Bug Priority Assessment in Cross-Project Context Using Entropy-Based Measure

Meera Sharma, Madhu Kumari, and V. B. Singh

1 Introduction

A large number of bugs are reported on bug tracking systems by different users, developers, and staff members located at different geographical locations. Bug priority (P1, the most important, to P5, the least important) is an important attribute which determines the importance and the order of fixing of the bugs in the presence of other bugs. To automate the bug priority prediction, we need historical data to train the classifiers. In reality, this data is not available easily in all software projects, especially in new projects. Cross-project priority prediction works well in such situation where we train the classifiers with historical data of projects other than the testing projects [1, 2].

The bug reports are reported by users having different levels of knowledge about the software which results in uncertainty and noise in bug reports data. "Without proper handling of these uncertainties and noise, the performance of learning strategies can be significantly reduced" [22]. The entropy-based measure has been used to calculate the uncertainty in bug summary reported by different users. In literature, researchers [1, 2] have made attempts for cross-project bug summary-based priority prediction. No attempt has been made to handle uncertainty in bug summary in cross-project context for bug priority prediction. We have proposed summary entropy-based cross-project priority prediction models using Support Vector Machine (SVM), k-Nearest Neighbors (k-NN), Naïve Bayes (NB), and Neural Network (NNET). In addition to the summary entropy, we have also considered bug severity and the derived bug summary weight attribute. Results show improvement in performance over summary-based cross-project priority prediction models [2].

M. Sharma
Swami Shraddhanand College, University of Delhi, Delhi, India

M. Kumari · V. B. Singh (✉)
Delhi College of Arts and Commerce, University of Delhi, Delhi, India
e-mail: vbsingh@dcac.du.ac.in

S. Patnaik et al. (eds.), *Advances in Machine Learning and Computational Intelligence*,
Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-15-5243-4_10

The rest of the paper is organized as follows: Sect. 2 deals with related work. Section 3 describes the data description, bug attributes, and model building required to perform the analysis. Results have been discussed in Sect. 4. Finally, the paper is concluded in Sect. 5.

2 Related Work

Bug priority assessment helps in correct resource allocation and bug fix scheduling. A bug priority recommender has been proposed by Kanwal and Maqbool [3] by using SVM classification technique. The study was further extended for comparison of SVM and NB performance with different feature sets by Kanwal and Maqbool [4]. An attempt for bug priority prediction has been made by Alenezi and Banitaan [5] using NB, Decision Tree (DT), and Random Forest (RF) for Firefox and Eclipse datasets. Lian Yu et al. [6] proposed defect priority prediction using Artificial Neural Network (ANN) technique. Results show that ANN performs better than Bayes algorithm. Tian et al. [7] proposed a new framework called DRONE (PreDicting PRiority via Multi-Faceted FactOr ANalysEs) for Eclipse projects and compared it with Severis^{Prio} and Severis^{Prio+} [8].

In literature, several studies have been conducted in cross-project context [9–16].

Bug summary-based cross-project priority prediction models have been proposed by [1, 2] using SVM, NB, k-NN, and NNET. Results show that cross-project bug priority prediction works well. Another attempt has been made by authors to propose bug summary-based cross-project severity prediction models [17].

Software are evolved through source code changes done in it to fix different issues, namely bugs, new features, and feature improvements reported by different users. These source code changes result in uncertainty and randomness in the system. In literature, researchers have used entropy-based measures to quantify the code change process for defects prediction [18]. Researchers have used entropy-based measures to predict the potential code change complexity [19]. A software reliability uncertainty analysis method has been proposed by Mierswa et al. [20].

To our knowledge, no work has been done for measuring trustworthiness of bug summary data in bug repositories. The uncertainty/noise present in bug summary data can affect the performance of prediction models. In this paper, we have measured the uncertainty in bug summary by using entropy-based measures. In addition to summary entropy, bug severity and summary weight for bug priority prediction in cross-project context have been considered. We have compared our proposed summary entropy-based bug priority prediction models with Sharma et al. [2] and found improvement in performance of the classifiers.

Table 1 Priority-wise number of bug reports of different projects

Project	Product	Priority-wise number of bug reports					
		P1	P2	P3	P4	P5	Total
Eclipse	V2	923	1416	8609	370	229	11,547
Eclipse	V3	361	963	26,667	320	136	28,447
OpenOffice	DB	76	472	2834	243	38	3663
OpenOffice	SST	82	518	4210	316	114	5240
OpenOffice	PPT	62	553	2688	90	37	3430

3 Description of Datasets, Bug Attributes, and Model Building

In this section, description of datasets and bug attributes used for validation and the model building have been discussed.

3.1 Description of Datasets

We have taken different products, namely Platform Version 2 (V2), Platform Version 3 (V3) of Eclipse project (http://bugs.eclipse.org/bugs/) and Database Access (DB), Spreadsheet (SST), Presentation (PPT) of OpenOffice project (http://bz.apache.org/000/). We have considered the bug report for status "verified," "resolved," and "closed." Table 1 shows the distribution of bug reports of different priority levels.

3.2 Bug Attributes

To predict bug priority in cross-project context, we considered three attributes, namely severity, summary weight, and entropy of summary. Severity is a nominal attribute, whereas summary weight and entropy are continuous attributes. Bug severity gives the impact of bug on the functionality of software or its components. It is divided into seven levels, namely "Blocker, Critical, Major, Normal, Minor, Trivial, and Enhancement." Blocker is the highest level, and Enhancement is lowest level. 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. The bug summary gives the textual description of the bug. Summary weight is extracted from the bug summary attribute, entered by the users.

The bug summary has been preprocessed with the RapidMiner tool [21] to calculate the summary weight of a reported bug [2].

Different users are reported bug on bug tracking system. The size of software repositories is also increasing by an enormous rate that enhances the noise and uncertainty in the bug priority prediction. If these uncertainties are not handled properly, the performance of the learning strategy can be significantly reduced [22]. We have proposed entropy-based measure to build the classifier for bug priority prediction to handle uncertainties in cross-project context. We have used Shannon entropy to build the classifier model.

Shannon entropy, S is defined as

$$S = -p_i \log_2 p_i$$

where $p_i = \frac{\text{Total number of occurrences of terms in } i\text{th bug report}}{\text{Total number of terms}}$.

The top 200 terms have been taken from all terms based on their weight. To rationalize the effect of the priority, we multiplied the entropy by 10 for P1 and P2 priority level bugs, 3 for the P3 priority level bug, and 1 for P4 and P5 priority level bugs [23].

3.3 Model Building

We have proposed summary entropy-based classifiers based on SVM, k-NN, NNET, and NB for bug priority prediction in cross-project context by taking bug attributes severity and summary weight. We have taken the bug reports of two products of Eclipse and three products of OpenOffice projects. To get the significant amount of performance, we have used the appropriate parameters values. "For SVM, we have taken polynomial kernel with degree 3, the value of k as 5 in case of k-NN and for NNET the training cycle as 100" [2]. Number of validations is taken as 10 and sampling types as stratified sampling for different classification techniques. The performance of the proposed models has been validated using different performance measures, namely Accuracy, Precision, Recall, and F-measure.

Figure 1 shows the main process of cross-project priority prediction.

4 Results and Discussion

We have validated the entropy-based classifier of different machine learning techniques, namely SVM, k-Nearest Neighbors, Naive Bayes, and Neural Network using 10 fold cross-validations for predicting the bug priority. We have compared the proposed entropy-based approach to Sharma et al. [2]. We have taken the same datasets and techniques as taken by Sharma et al. [2] to predict the bug priority in cross-project

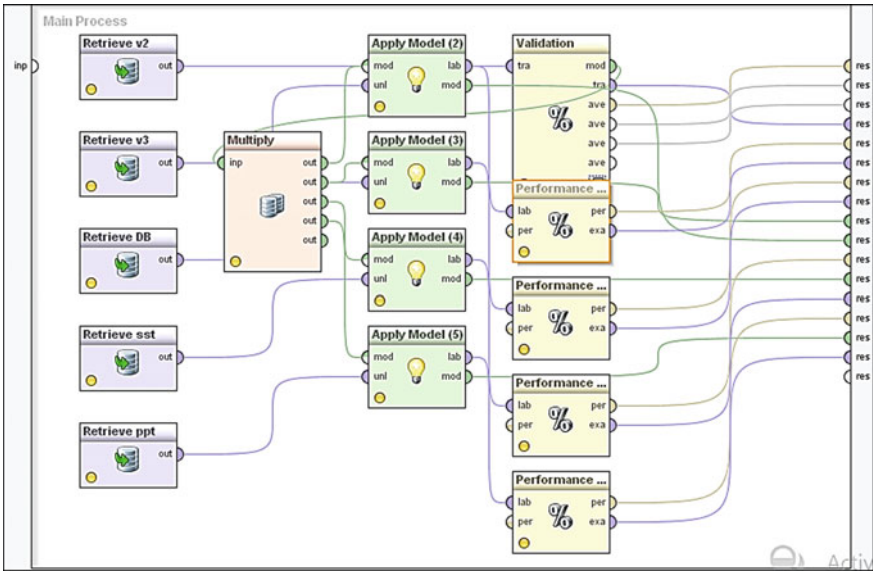

Fig. 1 RapidMiner process for bug priority prediction in cross-project context

context. Table 2 shows the Accuracy of different machine learning techniques to predict the priority of cross-validated projects.

Accuracy for Training Dataset V2

For testing dataset V3, our entropy-based approach improves the Accuracy by 3.46% and 91.93% for SVM and NB, respectively. Our entropy-based approach improves the Accuracy by 7.86%, 10.21%, 2.81%, and 82.85% for SVM, k-NN, NNET, and NB, respectively, for testing dataset DB. For testing dataset SST, our approach improves the Accuracy by 6.66%, 8.42%, 2.96%, and 82.08% for SVM, k-NN, NNET, and NB, respectively. Our entropy-based approach improves the Accuracy by 11.69%, 10.99%, 13.00%, and 85.19% for SVM, k-NN, NNET, and NB, respectively, for testing dataset PPT.

Accuracy for Training Dataset V3

Our entropy-based approach improves the Accuracy by 6.34%, 6.57%, 6.40%, and 82.10% for SVM, k-NN, NNET, and NB, respectively, for testing dataset DB. For testing dataset SST, our approach improves the Accuracy by 3.46% and 91.93% for SVM and NB, respectively. Our entropy-based approach improves the Accuracy by 9.39%, 8.16%, 7.41%, and 76.44% for SVM, k-NN, NNET, and NB, respectively, for testing dataset PPT.

Accuracy for Training Dataset DB

For testing dataset V2, our entropy-based approach improves the Accuracy by

Table 2 Accuracy (%) of cross-validated projects

Training versus testing dataset	Accuracy (%)			
	SVM	k-NN	NNET	NB
V2 versus V3	95.51	89.26	91.13	95.59
V2 versus DB	84.93	86.24	80.04	86.40
V2 versus SST	86.43	87.69	83.19	87.02
V2 versus PPT	89.53	88.13	91.02	89.48
V3 versus DB	86.13	86.27	86.21	86.32
V3 versus SST	86.66	86.89	86.74	86.95
V3 versus PPT	87.67	86.50	87.64	81.95
DB versus V2	77.73	83.29	83.14	68.21
DB versus V3	94.48	91.05	96.07	85.55
DB versus SST	85.10	92.18	96.53	86.34
DB versus PPT	86.73	83.97	86.82	69.30
SST versus V2	77.61	74.00	50.40	58.68
SST versus V3	94.47	88.15	81.09	80.72
SST versus DB	84.93	88.15	81.33	82.25
SST versus PPT	86.85	83.76	82.30	61.25
PPT versus V2	78.58	79.13	83.02	79.02
PPT versus V3	94.87	93.34	93.94	92.06
PPT versus DB	86.24	82.72	73.87	86.35
PPT versus SST	86.98	77.96	75.82	86.97

3.43%, 10.29%, 9.09%, and 60.21% for SVM, k-NN, NNET, and NB, respectively. Our entropy-based approach improves the Accuracy by 1.04%, 0.11%, 2.70%, and 79.27% for SVM, k-NN, NNET, and NB, respectively, for testing dataset V3. For testing dataset SST, our approach improves the Accuracy by 5.46%, 13.35%, 16.66%, and 76.19% for SVM, k-NN, NNET, and NB, respectively. Our entropy-based approach improves the Accuracy by 8.77%, 5.66%, 11.59%, and 60.12% for SVM, k-NN, NNET, and NB, respectively, for testing dataset PPT.

Accuracy for Training Dataset SST

For testing dataset V2, our entropy-based approach improves the Accuracy by 3.14%, 1.08%, and 49.11% for SVM, k-NN, and NB, respectively. Our entropy-based approach improves the Accuracy by 0.92% and 75.93% for SVM and NB, respectively, for testing dataset V3. For testing dataset DB, our approach improves the Accuracy by 7.89%, 11.11%, 4.18%, and 75.83% for SVM, k-NN, NNET, and NB, respectively. Our entropy-based approach improves the Accuracy by 8.89%, 5.25%, 4.02%, and 52.07% for SVM, k-NN, NNET, and NB, respectively, for testing dataset PPT.

Accuracy for Training Dataset PPT

For testing dataset V2, our entropy-based approach improves the Accuracy by 4.33%, 5.85%, 8.59%, and 70.95% for SVM, k-NN, NNET, and NB, respectively. Our entropy-based approach improves the Accuracy by 1.42%, 1.64%, 0.38%, and 84.93% for SVM, k-NN, NNET, and NB, respectively, for testing dataset V3. For testing dataset DB, our approach improves the Accuracy by 9.23%, 5.62%, and 72.92% for SVM, k-NN, and NB, respectively. Our entropy-based approach improves the Accuracy by 7.46% and 75.73% for SVM and NB, respectively, for testing dataset SST.

Out of 19 combination cases, SVM, k-NN, NNET, and NB outperform in 19, 16, 14, and 19 cases, respectively, in comparison with Sharma et al. [2]. Our approach improves the Accuracy 0.92–11.69% for SVM, 0.11–13.35% for k-NN, 0.38–16.66% for NNET, and 49.11–91.93% for NB across all the 19 combinations for bug priority prediction in cross-project context. SVM and NB outperforms for bug priority prediction across all the 19 combinations.

Table 3 shows the best training dataset with highest Accuracy for different machine learning techniques. Across all the machine learning techniques, on the basis of Accuracy, DB is the best training dataset for V2 testing dataset, DB is the best training dataset for V3 testing dataset, SST is best training dataset for DB testing dataset, DB is the best training dataset for SST testing dataset, and V2 is the best training dataset for PPT testing dataset.

Avg. F-Measure for Training Dataset V2

From Table 4, we observed that the value of F-measure (avg.) lies between 34.32%–48.49%, 30.69%–40.52%, 31.63%–40.04%, and 35.13%–39.44% for training candidates V3, DB, SST, and PPT, respectively, across all the machine learning techniques.

Avg. F-Measure for Training Dataset V3

We obtained the value of F-measure (avg.) that lies between 33.94%–35.22%,

Table 3 Classifier-wise best training candidate with highest accuracy

Best training dataset (Accuracy %)				
Testing datasets	SVM	k-NN	NNET	NB
V2	PPT (78.58)	DB (83.29)	DB (83.14)	PPT (79.02)
V3	V2 (95.51)	DB (91.05)	DB (96.07)	V2 (95.59)
DB	PPT (86.24)	SST (88.15)	V3 (86.21)	V2 (86.40)
SST	PPT (86.98)	DB (92.18)	DB (96.53)	PPT (86.97)
PPT	V2 (89.53)	V2 (88.13)	V2 (91.02)	V2 (89.48)

Table 4 Average precision (P), recall (R), and F-measure (F) for training dataset (V2 product)

Testing datasets	SVM			k-NN			NNET			NB		
	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
V3	40.75	33.47	36.19	52.29	53.91	48.49	31.53	41.44	34.32	76.83	40.15	42.58
DB	32.28	34.46	33.30	45.61	37.99	40.52	28.05	34.71	30.69	48.87	35.78	35.83
SST	33.94	33.53	33.52	48.91	36.81	40.04	28.78	35.84	31.63	54.93	34.10	34.76
PPT	39.90	34.14	35.13	65.55	35.33	38.56	36.27	35.72	35.88	45.58	37.45	39.44

33.53%–35.98%, and 32.04%–35.96% for training candidates DB, SST, and PPT, respectively, across all the machine learning techniques as given in Table 5.

Avg. F-Measure for Training Dataset DB
Table 6 shows the value of F-measure (avg.) that lies between of 25.23%–41.53%, 26.33%–38.09%, 30.93%–51.12, and 32.30%–42.75% for training candidates V2, V3, SST, and PPT, respectively, across all the machine learning techniques for bug priority prediction.

Avg. F-Measure for Training Dataset SST
From Table 7, we observed that the value of F-measure (avg.) lies between 25.00%–39.50%, 26.27%–38.80%, 32.46%–51.34%, and 31.84%–46.00% for training candidates V2, V3, DB, and PPT, respectively, across all the machine learning techniques.

Avg. F-Measure for Training Dataset PPT
Table 8 shows the value of F-measure (avg.) that lies between 26.71%–40.29%, 29.10%–39.88%, 27.88%–40.53%, and 27.64%–37.54% for training candidates V2, V3, DB, and SST, respectively, across all the machine learning techniques.

Table 9 shows the best training dataset with highest F-measure (avg.) for different machine learning techniques. Across all the machine learning techniques, on the basis of F-measure, DB is the best training candidate for V2 testing dataset, V2 is the best training candidate for V3 testing dataset, SST is best training candidate for DB testing dataset, DB is the best training candidate for SST testing dataset, and SST is the best training candidate for PPT testing dataset.

Figure 2 shows the Accuracy comparison using SVM machine learning technique for cross-project bug priority prediction.

Figure 3 shows the Accuracy comparison using k-NN machine learning technique for cross-project priority prediction.

Figure 4 shows the Accuracy comparison using NNET machine learning technique for cross-project priority prediction.

Figure 5 shows the Accuracy comparison using NB machine learning technique for cross-project priority prediction.

5 Conclusion

In the absence of data for building a classifier, cross-project study provides a solution. In this paper, we have proposed an approach for cross-project bug priority prediction using three attributes, bug severity, summary weight, and summary entropy. By considering learning from the uncertainty, we have derived an attribute termed as summary entropy using Shannon entropy. To build the classifier, we have used machine learning techniques, namely Support Vector Machine (SVM), k-Nearest Neighbors (k-NN), Naïve Bayes (NB), and Neural Network (NNET). The built-in classifiers

Table 5 Average precision (P), recall (R), and F-measure (F) for training dataset (V3 product)

Testing datasets	SVM			k-NN			NNET			NB		
	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
DB	34.96	33.67	33.94	52.33	34.55	35.22	34.92	33.87	34.06	33.82	34.99	34.31
SST	35.41	32.78	33.53	49.80	33.91	35.98	35.45	32.93	33.66	35.14	33.73	34.11
PPT	35.64	31.54	32.76	45.24	30.38	32.04	35.67	31.50	32.74	33.30	41.22	35.96

Table 6 Average precision (P), recall (R), and F-measure (F) for training dataset (DB product)

Testing datasets	SVM			k-NN			NNET			NB		
	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
V2	30.67	25.18	25.23	43.13	43.52	41.53	33.65	34.75	32.78	34.92	39.27	35.58
V3	35.26	24.36	26.33	40.30	43.85	38.09	38.57	34.30	34.36	29.05	42.27	31.58
SST	35.13	29.61	30.93	51.29	51.14	51.12	35.10	32.68	33.39	69.66	33.61	34.78
PPT	45.27	30.67	32.30	43.24	44.59	42.75	37.22	34.36	34.91	43.69	38.23	36.63

Table 7 Average precision (P), recall (R), and F-measure (F) for training dataset (SST product)

Testing datasets	SVM			k-NN			NNET			NB		
	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
V2	30.75	24.99	25.00	39.67	43.93	39.50	29.76	33.31	27.32	33.61	39.94	32.89
V3	35.17	24.32	26.27	41.63	42.40	38.80	34.33	35.96	32.87	31.54	42.31	31.46
DB	34.67	31.74	32.46	50.63	53.05	51.34	40.63	44.39	41.20	38.63	36.04	34.85
PPT	35.32	30.52	31.84	48.30	51.56	46.00	40.10	48.67	39.34	40.85	40.76	32.92

Table 8 Average precision (P), recall (R), and F-measure (F) for training dataset (PPT product)

Testing datasets	SVM			k-NN			NNET			NB		
	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
V2	30.45	26.57	26.71	49.55	32.43	35.41	29.40	34.77	31.84	41.53	39.68	40.29
V3	35.18	26.83	29.10	44.82	29.06	32.71	28.20	34.07	30.43	38.98	43.41	39.88
DB	34.13	34.65	34.24	47.71	41.70	40.53	25.48	33.72	27.88	33.76	35.04	34.30
SST	35.27	33.61	34.06	42.69	41.41	37.54	25.00	34.35	27.64	54.88	33.92	34.30

Table 9 Classifier-wise best training candidate with highest F-measure (average)

Best training dataset (average F-measure)				
Testing datasets	SVM	k-NN	NNET	NB
V2	PPT (26.71)	DB (41.53)	DB (32.78)	PPT (40.29)
V3	V2 (36.191)	V2(48.49)	DB (34.36)	V2 (42.58)
DB	PPT (34.24)	SST (51.34)	SST(41.20)	V2 (35.83)
SST	PPT (34.06)	DB (51.12)	V3 (33.36)	DB (34.78)
PPT	V2 (35.13)	SST (46.00)	SST (39.34)	V2 (39.44)

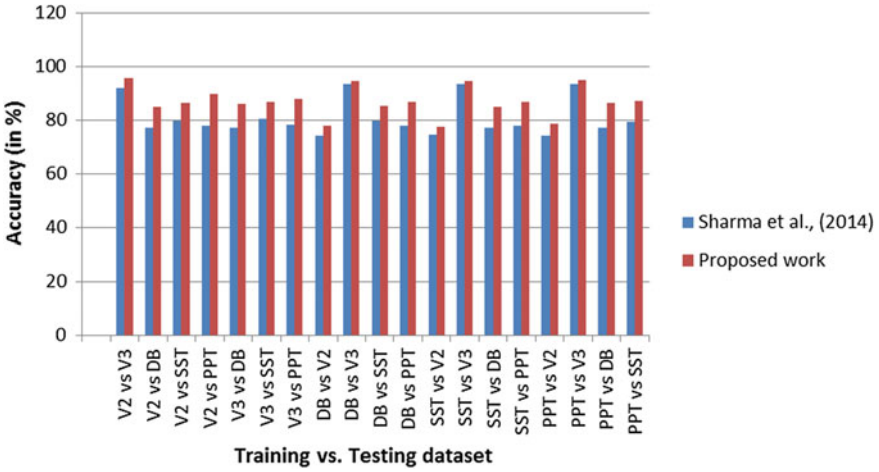

Fig. 2 SVM accuracy comparison (proposed work vs. Sharma et al., 2014 [2])

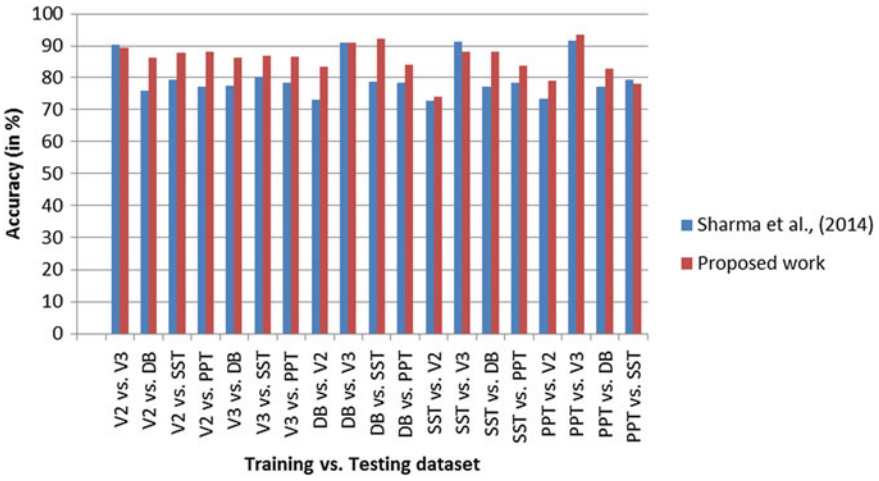

Fig. 3 k-NN accuracy comparison (proposed work vs. Sharma et al., 2014 [2])

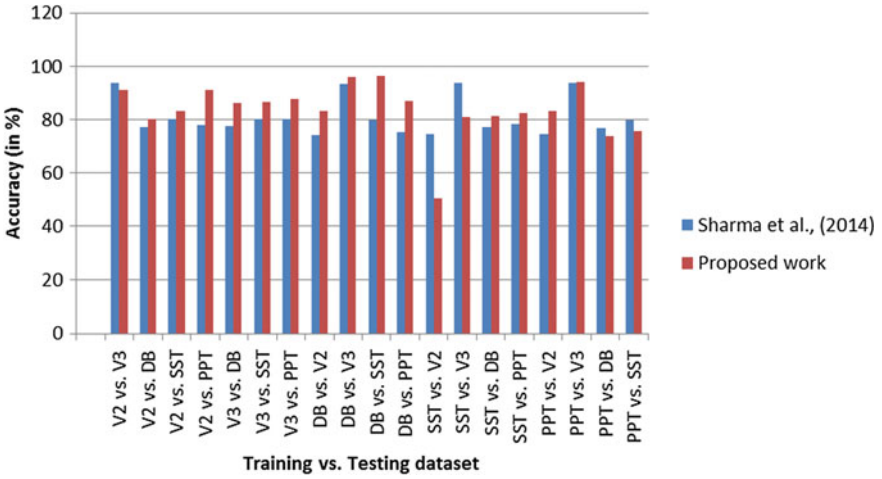

Fig. 4 NNET accuracy comparison (proposed work vs. Sharma et al., 2014 [2])

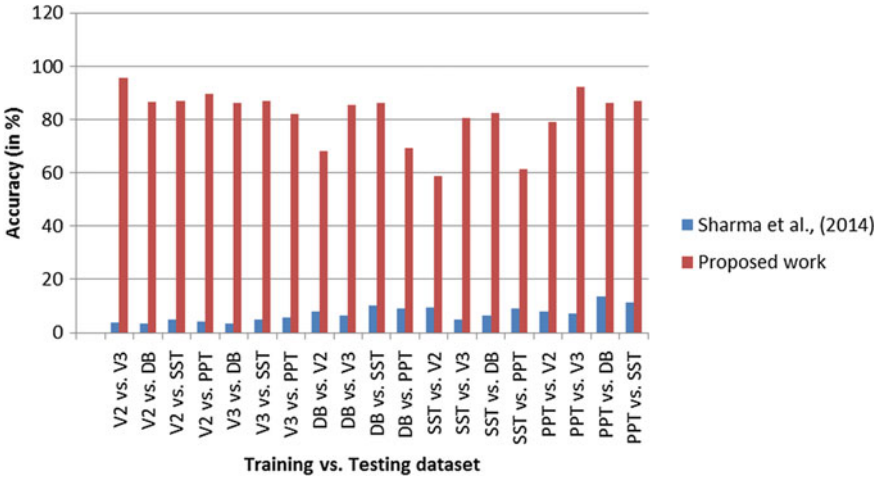

Fig. 5 NB accuracy comparison (proposed work vs. Sharma et al., 2014 [2])

based on these techniques predict the priority of a reported bug in cross-project context very accurately and outperform with the work available in the literature.

References

1. M. Sharma, P. Bedi, K.K. Chaturvedi, V.B. Singh, 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, 2012), pp. 539–545
2. M. Sharma, P. Bedi, V.B. Singh, An empirical evaluation of cross project priority prediction. Int. J. Syst. Assur. Eng. Manage. **5**(4), 651–663 (2014)
3. J. Kanwal, O. Maqbool, Managing open bug repositories through bug report prioritization using SVMs, in *Proceedings of the International Conference on Open-Source Systems and Technologies* (Lahore, Pakistan, 2010)
4. J. Kanwal, O. Maqbool, Bug prioritization to facilitate bug report triage. J. Comput. Sci. Technol. **27**(2), 397–412 (2012)
5. M. Alenezi, S. Banitaan, Bug reports prioritization: which features and classifier to use, in *12th International Conference on Machine Learning and Applications* (IEEE, 2013), pp. 112–116
6. L. Yu, W. Tsai, W. Zhao, F. Wu, Predicting defect priority based on neural networks, in *Proceedings of the 6th International Conference on Advanced Data Mining and Applications* (Wuhan, China, 2010), pp. 356–367
7. Y. Tian, D. Lo, C. Sun, DRONE: predicting priority of reported bugs by multi-factor analysis, in *IEEE International Conference on Software Maintenance* (2013), pp. 200–209
8. T. Menzies, A. Marcus, Automated severity assessment of software defect reports, in *Proceedings of International Conference on Software Maintenance* (IEEE, New York, 2008), pp. 346–355
9. T. Zimmermann, N. Nagappan, H. Gall, 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 Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering* (2009), pp. 91–100
10. B. Turhan, T. Menzies, A.B. Bener, J.D. Stefano, On the relative value of cross-company and within-company data for defect prediction. Empir. Softw. Eng. (2009). https://doi.org/10.1007/s10664-008-9103-7
11. Y. Ma, G. Luo, X. Zeng, A. Chen, Transfer learning for cross-company software defect prediction. Inf. Softw. Technol. **54**, 248–256 (2011)
12. Z. He, F. Shu, Y. Yang, M. Li, Q. Wang, An investigation on the feasibility of cross-project defect prediction, in *Automated Software Engineering* (2012), pp. 167–199
13. F. Peters, T. Menzies, A. Marcus, Better cross company defect prediction, in *10th IEEE Working Conference on Mining Software Repositories (MSR)* (IEEE, New York, 2013), pp. 409–418
14. G. Canfora, A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, S. Panichella, Multiobjective cross-project defect prediction, in *IEEE 6th International Conference on Software Testing, Verification and Validation (ICST)* (IEEE, New York, 2013), pp. 252–261
15. J. Nam, S.J. Pan, S. Kim, Transfer defect learning, in *Proceedings of International Conference on Software Engineering* (IEEE, New York, 2013), pp. 382–391
16. D. Ryu, O. Choi, J. Baik, Value-cognitive boosting with a support vector machine for cross-project defect prediction. Empir. Softw. Eng. **21**(1), 43–71 (2016)
17. V.B. Singh, S. Misra, M. Sharma, Bug severity assessment in cross project context and identifying training candidates. J. Inf. Knowl. Manage. **16**(01), 1750005 (2017)
18. A.E. Hassan, Predicting faults based on complexity of code change, in *Proceedings of International Conference on Software engineering (ICSE 09)* (2009), pp. 78–88
19. K.K. Chaturvedi, P.K. Kapur, S. Anand, V.B. Singh, Predicting the complexity of code changes using entropy-based measures. Int. J. Syst. Assur. Eng. Manage. Spr. **5**, 155–164 (2014)
20. S. Kamavaram, K. Goseva-Popstojanova, Entropy as a measure of uncertainty in software reliability, in *13th Int'l Symposium Software Reliability Engineering* (2002), pp. 209–210
21. I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, T. Euler, 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)
22. X. Wang, Y. He, Learning from uncertainty for big data. IEEE Syst. Man Cybern. Mag. 26–32 (2016)
23. [IEEE88], IEEE standard dictionary of measures to produce reliable software, IEEE Std 982.1-1988, Institute of Electrical and Electronics Engineers (1989). http://www.rapid-i.com