Table 2. Overall Results of XGBoost

S. No.	AUC	Sensitivity	Accuracy
1	0.795	69	76.55
2	**0.82**	**74.02**	**78.62**
3	0.80	70.71	76.9
4	0.80	69.71	76.55
5	0.75	62.55	73.45
6	0.80	70.24	76.55
7	0.81	71.99	76.21
8	0.79	69.23	72.41
9	0.81	71.07	77.59
10	0.82	73.61	76.21

As we can see from the above Table 1 that the AUC measure of the model comes out to be 0.91 for severity 2 and the corresponding sensitivity comes out to be 86.36%. Whereas, as the severity is decreasing (increasing number), the AUC measure declines to 0.83 for severity 3, corresponding sensitivity being 85.84%. For severity 4, these values further decrease to AUC being 0.81 and sensitivity being 70.77%. However, for the severity value 5, the AUC measure shows a slight increase, and the value comes out to be 0.85 while the sensitivity is 70%. The trend in these values shows that the model is the most efficient in predicting high severity (severity 2) values, though, other severities do not lag behind by much. Considering AUC and sensitivity as the measure for performance evaluation, we can say that the model is consistent with respect to all the severity levels.

From Table 2 we can see the overall results. The overall AUC which is obtained by the one versus one macro averaging comes out to be 0.82 while sensitivity comes out to be 74.02% with an overall accuracy of 78.62%.

4.4.2 CNN

The following Table 3 shows the individual results for each severity for various runs of CNN, and the overall results of the model obtained by taking the macro average of the metrics used are shown in Table 4. The results have been evaluated corresponding to the top 200 features and the evaluation metrics used are AUC, sensitivity, and accuracy.

Table 3. Individual Results of CNN

S.No.	Sev 2		Sev 3		Sev 4		Sev 5	
	AUC	Sensitivity	AUC	Sensitivity	AUC	Sensitiv-	AUC	Sensitivity
1	**0.95**	**87.23**	**0.89**	87.96	**0.91**	59.26	0.95	57.14
2	0.92	72.92	0.89	70.18	0.88	**84.51**	**0.96**	**72.78**
3	0.95	72.92	0.88	**92.97**	0.88	44.64	0.95	60
4	0.93	79.79	0.85	80.91	0.86	62.16	0.96	66.67
5	0.95	83.67	0.87	84.29	0.87	53.97	0.94	62.5

Unlike XGBoost, we can see from the above Table 3 that the highest value of AUC comes out to be 0.96 for severity 5 with corresponding sensitivity being 72.78%. The second highest value is obtained for severity 2 with a sensitivity of 87.23%. For severity 3, the AUC comes out to be 0.89, and the highest sensitivity obtained is 92.97%. For severity 4, the AUC obtained is 0.91, and the highest sensitivity obtained is 84.51%. As is clear from the above table, considering AUC and sensitivity as the evaluation measure, we can say that the performance of the model is consistent and the model is efficient for all the severities.

Table 4. Overall Results of CNN

S.No.	AUC	Sensitivity	Accuracy
1	**0.92**	72.90	**79**
2	0.91	**76.34**	75
3	0.91	67.63	76
4	0.90	72.38	75
5	0.91	71.11	77

The Table 4 shows the overall results of CNN over five runs of the partitioning variable. The overall results are calculated in a manner similar to that of XGBoost by taking the macro average of the individual results. The overall value of AUC is evaluated using one-versus-one macro averaging, which is insensitive to the class imbalance present in the data. The highest value of overall AUC comes out to be 0.92, while sensitivity and accuracy come out to be 76.34% and 79%, respectively. The high value of AUC suggests that the model is capable of differentiating between the classes.

4.4.3 RNN

Similar to XGBoost and CNN, RNN model performed various runs for different values of the partitioning variable. Individual results of each severity are being shown in Table 5. The results have been evaluated corresponding to the top 200 features, and the evaluation metrics used are AUC, sensitivity, and accuracy.

Table 5. Individual Results of RNN

S.No.	Sev 2		Sev 3		Sev 4		Sev 5	
	AUC	Sensitivity	AUC	Sensitivity	AUC	Sensitivity	AUC	Sensitivity
1	0.93	77.08	0.87	**87.72**	0.88	57.75	0.95	66.66
2	**0.96**	**85.42**	**0.88**	85.16	0.90	55.36	**0.96**	50
3	0.94	79.57	0.87	79.84	**0.91**	**72.73**	0.96	**71.43**
4	0.95	82.24	0.87	74.76	0.90	71.83	0.96	55.56
5	0.96	83.33	0.87	78.07	0.86	61.97	0.94	55.56

From the individual results for each severity (Table 5) we can deduce that the AUC value for both severity 2 and 5 comes out to be 0.96 with their respective sensitivities being 85.42% and 71.43%. For severity 3, AUC value is 0.88, and the sensitivity is 87.72%. For severity 4, AUC value is 0.91, and the sensitivity is 72.73%. Considering AUC and sensitivity as the evaluation measure, we can see that the performance of the model is consistent. The value of AUC being close to 1 suggests the model's capability to differentiate between the classes.

Table 6. Overall Results of RNN

S.No.	AUC	Sensitivity	Accuracy
1	0.91	72.3	76
2	0.91	68.98	78
3	**0.91**	**75.89**	**78**
4	0.91	71.09	76
5	0.89	69.73	75

Table 6 shows the overall results of the various runs of RNN obtained by macro averaging the individual results. The highest value of overall AUC comes out to be 0.91 while the highest sensitivity is 75.89 and the highest accuracy is 78. The value of AUC being so close to 1 suggests the model's capability to differentiate between the classes.

5. Conclusion

Softwares are present in every aspect of our lives, and it is quite inevitable for these softwares to have defects. Users report these defects with severity, and it helps developers to work on these defects to correct them in a priority wise manner. Our proposed approach employs XGBoost and deep learning techniques to predict the severity of defect reports. We have used the PITS A dataset available in NASA's PITS database to evaluate the model. The results were analysed using the AUC measure and sensitivity as the metrics. We concluded that with XGBoost, the model performs well in predicting the high severity (Severity 2) defects. While with both the deep learning techniques, CNN, and RNN, the models were exceptionally well in predicting the highest as well as the lowest priority of the defect reports (severity 2 and severity 5), and for the rest of the severities, the models' performances were fairly consistent.

As of now, we have based our work on PITS A dataset only and we aim to further extend it to other PITS datasets as well as other datasets.

References

[1] T. Menzies and A. Marcus, "Automated severity assessment of software defect reports", 2008 IEEE International Conference on Software Maintenance, Beijing, 2008, pp. 346-355.
[2] D. Cubranic and G. C. Murphy, "Automatic bug triage using text categorization", Proceedings of the International Conference on Software Engineering Knowledge Engineering, Alberta, 2004, pp.92-97.
[3] A. Lamkanfi, J. Péérez, S. Demeyer, "The eclipse and mozilla defect tracking dataset: A genuine dataset for mining bug information", Proc. 10th Work. Conf. Mining Softw. Repositories (MSR), May 2013, pp. 203-206.
[4] A. Lamkanfi, S. Demeyer, E. Giger, B. Goethals, "Predicting the severity of a reported bug", Proc. 7th IEEE Working Conf. Mining Softw. Repositories (MSR), May 2010, pp. 1-10.
[5] A. Lamkanfi, S. Demeyer, Q. D. Soetens, T. Verdonck, "Comparing mining algorithms for predicting the severity of a reported bug", Proc. 15th Eur. Conf. Softw. Maintenance Reeng., Mar. 2011, pp. 249-258.
[6] L. Huang, V. Ng, I. Persing et al., "AutoODC: Automated generation of orthogonal defect classifications", Autom Softw Eng 22, 2015, pp. 3-46.
[7] S. Patil, "Concept-Based Classification of Software Defect Reports," 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), Buenos Aires, 2017, pp. 182-186.
[8] C. Z. Yang, C. C. Hou, W. C. Kao, X Chen, "An empirical study on improving severity prediction of defect reports using feature selection", Proc. 19th Asia-Pacific Software Engineering Conference, 2012, pp. 240-249.
[9] G. Yang, S. Baek, J.-W. Lee, B. Lee, "Analyzing emotion words to predict severity of software bugs: A case study of open source projects", Proc. Symp. Appl. Comput., Apr. 2017, pp. 1280-1287.
[10] G. Yang, T. Zhang, B. Lee, "An emotion similarity based severity prediction of software bugs: A case study of open source projects", IEICE Trans. Inf. Syst., vol. E101.D, 2018, pp. 2015-2026.
[11] K. K. Chaturvedi, V. B. Singh, "Determining bug severity using machine learning techniques", Proc. CSI 6th Int. Conf. Softw. Eng. (CONSEG), Sep. 2012, pp. 1-6.
[12] W. Y. Ramay, Q. Umer, X. C. Yin, C. Zhu and I. Illahi, "Deep Neural Network-Based Severity Prediction of Bug Reports", in IEEE Access, vol. 7, 2019, pp. 46846-46857.
[13] T. Chen, C. Guestrin, "XGBoost: A Scalable Tree Boosting System", Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 785-794.
[14] J. Pennington, R. Socher, C. Manning, "Glove: Global Vectors for Word Representation", EMNLP. 14, 2014, pp. 1532-1543.
[15] R. Jindal, R. Malhotra, A. Jain, "Analysis of Software Project Reports for Defect Prediction Using KNN", Lecture Notes in Engineering and Computer Science, vol. 2211, no. 1, Jul. 2014, pp. 180–185.
[16] R. Jindal, R. Malhotra, A. Jain, "Software defect prediction using neural networks", Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization, Noida, 2014, pp. 1-6.
[17] R. Malhotra, N. Kapoor, R. Jain, S. Biyani, "Severity Assessment of Software Defect Reports using Text Classification", International Journal of Computer Applications, 83, 2013, pp. 13-16.