Comparison of Classification Methods on Software Defect Data Sets

Hnin Yi San, Dr. Khine Khine Oo
University of Computer Studies,Yangon
hninyisan@gmail.com,k2khine@gmail.com

Abstract

Nowadays it is difficult for us to imagine a life without devices that is controlled by software. Software quality prediction is the important process of software development processes. It is a process of utilizing software metrics such as code-level measurements and defect data to estimate software quality modules. A more useful and efficient mechanism is k Nearest Neighbor method to classify class of target data based on k nearest training dataset. By applying the concept of k-NN, we propose a new mechanism called Class Base Weighted k-NN with Biner Algorithm (CBW k-NN) to find the range of training dataset where the target data has the maximum likelihood of occurrence by Biner and classify class of target data based on this range. The main purpose of this paper is to know the effective classification method for software defect datasets that exploit information from the NASA MDP (PC1, CM1, JM1) datasets.

Keywords: *Biner, Class Based Weighted k Nearest Neighbor, Classification, k Nearest Neighbor, NASA MDP dataset, Software Defect Prediction*

1. Introduction

The costs of finding and correcting software defects have been the most expensive activity during both software development and software maintenance. Therefore, developing high quality software within the allotted time and budget is the key element for a productive and successful software project. The main software quality characteristic in concern is the software defect. A panel at IEEE Metrics 20022 also concluded that manual software reviews can find only 60 percent of defects [10]. Therefore, software defect prediction has been an important research topic in the software engineering field, especially to solve the inefficiency and ineffectiveness of existing industrial approach of software testing and reviews. 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.

Many mechanisms for software defect prediction are Decision trees, Ensemble Classifier, Random Forest, Naïve Bayes Classifiers, Support Vector Machine, Neural Networks, k Nearest Neighbor Classifier etc. that helps in improving the defect prediction performance. Machine learning techniques are proven to be useful in terms of software defect prediction [1]. The data from software repository contains lots of information in assessing software quality and machine learning techniques can be applied on them in order to extract software defect information. The classification process is sometimes called the supervised learning that is the machine learning task of inferring a function from labeled training data consist of a set of training examples. Among them, k Nearest Neighbor classifier is instance-based learning algorithms and a more useful and efficient classification method.

Classification consists of predicting a certain outcome based on a given input. It uses input data, also called training set where all objects are already known class labels. The objective of classification algorithm is to analyze and learn from the training dataset and classify test data for which the class labels are not known.

The main purpose of the paper is to describe the effective method for classification on software defect datasets using k Nearest Neighbors Classifier and Class Based Weighted k-NN with Biner Algorithm and solve classifiers on imbalance data set using both methods. For this purpose, it use three datasets related to NASA MDP software defect datasets named are PC1, CM1 and JM1. The results after classification of software defect data come in terms of certain efficiency parameters like Accuracy, Reliability, Mean Absolute Error, and Root Mean Squared Error in order to compare two methods.

This paper is organized as follows: the related work is described in section 2. Section 3 describes the background details used in the proposed approach. In section 4, we explain the proposed approach adopted to classify software defective or non-defective results. Experimental work is carried out by section 5 and finally section 6 present the conclusion and future work.

2. Related Work

Anil Kumar Singh et. al [8] performed Software Fault Prediction System by using two methods; Fuzzy c-means clustering approach and k-Nearest Neighbors Classifier technique with the real time data set named PC1, taken from NASA MDP software projects. The performance comparison of this system was recorded on the basis of accuracy, net reliability, RMSE and MAE values. They applied the training and testing methodology, wherein a project is chosen for training the system. Two classifier approach was applied on the same project and the final calculated values are then used to classify the modules of project as fault prone or fault free. Simulation was carried out using MATLAB 2010a. They recommended that the k-Nearest Neighbors Classifier method gives more accuracy and less error as compared to Fuzzy C-means clustering method on the basis of evaluation parameters: accuracy, reliability, MSE and RMSE.

Anu K P et. al [7] presented a wrapper based feature selection approach working along with an ensemble learning algorithm, RUSBoost, to solve the problem that affect the quality of training data is high dimensionality and class imbalance and thus improve the classification performance in the context of software quality prediction. And the J48 Decision tree classifier algorithm was used as the classifier that recursively splits a data set according to tests on attribute values in order to separate the possible predictions. They were applied two groups of software datasets KC3 and CM1 from PROMISE data repository. For the evaluation of the proposed model, they had included two more scenarios i.e. directly using RUSBoost without feature selection and without feature selection or boosting. They represented the wrapper-based feature selection method outperforms comparing to the other feature selection methods by using accuracy and RMSE value. Therefore, they recommended wrapper based feature selection followed by RUSBoost algorithm showed better performance than others.

Dazy Arya [2] described Software Fault Prediction System using Fuzzy c-means clustering approach and a hybrid technique (Combination of Fuzzy c-means and Particle Swarm Optimization) to compare performance evaluation results. Fuzzy clustering based techniques are discussed for the comparative analysis in order to predict level of impact of faults in NASA's PC1. The author was carried out the problem simulation by using MATLAB 2010a. The author was found that the hybrid method gives more accuracy and less error as compared to Fuzzy C-means clustering method on the basis of evaluation parameters such as accuracy, reliability, MSE and RMSE.

3. Background

Data mining is about solving problems by analyzing data already present in databases. Data mining is a powerful tool that can help to find patterns and relationships within data that go beyond simple analysis. Classification, one of the data mining techniques, is to find a derived model that describes and distinguishes data classes or concepts based on the analysis set of training data [12].

3.1 k Nearest Neighbor Classifier

The k Nearest Neighbor (k-NN) classification is a popular, most widely used classification method for classifying class of target data based on training data set. The training data set is used to classify each member of a target dataset [3]. A more sophisticated approach, k-NN classification finds a group of k objects in the training set that are closest to the test object, and bases the assignment of a label on the predominance of a particular class in this neighborhood. For each data point in the target dataset, the distance metric between target data and all training data are calculated and sorted. The threshold value (k) has to eliminate all distance values depend on threshold value (k) and taken into account based on classes of selected distance values to classify target data [6].

Algorithm 1: k Nearest Neighbor Algorithm

Inputs: X, C, k, x

Output: class label for Query instance x

1. Calculate $d(x_i,x)$, where (i=1,2,….n)

2. Order $d(x_i,x)$ from lowest to highest,

3. Eliminate the k nearest instances to x: D_x^k

4. Taken into account the imbalance class distribution around the neighborhood of the query instances: D_x^k

5. Return class label for Query instance x

Figure 1. k Nearest Neighbor Algorithm

Table 1 reveals the meaning of various symbols used in k Nearest Neighbor. This table is used to form better understanding of the algorithm.

Table 1. List of Symbols Used in the k Nearest Neighbor Algorithm

Symbol	Meaning
X	Each data of training dataset
C	Class labels of X
K	Threshold value
X	Query Instance or target data point
$d(x_i,x)$	Manhattan distance between query instance and training data
D_x^k	Distance values between query instance and training dataset based on k value

3.1.1 Threshold Value (k)

There are several key issues that affect the performance of k-NN. One is the choice of k. If k is too small, then the result can be sensitive to noise points. On the other hand, if k is too large, then the neighborhood may include too many points from other classes. Threshold value (k) should use at least square root value of train dataset and be less than half of train dataset. CBW k-NN with Biner algorithm calculates for each range if train dataset is greater than twice of k value. If the number of classes is 2, threshold value should be an odd number to classify clearly.

3.1.2 Manhattan Distance

There are many techniques to measure the similarity between the target data and each training dataset. One of such widely used techniques is Manhattan distance. It is one of the powerful calculating distance techniques to classify class of target data based on all training dataset.

$$d(x_i, x) = \sum_{j=1}^{t} \left| x_j - x \right| \qquad \textbf{(3.1)}$$

Where $d(x_i,x)$ is distance value between target data and training data. x are target data point and x_i represent each data point of train dataset respectively. Also i value is the number of records in training data set and j value is the number of attributes(column) in each data point respectively.

3.2 Class Based Weighted k-NN

By the regular k Nearest Neighbor classifier, distance values are no limitation range. So, Class based Weighted k Nearest Neighbor classifier calculate a weight is assigned to each of the class based on how its instances are classified in the neighborhood of query instance [4].

Algorithm 2: CBW k-NN Algorithm

Inputs: x, k, Range (start, end)

Output: class label for Query instance x

1. find closest range by using BINER function
2. Calculate $d(x_i,x)$ for subrange
3. Order $d(x_i,x)$ from lowest to highest
4. Eliminate the k nearest instances to x: D_x^k
5. Calculate weight value distance value $d(x_i,x)$
6. Calculate total weight values of each class
7. Multiply class based weighted factor and total weight values of each class
8. Compare final total weight values of all classes
9. Return class label for Query instance x

Figure 2. CBW k-NN Algorithm

Table 2 reveals the meaning of various symbols used to form better understanding of CBW k-NN.

Table 2. List of Symbols Used in the CBW k-NN Algorithm

Symbol	Meaning
Range (start, end)	Whole Training dataset range
Range (s, e)	Final closest subrange
K	Threshold value
X	Query Instance or target data point
Subrange	Return range of BINER function
$d(x_i,x)$	Manhattan distance between query instance and training data
D_x^k	Distance values between query instance and training dataset based on k value

$$\omega_i = \begin{cases} \dfrac{d_k - d_i}{d_k - d_1} & \text{if } d_k \neq d_1 \\ 1 & \text{if } d_k = d_1 \end{cases} \qquad \textbf{(3.2)}$$

Where w_i is weight value of i^{th} distance value and i value is 1 to k. d_k represent largest distance value of k eliminated distance values and d_1 is smallest distance value.

$$\omega(c) = 1/\ frequency[c] \qquad \textbf{(3.3)}$$

Where $w(c)$ is class based weighted factor for each class and $frequency[c]$ is total count of each class at training dataset [5], [6].

3.3 BINER Algorithm

Biner algorithm narrows down the range in which the response variable has the maximum likelihood of occurrence instead of directly predicting the value of response variable. The data is hierarchically partitioned in the preprocessing step, and search for the partition in which the response has the maximum likelihood of occurrence is carried out at the runtime and then interpolates to give the output [4].

Algorithm 3: Biner Algorithm

Inputs: x, k, Range (start, end)

Output: Range (s, e)

1. while end – start > 2 * k do
2. r =end – start
3. s_1, e_1 = start, start + r/2
4. s_2, e_2 = start +r/4 , start + 3r/4
5. s_3, e_3 = start + r/2, end
6. d_1 = getDistance (RangeMean(s_1, e_1), x)
7. d_2 = getDistance (RangeMean(s_2, e_2), x)
8. d_3 = getDistance (RangeMean(s_3, e_3), x)
9. if similar (d_1 , d_2, d_3) then
10. return (start, end)
11. else
12. start, end = s_i, e_i
13. end if
14. end while
15. return (start, end)

Figure 3. Biner Algorithm

Table 3 presents the various symbols used in Biner algorithm.

Table 3. List of Symbols Used in Biner Algorithm

Symbol	Meaning
Range (start, end)	Whole Training dataset range
K	Threshold value
X	Query Instance or target data point
s_1, e_1	First range of three subranges
d_1	Distance value between query instance and first range
RangeMean(s_1,e_1)	Mean values for first range of training dataset
s_i, e_i	Selected subrange to calculate next sub ranges

$$getDistance() = \sqrt{\sum \frac{(\mu_i - q_i)^2}{\sigma_i^2}} \quad \textbf{(3.4)}$$

Where q_i is the i^{th} attribute of the target data, μ_i is the mean of i^{th} attribute values in all data points in the range and σ_i is the standard deviation of values of the i^{th} attribute in the whole database.

$$\sigma_i^2 = \frac{1}{N}\left[\sum x_i^2 - \frac{1}{N}(\sum x_i)^2\right] \quad \textbf{(3.5)}$$

Where i is number of attributes of each data point and N is number of data point at training dataset.

3.4 Software Defect Dataset Nature

In this paper, we use three software defect prediction data sets from NASA MDP. Individual attributes per dataset, together with some general statistics and descriptions, are given in Table 4 and 5. The software defect datasets have various scales of line of code (LOC), various software modules coded by several different programming languages including C, C++ and Java, and various types of code metrics including code size, Halstead's complexity and McCabe's complexity [10]. The following tables are NASA MDP datasets nature and attribute nature taken from NASA MDP software projects [9].

Table 4. NASA MDP Dataset Nature

Name	Language	Modules	Attributes	System
PC1	C	1107	22	Flight software for earth orbiting satellite
CM1	C	372	22	NASA spacecraft instrument
JM1	C	1085	22	Real-time predictive ground system

Table 5. Dataset Attributes' Nature

No	Name	Description
1	Loc	McCabe's line count of code
2	V(G)	McCabe "cyclomatic complexity"
3	EV(G)	McCabe "essential complexity"
4	IV(G)	McCabe "design complexity"
5	N	Halstead total operators + operands
6	V	Halstead "volume"
7	L	Halstead "program length"
8	D	Halstead "difficulty"
9	I	Halstead "intelligence"
10	E	Halstead "effort"

11	B	Halstead "number of delivered bugs"
12	T	Halstead's time estimator
13	lOCode	Halstead's line count
14	lOComment	Halstead's count of lines of comments
15	lOBlank	Halstead's count of blank lines
16	loCodeAndCom-ment	Halstead's count of lines of code with comments
17	Uniq_Op	unique operators
18	Uniq_Opnd	unique operands
19	Total_Op	total operators
20	Total_Opnd	total operands
21	branchCount	Branch count of the flow graph
22	defects	{false,true} non-defect or defect

3.5 Performance Evaluation Methods

The performance evaluation methods use to compare output results with other methodologies on the same datasets by using Accuracy, Reliability, Mean Absolute Error (MAE), and Root Mean Square Error (RMSE) [11]. Accuracy calculate correct classification rate. Precision represent correctness and Recall represent defect detection rate. F-measure or Reliability combines precision and recall in a single efficiency measure by taking their harmonic mean [1].

$$Accuracy = \frac{(TP + TN)}{TP + TN + FP + FN} \quad \textbf{(3.6)}$$

$$\Pr ecision = \frac{TP}{TP + FP} \quad \textbf{(3.7)}$$

$$\mathrm{Re}\, call = \frac{TP}{TP + FN} \quad \textbf{(3.8)}$$

$$F - measure = \frac{2 * \Pr ecision * \mathrm{Re}\, call}{\Pr ecision + \mathrm{Re}\, call} \quad \textbf{(3.9)}$$

Where

TP- if a software module is defective and is classified as defective

FN- if a software module is defective and is classified as non-defective

TN- if a software module is non-defective and is classified as non-defective

FP- if a software module is non-defective and is classified as defective

MAE and RMSE can be used together to diagnose the variation in the errors in testing datasets. MAE is the average over the verification sample of the absolute values of the differences between predict and corresponding actual value. RMSE is calculated square of the difference between predict and corresponding value and then averaged over the sample.

$$MAE = \frac{\sum_{i=1}^{m} |x_i - \bar{x}|}{m} \quad \textbf{(3.10)}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{m} (x_i - \bar{x})^2}{m}} \quad \textbf{(3.11)}$$

Where i value is number of testing data point and x_i is class of i^{th} target data and is $\bar{x}$ its mean value.

4. Proposed Approach

4.1 Overview Design of Proposed System

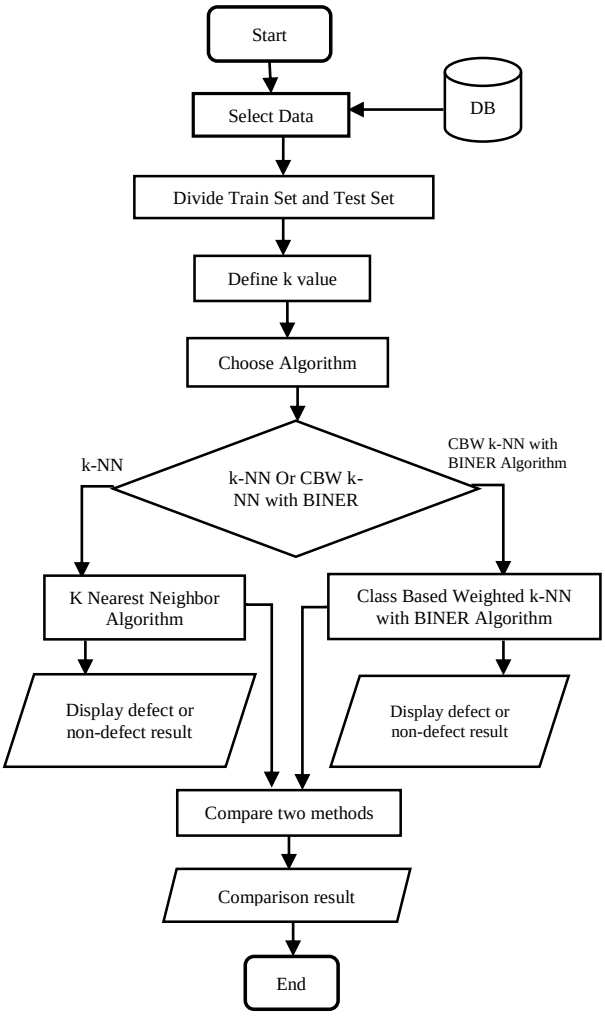

Figure 4. System Flow Diagram

In overview design of the system, the main process is to classify target data that are defective or non-defective, user can select dataset and divide training set and testing set and define k value that is the smaller distance range instead of the whole dataset. User can choose method such as k-NN and Class Based Weighted k-NN with BINER Algorithm for classification. The system displays to user classification results (defective or non-defective) according to chosen method. The system also compares results by calculating accuracy, reliability, and error rate methods (MAE and RMSE) for both methods.

4.2 Data Preprocessing

For two methods, data preparation and cleaning procedure is use by defect datasets to check unbalanced dataset and cleaning dataset.

```
Procedure: Software Defect data preparation and cleaning
(1) read all instances and given file name
(2) check the uploaded file for unbalanced instances
(3) while(!end of dataset file)
(4) begin
(5)     read record of dataset line by line
(6)     tokenize attribute value of record
(7) end
(8) count original dataset
(9) remove same record in dataset array
(10) count remaining record dataset
```

Figure 5. Software Defect Data Preparation and Cleaning Procedure

4.3 Problem Formulation with k-NN

In PC1 dataset, it used to create flight software for earth orbiting satellite and total instances are 932 records. It are divided training and testing dataset by ratio 2:1 as training is 622 and testing is 310. Firstly, it calculates distance values between testing datasets and the whole training datasets by using Manhattan distance. By ordering the instances according to distance values, it sorts and eliminates distance values by threshold value (k) 50. The threshold value is dynamic changes for various amounts of dataset. Finally, total true class value is 44 and total false class value is 6 by k value 50. Therefore, it classifies this instance class 'True' by using training true and false ratio 1:13.

4.4 Problem Formulation with BINER

It is sort training dataset by ascending order to divide training datasets by biner function. Firstly, it divides three range of whole dataset for PC1 such as Range 1(0,311), Range 2(156,467), and Range 3(311,622). And then it calculates sub range for each instance of testing dataset by Eq 3.4 and biner function according to selected next ranges. Like k-NN, it calculates distance values are between testing dataset and the result training dataset range. By ordering the instances according to distance values, it sorts and eliminates distance values by threshold value (k) 50. And then it calculates weight value for each eliminated distance value by Eq 3.2 and total weight value of each class as 'True' and 'False' class. For unbalance dataset, it calculates class based weighted factor of each class. Finally, it calculates final weight values of each class and classifies class result according to final weights.

5. Experimental Result

For experimental purpose, to demonstrate the defectiveness of our approach, we have evaluated Accuracy, F-measure, MAE and RMSE of PC1, CM1 and JM1 dataset. It measures the system performance by using various kind of k value as PC1(k=25,32,50), CM1(k=16,25,38) and JM1 (k=27,42,65). The performance range for evaluation criteria is 0% to 100%. The more range increase, the higher the accuracy except for MAE. For demonstration purpose, the evaluation results have shown in following figures. In figure 6, kNN is better than CBW kNN by k value (25).

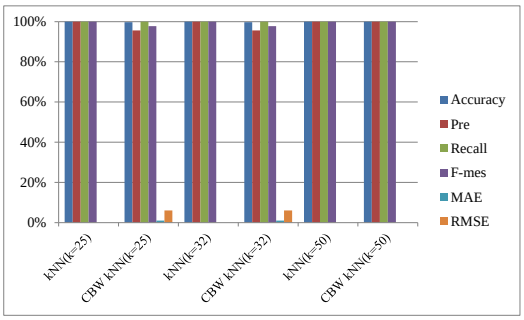

Figure 6. Performance Evaluation Result for PC1

In figure 7, kNN is better than CBW kNN by k value (16).

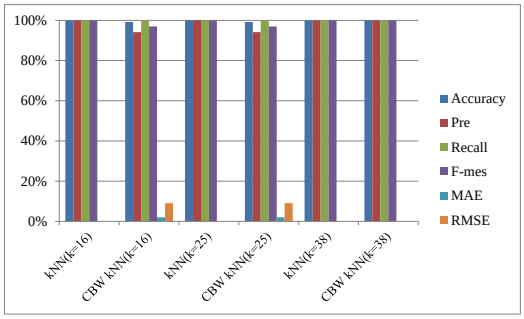

Figure 7. Performance Evaluation Result for CM1

In figure 8, CBW kNN is better than kNN by k value (27).

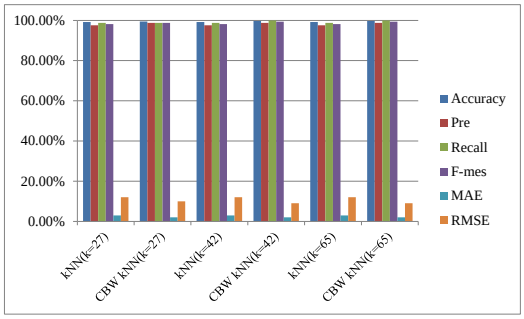

Figure 8. Performance Evaluation Result for JM1

6. Conclusion and Future Work

In this paper, we implement an approach called class based weighted k nearest neighbor with biner algorithm. For selecting an appropriate classifier to test software defective on imbalance data set, this system provide on the classification algorithm based on k-nearest neighbors. This system also makes the comparison of classification for software defective and non-defective using weighted k-nearest neighbor algorithm with class based weighted factor which is focused on the range of nearest data set using BINER function. This system evaluates accuracy, reliability, error-rate of two methods such as Class Based Weighted k-NN with BINER Algorithm and k-NN classifier to compare two methods. The factor observed according to this system implementation is that the more k value increase in CBW k-NN, the higher accuracy in classification result. In contrast, there is no change in k-NN classification according to k value. In future, this system should be upgraded to classify various types of software defect datasets.

References

[1] Saiqa Aleem, Luiz Fernando Capretz and Faheem Ahmed, "*Benchmarking Machine Learning Techniques for Software Defect Detection*", International Journal of Software Engineering & Applications (IJSEA), Vol.6, No.3, May 2015.

[2] Dazy Arya, "*PSO Optimized Software Fault Prediction system using Fuzzy C-Means*", International Journal of Digital Application & Contemporary research (Vol. 3, Issue 6, Jan 2015).

[3] Nitin Bhatia, "*Survey of Nearest Neighbor Techniques*", International Journal of Computer Science and Information Security (IJCSIS), Vol. 8, No. 2, 2010.

[4] Harshit Dubey, "*Efficient and Accurate kNN based Classification and Regression*" CENTER FOR DATA ENGINEERING, International Institute of Information Technology, Hyderabad - 500 032, INDIA, March 2013.

[5] Jianping Gou, Lan Du, Yuhong Zhang and Taisong Xiong, "*A New Distance-weighted k-nearest Neighbor Classifier*", Journal of Information & Computational Science 9: 6 (2012) 1429–1436.

[6] Klaus Hechenbichler and Klaus Schliep, *"Weighted k-Nearest-Neighbor Techniques and Ordinal Classification*", Sonderforschungsbereich 386, Paper 399 13th October 2004.

[7] Anu K P, BinuRajan, "*A Novel Approach for Improving Software Quality Prediction*", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-4 Issue-6, August 2015.

[8] Anil Kumar Singh, Rajkumar Goel and Pankaj Kumar, "*Comparative Analysis of Accuracy Prediction using Fuzzy C-Means and KNN Clasiffier*", International Journal of Digital Application & Contemporary research, Volume 2, Issue 7, February 2014.

[9] Manjula.C.M. Prasad, Lilly Florence and Arti Arya, "*A Study on Software Metrics based Software Defect Prediction using Data Mining and Machine Learning Techniques*", International Journal of Database Theory and Application, Vol.8, No.3 (2015).

[10] Romi Satria Wahono, Nanna Suryana Herman, *Genetic Feature Selection for Software Defect Prediction*", Advanced Science Letters, Vol. 20, 239–244, 2014.

[11] https://www.otexts.org/fpp/2/5, last access on 28 Dec. 2017

[12] Ian H. Witten and Eibe Frank, *Data mining: practical machine learning tools and techniques*, Diane Cerra, QA76.9.D343W58 2005.