Software Defect Prediction using Hybrid Approach

Myo Wai Thant, Nyein Thwet Thwet Aung
University of Information Technology, Yangon, Myanmar
myowaithant@uit.edu.mm, nyeinthwet@uit.edu.mm

Abstract

Defective software modules have significant impact over software quality leading to system crashes and software running error. Thus, Software Defect Prediction (SDP) mechanisms become essential part to enhance quality assurance activities, to allocate effort and resources more efficiently. Various machine learning approaches have been proposed to remove fault and unnecessary data. However, the imbalance distribution of software defects still remains as challenging task and leads to loss accuracy for most SDP methods. To overcome it, this paper proposed a hybrid method, which combine Support Vector Machine (SVM)-Radial Basis Function (RBF) as base learner for Adaptive Boost, with the use of Minimum-Redundancy-Maximum-Relevance (MRMR) feature selection. Then, the comparative analysis applied based on 5 datasets from NASA Metrics Data Program. The experimental results showed that hybrid approach with MRMR give better accuracy compared to SVM single learner, which is effective to deal with the imbalance datasets because the proposed method have good generalization and better performance measures.

Key Words- Software Defect Prediction, Feature Selection, Machine Learning, Support Vector Machine, AdaBoost

1. Introduction

Software products play an important role in our daily lives. Modern developed or developing countries were influenced by software systems and software product supports almost every field, including e-commerce, manufacturing, transportation, finance and other application areas. The software consists of a set of algorithms, practices, and running modules. Designing and developing the software system basically requires planning and allocating resources such as time, human experts, computer resources, tools, and infrastructure. As long as the role of software is important, software companies need to consider the defect rates for them. Even if the company has a lot of development experience, software failure rates are still increasing from time to time.

Software defect occurs when the results of a software application or product do not meet end-user expectations and software requirements. These defects are a type of programming error that can be caused by source code or requirement errors, resulting in failures, unpredictable or unexpected results. These defects adversely affect software quality and software reliability, leading to loss of capital, time and energy consumption. Maintenance costs and efforts are also increased in order to resolve failures. Thus, early prediction of software defects becomes a research area of concern.

Since the last two decades, researchers have been proposed a number of predictive models by using several types of classifiers such as Support Vector Machine (SVM), Random Forest (RF), Decision Tree (DS) to determine whether the software system defects are likely to occur. Unfortunately, the imbalance nature of the dataset becomes a major challenge in SDP process and leading to reduce the accuracy of the learning model. Imbalance nature means that the number of defective modules is relatively smaller than the number of non-defective module. If the target classes can be separated using the available features, then the distribution of the classes between them is not problematic. It only becomes an issue when this property affects the performance of the algorithms or the models that can be obtained.

Ensemble-based algorithms can handle and overcome imbalance issues to improve the accuracy and performance of prediction models. Among ensemble methods, AdaBoost has successful in building a strong learning model by combining weak classifiers as base learners. Ordinarily, AdaBoost uses decision tree as default base learner. However, the proposed system used SVM as base learner for AdaBoost because SVM is also a well-known classifier used in binary classification. Combining two famous algorithms can lead to get accurate results for imbalance nature. In addition, MRMR was also applied for feature selection. Experimental results of this hybrid approach have conducted on five NASA MDP datasets and comparison analysis applied between SVM single learner and proposed hybrid methods, which were evaluated by using Accuracy, Precision, Recall and F-score measures.

2. Related Work

A. Alsaeedi et.al [1] carried out a comparative study of supervised machine learning methods: Random Forest (RF), Decision Tree (DS), Support Vector Machine

(SVM), Linear Regression (LR) and ensemble classifiers AdaBoost and Bagging. The SMOTE oversampling technique was also applied to training data to conduct the experiments based on 10 NASA datasets. The outcomes of the findings demonstrate that RF, AdaBoost with RF, and bagging with DS performed well in imbalance issues.

A. Abdou et.al [2] investigated 3 types of ensemble learners: Boosting, Bagging and Random Forest with resample technique to carried out comparative analysis using 8 based learners which were tested on 7 datasets. The experimental result expressed that the accuracy of using ensemble learners is better than single learners.

Y. Gao et.al [3] proposed a method combining AdaBoost and Back-Propagation Neural Network (BPNN) to solve class imbalance problems and demonstrate the comprehensive performance with a BPNN model based on four datasets. It shows that BP-AdaBoost has a stronger generalization and better performance than BPNN, although it is still insufficient to predict minority class data. However, the author believes that the implementation of AdaBoost can handle imbalance dataset and feedback a high accuracy result if the suitable classifier is adapted for AdaBoost to deal with a variety of classification problems.

X. Li et.al [4] focus to proposed the algorithms named AdaBoostSVM and Diverse AdaBoostSVM which is the improve version to deal with accuracy/ diversity dilemma. Then, demonstrate that AdaBoostSVM achieves better performance than SVM on imbalance dataset problems and performs better than AdaBoost with neural network weak leaners.

S. Aleem et.al [5] also explored to compare performance of 11 methods: NB, MLP, SVM, AdaBoost, bagging, DS, RF, J48, KNN, RBF, and k-means. The results conducted on15 NASA datasets and claim that SVM and bagging techniques performed well in majority of datasets.

P. Mahajan [6] proposed 2 models: SVM combined with Principle Component Analysis (PCA) and AdaBoost SVM-RBF with PCA. And make a comparison among different types of SVM kernel functions and comparative analysis of 2 models demonstrate that the AdaBoost learning approach is much better than the others.

3. Proposed System Design

The proposed hybrid model for SDP based on boosting method is composed with 5 phases as shown in figure 1:

(1) **Data Preprocessing**: Check and remove missing value and invalid feature column.
(2) **Data Discretization**: Convert continuous data values into finite set of counterparts.
(3) **Feature Selection**: Filter the features based on MRMR discriminating criteria
(4) **Model Building**: Develop hybrid SVM AdaBoost model to get better results, and make a comparison with SVM based single learner.
(5) **Evaluation Measurements**: Accuracy and different performance measures was used to compare the hybrid method results against the single classifier results based on different size of imbalance datasets.

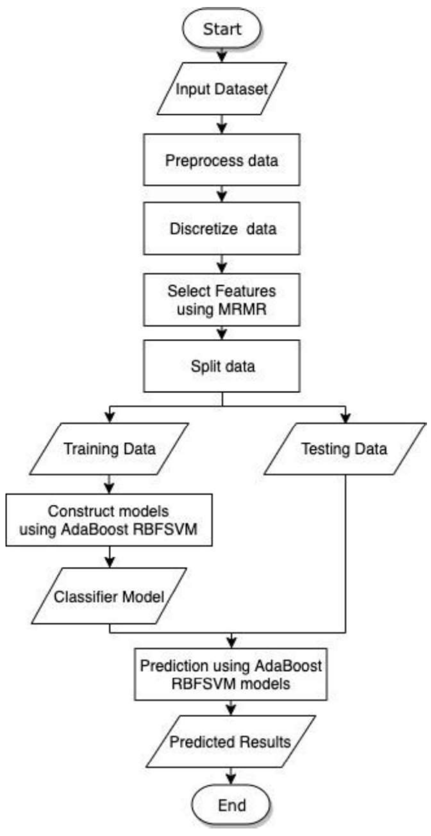


Figure 1. Proposed system design for software defect prediction

3.1. Dataset Description

As for datasets, there are five public SDP data available, namely PC1, PC2, PC3, PC4 and PC5 from NASA Metrics Data Program (MDP) defect datasets is used in this experiment. All of them are imbalance datasets which means the number of non-defective samples are much bigger than defective samples. These datasets have been collected from flight software written in C functions for earth orbiting satellite. Machine learning theory cannot apply directly to the software source code. Generally, each and every software modules had to measure with different software metrics or features extractors. The quality of selected datasets for this paper has measured with McCabe and Halstead extractors such as cyclomatic complexity, design complexity, time estimator, effort, operators, operands, length of the program, line count etc., and these metrics are the main input for defect prediction model. Defective modules identification can be divided into two classes. If the value is greater than zero, the software is classified as defect-prone, otherwise it is said to be free defect-prone. The detail structure of dataset is described in table 1.

Table 1. The detail of dataset

Data	Instances	Features	Defective (%)
PC1	1107	41	6.865
PC2	5589	41	0.411
PC3	1563	41	10.24
PC4	1458	41	12.21
PC5	17186	40	3.002

3.2. Data Preprocessing

Preprocessing stage is composed of two steps to handle invalid values. The first step is to check row by row instances and delete a particular instance if it has missing values or the values equal to '?' or the one which theoretically impossible occurrences. Some feature calculation theory can be referred as follow:

For examples, items were removed if:

- halstead length != num operands + num operators
- cyclomatic complexity > 1+ num operators
- call pairs > num operators
- average cyclomatic complexity > maximal cyclomatic complexity
- number of comments > lines of code
- public methods count > class methods count

The second step is to check column by column feature data and also remove particular column if it has the same constant value for the whole instances. The benefits of doing preprocessing is to facilitate the steps to extract desired information from the dataset and this works better in model building if the data has no noisy data.

3.3. Data Discretization

All of the datasets consist of continuous data which means the data metrics are described as an infinite number of possible values. Therefore, after obtaining a cleaned data set, a discretization method was applied to convert these continuous values into discrete equivalent values. This is because empirically discretization gives better results in discrete computation than continuous value computation in Mutual Information based feature selection methods like MRMR.

Discretization, also called binning or quantization, divides continuous numerical features into the predefined number of categories (bins), and makes the data discrete. The proposed system creates the instances with three bins, ordinal encoding based on the centroids of K-Means clustering procedure. The number of bins that we select has an impact on model building performance. Different datasets have the different number of bins to divide in order to get the best results. But the proposed system selects three fix bins for every dataset for making comparison purposes.

3.4. Feature Selection

After the data metrics have been preprocessed and discretized, one of the major development processes in machine learning is to select the appropriate feature selection algorithm for the dataset. This is because it has a great impact on the classification performance and accuracy of model predictions. Prediction performance is improved by removing redundant features and choose the right features. On the other hand, choosing irrelevant features can adversely affect not only performance but also the accuracy of the model.

Feature selection approaches can be roughly divided into three classes: wrapper-based methods, filter-based methods, and embedded methods. Among them, filter-based feature selection methods can perform independently from the learning procedure while the other methods, wrapper-based and embedded methods, combine feature selection and learning process to choose an optimal subset of features. Such kind of combined process typically requires the use of a nested cross-validation procedure which can increase computational cost and can lead to overfitting. For that reason, the proposed system focuses to use filter-based feature selection approach.

Among filter-based methods, the MRMR algorithm is the famous one that selects the features which are both maximally relevant to the target class and minimally redundant among themselves. In this algorithm, relevance and redundancy are calculated using Mutual Information for discrete data. For continuous data, there are two ways to apply MRMR. That is, firstly, apply data discretization method to convert continuous data into equivalent discrete data, and then apply Mutual Information. The second way is to calculate relevance using F-statistic and calculate redundancy using the Pearson correlation coefficient. MRMR website described that the first way leads to get better results than continuous value mutual information calculation. That's why the proposed system used data discretization method first, and then apply the Mutual Information equation to calculate relevance and redundancy.

Mutual Information calculation $I(X;Y)$ between two random variables, whose joint distribution $P(x,y)$ is defined in equation 1,

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} P(x,y) log \frac{P(x,y)}{P(x)P(y)} \qquad (1)$$

where, $P(x)$ and $P(y)$ are the marginal distribution of value x and y.

Relevance V and redundancy W calculation based on mutual information was defined in equations 2 and 3.

$$Maximum\ V = \frac{1}{|s|}\sum_{i\in s} I(h,i) \qquad (2)$$

$$Minimum\ W = \frac{1}{|s|^2}\sum_{i,j\in s} I(i,j) \qquad (3)$$

In these definitions, s is the set of features, i, j are features class and h is the target class (defective).

After getting the order of the features for relevance and redundancy, two common types of objective functions apply to extract important features. These two functions are Mutual Information Difference criterion (MID) and Mutual Information Quotient criterion (MIQ), which are representing the quotient or the difference of redundancy and relevance, as described in equation 4 and 5.

$$MRMR_{MID} = max_{i\in\Omega_s}[I(i,h) - \frac{1}{|s|}\sum_{j\in s} I(i,j)] \qquad (4)$$

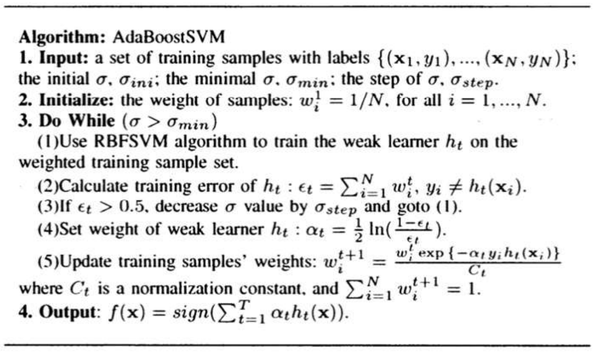
$$MRMR_{MIQ} = max_{i\in\Omega_s}\{I(i,h) \,/\, [\frac{1}{|s|}\sum_{j\in s} I(i,j)]\} \qquad (5)$$

The evaluation measurements value is vary based on the number of features that we select. According to the analysis results among ten, fifteen and twenty features, ten features have greater effects and higher accuracy than the others. Thus, the proposed system select ten features from all datasets and comparative analysis are applied based on these selected features.

3.5. Model Building

There are a lot of machine learning algorithms to build a predictive model for classification. Among them, SVM is considered one of the well-known classifiers to achieve high classification accuracy and it can deal with the linear problem as well as non-linear problems by using Kernel Function [6]. The main concept of SVM in binary classification problem is to find optimal separating hyperplane by maximizing the margin distance between them. Various types of kernels function like Linear, Quadratic, Polynomial, Radial Basis Function (RBF) and Multilayer Perceptron are used to map original data samples into higher dimensional feature space. The proposed system used RBF kernel function because it provides better results when the total number of features in dataset is small and the total number of the data sample is intermediate.

For the class imbalance problem, there are 2 levels to handle, that is to use the methods like resampling or feature selection at the data level and apply cost-sensitive or ensemble methods or single class learning at the algorithm level. Among ensemble methods, compare with bagging, boosting method performs better. As the most popular boosting methods, Adaptive Boosting was successful for binary classification and it trains the weak learner to be a strong learner by adjusting the weight value. Thus, this system used a Boosting method for algorithm level and MRMR feature selection algorithm for data level to overcome the imbalance problem. RBFSVM is used as a base learner for boosting method, but as mention before, SVM is a strong learner which can stand alone to divide classes for the data samples and it cannot work well as a base learner when integrate with boosting method. In order to use SVM and AdaBoost effectively, SVM has to be weakened first before combine with AdaBoost. The AdaBoost RBFSVM method flow is described in figure 2 [4].



```
**Algorithm:** AdaBoostSVM
**1. Input:** a set of training samples with labels $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$;
the initial $\sigma$, $\sigma_{ini}$; the minimal $\sigma$, $\sigma_{min}$; the step of $\sigma$, $\sigma_{step}$.
**2. Initialize:** the weight of samples: $w_i^1 = 1/N$, for all $i = 1, ..., N$.
**3. Do While** $(\sigma > \sigma_{min})$
    (1)Use RBFSVM algorithm to train the weak learner $h_t$ on the weighted training sample set.
    (2)Calculate training error of $h_t : \epsilon_t = \sum_{i=1}^{N} w_i^t,\ y_i \neq h_t(\mathbf{x}_i)$.
    (3)If $\epsilon_t > 0.5$, decrease $\sigma$ value by $\sigma_{step}$ and goto (1).
    (4)Set weight of weak learner $h_t : \alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$.
    (5)Update training samples' weights: $w_i^{t+1} = \frac{w_i^t \exp\{-\alpha_t y_i h_t(\mathbf{x}_i)\}}{C_t}$
    where $C_t$ is a normalization constant, and $\sum_{i=1}^{N} w_i^{t+1} = 1$.
**4. Output:** $f(\mathbf{x}) = sign(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}))$.
```

Figure 2 AdaBoost RBF-SVM

SVM is composed of two parameters: a Gaussian width σ and a regularization parameter C that impacts on classification performance. Overlarge σ makes RBFSVM too weak and smaller σ makes it stronger. And also, the variation of parameter C effects on the complexity of the learning model. Thus, adjusting both parameters is essential. When combine with AdaBoost, a relatively large σ value is preferred to get a weak learning ability of SVM. As described in figure 2, AdaBoost, machine learning meta-algorithm first initialize weight value to all data samples. When SVM training classifier obtains accuracy higher than 50%, update the weight for each data points and the process continue until σ decreases to a predetermined minimum value. By doing this, the proposed system moderately generates accurate SVM classifiers with uncorrelated errors and demonstrates better generalization performance than the existing AdaBoost methods on imbalance classification problems.

3.6. Evaluation Measurements

In the case of software defect prediction models, there are four possible outcomes for the entity after a prediction is made about whether the entity is defective or clean. The outcomes are as follows:

- A defective entity is classified as defective (true positive, TP)
- A defective entity is classified as clean (false negative, FN)
- A clean entity is classified as clean (true negative, TN)
- A clean entity is classified as defective (false positive, FP)

Based on these outcomes, accuracy measurement will be used for this work and the most popular measure, F-score will be applied to evaluate the performance. Higher F-score is a sign of a better model. Accuracy, Precision, Recall and F-score measurements can be calculated as follows:

Accuracy: percentage between the total number of correctly classified entities (TP + TN) and the whole entities (TP + TN + FP + FN).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (6)$$

Precision: a measure of result relevancy calculates by the proportion of correctly classified entities as defective (TP) among all entities classified as defective (TP + FP).

$$Precision = \frac{TP}{TP + FP} \qquad (7)$$

Recall: known as sensitivity to measure how many truly relevant results are returned, which can calculate by the proportion of correctly classified entities as defective (TP) among all defective entities (TP + FN).

$$Recall = \frac{TP}{TP + FN} \qquad (8)$$

F-score: a measurement that combines precision and recall as harmonic mean.

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \qquad (9)$$

4. Experimental Result

For the experiments, this system trained 5 imbalance datasets with SVM single learner and AdaBoost SVM with MRMR hybrid approach to analyze the performance and accuracy. Before building a learning model, ten features are selected by using MRMR methods. Different dataset has different types of selected features in order to get best results. The most common selected features are parameter_count, design_complexity, design_density, halstead_level and cyclomatic_density. All of them have higher MRMR value which are mostly relevance with target class in order to improve result prediction. After that, these datasets were split into two parts: training data and testing data. 70 percent from each dataset was taken as training data and the rest 30 percent was regarded as testing data for five datasets. Then, comparative analysis and evaluation measures has performed to the outcomes from both learning models: SVM and Hybrid approach with MRMR, which are summarizing in table 2 and table 3.

Table 2. Experimental result for SVM

Dataset	Accuracy	Precision	Recall	F-score
PC1	0.912	0.967	0.940	0.954
PC2	0.970	0.973	0.997	0.985
PC3	0.890	0.939	0.942	0.941
PC4	0.959	1	0.959	0.979
PC5	0.950	0.956	0.993	0.975

Table 3. Experimental result for MRMR + Hybrid method

Dataset	Accuracy	Precision	Recall	F-score
PC1	0.952	0.964	0.986	0.975
PC2	0.960	0.972	0.987	0.980
PC3	0.931	0.940	0.990	0.964
PC4	0.986	1	0.986	0.993
PC5	0.950	0.952	0.998	0.975

Generally, both of the learning methods give better results and most of the evaluation parameters values are over 90%. This is quite enough for some software systems to make early prediction by using only SVM. But as for safety critical systems, even 0.01% improvement rate is still important for them. Moreover, using correct classifier is also important to overcome imbalance problem as well.

The problem of imbalance dataset is that models trained on imbalance datasets often have poor results when they have to generalize in class prediction. The efficiency of a software defect prediction model also greatly depends on the training class distribution that means if we train the model with large amount of non-defective data, the predictive result will show as non-defective even we test with defect modules.

However, model building was performed using training data and evaluation was performed using unseen testing data. According to the experiments, hybrid method with MRMR give good generalization when classify these unseen observations. And predicted result also acceptable because the result did not come out the same class value for all testing data. This can be realized by checking precision and recall value.

The graph in figure 3 shows the accuracy and f-score measures for hybrid method results are obviously much better than single base learner for imbalance datasets. The average improve rate for accuracy, recall and f-score are

3.5 percent, 4 percent and 2 percent respectively. In the meanwhile, precision values for both methods are similar.

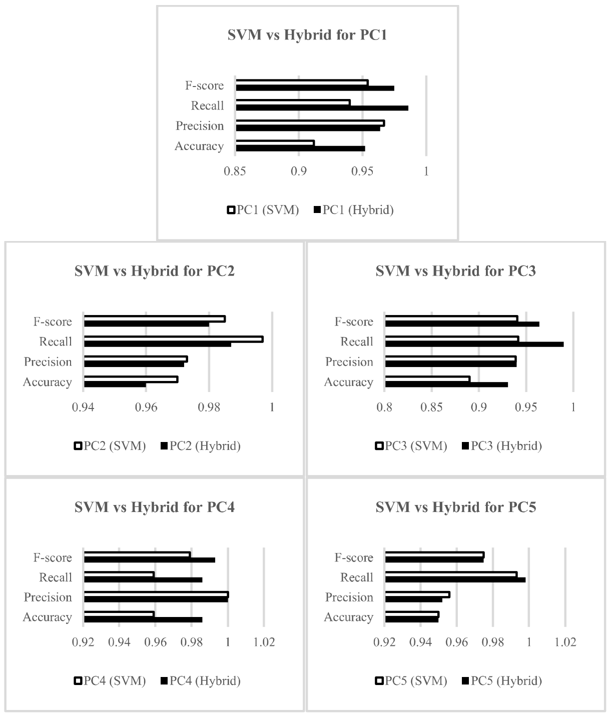


Figure 3 Comparative analysis of SVM and Hybrid method based on 5 datasets

Based on the experimental result, we can conclude that hybrid method is good enough and give better result than SVM in imbalance dataset. But in case of huge dataset which have less than 1 percent defective rate, SVM performs better than hybrid. Compare PC2 and PC5 graphs in figure 3, PC5 data samples size is double larger than PC2. Although PC5 have huge data size, the evaluation measures are not much difference between SVM and hybrid. As for PC2, which have 0.4% defective rate, hybrid method is slightly different than SVM. But both of them can still give great results for class imbalance problems. Thus, to pick suitable learning method based on the nature of dataset is important in defect prediction systems.

Moreover, the other factors also have impact on the accuracy and performance, such as the selected numbers for features, numbers of bin in discretization stage, the predefined value for σ in SVM and the splitting ratio for training and testing data size. For the purpose of making comparison, the proposed system used fix numbers for all datasets. Such kinds of value adjustments factors without using fix data might be tend to get higher accuracy and performance rather than current value. Thus, to choose correct values for other parameters and to build a suitable weak classifier are the key vectors for AdaBoost method.

5. Conclusion and Future Work

This paper focused on proposed a hybrid method that use SVM-RBF as base learner for AdaBoost and combine it with MRMR feature selection algorithm. According to my analysis, there is no paper was conduct the experiment for defect prediction system by combining these SVM-RBF AdaBoost with MRMR method. This proposed system aimed to enhance SVM in order to solve imbalance issues. The outcome showed that both methods have great results on the imbalance dataset. Actually, the hybrid approach is much better compared to the traditional SVM except the dataset has an extremely low defective rate. Moreover, as mention before, other factors have an impact on building models and performance. Therefore, one future direction is to explore every factor, adjust the value and figure out the most important factors which have greatly impact on performance and try to find the best optimal value combination to get the highest accuracy results.

6. References

[1] A. Alsaeedi and M.Z. Khan, “Software Defect Prediction Using Supervised Machine Learning and Ensemble Techniques: A Comparative Study”, *Journal of Software Engineering and Applications*, 2019, pp. 85-100.

[2] A. S. Abdou and N. R. Darwish, “Early Prediction of Software Defect Using Ensemble Learning: A Comparative Study”, *International Journal of Computer Applications*, 2018, Vol. 179 pp. 29-40.

[3] Y. Gao and C. Yang, “Software Defect Prediction based on Adaboost algorithm under Imbalance Distribution”, *Advances in Intelligent Systems Research*, 4th International Conference on Sensors, Mechatronics and Automation, 2016, Vol. 136 pp. 739-746.

[4] X. Li, L. Wang and E. Sung, “A Study of AdaBoost with SVM Based Weak Learners”, *Proceedings of International Joint Conference on Neural Network*, IEEE, Canada, 2005, pp. 196-201.

[5] S. Aleem, L. F. Capretz and F. Ahmed, “Benchmarking Machine Learning Technologies for Software Defect Detection”, *International Journal of Software Engineering & Applications*, 2015, Vol. 6 pp. 11-23.

[6] P. Mahajan, “Adaptive Boost Learning Approach: An Improved Method for Software Defect Prediction”, *International Journal of Advanced Research in Computer Science and Software Engineering*, 2015, Vol. 5 pp. 782-789.