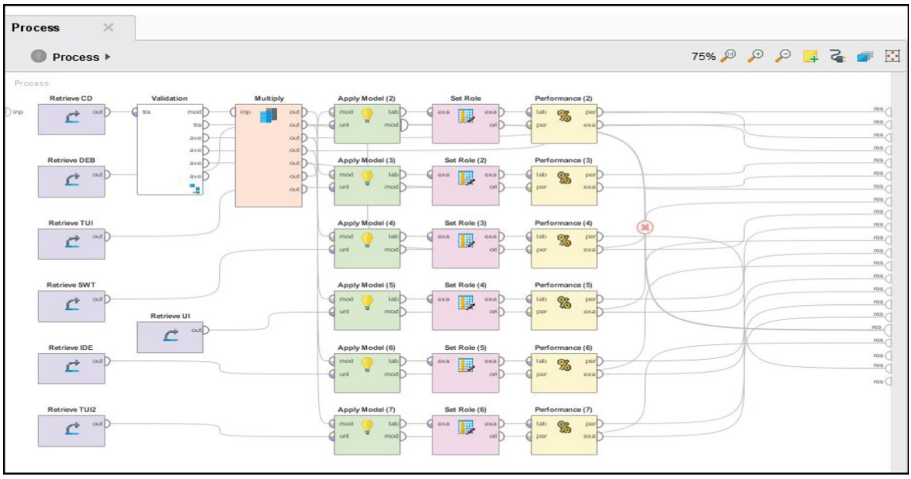

Fig. 2. Experimental Setup for Cross project bug severity prediction in RapidMiner

The parameter values used for tuning the classifier parameters, namely "k-Nearest Neighbor (k-NN)", "Support Vector Machine (SVM)" and "Naïve Bayes (NB)" have been shown in Table 3.

Table 3. Parameters Optimized for different Classifiers

Classifier	Parameters
NB	laplace_correction
k-NN	k
SVM	C (cost) G (gamma)

Using "Optimize Parameters (Grid)" operator in the RapidMiner tool, we obtained optimal parameter values. Table 4 shows the parameters optimized for each classifier.

Table 4. Optimal Parameter Values for Eclipse products

Eclipse products	NB	k-NN	SVM	
	laplace_correction	k	C	G
CD	False	8	1	1
Deb	False	8	1	1
TUI	False	13	3	3
SWT	True	2	2	2
UI	True	6	4	4
IDE	True	20	1	1
TUI2	True	4	3	3

5 Results and Discussion

We have proposed summary entropy based models using different classifiers, namely, "k-Nearest Neighbors (k-NN)", "Support Vector Machine (SVM)" and "Naive Bayes (NB)" for cross project bug severity prediction. We have compared the proposed entropy based approach with Singh et al. [6]. We have taken the same datasets and techniques as taken by the authors in [6] to predict bug severity. Singh et al. [6] considered the F-measure performance of different classifiers only for "Major" severity class, since fewer bug reports for other severity class than the "Major" severity class. This results in low performance for these severity classes. In order to compare with state of art literature [6] we have also considered the F-measure performance for "Major" severity class. Tables 5, 6 and 7 show the F-measure performance for "Major" severity class for different classifiers, namely "k-NN", "SVM" and "NB" respectively. Tables 8, 9 and 10 show the Accuracy of different classifiers, namely "k-NN", "SVM" and "NB" for different testing projects. Across Tables 5, 6, 7, 8, 9 and 10 '–' indicates that no analysis was performed on this particular combination of testing and training dataset, since the training and testing data sets are similar.

Table 5. k-NN F-measure (%) for "Major" severity class

Training Projects	Testing Projects						
	CD	Deb	TUI	SWT	UI	IDE	TUI2
CD	–	93.63	55.65	51.54	49.00	77.15	69.70
Deb	96.27	–	49.39	33.10	33.33	64.45	89.06
TUI	91.21	89.54	–	88.16	91.20	93.72	71.67
SWT	61.87	53.69	86.21	–	90.25	91.04	31.40
UI	64.19	58.94	95.04	96.71	–	95.72	38.57
IDE	96.77	90.00	88.56	87.97	86.90	–	62.88
TUI2	91.27	97.25	54.88	27.71	40.95	58.71	–

Table 6. SVM F-measure (%) for "Major" severity class

Training Projects	Testing Projects						
	CD	Deb	TUI	SWT	UI	IDE	TUI2
CD	–	90.64	58.29	54.95	53.38	80.07	62.03
Deb	95.90	–	54.54	36.43	35.09	65.71	96.27
TUI	66.97	65.84	–	96.08	95.45	96.45	52.50
SWT	58.62	50.64	89.71	–	90.51	89.73	33.07
UI	43.86	23.26	97.39	98.50	–	97.21	22.94
IDE	87.60	84.34	91.79	94.35	90.73	–	57.30
TUI2	96.69	98.84	59.52	39.45	39.57	69.74	–

Table 7. NB F-measure (%) for "Major" severity class

Training Projects	Testing Projects						
	CD	Deb	TUI	SWT	UI	IDE	TUI2
CD	–	95.57	73.18	51.49	62.23	77.61	69.70
Deb	94.87	–	87.38	69.87	77.53	88.11	92.51
TUI	56.08	80.57	–	89.08	92.70	92.65	66.66
SWT	39.64	51.09	76.01	–	83.61	73.57	43.27
UI	41.32	67.15	95.43	92.77	–	89.34	50.42
IDE	71.43	89.04	96.27	85.90	91.63	–	77.39
TUI2	88.51	96.08	84.29	63.65	78.43	89.65	–

Table 8. k-NN accuracy (%) for different testing candidates

Training projects	Testing projects						
	CD	Deb	TUI	SWT	UI	IDE	TUI2
CD	–	68.24	57.35	39.37	51.16	62.04	63.52
Deb	80.69	–	50.27	31.11	37.92	54.85	80.89
TUI	76.82	76.58	–	72.70	78.97	77.42	64.52
SWT	46.78	49.32	73.26	–	76.95	74.58	27.05
UI	53.65	51.58	82.62	80.32	–	79.10	33.75
IDE	78.97	78.15	78.61	73.65	76.44	–	61.04
TUI2	72.53	82.43	54.14	24.44	45.20	50.84	–

Table 9. SVM accuracy (%) for different testing candidates

Training projects	Testing projects						
	CD	Deb	TUI	SWT	UI	IDE	TUI2
CD	–	67.34	58.56	42.54	53.49	65.22	55.33
Deb	81.55	–	58.69	34.44	47.02	56.86	83.37
TUI	57.94	56.98	–	78.57	81.70	78.76	50.12
SWT	46.78	47.52	80.08	–	80.08	76.25	33.75
UI	39.91	25.45	83.16	81.59	–	79.93	21.59
IDE	69.96	73.20	80.61	79.05	78.67	–	55.83
TUI2	73.39	83.78	55.35	38.73	45.40	61.20	–

Table 10. NB accuracy (%) for different testing candidates

Training projects	Testing projects						
	CD	Deb	TUI	SWT	UI	IDE	TUI2
CD	–	70.43	66.22	41.49	58.70	62.41	65.25
Deb	80.26	–	76.34	57.94	68.86	71.24	81.89
TUI	45.49	69.82	–	73.81	80.79	75.92	66.00
SWT	31.33	47.75	67.78	–	74.52	61.37	40.69
UI	33.05	57.66	82.22	77.46	–	76.42	49.63
IDE	55.79	77.48	81.15	72.06	79.17	–	72.95
TUI2	66.09	82.43	69.12	53.33	64.41	75.59	–

We have designed 7 cases for 7 training projects given below.

Case 1: F-measure of Major Severity Level and Accuracy improvement over Singh et al. (2017) for training project CD

The proposed approach improved the F-measure performance by 29.73%, 1.98%, 15.56% and 25.16% for testing projects "Deb", "TUI", "IDE" and "TUI2" respectively for KNN classifier. For SVM the F-measure performance improved by 20.70%, 2.70%, 12.26% and 62.03% for testing projects "Deb", "TUI", "IDE" and "TUI2" respectively. For testing projects "Deb", "TUI", "SWT", "UI", "IDE" and "TUI2", the F-measure performance improve by 62.24%, 64.29%, 35.16%, 52.01%, 64.47% and 25.16% respectively for NB classifier.

The entropy based proposed approach improved the Accuracy performance by 20.94%, 20.45%, 11.12%, 17.56% and 33% for testing projects "Deb", "TUI", "UI", "IDE" and "TUI2" respectively for KNN classifier. For SVM the Accuracy performance improved by 19.37%, 21.13%, 13.05%, 20.57% and 26.3% for testing projects "Deb", "TUI", "UI", "IDE" and "TUI2" respectively. For testing projects "Deb", "TUI", "SWT", "UI", "IDE" and "TUI2", the F-measure performance improved by 46.78%, 50.04%, 25.93%, 39.89%, 45.35% and 46.64% respectively for NB classifier.

Case 2: F-measure of Major Severity Level and Accuracy improvement over Singh et al. (2017) for training project Deb

In case of KNN and SVM classifiers, F-measure performance improved by 34.27%, 3.60%, 44.21% and 30.97%, 1.44%, 93.59% for testing projects "CD", "IDE" and "TUI2" respectively. Our approach improved the F-measure performance by 60.62%, 81.49%, 60.63%, 68.18%, 83% and 82.20% for testing projects "CD", "TUI", "SWT", "UI", "IDE" and "TUI2" respectively for NB classifier.

The proposed approach improved the Accuracy performance by 37.34%, 13.24%, 10.54% and 53.35% for testing projects "CD", "TUI", "IDE" and "TUI2" respectively for KNN classifier. For SVM the Accuracy performance improved by 28.33%, 21.93%, 6.58%, 12.21% and 55.08% for testing projects "CD", "TUI", "UI", "IDE" and "TUI2" respectively. For testing projects "CD", "TUI", "SWT", "UI", "IDE" and "TUI2", the Accuracy performance improved by 59.23%, 65.78%, 42.38%, 56.32%, 60.37% and 70.23% respectively for NB classifier.

Case 3: F-measure of Major Severity Level and Accuracy improvement over Singh et al. (2017) for training project TUI

In case of KNN and SVM classifiers, the proposed approach improved the F-measure performance by 25.64%, 24.93%, 29.41%, 30.64%, 93.72%, 28.29% and 12.39%, 12.01%, 41.82%, 42.43%, 42.83%, 49.81% for testing projects "CD", "Deb", "SWT", "UI", "IDE" and "TUI2" respectively. For testing projects "CD", "Deb", "SWT", "UI", "IDE" and "TUI2", the F-measure performance improved by 35.03%, 67.71%, 80.08%, 86.44%, 83.50% and 51.65% respectively for NB classifier.

The entropy based proposed approach improved the Accuracy performance by 28.32%, 30.41%, 31.59%, 26.49%, 37.62% and 4.22% for testing projects "CD", "Deb", "SWT", "UI", "IDE" and "TUI2" respectively for KNN classifier. For SVM the Accuracy performance improved by 21.03%, 17.57%, 37.62%, 39.74%, 37.62 and 13.15% for testing projects "CD", "Deb", "SWT", "UI", "IDE" and "TUI2" respectively. For testing projects "CD", "Deb", "SWT", "UI", "IDE" and "TUI2", the Accuracy performance improved by 33.04%, 57.43%, 59.68%, 67.95%, 65.22% and 47.14% respectively for NB classifier.

Case 4: F-measure of Major Severity Level and Accuracy improvement over Singh et al. (2017) for training project SWT

We observed that the F-measure performance of our approach has improved by 34.05%, 36.03% and 70.76% for testing projects "TUI", "UI" and "IDE" respectively in case of KNN classifier. In case of SVM, the F-measure performance improved by 28.99%, 28.33%, 25.33% and 32.41% for testing projects "TUI", "UI", "IDE" and "TUI2" respectively. For testing projects "CD", "Deb", "TUI", "UI", "IDE" and "TUI2", the F-measure performance improved by 18.39%, 38.41%, 69.26%, 76.40%, 61.45% and 27.67% respectively for NB classifier.

In case of KNN classifier, our approach improved the Accuracy performance by 1.29%, 7.2%, 38.23%, 39.94%, 34.11% and 0.5% for testing projects "CD", "Deb", "TUI", "UI", "IDE" and "TUI2 respectively. In case of Accuracy values of SVM classifier, our approach improved by 43.05%, 40.04%, 31.06% and 4.72% for testing projects "TUI", "UI", "IDE" and "TUI2" respectively. In case of NB classifier, for testing projects "CD", "Deb", "TUI", "UI", "IDE" and "TUI2", the Accuracy performance improved by 16.74%, 38.29%, 55.35%, 61.68%, 47.49% and 28.53% respectively.

Case 5: F-measure of Major Severity Level and Accuracy improvement over Singh et al. (2017) for training project UI

The proposed approach improved the F-measure performance by 34.25%, 35.41% and 39.41% for testing projects "TUI", "SWT" and "IDE" respectively for KNN classifier. For SVM the F-measure performance improved by 43.52%, 41.21 and 39.82% testing projects "TUI", "SWT" and "IDE" respectively. For testing projects "CD", "Deb", "TUI", "SWT", "IDE" and "TUI2", the F-measure performance improved by 19.18%, 49.22%, 89.42%, 79.73%, 79.94% and 30.30% respectively for NB classifier.

The entropy based proposed approach improved the Accuracy performance by 2.15%, 5.18%, 29.28%, 36.99%, and 35.96% for testing projects "CD", "Deb", "TUI", "SWT" and "IDE" respectively for KNN classifier. For SVM the Accuracy performance improved by 46.26%, 37.15% and 36.45% for testing projects "TUI", "SWT"

and "IDE" respectively. For testing projects "CD", "Deb", "TUI", "SWT", "IDE" and "TUI2", the F-measure performance improved by 20.6%, 43.7%, 68.05%, 66.03%, 64.38% and 35.49% respectively for NB classifier.

Case 6: F-measure of Major Severity Level and Accuracy improvement over Singh et al. (2017) for training project IDE

In case of KNN, F-measure performance improved by 33.85%, 25.89%, 34.95% and 25.98%, 30.19% and 93.59% for testing projects "CD", "Deb", "TUI", "SWT", "UI" and "TUI2" respectively. For SVM, F-measure performance improved by 25.80%, 22.64%, 33.81%, 32.49%, 29.54% and 56.57% for testing projects "CD", "Deb", "TUI", "SWT", "UI" and "TUI2" respectively. The F-measure performance improved by 52.06%, 72.88%, 83.97%, 73.12%, 86.38% and 62.14% for testing projects "CD", "Deb", "TUI", "SWT", "UI" and "TUI2" respectively for NB classifier.

In case of KNN, the Accuracy performance improved by 33.05%, 31.53%, 41.71%, 29.68%, 35.69% and 30.77% for testing projects "CD", "Deb", "TUI", "SWT", "UI" and "TUI2" respectively. For SVM the Accuracy performance improved by 18.89%, 26.58%, 44.38%, 31.75%, 38.53% and 27.05% for testing projects "CD", "Deb", "TUI", "SWT", "UI" and "TUI2" respectively. For testing projects "CD", "Deb", "TUI", "SWT", "UI" and "TUI2", the Accuracy performance improved by 41.2%, 66.44%, 68.05%, 56.98%, 67.44% and 59.05% respectively for NB classifier.

Case 7: F-measure of Major Severity Level and Accuracy improvement over Singh et al. (2017) for training project TUI2

In case of F-measure performance of KNN classifier, our approach improved by 23.90% and 36.35% for testing projects "CD" and "Deb" respectively. In case of SVM, the F-measure performance improved by 51.90%, 54.56%, 22.80% and 66.83% for testing projects "CD", "Deb", "TUI" and "IDE" respectively. For testing projects "CD", "Deb", "TUI", "SWT", "UI" and "IDE", the F-measure performance improved by 70.98%, 82.83%, 71.43%, 47.40%, 71.29% and 78.46% respectively for NB.

The entropy based proposed approach improved the Accuracy performance by 21.89%, 39.86% and 10.87% for testing projects "CD", "Deb" and "IDE" respectively for KNN classifier. For SVM the Accuracy performance improved by 51.07%, 67.56%, 18.18%, 28.73%, 11.83 and 36.45% for testing projects "CD", "Deb", "TUI", "SWT", "UI" and "IDE" respectively. For testing projects "CD", "Deb", "TUI", "SWT", "UI" and "IDE", the F-measure performance improved by 55.36%, 62.16%, 46.53%, 37.46%, 40.04% and 57.53% respectively for NB.

Out of 42 cases, i.e. 7 training datasets * 6 testing datasets, the classifiers "k-NN", "SVM" and "NB" perform better in 27, 30 and 42 cases respectively in terms of F-measure performance for Major severity class in comparison with Singh et al. [6]. For Accuracy comparison the classifiers k-NN, SVM and NB perform better in 35, 35 and 42 cases respectively.

Figures 3, 4 and 5 show the F-measure performance comparison of "k-NN", "SVM" and "NB" techniques for proposed summary entropy based cross project severity prediction with Singh et al. [6].

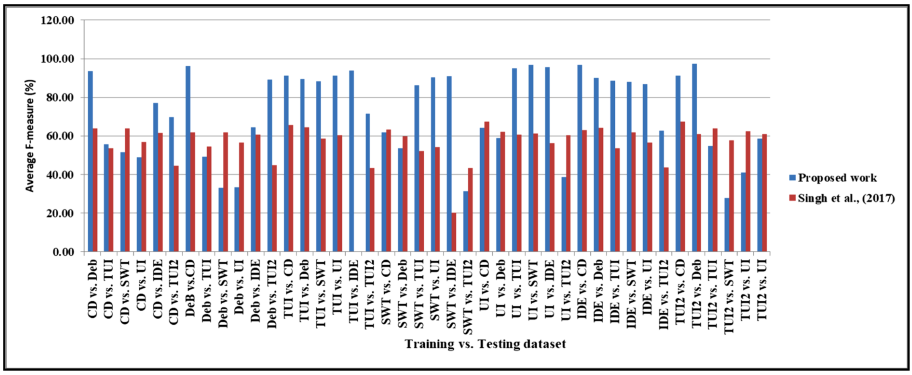

Fig. 3. k-NN F-measure comparison for "Major" severity level

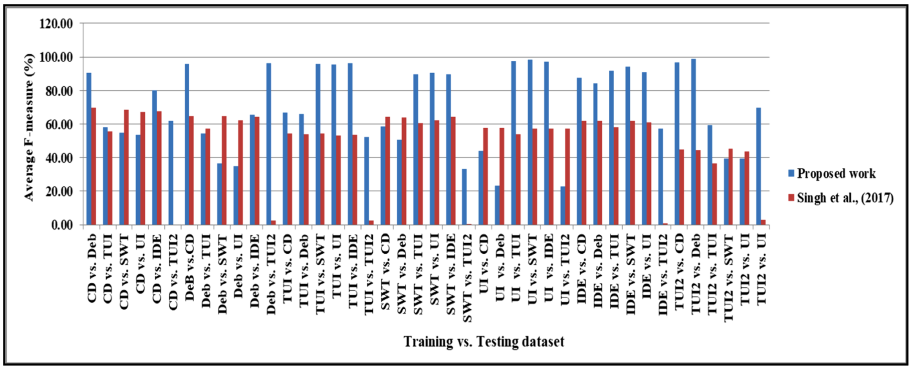

Fig. 4. SVM F-measure comparison for "Major" severity level

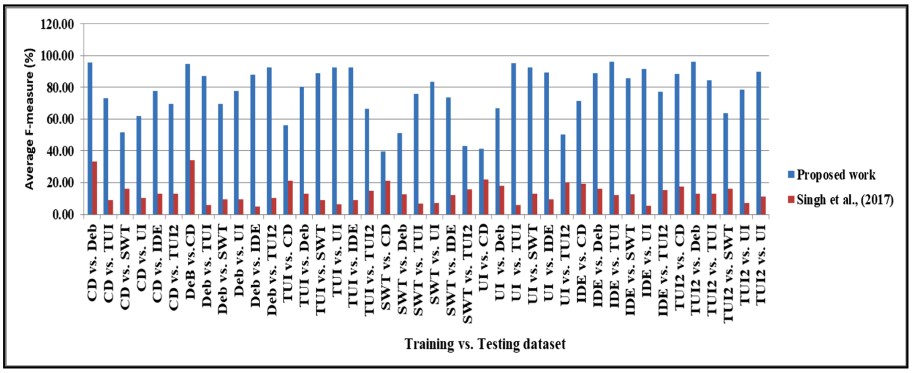

Fig. 5. NB F-measure comparison for "Major" severity level

The Accuracy comparison of the proposed entropy approach with Singh et al. [6] using k-NN, SVM and NB techniques for cross project severity prediction has been shown in Fig. 6, 7 and 8.

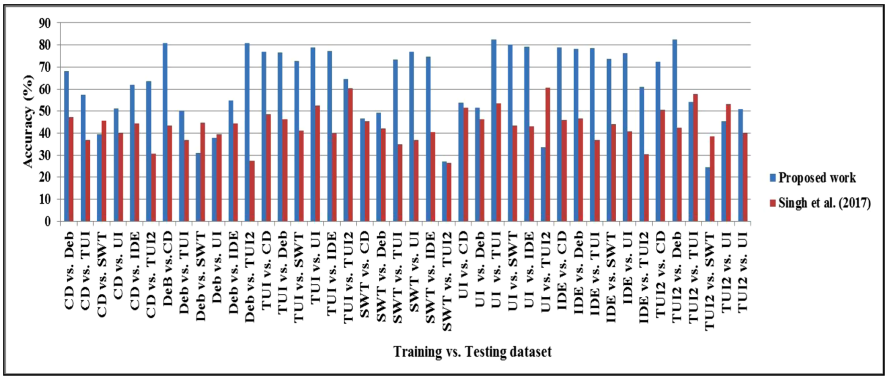

Fig. 6. k-NN accuracy comparison (proposed work vs. Singh et al. (2017))

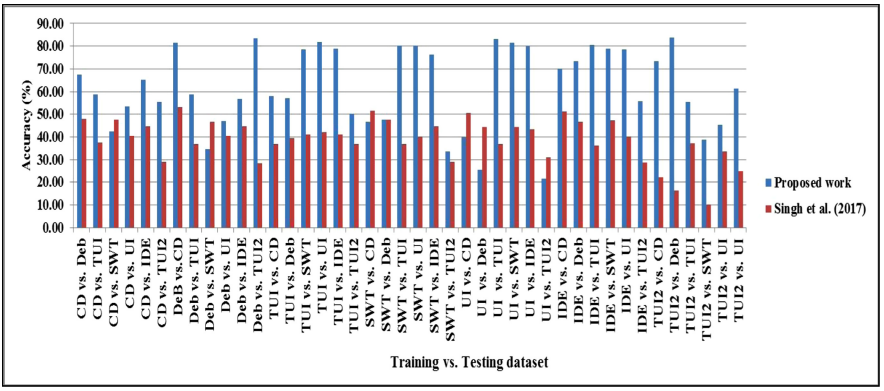

Fig. 7. SVM accuracy comparison (proposed work vs. Singh et al. (2017))

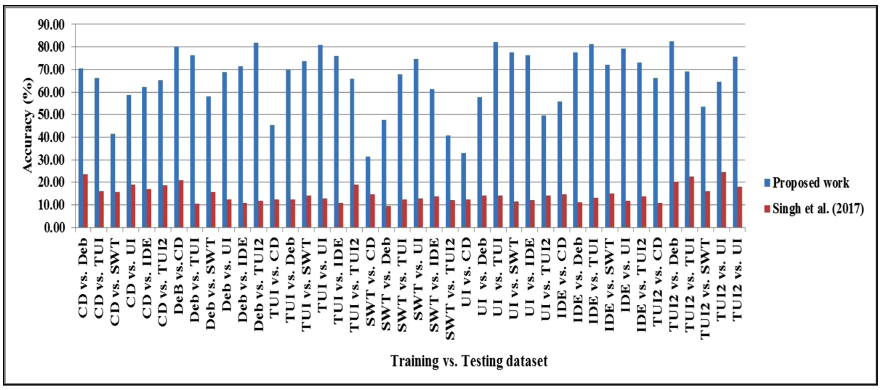

Fig. 8. NB accuracy comparison (proposed work vs. Singh et al. (2017))

6 Conclusion

In this paper, we have proposed an approach using bug priority, summary entropy and summary weight for cross project bug severity prediction. For taking care of uncertainty in bug summary attribute, we have derived an attribute termed as summary entropy using Shannon entropy. Summary weight is also derived by taking the sum of weights of summary terms using information gain criteria. We have used machine learning techniques, namely "k-Nearest Neighbors", "Support Vector Machine" and "Naïve Bayes" to build the classifiers. The empirical evaluation has been validated on seven products of Eclipse project. The built-in classifiers based on these techniques predicted the severity of bug reports in cross project context with significant Accuracy and F-measure. We have also optimized the parameters by using Grid Search. Our proposed approach outperform with the work available in the literature [6]. The proposed approach improved the F-measure for "k-NN", "SVM", "NB" by 1.98% to 93.72%, 1.44% to 93.59% and 18.39% to 89.42% respectively across all the 42 cases for cross project bug severity prediction in comparison with [6]. Our entropy based proposed approach improved the Accuracy from 0.5% to 53.35% for k-NN, 4.72% to 67.56% for SVM and 16.74% to 70.23% for NB across all the 42 cases. NB outperforms for bug severity prediction across all the 42 cases in terms of both F-measure and Accuracy performance. More analysis in the field of summary entropy based metric models may be performed in the future with other projects data. We can measure various forms of entropy and test the built in classifier with more techniques and data sets.

References

1. Singh, V.B., Chaturvedi, K.K.: Bug tracking and reliability assessment system. Int. J. Softw. Eng. Appl. **5**(4), 17–30 (2011)
2. Sharma, M., Bedi, P., Chaturvedi, K.K., Singh, V.B.: Predicting the priority of a reported bug using machine learning techniques and cross project validation. In: International Conference Intelligent Systems Design and Applications (ISDA), pp. 27–29. IEEE (2012)
3. Kumari, M., Sharma, M., Singh, V.B.: Severity assessment of a reported bug by considering its uncertainty and irregular state. Int. J. Open Source Softw. Process. (IJOSSP) **9**(4), 20–46 (2018)
4. Sharma, M., Bedi, P., Singh, V.B.: An empirical evaluation of cross project priority prediction. Int. J. Syst. Assurance Eng. Manage. **5**(4), 651–663 (2014). https://doi.org/10.1007/s13198-014-0219-4
5. Sharma, M., Kumari, M., Singh, R.K., Singh, V.B.: Multiattribute based machine learning models for severity prediction in cross project context. In: Murgante, B., et al. (eds.) ICCSA 2014. LNCS, vol. 8583, pp. 227–241. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09156-3_17
6. Singh, V.B., Misra, S., Sharma, M.: Bug severity assessment in cross project context and identifying training candidates. J. Inform. Knowl. Manage **16**(1), 1750005 (2017)
7. Menzies, T., Marcus, A.: Automated severity assessment of software defect reports, pp. 346–355. Int. Conf. Softw. Maintenance, IEEE (2008)

8. Lamkanfi, A., Demeyer, S., Giger, E., Goethals, B.: Predicting the severity of a reported bug. In: Mining Software Repositories (MSR), pp. 1–10 (2010)
9. Lamkanfi, A., Demeyer, S., Soetens, Q.D., Verdonck, T.: Comparing mining algorithms for predicting the severity of a reported bug. In: CSMR, pp. 249–258 (2011)
10. Chaturvedi, K.K., Singh, V.B.: Determining bug severity using machine learning techniques. In: CSI-IEEE International Conference on Software Engineering (CONSEG), pp. 378–387 (2012)
11. Tian, Y., Lo, D., Sun, C.: Information retrieval based nearest neighbor classification for fine-grained bug severity prediction. In WCRE, Kingston, ON, Canada, pp. 215–224. IEEE, New York (2012)
12. Chaturvedi, K.K., Singh, V.B.: An empirical comparison of machine learning techniques in predicting the bug severity of open and close source projects. Int. J. Open Source Softw. Process. **4**(2), 32–59 (2013)
13. Kumari, M., Singh, V.B.: An Improved classifier based on entropy and deep learning for bug priority prediction. In: Abraham, A., Cherukuri, A.K., Melin, P., Gandhi, N. (eds.) ISDA 2018 2018. AISC, vol. 940, pp. 571–580. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-16657-1_53
14. Singh, V.B., Sharma, M., Pham, H.: Entropy based software reliability analysis of multi-version open source software. IEEE Trans. Softw. Eng. **44**(12), 1207–1223 (2017)
15. Kumari, M., Misra, A., Misra, S., Fernandez Sanz, L., Damasevicius, R., Singh, V.B.: Quantitative quality evaluation of software products by considering summary and comments entropy of a reported bug. Entropy **21**(1), 91 (2019)
16. IEEE88: IEEE Standard Dictionary of Measures to Produce Reliable Software. IEEE Std 982.1-1988, Institute of Electrical and Electronics Engineers (1989)
17. Shannon, C.E.: A mathematical theory of communication. Bell System Tech. J., **27**, 379–423, 623–656 (1948)
18. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: 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) (2006). http://www.rapid-i.com
19. Chaturvedi, K.K., Singh, V.B.: An empirical comparison of machine learning techniques in predicting the bug severity of open and closed source projects. Int. J. Open Source Softw. Process. (IJOSSP) **4**(2), 32–59 (2012)
20. Yang, C.Z., Hou, C.C., Kao, W.C., Chen, X.: An empirical study on improving severity prediction of bug reports using feature selection. In: Software Engineering Conference (APSEC), 19th Asia-Pacific, vol. 1, pp. 240–249. IEEE (2012)
21. Iliev, M., Karasneh, B., Chaudron, M. R., Essenius, E.: Automated prediction of bug severity based on codifying design knowledge using ontologies. In: Proceedings of the First International Workshop on Realizing AI Synergies in Software Engineering, pp. 7–11. IEEE Press (2012)