Jurnal Teknologi

Full Paper

TACKLING IMBALANCED CLASS IN SOFTWARE DEFECT PREDICTION USING TWO-STEP CLUSTER BASED RANDOM UNDERSAMPLING AND STACKING TECHNIQUE

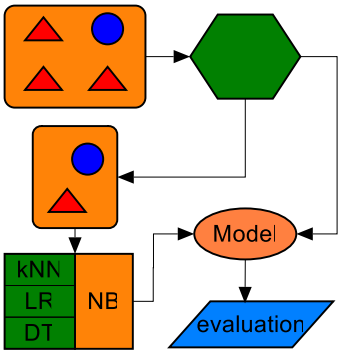
Adi Wijaya[a,c*], Romi Satria Wahono[b]

[a]Informatics Engineering Department, MH Thamrin University, Jakarta, Indonesia
[b]Faculty of Computer Science, Dian Nuswantoro University, Semarang, Indonesia
[c]IT Department, STIKIM, Jakarta, Indonesia

*Corresponding author
adiwjj@stikim.ac.id

Article history
Received
1 February 2017
Received in revised form
15 July 2017
Accepted
6 September 2017

Graphical abstract



Abstract

The cost of finding and correcting the software defects are high and increases exponentially in the software development. The software defect prediction (SDP) can be used in the early phases to reduce the testing and maintenance time, cost and effort; thus, improves the quality of the software. SDP performance is poor caused by imbalanced class in datasets where defective modules as minority compared to defect-free ones. In this study, we propose the combination of random undersampling based on two-step cluster and stacking technique for improving the accuracy of SDP. In stacking technique, Decision Tree, Logistic Regression and k-Nearest Neighbor are used as base learner while Naive Bayes as stacking model learner. The proposed method is evaluated using nine datasets from NASA metrics data program repository and area under curve (AUC) as main evaluation. Results have indicated that the proposed method yield excellent performance for 5 of 9 datasets (AUC > 0.9). Compared to the prior researches, the proposed method has first position for 3 datasets, second position for 5 datasets and only 1 dataset in third position for AUC value comparison. Therefore, it can be concluded that the proposed method has an impressive and promising result in prediction performance for most datasets compared with prior research performance.

Keywords: Software defect prediction, two-step cluster, random undersampling, ensemble learning, stacking technique

1.0 INTRODUCTION

Software defect prediction (SDP) is the process of predicting which parts of a code are defective and which are not [1]. The consequences of software failures may result in monetary and human losses [2] since the cost of correcting the defects increases exponentially if the defects are encountered later in the software development [3]. The accurate prediction of defect-prone software modules can help direct test effort, reduce costs and also improve the software testing process by focusing on fault-prone modules [4]. The SDP models can be used in the early phases of software development life cycle [3].

SDP becoming challenging task since software defect data in nature have a class imbalance because of the skewed distribution of defective and non-defective modules [5]. Although various techniques have been proposed by various researchers to address class imbalance problem issue, but no single technique outperformed the others in all the studies [6]. In many domain applications, learning with class imbalance distribution happens regularly. Imbalanced class distribution in datasets occur when one class, often

the one that is of more interest, that is insufficiently represented [7].

Various methods has been introduced to tackle this problem with two common approaches: data level and algorithm level [7]. Data level approach employs a pre-processing step to rebalance the class distribution. This is done by either employing undersampling or oversampling to reduce the imbalance ratio in training data [1]. Undersampling removes a smaller number of instances from majority class in order to minimize the discrepancy between the two classes whereas oversampling duplicates instances from minority class. Meanwhile, the algorithm level methods could be categorized as dedicated algorithms that directly learn the imbalance distribution from the classes in the datasets, such as: one class learning classifications, cost-sensitive learning and ensemble learning [7].

Undersampling which reduce certain instances from a majority class could lead to loss of potentially important information about the class; while in oversampling, the duplication only increase the number of examples but do not provide new information about the class [7]. In algorithm level, although various techniques have been proposed by various researchers to address this issue, but no single technique outperformed the others in all the studies [6]. A combination of data level and algorithm level as proposed by [8]–[10] yield an impressive performance in handling imbalanced class classification in SDP.

Many researches have been conducted in SDP both in class imbalance issue and noisy attribute issue. SDP datasets have imbalance class in nature, since positive class (defective module) is minority compared to majority class (non-defective module). To tackle these problems, some approaches were proposed by researcher, such as: data level by using association rule [11], [12], algorithm level by using optimization [5] and ensemble or meta-learning technique [8], [9], [13].

A novel supervised method for detecting software entities with defects, based on relational association rule mining, called DPRAR (Defect Prediction using Relational Association Rules) was proposed by [11]. Their classifier is based on the discovery of relational association rules for predicting whether a software module is or it is not defective. They used ten NASA metrics data program (NASA MDP) datasets, such as: CM1, KC1, KC3, PC1, JM1, MC2, MW1, PC2, PC3 and PC4. They used five evaluations, such as: area under curve (AUC), accuracy, recall or sensitivity, specificity and precision. Their model yield potential and promising result.

Another method from association mining approach was proposed by [12] and combines with Naïve Bayes (NB). The proposed algorithm preprocesses data by setting specific metric values as missing and improves the prediction of defective modules. NB classifier has been developed before and after the proposed preprocessing data. They used 5 NASA MDP datasets, such as: CM1, KC3, PC3, MC1 and AR4 and used AUC, accuracy, recall or sensitivity, specificity and precision as evaluation. Their method showed that recall of the classifier after the proposed preprocessing has improved and has resulted in up to 40% performance gain.

A combination of traditional Artificial Neural Network (ANN) and the novel Artificial Bee Colony (ABC) algorithm are used by [5]. ABC was used to find optimal weights of ANN. They used accuracy, probability of detection, probability of false alarm, balance, AUC, and Normalized Expected Cost of Misclassification as the main performance indicators. Five NASA MDP datasets were used such as: KC1, KC2, CM1, PC1 and JM1. Their proposed method is better compared to other five algorithms although the performance difference is not significant.

Another approach to deal with imbalance class classification in SDP was used Bagging as ensemble technique and using feature selection using Genetic Algorithm as proposed by [8] and Particle Swarm Optimization as proposed by [9]. Their used 9 NASA MDP datasets, such as: CM1, KC1, KC3, MC2, MW1, PC1, PC2, PC3 and PC4. They used AUC as main evaluation and their proposed method yield impressive performance compared to ten standard algorithms.

A combined selected ensemble learning models with efficient feature selection was proposed by [13] to address data imbalance and feature redundancy and mitigate their effects on the defect classification performance. They used 4 NASA MDP datasets such as: KC3, MC1, PC2, PC4 and AUC as evaluation. The ensemble technique, so called average probability ensemble (APE) combined with greedy forward selection was gain optimal result for AUC values of above 0.9 for the NASA MDP datasets such as: PC2, PC4, and MC1.

While there is no single method that achieves the best performance for all NASA MDP datasets, this indicated that SDP still open issue and challenging task. In this study, a combination of data level and ensemble technique is proposed. Stacking technique is chosen as ensemble technique due to its performance is often astonishingly good [14].

In this research, we propose the combination of two-step cluster (TSC) based random undersampling (RUS) and stacking technique (TSC-RUS+S) for improving the accuracy of SDP. TSC-RUS is applied to deal with the imbalanced class and Stacking technique is used to leverage the performance of classifier in SDP. RUS is chosen since many prior research used this approach when deal with imbalanced class classification. While TSC is chosen as cluster algorithm since TSC promises to solve at least some of these problems e.g.: the ability to deal with mixed-type variables and large data sets, automatic determination of the optimum number of clusters, and variables which may not be normally distributed [15].

This paper is organized as follows. In section 2, the methodology of this study is explained including the proposed method. The experimental results and discussion of comparing the proposed method with other prior researches are presented in section 4. Finally, our work of this paper is summarized in the last section.

2.0 METHODOLOGY

We propose a method called TSC-RUS+S, a random undersampling based on two-step cluster (TSC) and stacking technique for tackle imbalanced class problem in software defect prediction. TSC is one of clustering algorithm that developed firstly by [16] and designed to handle very large datasets. TSC is provided by the statistical package SPSS. TSC able to handle both continuous and categorical variables [15], [17]. In stacking technique, Decision Tree (DT), Logistic Regression (LR) and k-Nearest Neighbor (kNN) are used as base learner while Naive Bayes (NB) as stacking model learner. Figure 1 shows block diagram of the proposed method.

The proposed method is evaluated using nine NASA metrics data program (NASA MDP) datasets [18], i.e.: CM1, KC1, KC3, MC2, MW1, PC1, PC2, PC3, PC4 as used by [8], [9], [11]. As shown in Figure 1, nine NASA MDP datasets multiplied first and feed to training phase and testing phase respectively; where training phase is used for build the model and testing phase is used to test the model and evaluate its performance.

In training phase, dataset then clustered using TSC algorithm. Number of cluster is setup to 4 clusters as the same idea with create 4-binning or 4 quartile. Then, we do random undersampling for each cluster, so majority class and minority class is the same number for each cluster. After that, the new dataset is made by combining from all clusters with same proportion for each class.

After that, the new dataset is feed to stacking technique with 10 fold cross validation approach where dataset will split into 10 parts dataset, 1 part as testing dataset and the rest as training datasets and this process repeated 10 times. We use DT, LR and kNN as base learner and NB as model learner in stacking technique. After learning process complete, the model will feed with testing dataset in testing phase and then we record the evaluation result.

In this study, proposed method evaluated by using the classifier effectiveness based on confusion matrix with the main evaluation is area under curve (AUC) as used by [5], [8], [9], [11], [12], [13]. AUC has the potential to significantly improve convergence across empirical experiments in software defect prediction [8] and the use of the AUC to improve cross-study comparability [19]. A basic guide for classifying the accuracy of a diagnostic test based on AUC as stated by [20] as follows:

- 0.90 - 1.00 = excellent classification
- 0.80 - 0.90 = good classification
- 0.70 - 0.80 = fair classification
- 0.60 - 0.70 = poor classification
- 0.50 - 0.60 = failure.

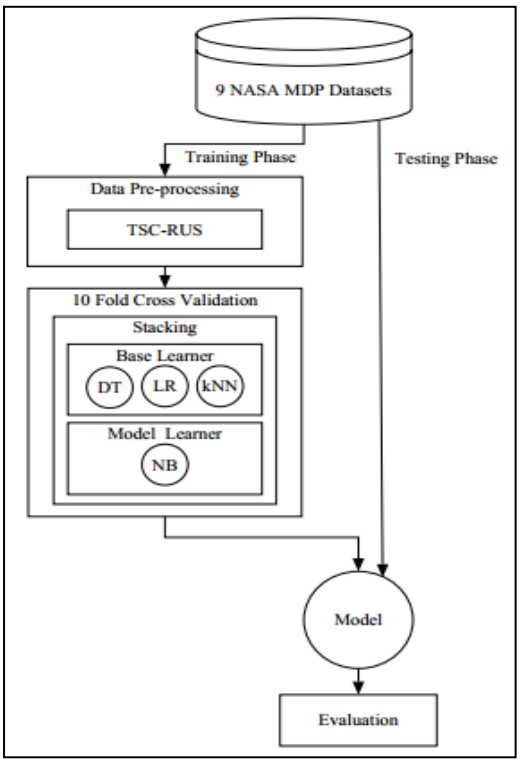


Figure 1 Block diagram of the proposed method

Another evaluation of the proposed method i.e.: recall or sensitivity (SN) as used by [5], [11], [12]; specificity (SP) and precision (PR) as used by [11], [12]. These evaluations based on the confusion matrix which contains the value true positive (TP), true negative (TN), false positive (FP) and false negative (FN) as shown in Table 1. TP means when predicted label is defective and the actual label is defective too. When predicted label is defective but the actual label is non-defective, it called FP. TN is the same with TP but in matter of non-defective label, while FN is when predicted label is non-defective but actually it label is defective. It calculated based on confusion matrix that produces from the model. Based on confusion matrix, the measurement calculation are as follows:

(i) SN : measures the proportion of positive pattern instances that are correctly recognized as positive

$$SN = TP / (TP + FN) \quad (1)$$

(ii) SP : measures the proportion of negative pattern instances that are correctly recognized as negative

$$SP = TN / (TN + FP) \quad (2)$$

(iii) PR : measures the probability that a positively predicted pattern instance is labeled as positive

$$PR = TP / (TP + FP) \quad (3)$$

Table 1 Confusion matrix

Predicted	Actual	
	Defective	non-Defective
Defective	TP	FP
non-Defective	FN	TN

3.0 RESULTS AND DISCUSSION

The experiments are conducted using a computing platform based on Intel Core 2 Duo 2.2 GHz CPU, 2 GB RAM and Microsoft Windows 7 32-bit operating system, Rapidminer version 5.3 as data analytics tool and also IBM SPSS Statistics version 20 as statistics tool. Rapidminer will produce both AUC and confusion matrix as the calculation output; while IBM SPSS Statistics will produce t-test for comparison between proposed method and prior research statistically.

3.1 Stacking Only Technique

First of all, we conducted experiment on nine NASA MDP datasets with stacking technique only. The confusion matrix as shown in Table 2 is produced for each datasets from Rapidminer and then we evaluate the method by calculate SN, SP and PR, while AUC is directly calculated by Rapidminer. Table 3 shows the complete method evaluation.

Table 2 Confusion matrix for stacking only technique

Dataset	TP	TN	FP	FN
CM1	16	271	31	26
KC1	128	1607	164	197
KC3	13	147	17	23
MC2	19	76	7	25
MW1	15	202	35	12
PC1	22	651	47	39
PC2	8	1515	54	8
PC3	132	239	746	8
PC4	1187	97	81	34

As shown in Table 3, the method yields 2 AUC with excellent classification, 3 good classifications and 4 fair classifications. Meanwhile, SN vary from 0.361 – 0.972, SP vary from 0.243 – 0.966 and PR vary from 0.129 – 0.936. In matter of AUC, the method mostly produced fair classification; while in matter of SN and PR, the method produced mostly low result; while in matter of SP, the method produced excellent result mostly. However, based on this result, the method is promising enough since it still produced 2 excellent classification and the recall (SN) is still better than its precision (PR). As one of the objective of the research conducted by [12] that try to improving the recall.

Table 3 The method evaluation for stacking only technique

Dataset	AUC	SN	SP	PR
CM1	0.785[c]	0.381	0.897	0.34
KC1	0.804[b]	0.394	0.907	0.438
KC3	0.743[c]	0.361	0.896	0.433
MC2	0.796[c]	0.432	0.916	0.731
MW1	0.785[c]	0.556	0.852	0.3
PC1	0.805[b]	0.361	0.933	0.319
PC2	0.949[a]	0.5	0.966	0.129
PC3	0.804[b]	0.943	0.243	0.15
PC4	0.953[a]	0.972	0.545	0.936

[a.] excellent, [b.] good, [c.] fair

3.2 TSC-RUS+S Technique

In the second experiment, we implemented random undersampling based on two-step cluster (TSC-RUS) and combined with stacking technique (TSC-RUS+S). The experimental result showed in Table 4 and Table 5. The confusion matrix is produced for nine datasets from Rapidminer and then we calculate SN, SP and PR, while AUC is directly calculated by Rapidminer.

Table 4 Confusion matrix for TSC-RUS+S technique

Dataset	TP	TN	FP	FN
CM1	37	91	211	5
KC1	209	1547	224	116
KC3	27	99	65	9
MC2	12	79	4	32
MW1	22	156	81	5
PC1	12	681	17	49
PC2	16	1299	270	0
PC3	92	835	150	48
PC4	1185	60	118	36

As shown in Table 5, the second experiment yield better result in all evaluation rather than the first experiment. In matter of AUC, 6 classified as excellent and 3 classified as good. SN vary from 0.262 – 1; SP vary from 0.603 – 0.979 and PR vary from 0.047 – 0.962. In matter of SN and PR, the second experiment produced less in lower value and higher in upper value rather than the first experiment. While in matter of SP, the second experiment produced higher result lower value and upper value rather than the first experiment.

Table 5 The method evaluation for TSC-RUS+S technique

Dataset	AUC	SN	SP	PR
CM1	0.86[b]	0.952	0.652	0.276
KC1	0.916[a]	0.935	0.738	0.396
KC3	0.843[b]	1	0.64	0.379
MC2	0.926[a]	0.409	0.952	0.818
MW1	0.916[a]	1	0.603	0.223
PC1	0.893[b]	0.262	0.979	0.516
PC2	0.955[a]	1	0.792	0.047
PC3	0.917[a]	1	0.644	0.285
PC4	0.952[a]	0.93	0.747	0.962

[a]. excellent, [b]. good

3.3 Experimental Results Comparison

In order to more detailed comparison between the first and the second experiment, we presented the comparison in Table 6. The bold font face indicates that the best value for each evaluation. As shown in Table 6, the second experiment (TSC-RUS+S) is outperforms in almost datasets in matter of AUC (8 of 9 datasets). Meanwhile, the first experiment (stacking only) outperforms the second experiment only in PC4.

Table 6 Result comparison stacking only vs. TSC-RUS+S technique

Data-set	AUC		SN		SP	
	(1)	(2)	(1)	(2)	(1)	(2)
CM1	0.785	**0.86**	0.381	**0.952**	**0.897**	0.652
KC1	0.804	**0.916**	0.394	**0.935**	**0.907**	0.738
KC3	0.743	**0.843**	0.361	**1**	**0.896**	0.64
MC2	0.796	**0.926**	**0.432**	0.409	0.916	**0.952**
MW1	0.785	**0.916**	0.556	**1**	**0.852**	0.603
PC1	0.805	**0.893**	**0.361**	0.262	0.933	**0.979**
PC2	0.949	**0.955**	0.5	**1**	**0.966**	0.792
PC3	0.804	**0.917**	0.943	**1**	0.243	**0.644**
PC4	**0.953**	0.952	**0.972**	0.93	0.545	**0.747**

(1). stacking only; (2). TSC-RUS+S

In matter of SN, the second experiment outperforms the first experiment in almost dataset (6 of 9 dataset). Meanwhile, in matter of SP, the first experiment outperforms the second experiment in almost dataset (5 of 9 datasets). However, overall the second experiment outperform and better than the first experiment since the main evaluation in imbalanced class classification such as SDP is AUC as stated by [8], [19].

3.4 AUC Comparison with Prior Researches

Since this research used the public dataset and many existing prior researches conducted and using the same datasets, therefore we must compare our research result with those prior researches. Six prior researches were selected and the comparison is presented in Table 7. Three prior researches conducted with the same nine datasets, while 3 prior researches have conducted with the same three datasets. In this comparison, we use AUC since AUC is main evaluation in imbalanced class classification.

Table 7 AUC comparison with prior researches

DS	(1)	(2)	(3)	(4)	(5)	(6)	(7)
CM1	0.77	0.70	**0.89**	-	0.73	0.77	0.860
KC1	0.85	0.79	0.82	-	-	0.80	**0.916**
KC3	0.71	0.68	0.85	0.86	**0.92**	-	0.843
MC2	0.73	0.74	0.87	-	-	-	**0.926**
MW1	0.75	0.72	**0.92**	-	-	-	0.916
PC1	0.79	0.78	**0.92**	-	-	0.82	0.893
PC2	0.82	0.81	**0.96**	0.95	-	-	0.955
PC3	0.78	0.78	**0.91**	-	0.77	-	**0.91**
PC4	0.85	0.86	0.89	**0.96**	-	-	0.952

Note that, (1) method called PSOFS+B proposed by [9], (2) method called GAFS+B proposed by [8], (3) method called DPRAR proposed by [11], (4) method called Enhanced APE proposed by [13], (5) method called NB+AM proposed by [12], (6) method called ANN+ABC proposed by [5] and (7) is the proposed method called TSC-RUS+S. In this comparison, bold font face means the best value in AUC, while underline font face means the second best value in AUC.

As presented in Table 7, proposed method outperforms 3 of 9 datasets and became the second best 5 of 9 datasets and only 1 dataset had fourth position. This result indicated that the proposed method is promising and yield excellent result since produced excellent AUC in almost datasets (6 of 9 datasets). In this comparison, we also test our proposed method result compared to other prior research statistically as shown in Table 8. We used t-test to compare between our proposed method with method (1), (2) and (3); since these methods had the same all datasets.

Table 8 T-Test AUC comparison with prior researches

Comparison schemes	p-value	Mean Difference	Difference
Proposed Method vs. (1)	0.000	0.127	Significant
Proposed Method vs. (2)	0.000	0.145	Significant
Proposed Method vs. (3)	0.394	0.013	not Significant

We conducted t-test to detect whether there is difference between proposed method and others or not and also to show which methods is the better performance. As shown in Table 8, in pair 1, proposed method vs. PSOFS+B (1), p-value = 0.000 which means there is significant difference between proposed method with PSOFS+B. Proposed method is the better method rather than PSOFS+B since the mean difference is positive value (0.127); it indicated that the first method (which is proposed method) had higher value in AUC. In pair 2, proposed method vs. GAFS+B (2), has the same result with pair 1, where there is significant difference (p-value = 0.000) and proposed method gain better result since the mean difference is 0.145 (positive value). In pair 3, proposed method vs. DPRAR (3), there is not significant difference since p-value = 0.394 (p-value > 0.05) and the mean difference is also very small (0.013). Based on t-test, proposed method indicated excellent and competitive result with the state-of-the-art research result.

4.0 CONCLUSION

A novel hybrid method that integrates random undersampling as data level approach based on two-step cluster and stacking technique as algorithm approach is proposed in this paper, to improve the accuracy of software defect prediction (SDP). The proposed method is applied to deal with the class imbalance problem in SDP. Experimental results show that the proposed method yields an impressive and promising improvement in prediction performance for most datasets and prior research results both in AUC and recall or sensitivity. This promising result in line with several prior research were conducted which aim to improve not just AUC but also recall or sensitivity as well.

Future research will be concerned with benchmarking the proposed method with other clustering techniques, such as DBSCAN, Fuzzy C-means, etc. and other meta-learning techniques, such as bagging and boosting. Feature discretization based on clustering technique to tackle noisy attribute as nature of SDP dataset also challenging to be studied in our future work.

Acknowledgement

We would like to express our gratitude to RSW Intelligent Systems Research Group (RSW-ISRG) for warm discussion about this research.

References

[1] M. J. Siers and M. Z. Islam. 2015. Software Defect prediction using a Cost Sensitive Decision Forest and Voting, and a Potential Solution to the Class Imbalance Problem. *Inf. Syst.* 51: 62-71.

[2] H. B. Yadav and D. K. Yadav. 2015. A Fuzzy Logic Based Approach for Phase-wise Software Defects Prediction Using Software Metrics. *Inf. Softw. Technol.* 63: 44-57.

[3] R. Malhotra. 2016. An Empirical Framework for Defect Prediction Using Machine Learning Techniques with Android Software. *Appl. Soft Comput.* 1-17.

[4] C. Catal. 2011. Software Fault Prediction : A Literature Review and Current Trends. *Expert Syst. Appl.* 38(4): 4626-4636.

[5] Ö. F. Arar and K. Ayan. 2015. Software Defect Prediction Using Cost-sensitive Neural Network. *Appl. Soft Comput.* 33: 263-277.

[6] I. Arora, V. Tetarwal, and A. Saha. 2015. Open Issues in Software Defect Prediction. *Procedia Comput. Sci.* 46:. 906-912.

[7] A. Ali, S. M. Shamsuddin, and A. L. Ralescu. 2015. Classification with Class Imbalance Problem: A Review. *Int. J. Adv. Soft Comput. its Appl.* 7(3): 176-204.

[8] R. S. Wahono and N. S. Herman. 2013. Genetic Feature Selection for Software Defect Prediction. *Adv. Sci. Lett.* 4(2): 400-407.

[9] R. S. Wahono and N. Suryana. 2013. Combining Particle Swarm Optimization based Feature Selection and Bagging Technique for Software Defect Prediction. *Int. J. Softw. Eng. Its Appl.* 7(5): 153-166.

[10] I. H. Laradji, M. Alshayeb, and L. Ghouti. 2015. Software Defect Prediction Using Ensemble Learning on Selected Features. *Inf. Softw. Technol.* 58: 388-402.

[11] G. Czibula, Z. Marian, and I. G. Czibula. 2014. Software Defect Prediction Using Relational Association Rule Mining. *Inf. Sci. (Ny).* 264: 260-278.

[12] Z. A. Rana, M. A. Mian, and S. Shamail. 2015. Improving Recall of software Defect Prediction Models Using Association Mining. *Knowledge-Based Syst.* 90: 1-13.

[13] I. H. Laradji, M. Alshayeb, and L. Ghouti. 2015. Software Defect Prediction Using Ensemble Learning on Selected Features. *Inf. Softw. Technol.* 58: 388-402.

[14] R. S. Wahono. 2015. A Systematic Literature Review of Software Defect Prediction: Research Trends, Datasets, Methods and Frameworks. *J. Softw. Eng.* 1: 1.

[15] C. Michailidou, P. Maheras, a. Arseni-Papadimititriou, F. Kolyva-Machera, and C. Anagnostopoulou. 2008. A Study of Weather Types at Athens and Thessaloniki and Their Relationship to Circulation Types for the Cold-wet Period, Part I: Two-Step Cluster Analysis. *Theor. Appl. Climatol.* 97(1–2): 163-177.

[16] T. Chiu, D. Fang, J. Chen, Y. Wang, and C. Jeris. 2001. A Robust and Scalable Clustering Algorithm for Mixed Type Attributes in Large Database Environment. *Proceedings of the 7th ACM SIGKDD Internation-al Conference on Knowledge Discovery and Data Mining*. 263-268.

[17] S. M. Satish and S. Bharadhwaj. 2010. Information Search Behaviour Among New Car Buyers: A Two-step Cluster Analysis. *IIMB Manag. Rev.* 22(1–2): 5-15.

[18] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson. 2012. Reflections on the NASA MDP Data Sets. *IET Softw.* 6(February): 549-558.

[19] S. Lessmann, S. Member, B. Baesens, C. Mues, and S. Pietsch. 2008. Benchmarking Classification Models for Software Defect Prediction : A Proposed Framework and Novel Findings. *IEEE Trans. Softw. Eng.* 34(4): 485-496.

[20] F. Gorunescu. 2011. *Data Mining: Concepts,Models and Techniques*. Springer-Verlag Berlin Heidelberg.