Absolute Correlation Weighted Naïve Bayes for Software Defect Prediction

Rizky Tri Asmono, Romi Satria Wahono and Abdul Syukur
Faculty of Computer Science, Dian Nuswantoro University, Semarang, Indonesia
rtriasmono@gmail.com, romi@romisatriawahono.net, abdul_s@dosen.dinus.ac.id

Abstract: The maintenance phase of the software project can be very expensive for the developer team and harmful to the users because some flawed software modules. It can be avoided by detecting defects as early as possible. Software defect prediction will provide an opportunity for the developer team to test modules or files that have a high probability defect. Naïve Bayes has been used to predict software defects. However, Naive Bayes assumes all attributes are equally important and are not related each other while, in fact, this assumption is not true in many cases. Absolute value of correlation coefficient has been proposed as weighting method to overcome Naïve Bayes assumptions. In this study, Absolute Correlation Weighted Naïve Bayes have been proposed. The results Wilcoxon signed-rank test on experiment results show that the proposed method improves the performance of Naïve Bayes for classifying defect-prone on software defect prediction.

Keywords: Software Defect Prediction, Naïve Bayes, Absolute Correlation Weighted Naïve Bayes, Correlation Coefficient, Absolute Correlation

1 INTRODUCTION

Software is computer programs and related documentation. Software products can be expanded for a specific customer or may be developed for the common marketplace in accordance with the functions and needs. Develop a flawless software is difficult and often times there are some errors or bugs unknown or unexpected defects, although the software-development methodology has been applied with cautious(Okutan & Yıldız, 2012). The maintenance phase of the software project will be very expensive for the developer team and harmful to the users because some flawed software modules. Surely, it can be avoided by detecting defects as early as possible. Defect prediction will provide an opportunity for the developer team to test modules or files that have a high probability defect. The completion of defect prediction problems currently focusing on 1) estimate the number of defects in the existing software systems, 2) discovering defect associations and 3) classification on the defect-prone of software, specially defect and non-defect label(Song, Jia, Shepperd, Ying, & Liu, 2011). The things that detrimental to users and developer team can be avoided as early as possible with a software defect prediction.

For classifying defect-prone, Hall conducted an investigation on software defect prediction(T. Hall, Beecham, Bowes, Gray, & Counsell, 2012). Hall compared Decision Tree, Logistic Regression, Naïve Bayes, Neural Network, C4.5 etc. The results of the investigation showed the two best methods that can be used to predict software defects are Naive Bayes (NB) and Logistic Regression. Logistic Regression is a statistical probabilistic classification method. The advantages of logistic regression are computationally inexpensive, slight to implement and mild to interpret knowledge representation. The disadvantages of logistic regression are prone to under fitting and may have a low accuracy(Harrington, 2012). Naïve Bayes is a modest probabilistic classifier. It is very comfortable because it does not require any complicated parameter estimation. Therefore, Naive Bayes ready to be used for large amounts of data. Moreover, Naive Bayes is also very facile to explain so the users who do not have the technological classification capability can understand the reason why the classification was made(X. Wu & Kumar, 2009). However, Naive Bayes assumes all attributes are equally important and are not related each other while, in fact, this assumption is not true in many cases(J. Wu & Cai, 2011), (Turhan & Bener, 2009), (Liangxiao Jiang, 2011). The assumption made by Naive Bayes can be detrimental to its performance in real data mining applications.

Naïve Bayes assumes that all the attributes are not dependent on each other, in fact, the class depends on others attribute. Naïve Bayes also assumes the relationship between class and one attribute as strong as the relationship between class and other attribute(Turhan & Bener, 2009). The case mentioned previously clearly unrealistic. For example, data set for evaluate risk of loan application, it seems not fair to assume that between income, age and education levels are equally important. The assumption made by Naïve Bayes harming the performance of classification in reality(Webb, Boughton, & Wang, 2005). This assumption can cause the unwanted error increase.

Many methods have been developed to cover this attribute independence assumption. Jiang(Liangxiao Jiang, Wang, Cai, & Yan, 2007), (L. Jiang, Cai, & Wang, 2010) categorizes solutions to these problems into five: 1) Attribute selection, 2) Local Learning, 3) Attribute Weighting, 4) Instance Weighting and 5) Structure Extension. Previous researchers have proposed many useful methods to evaluate the important attributes. Ratanamahatana use Decision Tree as feature selection on Naïve Bayes(Ratanamahatana & Gunopulos, 2003). Zhang use Gain Ratio to determine attribute weight on Naïve Bayes(Zhang, 2004). Wu use Differential Evolution Algorithm to weighting attribute(J. Wu & Cai, 2011). Decision Tree-based attribute weighting for Naïve Bayes proposed by Hall(M. Hall, 2007). Averaged n-Dependence Estimators (AnDE) was proposed by Webb(Webb, Boughton, Zheng, Ting, & Salem, 2011). AnDE was developed from Averaged One-Dependence estimators (Aode) which reduce the Naive Bayes independence assumption(Webb et al., 2005). Zaidi proposed Weighting attributes to Alleviate Naive Bayes Independence Assumption (WANBIA) by set all weights to a single value(Zaidi, Cerquides, Carman, & Webb, 2013). Taheri proposed Attribute Weighted Naive Bayes (AWNB) which define more than one weight for each attribute(Taheri, Yearwood, Mammadov, & Seifollahi, 2013). AWNB limited to binary classification.

Because of the attribute does not have the same role, some of them more important than the others, one of the ways to develop Naïve Bayes is set a different weight value of each attributes. It is becoming the main idea of the new algorithm called Weighting Naive Bayes, abbreviated WNB, weight value depending on how significant these attributes in the probability, more influential an attribute in probabilities, higher the weight value. For weighting the attribute, this study using correlation coefficients to measure relevancy between class and attributes. Arauzo-Azora has been conducting an empirical study of feature selection method(Arauzo-Azofra, Aznarte, & Benítez, 2011). The study evaluated a broad overview of feature selection methods. For weighting attributes by calculating the relevance between attributes get the highest average rank from all the ways for Naïve Bayes. Correlation coefficient can be used to measure the relevance between attributes(Golub, 1999), (Furey et al., 2000), (Pavlidis, Weston, Cai, & Grundy, 2001). It used to measure the strength of relationship between two attributes(Freund & Wilson, 2003). The value of the correlation coefficients is between +1 and -1. Value of +1 and -1 indicate positive and negative relationship. Because it only requires the strength of the relationship between the attributes then absolute value of the correlation coefficients is used.

In this study, Absolute Correlation Weighted Naïve Bayes has been proposed. Absolute correlation is used as a weight because it shows how strong relevance between attributes. The purpose of this study is to improve the performance of Naïve Bayes for classifying defect-prone on software defect prediction.

2 RELATED WORK

While many studies, including individual study report the performance comparison of modeling techniques, there is no explicit consensus appear that conduct best when distinctive studies that looked at in isolation(T. Hall et al., 2012). Mizuno and Kikuno(Mizuno & Kikuno, 2007) reported, the techniques they learned, Orthogonal Sparse Bigrams Markov models (OSB) are most fit for the defect prediction. Bibi et al.(Bibi, Tsoumakas, Stamelos, & Vlahvas, 2006) reported that Regression via Classification (RVC) works fine. Khoshgoftaar et al.(Khoshgoftaar, Yuan, Allen, Jones, & Hudepohl, 2002) reported that the defect-prone modules predicted as uncertain, can be effectively classified using Tree Disc (TD) technique. Khoshgoftaar and Seliya(Khoshgoftaar & Seliya, 2004) also reported that the Case-Based Reasoning (CBR) did not predict well with C4.5 as well under performing. Arisholm et al.(Arisholm, Briand, & Johannessen, 2010) reported that their comprehensive performance comparison showed there is no difference between predictive modeling techniques they investigated.

A clearer picture appears to arise from the detailed analysis conducted by Hall(T. Hall et al., 2012) on the performance of the model. Hall(T. Hall et al., 2012) findings indicate that actually performance can be associated with modeling techniques that is used. Their comparative analysis showed that studies using Support Vector Machine (SVM) technique appear less well. It probably performed poorly because they need the optimization of parameters where it is uncommon done in the study of defect prediction for best performance(Hsu, Chang, & Lin, 2003). C4.5 model apparently poor performing if they use imbalanced data (Arisholm, Briand, & Fuglerud, 2007; Arisholm et al., 2010). The comparative analysis has been done by Hall also shows that the model performs proportionately correctly are comparatively easy technique that simple to use and rightly understood. Logistic Regression and Naïve Bayes, specifically, appears to be technique that is used in the model are performing relatively well. The model appears to work correctly when the proper techniques have been appropriate for the dataset.

Lin et al.(Lin & Yu, 2011) have a research problem that Naïve Bayes assumption that assumes the value of attributes are independent with other attributes. Lin et al.(Lin & Yu, 2011) proposed PSO-based Weighted Naïve Bayesian Classifier to mitigate this assumption. Particle Swarm Optimization algorithm applied on a weighted naive Bayes classification, through the automatic search behavior of particles, to avoid the mistakes of other methods. The value of the search results by the swarm is used as a weight to each attribute. The research results show that the correct rate-based Weighted Naïve Bayesian Classifier higher than the Naïve Bayes.

Taheri et al.(Taheri et al., 2013) have a research problem that attributes independence assumption created by NB classifier adverse classification performance when, in fact, infringed. On research, Taheri et al.(Taheri et al., 2013) proposed a new attribute weighted Naïve Bayes classifier, called AWNB, which provide more than one weight for every attribute. The results presented show that the accuracy of the proposed method far better compared to Naïve Bayes in every data set(Taheri et al., 2013). It also indicates greater accuracy from AWNB, in general, compared with the results obtained by INB and TAN.

The research issue of Wu et al.(J. Wu & Cai, 2011) is independence assumption made by Naive Bayes that all attributes not related to one another adverse classification performance when it is infringed in reality. In order to weaken the assumption of independent attributes, Wu et al.(J. Wu & Cai, 2011) suggested Different Evolution as a weighting method on Weighted Naïve Bayes. Wu et al.(J. Wu & Cai, 2011) comparing Different Evolution with some other weighting methods and the performance of the Different Evolution is better than other weighting methods.

Naïve Bayes is easy to build, because it has a very simple structure(X. Wu & Kumar, 2009). Learning Naïve Bayes only involves teaching the probability table, to be specific, the conditional probability tables for each attribute, from training examples. This means, the values of the probability $p\ (a_i|c)$ must be determined from the training sample, for each value a_i of attribute A_i considering the value of the variable c on class C.

In Naïve Bayes, it is assumed that the attributes are independent one another provided class(J. Wu & Cai, 2011), (Turhan & Bener, 2009), (L. Jiang et al., 2010). Each attribute only has class variables as its parent(J. Wu & Cai, 2011), (Liangxiao Jiang et al., 2007), $P\ (E|c)$ is calculated by:

$$p(E|c) = p(a_1, a_2, \dots, a_n|c) = \prod_{i=1}^{n} p(a_i|c)$$

Where $p\ (a_i|c)$ is referred the likelihood of A_i, and the instance $E = (a_1, a_2, \dots, a_n)$.

Due to each sample E the value of $p\ (E)$ is constant. The possibility of the establishment of a class label for an example is:

$$p(c|E) = p(c) \prod_{i=1}^{n} p(a_i|c)$$

Example E are classified into the class $C = c'$ if and only if

$$p(c'|E) = \arg\max_{c} p(c|E)$$

More exactly, the classification of which granted by Naïve Bayes, denoted by $V_{nb}(E)$, is defined as follows

$$V_{nb}(E) = \arg\max_{c} p(c) \prod_{i=1}^{n} p(a_i|c)$$

Because of the conditional independence assumption uncommon properly, in reality, it is reasonable to extend the Naïve Bayes to relax the assumption of conditional independence. There are two primary ways to relax the assumption (Zhang, 2004). First, Naïve Bayes structure is extended to explicitly represent dependencies between attributes, and generated model referred to augmented Naïve Bayes (ANB)(Friedman, Geiger, & Goldszmidt, 1997). Second, Attributes are weighted differently, and resultant model is called Weighted Naïve Bayes (WNB). Weighted Naïve Bayes is formally defined as follows.

$$V_{wnb}(E) = \arg\max_{c} p\,(c) \prod_{i=1}^{n} p(a_i|c)^{w_i}$$

Where $V_{wnb}(E)$ shows the classification provided by Weighted Naïve Bayes, and w_i is weight of attribute A_i.

3 PROPOSED METHOD

The correlation coefficient measures the power of linear relationship between two quantitative variables, typically a ratio or interval(Freund & Wilson, 2003). The correlation coefficient has the properties, which are 1) its value is among +1 and -1 inclusively, 2) the values +1 and -1 indicate a positive relationship and negative exact, respectively, among the variables, 3) a correlation of zero shows there is no linear relationship exists between two variables and 4) the correlation coefficient is symmetric to x and y. Thus the size of the power of the linear relationship irrespective of whether x or y is the independent variable.

The correlation coefficient can be defined as follows

$$r = \frac{\sum(x - \bar{x})\,(y - \bar{y})}{\sum\,(x - \bar{x})^2 \sum\,(y - \bar{y})^2}$$

Where r symbolize correlation coefficient, $\bar{x}$ symbolize mean value of x and $\bar{y}$ symbolize mean value of y.

Various correlation coefficients are used as a weighting method(Guyon, Weston, Barnhill, & Vapnik, 2002). The coefficient used in Golub(Golub, 1999) is defined as

$$w_i = \frac{(\mu_i(+) - \mu_i(-))}{(\sigma_i(+) + \sigma_i(-))}$$

Where μ_i and σ_i are the mean and standard deviation of the values of attribute i for all of class $(+)$ or class$(-), i = 1, 2, \ldots, n$, w_i large positive value shows a strong correlation with class $(+)$ while the large negative value of w_i show a strong correlation with class $(-)$. Other people(Furey et al., 2000) have used the absolute value of w_i as a weighting method. Recently, Pavlidis(Pavlidis et al., 2001) has been using the associated coefficients which defined below

$$w_i = \frac{(\mu_i(+) - \mu_i(-))^2}{(\sigma_i(+)^2 + \sigma_i(-)^2)}$$

Zhang(Zhang, 2004) extended the Naïve Bayes to relax the assumption of conditional independence. Zhang(Zhang, 2004) performed Weighted Naïve Bayes. Correlation coefficient is used as a weight because it shows how strong relevance between attributes. Based on the coefficient that used in Golub(Golub, 1999), the proposed method uses the absolute correlation coefficient because it only requires the strength of the relationship between the attributes. This idea is similar with Furey et al.(Furey et al., 2000) that can be defined as follows.

$$w_i = \left|\frac{(\mu_{ij} - \mu_{i\bar{j}})}{(\sigma_{ij} + \sigma_{i\bar{j}})}\right|$$

Where w_i is a weight of attribute i, μ_{ij} is mean values of attribute i for class j, $\mu_{i\bar{j}}$ is mean values of attribute i for class non j, σ_{ij} is standard deviation of the values of attribute i for class j and $\sigma_{i\bar{j}}$ is standard deviation of the values of attribute i for class non j.

The proposed method in study is Absolute Correlation Weighted Naïve Bayes (AC-WNB) that can be described in Figure 1. There are three different processes in AC-WNB model figured with shaded block, which are calculate the weight, calculate the weighted likelihood and calculate prior. Calculate the weight is a process that calculates weight for each attribute in the training process. The weight is calculated the absolute value of correlation coefficient by using the mean and standard deviation of each attribute. Calculate the weighted likelihood is a process that calculates the likelihood for each attribute to classify the testing dataset. These likelihood squares by weight that generated on calculating the weight process. Calculate the prior used weighted likelihood. The weighted likelihood of class was divided by sum of all weighted likelihood.

As shown on Figure 1, dataset is divided into data training and data testing using 10-fold cross validation method. Data training is used to training process. In training process, means, standard deviations and weight of each attributes will be calculated. Weight will be calculated using mean and standard deviation. The absolute value of the difference between mean of class and the other classes that have been divided with the summation of standard deviations is used as weight value.

The detail of training process of AC-WNB as follows:

1. Calculate the mean value of attribute in each class
   The mean value obtained by summing all instances value on a specific attribute then divide by the number of instances. The mean value can be formulated as follows
   $$\mu = \frac{x_1 + x_2 + \cdots + x_n}{n}$$
2. Calculate the standard deviation of attribute in each class
   The standard deviation is the square root value of the variance. The variance is obtained by subtract each value with mean value and square the result then divided by the number of instances. There are two types of standard deviation that are the sample and population. The population divide the square results with the number of instances (N) and the sample divide the square results with the number of instances minus one (N-1). Absolute Correlation based Weighted Naïve Bayes use the sample to calculate standard deviation. The standard deviation can be formulated as follows
   $$\sigma = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2$$
3. Calculate the weight of attribute
   The weight can be obtained by calculation using mean and standard deviation that formulated as follows
   $$w_i = \left|\frac{(\mu_{ij} - \mu_{i\bar{j}})}{(\sigma_{ij} + \sigma_{i\bar{j}})}\right|$$

4. Repeat steps 1-3 for all attribute

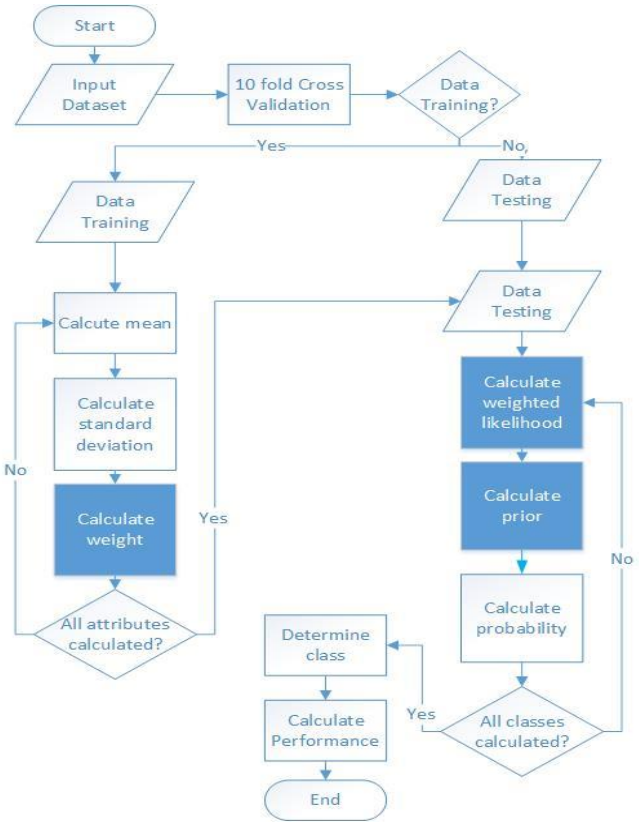

Figure 1 Flow Chart of AC-WNB Method

The model that has been generated in training process will be tested using data testing. AC-WNB has to calculate likelihood, prior, weighted likelihood and weighted prior to classifying the class. Likelihood is commonly known as conditional probabilities that is calculated using normal distribution of each attribute. This normal distribution of each attribute would be squared by weight that has been generated at training process. The product (π) of weighted normal distribution of all attributes is used to weighted likelihood value of class. The prior is also known as class probabilities. The value of prior of class calculated by divided weighted likelihood of class with sum of all weighted likelihood class. Then, weighted likelihood value would be multiplied with prior to classify. The detail of testing process of AC-WNB as follows:

1. Calculate weighted likelihood
   For numeric attribute, likelihood calculates by using normal distribution. In this study, all attributes were numeric. The normal distribution was formulated as follows

$$p(a|c) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

   Then, weighted likelihood can be calculated with formula as follows

$$L = \prod_{i=1}^{n} p(a_i|c)^{w_i}$$

2. Calculate prior
   The prior of class calculated by divided weighted likelihood of class with sum of all weighted likelihood class. The prior can be formulated as follows

$$p(c) = \frac{\prod_{i=1}^{n} p(a_i|c)^{w_i}}{\prod_{i=1}^{n} p(a_i|c)^{w_i} + \prod_{i=1}^{n} p(a_i|\bar{c})^{w_i}}$$

3. Calculate probabilities of class
   In order to calculate probability of class, weighted likelihood value will be multiplied prior. The probabilities of class can be formulated as follows

$$p(c|E) = p(c) \prod_{i=1}^{n} p(a_i|c)$$

4. Repeat steps1-3 for all class
5. The predicted class was determined by the highest probability.

4 DATA GATHERING

NASA dataset that was used in this study was obtained from the MDP (Metric Data Program) Repository ("NASA-SoftwareDefectDataSets," n.d.), which can also be obtained from the PROMISE Repository. NASA datasets have been widely used for research in the field of software engineering(T. Hall et al., 2012). This dataset is devoted to research on the topic of software defect and software failures. Therefore, most studies use this dataset to do some research, ranging from doing predictions, associations, and to the development of a model for research.

NASA dataset currently available from many Repository (MDP and PROMISE), therefore, the researchers used a dataset derived from the MDP Repository who has been repaired by Martin Shepherd(Shepperd, Song, Sun, & Mair, 2013), with the following specifications(Gray, Bowes, Davey, Sun, & Christianson, 2012) in Table 1. All attributes in NASA MDP dataset are numeric. Dataset has been fixed by removing data null or no value. Accordingly, this study used the dataset to do some research in determining the proposed model in the prediction of software defects.

Table 1 Dataset Specifications

Attribute	NASA MDP Dataset									
	CM 1	JM 1	KC 1	KC 3	MC 1	MC 2	PC 1	PC3	PC 4	PC 5
LOC_BLANK	√	√	√	√	√	√	√	√	√	√
BRANCH_COUNT	√	√	√	√	√	√	√	√	√	√
CALL_PAIRS	√			√	√	√	√	√	√	√
LOC_CODE_AND_COMMENT	√	√	√	√	√	√	√	√	√	√
LOC_COMMENTS	√	√	√	√	√	√	√	√	√	√
CONDITION_COUNT	√			√	√	√	√	√	√	√
CYCLOMATIC_COMPLEXITY	√	√	√	√	√	√	√	√	√	√
CYCLOMATIC_DENSITY	√			√	√	√	√	√	√	√
DECISION_COUNT	√			√	√	√	√	√	√	√
DECISION_DENSITY	√			√		√	√	√	√	
DESIGN_COMPLEXITY	√	√	√	√	√	√	√	√	√	√
DESIGN_DENSITY	√			√	√	√	√	√	√	√
EDGE_COUNT	√			√	√	√	√	√	√	√
ESSENTIAL_COMPLEXITY	√	√	√	√	√	√	√	√	√	√
ESSENTIAL_DENSITY	√			√	√	√	√	√	√	√
LOC_EXECUTABLE	√	√	√	√	√	√	√	√	√	√
PARAMETER_COUNT	√			√	√	√	√	√	√	√
GLOBAL_DATA_COMPLEXITY				√	√	√				√
GLOBAL_DATA_DENSITY				√	√	√				√
HALSTEAD_CONTENT	√	√	√	√	√	√	√	√	√	√
HALSTEAD_DIFFICULTY	√	√	√	√	√	√	√	√	√	√
HALSTEAD_EFFORT	√	√	√	√	√	√	√	√	√	√
HALSTEAD_ERROR_EST	√	√	√	√	√	√	√	√	√	√
HALSTEAD_LENGTH	√	√	√	√	√	√	√	√	√	√
HALSTEAD_LEVEL	√	√	√	√	√	√	√	√	√	√
HALSTEAD_PROG_TIME	√	√	√	√	√	√	√	√	√	√
HALSTEAD_VOLUME	√	√	√	√	√	√	√	√	√	√
MAINTENANCE_SEVERITY	√			√	√	√	√	√	√	√
MODIFIED_CONDITION_COUNT	√			√	√	√	√	√	√	√
MULTIPLE_CONDITION_COUNT	√			√	√	√	√	√	√	√
NODE_COUNT	√			√	√	√	√	√	√	√
NORMALIZED_CYLOMATIC_COMPLEXITY	√			√	√	√	√	√	√	√
NUM_OPERANDS	√	√	√	√	√	√	√	√	√	√
NUM_OPERATORS	√	√	√	√	√	√	√	√	√	√
NUM_UNIQUE_OPERANDS	√	√	√	√	√	√	√	√	√	√
NUM_UNIQUE_OPERATORS	√	√	√	√	√	√	√	√	√	√
NUMBER_OF_LINES	√			√	√	√	√	√	√	√
PERCENT_COMMENTS	√			√	√	√	√	√	√	√
LOC_TOTAL	√	√	√	√	√	√	√	√	√	√
Number of Code Attributes	37	21	21	39	38	39	37	37	37	38
Programming Language	C	C	C++	Java	C and C++	C	C	C	C	C++
Number of Modules	327	7782	1183	194	1988	125	705	1077	1287	1711
Number of defect Modules	42	1672	314	36	46	44	61	134	177	471

5 RESULT AND ANALYSIS

The proposed model was developed using Java using NetBeans IDE 7.3.1. Datasets that used in this study are: CM1, JM1, KC1, KC3, MC1, MC2, PC1, PC3, PC4 and PC5. The weight values that calculated by AC-WNB shown in Table 2.

Table 2 Weight of attribute

Attribute	NASA MDP Dataset									
	CM 1	JM 1	KC 1	KC 3	MC 1	MC 2	PC 1	PC 3	PC 4	PC 5
LOC_BLANK	0.18	0.21	0.25	0.34	0.37	0.37	0.38	0.49	0.24	0.15
BRANCH_COUNT	0.19	0.20	0.22	0.25	0.20	0.37	0.19	0.09	0.01	0.23
CALL_PAIRS	0.28			0.30	0.29	0.24	0.24	0.24	0.11	0.21
LOC_CODE_AND_COMMENT	0.04	0.12	0.05	0.38	0.42	0.17	0.32	0.29	0.48	0.28
LOC_COMMENTS	0.35	0.17	0.17	0.10	0.31	0.39	0.44	0.38	0.11	0.19
CONDITION_COUNT	0.19			0.20	0.17	0.38	0.18	0.08	0.22	0.21
CYCLOMATIC_COMPLEXITY	0.19	0.18	0.22	0.26	0.12	0.37	0.20	0.09	0.01	0.23
CYCLOMATIC_DENSITY	0.28			0.19	0.29	0.07	0.52	0.21	0.33	0.22
DECISION_COUNT	0.19			0.20	0.16	0.38	0.17	0.07	0.23	0.20
DECISION_DENSITY	0.03			0.00		0.16	0.01	0.13	0.48	
DESIGN_COMPLEXITY	0.23	0.16	0.22	0.26	0.03	0.31	0.19	0.09	0.06	0.23
DESIGN_DENSITY	0.16			0.03	0.12	0.13	0.05	0.00	0.13	0.06
EDGE_COUNT	0.20			0.28	0.22	0.40	0.20	0.10	0.05	0.22
ESSENTIAL_COMPLEXITY	0.12	0.14	0.15	0.17	0.13	0.36	0.13	0.01	0.14	0.18
ESSENTIAL_DENSITY	0.00			0.05	0.00	0.34	0.01	0.03	0.11	0.11
LOC_EXECUTABLE	0.29	0.21	0.25	0.29	0.24	0.34	0.33	0.13	0.19	0.19
PARAMETER_COUNT	0.07			0.06	0.00	0.14	0.17	0.15	0.14	0.07
GLOBAL_DATA_COMPLEXITY				0.26	0.01	0.34				0.21
GLOBAL_DATA_DENSITY				0.11	0.23	0.06				0.09
HALSTEAD_CONTENT	0.31	0.22	0.24	0.31	0.23	0.11	0.47	0.20	0.11	0.03
HALSTEAD_DIFFICULTY	0.21	0.18	0.30	0.22	0.13	0.11	0.15	0.04	0.15	0.28
HALSTEAD_EFFORT	0.14	0.09	0.22	0.23	0.04	0.38	0.16	0.02	0.13	0.18
HALSTEAD_ERROR_EST	0.26	0.20	0.27	0.26	0.20	0.34	0.29	0.08	0.17	0.13
HALSTEAD_LENGTH	0.27	0.22	0.28	0.27	0.21	0.35	0.31	0.11	0.19	0.16
HALSTEAD_LEVEL	0.27	0.14	0.18	0.21	0.09	0.34	0.24	0.21	0.18	0.30
HALSTEAD_PROG_TIME	0.14	0.09	0.22	0.23	0.04	0.38	0.16	0.02	0.13	0.18
HALSTEAD_VOLUME	0.26	0.20	0.27	0.26	0.20	0.34	0.29	0.08	0.17	0.13
MAINTENANCE_SEVERITY	0.13			0.09	0.01	0.18	0.14	0.16	0.28	0.21
MODIFIED_CONDITION_COUNT	0.19			0.20	0.18	0.37	0.18	0.09	0.22	0.22
MULTIPLE_CONDITION_COUNT	0.19			0.20	0.17	0.38	0.18	0.09	0.22	0.20
NODE_COUNT	0.19			0.28	0.22	0.40	0.19	0.10	0.06	0.21
NORMALIZED_CYLOMATIC_COMPLEXITY	0.32			0.28	0.26	0.13	0.54	0.27	0.37	0.14
NUM_OPERANDS	0.26	0.22	0.28	0.26	0.20	0.35	0.31	0.11	0.18	0.17
NUM_OPERATORS	0.27	0.21	0.27	0.28	0.21	0.34	0.31	0.11	0.18	0.15
NUM_UNIQUE_OPERANDS	0.31	0.23	0.30	0.29	0.31	0.21	0.37	0.24	0.15	0.14
NUM_UNIQUE_OPERATORS	0.34	0.17	0.30	0.26	0.24	0.41	0.33	0.19	0.14	0.34
NUMBER_OF_LINES	0.30			0.30	0.33	0.37	0.43	0.29	0.23	0.20
PERCENT_COMMENTS	0.29			0.22	0.47	0.22	0.38	0.43	0.44	0.10
LOC_TOTAL	0.26	0.22	0.26	0.29	0.28	0.34	0.35	0.16	0.44	0.20

One of the performance indicators to evaluate the performance of the classifier in the experiment, area under the curve (AUC) was applied. AUC used in this study to evaluated the classifier on class imbalance data as recommended by Lessmann et al. (Lessmann, Baesens, Mues, & Pietsch, 2008) and Brown et al.(Brown & Mues, 2012). The results of all the methods would be compared using Wilcoxon signed-rank test to verify whether there is a significant difference between methods.

Table 3 Performance of the NB and AC-WNB Model

Dataset	NB		AC-WNB		AC-WACNB	
	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
CM1	81.04	0.768	81.04	0.814	81.35	0.857
JM1	78.09	0.803	78.17	0.810	78.08	0.813
KC1	72.02	0.793	72.19	0.807	72.19	0.834
KC3	79.38	0.830	79.38	0.879	79.38	0.851
MC1	88.58	0.836	88.98	0.855	89.03	0.833
MC2	72.00	0.840	72.80	0.843	72.00	0.817
PC1	87.66	0.800	87.94	0.817	87.80	0.847
PC3	28.04	0.873	69.73	0.862	70.66	0.890
PC4	86.01	0.797	82.28	0.838	82.13	0.906
PC5	74.63	0.782	74.93	0.805	74.93	0.852

As shown in Table 3, Absolute Correlation based Weighted Naïve Bayes have better average accuracy that is 78.74%, followed by Naïve Bayes with 74.74%, while for AUC values, the average of AUC values of Absolute Correlation Weighted Naïve Bayes is higher than Naïve Bayes. The average of AUC values of Absolute Correlation Weighted Naïve Bayes is 0.833, and then Naïve Bayes is 0.812

The results of the comparison of the AUC can be described in Figure 2. The AUC values of AC-WNB higher than NB. It can be concluded that Absolute Correlation based Weighted Attribute-class Naïve Bayes is much better than others method. However, this increase should be examined more deeply with significance test.

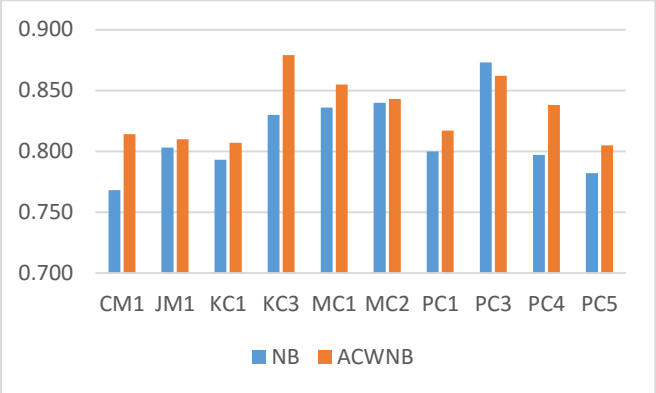

Figure 2 Performance (AUC) of the Models

AUC values of Naïve Bayes and Absolute Correlation based Weighted Naïve Bayes would be compared using Wilcoxon signed-rank test. A significant different in performance was considered when the results of Wilcoxon signed-rank test showed that P-value<alpha (0.05). Wilcoxon signed-rank test results on the statistical test of AUC of Naïve Bayes and Absolute Correlation based Weighted Naïve Bayes was shown in Table 4. The average of AUC values for AC-WNB was higher than NB that is 0.833 with P-value 0.01. As the computed P-value was lower than the significance level alpha, the null hypothesis (H0) that is the two samples follow the same distribution should be rejected and accept the alternative hypothesis (Ha) that is the distributions of the two samples are different. It means that NB and AC-WNB had significant differences P-value<alpha (0.05). Therefore, it can be concluded that AC-WNB makes an improvement when compared with NB in prediction performance. Hence, it means that the absolute value of correlation coefficient can improve the performance of Naïve Bayes for classifying on software defect prediction

Table 4 Wilcoxon Signed-Rank Test of AUC of NB and AC-WNB

	NB	*AC-WNB*
Observation	10	10
Mean	0.8122	0.833
Median	0.8015	0.8275
Standard Deviation	0.030085877	0.02484351
The Test Procedure		
Hypothetical Mean Difference	0	
Nb. Of Zero Differences	0	
Rank Sum	55	
Rank Average	5.5	
Test Statistic (S+)	52	
Significance Level	0.05	
Exact Procedure		
Critical Value	8	
Decision Rule	Reject H0 if (S+) > 8	
Final Decision	The Null Hypothesis Cannot be Rejected due to Insufficient Evidence in the Sample	
P-Value	0.01	

6 CONCLUSION

Naive Bayes proved effective in predicting software defects. However, Naïve Bayes perform less well for predicting software defects due to the assumption that all attributes are equally important and are not related to each other while, in fact, this assumption is not true in many cases. One of the ways to develop Naïve Bayes is set a different weight value of each attributes.

Absolute Correlation Weighted Naïve Bayes has been proposed. Absolute Correlation Weighted Naïve Bayes provides weight to each attribute. The weight values are depending on how relevant the attribute with class. The correlation coefficient is used to measure the relevance between class and attribute. Therefore, absolute value of correlation coefficients was used as weight at Absolute Correlation Weighted Naïve Bayes.

The results of experiments showed that the mean of AUC value of Naïve Bayes for classifying software defect was 0.8122, while the average AUC value of Absolute Correlation Weighted Naïve Bayes was 0.833. The results of Wilcoxon signed-rank test showed that Absolute Correlation Weighted Naïve Bayes had significant differences with Naïve Bayes P-value 0.008 < alpha (0.05). Therefore, it can be concluded that absolute correlation coefficient can improve the performance of Naïve Bayes for classifying on software defect prediction.

7 FUTURE WORK

In this study, absolute coefficient correlation could improve the performance of Naive Bayes for classifying on software defect prediction. Absolute correlation was used as weight for attribute by calculated the relevance between attribute. It used to measure the strength of relationship between two attributes(Freund & Wilson, 2003). Taheri et al.(Taheri et al., 2013) proposed a new attribute weighted Naïve Bayes classifier, called AWNB, which provide more than one weight for every attribute. Therefore, investigation to improve the performance of Naive Bayes for classifying on software defect prediction by using absolute correlation to provide more than one weight is one of main direction for future work.

REFERENCES

Arauzo-Azofra, A., Aznarte, J. L., & Benítez, J. M. (2011). Empirical study of feature selection methods based on individual feature evaluation for classification problems. *Expert Systems with Applications*, *38*(7), 8170–8177. doi:10.1016/j.eswa.2010.12.160

Arisholm, E., Briand, L. C., & Fuglerud, M. (2007). Data Mining Techniques for Building Fault-proneness Models in Telecom Java Software. In *The 18th IEEE International Symposium on Software Reliability (ISSRE '07)* (pp. 215–224). IEEE. doi:10.1109/ISSRE.2007.22

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

Bibi, S., Tsoumakas, G., Stamelos, I., & Vlahvas, I. (2006). Software Defect Prediction Using Regression via Classification. In *IEEE International Conference on Computer Systems and Applications, 2006.* (pp. 330–336). IEEE. doi:10.1109/AICCSA.2006.205110

Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, *39*(3), 3446–3453. doi:10.1016/j.eswa.2011.09.033

Freund, R. J., & Wilson, W. J. (2003). *Statistical Methods* (2nd ed.). Academic Press.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, *29*(2-3), 131–163.

Furey, T. S., Cristianini, N., Duffy, N., Bednarski, D. W., Schummer, M., & Haussler, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, *16*(10), 906–914. doi:10.1093/bioinformatics/16.10.906

Golub, T. R. (1999). Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, *286*(5439), 531–537. doi:10.1126/science.286.5439.531

Gray, D., Bowes, D., Davey, N., Sun, Y., & Christianson, B. (2012). Reflections on the NASA MDP data sets. *IET Software*, *6*(6), 549. doi:10.1049/iet-sen.2011.0132

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, *46*(1-3), 389–422. doi:10.1023/A:1012487302797

Hall, M. (2007). A decision tree-based attribute weighting filter for naive Bayes. *Knowledge-Based Systems*, *20*(2), 120–126. doi:10.1016/j.knosys.2006.11.008

Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012). A Systematic Literature Review on Fault Prediction Performance in Software Engineering. *IEEE Transactions on Software Engineering*, *38*(6), 1276–1304. doi:10.1109/TSE.2011.103

Harrington, P. (2012). *Machine Learning in Action*. Connecticut: Manning Publications Co. Greenwich, CT, USA.

Hsu, C., Chang, C., & Lin, C. (2003). A Practical Guide to Support Vector Classification. *Department of Computer Science and Information Engineering, National Taiwai University*.

Jiang, L. (2011). Random one-dependence estimators. *Pattern Recognition Letters*, *32*(3), 532–539. doi:10.1016/j.patrec.2010.11.016

Jiang, L., Cai, Z., & Wang, D. (2010). Improving Naive Bayes for Classification. *International Journal of Computers and Applications*, *32*(3). doi:10.2316/Journal.202.2010.3.202-2747

Jiang, L., Wang, D., Cai, Z., & Yan, X. (2007). Survey of improving naive Bayes for classification. *Advanced Data Mining and Applications*. doi:10.1007/978-3-540-73871-8_14

Khoshgoftaar, T. M., & Seliya, N. (2004). Comparative Assessment of Software Quality Classification Techniques: An Empirical Case Study. *Empirical Software Engineering*, *9*(3), 229–257. doi:10.1023/B:EMSE.0000027781.18360.9b

Khoshgoftaar, T. M., Yuan, X., Allen, E. B., Jones, W. D., & Hudepohl, J. P. (2002). Uncertain Classification of Fault-Prone Software Modules. *Empirical Software Engineering*, *7*(4), 297–318. doi:10.1023/A:1020511004267

Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings. *IEEE Transactions on Software Engineering*, *34*(4), 485–496. doi:10.1109/TSE.2008.35

Lin, J., & Yu, J. (2011). Weighted Naive Bayes classification algorithm based on particle swarm optimization. In *2011 IEEE 3rd International Conference on Communication Software and Networks* (pp. 444–447). IEEE. doi:10.1109/ICCSN.2011.6014307

Mizuno, O., & Kikuno, T. (2007). Training on errors experiment to detect fault-prone software modules by spam filter. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering - ESEC-FSE '07* (p. 405). New York, New York, USA: ACM Press. doi:10.1145/1287624.1287683

NASA-SoftwareDefectDataSets. (n.d.). Retrieved from http://nasa-softwaredefectdatasets.wikispaces.com/

Okutan, A., & Yıldız, O. T. (2012). Software defect prediction using Bayesian networks. *Empirical Software Engineering*, *19*(1), 154–181. doi:10.1007/s10664-012-9218-8

Pavlidis, P., Weston, J., Cai, J., & Grundy, W. N. (2001). Gene functional classification from heterogeneous data. In

Proceedings of the fifth annual international conference on Computational biology - RECOMB '01 (pp. 249–255). New York, New York, USA: ACM Press. doi:10.1145/369133.369228

Ratanamahatana, C., & Gunopulos, D. (2003). Feature selection for the naive bayesian classifier using decision trees. *Applied Artificial Intelligence*, 475–487. doi:10.1080/08839510390219327

Shepperd, M., Song, Q., Sun, Z., & Mair, C. (2013). Data Quality: Some Comments on the NASA Software Defect Datasets. *IEEE Transactions on Software Engineering*, *39*(9), 1208–1215. doi:10.1109/TSE.2013.11

Song, Q., Jia, Z., Shepperd, M., Ying, S., & Liu, J. (2011). A General Software Defect-Proneness Prediction Framework. *IEEE Transactions on Software Engineering*, *37*(3), 356–370. doi:10.1109/TSE.2010.90

Taheri, S., Yearwood, J., Mammadov, M., & Seifollahi, S. (2013). Attribute weighted Naive Bayes classifier using a local optimization. *Neural Computing and Applications*. doi:10.1007/s00521-012-1329-z

Turhan, B., & Bener, A. (2009). Analysis of Naive Bayes' assumptions on software fault data: An empirical study. *Data & Knowledge Engineering*, *68*(2), 278–290. doi:10.1016/j.datak.2008.10.005

Webb, G. I., Boughton, J. R., & Wang, Z. (2005). Not So Naive Bayes: Aggregating One-Dependence Estimators. *Machine Learning*, *58*(1), 5–24. doi:10.1007/s10994-005-4258-6

Webb, G. I., Boughton, J. R., Zheng, F., Ting, K. M., & Salem, H. (2011). Learning by extrapolation from marginal to full-multivariate probability distributions: decreasingly naive Bayesian classification. *Machine Learning*, *86*(2), 233–272. doi:10.1007/s10994-011-5263-6

Wu, J., & Cai, Z. (2011). Attribute weighting via differential evolution algorithm for attribute weighted naive bayes (WNB). *Journal of Computational Information Systems*, *7*(12), 1672–1679.

Wu, X., & Kumar, V. (2009). *The top ten algorithms in data mining*. *International Statistical Review* (Vol. 78, pp. 158–158). Taylor & Francis Group.

Zaidi, N., Cerquides, J., Carman, M., & Webb, G. (2013). Alleviating Naive Bayes Attribute Independence Assumption by Attribute Weighting. *Journal of Machine Learning Research*, *14*, 1947–1988.

Zhang, H. (2004). Learning Weighted Naive Bayes with Accurate Ranking. In *Fourth IEEE International Conference on Data Mining (ICDM'04)* (pp. 567–570). IEEE. doi:10.1109/ICDM.2004.10030

Arauzo-Azofra, A., Aznarte, J. L., & Benítez, J. M. (2011). Empirical study of feature selection methods based on individual feature evaluation for classification problems. *Expert Systems with Applications*, *38*(7), 8170–8177. doi:10.1016/j.eswa.2010.12.160

Arisholm, E., Briand, L. C., & Fuglerud, M. (2007). Data Mining Techniques for Building Fault-proneness Models in Telecom Java Software. In *The 18th IEEE International Symposium on Software Reliability (ISSRE '07)* (pp. 215–224). IEEE. doi:10.1109/ISSRE.2007.22

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

Bibi, S., Tsoumakas, G., Stamelos, I., & Vlahvas, I. (2006). Software Defect Prediction Using Regression via Classification. In *IEEE International Conference on Computer Systems and Applications, 2006.* (pp. 330–336). IEEE. doi:10.1109/AICCSA.2006.205110

Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, *39*(3), 3446–3453. doi:10.1016/j.eswa.2011.09.033

Freund, R. J., & Wilson, W. J. (2003). *Statistical Methods* (2nd ed.). Academic Press.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, *29*(2-3), 131–163.

Furey, T. S., Cristianini, N., Duffy, N., Bednarski, D. W., Schummer, M., & Haussler, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, *16*(10), 906–914. doi:10.1093/bioinformatics/16.10.906

Golub, T. R. (1999). Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, *286*(5439), 531–537. doi:10.1126/science.286.5439.531

Gray, D., Bowes, D., Davey, N., Sun, Y., & Christianson, B. (2012). Reflections on the NASA MDP data sets. *IET Software*, *6*(6), 549. doi:10.1049/iet-sen.2011.0132

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, *46*(1-3), 389–422. doi:10.1023/A:1012487302797

Hall, M. (2007). A decision tree-based attribute weighting filter for naive Bayes. *Knowledge-Based Systems*, *20*(2), 120–126. doi:10.1016/j.knosys.2006.11.008

Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012). A Systematic Literature Review on Fault Prediction Performance in Software Engineering. *IEEE Transactions on Software Engineering*, *38*(6), 1276–1304. doi:10.1109/TSE.2011.103

Harrington, P. (2012). *Machine Learning in Action*. Connecticut: Manning Publications Co. Greenwich, CT, USA.

Hsu, C., Chang, C., & Lin, C. (2003). A Practical Guide to Support Vector Classification. *Department of Computer Science and Information Engineering, National Taiwai University*.

Jiang, L. (2011). Random one-dependence estimators. *Pattern Recognition Letters*, *32*(3), 532–539. doi:10.1016/j.patrec.2010.11.016

Jiang, L., Cai, Z., & Wang, D. (2010). Improving Naive Bayes for Classification. *International Journal of Computers and Applications*, *32*(3). doi:10.2316/Journal.202.2010.3.202-2747

Jiang, L., Wang, D., Cai, Z., & Yan, X. (2007). Survey of improving naive Bayes for classification. *Advanced Data Mining and Applications*. doi:10.1007/978-3-540-73871-8_14

Khoshgoftaar, T. M., & Seliya, N. (2004). Comparative Assessment of Software Quality Classification Techniques: An Empirical Case Study. *Empirical Software Engineering*, *9*(3), 229–257. doi:10.1023/B:EMSE.0000027781.18360.9b

Khoshgoftaar, T. M., Yuan, X., Allen, E. B., Jones, W. D., & Hudepohl, J. P. (2002). Uncertain Classification of Fault-Prone Software Modules. *Empirical Software Engineering*, *7*(4), 297–318. doi:10.1023/A:1020511004267

Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings. *IEEE Transactions on Software Engineering*, *34*(4), 485–496. doi:10.1109/TSE.2008.35

Lin, J., & Yu, J. (2011). Weighted Naive Bayes classification algorithm based on particle swarm optimization. In *2011 IEEE 3rd International Conference on Communication Software and Networks* (pp. 444–447). IEEE. doi:10.1109/ICCSN.2011.6014307

Mizuno, O., & Kikuno, T. (2007). Training on errors experiment to detect fault-prone software modules by spam filter. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering - ESEC-FSE '07* (p. 405). New York, New York, USA: ACM Press. doi:10.1145/1287624.1287683

NASA-SoftwareDefectDataSets. (n.d.). Retrieved from http://nasa-softwaredefectdatasets.wikispaces.com/

Okutan, A., & Yıldız, O. T. (2012). Software defect prediction using Bayesian networks. *Empirical Software Engineering*, *19*(1), 154–181. doi:10.1007/s10664-012-9218-8

Pavlidis, P., Weston, J., Cai, J., & Grundy, W. N. (2001). Gene functional classification from heterogeneous data. In *Proceedings of the fifth annual international conference on Computational biology - RECOMB '01* (pp. 249–255). New York, New York, USA: ACM Press. doi:10.1145/369133.369228

Ratanamahatana, C., & Gunopulos, D. (2003). Feature selection for the naive bayesian classifier using decision trees. *Applied Artificial Intelligence*, 475–487. doi:10.1080/08839510390219327

Shepperd, M., Song, Q., Sun, Z., & Mair, C. (2013). Data Quality: Some Comments on the NASA Software Defect Datasets. *IEEE Transactions on Software Engineering*, *39*(9), 1208–1215. doi:10.1109/TSE.2013.11

Song, Q., Jia, Z., Shepperd, M., Ying, S., & Liu, J. (2011). A General Software Defect-Proneness Prediction Framework. *IEEE Transactions on Software Engineering*, *37*(3), 356–370. doi:10.1109/TSE.2010.90

Taheri, S., Yearwood, J., Mammadov, M., & Seifollahi, S. (2013). Attribute weighted Naive Bayes classifier using a local optimization. *Neural Computing and Applications*. doi:10.1007/s00521-012-1329-z

Turhan, B., & Bener, A. (2009). Analysis of Naive Bayes' assumptions on software fault data: An empirical study. *Data & Knowledge Engineering*, *68*(2), 278–290. doi:10.1016/j.datak.2008.10.005

Webb, G. I., Boughton, J. R., & Wang, Z. (2005). Not So Naive Bayes: Aggregating One-Dependence Estimators. *Machine Learning*, *58*(1), 5–24. doi:10.1007/s10994-005-4258-6

Webb, G. I., Boughton, J. R., Zheng, F., Ting, K. M., & Salem, H. (2011). Learning by extrapolation from marginal to full-multivariate probability distributions: decreasingly naive Bayesian classification. *Machine Learning*, *86*(2), 233–272. doi:10.1007/s10994-011-5263-6

Wu, J., & Cai, Z. (2011). Attribute weighting via differential evolution algorithm for attribute weighted naive bayes (WNB). *Journal of Computational Information Systems*, *7*(12), 1672–1679.

Wu, X., & Kumar, V. (2009). *The top ten algorithms in data mining*. *International Statistical Review* (Vol. 78, pp. 158–158). Taylor & Francis Group.

Zaidi, N., Cerquides, J., Carman, M., & Webb, G. (2013). Alleviating Naive Bayes Attribute Independence Assumption by Attribute Weighting. *Journal of Machine Learning Research*, *14*, 1947–1988.

Zhang, H. (2004). Learning Weighted Naive Bayes with Accurate Ranking. In *Fourth IEEE International Conference on Data Mining (ICDM'04)* (pp. 567–570). IEEE. doi:10.1109/ICDM.2004.10030

BIOGRAPHY OF AUTHORS

Rizky Tri Asmono Received bachelor degree and master degree in Computer Science in 2012 and 2014 from Dian Nuswantoro University, Indonesia. He is an IT Executive at PT. Sree International Indonesia in Indonesia. His current research interests are Software Engineering, Data Mining and Machine Learning.

Romi Satria Wahono. Received B.Eng and M.Eng degrees in Computer Science respectively from Saitama University, Japan, and Ph.D in Software Engineering from Universiti Teknikal Malaysia Melaka. He is a lecturer at the Faculty of Computer Science, Dian Nuswantoro University, Semarang, Indonesia. He is also a founder and chief executive officer of PT Brainmatics Cipta Informatika, a software development company in Indonesia. His current research interests include software engineering and machine learning. Professional member of the ACM, PMI and IEEE Computer Society

Abdul Syukur. Received bachelor degree in Mathematics from Universitas Diponegoro Semarang, master degree in management from from Universitas Atma Jaya Yogyakarta, and doctoral degree in economic from Universitas Merdeka Malang. He is a lecturer and a dean at the Faculty of Computer Science, Dian Nuswantoro University, Semarang, Indonesia. His current research interests include decision support systems and information management systems.