3/15/2017

Innovative Method for the Prediction of Software Defects Based on Class Imbalance Datasets

Sreekanth Dekkati, Upendar Rao Thaduri

技术与管理回顾

HTTPS://UPRIGHT.PUB/INDEX.PHP/TMR/

Tech nology and Management Review
技术与管理回顾

Vol 2, No 1 (2017)

Innovative Method for the Prediction of Software Defects Based on Class Imbalance Datasets

[1]**Sreekanth Dekkati,** *System Administrator, MUFG Bank, Arizona, USA*
[2]**Upendar Rao Thaduri,** *Programmer Analyst, Ameritek Global Inc, New Jersey, USA*

Abstract:
The statistics that represent the real world are not perfectly balanced because any class can predominantly increase in ratio with the other classes. This category of data sources is sometimes called skewed or class imbalance data sources. One of the most effective methods for locating defective modules is called software defect prediction, which uses data mining techniques. The classification methods that are now available can be utilized for effective knowledge discovery on datasets that have a class balance. As the class imbalance nature of the datasets becomes more prevalent, the defect prediction rate for those datasets will, in turn, become less accurate. The proposed algorithms include a novel oversampling strategy and an under-sampling technique. These techniques are implemented by deleting noisy and weak instances from both the majority and the minority. This is done to improve the performance of class imbalance data streams. The findings prove that the class imbalance issue in software defect datasets may be effectively addressed and resolved. Experiments are carried out on datasets of software defects that are class-imbalanced.

Keywords: Software Defect Prediction, Software Defect Datasets, Imbalanced Datasets

INTRODUCTION

The capabilities of the currently available software defect prediction techniques must be improved to resolve 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 suitable gap or scope for this need (Desamsetti, 2016). For an effective solution to the problem, a comprehensive grasp of the nature of the class imbalance, including concepts such as class disjunction, class drift, class imbalance ratio, and borderline instance overlapping, should be examined. The unique algorithms that have been developed have accounted for various viewpoints and scenarios to provide a comprehensive solution to the class imbalance problem inherent in software defect prediction.

Software engineering refers to the process of developing software that has the characteristics that the user wants it to have. Software engineering encompasses various processes, each contributing to the final product. These processes include requirement analysis, designing, coding, and testing. It is a laborious task to perform thorough or exhaustive testing to locate all of the flaws in the software modules (Thaduri et al., 2016).

The following is included in the remaining portion of the paper: The most important current works in connection with software defect prediction are discussed in this article. In the next part, you can find the conceptual frameworks for the comparative research approaches. A conclusion and recommendations for subsequent work are given after the presentation of the experimental methods, evaluation criteria, and findings discussion.

ISSUES IN SOFTWARE DEFECT PREDICTION

This section provides an introduction to the challenges present in software defect prediction and the solutions that prominent researchers have suggested for these challenges. Additionally, it addresses the problems that have yet to be solved in this field.

Relationship between the Characteristics and the Defect

The researchers cannot pinpoint a broad subset of characteristics that function as a significant determinant in determining whether or not a module is flawed. In addition, there has been debate on the degree of metrics that should be applied for analysis: requirement, design, or source code metrics. The following is an overview of the studies that have attempted to solve the abovementioned issue through various research methods. This research made use of object-oriented principles in its methodology. The analysis was carried out using the area under the ROC curve (AUC) serving as the criterion for evaluation. The findings led the researchers to conclude that fault proneness is highly connected with inheritance and export coupling (EC).

There are no standardized measures for evaluating employee performance.

The selection of the performance reporting metrics is fraught with significant variation across various topic domains. For analyzing and comparing the defect prediction models, no common criterion has been established. This section provides a condensed account of the work that several distinguished researchers have done in this area.

Cross-Project Defect Prediction Issues

Using locally accessible data, often identical to the data to be evaluated, is desirable for model learning. This local data can come from previous versions of the same project or another similar project using the same programming language (Gutlapalli, 2016a). However, the risk management team of a business sometimes needs more local site information. Training data may be unavailable because no equivalent project has been established or technology has evolved. Researchers developed cross-project defect prediction to overcome this challenge by developing a model for one project and testing it on another. These two projects may be identical or distinct. Unfortunately, cross-company data models have performed poorly.

There Is Not a Comprehensive Plan Available

One more challenge posed by this burgeoning discipline is that numerous researchers and academics in the field have employed a variety of methodologies on various data sets. However, there has yet to be a standardized structure or approach that can be followed in order to use a software defect prediction process on a local or cross-company project. The implementation of software defect prediction models has been the subject of a few research studies, each offering a unique framework.

Software defect prediction and its economic implications

The irony of the field of software defect prediction is that the majority of the work that has been done on the subject has concentrated on the program's ease of use. At the same time, very few studies have investigated the field's economic standing. Misclassification can be rather costly, particularly when incorrectly identifying a damaged component as healthy. Therefore, it is vital to determine when it can be helpful and how much utility it has.

Poor Class Balance

Training data class distribution strongly affects Software Fault prediction model efficiency. Class distribution is the number of instances of each class in the training dataset. Class imbalance occurs when one class has many more instances than another. The majority class has more occurrences than the minority class. When the defective class has fewer instances, the problem grows.

PROPOSED ALGORITHMS

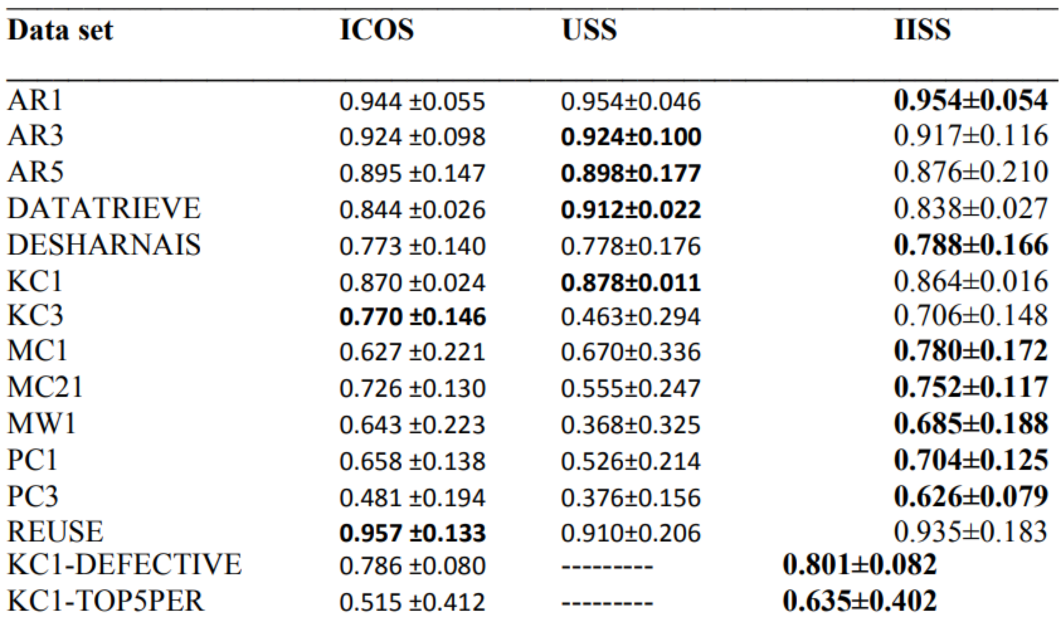
The acronym ICOS stands for improved correlation oversampling.

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 of class imbalance datasets of software defect prediction was carried out by Gutlapalli (2016a).

RESULTS AND DISCUSSION

According to the findings, the IISS algorithm has done better than the ICOS and USS algorithms on all analyzed datasets. The up-sampling and the down-sampling are incorporated into the IISS algorithm, which is one of the reasons for the improved performance.

Table 1: Precision results of the IISS model on SDP datasets



Data set	ICOS	USS	IISS
AR1	0.944 ±0.055	0.954±0.046	**0.954±0.054**
AR3	0.924 ±0.098	**0.924±0.100**	0.917±0.116
AR5	0.895 ±0.147	**0.898±0.177**	0.876±0.210
DATATRIEVE	0.844 ±0.026	**0.912±0.022**	0.838±0.027
DESHARNAIS	0.773 ±0.140	0.778±0.176	**0.788±0.166**
KC1	0.870 ±0.024	**0.878±0.011**	0.864±0.016
KC3	**0.770 ±0.146**	0.463±0.294	0.706±0.148
MC1	0.627 ±0.221	0.670±0.336	**0.780±0.172**
MC21	0.726 ±0.130	0.555±0.247	**0.752±0.117**
MW1	0.643 ±0.223	0.368±0.325	**0.685±0.188**
PC1	0.658 ±0.138	0.526±0.214	**0.704±0.125**
PC3	0.481 ±0.194	0.376±0.156	**0.626±0.079**
REUSE	**0.957 ±0.133**	0.910±0.206	0.935±0.183
KC1-DEFECTIVE	0.786 ±0.080	---------	**0.801±0.082**
KC1-TOP5PER	0.515 ±0.412	---------	**0.635±0.402**

Please note that the boldface value indicates the algorithm with the best performance.

The area under the curve (AUC) evaluation measure is one of the prominent metrics employed in many benchmark research investigations of the class imbalance nature of software defect prediction datasets. The AUC results obtained using the IISS algorithm are significantly superior to those obtained through the ICOS and USS algorithms for each of the datasets. Other well-known criteria for evaluating software defect prediction for class imbalance learning include precision, recall, and F-measure. These metrics are used to evaluate the accuracy of the program's defect prediction (Mandapuram, 2016). Compared to ICOS and USS, the performance of IISS was also entirely satisfactory across most of the datasets. The IISS strategy is one of the best methods for efficient knowledge discovery in predicting software defects caused by class imbalance datasets.

CONCLUSION

Additional research can be conducted utilizing all combinations and subsets of attributes, as well as any and all classifiers, to determine whether or not there is a connection between attributes and flaws. In addition, the investigations could determine whether or not there is a connection between characteristics and deficiencies. Finding out the importance of each measure can be done to evaluate the performance of a prediction model. Then, depending on the value of the individual measurements, a new measure can be offered that is a weighted combination of all of the measures. In the case of cross-project defect prediction, the data should first be improved by some pre-processing approach to become comparable to the localized data before the model can be trained using the cross-project defect data. This should be done before the model is trained using the cross-project defect data. In order to create a generic framework, one should consider the influence of classifier combination, the order of training data, the effect of pre-processing techniques, and attribute selection on the prediction model's performance.

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 effectively solves the problem of class drifts in data streams by employing novel oversampling and upsampling methodologies. Regarding area under the curve (AUC), accuracy, precision, and measure, empirical findings have demonstrated that the proposed algorithms perform better than other approaches.

REFERENCES

Desamsetti, H. (2016). Issues with the Cloud Computing Technology. *International Research Journal of Engineering and Technology (IRJET)*, *3*(5), 321-323.

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

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

Thaduri, U. R., Ballamudi, V. K. R., Dekkati, S., & Mandapuram, M. (2016). Making the Cloud Adoption Decisions: Gaining Advantages from Taking an Integrated Approach. *International Journal of Reciprocal Symmetry and Theoretical Physics*, *3*, 11–16. https://upright.pub/index.php/ijrstp/article/view/77