Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 7, July 2018, pp. 1468-1480
DOI: 10.23940/ijpe.18.07.p10.14681480

A Two-Stage Feature Weighting Method for Naive Bayes and Its Application in Software Defect Prediction

Haijin Ji[a,b], Song Huang[a,*], Xuewei Lv[a,b], Yaning Wu[a], and Zhanwei Hui[a]

[a]*Command and Control Engineering College, Army Engineering University of PLA, Nanjing, 210007, China*
[b]*School of Computer Science and Technology, Huaiyin Normal University, Huai'an, 223300, China*

Abstract

Software defect prediction (SDP) models facilitate software practitioners to find out defect-prone software modules in software. Software practitioners can then test these defect-prone software modules with limited testing resources to minimize software defects. Among various SDP models, Naive Bayes (NB) has been widely used in SDP because of its simplicity, effectiveness and robustness. The NB classifier is an effective classification approach, especially for data sets with discrete attributes. In NB, the attributes are assumed to be independent and thus equally important. However, in common practice, the attributes of software defect data sets are usually continuous or numeric, and because they are designed for different purposes, their contributions to prediction are different. Therefore, this paper proposes a new NB method called TSWNB, which contains two stages: feature (i.e. attribute) discretization and feature weighting. More specifically, for the stage of feature discretization, we make the comparison between two discretization methods i.e. equal-width discretization method and equal-frequency discretization method, and identify the most appropriate one. For the stage of feature weighting, we use the feature weighting technique to alleviate the equal importance assumption, which combines the obtained feature weights into the NB formula and its likelihood estimations. To evaluate the proposed method, we carry out experiments on 5 software defect data sets of NASA MDP provided by PROMISE repository. Three well-known classification algorithms and two feature weighting techniques are included for comparison. The experimental results reveal the effectiveness and practicability of the two-stage feature weighting method TSWNB.

Keywords: software defect prediction; Naive Bayes; feature discretization; feature weighting; PROMISE repository

(Submitted on April 5, 2018; Revised on May 23, 2018; Accepted on June 20, 2018)

1. Introduction

Software quality is considered to be determined by the number of software defects. The cost of these software defects is very large per year, and the testing resources are very limited in many software organizations 1. Therefore, early prediction of software defects is very meaningful to software organizations. In the field of software testing, software defect prediction (SDP) is used to accurately estimate defective software modules and facilitates software practitioners to assign valuable testing resources to modules that are more likely to be defective 2. Therefore, the main task of SDP models is to classify software modules into two categories, i.e. defective and non-defective. Then, practitioners can test these defect-prone software modules with limited testing resources to minimize software defects.

Among various SDP models, Naive Bayes (NB) has been widely used in SDP 3. NB is an exceptionally simple and very effective method of classification problem 4. Compared with complex classification models such as rule learning, decision trees, and instance-based learning, the prediction accuracy of NB is very well because of its simplicity, effectiveness and robustness, especially when the data size is small [4, 5].

The NB classifier is an effective classification approach, especially for data sets with discrete attributes 6. When the attributes of a dataset are continuous or numeric, they are discretized or presumed to have a normal distribution. The discretization methods generally yield better classification performance when the continuous or numeric attributes are not

* Corresponding author.
E-mail address: songhuangabc@hotmail.com

normally distributed 7. In common practice, the attributes of software defect data sets are usually continuous or numeric. Therefore, it is necessary to discretize these attributes.

The NB classifier is a simple statistical technique on basis of the presumption that attributes are independent and equally important; in many cases, this is not the case 8. However, in common practice, because they are designed for different purposes, their contributions to classification (i.e. prediction) are different. Therefore, it is not appropriate to treat the attributes equally.

Aiming at addressing the two issues, this study proposes a new NB method called TSWNB, which contains two stages: feature (i.e. attribute) discretization and feature weighting. More specifically, for the stage of feature discretization, we make the comparison between two discretization methods i.e. equal-width discretization method and equal-frequency discretization methods, and identify the most appropriate one. For the stage of feature weighting, we use the feature weighting technique to alleviate the equal importance assumption, which combines the obtained feature weights into the NB formula and its likelihood estimations. To validate the performance of TSWNB, we design experiments to address the three research questions below.

RQ1. Between equal-width and equal-frequency discretization, which method is better for TSWNB?

RQ2. What is the defect prediction performance of TSWNB when compared with the classic methods?

RQ3. What is the performance of TSWNB when compared with other weighted NB methods?

The remainder of this article is organized as follows. Section 2 describes the related work, and Section 3 gives the proposed method TSWNB. The detailed experimental setups are illustrated in Section 4. The experimental results are analyzed in Section 5. At last, we present the conclusion of this article in Section 6.

2. Related Work

2.1. Software Defect Prediction

In the field of software testing, SDP has been studied by many researchers for about 30 years. Current SDP work can be divided into two categories: 1) identifying the number of defects of the software modules, and 2) classifying the software modules as two categories, i.e. defective and non-defective. In this article, we attempt to solve some of the problems in the second category.

In the first type of work, many methods have been employed to assess the number of defects of software modules, such as statistical approaches 9, capture-recapture models 10, and ensemble learning methods 11. The results of defect prediction can provide guidance for software testing; furthermore, the delivered quality of a release can be gauged.

The second category of work can divide the software modules into defective and non-defective based on the metrics 12. The most widely used metrics are Halstead's software science measures, McCabe's cyclomatic complexity and lines of code counts 22. Therefore, the defect prediction is often treated as the learning problem in the second type of work. Various machine learning approaches have been applied to SDP, such as Naive Bayes [3, 13], Logistic Regression (LR) 14 and Random Tree (RT) 24.

2.2. Naive Bayes

The NB classifier is a simple statistical technique. It is good at dealing with discrete data while the attributes of software defect data sets are usually continuous or numeric 6. The attributes are assumed to be independent and equally important, which is incorrect in many cases 8. In order to solve these two problems, many researchers proposed their solutions.

There are many discretization approaches that can be used to improve the performance of NB. These discretization approaches can be classified into two kinds: unsupervised and supervised, depending on whether the class attribute is considered for discretization 15. Among these discretization approaches, the unsupervised discretization approaches of equal-width and equal-frequency are the most popular discretization approaches. The equal-width discretization approach splits a continuous or numeric attribute into the same width intervals while equal-frequency discretization method splits a continuous or numeric attribute into the same frequency ones 16.

Since attributes are designed for different purposes, they contribute differently to classification (or defect prediction). In order to address this issue, many feature (i.e. attribute) weighting approaches have been used to improve the prediction accuracy of NB. For example, in the literature 17, the features are assigned higher weights when they have the higher gain ratio (GR) in the feature weighted Naive Bayes. In addition, feature selection is a particular kind of feature weighting technique. In this approach, the weights of those selected features are set to 1 while the weights of those features that are not selected are set to 0, such as correlation-based feature selection (CFS) 18. However, all these feature weighting methods just simply apply the obtained feature weights to the formula of NB, yet do not apply the obtained feature weights to their likelihood estimations. The literature 19 proposed a new feature weighting method for text classification, which estimates the likelihood of NB by computing the weighted frequencies based on training data and obtained a better performance than the standard Naive Bayes in the field of text classification. In this article, we employ this weighting technique to our proposed method TSWNB in SDP.

3. Proposed Methodology

This section presents the proposed method TSWNB, i.e. the two-stage feature weighting method for Naive Bayes that combines feature discretization methods and a novel weighting technique to improve the performance of NB for SDP. More specifically, for the stage of feature discretization, we make the comparison between two broadly used discretization approaches i.e. equal-width discretization method and equal-frequency discretization approaches. We identify the most appropriate one for TSWNB. For the stage of feature weighting, we use the feature weighting technique to alleviate the equal importance assumption, which combines the obtained feature weights into the NB formula and its likelihood estimations.

3.1. The First Stage: Feature Discretization

In this stage, we apply the discretization methods to discrete the continuous attributes (i.e. features). Because the NB classifier is good at dealing with discrete data while the attributes of software defect data sets are usually continuous or numeric 6, the unsupervised discretization approaches equal-width and equal-frequency are the most popular discretization methods. Therefore, we make the comparison between these two discretization methods to identify the most appropriate discretization for TSWNB.

3.2. The Second Stage: Feature Weighting

In the second stage, we use the feature weighting technique to overcome the equal important assumption of NB by increasing the weights of highly predictive features and decreasing the weights of less predictive features. In TSWNB, the value of weight is measured by the gain ratio (GR) 17. Unlike previous weighted Naive Bayes method that only applies the obtained feature weights to the NB formula, we also combine the obtained feature weights into its likelihood estimations. We will describe the details of TSWNB in the following subsections, referring to Jiang et al. 19.

3.2.1. Weighted Naive Bayes

According to Bayes theorem 20, the posterior probability of an instance is proportional to the likelihood and the prior probability of this instance. Formally:

$$P(C_i \mid x) = \frac{P(x \mid C_i)P(C_i)}{P(x)} \tag{1}$$

where x refers to the instance (i.e. a software module in SDP), C_i is the category of the instance x (i.e. C_0 represents a non-defective software module and C_1 is the defective software module in this paper).

In equation (1), the denominator $P(x)$ can be computed as follow:

$$P(x) = \sum_i P(x \mid C_i)P(C_i) \tag{2}$$

The denominator $P(x)$ is the same for the two categories; thus, it can be discarded. Equation (1) becomes:

$$P(C_i \mid x) = P(x \mid C_i)P(C_i) \tag{3}$$

The prior probability $P(C_i)$ in equation (1) is computed using the below equation:

$$P(C_i) = \frac{\sum_{j=1}^{m} \delta(C_j, C_i) + 1}{m + 2} \tag{4}$$

where m is the amount of training software modules. The value of $\delta(\bullet)$ is 1 if the value of C_j and C_i are the same, otherwise, the value of $\delta(\bullet)$ is 0.

In a classification problem, the category with higher posterior probability $P(C_i \mid x)$ is the category of the software module. Because NB is on basis of the presumption that attributes are independent and equally important, equation (3) becomes:

$$P(C_i \mid x) = P(C_i)\prod_{d=1}^{n} P(x_d \mid C_i) \tag{5}$$

where x_d refers to the *dth* feature of the software module x, n is the number of features of software module x. Since each feature contributes to classification differently, weight w_d for each feature is introduced to alleviate this assumption. Therefore, the formula of weighted Naive Bayes is defined as:

$$P(C_i \mid x) = P(C_i)\prod_{d=1}^{n} P(x_d \mid C_i)^{w_d} \tag{6}$$

The likelihood $P(x_d \mid C_i)$ in equation (6) is computed using equation (7):

$$P(x_d \mid C_i) = \frac{\sum_{j=1}^{m} \delta(x_{jd}, x_d)\delta(C_j, C_i) + 1}{\sum_{j=1}^{m} \delta(C_j, C_i) + n_d} \tag{7}$$

where n_d denotes the number of different values of the *dth* feature of software module x, x_{jd} is the *dth* feature of *jth* software module in the training software modules.

From equation (7), we can see that the feature weight w_d is not included in the computation of likelihood $P(x_d \mid C_i)$. According to Jiang et al. 19, we apply the feature weight w_d to the computation of likelihood $P(x_d \mid C_i)$ for SDP. Therefore, equation (7) can be written as:

$$P(x_d \mid C_i) = \frac{\sum_{j=1}^{m} w_d \delta(x_{jd}, x_d)\delta(C_j, C_i) + 1}{\sum_{j=1}^{m} w_d \delta(C_j, C_i) + n_d} \tag{8}$$

where the feature weight w_d is learned by GR.

3.2.2. Gain Ratio

Gain ratio (GR) was first applied in the decision tree to choose the best attribute that has the best classification performance among a set of attributes 21. In this paper, we use GR to learn the feature (i.e. attribute) weights. In other words, an attribute with a higher GR deserves higher weight. Therefore, as suggested by Zhang et al. 17, the feature weight w_d can be defined as follows:

$$w_d = \frac{GR(Td, x_d) \times n}{\sum_{d=1}^{n} GR(Td, x_d)} \tag{9}$$

where n refers to the number of attributes, Td refers to the training dataset, x_d refers to the *dth* feature of the software module x. The GR of x_d is defined as 21.

4. Experimental Setup

In the experimental setup, experiments are designed based on three research questions to verify the effectiveness and practicality of the two-stage feature weighting method for Naive Bayes TSWNB. First, we present the software defect data sets employed in our experiments, 10 × 10 – fold cross-validation method and performance measures. Then, we introduce the experimental design for TSWNB.

4.1. Data Collection

The 5 software defect data sets from NASA MDP provided by PROMISE repository[1] employed in our experiments are shown in Table 1. This table lists the number of attributes, the number of software modules and the defect rate for each software defect data set. All these software defect data sets have 38 attributes (37 software metric attribute and 1 class attribute). The 38 attributes are shown in Table 2. The details of these attributes are illustrated in the literature 22.

Table 1. Data sets for software defect prediction

Name	#Attributes	#Modules	#Defect Rate (%)
CM1	38	327	12.8
MW1	38	253	10.7
PC1	38	705	8.7
PC3	38	1563	10.2
PC4	38	1287	13.8

Table 2. The 38 attributes

Attributes	Attributes
LOC_BLANK	HALSTEAD_EFFORT
BRANCH_COUNT	HALSTEAD_ERROR_EST
CALL_PAIRS	HALSTEAD_LENGTH
LOC_CODE_AND_COMMENT	HALSTEAD_LEVEL
LOC_COMMENTS	HALSTEAD_PROG_TIME
CONDITION_COUNT	HALSTEAD_VOLUME
CYCLOMATIC_COMPLEXITY	MAINTENANCE_SEVERITY
CYCLOMATIC_DENSITY	MODIFIED_CONDITION_COUNT
DECISION_COUNT	MULTIPLE_CONDITION_COUNT
DECISION_DENSITY	NODE_COUNT
DESIGN_COMPLEXITY	NORMALIZED_CYLOMATIC_COMPLEXITY
DESIGN_DENSITY	NUM_OPERANDS
EDGE_COUNT	NUM_OPERATORS
ESSENTIAL_COMPLEXITY	NUM_UNIQUE_OPERANDS
ESSENTIAL_DENSITY	NUM_UNIQUE_OPERATORS
LOC_EXECUTABLE	NUMBER_OF_LINES
PARAMETER_COUNT	PERCENT_COMMENTS
HALSTEAD_CONTENT	LOC_TOTAL
HALSTEAD_DIFFICULTY	Defective

4.2. 10 × 10 – Fold Cross-validation Method

In our experiments, the 10 × 10 – fold cross-validation method is used. According to the 10 × 10-fold cross validation, the data set is divided into 10 portions. Each portion will be the test set once and the others will be the train set. Then, the above process is repeated 10 times. Therefore, the mean of 100 experimental results is used for each method in our experiments.

[1] http://openscience.us/repo/

4.3. Performance Measures

In this article, we employ *Precision*, *Recall* and *F-measure* as the performance measures. Equation (10) and (11) give the formal definitions of these performance criteria, which are evolved from the confusion matrix shown in Table 3. *Precision* is the correctly detected defective modules over the total amount of modules predicted as defective modules. *Recall* is the correctly detected defective modules over the total amount of defective modules. *F-measure* is the weighted average of *Precision* and *Recall*. Therefore, the higher the values of these criteria is better.

In addition, Area Under the Curve (AUC) is also employed to evaluate the performance of classifiers. AUC calculates the area under an ROC curve, which illustrates relative trade-offs between True Positive Rate (TPR) and False Positive Rate (FPR) for each threshold of a classification. *TPR* and *FPR* are also evolved from the confusion matrix shown in Table 3. Equation (12) is the definitions of them. Higher *TPR* and lower *FPR* are desired. Since AUC is the area under ROC, the higher the AUC value, the better the classifier.

Table 3. Confusion matrix

	predicted	
Real	Defective	Non-defective
Defective	A	C
Non-defective	B	D

$$Precision = \frac{A}{A+B} \quad Recall = \frac{A}{A+C} \tag{10}$$

$$F-measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{11}$$

$$TPR = \frac{A}{A+C} \quad FPR = \frac{B}{B+D} \tag{12}$$

4.4. Experimental Design

In order to verify the feasibility and effectiveness of our proposed method TSWNB, we design three experiments based on the research questions.

RQ1. Between equal-width discretization and equal-frequency discretization, which method is better for TSWNB?

We will make the comparison between equal-width discretization and equal-frequency discretization methods for TSWNB in this experiment. After making a comparison among intervals of 10, 15, 20, 25 and 30, we set intervals at 20 for both discretization methods. Taking interval 20 as an example, let the max and min values of feature x_d in a software defect dataset with m software modules be d_{max} and d_{min}, respectively. In equal-width discretization method, the attribute x_d is divided into 20 equal-width intervals by computing 19 splitting points $d_{min} + c\ (d_{max} - d_{min}) / 20$ for $c = 1, 2, \ldots, 19$. In equal-frequency discretization, the attribute x_d is sorted into ascending order and then its domain is divided into 20 intervals. After dividing, each interval has $\lfloor m/20 \rfloor$ software modules, where $\lfloor m/20 \rfloor$ refers to the largest integer less than or equal to $m/20$.

RQ2. What is the defect prediction performance of TSWNB when compared with the classic methods?

In this experiment, we implement three classification models (i.e. Naive Bayes (NB), Logistic Regression (LR) and Random Tree (RT)), which are widely employed in SDP.

NB is an exceptional, simple and very effective method of classification problem 4. In NB, the features are assumed to be independent, thus it is termed as 'naive'. Compared with those more complex classification models, the prediction accuracy of NB is very well because of its simplicity, effectiveness and robustness, especially when the amount of data is small [4, 5].

LR can also be used to software defect prediction via fitting software defect dataset to a logistic curve 23. It is a kind of probabilistic statistical regression model. When LR works as a binary predictor, it can be used in software defect predictor.

RT is one of the simplest hypothesis spaces possible 24. It contains two parts: a schema (i.e. a set of features), and a body (i.e. a set of labeled samples).

RQ3. What is the performance of TSWNB when compared with other weighted NB methods?

In this experiment, two feature weighting techniques are included for comparison. The first is traditional weighted Naive Bayes method implemented by GR 17, and the feature weight is only used in the formula of NB, i.e. it is implemented by equation (6) and equation (7). Another feature weighting technique for NB is feature selection Method CFS. Feature selection is a particular kind of feature weighting technique in which the weights of selected features are 1 while those of unselected features are 0 18. Therefore, CFS is included for comparison as a special kind of weighted Naive Bayes.

The purpose of CFS is to search the best feature subsets that are highly correlated with the class attribute, yet uncorrelated with each other. The equation (13) evaluates the merit of a feature subset *S* containing *k* features:

$$Merit_s = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \tag{13}$$

Where $\overline{r_{cf}}$ is the average feature-class correlation, and $\overline{r_{ff}}$ is the average feature-feature inter-correlation. Therefore, a bigger merit needs a bigger $\overline{r_{cf}}$ or a smaller $\overline{r_{ff}}$.

5. Experimental Results

In this section, the experiments are conducted based on the above experimental design and the results are analyzed to answer the three *RQ*. All the experiments are conducted according to the 10 × 10 – fold cross-validation method.

RQ1. Which method is better between equal-width and equal-frequency discretization methods for TSWNB?

First, we made the comparison among 10, 15, 20, 25, and 30 intervals with TSWNB based on *F-measure*. The experimental results are preset in Table 4. From the table, we can find that the interval 20 wins 4 times in both equal-width discretization and equal-frequency discretization methods. Therefore, we select 20 as the interval in our experiments.

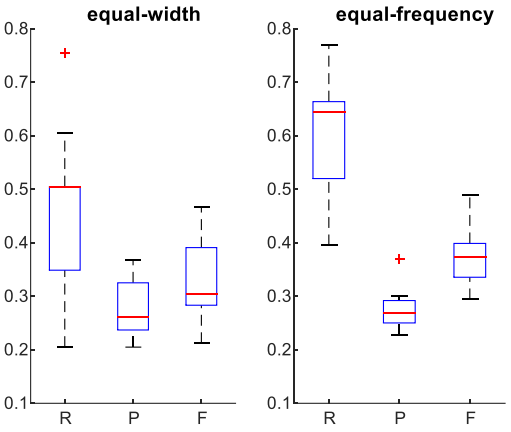
We list the more detailed performance indices of equal-width and equal-frequency discretization methods with interval 20 in Table 5 and Figure 1. From Table 5, considering the comprehensive evaluation index *F-measure*, we can see that TSWNB with equal-frequency discretization method is superior to equal-width discretization method for all the five software defect data sets. Figure 1 illustrates the experimental results in a more intuitive way. As shown in Table 5 and Figure 1, the equal-frequency discretization method is more suitable for TSWNB. Therefore, the equal-frequency discretization method is employed in TSWNB. The average value of parameter W_d of 10 ×10 fold Cross-validation for each feature used in the proposed method TSWNB with equal-frequency discretization method is listed in Table 6.

Table 4. The comparison among the intervals of 10, 15, 20, 25 and 30 with TSWNB based on *F-measure* (wins in bold)

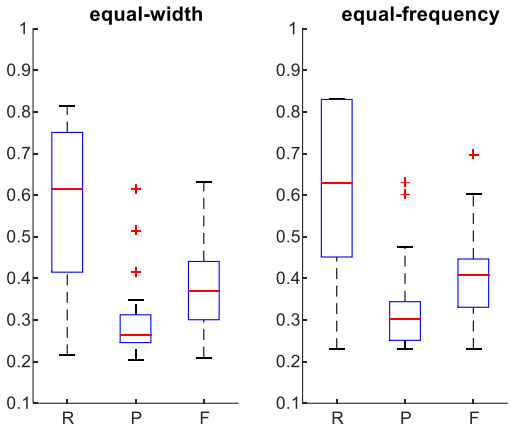
Name	equal-width					equal-frequency				
	10	15	20	25	30	10	15	20	25	30
CM1	0.3051	**0.3325**	0.3291	0.3156	0.3011	0.3025	0.3425	**0.3676**	0.3662	0.3012
MW1	0.3412	0.3681	**0.3792**	0.3709	0.3581	0.3523	0.4042	**0.4076**	0.3911	0.3620
PC1	0.3051	0.3321	**0.3500**	0.3215	0.3058	0.3021	0.3527	**0.3751**	0.3596	0.3124
PC3	0.3395	0.3612	**0.3735**	0.3721	0.3271	0.3051	0.3696	**0.3759**	0.3710	0.3307
PC4	0.4405	0.4608	**0.4808**	0.4528	0.4012	0.4913	0.5221	0.5376	**0.5406**	0.5068

Table 5. The comparison between TSWNB with equal-width discretization and TSWNB with equal-frequency discretization method (wins in bold)

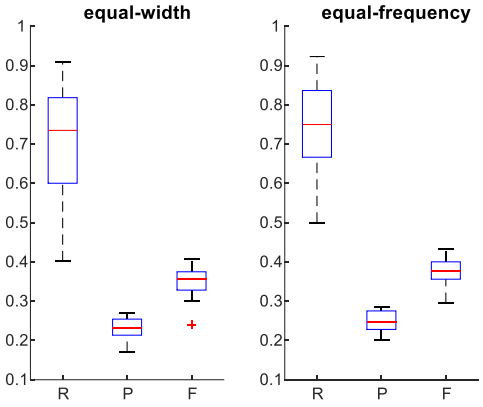
Name	equal-width			equal-frequency		
	Recall	Precision	F-measure	Recall	Precision	F-measure
CM1	0.4390	0.2734	0.3291	0.5920	0.2704	**0.3676**
MW1	0.5624	0.3024	0.3792	0.5911	0.3261	**0.4076**
PC1	0.7196	0.2334	0.3500	0.7610	0.2510	**0.3751**
PC3	0.7788	0.2459	0.3735	0.7762	0.2483	**0.3759**
PC4	0.6271	0.3908	0.4808	0.8592	0.3919	**0.5376**



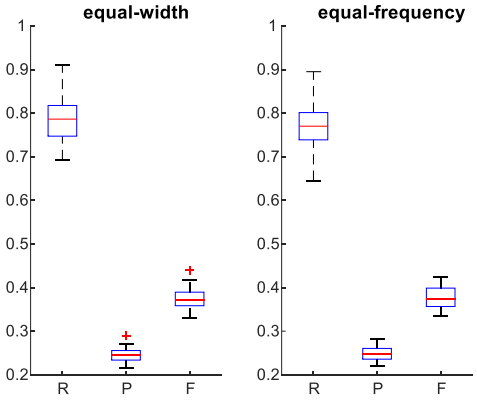
(a) experimental results on CM1



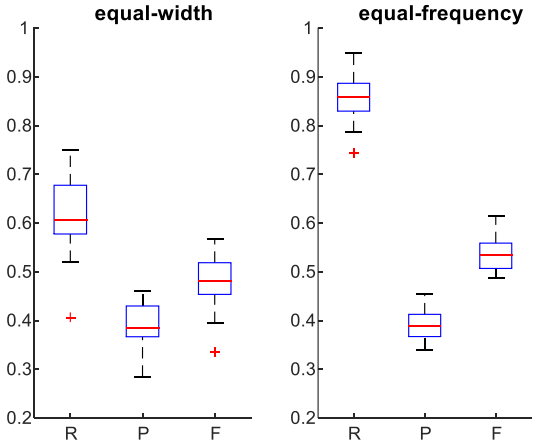
(b) experimental results on MW1



(c) experimental results on PC1



(d) experimental results on PC3



(e) experimental results on PC4

Figure 1. The standardized boxplots of the performances of TSWNB implemented by equal-width discretization and equal-frequency discretization method respectively. From the bottom to the top of a standardized box plot: minimum, first quartile, median, third quartile, and maximum. The outliers are plotted as some crosses.

Table 6. The average value of parameter W_d of 10 ×10 fold cross-validation for each feature used in TSWNB with equal-frequency discretization method.

Attributes	CM1	MW1	PC1	PC3	PC4
LOC_BLANK	0.8924	1.1025	2.2389	2.1454	1.5790
BRANCH_COUNT	0.7270	1.1463	0.4656	0.8546	0.3289
CALL_PAIRS	1.5275	1.1749	0.8122	0.7788	0.2864
LOC_CODE_AND_COMMENT	0.8682	0.4322	2.0733	2.3091	5.3435
LOC_COMMENTS	1.9741	1.1345	1.7280	1.6182	1.4822

CONDITION_COUNT	0.7125	1.3820	0.4810	0.6684	2.0145
CYCLOMATIC_COMPLEXITY	0.7132	1.1216	0.4586	0.7961	0.3758
CYCLOMATIC_DENSITY	0.6819	0.4852	1.5091	0.3074	1.3262
DECISION_COUNT	0.7741	1.3850	0.4459	0.6454	1.9716
DECISION_DENSITY	0.8748	1.5261	0.5424	0.5630	2.4306
DESIGN_COMPLEXITY	1.0855	1.5456	0.6133	0.9315	0.4712
DESIGN_DENSITY	1.0424	0.5830	0.4951	0.4708	0.2185
EDGE_COUNT	0.7962	1.1453	0.7056	0.9726	0.3971
ESSENTIAL_COMPLEXITY	0.6950	1.6643	0.1651	0.3709	0.3698
ESSENTIAL_DENSITY	1.2110	0.2672	0.2940	0.3762	0.2948
LOC_EXECUTABLE	1.1831	1.6322	1.4102	1.0230	0.5468
PARAMETER_COUNT	1.2212	0.1451	0.6131	0.2501	0.7077
HALSTEAD_CONTENT	1.2470	1.2429	1.9403	1.6235	0.6177
HALSTEAD_DIFFICULTY	0.6081	0.3221	0.4820	0.5119	0.3610
HALSTEAD_EFFORT	0.8972	0.8649	0.8365	1.0434	0.4480
HALSTEAD_ERROR_EST	1.1927	1.6238	1.2227	1.3107	0.6163
HALSTEAD_LENGTH	0.8542	1.0902	1.1925	1.3159	0.5592
HALSTEAD_LEVEL	0.7503	0.2927	0.4803	0.6736	0.3907
HALSTEAD_PROG_TIME	0.8972	0.8649	0.8365	1.0434	0.4480
HALSTEAD_VOLUME	1.1194	1.2799	1.2990	1.3432	0.5623
MAINTENANCE_SEVERITY	0.6707	0.6504	0.5482	0.8293	0.6291
MODIFIED_CONDITION_COUNT	0.6970	1.1694	0.4182	0.6449	1.9858
MULTIPLE_CONDITION_COUNT	0.7079	1.3032	0.4809	0.6663	2.0137
NODE_COUNT	0.6918	1.0757	0.6881	0.9370	0.3529
NORMALIZED_CYLOMATIC_COMPLEXITY	0.9168	0.1368	1.8461	0.5841	1.2209
NUM_OPERANDS	1.0473	1.0757	1.2796	1.3763	0.5178
NUM_OPERATORS	1.0096	1.4231	1.3016	1.2889	0.6047
NUM_UNIQUE_OPERANDS	1.3018	1.6083	1.6967	1.6790	0.5637
NUM_UNIQUE_OPERATORS	1.5603	0.4357	0.9114	0.8578	0.5171
NUMBER_OF_LINES	1.3615	1.1327	1.8974	1.5760	1.0977
PERCENT_COMMENTS	1.2639	0.1445	1.1606	1.4133	2.3611
LOC_TOTAL	1.2253	1.3899	1.4297	1.2002	0.9876

RQ2. What is the defect prediction performance of TSWNB when compared with the classic methods?

In this experiment, we use the equal-frequency discretization method in TSWNB, which has been illustrated in *RQ*1. The experimental results of the comparison among NB, LR, RT and TSWNB are presented in Table 7 and Figure 2. Considering the comprehensive evaluation index *F-measure*, we can see that TSWNB is superior to NB, LR and RT.

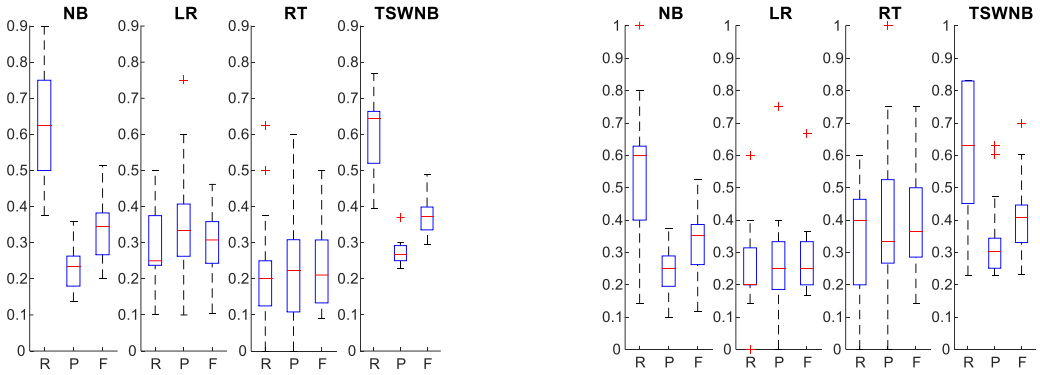
Table 7. The comparison among TSWNB and NB, LR, RT. (wins in bold)

Name	NB			LR			RT			TSWNB		
	Recall	Precision	F-measure	Recall	Precision	F-measure	Recall	Precision	F-measure	Recall	Precision	F-measure
CM1	0.6380	0.2278	0.3333	0.2780	0.3423	0.2909	0.2090	0.2182	0.2356	0.5920	0.2704	**0.3676**
MW1	0.5771	0.2427	0.3374	0.2320	0.2412	0.2755	0.3429	0.4211	0.3951	0.5911	0.3261	**0.4076**
PC1	0.6654	0.1697	0.2693	0.2403	0.3781	0.3026	0.3249	0.2925	0.3039	0.7610	0.2510	**0.3751**
PC3	0.7375	0.2169	0.3349	0.0900	0.3411	0.1405	0.2763	0.3192	0.2929	0.7762	0.2483	**0.3759**
PC4	0.7486	0.2985	0.4260	0.4735	0.5472	0.4920	0.5059	0.5393	0.5185	0.8592	0.3919	**0.5376**

To further verify the performance of the classifier, we also compare NB, LR, RT and TSWNB with AUC values. The experimental results are shown in Table 8. Each value presented in Table 8 is the average AUC value for every classification model obtained from the 10 ×10 fold Cross-validation. We can see that the performance of TSWNB is still better than NB, LR and RT. These AUC values are the area of ROC in each experiment. Taking the dataset PC4 as an example, a ROC curve among the 10 × 10 = 100 experiments is shown in Figure 3. We can see that TSWNB is superior to NB, LR and RT on each threshold.

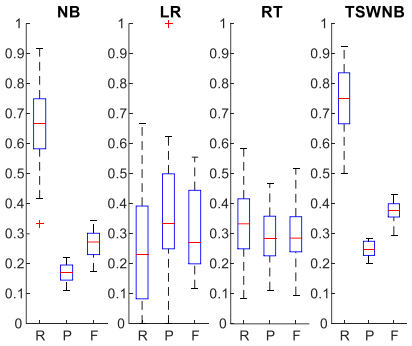
Table 8. The comparison among NB, LR, RT and TSWNB with AUC values (wins in bold)

Name	NB	LR	RT	TSWNB
CM1	0.7421	0.7043	0.6902	**0.7685**
MW1	0.6895	0.6428	0.7183	**0.7829**
PC1	0.7114	0.6237	0.6729	**0.7795**
PC3	0.6797	0.6156	0.6203	**0.7803**
PC4	0.6352	0.6501	0.7186	**0.7835**

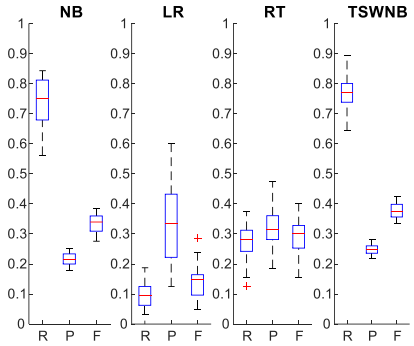


(a) Experimental results on CM1

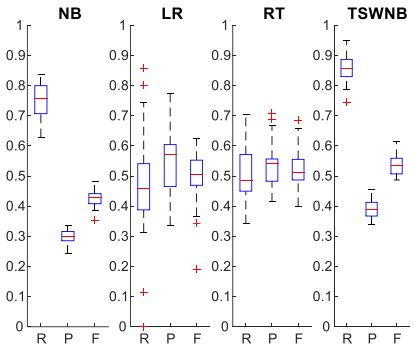
(b) Experimental results on MW1



(c) Experimental results on PC1



(d) Experimental results on PC3



(e) Experimental results on PC4

Figure 2. Standardized boxplots of the performances of NB, LR, RT and TSWNB respectively

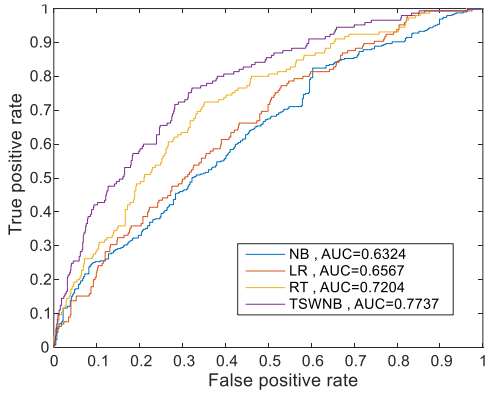
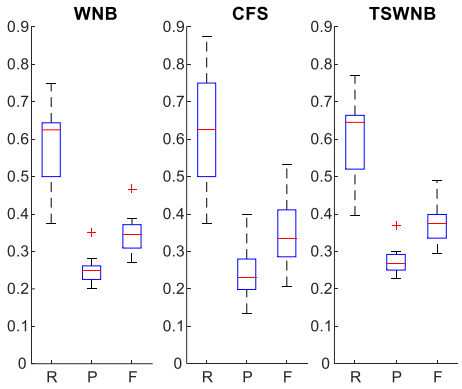


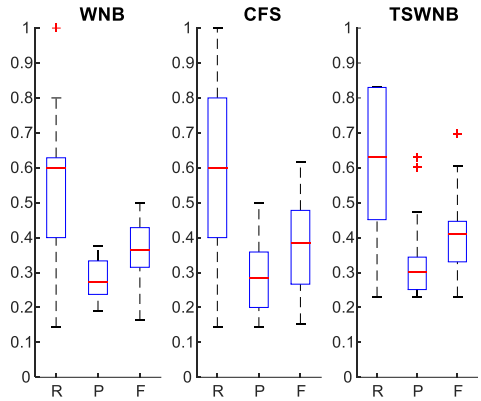
Figure 3. A ROC curve obtained from PC4 among the 10 × 10 = 100 experiments

RQ3. What is the performance of TSWNB when compared with other weighted NB methods?

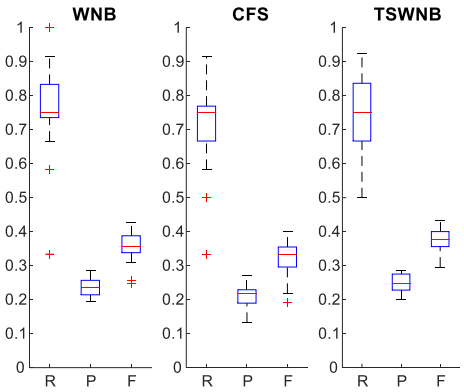
In this experiment, two feature weighting techniques are included for comparison, i.e. the weighted Naive Bayes method implemented by GR and the special kind of feature weighting technique CFS. We use the equal-frequency discretization method in TSWNB illustrated in *RQ*1. From Table 9 and Figure 4, considering the comprehensive evaluation index *F-measure*, we can see that TSWNB performs better than other two feature weighting techniques except for PC3. But even in that case, TSWNB is also comparable to CFS. Note that the number of software modules in PC3 is the largest. NB is good at handling data sets with a small amount of data [4, 5]. Maybe, the same is true for TSWNB.



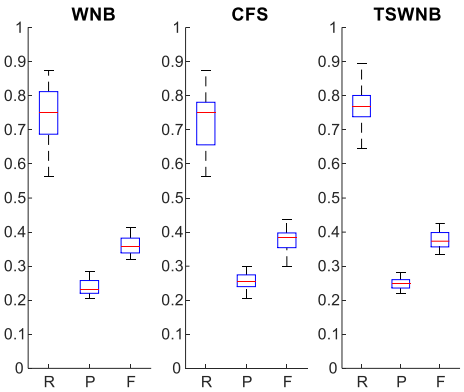
(a) Experimental results on CM1



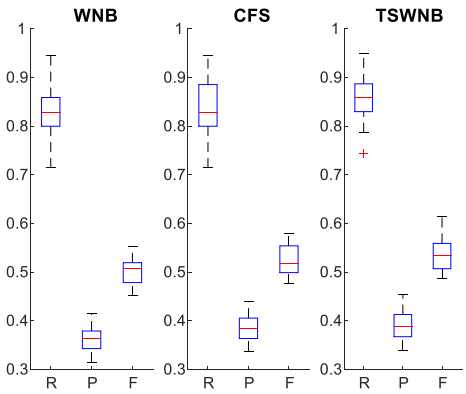
(b) Experimental results on MW1



(c) Experimental results on PC1



(d) Experimental results on PC3



(e) Experimental results on PC4

Figure 4. Standardized boxplots of the performances of two weighted Naive Bayes and TSWNB

Table 9. The comparison among two weighted Naive Bayes and TSWNB. (wins in bold)

Name	WNB			CFS			TSWNB		
	Recall	Precision	F-measure	Recall	Precision	F-measure	Recall	Precision	F-measure
CM1	0.5720	0.2473	0.3416	0.6200	0.2418	0.3451	0.5920	0.2704	**0.3676**
MW1	0.5771	0.2786	0.3682	0.5714	0.2870	0.3751	0.5911	0.3261	**0.4076**
PC1	0.7438	0.2384	0.3583	0.7179	0.2109	0.3248	0.7610	0.2510	**0.3751**
PC3	0.7512	0.2399	0.3612	0.7288	0.2548	**0.3769**	0.7762	0.2483	0.3759
PC4	0.8277	0.3612	0.5019	0.8367	0.3836	0.5249	0.8592	0.3919	**0.5376**

6. Conclusions

In this empirical study, to boost the potential of Naive Bayes for software defect prediction, we design and implement a two-stage feature weighting method TSWNB, feature discretization and feature weighting. For the stage of feature discretization, since NB is good at dealing with discrete data while the attributes of software defect data sets are usually continuous or numeric, we make the comparison between equal-width discretization method and equal-frequency discretization method for NB, and find that equal-frequency discretization method is more appropriate for TSWNB. For the stage of feature weighting, as features are assumed to be independent and thus equal important while they are often not the case in common practice, we use the feature weighting technique to alleviate the equal importance assumption, which combines the obtained feature weights into the NB formula and its likelihood estimations. To validate the performance of TSWNB, we conduct experiments on 5 software defect data sets from NASA MDP provided by PROMISE repository. Three well-known classification algorithms and two feature weighting techniques are included for comparison. The final experimental results show the potential of TSWNB in boosting the performance of NB for software defect prediction.

Acknowledgements

This paper was supported in part by the National Natural Science Foundation of China (Grant No. 61702544), the Natural Science Foundation of Jiangsu Province, China (No.BK20160769) and Youth Talent Support Program of Huaiyin Normal University (No.13HQNZ07).

References

1. R. Özakıncı, A. Tarhan, “Early Software Defect Prediction: A Systematic Map and Review,” Journal of Systems & Software, 2018
2. R. Malhotra, “An empirical framework for defect prediction using machine learning techniques with Android software,” Applied Soft Computing, 2016.
3. T. Menzies, J. Greenwald, and A. Frank, “Data mining static code attributes to learn defect predictors,” IEEE Transactions on Software Engineering, vol. 33, no. 1, pp. 2-13, 2007.
4. J. Hand, K. Yu, “Idiot’s Bayes - not so stupid after all?,” International Statistical Review, vol. 69, no. 3, pp. 385-398, 2001.
5. B. Cestnik, I. Kononenko, and I. Bratko, “Assistant 86: A knowledge-elicitation tool for sophisticated users,” In Proceedings of the Second European Working Session on Learning, pp. 31-45, Wulmslow, UK: Sigma Press, 1987.
6. T. T. Wong, “A hybrid discretization method for naïve Bayesian classifiers,” Pattern Recognition, vol. 45, no. 6, pp. 2321-2325, 2012.
7. R. Kohavi, M. Sahami, Error-based and entropy-based discretization of continuous features. Proceedings of the Second International Conferenceon Knowledge Discovery and Data Mining, Portland, OR, pp.114–119, 1996.
8. B. Turhan, A. B. Bener, “Software Defect Prediction: Heuristics for Weighted Naive Bayes,” Proceedings of the Second International Conference on Software and Data Technologies, pp. 244-249, 2007.
9. N. B. Ebrahimi, “On the statistical analysis of the number of errors remaining in a software design document after inspection,” IEEE Trans. Softw. Eng., vol. 23, no. 8, pp. 529–532, 1997.
10. K. El Emam and O. Laitenberger, “Evaluating capture-recapture models with two inspectors,” IEEE Trans. Softw. Eng., vol. 27, no. 9, pp. 851–864, 2001.
11. S. S. Rathore, S. Kumar, “Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems,” Knowle dge-Base d Systems, vol. 119, pp. 232-256, 2017.
12. K. Ganesan, T. M. Khoshgoftaar, and E. Allen, “Case-based software quality prediction,” Int’l J. Software Eng. and Knowledge Eng., vol. 10, no. 2, pp. 139–152, 2000.
13. B. Turhan and A. Bener, “Analysis of naive bayes’ assumptions on software fault data: An empirical study,” Data Knowledge Eng., vol. 68, no. 2, pp. 278–290, 2009.

14. H. M. Olague, S. Gholston, S. Quattlebaum, “Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes,” IEEE Transactions on Software Engineering, vol. 33, no. 6, pp. 402–419, 2007.
15. J. Dougherty, R. Kohavi, M. Sahami, Supervised and unsupervised discretization of continuous features. Proceedings of the 12th International Conference on Machine Learning, SanFrancisco, CA:MorganKaufmann, pp.194 – 202, 1995.
16. J. Catlett, “On changing continuous attributes into ordered discrete attributes,” Proceedings of the 5th European Working Sessionon Learning on Machine Learning, Porto, Portugal, , pp.164–178, 1991.
17. H. Zhang, S. Sheng, “Learning weighted naïve Bayes with accurate ranking,” In Proceedings of the 4th International Conference on Data Mining. IEEE, Brighton, UK, pp.567–570, 2004.
18. M. Hall, “Correlation-based feature selection for discrete and numeric class machine learning,” In: Proceedings of the 17th International Conferenceon Machine Learning, pp.359–366, 2000.
19. L. Jiang, C. Li, S. Wang, L. Zhang, “Deep feature weighting for naive Bayes and its application to text classification,” Engineering Applications of Artificial Intelligence, vol. 52, no. C, pp. 26 –39, 2016.
20. E. Alpaydin, “Introduction to Machine Learning,” The MIT Press, October 2004.
21. J. R. Quinlan, “C4.5: programs for machine learning,” 1993.
22. S. Huda, et al., “A Framework for Software Defect Prediction and Metric Selection,” IEEE Access, no. 99, 2017
23. H. M. Olague, S. Gholston, S. Quattlebaum, “Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes,” IEEE Transactions on Software Engineering, vol. 33, no. 6, pp. 402–419, 2007.
24. G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright, "A Practical Differentially Private Random Decision Tree Classifier," in IEEE International Conference on Data Mining Workshops, pp. 114-121, 2009.

Haijin Ji received his M.A.SC. in Information Science from Jiangnan University, Wuxi, China. He is now a Doctor candidate in Army Engineering University of PLA. His current research interests include software testing, defects prediction and fuzzy information fusion.

Song Huang Song Huang is a professor of software engineering at Software Testing and Evaluation Center of Army Engineering University of PLA. His current research interests include software testing, quality assurance, empirical software engineering. He is a member of CCF and ACM.

Xuewei Lv is now a Doctor candidate in Army Engineering University of PLA. His current research interests include software testing, defects prediction and fuzzy information fusion.

Yaning Wu She is now a Doctor candidate in Army Engineering University of PLA. Her current research interests include software testing, defects prediction, linguistic preference modelling and fuzzy information fusion.

Zhanwei Hui received his M.A.SC. in Information Science from Army Engineering University of PLA, Nanjing, China. His current research interests include software testing and defects prediction.