International Journal of Rough Sets and Data Analysis
Volume 6 • Issue 2 • April-June 2019

Weighted SVMBoost based Hybrid Rule Extraction Methods for Software Defect Prediction

Jhansi Lakshmi Potharlanka, Vignan's Foundation for Science Technology and Research (Deemed to be) University, Vadlamudi, India

Maruthi Padmaja Turumella, Vignan's Foundation for Science Technology and Research (Deemed to be) University, Guntur, India.

ABSTRACT

The software testing efforts and costs are mitigated by appropriate automatic defect prediction models. So far, many automatic software defect prediction (SDP) models were developed using machine learning methods. However, it is difficult for the end users to comprehend the knowledge extracted from these models. Further, the SDP data is of unbalanced in nature, which hampers the model performance. To address these problems, this paper presents a hybrid weighted SVMBoost-based rule extraction model such as WSVMBoost and Decision Tree, WSVMBoost and Ripper, and WSVMBoost and Bayesian Network for SDP problems. The extraction of the rules from the opaque SVMBoost is carried out in two phases: (i) knowledge extraction, (ii) rule extraction. The experimental results on four NASA MDP datasets have shown that the WSVMBoost and Decision tree hybrid yielded better performance than the other hybrids and WSVM.

KEYWORDS

Bayes Network, Boosting, J4.8, Ripper, Software Defect Prediction, WSVM

INTRODUCTION

Most of the IT organizations, before the delivery of a product, they are not very sure about the quality of the product. The product could be high quality or low quality. Software testing efforts and costs are subsided by using effective software code defect, prediction methods. Usually, the software defect prediction models identify the bug-prone software artifacts in software projects so that the quality assurance team can allow limited resources in an effective way for testing the software projects (Nam, 2014).

In order to develop effective defect prediction models machine learning techniques are devised prominently used besides statistical methods. Along with tradition machine learning methods like Navie Bayes, Decision Trees, Linear and Non-Linear Regression advanced methods like group method of data handling, Support Vector Machines were also adopted to device effective defect prediction models.

However, the class imbalance nature of the Software Defect Prediction (SDP) data is a critical issue in developing effective models, where the machine learning algorithms fail to exhibit better predictions from underrepresented defect class (Wang & Yao, 2013). Many solutions are proposed to develop effective SDP models using conventional classification algorithms and the methods

DOI: 10.4018/IJRSDA.2019040104

that address the class imbalance problem. However, the knowledge extracted from these models is not much able to comprehend by the naive users. From the explorative studies of class imbalance problem, it is identified that (Japkowicz & Stephen, 2002) the Support Vector Machines (SVM) is less sensitive to class imbalance problem and Boosting of Weighted Support Vector Machines (WSVM) is used to improve the model performance further. Addition to this, the SDP models build with SVMs are exhibited good performances compared with other algorithms (Gray, Bowes, Davey, Sun, & Christianson, 2009). Usually, the SVM identifies the decision boundary in the form of separating hyperplane $y = W\,x + b$ among the classes, where W is the width and b is the bias of the hyperplane. However, the naive user seldom understands the extracted knowledge in the form of mathematical models.

To develop efficient SDP model, which can improve defect prediction rate by countering class imbalance problem as well as having explanation ability in the form of If-then rules, this paper proposes three hybrid rule extraction models using WSVMBoost. The hybrids include WSVMBoost and Decision Tree (DT), WSVMBoost and RIPPER, WSVMBoost and Bayesian Network (BN). Due to their ability to represent the learned knowledge in the form of if then else rule Decision Tree (DT), RIPPER and Bayesian Network (BN) are used for extracting rules form learned WSVMBoost boundary. The performance of the proposed approach is validated using G-Mean and F-Measure performance measures.

The paper is organized into five sections. The related work is presented in Section 2. Section 3 depicts the proposed hybrid methods. The discussion of obtained results is depicted in section 4. Finally, this paper concludes in Section 5.

RELATED WORK

Researchers used different analysis techniques ranging from Statistics to Machine Learning (Nam, 2014) (Kamei & Shihab, 2016) for effective prediction models. Recently, Li et al., categorized the recent SDP efforts in to machine learning-based prediction algorithms, methods to manipulating the data and mechanisms for effort-aware prediction (Li, Jing, & Zhu, 2018). Nagappan and Ball applied (Nagappan & Ball, 2005) PREfast and PREfix statistical analysis tools for predicting defects and reported 82.91% accuracy of the model. From the studies, which adopted machine learning techniques, Naive Bayes (Menzies, Greenwald, & Frank, 2007) reported 71% accuracy, a Bayesian network of Metrics and Defect Proneness (Okutan et al., 2014) reported 72.5% average accuracy. The Support Vector Machines (Gray et al., 2009) as base learners achieved 80% accuracy. From the combined models of Support Vector Machines (SVM) and Probabilistic Neural Network (PNN) (Al-Jamimi & Ghouti, 2011) reported 87.62% accuracy. Neural Network (NN), Decision Tree (DT), PART, Logistic Regression (LR) and Ada Boost (Arisholm, Briand, & Johannessen, 2010) and achieved 75.6% average Accuracy.

There are works that applied both traditional machine learning methods and the methods that address the class imbalance problem. Wang et al., explores the significance of considering class imbalance problem in SDP data to improve the defect prediction rate (Wang, & Yao 2013)). An extensive systematic study with seven base classifiers, seventeen imbalanced learning methods on twenty-seven SDP datasets concluded that, adopting class imbalance methods is beneficial when the imbalance rate is either high or moderate. That class imbalance method and corresponding base classifier chosen carefully (Song, Guo, & Shepperd, 2018). The methods that are tailored to SDP such as Coding based Multi-Classifier Modelling (Sun, Song, & Zhu, 2012) had accomplished with 84% accuracy. A study (Lessmann, Baesens, Mues, & Pietsch, 2008) in comparison with 22 different classification algorithms over 10 NASA MDP data sets reported that Random Forest yielded the best mean rank in terms of performance. Recently, Weighted Least Square Support Vector Machine studied on SDP problem and yielded 77% of F-measure.

In few other studies feature ranking methods such as Correlation Ranking (CR), Information Gain (IG), ensemble of feature ranking methods (Chandrashekar & Sahin, 2014; Gao, Khoshgoftaar, & Napolitano, 2015; Xu, Liu, Yang, An, & Jia, 2016) are applied on several base and ensemble learners and concluded that feature ranking with ensembles improved the defect prediction rate significantly. Except for the DT based models, the other SDP models are poor in explanation ability. However, in this case, it is difficult for the test architect to comprehend the knowledge as the models are of opaque due to their rich mathematical modeling. Thus, limits the decision of optimal resource allocation for software testing.

There are research efforts in extracting explainable knowledge from opaque models such as Support Vector Machines (SVM) and Neural Networks (NN) (Barakat & Bradley, 2010; Zięba, Tomczak, Lubicz, & Świątek, 2014). These methods work on the principle of extracting the knowledge, re-labeling the training data, which includes inducing the rules on the new training data. Though, there are prominent rule extraction methods to extract meaningful rules from NN, as a base classifier SVM has been exhibited better generalization abilities on class imbalance problem compared with other base classifiers (Zughrat, Mahfouf, Yang, & Thornton, 2014) (Yuchun Tang, Yan-Qing Zhang, Chawla, & Krasser, 2009). Therefore, this paper focuses on developing effective SDP models by extracting rules from WSVMBoost using rule extraction-based learning algorithms.

BACKGROUND

Weighted SVM

A Weighted Support Vector Machine classifier (WSVM Boost) (Benjamin & Japkowicz, 2010) is used to cope with the class imbalance problem. It boosts the outlier subtlety problem of support vector machine (SVM) for class imbalance problem. The elemental intention is to empower different weights to divergent points such that the WSVM training algorithm determines the decision surface corresponding to the relative importance of data points in the training data set. In WSVM the weights are provoked by kernel – based possibilistic c – means (KPCM) algorithm. The objective function in equation (1) for WSVM includes two cost parameters, C^+ and C^- to perceive high priority for defect class usually which is equal to the imbalance ratio of the dataset.

$$\min_{(W,,b,\xi)} (1/2) \left\| W \right\|^2 + C^+ \Sigma^{N+}{}_{i \mid yi=+1} \xi_i + C^- \Sigma^{N-}{}_{i \mid yj=-1} \xi_j \quad (1)$$

$$\text{for all k: } [Wx_k + b] \geq (1 - \xi_k), \text{ where } \xi_k = \max[0, 1\text{-}y_k(w_k x_k + b)] \quad (2)$$

Here, N^+ and N^- are the number of defects and normal class samples respectively.

Rule Extraction Methods

DT (J 4.8):

This algorithm in (Quinlan, 1986) exploits a greedy strategy to build up a tree structure of training datasets to classify the test dataset. The vital perception behind decision tree is as follows: Opening from the training data, we desire to construct a predictive model, which is generalized to the tree structure. The objective is to attain a splendid classification with the nominal figure of decisions. The leaf nodes of the decision tree serve as predicted variable or decision whereas the non-leaf nodes serve as attributes. The key algorithm for decision tree is: Opening from the perfect training data, choose an attribute, so that gives the best split, build child nodes based on the best split, for every iteration the best attribute to split is selected by using Information Gain or Gini Index or Hellinger distance. The construction of the tree abolished when all the attributes are doused to split further.

Formerly the tree is build-up in this fad it may over fits for the given data. By practicing pruning, one can obtain optimal rules. In this work, we have recycled J4.8 algorithm for our experiments. J4.8 frames a multi-way tree using Information Gain as the attribute selection criteria for splitting the data.

Ripper

Repeated Incremental Pruning to Produce Error Reduction (RIPPER) is a direct method in building the classification rules; it extracts rules directly from the whole data. It is positioned on association rules with Reduced Error Pruning (REP). This algorithm is designed in different stages: Growth stage, Pruning stage, Optimization stage, and Selection stage. In the growth stage, a rule is grown by greedily adding attributes to the rule until the rule confirmed the benchmark. In the prune stage, every rule is pruned incrementally by granting the pruning of the final arrangement of the attributes, until the criteria meet. In the optimization phase, all the developed rules are further optimized by i) greedily joining the attributes to the initial rule ii) by individually developing a new rule. In the selection stage, the most elegant rules are considered, and the remaining rules are discarded from the model.

Bayes Network

Bayes Network (Okutan et al., 2014) is a graphical representation used to resolve the probabilistic authoritative relationships between software metrics and defects. It is a DAG (Directed Acyclic Graph) consists of E edges and V vertices, which serve as the joint probability distribution of a set of variables. By this, each vertex serves as a variable, and every edge serves as a causal impact of one variable to its successor in the network. When Bayes Network is used in affiliation with statistical techniques, the graphical model has handful preferences for data modeling. Bayes Network can promptly handle fragmentary data sets. It is a graphical model that comfortably encodes the probability distribution.

Performance Measures

To measure the efficiency of the developed Software Defect Prediction models (Fawcett, 2006) G-Mean and F-Measure are used as performance metrics.

G – Mean: It is the geometric mean of positive class accuracy and negative class accuracy.

$$\text{G-mean} = \sqrt{(\text{accuracy of positive class} * \text{accuracy of negative class})} \quad (3)$$

F – Measure: A measure that associates with precision and recall, it is the harmonic mean of precision and recall, the universal F – Measure is:

$$\text{F-Measure} = (\beta * \text{precision} * \text{recall}) / (\text{Precision} + \text{recall}) \quad (4)$$

Where, *precision* defines the trade-off between *True Positive* and *False Positive* rates whereas *recall* defines the *True Positive* rate and β is a user-defined parameter. Usually, its value is considered as 2.

Friedman's Ranking Test

It is a nonparametric statistical test, used to identify the differences between the distributions caused by different test attempts. Initially, each of the distributions ranked between 1 to n, where n is the largest rank among the distribution (Demšar, 2006). Here Friedman's ranking is used to identify the method with best average mean ranking compared with other methods once their performance is ranked on all datasets. To apply the Friedman's ranking test the data should meet some requests, like data must be unique; the sample stood fashioned with a random sampling method.

PROPOSED HYBRID WSVM BOOST BASED RULE EXTRACTION METHODS

Proposed hybrid rule extraction methods learn in two phases.

Prediction Phase

In this phase, the weak weighted SVM (Benjamin & Japkowicz, 2010) is boosted for t = 1 to 'n' times, where 'n' is the user's input. In every iteration, the weights of miss-classified instances are increased such that, they can be classified in next iterations. The final target prediction of each sample is the sign of the sum of the predictions of all n weak learners. Once after predicting the class labels of the whole dataset, the actual targets are replaced with the new predicted class labels and referred as *P*.

Rule Extraction Phase

In this phase rule, extraction algorithms such as decision tree, RIPPER, Bayesian Network are learned on predicted dataset *P* obtained on previous Prediction Phase. Consequently, three hybrid rule extraction methods WSVMBoost and Decision Tree, WSVMBoost and RIPPER, WSVMBoost and Bayesian Network with the final predictions in the form of if-then rules resulted. Figure 1 depicts a flow diagram for proposed hybrids with two phases.

DATASET DESCRIPTION AND EXPERIMENTAL SETUP

Dataset Description

Proposed hybrids are validated on four defect prediction datasets from NASA MDP repository (Sayyad Shirabad & Menzies, 2005). These datasets are having 21 method level metrics, including 1 class label. The statistical details of the considered datasets are presented in Table 1. Here D refers to Defect class and ND refers to Non Defect class.

Experimental Setup

Different classifiers Decision Tree (J4.8), RIPPER, Bayes Network are considered from Weka Software (Witten, Frank, & Eibe, 2005). The base learner WSVM is considered from LibSVM (Chang & Lin, 2011) and WSVMBoost is implemented on MATLAB MEX interface. The results presented on each dataset are the average of the G-mean and F-measure performances of the ten - fold cross-validation. In cross-validation, each sample in cross-validation is at least trained once and tested once so that effective generalization is taken place without losing either modeling or testing capabilities.

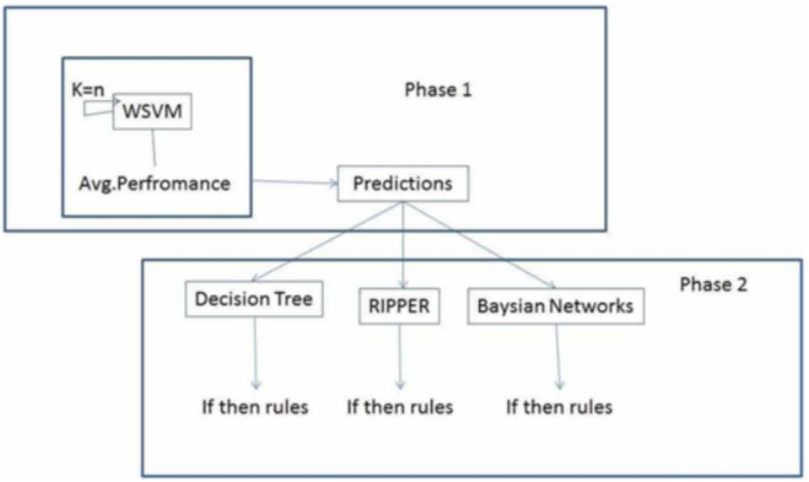
RESULTS AND DISCUSSION

Presented hybrids are compared with the base WSVM to validate the expressive capabilities of the 'if-then' rules. Table 2 presents the performance of the considered hybrids and base WSVM classifier

Table 1. Dataset description

S. No.	Dataset	No. Of Attributes	No. of Instances	D:ND	Imbalanced Rate
1	CM1	22	498	34:315	9.16
2	KC1	22	2109	326:1783	5.46
3	KC2	22	522	107:415	3.87
4	PC1	22	1109	77:1032	13.4

Figure 1. Flow diagram for proposed hybrid approaches



in terms of both Defective(D) and Non-Defective class G-Mean. From Table 2 it is observed that the base Weighted Support Vector Machine (WSVM) classifier resulted in poor performance (<80%) compared with all hybrid rule extraction methods. Among the hybrid rule extraction methods, it is observed that, the hybrid of WSVMBoost and Decision Tree exhibited better performance over all data sets. Whereas the hybrid of WSVMBoost and Ripper WSVMBoost and Bayesian Network resulted with the second-best performance for KC1 and PC1 data sets. On the other hand, WSVMBoost and Bayesian Network yeilded second best performance with CM1 and KC2 datasets. To identify the winner among the presented hybrid rule extraction methods and base WSVM classifier, the G-Mean, and F-Measure performance measures are ranked using Friedman's Ranking (Demšar, 2006). The Friedman's Ranking is a statistical method to identify the differences among the experiments by ranking each attempt (Table 3). Finally, the winner is identified with a mean ranking of G-Mean and F-Measure performances of the considered methods. From Table 3 it can be identified that the hybrid of WSVMBoost and Decision Tree (J4.8) yielded better Friedman's mean rank over both F – Measure: 4.0 and G – Mean: 4.0. The hybrid of WSVMBoost and Ripper, WSVMBoost and Bayesian Network yielded next best mean rank. Obtained rules of KC2 for each of the hybrids are presented in Rules 1, 2, 3.

Rule 1: WSVM Boost +Decision Tree

```
v <= 0.018584
|   b <= 0.011364
|   |   loc <= 0.059028
|   |   |   total_Op <= 0.022124: ND (1190.0/1.0)
|   |   |   total_Op > 0.022124
|   |   |   |   ev(g) <= 0.053846
|   |   |   |   |   uniq_Opnd <= 0.033333: D (2.0)
|   |   |   |   |   uniq_Opnd > 0.033333: ND (35.0/1.0)
|   |   |   |   ev(g) > 0.053846: D (2.0)
|   |   loc > 0.059028
|   |   |   uniq_Op <= 0.027027: ND (20.0)
|   |   |   uniq_Op > 0.027027: D (12.0/3.0)
|   b > 0.011364
|   |   uniq_Opnd <= 0.083333
```

Table 2. F-Measure and G-Mean performance, superscript represents the Friedman's ranking

Data Set	WSVM		WSVMBoost + BN			WSVMBoost + RIPPER			WSVMBoost + DT		
	F-Measure	G-Mean	F-Measure		G-Mean	F-Measure		G-Mean	F-Measure		G-Mean
			D	ND		D	ND		D	ND	
CM1	0.7026[1]	0.7421[1]	0.716	0.87	0.848[3]	0.746	0.91	0.837[2]	0.811	0.939	**0.868**[4]
KC1	0.7309[1]	0.7274[1]	0.96	0.976	0.969[2]	0.976	0.986	0.981[3]	0.978	0.987	**0.982**[4]
KC2	0.6777[1]	0.7761[1]	0.939	0.968	0.958[3]	0.937	0.968	0.951[2]	0.951	0.976	**0.96**[4]
PC1	0.7912[1]	0.7780[1]	0.627	0.791	0.753[2]	0.777	0.914	0.843[3]	0.796	0.919	**0.859**[4]

```
|   |   |   ev(g) <= 0.038462
|   |   |   |   lOCode <= 0.049618
|   |   |   |   |   uniq_Opnd <= 0.075
|   |   |   |   |   |   i <= 0.082617
|   |   |   |   |   |   |   v(g) <= 0.044444
|   |   |   |   |   |   |   |   i <= 0.048482: D (2.0)
|   |   |   |   |   |   |   |   i > 0.048482: ND (5.0/1.0)
|   |   |   |   |   |   |   v(g) > 0.044444: D (6.0)
|   |   |   |   |   |   i > 0.082617: ND (67.0/2.0)
|   |   |   |   |   uniq_Opnd > 0.075
|   |   |   |   |   |   loc <= 0.045139: ND (6.0/1.0)
|   |   |   |   |   |   loc > 0.045139: D (9.0)
|   |   |   |   lOCode > 0.049618
|   |   |   |   |   lOBlank <= 0
|   |   |   |   |   |   e <= 0.003127: ND (2.0)
|   |   |   |   |   |   e > 0.003127: D (3.0)
|   |   |   |   |   lOBlank > 0: D (9.0)
|   |   |   ev(g) > 0.038462: D (10.0)
|   |   uniq_Opnd > 0.083333: D (17.0)
v > 0.018584: D (712.0/2.0)
Number of Leaves :          18
Size of the tree:          35
```

Rule 2: WSVM Boost + Ripper

```
(v >= 0.018776) and (locCodeAndComment <= 0.083333) => defects=D
(667.0/0.0)
(v >= 0.013638) and (uniq_Opnd >= 0.083333) => defects=D
(74.0/5.0)
(v >= 0.01084) and (e >= 0.002577) and (i <= 0.084948) and (v(g)
>= 0.066667) => defects=D (19.0/0.0)
(total_Opnd >= 0.021028) and (loc >= 0.059028) and
(locCodeAndComment <= 0.083333) and (n >= 0.027125) => defects=D
(14.0/0.0)
(lOComment >= 0.045455) and (loc >= 0.065972) => defects=D
(7.0/1.0)
(b >= 0.011364) and (i <= 0.046359) => defects=D (4.0/0.0)
(i >= 0.133378) and (loc >= 0.052083) and (loc >= 0.0625) =>
defects=D (3.0/0.0)
```

Table 3. Friedman's mean ranking

	WSVM	WSVMBoost + BN	WSVMBoost + RIPPER	WSVMBoost + DT
F-Measure	1	2.25	2.75	4
G-Mean	1	2.5	2.5	4

```
 => defects=ND (1321.0/3.0)
Number of Rules: 8
```

Rule 3: WSVM Boost + Bayesian Network

```
LogScore Bayes: -30601.284763589785
LogScore BDeu: -30908.326530872
LogScore MDL: -31003.013745034103
LogScore ENTROPY: -30417.48510272746
LogScore AIC: -30570.48510272746
```

CONCLUSION

This paper proposes three hybrid rule extraction methods such as WSVMBoost and Decision Tree, WSVMBoost and RIPPER and WSVMBoost and Bayesian Network is proposed to improve the defect prediction rate as well to improve the explanation ability of the model to the end user. From the experimental results on four datasets, it can be identified that proposed hybrids yielded better performance than the base Weighted Support Vector Machine (WSVM) and the extracted knowledge in the form of if then else rules can also be well comprehended by the end users.

FUTURE WORK

In future we are planning to compare the proposed hybrids with other WSVMBoost hybrids, such as fuzzy rule learning methods, to identify the best model that better suits the SDP problem. Both cross-validation and hold out method with validation set are planned to be used for performance evaluation.

REFERENCES

Al-Jamimi, H. A., & Ghouti, L. (2011). Efficient prediction of software fault proneness modules using support vector machines and probabilistic neural networks. In *Proceedings of the 2011 Malaysian Conference in Software Engineering* (pp. 251–256). Academic Press. doi:10.1109/MySEC.2011.6140679

Arisholm, E., Briand, L. C., & Johannessen, E. B. (2010). A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. *Journal of Systems and Software*, *83*(1), 2–17. doi:10.1016/j.jss.2009.06.055

Barakat, N., & Bradley, A. P. (2010). Rule extraction from support vector machines: A review. *Neurocomputing*, *74*(1–3), 178–190. doi:10.1016/j.neucom.2010.02.016

Benjamin, X. W., & Japkowicz, N. (2010). Boosting support vector machines for imbalanced data sets. *Knowledge and Information Systems*, *25*(1), 1–20. doi:10.1007/s10115-009-0198-y

Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, *40*(1), 16–28. doi:10.1016/j.compeleceng.2013.11.024

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol. Article*, *2*(27). doi:10.1145/1961189.1961199

Demšar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, *7*, 1–30. doi:10.1016/j.jecp.2010.03.005

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. doi:10.1016/j.patrec.2005.10.010

Gao, K., Khoshgoftaar, T. M., & Napolitano, A. (2015). Investigating Two Approaches for Adding Feature Ranking to Sampled Ensemble Learning for Software Quality Estimation. *International Journal of Software Engineering and Knowledge Engineering*, *25*(1), 115–146. doi:10.1142/S0218194015400069

Gray, D., Bowes, D., Davey, N., Sun, Y., & Christianson, B. (2009). Using the Support Vector Machine as a Classification Method for Software Defect Prediction with Static Code Metrics. In *Engineering Applications of Neural Networks* (pp. 223–234). Springer. doi:10.1007/978-3-642-03969-0_21

Japkowicz, N., & Stephen, S. (2002). The Class Imbalance Problem: A Systematic Study. *Intelligent Data Analysis*, *6*(5), 429–449. doi:10.3233/IDA-2002-6504

Kamei, Y., & Shihab, E. (2016). Defect Prediction: Accomplishments and Future Challenges. In *Proceedings of the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)* (Vol. 1, pp. 33–45). IEEE. doi:10.1109/SANER.2016.56

Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, *34*(4), 485–496. doi:10.1109/TSE.2008.35

Li, Z., Jing, X., & Zhu, X. (2018). Progress on approaches to software defect prediction. *IET Software*, *12*(3), 161–175. doi:10.1049/iet-sen.2017.0148

Menzies, T., Greenwald, J., & Frank, A. (2007). Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering*, *33*(9), 635–636. doi:10.1109/TSE.2007.70721

Nagappan, N., & Ball, T. (2005). Static analysis tools as early indicators of pre-release defect density. In *Proceedings. 27th International Conference on Software Engineering ICSE 2005* (pp. 580–586). Academic Press. doi:10.1109/ICSE.2005.1553604

Nam, J. (2014). *Survey on Software Defect Prediction* [Master's Thesis]. Retrieved from http://www.cse.ust.hk/~jcnam/files/PQE_Survey_JC.pdf

Okutan, A., & Yildiz, O. T. (2014). Software defect prediction using Bayesian networks. *Empirical Software Engineering*, *19*(1), 154–181. doi:10.1007/s10664-012-9218-8

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, *1*(1), 81–106. doi:10.1007/BF00116251

Sayyad Shirabad, J., & Menzies, T. J. (2005). PROMISE Software Engineering Repository. Retrieved from http://promise.site.uottawa.ca/SERepository/

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

Sun, Z., Song, Q., & Zhu, X. (2012). Using coding-based ensemble learning to improve software defect prediction. *IEEE Transactions on Systems, Man and Cybernetics. Part C, Applications and Reviews*, *42*(6), 1806–1817. doi:10.1109/TSMCC.2012.2226152

Wang, S., & Yao, X. (2013). Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability*, *62*(2), 434–443. doi:10.1109/TR.2013.2259203

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Xu, Z., Liu, J., Yang, Z., An, G., & Jia, X. (2016). The Impact of Feature Selection on Defect Prediction Performance: An Empirical Comparison. In *Proceedings of the 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)* (pp. 309–320). IEEE. doi:10.1109/ISSRE.2016.13

Tang, Y., Zhang, Y.-Q., Chawla, N. V., & Krasser, S. (2009). SVMs Modeling for Highly Imbalanced Classification. *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics*, *39*(1), 281–288. doi:10.1109/TSMCB.2008.2002909 PMID:19068445

Zięba, M., Tomczak, J. M., Lubicz, M., & Świątek, J. (2014). Boosted SVM for extracting rules from imbalanced data in application to prediction of the post-operative life expectancy in the lung cancer patients. *Applied Soft Computing*, *14*, 99–108. doi:10.1016/j.asoc.2013.07.016

Zughrat, A., Mahfouf, M., Yang, Y. Y., & Thornton, S. (2014). Support vector machines for class imbalance rail data classification with bootstrapping-based over-sampling and under-sampling. *IFAC Proceedings Volumes*, *47*(3), 8756-8761. doi:10.3182/20140824-6-ZA-1003.00794

P.Jhansi Lakshmi, received the M.Tech degree in computer science and engineering from the University college of Engineering, ANU, Guntur, India in 2010 and currently pursuing Ph.D. in the department of Computer Science and Engineering, VFSTR(Deemed to be University), Vadlamudi, Guntur, INDIA. From 2010 to 2016, she was a Faculty Member of the Department of Computer Science and Engineering, VFSTR (Deemed to be University), Vadlamudi, Guntur, INDIA.

T. Maruthi Padmaja received Ph.D. degree in computer Science from University of Hyderabad, Hyderabad, India, at the same time she was a Research Fellow at IDRBT, Hyderabad. Prior to that, she received a M.Tech degree from Tezpur University, Assam. Currently, she is a faculty member in the Department of Computer Science and Engineering, Vignan Institute of Technology and Science, Hyderabad, India.