A Cost-Effective Strategy for Software Vulnerability Prediction Based on Bellwether Analysis*

Patrick Kwaku Kudjo
School of Computer Science & Communication Engineering
Jiangsu University
Zhenjiang, 212013, China
kudjo@ujs.edu.cn

Jinfu Chen*
School of Computer Science & Communication Engineering
Jiangsu University
Zhenjiang, 212013, China
*Corresponding author: jinfuchen@ujs.edu.cn

ABSTRACT

Vulnerability Prediction Models (VPMs) aims to identify vulnerable and non-vulnerable components in large software systems. Consequently, VPMs presents three major drawbacks (i) finding an effective method to identify a representative set of features from which to construct an effective model. (ii) the way the features are utilized in the machine learning setup (iii) making an implicit assumption that parameter optimization would not change the outcome of VPMs. To address these limitations, we investigate the significant effect of the *Bellwether* analysis on VPMs. Specifically, we first develop a *Bellwether* algorithm to identify and select an exemplary subset of data to be considered as the *Bellw*ether to yield improved prediction accuracy against the *growing portfolio* benchmark. Next, we build a machine learning approach with different parameter settings to show the improvement of performance of VPMs. The prediction results of the suggested models were assessed in terms of precision, recall, F-measure, and other statistical measures. The preliminary result shows the *Bellwether* approach outperforms the benchmark technique across the applications studied with F-measure values ranging from 51.1% - 98.5%.

CCS CONCEPTS

• **Software and its engineering →Data mining and security**

KEYWORDS

Software vulnerability, Bellwether, Machine learning, Tuning

ACM Reference format:

Patrick Kwaku Kudjo, Jinfu Chen. 2019. A Cost-Effective Strategy for Software Vulnerability Prediction Based on Bellwether Analysis. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'19),* July 15-19, 2019, Beijing, China. ACM, New York, NY, USA, 4 pages. http://doi.org/10.1145/3293882.3338985

*It is a special policy of our institution for the supervisor of a foreign student to be included in the related articles to meet the graduation criterion.

1 INTRODUCTION

Developing accurate vulnerability prediction models (VPMs) are valuable assets for researchers and practitioners in the software industry. The classification of vulnerability is a crucial procedure since it can be used by security experts to direct software assurance efforts to focus on the components classified as vulnerable, which are more likely to contain vulnerabilities than the neutral components [1]. Most studies rely on machine learning algorithms on software metrics and text mining features to predict vulnerable software components. For instance, Tang et al. [2] investigated the predictive power of software metrics and text mining features from an effort-aware perspective. Stuckman et al. [1] examined the effect of dimensionality reduction techniques using software metrics and text mining features. In this context, the features represent the independent variable or explanatory variable.

According to Stuckman et al. [1], the quality of the trained models depends on the characteristics of the features. In other words, the performances of VPMs are known to be dependent on the features for training these models. Despite that some researchers have conducted empirical studies to examine the impact of feature selection on VPMs, most of these studies failed to: (i) identify a representative set of features from which to construct an effective and efficient model (exceptions:[1],[3]). (ii) explore the significant effect of feature subset on VPMs. (iii) acknowledge the impact of parameter optimization (i.e. parameter tuning) on VPMs (exceptions: [4],[5],[6]) (iv) perform a comprehensive evaluation of the models. Meanwhile, it is clearly established in other domains of studies such as defect prediction [7], bug prediction [8] that using a small number of features avoids the so-called '*dimensionality problem*' and improves prediction accuracy. Similarly, Fu et al. [9] showed that tuning a machine learning algorithm can alter detection precision from 0%-60%. Considering these points, the goal of our research is to develop a VPM using *Bellwether* (i.e. *exemplary data*), which has been proven reliable by Chen et al. [10] and Krishna et al. [11]. Specifically, we first develop a *Bellwether* algorithm to identify and select an exemplary subset of data to be considered as the *Bellwether* to yield improved prediction accuracy against the *growing portfolio.*

Next, we build a machine learning approach with different parameter settings. Furthermore, the prediction results of the suggested models were assessed in terms of precision, recall, F-measure and other statistical measures.

Our expected contributions are :

i. We empirically validate the significant effect of the *Bellwether* analysis on VPMs.
ii. We propose a novel strategy to identify and select an exemplary subset of data to be considered as the training set for improved prediction accuracy.
iii. Conduct an extensive experiment using three publicly available vulnerability datasets.
iv. We investigate the extent to which parameter tuning affect the performance of vulnerability prediction models.

2 BACKGROUND AND RELATED WORK

This section provides the background and concept that are vital for understanding the proposed approach.

2.1 Feature Selection

A common and widely known problem in machine learning is the *curse of dimensionality*, which usually affect the reliability of machine algorithms. An efficient solution is to apply feature selection and feature engineering techniques. Feature selection is the process of extracting a number of features subsets, which are the most representative of the original subsets [12]. The feature selection algorithms can be categorized as supervised, unsupervised and semi-supervised algorithms. From the perspective of subset selection strategy, the feature selection techniques are broadly classified into three main categories: filter, wrapper and embedded. The filter methods commonly referred to as the relevance metric uses a heuristic criterion to evaluate the intrinsic characteristics of each feature [13]. The wrapper method employs a search strategy to select the relevant features and evaluate the obtained subset by using a learning mechanism such as C4.5 and Bayesian Network [14]. The embedded methods perform variable selection during the process of training and are generally specific to a given learning algorithm. Strictly speaking, the filter methods are usually fast but less accurate than the wrapper method because they do not take the underlying machine learning algorithms into account [8]. Unfortunately, the importance of feature selection for building quality prediction models in often over-looked in VPMs. To the best of our knowledge, there are only two empirical studies [1],[15] that examined the impact of feature selection on VPMs among the vast plethora of vulnerability prediction studies. For example, Pang et al. [15] proposed a statistical feature selection method to select a subset of features to improve vulnerability prediction. They used a feature ranking approach for excluding a large number of irrelevant and less important features where a small number of features are kept and the rest is discarded. Similarly, Stuckman et al. [1] examined the effect of three dimensionality reduction techniques on VPMs. This observation leads us to investigate the effect of the *Bellwether analysis* on the performance of VPMs. The current study differs from the approach presented in [1],[15] in that we used a different strategy as a measure to show the improvement of performance of VPMs.

2.2 Concept of Bellwether

To the best of our knowledge, the *Bellwether* concept was first considered by Chen et al. [10] in the context of Online Analytical Processing. Their aim was to find a *small subset* of queries that can be used to make an accurate prediction of the target of a new query. In defect prediction, Krishna et al. [11] implemented the concept to find *exemplary projects* from a set of non chronological projects to successfully make predictions on the remaining projects. Each project was considered as a potential *Bellwether* in each iteration on the prediction of the remaining projects until the project data with the best prediction accuracy was obtained. The *Bellwether* concept uses two main approaches, namely the *Bellwether effect* and *Bellwether method.* The *Bellwether effect* defines the existence of exemplary projects within historical dataset that is used to build an accurate prediction model. The *Bellwether method* defines the search process used in obtaining exemplary projects [11]. Chen et al. [10] defined three main *Bellwether* problems, namely basic, subset and combinatorial *Bellwether* problem. The subset problem essentially explores how to create subsets of items and find the *Bellwether* region for each created subset, so that we can make the best predictions for new items by using the *Bellwether* for these subsets. In our case, given a set of N vulnerability data from a particular dataset D, we seek to obtain a subset of n cases that can be considered as the *Bellwether* to yield improved prediction accuracy against the *growing portfolio* benchmark.

2.3 Benchmark Technique

In software engineering specifically in the domain of software effort estimation, *growing portfolio* refers to a set of all available historical data extracted from software project cases [16]. It is referred to as the *growing portfolio* because the data is collected over time, thus the relevance of the term growing is the accumulation of data over a period. Lokan and Mendes [16] as well as Amasaki and Lokan [17] sampled a subset of *growing portfolio* based on a heuristic approach and referred to that subset as the moving window. A moving window refers to a subset of recently completed projects.

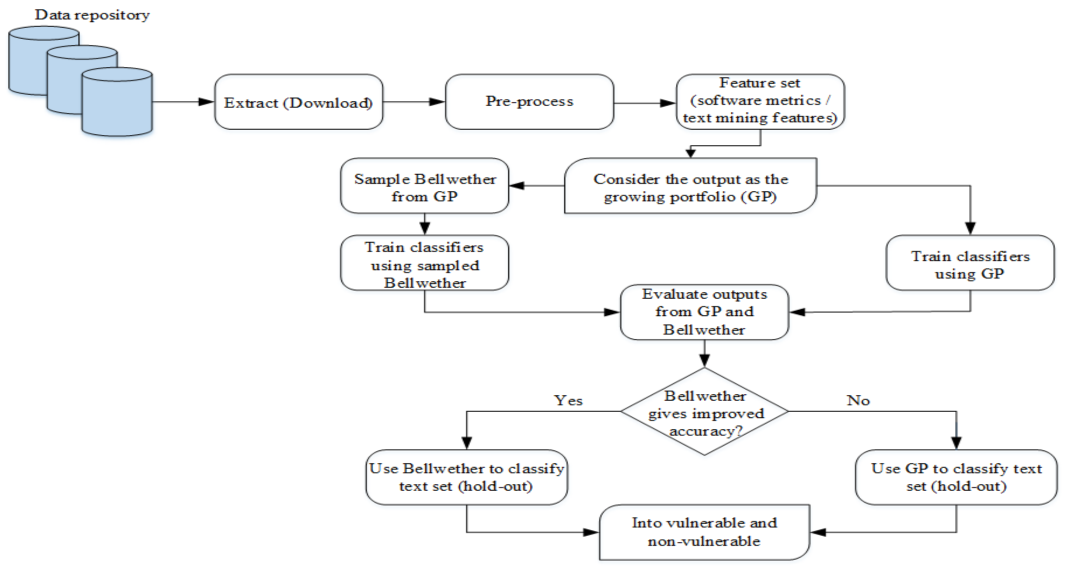
Data repository
Extract (Download)
Pre-process
Feature set (software metrics / text mining features)
Consider the output as the growing portfolio (GP)
Sample Bellwether from GP
Train classifiers using sampled Bellwether
Train classifiers using GP
Evaluate outputs from GP and Bellwether
Bellwether gives improved accuracy?
Yes
No
Use Bellwether to classify text set (hold-out)
Use GP to classify text set (hold-out)
Into vulnerable and non-vulnerable

Figure 1: Proposed Framework for Software Vulnerability Prediction

This *moving window* subset was used to investigate whether it will yield improved. With inspiration from studies by Chen et al. [10] and Lokan and Mendes [16], we seek to investigate the existence of a subset of exemplary vulnerability data extracted from entire collection of available vulnerability cases (referred to as the *growing portfolio),* that can yield improved prediction accuracy against the growing portfolio benchmark in relation to setting up vulnerability prediction models. Thus, *growing portfolio* refers to the entire collection of available vulnerability data for a given dataset from which a label (vulnerable or neutral) exists.

3 PROPOSED FRAMEWORK

Fig.1 presents the overall framework of our approach.

3.1 Dataset

As a feasibility study for our research problem, we selected applications for this research based on the following criteria: (i) the application must be an open-source project (ii) the selected applications must have a large number of confirmed security vulnerabilities. (iii) the selected application has been validated in previous studies. To this aim, we conducted the experiment on vulnerability prediction dataset provided by Walden et al. [18]. The dataset contains a total number of 223 vulnerabilities from multiple versions of three applications systems, namely Drupal, Moodle, and PhpMyAdmin. We utilized a java utility program, R-script, and other applications in [19] to extract and organize the features.

3.2 Dependent and Independent Variable

The dependent variable used in our study is the same as that in [1],[18]. Specifically, we extracted vulnerability reports from vulnerability database and localized each vulnerability to the file where it exists during the analysis of the version. The mapping from vulnerabilities to files was used to define the dependent variable. In this case, a file is labeled vulnerable if it contains at least one vulnerability and non-vulnerable if there are no vulnerabilities. The independent variable is also the same as those in [1],[18]. That is software metrics and text mining features.

4 IMPLEMENTATION

This section discusses our empirical evaluation.

4.1 Machine Learning Techniques

We used five different classification algorithms, namely Random forest, deep neural network, k-nearest neighbor, Logistic regression and Support Vector Machine. We chose these models because they fall into different categories of learning methods and partly due to the fact that they achieved state-of-the-art performance in previous studies [1],[15]. Note that we used the open-source suite of machine learning software Weka tool [20] for the experiment.

4.2 Evaluation Metrics

For each application, we ran a stratified 10-fold cross-validation experiment with the Weka tool and assessed the performance of the models using precision, recall, and F-measure. In the 10-fold cross-validation approach, we randomly split each dataset into 10-fold, where each time 9-fold are used as training dataset and the remaining one fold for validation purpose.

5 RESULTS

We present the preliminary results of the experiment in this section. Our models present precision values ranging from 48.5%-89.6% and 40.9%-89.4% for the *Bellwether* approach and *growing portfolio* respectively. In terms of recall, we recorded values ranging from 13.5%-99.9% for the *Bellwether* approach and 10.9%-88.99% *growing portfolio.* Although we achieved low recall values in some cases, what really matters is the performance comparison of the model in terms of F-measure. Our F-measure values ranging from 54%-98.5% across the applications studied for the *Bellwether* approach which is an improvement over the benchmark technique. Furthermore, we noticed that a combination of random forest and the *Bellwether* approach yields the highest F-measure value. Finally, the k-nearest neighbor and deep neural network are the most stable classification models measured by our performance metrics. We think that the reduction in the size of the training dataset accounts for the improved prediction accuracy of our models. The findings of this study confirm prior research in [9] that parameter turning improves predictive performance of data miners.

6 CONCLUSION AND FUTURE WORK

This paper proposed a framework for vulnerability prediction based on the *Bellwether* analysis using machine learning techniques. In addition, we develop a framework to show the improvement of performance of VPMs when automated parameter is applied. We have demonstrated through a case study that this goal is feasible and achieve better accuracy (measured in terms of F-measure) at a lower cost. The remaining of this work now is to build on the study to improve its application to other datasets such as Android applications, blocking bug prediction, intrusion detection, exploit detection and vulnerability severity assessment. We also intend to explore the practicalities of the combinatorial *Bellwether* problem within the context of vulnerability prediction.

ACKNOWLEDGMENTS

This study was funded by the National Natural Science Foundation of China (NSFC grant numbers: U1836116 and 61872167), the Project of Jiangsu Provincial Six Talent Peaks (Grant numbers: XYDXXJS-016), and the Graduate Research Innovation Project of Jiangsu Province (Grant numbers: KYCX17 1807).

REFERENCES

[1] J. Stuckman, J. Walden, and R. Scandariato, "The effect of dimensionality reduction on software vulnerability prediction models,"*IEEE Transactions on Reliability*, vol. 66, no. 1, pp. 17–37, 2017.

[2] Y. Tang, F. Zhao, Y. Yang, H. Lu, Y. Zhou, and B. Xu, "Predicting vulnerable components via text mining or software metrics? an effort-aware perspective," *IEEE International Conference on Software Quality, Reliability and Security (QRS), 2015*, pp. 27–36, 2015.

[3] Y. Pang, X. Xue, and A. S. Namin, "Predicting vulnerable software components through n-gram analysis and statistical feature selection," in *Proceedings of the 14th IEEE International Conference on Machine Learning and Applications (ICMLA), 2015*, 2015, pp. 543–548.

[4] Z. Han, X. Li, Z. Xing, et al., "Learning to Predict Severity of Software Vulnerability Using Only Vulnerability Description," in *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2017, pp. 125–136.

[5] R. Scandariato, J. Walden, A. Hovsepyan, and W. Joosen, "Predicting vulnerable software components via text mining," *IEEE Transactions on Software Engineering*, vol. 40, pp. 993–1006, 2014.

[6] F. Wu, J. Wang, J. Liu, and W. Wang, "Vulnerability detection with deep learning," in *Proceedings of the 3rd IEEE International Conference on Computer and Communications (ICCC), 2017*, 2017, pp. 1298–1302.

[7] M. Kondo, C.-P. Bezemer, Y. Kamei et al., "The impact of feature reduction techniques on defect prediction models," *Empirical Software Engineering*, pp. 1–39, 2019.

[8] O.Nierstrasz, H. Osman, and M.Ghafari, "Automatic feature selection by regularization to improve bug prediction accuracy," in *IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE)*, 2017, pp. 27–32.

[9] W. Fu, T. Menzies, and X. Shen, "Tuning for software analytics: Is it really necessary?," *Information and Software Technology*, vol. 76, pp. 135–146, 2016.

[10] B. Chen, R. Ramakrishnan, J. W. Shavlik, and P. Tamma, "Bellwether Analysis : Searching for Cost-Effective Query-Defined Predictors in Large Databases," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 3, p. 5, 2009.

[11] R. Krishna, T. Menzies, and W. Fu, "Too much automation? The bellwether effect and its implications for transfer learning," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016, pp. 122–131.

[12] E. F. Combarro, E. Montanes, I. Diaz et al., "Introducing a family of linear measures for feature selection in text categorization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 1223–1232, 2005.

[13] C.-H. Chen, "Feature selection for clustering using instance-based learning by exploring the nearest and farthest neighbors," *Information Sciences*, vol. 318, pp. 14–27, 2015.

[14] L. M. Abualigah, A. T. Khader, and E. S. Hanandeh, "A new feature selection method to improve the document clustering using particle swarm optimization algorithm," *Journal of Computational Science*, 2017.

[15] Y. Pang, X. Xue, and H. Wang, "Predicting Vulnerable Software Components through Deep Neural Network," in *Proceedings of the 2017 International Conference on Deep Learning Technologies*, 2017, pp. 6–10.

[16] C. Lokan and E. Mendes, "Investigating the use of moving windows to improve software effort prediction: a replicated study," *Empirical Software Engineering*, vol. 22, no. 2, pp. 716–767, 2016.

[17] S. Amasaki and C. Lokan, "Towards Better Selection Between Moving Windows and Growing Portfolio," in *Proceedings of the 17th International Conference Product-Focused Software Process Improvement: PROFES*, 2016, vol. 17, pp. 627–630.

[18] J.Walden J.Stuckman and R.Scandariato, "Predicting vulnerable components: Software metrics vs text mining," *Proceedings of the 25th IEEE International Symposium on Software Reliability Engineering (ISSRE)*, pp. 23–33, 2014.

[19] De Vries E, Gilbert J., "Design and implementation of a PHP compiler front-end. Technical report," *Trinity College Dublin, Ireland, Tech. Rep*, 2007.

[20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, pp. 10–18, 2009.

[21] X.Yang, D.Lo, X.Xia, and J. Sun, "TLEL: A two-layer ensemble learning approach for just-in-time defect prediction," *Information and Software Technology*, vol. 87, pp. 206–220, 2017.