UPRIGHT PUBLICATIONS
https://upright.pub/index.php/ajase/

Original Contribution

A Study of Innovative Class Imbalance Dataset Software Defect Prediction Methods

Md Saikat Hosen[1], Wahiduzzaman Khan[2], Sai Srujan Gutlapalli[3]✉

Keywords: Software Defect Prediction, Imbalance Dataset, Innovative Class Imbalance

Asian Journal of Applied Science and Engineering

Vol. 10, Issue 1, 2021 [Pages 52-55]

Data mining for software defect prediction is a best approach for detecting problematic modules. On-hand classification methods can speed up knowledge discovery on class balance datasets. Actual facts are not balanced since one class dominates the other. These are class imbalance or skewed data sources. As class imbalance increases, fault prediction rate decreases. For class imbalance data streams, the suggested algorithms use unique oversampling and under sampling strategies to remove noisy and weak examples from both majority and minority. We test three techniques on class imbalance software defect datasets using four assessment measures. Results indicate that class imbalanced software defect datasets can be solved.

INTRODUCTION

Software defect prediction is the process of constructing predictive models that help in the early identification of defect-prone modules based on software metrics and defect data. These models are developed through a method called software defect prediction. It gives the project managers the ability to distribute the available resources in the most effective manner. In recent years, search-based techniques have seen widespread adoption as a means of offering optimal solutions to the development of efficient software defect prediction models. The term "hybridization" refers to the process of these strategies giving birth to new methods when combined with statistical or machine learning approaches. Software engineering refers to the process of developing software that has the characteristics that the user wants it to have. Software engineering encompasses a wide range of processes, each of which contributes to the final product. These processes include requirement analysis, designing, coding, and testing. It is a laborious task to perform thorough or exhaustive testing in order to locate all of the flaws that are present in the software modules.

The capabilities of the currently available software defect prediction techniques are insufficient for resolving the problems associated with class imbalance learning. There is a significant need for novel algorithms that can effectively train to anticipate software defects based on real-world class imbalance data sources, and there is a good gap or scope for this need. For an effective problem solution, a comprehensive grasp of the nature of the class imbalance, including concepts such as class disjunction, class drift, class imbalance ratio, and boarder line instance overlapping, needs to be explored. For the purpose of providing a comprehensive solution to the class imbalance problem inherent in software defect prediction, the unique algorithms that have been developed have accounted for a variety of viewpoints and scenarios.

[1]College of Management, Capital Normal University, Haidian District, Beijing, CHINA
[2]Professor & Head, School of Business, Ahsanullah University of Science and Technology, Dhaka, Bangladesh
[3]Data Engineer, TechnoVision Solutions LLC, Farmington Hills, MI 48335, USA
✉ Corresponding email: sgutlapalli07@gmail.com

LITERATURE REVIEW

A unique method for classification algorithm has been proposed by Mandapuram and Hosen (2018). This method makes use of the idea of simulated annealing in conjunction with under sampling techniques for a variety of jobs. Gutlapalli (2016b) has suggested combining minority sampling with SMOTE and the k-means algorithm in order to increase the predicted accuracy for both the majority and the minority class. Alsawalqah et al. (2017) have established a software data set approach for class imbalance software defects analysis specifically for the purpose of improved module identification for software flaws. Using kernel principal component analysis and a cost-sensitive technique for dimensionality and class imbalance reduction, Aleem et al. (2015) have developed a new method for learning about class imbalances.

A unique software defect prediction ensemble model that employs an oversampling technique has been proposed by Mandapuram et al. (2018) in order to achieve reliable prediction of defective minority samples. Gutlapalli (2017b) has presented methods that can effectively solve problems relating to the intrinsic characteristics of data, such as overlapping classes, borderline occurrences, and minor discounts, amongst other things. Song and Ye (2014) looked at the effects of class imbalance on software defect prediction datasets using a variety of under sampling techniques. An effective synthetic and adaptive oversampling technique has been proposed by Chang et al. (2011) for the purpose of performing effective outlier detection.

Song et al. (2018) used an artificial neural network multi class classifier to study a variety of resampling effects on datasets with class imbalances. Ge et al. (2018) have provided a comprehensive analysis of various strategies for addressing class imbalance data with the goal of enhancing the accomplishments of both majority and minority subclasses. By utilizing stop words and a variety of other one-of-a-kind methods, Desuky and Hussain (2021) have given a variety of methods for distinguishing fake jobs from original ones that are of a class imbalance character.

By implementing chunk-based reuse of rare class instances, Mandapuram et al. (2020) have presented an ensemble clustering approach for textually imbalanced data, particularly with concept drift. An ensemble-based deepboost classifier has been suggested by Desuky and Hussain (2021) to detect software fault modules in the scenario of class imbalance nature and high dimensionality. The influence of various resampling methods and decision tree classifiers on class imbalance data sources was analyzed by Mandapuram (2017b), and the results were compared and discussed. In addition to this, the study focuses on methods that are sensitive to costs in order to address the class imbalance problem. An improved strategy to under sampling that may be applied to cardio vascular data has been proposed by Reddy et al. (2020). In order to find a solution to the class imbalance learning problem, Mandapuram (2017a) conducted research on a number of different shot learning algorithms.

PROPOSED ALGORITHMS

Improved Correlation over Sampling (ICOS)

This approach makes use of the recently developed strategy of better correlation for up sampling of minority subsets of instances in order to improve the performance of software defect datasets. An in-depth investigation on class imbalance datasets of software defect prediction was carried out by Bodepudi et al. (2019).

SOFTWARE METRICS

A proportion of quantifiable or countable qualities that can be used to measure and predict the quality of software is referred to as a software metric. A metric is an indication that describes a particular characteristic of a piece of software. The determination and measurement of software metrics is essential for a variety of reasons, including the estimation of programming execution, the measurement of the effectiveness of software processes, the estimation of the required efforts for processes, the deduction of defects during software development, and the monitoring and controlling of software project executions.

A wide variety of software metrics have frequently been utilized for the purpose of defect prediction. The very first category of software metrics is referred to as lines of code (LOC) metrics, and they are regarded as the fundamental software metrics. Metrics based on lines of code are representative of normal software development proportions. Numerous studies conducted in the field of SDP have conclusively demonstrated a direct connection between LOC measurements and defect prediction. The cyclomatic complexity metrics are one of the most prevalent software metrics that are commonly used for SDP. These metrics, which were proposed by Mandapuram (2016) and are used to represent the complexity of software products, are one of the most widely used software metrics. Counting the

number of nodes, arcs, and related components is how Mandapuram's metrics, also known as cyclomatic metrics, are generated. These metrics are based on the control flow graphs of a source code. Before beginning the coding process, Gutlapalli et al. (2019) utilized McCabe's cyclomatic metrics to make an accurate prediction of defect-prone modules. McCabe's cyclomatic metrics have been utilized in the construction of SDP models in a great number of earlier works. The software size measurements that were proposed by Gutlapalli (2017a) are yet another grouping of software measures. The amount of operands and operators from source codes are used as the basis for the program size metrics used by Halstead. In addition, these metrics have been employed in SDP, and they are connected to the scale of the program in terms of its vocabulary, length, volume, difficulty, effort, and time.

The majority of software failure prediction approaches, as stated by Gutlapalli (2017c), focus on object-oriented software metrics. Several software measures, collectively referred to as CK object-oriented metrics, were proposed by Gutlapalli (2016a). These metrics include the depth of inheritance tree (DIT), weighted method per class (WMC), number of children (NOC), and others. SDP has made use of a great number of research that employ object-oriented metrics. Song and Ye (2014) discovered useful software metrics that were implemented in SDP with the intention of improving software quality by locating flaws. According to the findings of their research (Aleem et al., 2015), object-oriented and process metrics were more helpful in discovering problems than other size and complexity metrics. This was the conclusion drawn from their investigation.

RESULTS AND DISCUSSION

The findings of the experiments and the comparison study are reported here. Based on the findings, it appears that the IISS algorithm performed better than the ICOS and USS algorithms when applied to all of the datasets that were analyzed. The up sampling as well as the down sampling are both incorporated into the IISS algorithm, which is one of the reasons for the improved performance.

The area under the curve (AUC) evaluation measure is one of the prominent metrics that is employed in a great number of benchmark research investigations of the class imbalance nature on software defect prediction datasets. When compared across all datasets, the AUC scores produced by the IISS algorithm are noticeably superior to those produced by the ICOS and USS algorithms. Some of the other well-known criteria that are used for evaluating software defect prediction for class imbalance learning include precision, recall, and the F-measure. In comparison to ICOS and USS, the performance of IISS was also quite satisfactory across the majority of the datasets. We are able to reach the conclusion that the IISS strategy is one of the best methods for efficient knowledge discovery in the context of the prediction of software defects caused by class imbalance datasets.

CONCLUSION

In this study, different unique software defect classification algorithms are contrasted with one another in order to identify the benefits and drawbacks of each approach. This technique solves the problem of class drifts in data streams in an effective manner by employing novel oversampling and up sampling methodologies. The presented methods have been demonstrated to be more effective than other approaches that have been examined in terms of their AUC, accuracy, and precision, as well as their f-measure.

REFERENCES

Bodepudi, A., Reddy, M., Gutlapalli, S. S., & Mandapuram, M. (2019). Voice Recognition Systems in the Cloud Networks: Has It Reached Its Full Potential?. *Asian Journal of Applied Science and Engineering*, *8*(1), 51–60. https://doi.org/10.18034/ajase.v8i1.12

Gutlapalli, S. S. (2016a). An Examination of Nanotechnology's Role as an Integral Part of Electronics. *ABC Research Alert*, *4*(3), 21–27. https://doi.org/10.18034/ra.v4i3.651

Gutlapalli, S. S. (2016b). Commercial Applications of Blockchain and Distributed Ledger Technology. *Engineering International*, *4*(2), 89–94. https://doi.org/10.18034/ei.v4i2.653

Gutlapalli, S. S. (2017a). Analysis of Multimodal Data Using Deep Learning and Machine Learning. *Asian Journal of Humanity, Art and Literature*, *4*(2), 171–176. https://doi.org/10.18034/ajhal.v4i2.658

Gutlapalli, S. S. (2017b). The Role of Deep Learning in the Fourth Industrial Revolution: A Digital Transformation Approach. *Asian Accounting and Auditing Advancement*, *8*(1), 52–56. Retrieved from https://4ajournal.com/article/view/77

Gutlapalli, S. S. (2017c). An Early Cautionary Scan of the Security Risks of the Internet of Things. *Asian Journal of Applied Science and Engineering*, *6*, 163–168. Retrieved from https://ajase.net/article/view/14

Gutlapalli, S. S., Mandapuram, M., Reddy, M., & Bodepudi, A. (2019). Evaluation of Hospital Information Systems (HIS) in terms of their Suitability for Tasks. *Malaysian Journal of Medical and Biological Research*, *6*(2), 143–150. https://doi.org/10.18034/mjmbr.v6i2.661

Mandapuram, M. (2016). Applications of Blockchain and Distributed Ledger Technology (DLT) in Commercial Settings. *Asian Accounting and Auditing Advancement*, *7*(1), 50–57. Retrieved from https://4ajournal.com/article/view/76

Mandapuram, M. (2017a). Application of Artificial Intelligence in Contemporary Business: An Analysis for Content Management System Optimization. *Asian Business Review*, *7*(3), 117–122. https://doi.org/10.18034/abr.v7i3.650

Mandapuram, M. (2017b). Security Risk Analysis of the Internet of Things: An Early Cautionary Scan. *ABC Research Alert*, *5*(3), 49–55. https://doi.org/10.18034/ra.v5i3.650

Mandapuram, M., & Hosen, M. F. (2018). The Object-Oriented Database Management System versus the Relational Database Management System: A Comparison. *Global Disclosure of Economics and Business*, *7*(2), 89–96. https://doi.org/10.18034/gdeb.v7i2.657

Mandapuram, M., Gutlapalli, S. S., Bodepudi, A., & Reddy, M. (2018). Investigating the Prospects of Generative Artificial Intelligence. *Asian Journal of Humanity, Art and Literature*, *5*(2), 167–174. https://doi.org/10.18034/ajhal.v5i2.659

Mandapuram, M., Gutlapalli, S. S., Reddy, M., Bodepudi, A. (2020). Application of Artificial Intelligence (AI) Technologies to Accelerate Market Segmentation. *Global Disclosure of Economics and Business* *9*(2), 141–150. https://doi.org/10.18034/gdeb.v9i2.662

Reddy, M., Bodepudi, A., Mandapuram, M., & Gutlapalli, S. S. (2020). Face Detection and Recognition Techniques through the Cloud Network: An Exploratory Study. *ABC Journal of Advanced Research*, *9*(2), 103–114. https://doi.org/10.18034/abcjar.v9i2.660

Desuky, A., S., & Hussain, S. (2021). An Improved Hybrid Approach for Handling Class Imbalance Problem. *Arabian Journal for Science and Engineering*, *46*, 3853– 3864. https://doi.org/10.1007/s13369-021-05347-7

Ge, J., Liu, J. and Liu, W. (2018). Comparative Study on Defect Prediction Algorithms of Supervised Learning Software Based on Imbalanced Classification Data Sets. 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 27-29 June 2018, Busan, 399-406. https://doi.org/10.1109/SNPD.2018.8441143

Song, Q., Guo, Y. and Shepperd, M. (2018). A Comprehensive Investigation of the Role of Imbalanced Learning for Software Defect Prediction. IEEE Transactions on Software Engineering, 1. https://doi.org/10.1109/TSE.2018.2836442

Chang, R.H., Mu, X.D. and Zhang, L. (2011). Software Defect Prediction Using Non-Negative Matrix Factorization. Journal of Software, 6, 2114-2120. https://doi.org/10.4304/jsw.6.11.2114-2120

Alsawalqah, H., Faris, H., Aljarah, I., Alnemer, L. and Alhindawi, N. (2017). Hybrid Smote-Ensemble Approach for Software Defect Prediction. In: Silhavy, R., Silhavy, P., Prokopova, Z., Senkerik, R. and Oplatkova, Z., Eds., Software Engineering Trends and Techniques in Intelligent Systems, Springer, Berlin, 355-366. https://doi.org/10.1007/978-3-319-57141-6_39

Aleem, S., Capretz, L. and Ahmed, F. (2015). Benchmarking Machine Learning Technologies for Software Defect Detection. International Journal of Software Engineering & Applications, 6, 11-23. https://doi.org/10.5121/ijsea.2015.6302

Song, G. and Ye, Y. (2014). A Dynamic Ensemble Framework for Mining Textual Streams with Class Imbalance. *Scientific World Journal*, Article ID 497354. https://doi.org/10.1155/2014/497354

--0--