A Novel Approach for Improving Software Quality Prediction

Anu K P, BinuRajan

Abstract—Software quality prediction is a process of utilizing software metrics such as code-level measurements and defect data to build classification models that are able to estimate the quality of program modules. These kinds of estimations can help software managers to effectively allocate potentially limited project resources, focusing on program modules that are of poor quality or likely to have a high number of faults. However, the effectiveness of such models depends on the quality of training data and also the underlying classification technique used for model calibration. The major problem that affects the quality of training datasets is high dimensionality and class imbalance. These problems can be alleviated by choosing necessary data preprocessing techniques before performing the classification. This paper presents an approach for using feature selection and data sampling together to deal with the problems. In this paper a wrapper based feature selection approach is used as the feature selection method and the ensemble learning method used is RUSBoost, in which random undersampling (RUS) is integrated into a boosting algorithm. The main purpose of this paper is to investigate the impact of feature selection along with RUSBoost approach, on the classification performance in the context of software quality prediction.

Index Terms—Software Quality Prediction, Feature Selection, RUSBoost.

I. INTRODUCTION

Nowadays it is difficult for us to imagine a life without devices that is controlled by software. So developing high quality software within the allotted time and budget is the key element for a productive and successful software project. This software quality can be measured in terms of software attributes such as efficiency, usability, reliability, portability, reusability and maintainability. James McCall and Barry Boehm have suggested various software quality characteristics.

The main software quality characteristic in concern is the reliability. According to the standard IEEE [1] the reliability is defined as: "The probability that software will not cause a failure of a system for a specified time under specified conditions". So we are able to measure software reliability with various factors such as the number of faults or time between failures. The main goal of software developers is to minimize the number of faults in the delivered code. Therefore, we need to fix the faults as early as possible, in order to ensure the reliability of software systems. Moreover, it is well known that earlier an error is identified, the better and more cost effectively it can be fixed. Therefore, there is a need to predict these software faults across the stages of software development process.

Revised Version Manuscript Received on August 20, 2015.

Anu K P, Department of Computer Science, Sree Chitra Thirunal College of Engineering, University of Kerala, Trivandrum, India.

Binu Rajan, Department of Computer Science, Sree Chitra Thirunal College of Engineering, University of Kerala, Trivandrum, India.

In this paper we focus on the prediction of the defect proneness of a software system for decreasing the project failures and lessen the total cost during the development and maintenance phases. Software metrics and fault data collected by software practitioners during the software development process has an important role in the prediction of software quality. Software metrics provide an automated way for software practitioners to assess the quality of software [2]. They are used to obtain reproducible measurements that can be useful for quality assurance, performance, debugging, management, and estimating costs. They are also used for finding defects in code (post release and prior to release), predicting defective code, predicting project success, and predicting project risk .There are three types of software metrics: Product metrics, Process metrics and Project metrics. Product metrics describe the characteristics of the product such as size, design features, complexity, performance and quality level. Process metrics can be used to improve software development and maintenance. These are very useful for improving software quality and productivity by quantitatively measuring the models of software development process [3]. Project metrics describe the project characteristics and execution. Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule and productivity [4].

This paper focuses on utilizing software metrics, such as code level measurements and defect data, to build defect predictors or software quality models. Predictive accuracy of a classification model is affected by the quality of training data. The major problem that affects the quality of training datasets are high dimensionality and class imbalance. High dimensionality occurs when too many features are used for building classification models. A number of problems may arise due to high dimensionality, such as longer learning time, a decline in prediction performance, and difficulty of comprehending the model [5]. Feature selection can be performed to avoid the problems caused due to high dimensionality.

Feature selection is an important data preprocessing activity and has been extensively studied in the data mining and machine learning community. The main goal of feature selection is to minimize the prediction errors of classifiers by selecting a subset of features. There are two categories of Feature selection techniques: filter-based approach and wrapper-based approach. Filters are feature selection algorithms in which a feature subset is selected without involving any learning algorithm while a wrapper-based approach is learner-dependent, because they use feedback from a learning algorithm to determine which feature(s) to include in building a classification model[6]. Recent research shows that filter-based feature selection techniques are

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

simple, quick, and effective methods to deal with this problem [7].

Another challenge for software measurement data is class imbalance. It occurs when instances of one class are outnumbered by the instances of the other class for a given dataset. The main disadvantage of such imbalanced data is that a traditional classification algorithm would tend to misclassify minority class instances as majority instances. The methods such as data sampling, boosting and bagging can be used to eliminate the class imbalance problems.

This paper presents an approach for using feature selection and data sampling together to deal with the problems. We used wrapper based feature selection method. Following feature selection, we employ ensemble boosting. To determine the role of feature selection and boosting, we also built a set of models which use no feature selection at all (either with Boost or simply by model building) and which use feature selection but no boosting.

The rest of the paper is organized as follows. Section II discusses related work. Section III provides methodology, including more detailed information about the feature selection process, RUSBoost,datasets and performance metric. Section IV presents the experimental results. Finally, the conclusion and future work are summarized in Section V.

II. RELATED WORKS

A number of approaches have been used to predict the defects in software subsystems. Recently, machine-learning and data mining approaches have been widely used for defect prediction. Introduction of machine learning approaches induces development of new software metrics for fault-prone module detection. Thus, several new metrics have been proposed so far.

Gao et al [8] presents an approach for using feature selection and data sampling together to deal with the problems caused due to high dimensionality and class imbalance. They have used software measurement datasets which is obtained from the PROMISE repository. They have used nine filter-based feature selection techniques. Among the nine techniques, three of them are standard feature ranking methods (chi-square, Information Gain, Relief), five of them are threshold-based feature selection techniques [9], and the last one is signal-to-noise. The data sampling method used was random undersampling and the SVM classifier was employed to build classification models.Their result shows that data sampling performed prior to feature selection resulted in significantly better performance than data sampling performed after feature selection.

Chawla et al [10] has proposed a SMOTEBoost algorithm that combines the Synthetic Minority Oversampling Technique (SMOTE) [11] and the standard boosting procedure. By using the SMOTE algorithm, the classification problems caused due to class imbalance has been eliminated. In short we can say that SMOTEBoost algorithm has utilized SMOTE for improving the prediction of the minority classes and utilized boosting to improve the predictive accuracy of classifiers. The main drawback of SMOTEBoost Algorithm is that it increases the complexity of the algorithm and takes longer training times since SMOTE is an oversampling technique.

Liu et al [12] has proposed a feature selection framework FECAR using Feature Clustering and feature Ranking. This framework firstly performs clustering, that is partitions original features into k clusters based on FF-Correlation measure. The main purpose of clustering is to make inner-cluster features strongly correlate with each other, while inter-cluster features are relatively independent. Then it selects relevant features from each cluster based on FC-Relevance measure and constructs the final feature subset.

Khoshgoftaar, et al [13] has proposed a neural-network model that be used to identify the modules which have high-risk of error. They have conducted their analysis on the data of a large telecommunication system. Principal components analysis was performed in the nine software design metrics and the resultant principal components where used as the input for the classification model. They have compared the classification results with the results of nonparametric discriminant analysis based classification and found the neural-network model had better predictive accuracy. Also neural networks based techniques are simpler and easier to use and produce more accurate models as compared to discriminant analysis.

Pandey et al [14] has proposed a Fuzzy Model for Early Software Fault Prediction Using Software Metrics. Basic steps of this model are Early Information gathering Phase, Information processing phase and Fault Prediction Phase. In Early Information gathering Phase, input/output variables are identified and fuzzy profile of these input/output variables are developed .In this stage we also define the fuzzy rule base that dictates how inputs will be utilized in getting the output. In the Information processing phase fuzzy inference process and defuzzification is performed and in the last phase, total number of faults in the software is predicted. But the main drawback is that the features are predefined.

Chen et al [15] has proposed a Two-stage Data Preprocessing Approach which includes Attribute selection and Instance filtering for Software Fault Prediction. In the attribute selection stage, firstly feature ranking is done to eliminate irrelevant features, and then a novel threshold based clustering algorithm is performed to remove redundant features. And in the next stage, that is instance reduction, random sampling is applied to reduce nfp instances (i.e., modules) since the similarity among nfp instances is usually high. By combing the attribute selection and instance reduction, a balanced and effective dataset can be constructed to improve the quality of training data for classification models. But the main disadvantage of this approach is that for instance reduction, random sampling is used which is not an effective approach.

Gao et al [16] has proposed an iterative feature selection approach working with an ensemble learning method to solve problems caused due to high dimensionality and class imbalance. The iterative feature selection approach samples the dataset k times and applies feature ranking to each sampled dataset. And to create a single feature ranking, the k different rankings are aggregated. They have used six filter-based feature ranking techniques. Following feature selection an ensemble boosting is performed. The ensemble learning method used is RUSBoost, in which random undersampling (RUS) is integrated into a boosting algorithm.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

III. METHODOLOGY

The main goal of this paper is to build a high-quality data set from the original dataset, which may contain irrelevant features and imbalanced classes. So we propose a model that combines both feature selection and instance reduction working with an ensemble learning method to solve both of these problems. A wrapper-based approach which uses decision tree as the learner and selectsthe features based on the balance measure is used as the feature selection method. Wrapper methods generally result in better performance than filter methods becausethe feature selection process is optimized for the classification algorithm to be used.We have used RUSBost as an ensemble learning method, in which random undersampling technique is integrated into a boosting algorithm to improve the performance of a learning algorithm. Complete framework of the proposed work is depicted in Figure 1.

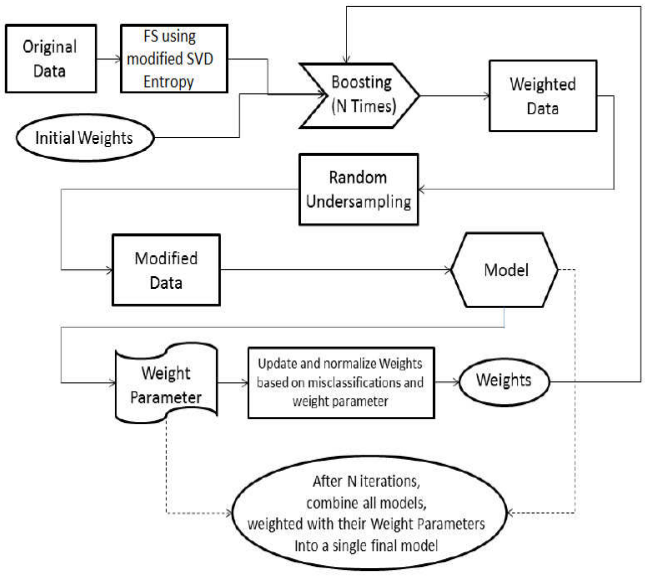

Figure 1: Framework of the proposed approach

A. *Wrapper based Feature selection*

In this paper we have used a wrapper-based approach[18]for feature selection which uses decision tree as the learner and selects the features based on the balance measure.

Balance measure is used to evaluate the relevance of each feature. It is calculatedusing probability of detection (pd) and probability of false alarm (pf) and is based onTP,FP,FN and TN.

- True positive (TP): If a software module is defective and is classified as defective.
- False negative (FN): If a module is defective and is classified as nondefective.
- True negative (TN): If a module is nondefective and is classified as nondefective.
- False positive (FP): If a module is nondefective and is classified as defective.

Target class	Defective	Non defective
Defective	TP	FN
Non defective	FP	TN

Table 1: **A Sample Confusion Matrix**

Collections of these values can be put into a confusion matrix as shown in Table 1. Formal definition of pd and pf are given in equation below

$$pd = \frac{TP}{TP + FN}$$

$$pf = \frac{FP}{FP + TN}$$

Thus the balance is calculated using the equation,

$$balance = 1 - \frac{\sqrt{(1-pd)^2 + (0-pf)^2}}{\sqrt{2}}$$

Algorithm 1: Wrapper Based Feature Selection
Input: Dataset $X_{n\times p}$
Output: Overall ranking of features
1: Begin
2: $P \leftarrow$ all possible pairwise combination of attributes from X (the cardinality will be nC_2)
3: $B = \varphi$, $A \leftarrow \varphi$
4: for all $P_i \in P_i$do
5: Create a dataset $d_i$using P_i
6: Classify $d_i$using any Classifier
7: Evaluate pd,pf,balance
8: $B_i \leftarrow$ {*pd,pf, balance*}
9: $B = BUBi$
10: end for
11: Sort P in decreasing order of balance using B
12: for all $P_i \in P$ do
13: for all $a_j \in P_i$do
14: $count_j$++
15: $A \leftarrow AU\{a_j\}$
16: end for
17: end for
18: Sort A on decreasing order of *countj*
19: $F \leftarrow \{a_1 \cup A\}$
20: *previousResult* ←balance evaluated by using F
21: for all i=2,...|A| do
22: $F \leftarrow F \cup \{a_i\}$
23: *currentResult* ←*balance* evaluated by using F
24: if currentResult>previousResult then
25: previousResult = currentResult
26: else
27: $F \leftarrow F \cup \{a_i\}$
28: end if
29: end for
30: Return the set F , and calculate pd, pf, and balance using F
31: end
32: return F

The data set contains different attributes that are used to classify software to be defective or non-defective. Combinations of all pairs of attributes are tested first along with a classifier. Then they are sorted based on their balance value. From this pool of attributes, pair of attributes is selected till there is a repetition of balance or very low value of balance. Then the frequency of each attribute is calculated from the selected attributes. The whole set of attributes is then arranged according to their frequency value. Finally we select one by one attribute from this sorted attribute pool until

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

the balance value increases which result the final selected attribute list. The algorithm is given in Algorithm 1.

B. RUS Boost

RUSBoost [19] is an ensemble learning method in which random undersampling is applied into each iteration of the boosting algorithm. Boosting is a meta learning technique for improving the classification performance of weak learners by iteratively creating an ensemble of weak hypothesis which are combined to predict the class of unlabeled examples. RUSBoost uses AdaBoost [20] as the boosting algorithm. Random undersampling (RUS) is a data sampling technique which balances the dataset by randomly removing the instances of majority class and thus alleviates the problem of class imbalance.

The algorithm first assigns equal weights to all the instances in training set. A weak hypothesis is formed in each iteration by the base learner. Then the error that is associated with the hypothesis is calculated and the weights of instances are updated such that the weights of the misclassified instances are increased and the weights of the correctly classified instances are decreased. This will allow the subsequent iteration of boosting to generate an hypothesis that are more likely to correctly classify the previously misclassified instances. In each iteration, a model will be created and a weight is assigned based on how well they classified the instances. This weight of all the models is used to classify the unlabeled instances. Random undersampling (RUS) is applied to the training set in each iteration of the boosting algorithm before building the classification models.

C. Datasets

To evaluate the effectiveness of the proposed approach, we have used the publicly available datasets provided by NASA. Each module of each data set describes the attributes of that module, plus the number of defects known for that module. The software metrics in the NASA project is mainly static code metrics that are directly calculated from the source code and give an idea about the complexity and the size of the source code. It is composed of line of code (LOC) metrics, McCabe metrics and Halstead metrics in general. Line of code metrics are directly related to the number of source code lines. This includes loc total, loc blank etc. McCabe metrics were developed by Thomas J. McCabe to measure the complexity of the source code. For egCyclomatic Complexity, Design Complexity etc. Halstead complexity metrics estimate reading complexity by counting operators and operands in a module.

Data set used here for the evaluation of the model is CM1 and KC1. CM1 is a “C" system implementing Spacecraft Instrument and KC3 is a “C++" system implementing storage management for receiving and processing ground data. CM1 has a total of 344 modules with 42 defective Instances and while KC3 has a total of 200 modules with 36 defective instances.CM1 has 38 attributes and KC3 has 40 attributes.

D. Performance Measures

The effectiveness of the prediction approach can be assessed by evaluating the classification performance of the model subsequently trained and tested with that particular approach.

To compare our results with other methodologies on the same data sets, we have used the performance measures such as Accuracy and Root Mean Square Error(RMSE) value. Accuracy considers both true positives and true negatives over all instances. In other words, accuracy shows the ratio of all correctly classified instances. Root Mean Squared Error (RMSE) is the square root of the mean of the squares of the probability score’s complement, divided by the number of cases in the partition.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

The feature selection approach has applied to two groups of software datasets (KC3 and CM1) which is available in the PROMISE data repository. The J48 Decision tree classifier algorithm is used as the classifier. It recursively splits a data set according to tests on attribute values in order to separate the possible predictions. The performance of the learners was assessed using accuracy and RMSE value.

The primary objective of the experiment is to evaluate the effectiveness of feature selection techniques followed by the RUSBoost algorithm. The sampling class ratio was set to 35:65 between fp and nfp modules throughout the experiment. To determine the role of feature selection and RUSBoost algorithm, we have also built a set of

Feature Selection Method	Number of Correctly Classified Instances		Accuracy		RMSE value	
	CM1	KC3	CM1	KC3	CM1	KC3
Information Gain	304	189	88.3721	94.5	0.311	0.2215
Gain Ratio	304	189	88.3721	94.5	0.311	0.2215
ReliefF	311	184	90.407	92	0.293	0.2512
Symmetric Uncertainty	304	189	88.3721	94.5	0.311	0.2215
Wrapper-based	327	190	97.3837	95	0.148	0.2136

Table I: Performance measures of CM1 and KC3 using various Feature Selection Method

models which use no feature selection at all (either with RUSBoost or simply by model building) and which use feature selection but no RUSBoost, i.e. we have considered three scenarios:

Scenario 1: Using both feature selection and RUSBoost

Scenario 2: Directly using RUSBoost without feature Selection

Scenario 3: With feature Selection Alone

Scenario 4: Without feature selection or boosting

Other than the modified wrapper-based feature selection method, we have also used feature selection method such as Information gain, gain ratio, reliefF and Symmetric Uncertainty inorder to evaluate the effectiveness of feature selection method.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

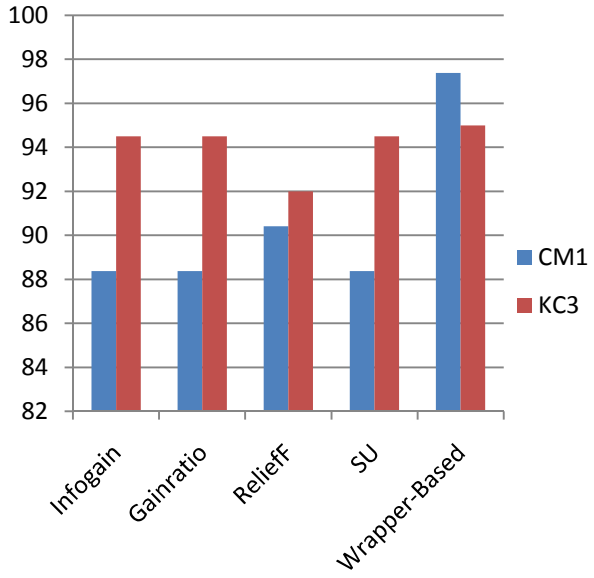

Figure 2: Barchart showing the accuracy of various feature selection method

Table 1 shows that the wrapper-based feature selection method outperforms comparing to the other feature selection method. Here we have considered the entire dataset as train and test data. Graphical representation of the accuracy of different dataset is depicted in Figure2.

	Accuracy		RMSE Value	
	CM1	KC3	CM1	KC3
Scenario 1	88.4058	87.5	0.2109	0.352
Scenario 2	85.5072	82.5	0.3653	0.4082
Scenario 3	86.9565	85	0.3083	0.3616
Scenario 4	82.6087	80	0.3974	0.4323

TABLE II: Performances of dataset CM1 and KC3 in different scenarios

Table II shows that Scenario 1outperforms the other scenarios for both the dataset, i.ewrapper based feature selection followed by the RUSBoost algorithm shows better performance than the others.Here we have considered 80% of the dataset as train data and the rest 20% is taken as test dataset. Graphical representation of the accuracy of different dataset is depicted in Figure3.

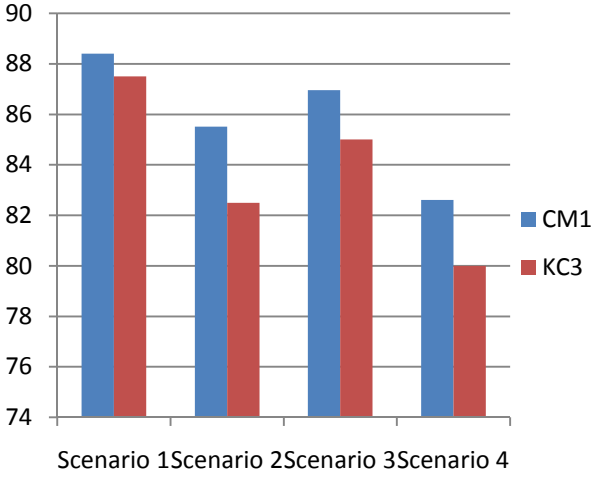

Figure 3: Barchart showing the accuracy of different Scenarios

V. CONCLUSION

The main problems that affect the quality of training data is high dimensionality and class imbalance. In this study, we present a wrapper based feature selection approach working along with an ensemble learning algorithm (RUSBoost) to solve these problems and thus improve the classification performance. For the evaluation of the proposed model, we have included two more scenarios i.e. directly using RUSBoost without feature selection and without feature selection or boosting. We have used CM1 and KC1 to evaluate these and found that our model shows better performance than the others

REFERENCES

1. IEEE recommended practice on software reliability. IEEE STD 1633-2008, pages c1–72, June 2008.
2. Cara Stein, Letha Etzkorn, Dawn Utley (2004) 'Computing Software Metrics from Design Documents.' ACMSE.
3. Yi Liu, jeng-Foung Yao, Gita Williams and Gerald Adkins (2007) 'Studying Software Metrics Based on Real-World Software Systems.' Journal of Computing Sciences in Colleges 22.
4. Stephen H. Kan (2002) Software Quality Metrics Overview [Online] 2nd ed. Boston: Addison Wesley Professional.
5. KehanGao , TaghiKhoshgoftaar and Amri Napolitano 'Improving Software Quality Estimation by Combining Boosting'and Feature Selection' 2013 12th International Conference on Machine Learning and Applications
6. Huanjing Wang, Taghi M. Khoshgoftaar and NaeemSeliya 'How Many Software Metrics Should be Selected for Defect Prediction?' Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference.
7. T. M. Khoshgoftaar and A. Napolitano, 'An empirical study of feature ranking techniques for software quality prediction,' International Journal of Software Engineering and Knowledge Engineering, 2012.
8. KehanGao and Taghi M. Khoshgoftaar ,'Software Defect Prediction for High-Dimensional and Class-Imbalanced Data,' Proceedings of the 23rd International Conference on Software Engineering & Knowledge Engineering (SEKE'2011), Eden Roc Renaissance, Miami Beach, USA, July 7-9, 2011.
9. T. M. Khoshgoftaar and K. Gao, 'A novel software metric selection technique using the area under roc curves,' in Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering, San Francisco, CA, July 1-3 2010, pp. 203–208.
10. N. V. Chawla, A. Lazarevic and K. Bowyer, 'Smoteboost: Improving prediction of the minority class in boosting,' Proceedings of Principles of Knowledge Discovery in Databases, 2003.
11. N. V. Chawla, K.W. Bowyer, 'Smote: Synthetic minority over-sampling technique,' Journal of Artificial Intelligence Research, 2002.
12. Shulong Liu, Xiang Chen, Wangshu Liu, Jiaqiang Chen, Qing Gu, Daoxu Chen, 'Fecar: A feature selection framework for software defect prediction,' IEEE 38th Annual International Computers, Software and Applications Conference, 2014.
13. Taghi M. Khoshgoftaar, Edward B. Allen and S. J. Aud, 'Application of neural networks to software quality modeling of a very large telecommunications system,' IEEE Transactions On Neural Networks, 1997.
14. K. Pandey . N. K. Goyal, 'Fuzzy model for early software fault prediction using process maturity and software metrics,' International Journal of Electronics Engineering, 2009.
15. Jiaqiang Chen, Shulong Liu, Wangshu Liu, Xiang Chen, Qing Gu, Daoxu Chen, 'A two-stage data preprocessing approach for software fault prediction,' IEEE Conference on Software Security and Reliability, 2014.
16. K. Gao, T. M. Khosgoftaar, and A. Napolitano, " Improving Software Quality Estimation by Combining Boosting and Feature Selection ", in 2013 12th International Conference on Machine Learning and Applications.
17. R. Varshavsky, A. Gottlieb, M. Linial, D. Horn, Novel unsupervised feature filtering of biological data, Bioinformatics 22 (14) (2006) e507–e513.
18. Md. SaeedSiddik Md. HabiburRahman Shah MostafaKhaled Mohammad ShoyaibJobaer Islam Khan, AlimUlGias. An attribute selection process for software defect prediction. 3rd International Conference On Informatics,Electronics & Vision, 2014.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

19. C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, “Rusboost: A hybrid approach to alleviating class imbalance,” IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 40, no. 1, pp. 185–197, January 2010.
20. R. Varshavsky, A. Gottlieb, M. Linial, D. Horn, Novel unsupervised feature filtering of biological data, Bioinformatics 22 (14) (2006) e507–e513. W.-K. Chen, Linear Networks and Systems (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

Anu K P, is currently doing herMTech in Computer Science and Engineering at Sree Chitra Thirunal College of Engineering under Kerala University, Trivandrum, Kerala, India. She received her B Tech Degree in Computer Science and Engineering at Cochin University College of Engineering in 2011.She has worked in the industry for one and half years and her interest domain include software engineering, machine learning and image processing.

Binu Rajan M R, is currently working as Assistant Professor in the Department of Computer Science and Engineering, Sree Chitra Thirunal College of Engineering, Trivandrum, Kerala. She received ME degree in Computer Science and Engineering from Anna University, Tamilnadu in 2005.She is also a research scholar at Kerala University. Her area of interest is machine learning and image processing.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.