International Journal of Latest Trends in Engineering and Technology
Vol.(14)Issue(1), pp.016-022
DOI: http://dx.doi.org/10.21172/1.141.04
e-ISSN:2278-621X

APPLICATION OF BOOSTING, BAGGING AND BLENDING TO PREDICT SEVERITY OF SECURITY DEFECTS

Ankita Bansal[1], Abha Jain[2]

Abstract- In today's era where there are continuous virus-alerts along with a threat of cyber terrorism and malicious crackers, defects which hamper the security are bound to enter the system, thereby exposing the system to security vulnerabilities. Thus, dealing with security- related defects is the necessity which needs to be dealt with utmost caution for failure-free functioning of the software. This would in turn lead to an overall benefit of the organization. But, due to limited resources, it is not possible to pay equal attention to all the security- related defects introduced in the software. Thus, security- related defects need to be assigned with an appropriate severity level which would signify the extent to which that particular defect can be harmful for the system. Assigning severity levels to the security- related defects can prove to be very useful as it would help industry professionals to prioritize defects, thereby allocating available resources and man-power to the defects which have a higher severity. Therefore, in the paper, we mine the defect descriptions using text mining techniques and thereafter develop three models corresponding to each severity level of the defects viz. high, medium and low. Due to various advantages of ensemble methods, we have used three popular ensemble methods in this study; i.e. boosting, bagging and blending. Each model is first validated using J48 decision tree classifier and then J48 is used with the above mentioned ensemble methods. We want to investigate whether the performance of J48 improves after using ensemble methods. The results are validated using open-source Apache software, Tomcat. The results showed that the performance significantly improves in majority of the cases when J48 is used with ensemble methods as compared to the performance of J48 alone.
Keywords – Text mining, Security- related defects, Security vulnerabilities, Defect prediction, Ensemble learning

1. INTRODUCTION

With an ever-increasing expectation and demands of the customers from the software developed, size and complexity of the software is increasing exponentially. Given the large size and complex nature of the software, penetration of defects in the software has become unavoidable and this rate of penetration of defects is increasing day by day. The defects introduced can cause a lot of harm to the software if not handled properly at the right time and with right amount of resources, leading to functional failures [1]. Among all types of defects, the most crucial defects are those which tend to hamper security of the software. Security- related defects are the most critical defects which must be identified and removed from the system as early as possible. These defects expose the system to security vulnerabilities and thereby provide access to exploit confidential privileges of the system like data integrity, availability and confidentiality [2]. They also allow the attackers to gain unauthorized access to user's rights and privileges. Not all the security- related defects which are introduced in the software are of the similar type and nature. In other words, security- related defects vary in terms of their severity levels. A particular severity level is associated with each defect. This severity level determines how harmful a defect can be for the software [3] and accordingly appropriate measures can be taken. A severity level is the most significant aspect of the defect which has a wide range from mild to disastrous. Disastrous security- related defects are the ones which must be handled immediately without causing any delay with the necessary amount of resources in terms of man-power and money [4,5]. In contrast, mild security- related defects are the ones which can be deferred for some time as their presence would not cause much harm to the system.

Keeping in mind the above issue, we intend to assign a severity level to each of security defect found in a system so that the higher severity defects can be paid more attention than the lower severity defects. A series of text mining steps are applied to the defect descriptions in order to extract the relevant information which is in the form of a few set of words. Thereafter, three different models are developed with respect to three severities of defects viz. high (1), medium (2) and low (3) which have been considered in this work.

To validate each of these models, a decision tree method is used known as J48 (uses C4.5 algorithm). J48 is used for constructing a model which can be used to predict the severity of security related defects having low, medium or high levels. However, it is seen that the performance of a single classifier may not be as good as the performance of multiple classifiers combined together to produce the output for the same prediction problem. Combining multiple classifiers or weak learners to produce an output is known as ensemble learning. Ensemble learning can lead to better generalization performance and

[1] Department of Information Technology, Netaji Subhas University of Technology, Delhi, India
[2] Department of Computer Science, Shaheed Rajguru College of Applied Sciences for Women, University of Delhi, Vasundhara Enclave, Delhi, India

reduces the bias and variance of a model [6]. In this study, we have used three popular ensemble learning methods; boosting, bagging and blending (also known as stacking).
We want to investigate whether the prediction performance of J48 improve after using the ensemble methods. Thus, in this study, we aim to test the following set of null and alternate hypothesis:
We will test the following null hypothesis in this study:
H10: The performance of J48 does not improve when it is used with boosting
H1A: The performance of J48 improves when it is used with boosting
H20: The performance of J48 does not improve when it is used with bagging
H2A:The performance of J48 improves when it is used with bagging
H30: The performance of J48 does not improve when it is used with blending
H3A: The performance of J48improves when it is used with blending

The results are validated against open-source Apache software; Tomcat and then performance of the models are compared using accuracy as the performance measure. The results indicated that the performance of J48 has been improved after employing boosting, bagging and blending.
The sections in this paper have the following components: The literature pertaining to the work has been highlighted in section II. This is followed by section III explaining the methodology of the work being carried out. Then section IV elaborates on research background followed by analyzing the results in section V. Finally, the paper is concluded in section VI.

2. LITERATURE REVIEW

Till date, the field of defect prediction has gained popularity and a lot of work has been done in this area. The detection of duplicate reports by incorporating the technique of Natural Language Processing was done by the authors Runeson et al. [7] and Wang et al. [8] through the analysis of the defect reports. An automated method was proposed by Cubranic and Murphy [9] that was based on the analysis of an incoming bug report. This method would help in bug classification so that an appropriate developer suitable for the bug could be chosen who would work on the bug based its description. The handling of new change request in the form of a bug or an enhancement feature was addressed by the authors Canfora and Cerulo [10]. All the above mentioned work talks about defect prediction in general without focusing on the severity of these defects. Till date, very little work has been done in assigning the severity levels to the defects [11, 12, 13]. The authors Menzies et.al [12] have presented an automated method named SEVERIS (SEVERity Issue assessment) which is used to assign severity levels to the defect reports by employing text mining techniques using a rule learning method as their classification method. Similar work has also been done by Sari et al. [13]. The textual descriptions of the defect documents have also been analyzed in the paper by Lamkanfi et al. [11] where the aim was to predict the severity of the defects. The authors had employed Naïve Bayes algorithm as their learning technique and bugs were tracked using Bugzilla. The validation was performed against three open –source projects viz. Mozilla, Eclipse and GNOME. All this work done has considered any type of defect or bug introduced in the software and not necessarily a security defect. Security is the main aspect of the software which must be dealt with seriously for an overall benefit of the organization [2]. Development of software that is secure and reliable has been the main issue of concern for the software developers. Thus, a lot of work has been done to ensure that the requirement engineers are able to elicit the security requirements as clearly as possible during requirement engineering process [14]. The earlier the security requirements are specified, the easier is their incorporation while developing the software.
In this work, we consider the security- related defects introduced in the software and assign severity levels (high, medium, low) to these defects. This can prove to be of great help to the industry professionals as they would be able to focus their resources on the defects that have a higher severity, thus leading to an optimum utilization of the available resources. The ensemble methods are used to construct prediction models. Ensemble learning is widely used in the field of defect prediction. Some of the recent studies using ensemble learning for defect or fault prediction are [6, 15, 16, 17]. However, the use of ensemble learners to predict severity of security related defects is a nascent field and not explored by the researchers till date. This has motivated the authors to carry this work and bring out important inferences.

3. PROPOSED METHODOLOGY

This section discusses the methodology which has been used for predicting severity of security- related defects by mining their textual descriptions. As depicted in Figure 1, there are three steps used to accomplish the work which have been elaborated in the subsequent sub-sections.

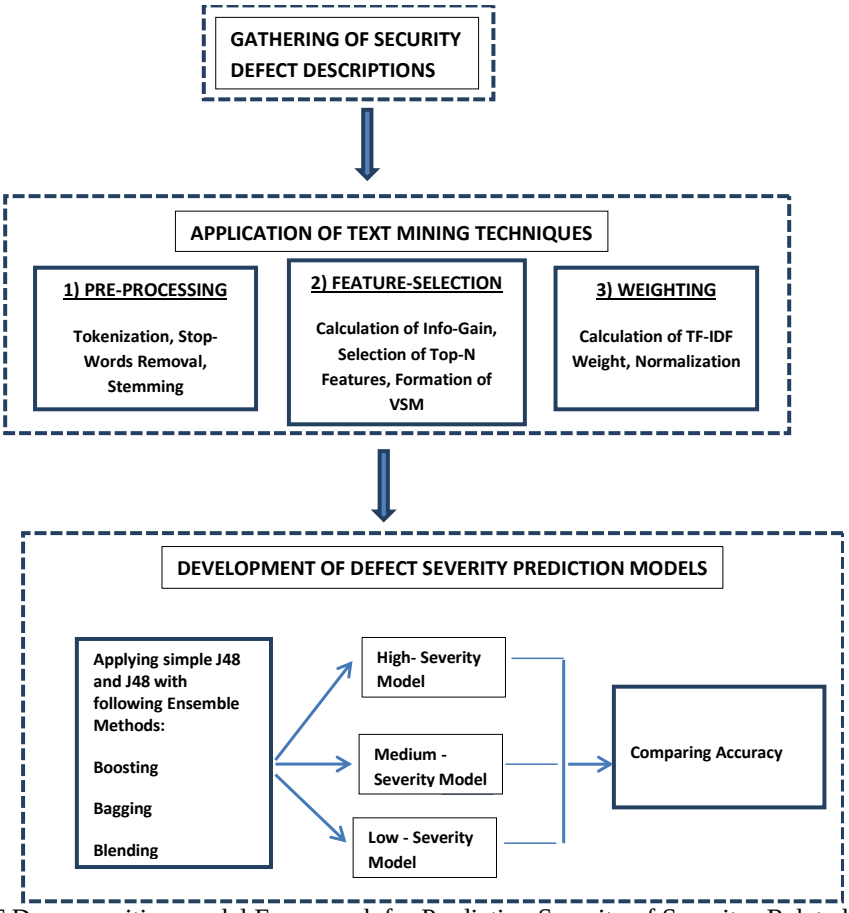

Figure 1 DWT Decomposition model Framework for Predicting Severity of Security- Related Defects

3.1 Gathering of Security Defect Descriptions

The data related to security- related defects has been collected from open-source Apache software, Tomcat. The dataset consists of a total of 178 defect reports wherein each report constitutes of the following components: defect ID, defect description and an associated severity level. In this work, three levels of defect severities are considered viz. high (1), medium (2) and low (3). An entire set of 178 defect descriptions were extracted from the defect reports and thereafter analyzed using text mining steps. The description of each of the text mining steps has been explained in the consequent sub-section.

3.2 Application of Text Mining Steps

In order to develop models which can be used for the prediction of unknown information, we need textual data in the structured format. The defect descriptions retrieved from the defect reports is the data which we need to analyze for further interpretation of unknown information. But the problem here is that this data is in the unstructured format which is difficult to analyze as this kind of data is vague and ambiguous resulting in incorrect prediction. Hence, there arises a need to convert this unstructured data to a structured format for appropriate and accurate prediction of the data. This is accomplished by employing a series of text mining techniques. Figure 1 depicts these techniques beginning from pre-processing, then selection of relevant features, thereafter assigning weights to the selected features using a suitable approach of Tf-Idf and finally normalization. Text mining begins with first applying pre-processing tasks which comprises of tokenization, removal of stop words and finally stemming [18]. Here, the focus is on the removal of words from the data that are not relevant to the application domain and may reduce the prediction capability of the model so constructed. The process of tokenization results into a collection of meaningful tokens by breaking down a stream of textual content into distinct words or terms. After tokenization is performed, different kinds of stop words (prepositions, articles, conjunctions, verbs, pronouns, nouns, adjectives, adverbs) that may appear throughout the textual content are removed from the document. Finally, the words having the same stem are removed and the stem is retained as the selected feature. This is known as the process of stemming. The set of words obtained after applying pre-processing tasks are termed as 'features'. Although, application of pre-processing tasks greatly reduces the size of the feature set, but still size needs to be further reduced. The reduced size of feature set is highly desirable for any learning technique that is used to construct the prediction model. Therefore, we need to apply an appropriate feature selection method that will extract some significant features from the available feature set. In this work, we have used Information-Gain (Info-Gain) measure as the feature selection method. This method selects the top 'N'

features based on their scores obtained after applying the ranking algorithm. This ranking algorithm is applied on all the features obtained after pre-processing. These 'N' selected features form a part of each document. All the documents are then represented in the form of a vector creating a Vector Space Model (VSM). Thereafter, term frequency and inverse document frequency (tf-idf) approach is used to weight each term in the vector. Finally, normalization of vectors is carried out.

3.3 Development of Prediction Models

Here, the task is to develop the prediction models using suitable data analysis techniques that would be used to predict the severity of security- related defects based on their descriptions. In this work, three levels of defect severity have been considered viz. 'High', `Medium' and `Low'. Corresponding to each level of defect severity, a prediction model has been developed. As can be seen from Figure 1, there are in total three prediction models developed specific to each level of defect severity. For instance, prediction model 1 predicts the security-related defects pertaining to `High' severity. Similarly, prediction model 2 and 3 predicts the defects pertaining to `Medium' and `Low' severity respectively. Each of the prediction model is considered as a 'binary model' wherein the value of the dependent variable (severity level) could be either 1 or 0. For instance, prediction model 1 will have the value of the dependent variable as 1 for all the instances that are pertaining to `High' severity level and the value of 0 for all the other remaining instances. Similarly, we construct prediction models 2 and 3. Each of the three prediction models is evaluated using J48 (as a single classifier) and J48 with different ensemble methods viz. boosting, bagging and blending. Finally, the accuracy of all the methods is evaluated and compared.

4. RESEARCH BACKGROUND

In this section, an overview of the research background is provided beginning with an explanation of Info Gain measure used in the work for feature selection. This is followed with an overview of the ensemble learning process and the methods used for it.

4.1 Feature Selection of Relevant Features

The most popularly used method for selecting the relevant features is the Info-Gain measure. This measure is based on the concept of selecting those features from the feature set that simplifies the target concept [12]. The number of bits required to encode an arbitrary class distribution C is H (C). It is given by the following formula:

$$N = \sum_{c \in C} n(c) \quad (1)$$

Where, $n(c) =$ Total number of instances belonging to class c

$$p(c) = {}^{n(c)}/_{N} \quad (2)$$

$$H(C) = -\sum_{c \in C} p(c) \log_2 p(c) \quad (3)$$

Now, suppose A is a group of attributes, then the total number of bits needed to code a class once an attribute has been observed is given by the following formula:

$$H(C|A) = -\sum_{a \in A} p(a) \sum_{c \in C} p(c|a) \log_2(p(c|a)) \quad (4)$$

Now, the attribute which obtains the highest information gain is considered to be the highest ranked attribute which is denoted by the symbol Ai.

$$\text{Infogain}(A_i) = H(C) - H(C|A_i) \quad (5)$$

4.2 Ensemble Learning Techniques

In this work, we have incorporated the usage of ensemble learners for an extensive empirical evaluation of the prediction models. In this section, we explain in detail the use and types of ensemble learning methods used in this study.

The model construction consists of broadly three main steps: 1) training/learning; 2) testing and 3) predicting the unknown variable. There are three important factors which play an important role while training a model. These factors are variance, noise and bias. We understand that majority of the errors which occur during training are because of these factors and hence, they must be reduced as much as possible. This can be done with the help of ensemble methods. Instead of using only a single classifier for prediction, ensemble methods are based on the concept of combining multiple classifiers to produce the output. The diagrammatic representation of the same is shown in figure 2. The process of combining multiple classifiers to solve a computational problem helps to reduce the chances of making a poor or wrong selection. It may or may not improve the performance over a single classifier, but it certainly reduces the risk of poor selection. In addition, the errors which occur during training (bias, variance and noise) are also minimized to a large extent as we avoid any random errors from a single source.

In this study, we have used the following three popular ensemble methods:

4.2.1 Boosting

Boosting [19] starts by running a classifier on the training data. Then, a second classifier is run on the same training data with the aim to emphasis on the samples/instances which were wrongly classified by the previous classifier. This leads to

improved prediction accuracy in each run. For achieving this aim, the weights of the wrongly classified samples are increased and the weights of the correctly classified samples are decreased. This process continues until some threshold is achieved in terms of either the number of models or the desired accuracy. Bosting decreases the model's bias.

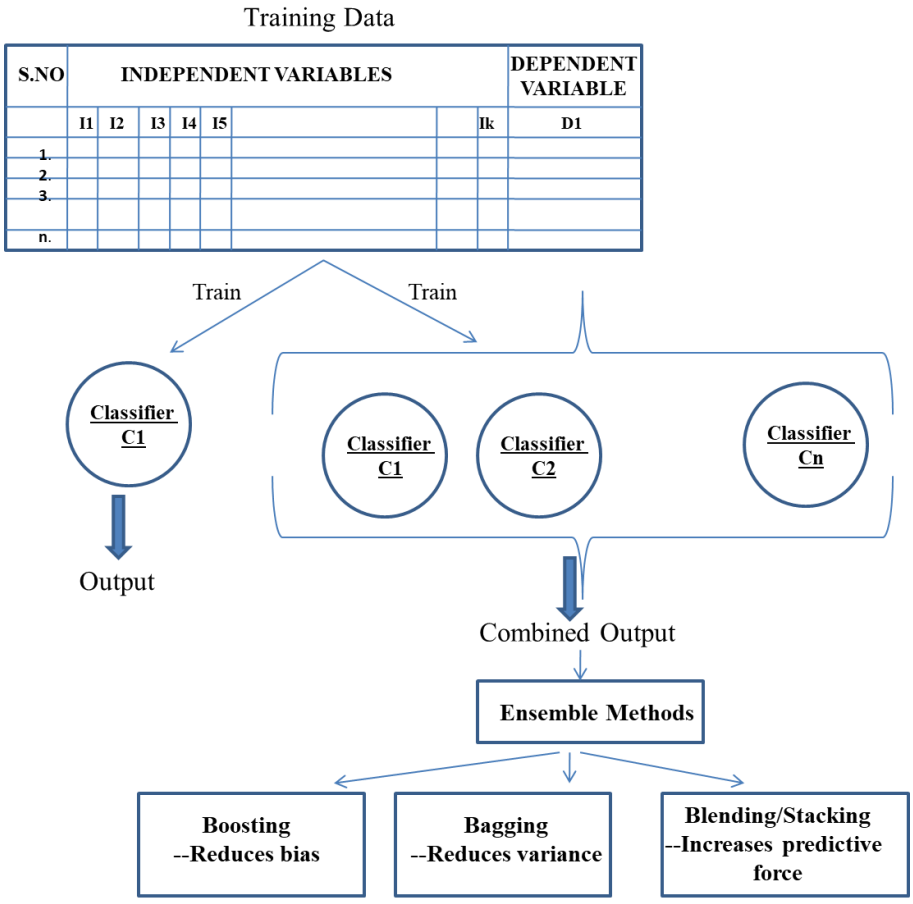

Figure 2 Ensemble Learning Process

4.2.2 Bagging

Bagging [20], also known as Boostrap Aggregating draws different samples with replacement form the training data and builds a classifier on each sample. The output of each classifier is combined by either taking the average or by voting mechanism. Since, each sample of the training data will be different, thus, the performance of each classifier will be different. Bagging decreases the variance of the model.

4.2.3 Blending

Blending involves the use of multiple different classifiers which are trained on the training data. Once all the classifiers are trained, a meta classifier is selected that learns how to take the predictions of each classifier and make accurate predictions on unseen data. Blending is also known as stacking. It increases the predictive force of the classifier.

In this study, we have used J48 decision tree as the base classifier to predict security related defects of all the three models (model 1,2 and 3). After analyzing its performance on all the three models, we have used J48 as the base classifier for boosting, bagging and blending. We want to investigate whether the performance of J48 improve with the ensemble methods. Along with J48, k-nearest neighbor is also used with blending method. The meta-classifier used is logistic regression which is used to combine predictions of J48 and k-nearest neighbor. We have used Logistic regression as it is simple to understand and is widely used for binary classification problems.

5. RESULT ANALYSIS

This section presents the performance of J48 and the performance of J48 when used with different ensemble methods; boosting, bagging and blending. Thus, in total we have four classifiers which are applied on each of the three models viz. 'High', 'Medium' and 'Low' severity models. The performance of the models is depicted in terms of accuracy.

It is advisable not to use the same data for training as well as testing. Thus, the data is partitioned into two sets; one of which is used for training and the other is used for testing. This is done using a validation technique. The validation technique which

has been used in this work is k-fold cross validation. This technique is based on partitioning the complete dataset into k parts which are of equal size. Out of these k parts, one part validates the data and rest of the k-1 parts train the data [21]. This process of validation is repeated k times to ensure the validation of each data point once. A single estimation is then obtained by combining these k results. The k value is considered as 10 in this work.

Table -1 Experiment Result

Models	Base Classifier	Ensemble Methods		
	J48	J48 with boosting	J48 with bagging	J48 with blending
Model 1	85.71	90.80 (v)	87.94	92.47 (v)
Model 2	87.98	94.21 (v)	89.21	92.92 (v)
Model 3	84.44	89.27	85.16	91.63 (v)

The table 1 shows the performance of the base classifier, i.e. J48 when it is used independently without any ensemble method and its performance when it is used with boosting, bagging and blending. We can observe from the table that for model 1 (high severity), the accuracy of J48 is 85.71%. When compared with the other three ensemble techniques, we can observe that the accuracy of all of them is higher than simple J48. The symbol (v) along with the accuracy of boosting and blending denotes that the accuracy is significantly different and is higher than the base classifier (J48). Blending has shown the highest accuracy of 92.47%. We can see that there is no symbol with bagging, showing that the result is not significantly different than J48. Thus, to predict high severity defects, boosting and blending can be used with J48.

Next, we discuss the performance of the classifiers on model 2 (medium severity). We can again infer the similar observations. J48 when used without any ensemble method has shown the lowest accuracy. Whereas, boosting and blending have shown higher accuracies than J48. The results are also significantly different than J48 as denoted by symbol (v). Boosting has depicted the highest accuracy of 94.21%. Although bagging has also shown higher accuracy than J48, but the result is not significantly different. Thus, for predicting medium severity defects, boosting and blending can be used.

Finally, we discuss the performance of classifiers on model 3, i.e. for predicting low severity defects. We can observe that the performance of J48 is lowest (84.44%). When blending is used, highest accuracy of 91.63% is achieved. The result is also significantly different than J48. Although higher accuracy is achieved when boosting and bagging are used than when J48 is used, but the result is not significantly different.

Overall, we can conclude that the performance of J48 improves after using the ensemble methods; boosting, bagging and blending. Hence, we reject the null hypothesis, H10, H20 and H30 and accept the corresponding alternate hypothesis. For all the three models, the significant improvement is seen when J48 is used with blending, whereas boosting also shows significant improvement for model 1 and 2. Although higher values of accuracy are achieved when bagging is used, but the results are not significantly different. Thus, we suggest the use of ensemble methods for obtaining improved prediction accuracy.

6. CONCLUSION

Assigning correct severity level to the security- related defects is the need of today's organization where there are limited resources and therefore, paying equal amount of attention to all the type of security- related defects introduced in the software is not possible. As a result, the work in this paper concerns with developing models corresponding to three levels of defect severities viz. high, medium and low with the intent to address the high severity defects on a priority basis. This would result in failure-free functioning of the software, thus leading to an overall benefit of an organization. For developing the models at high, medium and low severity levels, three popular ensemble learning methods have been used. These three methods are boosting, bagging and blending. Instead of using only a single classifier to produce the output, ensemble learning methods combine multiple classifiers which leads to low bias, low variance and high predictive performance. We have used J48 as the base classifier for boosting and bagging. For blending, two classifiers are used; i.e. K-nearest neighbor and J48.The results indicate improved accuracy performance of J48 when the ensemble methods are used. The performance of all the ensemble methods, boosting, bagging and blending is improved as compared to the performance of single classifier, J48. However, the performance of bagging is not significantly different from J48. Thus, we recommend the use of the ensemble methods to predict severity levels of security related defects.

As future work, we plan to implement more ensemble methods and perform an exhaustive comparison. In addition, instead of applying a machine learning classifier as the base classifier, we would try meta-heuristic algorithms as well. Heterogeneous ensemble learning can also be implemented and compared with homogenous ensemble learning.

7. REFERENCES

[1] Myers,G., Badgett, T., Thomas, T. and Sandler, C., "The Art of Software Testing", Second ed. John Wiley & Sons, Inc., Hoboken, NJ, 2004.
[2] Amoroso, E. G., "Fundamentals of Computer Security Technology", Prentice-Hall, ISBN: 0-13-305541-8, 1994.
[3] Singh, Y., Kaur, A. and Malhotra, R., "Empirical validation of object-oriented metrics for predicting fault proneness models", Software Quality Journal, 18: 3-35, 2010.
[4] Emam, K.E. and Melo, W., "The Prediction of Faulty Classes Using Object-Oriented Design Metrics", Technical report: NRC 43609, 1999.

[5] Aggarwal, K.K., Singh, Y., Kaur, A. and Malhotra, R., "Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: A replicated case study", Software Process: Improvement and Practice, 16(1), 39-62, 2009.
[6] Rodrigo A. Coelho, Fabrício dos R.N. Guimarães and Ahmed A.A. Esmin, "Applying Swarm Ensemble Clustering Technique for Fault Prediction Using Software Metrics", 13th International Conference on Machine Learning and Applications, 2014.
[7] Runeson, P., Alexandersson, M. and Nyholm, O., "Detection of Duplicate Defect Reports Using Natural Language Processing", 29th IEEE International Conference on Software Engineering (ICSE), 499 – 508, 2007.
[8] Wang, X., Zhang, L., Xie, T., Anvik, J. and Sun, J., "An Approach to Detecting Duplicate Bug Reports using Natural Language and Execution Information", Association for Computing Machinery, 2008.
[9] Cubranic, D. and Murphy, G.C., "Automatic bug triage using text categorization", Proceedings of the Sixteenth International Conference on Software Engineering and Knowledge Engineering, 2004.
[10] Canfora, G. and Cerulo, L., "How Software Repositories can Help in Resolving a New Change Request", Workshop on Empirical Studies in Reverse Engineering, 2005.
[11] Lamkanfi, A., Serge, D., Giger, E. and Goethals, B., "Predicting the Severity of a Reported Bug", 7th IEEE working conference on Mining Software Repositories (MSR), PP. 1-10, 2010.
[12] Menzies, T. and Marcus, A., "Automated Severity Assessment of Software Defect Reports", IEEE International Conference on Software Maintenance (ICSM), 2008.
[13] Sari, G. I. P. and Siahaan, D. O., "An attribute Selection For Severity level Determination According To The Support Vector Machine Classification Result", Proceedings of The 1st International Conference on Information Systems For Business Competitiveness (ICISBC), 2011.
[14] Firesmith, D.G., "Engineering Security Requirements", Journal of Object Technology. Vol. 2, No. 1, pages 53-68, 2003.
[15] Yohannese , C.W., Li , T., Simfukwe , M. and Khurshid, F., "Ensembles based combined learning for improved software fault prediction: A comparative study", 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), 2017.
[16] Li , Z., Jing , X.Y. , Zhu , X. and Zhang, H., "Heterogeneous Defect Prediction Through Multiple Kernel Learning and Ensemble Learning", IEEE International Conference on Software Maintenance and Evolution (ICSME), 2017.
[17] Young , S., Abdou , T., Bener, A., "A Replication Study: Just-in-Time Defect Prediction with Ensemble Learning", IEEE/ACM 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2018.
[18] Sebastiani, F., "Machine Learning in Automated Text Categorization", ACM Computing Surveys, Vol. 34, No.1, 2002.
[19] Freund, Y., and Schapire, R. A short introduction to boosting. Journal of Japanese Society for Artificial Intelligence 14, 5 (1999), pp. 771-780.
[20] L. Breiman, Bagging predictors, Machine Learning, vol. 24, no. 2, pp. 123–140, 1996.
[21] Stone, M., "Cross-validatory choice and assessment of statistical predictions", Journal Royal Stat. Soc., pp.111-147, 1974.