



A new weighted naive Bayes method based on information diffusion for software defect prediction

Haijin Ji^{1,2} · Song Huang² · Yaning Wu² · Zhanwei Hui² · Changyou Zheng²

Published online: 02 January 2019

© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

Software defect prediction (SDP) plays a significant part in identifying the most defect-prone modules before software testing and allocating limited testing resources. One of the most commonly used classifiers in SDP is naive Bayes (NB). Despite the simplicity of the NB classifier, it can often perform better than more complicated classification models. In NB, the features are assumed to be equally important, and the numeric features are assumed to have a normal distribution. However, the features often do not contribute equivalently to the classification, and they usually do not have a normal distribution after performing a Kolmogorov-Smirnov test; this may harm the performance of the NB classifier. Therefore, this paper proposes a new weighted naive Bayes method based on information diffusion (WNB-ID) for SDP. More specifically, for the equal importance assumption, we investigate six weight assignment methods for setting the feature weights and then choose the most suitable one based on the *F-measure*. For the normal distribution assumption, we apply the information diffusion model (IDM) to compute the probability density of each feature instead of the acquiescent probability density function of the normal distribution. We carry out experiments on 10 software defect data sets of three types of projects in three different programming languages provided by the PROMISE repository. Several well-known classifiers and ensemble methods are included for comparison. The final experimental results demonstrate the effectiveness and practicability of the proposed method.

Keywords Software defect prediction · Naive Bayes · Feature weighting · Information diffusion

✉ Song Huang
hs0317@163.com

¹ School of Computer Science and Technology, Huaiyin Normal University, Huaian 223300, China

² Command & Control Engineering College, Army Engineering University of PLA, Nanjing 210007, China

1 Introduction

In the software development lifecycle, software testing is one of the most important phases. At this stage, software testing engineers should identify as many software defects as possible because the cost of rectifying a software defect after release is much higher than during the software development stage (Pelayo and Dick 2007). Furthermore, in many software organizations, the project budget is often limited, or the entire software system is large and cannot be tested exhaustively. Therefore, how to effectively allocate the limited testing resources to those software modules (i.e., the smallest unit of functionality) that are most likely to contain defects is particularly crucial to a software organization. To solve this problem, the notion of software defect prediction (SDP) is proposed and has attracted the attention of a growing number of researchers in the field of software engineering (Turhan et al. 2009; Menzies et al. 2007; Yang et al. 2015; Ghotra et al. 2015; Kamei et al. 2013; Xia et al. 2016). The main task of SDP models is to classify software modules into two types, defective or non-defective, using software metrics (i.e., features or attributes), such as McCabe (McCabe 1976) metrics and Halstead (Halstead 1977) metrics. The performance indices of SDP models can be repeated and compared in publicly available data sets, such as the PROMISE repository (Boetticher et al. 2007).

Naive Bayes (NB) is one of the most commonly used classifiers in SDP (Malhotra 2015), and its performance is usually superior to the more complex classifiers (Menzies et al. 2007; Hand and Yu 2001; Zaidi et al. 2013). NB is also used in many other learning scenarios, such as text classification, web page mining, fraud detection, and image classification because of its simplicity and effectiveness. However, there are still some issues to be resolved, especially concerning the assumptions of features in NB, including the equal importance assumption (Turhan et al. 2009; Turhan and Bener 2007; Zhang and Sheng 2005), the normal distribution assumption (Witten et al. 2011), and the independence assumption (Zaidi et al. 2013; Arar and Ayan 2017; Turhan et al. 2009), which are not true in many problems. In this paper, our study mainly focuses on the issues of the equal importance assumption and the normal distribution assumption to improve the performance of NB for SDP.

The equal importance assumption means that each feature of the software contributes equivalently to defect prediction. However, in practice, as different features are designed for different purposes, not all of them are good indicators of software defects (Turhan and Bener 2007). Therefore, the equal importance assumption is inappropriate. One solution to this problem is feature-weighting techniques, which assign each feature weight based on its degree of importance. This technique is commonly called weighted naive Bayes (WNB). Turhan and Bener (2007) applied a heuristic method for feature-weighting in WNB. In more specific terms, they used information gain (IG), gain ratio (GR), and principal component analysis (PCA) to evaluate the degree of importance of features. The experimental results illustrated that IG and GR are more suitable for WNB than PCA. However, a number of other algorithms are still available for importance measures, for example, Chi square (CS) (Plackett 1983), IG (Witten et al. 2011), GR (Quinlan 1993), symmetrical uncertainty (SU) (Yu and Liu 2003), ReliefF (RF) (Kira and Rendell 1992; Robnikšikonja and Kononenko 2003), and information flow (IF) (Liang 2014; Bai et al. 2018), to name a few. Thus, we must identify which method is more appropriate to assign the feature weights in WNB.

The normal distribution assumption refers to the general assumption that a data set with continuous or numeric features follows a normal distribution. However, after performing a normal distribution hypothetical test on 10 data sets (the data sets are illustrated in Sect. 4), we

found that the features do not follow a normal distribution. More specifically, we conduct a Kolmogorov-Smirnov (KS) test (Razali and Wah 2011) 10,000 times on each data set based on a “kstest” function in MATLAB; the acquiescent null hypothesis is that each feature comes from a normal distribution; the return value h equals 1 if it rejects the null hypothesis at a significance level 0.05; otherwise, h equals 0. We take the data set Xalan-2.6 as an example, the results of which are provided in Table 1. As shown in the table, the normal distribution assumption is often violated, and as a result, its probability density estimation at each value of numeric features for NB is often suboptimal.

Our solution to the normal distribution assumption is to use the information diffusion model (IDM) to improve the WNB classifier in this paper. Bai (Bai et al. 2015) has illustrated that IDM is effective when dealing with the small sample issue based on the fuzziness of incomplete data and the acoustic wave propagation. More specifically, the incomplete data set is regarded as a piece of fuzzy information in IDM, and thus, IDM can extract some additional information via diffusion (Huang 1997). Greater reliability for time-series analysis (Bai et al. 2014), risk assessment (Huang 2002), exceeding probability estimation (Huang and Shi 2012), and relation detection (Bai et al. 2017) has been achieved using IDM. Therefore, in this paper, we treat a training defect data set as an incomplete data and use IDM to compute the feature probability density instead of the normal distribution probability density function in WNB for SDP.

As discussed above, we propose a new weighted naive Bayes method based on information diffusion (WNB-ID). It combines WNB and IDM to improve the performance of NB in SDP. We intend to apply different feature-weighting techniques in WNB to explore the more appropriate weight for each feature and then use IDM in probability density estimation. To the best of our knowledge, no study has been reported in SDP literature that has used this method in SDP. Thus, we design experiments to address the following research questions.

- *RQ1*. Which feature-weighting technique is more suitable for alleviating the equal importance assumption in WNB?
- *RQ2*. How can IDM appropriately alleviate the normal distribution assumption to improve the performance of NB for SDP?
- *RQ3*. What is the defect prediction performance of WNB-ID compared with other prediction methods?

Table 1 The results of Xalan-2.6 for the Kolmogorov-Smirnov test

Feature	$h = 0$	$h = 1$	Feature	$h = 0$	$h = 1$
WMC	2	9998	LOC	7	9993
DIT	4	9996	DAM	8	9992
NOC	1	9999	MOA	7	9993
CBO	4	9996	MFA	5	9995
RFC	6	9994	CAM	8	9992
LCOM	7	9993	IC	8	9992
CA	8	9992	CBM	7	9993
CE	9	9991	AMC	6	9994
NPM	5	9995	MAX_CC	8	9992
LCOM3	9	9991	AVG_CC	4	9996
BUG	1	9999			

In this paper, our primary contributions are as follows:

1. The paper proposes a new weighted naive Bayes method based on information diffusion (i.e., WNB-ID). To our knowledge, this is the first paper to integrate feature-weighting techniques and the information diffusion model to boost the potential of NB for SDP.
2. The paper conducts a comprehensive investigation of feature-weighting techniques for weight assignment, including CS, IG, GR, SU, RF, and IF. They are evaluated on 10 real-world data sets of three types of projects in three different programming languages. The most appropriate feature-weighting algorithm, i.e., IG, is used to alleviate the equal importance assumption of NB for SDP. Meanwhile, IDM is used to overcome the normal distribution assumption to further improve the performance of WNB for SDP by calculating the probability density of each feature instead of the normal probability density function of the normal distribution.
3. The paper evaluates the effectiveness of WNB-ID via several group experiments. A set of experiments are designed to verify the improvement of WNB-ID, and the main components of WNB-ID are also verified and analyzed. Classic classification algorithms and the state-of-the-art method are also included for comparison. The final experimental results demonstrate the effectiveness of WNB-ID.

This work is based on our earlier work (Wu et al. 2018). However, this paper differs from our previous work in the following aspects:

- The application backgrounds of our new proposed method WNB-ID and our previous work are different. We apply our new proposed method WNB-ID in within-project defect prediction (WPDP), while our previous work focuses on cross-project defect prediction (CPDP). In WPDP, the SDP models are trained from the data of historical releases in the same project. In CPDP, the SDP models are trained from the historical data from other projects.
- We apply the feature-weighting techniques to address the equal importance assumption of NB, which is an extension of our previous work. In our previous work, the features of data sets are assumed to be equally important. In this paper, we investigate six feature-weighting techniques to address the equal importance assumption in NB, i.e., CS, IG, GR, SU, RF, and IF, and we select the best feature-weighting technique IG in our proposed method based on the aggregative indicator *F-measure*. The detailed feature weights are also listed in the paper, which may be helpful for further studies. We find that the performance of our proposed method WNB-ID is significantly improved after applying feature-weighting techniques. In addition, IF is first introduced to assign feature weights in SDP.
- We use a more comprehensive set of data sets to evaluate the performance of our proposed method. In the original study, only the java projects are used in our experiments. In this paper, ten real-world data sets of three types of projects in three different programming languages are used to evaluate the performance of our proposed method. These data sets include balanced and imbalanced data sets. The reason for choosing these data sets is to verify the generality of our proposed method. The experimental results demonstrate that our proposed method WNB-ID is effective in the three types of projects using both balanced and imbalanced data sets.

- The state-of-the-art ensemble method is also included for comparison. In our previous work, we only make comparisons between the proposed method and classic classifiers. In this paper, the ensemble learning method STSE is also included for comparison. After comparison, we find that our proposed method WNB-ID is superior to STSE.
- We provide a more comprehensive evaluation and comparison. In this paper, we first compare WNB (CS), WNB (IG), WNB (GR), WNB (SU), WNB (RF), and WNB (IF) with standard NB to verify whether the feature-weighting techniques can alleviate the equal importance assumption of NB and select the most appropriate feature-weighting technique. Subsequently, IDM is used to alleviate the normal distribution assumption of NB. Finally, we compare our proposed method WNB-ID with classic classifiers (NB, SVM, LR, and RT), its two components (WNB (IG) and NB-ID), and the state-of-the-art method, STSE. In all the experiments, the 10×10 -fold cross-validation method is used to guarantee the stability of the experiment, and Student's t test is conducted to further verify the results.

The remainder of this paper is organized as follows: Section 2 summarizes related work. Section 3 describes the new proposed methodology for SDP. Sections 4 and 5 describe the detailed experimental setups and the results of the primary analysis, respectively. Section 6 presents the threats to validity. Discussions about our research are presented in Sect. 7. Finally, we present the conclusions in Sect. 8.

2 Related work

2.1 Software defect prediction

SDP models are designed to improve the software quality and reduce the costs of developing software systems. The field developed rapidly after the PROMISE repository (Boetticher et al. 2007) was established in 2005. This repository includes a great number of defect data sets from real-world projects for public use and can be used to build repeatable, comparable models for SDP studies. Great research efforts for SDP can be divided into two categories.

The first category is the metrics (i.e., features or attributes) describing a module for SDP model building. The widely used static code metrics have been defined by McCabe (McCabe 1976) and Halstead (Halstead 1977). McCabe metrics represent the complexity of pathways of a module's flow graph. Halstead metrics use the number of operators and operands to estimate the reading complexity of a module. Many other software metrics have also been proposed for SDP in recent years, including network metrics (Ma et al. 2016a, b), web metrics (Bicer and Diri 2015), cascading style sheet metrics (Bicer and Diri 2016), developer micro interaction metrics (Lee et al. 2016), change metrics (Kim and Zhang 2008), package modularization metrics (Zhao et al. 2015), mutation-based metrics (Bowes et al. 2016), lines of comments (Aman et al. 2015), context-based package cohesion metrics (Zhao et al. 2017), and code smell metrics (Hall et al. 2014). The more complex a module is, the more likely it is to be defective. However, since metrics contribute differently to defect prediction, we should treat them accordingly.

Another significant category is the learning algorithms applied in SDP. SDP can be treated as a learning problem in software engineering (Tantithamthavorn et al. 2017), and many machine learning techniques have been employed in SDP, such as naive Bayes (NB)

(Menzies et al. 2007; Turhan et al. 2009; Olague et al. 2007), support vector machine (SVM) (Jin and Liu 2010), logistic regression (LR) (Olague et al. 2007), random tree (RT) (Jagannathan et al. 2009), random forest (Lee et al. 2016), decision trees (Khoshgoftaar and Seliya 2002), and neural networks (Zheng 2010). In recent years, because of the great potential contribution to software organizations, this problem has attracted the attention of a large number of researchers (Malhotra 2015; Lenz et al. 2013). Some improved algorithms have been proposed and used for SDP, such as ensemble learning techniques (Tong et al. 2018; Yang et al. 2017), association rules (Miholca et al. 2018), and multi-objective optimization (Chen et al. 2018). However, Menzies et al. (2007) illustrated that NB may be more suitable than the complicated classifiers in SDP. This is the motivation of our study.

2.2 Naive Bayes

NB is one of the simplest classifiers based on the conditional probability (Hand and Yu 2001). In NB, the features are assumed to be independent, which is why the classifier is termed 'naïve'. It can usually perform better than more complex classification models (Menzies et al. 2007; Hand and Yu 2001; Zaidi et al. 2013), and it has been used in many learning domains, such as text classification (Tang et al. 2016), web mining (Shirakawa et al. 2015), fraud detection (Ma et al. 2016a, b), and image classification (Vitello et al. 2014). Because of its steady performance in many learning scenarios, NB is still an important research topic. NB has three assumptions, e.g., the equal importance assumption, the normal distribution assumption, and the independence assumption, which have been explained in Sect. 1. However, these assumptions are often violated in real-world applications, and they lead to suboptimal classification results. To address this issue, researchers have exerted some efforts to alleviate these assumptions. They mainly concentrate on alleviating the equal importance and independence assumptions, including feature-weighting NB and semi-NB. For the feature-weighting NB, the weights of highly predictive features should be increased while those of less predictive features should be decreased (Zaidi et al. 2013; Wong 2012; Zhang and Sheng 2005). Following a different course, semi-NB studies have mainly focused on alleviating the independence assumption (Yang et al. 2016; Zheng and Webb 2005).

2.3 Naive Bayes for software defect prediction

According to Malhotra's statistics in his literature review (Malhotra 2015), Bayesian classifiers are widely used in the domain of SDP and account for 47.7% among all SDP studies. However, most of them use standard NB, which account for 74%, while only 5% of them use weighted naive Bayes (WNB), and few works have attempted to alleviate the normal distribution assumption of NB. These statistics show that room exists to improve NB by alleviating the equal importance and normal distribution assumptions in the field of SDP.

Menzies et al. (2007) demonstrated that NB with log filtering and feature selection obtained the best results. In another study, Turhan and Bener (2009) alleviated the independence assumption using principal component analysis and the equal importance assumption by the feature-weighting technique. The results were better than those using standard NB. Similar research was also conducted in Arar and Ayan (2017), and more successful results than standard NB were obtained.

For feature-weighting NB, the weights of highly predictive features should be increased, while those of less predictive features should be decreased (Zaidi et al. 2013; Wong 2012;

Zhang and Sheng 2005). There are a number of feature-weighting algorithms that can be used to measure the importance of features, such as Chi square (CS), information gain (IG), gain ratio (GR), symmetrical uncertainty (SU), ReliefF (RF), and information flow (IF) (Liang 2014; Bai et al. 2018). Some studies have used them in SDP. For example, Turhan and Bener (2007) used IG, GR, and principal component analysis (PCA) to alleviate the equal importance assumption and found that assigning weights to software metrics significantly increased the prediction performance. Hosseini et al. (2018) used IG for feature selection to extend their previous work, and they found that IG is effective in their proposed method. Zhou et al. (2017) employed CS to perform feature selection to address the class imbalance of data sets. The experimental results demonstrated that CS is helpful in improving the performance of their proposed method. In addition, IF is a newly proposed method in physics and has achieved good results in many areas (Kaufman et al. 2011; Macias et al. 2016). IF has never been employed for feature-weighting in SDP.

Although these feature-weighting techniques can improve the performance of NB in SDP, we need to explore the best one for the appropriate weights in NB. Another issue in the lack of SDP research is the normal distribution assumption, which is also addressed in this paper.

3 Proposed methodology for software defect prediction

This section presents the weighted naive Bayes method based on information diffusion (WNB-ID), which combines weighted naive Bayes (WNB) and information diffusion model (IDM) to alleviate the equal importance and normal distribution assumptions in NB for SDP. In particular, for WNB, we apply different feature-weighting techniques with the purpose of identifying the most appropriate technique for assigning weights for features. For IDM, we use it to estimate the probability density instead of the normal distribution probability density function for each feature. Finally, we propose a novel IDM-based WNB algorithm, which would improve the performance of NB for SDP.

3.1 The entire framework

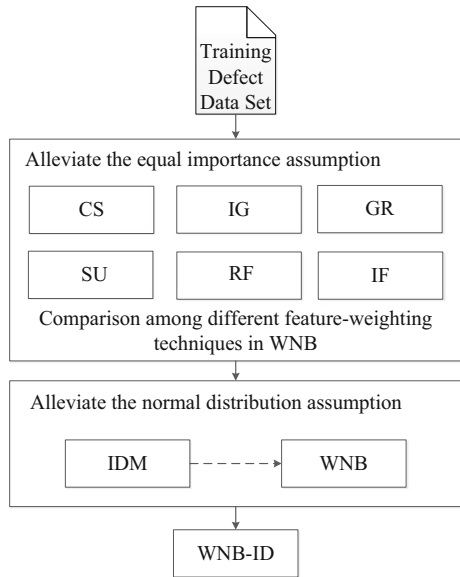
Figure 1 illustrates the entire framework of the proposed method WNB-ID. The first stage is the application of WNB, which aims to alleviate the equal importance assumption. The other stage is the application of IDM, which aims to alleviate the normal distribution assumption.

In our method, the stage of application of WNB is performed before the stage of the application of IDM. Specifically, the former applies different feature-weighting techniques in WNB to assign appropriate weights for features, while the latter performs the probability density estimation via the diffusion methods of IDM.

3.2 Application of weighted naive Bayes

In this stage, we apply WNB method to alleviate the equal importance assumption in NB. The idea of WNB is to increase the weight of highly predictive features and decrease that of less predictive features (Zaidi et al. 2013; Wong 2012; Zhang and Sheng 2005). We apply the WNB method, which uses a specific algorithm to measure the degree of importance of features. The expected result of this stage is to obtain a series of weights of features that can help WNB improve performance.

Fig. 1 The entire framework of the WNB-ID method



There are many different algorithms that can measure the degree of importance of features in WNB, such as Chi square (CS) (Plackett 1983), information gain (IG) (Witten et al. 2011), gain ratio (GR) (Quinlan 1993), symmetrical uncertainty (SU) (Yu and Liu 2003), ReliefF (RF) (Kira and Rendell 1992; Robnikšikonja and Kononenko 2003), and information flow (IF) (Liang 2014; Bai et al. 2018). Thus, it is necessary to determine which algorithm to use in WNB-ID based on *F-measure*. For ease of readers' understanding, we will explain the details of this method in the following subsections.

3.2.1 Weighted naive Bayes

In this subsection, we present a complete derivation of WNB referring to Zhang and Sheng (2005).

According to Bayesian theory (Witten et al. 2011), the posterior probability of an instance is proportional to the prior probability and likelihood, and features (i.e., attributes or metrics) are assumed to be independent in NB. More formally, let $A = \{A_1, A_2, \dots, A_n\}$ be a set of software features, $X = \{(A_1, a_1), (A_2, a_2), \dots, (A_n, a_n)\}$ be a software module, and $a = \{a_i \mid i = 1, 2, \dots, n\}$ be the associated value. The class of a software module is then defined as $C \in \{C_0, C_1\}$, where C_0 is the non-defective class and C_1 is the defective one. Therefore, the probability of a specific class to which a software module belongs can be computed by Eq. (1):

$$P(C_k|X) = \frac{P(C_k) \prod_{i=1}^n P(a_i|C_k)}{P(X)} \quad (1)$$

We use training defect data sets to train the classifier and compute the value of $\text{argmax}_k(P(C_k|X))$, which will determine the class of a new software module. As shown by Eq. (2), the classifier will classify software module X to the class with the higher $V(X)$. Please note that the denominator $P(X)$ in Eq. (1) is the same for all classes, and it will not affect the classification. Thus, it is removed in Eq. (2).

$$V(X) = \operatorname{argmax}_k \left(P(C_k) \prod_{i=1}^n P(a_i|C_k) \right) \quad (2)$$

To alleviate the equal importance assumption in NB, feature weights are introduced. Thus, we update Eq. (2) as in Eq. (3), and this model is called the weighted naive Bayes (WNB).

$$V(X) = \operatorname{argmax}_k \left(P(C_k) \prod_{i=1}^n P(a_i|C_k)^{w_i} \right) \quad (3)$$

where w_i refers to the weight of feature A_i .

$P(C_k)$ in Eq.(3) is computed using Eq.(4),

$$P(C_k) = \frac{N_k}{N} \quad (4)$$

where N_k refers to the number of software modules that belong to class k and N is the total number of software modules in the training data sets.

In NB, numeric features are assumed to have a normal distribution (Witten et al. 2011). Therefore, $P(a_i|C_k)$ in Eq. (3) can be calculated by Eq. (5),

$$P(a_i|C_k) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(a_i-\mu_i)^2}{2\sigma_i^2}} \quad (5)$$

where the parameters of μ_i and σ_i are the mean and the standard deviation of feature A_i in training data sets belonging to class k .

The above has briefly described the principle of WNB. From the above derivation, we can see that the value of w_i is crucial to WNB; it is based on the degree of importance of features. Thus, we perform experiments to identify the most suitable feature-weighting technique in existing methods based on *F-measure*, and the feature-weighting techniques will be illustrated in the following subsections. Another significant problem is the normal distribution assumption shown in Eq. (5), which has been experimentally proved not true. This problem will be discussed in Sect. 3.3.

3.2.2 Chi square

The CS test (Plackett 1983) is used to evaluate whether there is a significant relationship between the distribution of values of a feature and class. The null hypothesis is that there is no correlation between them. Given this null hypothesis, the χ^2 -statistic computes the deviation degree of the actual value from the expected value (Plackett 1983). In this paper, we use the χ^2 value of each feature to measure its degree of importance. The larger the χ^2 value is, the more this feature is associated with the class. Therefore, the weight w_i of feature A_i can be computed by Eq. (6), where $i = 1, 2, \dots, n$, and n is the number of features.

$$w_i = \frac{\chi^2 \text{statistic}(i) \times n}{\sum \chi^2 \text{statistic}(i)} \quad (6)$$

3.2.3 Information gain

IG is based on the concept of entropy, which is from information theory (Witten et al. 2011). Entropy is the measure of the uncertainty of a random variable. IG measures the decrease in the

impurity of the whole data set after given the value of another feature A . The larger the IG value is, the more likely it is that the feature is relevant to the class. Turhan and Bener (2007) use this method for feature weighting. Menzies et al. (2007) use IG to perform feature ranking. Li (2012) describes in detail the principle of IG. In this paper, the weight of a feature w_i can be written as follow, where $i = 1, 2, \dots, n$, n is the number of features and D refers to the training data set.

$$w_i = \frac{ig(D, i) \times n}{\sum ig(D, i)} \quad (7)$$

However, IG has a drawback in that it tends to prefer features with a large number of different values. More specifically, if a feature has a large number of different values, $ig(D, A)$ will appear to become larger than it actually is. GR and SU are the two methods used to overcome this problem.

3.2.4 Gain ratio

GR (Quinlan 1993) can be used to alleviate the limitations of IG bias towards attributes with more values by penalizing the features with a large number of different values. The larger the GR value is, the more likely it is that the feature is relevant to the class. Zhang and Sheng (2005) use GR to assign feature weights. In this paper, the weight of a feature w_i in Eq. (3) can be written as follows:

$$w_i = \frac{gr(D, i) \times n}{\sum gr(D, i)} \quad (8)$$

where $i = 1, 2, \dots, n$, n is the number of features and D refers to the training data set.

3.2.5 Symmetrical uncertainty

SU is another way to alleviate the limitations of IG by dividing by the sum of $ig(D, A)$ and $ig(D)$ (Yu and Liu 2003). Thus, the weight of a feature w_i in Eq. (3) can be written as follows:

$$w_i = \frac{su(D, i) \times n}{\sum su(D, i)} \quad (9)$$

where $i = 1, 2, \dots, n$, n is the number of features and D refers to the training data set.

3.2.6 ReliefF (RF)

ReliefF is an instance-based feature-weighting technique that was introduced by Kira and Rendell (1992) and Robnikšikonja and Kononenko (2003), and it can also be used to reveal the relationship between the features and the class. ReliefF can measure how well a feature differentiates instances from different classes by searching for the nearest neighbors of the instances from both the same and different classes (Kononenko 1994). The larger the RF value is, the more likely it is that the feature is relevant to the class. Therefore, the weight of a feature w_i in Eq. (3) can be written as follows, where $i = 1, 2, \dots, n$, n is the number of features and D refers to the training data set.

$$w_i = \frac{rf[i] \times n}{\sum rf[i]} \quad (10)$$

3.2.7 Information flow

Considering that each software module is completed in time sequence, the software attributes are accordingly regarded as the pseudo time series (PST). Specifically, as shown in Table 2, one software release is composed of a number of modules or classes that are completed in time sequence. Thus, the numerical values of these modules for the specific feature can be seen as a PST. Then, we explain how to use IF to reveal the relationship between the features and the class.

Given two PSTs X_1 and X_2 , Liang (2014) proved the maximum likelihood estimator of the rate of the IF from X_2 to X_1 as follows:

$$T_{2 \rightarrow 1} = \frac{C_{11}C_{12}C_{2,d1} - C_{12}^2C_{1,d1}}{C_{11}^2C_{22} - C_{11}C_{12}^2} \quad (11)$$

where C_{ij} denotes the covariance between X_i and X_j . In this paper, X_1 always represents the PST of the class and X_2 denotes the one of each software attribute. $C_{i,dj}$ is determined as follows. Let \dot{X}_j be the finite-difference approximation of dX_j/dt using the Euler forward scheme: $\dot{X}_{j,n} = X_{j,n+k} - X_{j,n}/k\Delta t$ with $k = 1$ or $k = 2$ (the details about how to determine k are provided by Liang (2014)) and Δt being the time step. $C_{i,dj}$ in Eq. (11) is the covariance between X_i and \dot{X}_j . Ideally, if $T_{2 \rightarrow 1} = 0$, then X_2 does not cause X_1 ; otherwise, it is causal. To quantify the relative importance of a detected causality, Bai et al. (2018) developed an approach to normalize the IF (*NIF*) (the associated MATLAB implementation of this approach is available at <https://cn.mathworks.com/matlabcentral/fileexchange/62471-the-normalized-information-flow>). In this study, we employed Bai et al. method for our business. The closer the *NIF* is to 1, the more likely it is that the feature is relevant to the class. Therefore, the weight of a feature w_i in Eq. (3), can be calculated as follow, where $i = 1, 2, \dots, n$, and n is the number of features.

$$w_i = \frac{NIF[i] \times n}{\sum NIF[i]} \quad (12)$$

3.3 Application of the information diffusion model

As discussed in Sect. 3.2.1, the normal distribution probability density function is used in Eq. (5) for computing the likelihood that is experimentally demonstrated to be inappropriate in Sect. 1. Thus, in this stage, we apply the information diffusion model (IDM) to compute the

Table 2 Brief instruction of PST

Feature Release		Feature 1	Feature 2	...	Class
Release-1.0	module 1	a1	b1	u1
	module 2	a2	b2	u2

	module n	an	bn	un
		PST 1	PST 2		PST N

feature probability density instead of the normal distribution probability density function in Eq. (5) to alleviate the normal distribution assumption. Finally, we obtain a novel classifier and name it the weighted naive Bayes method based on information diffusion (WNB-ID). For ease of readers' understanding, we first briefly review several basic concepts of IDM referring to Huang (1997) and Bai et al. (2015).

3.3.1 Information diffusion model

Information diffusion denotes making an affirmation: when a knowledge sample is given, it can be used to calculate a relationship \mathbb{R}^1 between the population and sample. Given an instance $X = (x_1, x_2, \dots, x_m)$, U is the universe of discourse. If and only if X is incomplete, there must be a reasonable information diffusion function $\mu(x_i, u)$, $u \in U$, which can accurately compute the real relation \mathbb{R}^1 . This is the principle of information diffusion (Huang 1997). Let x_i ($i = 1, 2, \dots, m$) be independent and identically distributed instances drawn from a population with density $p(x)$, $x \in \mathbb{R}^1$. Suppose $\mu(y)$ is a Borel measurable function in $(-\infty, \infty)$, and $d > 0$ is a constant.

$$\hat{f}(x) = \frac{1}{nd} \sum_{j=1}^m \mu\left(\frac{x-x_j}{d}\right) \quad (13)$$

is an information diffusion estimator of $p(x)$, where $\mu(y)$ is the diffusion function (Huang 1997). Inspired by nearby criteria and acoustic wave propagation (i.e., the equation of the vibrating string), Bai et al. (2015) defined the diffusion function in Eq. (14), Eq. (15), and Eq. (16):

$$\mu(y) = \frac{1}{2} e^{-\frac{|y|}{d}} \quad (14)$$

$$d = \frac{k}{m-1} [\max(x) - \min(x)] \quad (15)$$

$$\hat{f}(x) = \frac{1}{2nd} \sum_{j=1}^m \exp\left(-\frac{|x-x_j|}{d^2}\right) \quad (16)$$

The value of k can be obtained from Table 3.

Equation (16) is defined as IDM or an information diffusion estimator about $p(x)$.

When we treat the training data as incomplete data, the IDM can estimate the probability density function accurately for a non-normal or normal distribution. In the next section, we evolve WNB with IDM.

3.3.2 The proposed method WNB-ID

As stated in the previous sections, the features of software modules provide different contributions to the classification, and the numeric features do not always have a normal distribution. Therefore, we use WNB-ID to improve NB in this section.

Table 3 The values of k

m	2	3	4	5	6	7
k	1.75260	1.55885	1.49338	1.46630	1.45405	1.44826
m	8	9	10	11	≥ 12	
k	1.44544	1.44406	1.44338	1.44303	1.44270	

Let $A = \{A_1, A_2, \dots, A_n\}$ be a set of software features, $X = \{(A_1, a_1), (A_2, a_2), \dots, (A_n, a_n)\}$ be a software module, and $a = \{a_i \mid i = 1, 2, \dots, n\}$ be the associated value. Let x_i ($i = 1, 2, \dots, m$) be m training software modules, and $x_{j,i}$ refers to the feature A_i of software module x_j .

The category notion of software module is defined as $C \in \{C_0, C_1\}$, where C_0 is defined as the non-defective category, and C_1 is defined as the defective one. According to Sect. 3.2.1, a software module X belonging to C_0 or C_1 will be calculated based on Eq. (3). The prior probability $P(C_k)$ in Eq. (3) can be computed by Eq. (4).

There are two differences between NB and WNB-ID.

- (1) We investigate six weight assignment methods to obtain the best w_i in Eq. (3), while NB regards the features as equally important.
- (2) The likelihood $P(a_i|C_k)$ in Eq. (3) can be computed by Eq. (17).

$$P(a_i|C_k) = \frac{1}{2nd_i} \sum_{j=1}^m \exp\left(-\frac{|a_i - x_{j,i}|}{d_i^2}\right) \quad (17)$$

where d_i is the diffusion coefficient of feature A_i . The process of WNB-ID can be described by the following algorithm.

Algorithm 2 WNB-ID algorithm.

Input: A new software module X ; The given training samples x_i ($i = 1, 2, \dots, m$)

Output: The category of X .

1. Compute the weight of each feature using the best weight assignment method that has been optimally demonstrated in Section 3.2.
 2. Calculate the diffusion coefficient d_i for each feature A_i based on Eq. (15).
 3. Compute the prior probability $P(C_k)$ based on Eq. (4).
 4. Compute the likelihood value for each feature value $P(a_i|C_k)$ in software module X using Eq.(17).
 5. Obtain the category of X based on Eq. (3).
-

4 Experimental setup

In the experimental setup, we design experiments to validate the effectiveness and practicality of the proposed method WNB-ID. First, we illustrate the data sets used in the experiments, 10×10 -fold cross-validation method, the evaluation measures, and Student's t test. Then, on the basis of the research questions, we propose the experimental design.

4.1 Data collection

In our experiments, the original 10 releases are listed in Table 4, where #Modules and #DP are the number of instances and the number of defects, respectively, %DP is the ratio of defective

Table 4 Details of the 10 data sets

Release	#Modules	#DP	%DP	#Language	#Features
Ivy-1.1	111	63	56.8	JAVA	20
Lucene-2.2	247	144	58.3	JAVA	20
Lucene-2.4	340	203	59.7	JAVA	20
Poi-1.5	237	141	59.5	JAVA	20
Poi-2.5	385	248	64.4	JAVA	20
Velocity-1.6	214	142	66.4	JAVA	20
Xalan-2.6	885	411	66.4	JAVA	20
CM1	498	49	9.83	C++	21
MW1	253	27	10.7	C++	37
PC4	1458	178	12.2	C	37

modules to all modules, #Language is the project type, and #Features are the number of metrics. Based on the proposal from He et al. (2015) and Rathore & Kumar (2017), the 10 releases are randomly chosen from eight real software applications in different programming languages provided by the PROMISE repository (Boetticher et al. 2007). The first seven releases are java projects, and the last three releases are c or c++ projects. The 21 software features of the java projects (20 code metric features and a dependent feature BUG) are listed in Table 5. Since BUG is a numeric feature, we should transform BUG into a binary classification in the following experiments. More specifically, a module is non-defective (denoted by 0) only if its number of bugs is 0. Otherwise, it is defective (denoted by 1). The features of c and c++ projects are listed in Table 6. Please note that the features shown in boldface in Table 6 are the 22 features of CM1.

Table 5 Descriptions of the features of java data sets

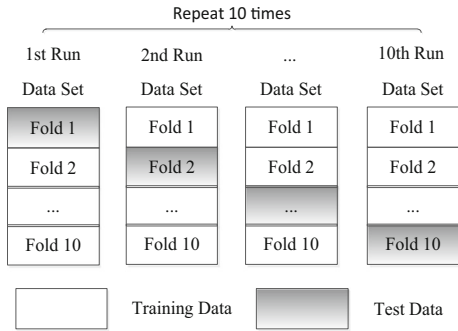
Feature	Description
WMC	Weighted Methods per Class
DIT	Depth of Inheritance Tree
NOC	Number of Children
CBO	Coupling Between Object Classes
RFC	Response for a Class
LCOM	Lack of Cohesion in Methods
CA	Afferent couplings
CE	Efferent couplings
NPM	Number of Public Methods
LCOM3	Normalized version of LCOM
LOC	Lines of code
DAM	Data access metric
MOA	Measure of aggregation
MFA	Measure of function abstraction
CAM	Cohesion among methods of class
IC	Inheritance Coupling
CBM	Coupling between methods
AMC	Average method complexity
MAX_CC	Maximum values of methods in the same class
AVG_CC	Mean values of methods in the same class
BUG	Number of bugs in the module

Table 6 Descriptions of the features of c and c++ data sets

Feature	Description
LOC_BLANK	The number of blank lines in a module.
BRANCH_COUNT	Branch count metrics.
CALL_PAIRS	Executable calls between modules (lines in Battlemat)
LOC_CODE_AND_COMMENT	The number of lines which contain both code & comment in a module.
LOC_COMMENTS	The number of lines of comments in a module.
CONDITION_COUNT	The number of conditionals in a given module
CYCOMATIC_COMPLEXITY	The cyclomatic complexity of a module.
CYCOMATIC_DENSITY	Related to “logic density” – the amount of logic per source line.
DECISION_COUNT	The number of decision points in a given module.
DECISION_DENSITY	Condition_count / decision_count
DESIGN_COMPLEXITY	The design complexity of a module.
DESIGN_DENSITY	Design Complexity (number of paths which call something) divided by Cyclomatic Complexity (total paths).
EDGE_COUNT	The number of edges in a given module.
ESSENTIAL_COMPLEXITY	The essential complexity of a module.
ESSENTIAL_DENSITY	(ESSENTIAL_COMPLEXITY-1)/ (CYCOMATIC_COMPLEXITY-1)
LOC_EXECUTABLE	The number of lines of executable code for a module (not blank or comment)
PARAMETER_COUNT	The number of parameters in a given module.
HALSTEAD_CONTENT	The Halstead length content of a module.
HALSTEAD_DIFFICULTY	The Halstead difficulty metric of a module.
HALSTEAD_EFFORT	The amount of work that it would take to recode a particular method.
HALSTEAD_ERROR_EST	The Halstead error estimate metric of a module.
HALSTEAD_LENGTH	The Halstead length metric of a module.
HALSTEAD_LEVEL	The Halstead level metric of a module.
HALSTEAD_PROG_TIME	The Halstead programming time metric of a module.
HALSTEAD_VOLUME	The Halstead volume metric of a module.
MAINTENANCE_SEVERITY	It measures how difficult it is to maintain a module.
MODIFIED_CONDITION_COUNT	The number of modified conditions in a given module.
MULTIPLE_CONDITION_COUNT	The number of multiple conditions in a given module.
NODE_COUNT	The number of nodes in a given module.
NORMALIZED_CYCOMATIC_COMPLEXITY	Normalized complexity is simply a module’s cyclomatic complexity divided by the number of lines of code in the module.
NUM_OPERANDS	The number of operands contained in a module.
NUM_OPERATORS	The number of operators contained in a module.
NUM_UNIQUE_OPERANDS	The number of unique operands contained in a module.
NUM_UNIQUE_OPERATORS	The number of unique operators contained in a module.
NUMBER_OF_LINES	Calculated as the end line minus the start line in the listing.
PERCENT_COMMENTS	The percentage of the code that is comments.
LOC_TOTAL	The total number of lines for a given module.
Defective	Buggy or not.

4.2 The 10 × 10-fold cross-validation method

As shown in Fig. 2, we use 10 × 10-fold cross-validation in all experiments with reference to Song et al. (Song et al. 2011). More specifically, we divide a data set into 10 bins, of which 9 bins are used for training and 1 for testing. All the 10 bins are from a same data set. This process performs 10 runs to ensure that each bin is used for testing to minimize sampling bias.

Fig. 2 10×10 cross-validation method

On the other hand, to reduce the ordering effect, each holdout experiment is repeated 10 times, and the ranking of data records of a data set in each iteration is randomized to achieve reliable statistics. Therefore, we perform a total of $10 \times 10 = 100$ experiments for each method, and our reported results represent the mean of these 100 experiments.

4.3 Evaluation measures

In this paper, the evaluation of prediction results is based on the *F-measure*. *F-measure* computes the accuracy based on both precision and recall, which can be explained as a weighted average of *Precision* and *Recall*:

$$F = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (18)$$

where *precision* and *recall* are two accuracy indicators, and

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (19)$$

In Eq. (19), TP (i.e., true positive) and TN (i.e., true negative) denote correctly classified defective and non-defective modules, respectively. FP (i.e., false positive) refers to non-defective modules that are incorrectly classified as defective, while FN (i.e., false negative) refers to defective modules that are incorrectly classified as non-defective.

4.4 Student's *t* test

In this paper, the performance index of each SDP model is the mean of 100 experimental results of 10×10 -fold cross-validation. Therefore, we utilize Student's *t* test to verify whether the difference between two performance indices is due only to chance. More specifically, Student's *t* test refers to the independent two-sample *t* test in this paper, which is the most common form of Student's *t* test. It examines the significance level of the difference between the means of two sets of data. The null hypothesis for Student's *t* test is that the means of two sets of data are the same, and the difference between them are due only to chance, while the alternate hypothesis is that the pair of means is different. The significance level for Student's *t* test is $\alpha = 0.05$ (i.e., at a confidence level of 95%). If the *p* value (i.e., Student's *t* test value) is less than or equal to the value of α , then the alternate hypothesis is accepted; otherwise, the null hypothesis is accepted.

4.5 Experimental design

4.5.1 Weighted naive Bayes methods

In the application of the WNB stage, we use the WNB method, which measures the degree of importance via feature-weighting techniques to alleviate the equal importance assumption in NB for SDP. To identify which feature-weighting technique performs best in WNB, we make comparative experiments among CS, IG, GR, SU, RF, and IF, which have been explained in Sect. 3.2. For each feature-weighting technique used in a given data set, we conduct 10×10 -fold cross-validation experiments and utilize Student's t test at a confidence level of 95% for the statistical significance test. Finally, the F -measure is used to choose the best feature-weighting technique in WNB. In addition to the six types of WNB methods, standard NB is also included for comparison.

For CS, IG, GR, and SU, we implement these four feature-weighting algorithms in MATLAB, and they have no extra parameters that need to set. For RF, we use the “relieff” function in the tool box of MATLAB. Parameter k (i.e., the number of nearest hits or misses) in RF and IF is maintained at 10 and 1, respectively. Parameter m (i.e., the number of iterations of the RF algorithm) is acquiescently equal to the number of modules of data sets in MATLAB.

Standard NB is one of the simplest classifier based on conditional probability (Hand and Yu 2001). In the classifier, the features are assumed to be independent, which is why the classifier is termed ‘naive’. In practice, the NB classifier often performs better than more sophisticated classifiers, although the independence assumption is often violated (Menzies et al. 2007; Hand and Yu 2001; Zaidi et al. 2013). The prediction model built by this classifier is a set of probabilities. The probability that a new instance is defective is calculated by the individual conditional probabilities for the feature values of the instance. We use standard NB with the “NaiveBayes.fit” and “predict” functions in MATLAB.

4.5.2 Information diffusion model method

This experiment is designed to verify whether IDM can alleviate the normal distribution assumption of NB, and standard NB is included for comparison. To facilitate the experiment, we name it NB-ID. In this experiment, we use the diffusion function of IDM to calculate the probability density for each feature instead of the acquiescent probability density function of the normal distribution in NB (i.e., use Eq. (17) instead of Eq. (5)). Please note that, in this experiment, feature-weighting techniques are not used. We also conduct 10×10 -fold cross-validation experiments for NB-ID and standard NB and verify whether NB-ID can improve the performance of NB in each data set.

4.5.3 Classification models

During this experiment, we implement the proposed method WNB-ID, which combines WNB and IDM to alleviate the equal importance and normal distribution assumptions in NB. We apply four other commonly used classification models of software defect prediction for comparison. They are standard naive Bayes (NB), support vector machine (SVM), logistic regression (LR), and random tree (RT). Moreover, WNB, NB-ID, and the state-of-the-art approach are also included for comparison. We also conduct 10×10 -fold cross-validation experiments for each prediction model in each data set.

Table 7 Performances of different feature-weighting techniques

Releases	NB	WNB(CS)	WNB(IG)	WNB(GR)	WNB(SU)	WNB(RF)	WNB(IF)
Ivy-1.1	0.6750	0.7018	0.7289	0.7025	0.7019	0.7347	0.7067
Lucene-2.2	0.6582	0.6013	0.7006	0.6955	0.6805	0.6868	0.6575
Lucene-2.4	0.7002	0.7258	0.7542	0.7546	0.7253	0.7290	0.7248
Poi-1.5	0.7215	0.7755	0.7703	0.7488	0.7503	0.7467	0.7832
Poi-2.5	0.7912	0.8212	0.8298	0.8005	0.8177	0.8185	0.7937
Velocity-1.6	0.7056	0.7677	0.7906	0.6978	0.7372	0.7895	0.7511
Xalan-2.6	0.6527	0.7091	0.7303	0.7325	0.7312	0.7081	0.7296
CM1	0.2964	0.2986	0.3366	0.3001	0.3105	0.3012	0.3423
MW1	0.3604	0.3625	0.3899	0.3875	0.3603	0.3621	0.3701
PC4	0.4082	0.4201	0.4642	0.4586	0.4113	0.4206	0.4321
Average (F-measure)	0.5969	0.6184	0.6495	0.6278	0.6226	0.6297	0.6291

SVM is a kind of supervised learning algorithms, which is typically applied in classification and regression analyses. It can be used to search the optimal hyper plane and maximally divide the instances into two categories (Jin and Liu 2010). We use SVM with the “svmtrain” and “svmclassify” functions in MATLAB.

LR is a kind of probabilistic statistical regression model for classification via fitting data to a logistic curve (Olague et al. 2007). It can also be used in a binary classification to predict a binary response. In SDP, the labeled feature (either defective or not) is binary; thus, it is suitable for SDP. In this paper, we use LR with the “glmfit” and “glmval” functions in MATLAB.

RT, as a hypothesis space for the supervised learning algorithm, is one of the simplest hypothesis spaces possible (Jagannathan et al. 2009). It consists of two parts: a schema and a body. The schema is a set of features, and the body is a set of labeled instances. In this paper, we use RT with the “ClassificationTree.fit” and “predict” functions in MATLAB.

4.5.4 The ensemble method

We first present a general idea of how ensemble methods perform SDP. $\mathbf{L} = \phi(\mathbf{F})$ denotes the learning function, where $\mathbf{L} = \{l_i\}_{i=1, 2, \dots, n}$ is the BUG attribute (see Table 5 and 6) and $\mathbf{F} = \{f_i\}_{i=1, 2, \dots, n}$ is a set of features. The training dataset S can train each learning algorithm for determining the associated parameters with \mathbf{L} . Then, the learning techniques can obtain the corresponding \mathbf{L} values for the unknown test datasets T . Let us call these

Table 8 Student’s t test for Ivy-1.1 (the value in boldface is greater than 0.05)

	RF \times NB	RF \times CS	RF \times IG	RF \times GR	RF \times SU	RF \times IF
p value	0.0000	0.0001	0.0913	0.0015	0.0009	0.0036

Table 9 Student’s t test for Lucene-2.2

	IG \times NB	IG \times CS	IG \times GR	IG \times SU	IG \times RF	IG \times IF
p value	0.0003	0.0000	0.0865	0.0721	0.0842	0.0001

Table 10 Student's t test for Lucene-2.4

	GR × NB	GR × CS	GR × IG	GR × SU	GR × RF	GR × IF
p value	0.0005	0.0123	0.1023	0.0081	0.0245	0.0075

corresponding outputs $\mathbf{E} = \{e_i\}_{i=1, 2, \dots, m}$. The ensemble method combines these estimated values in the linear form:

$$P = \sum_{i=1}^m w_i e_i \quad (20)$$

where w_i is the weights derived from base learner. The classification of the data sets T is determined by P .

In this paper, the state-of-the-art of the ensemble method (Tong et al. 2018) is also included for comparison. It is denoted as STSE for simplicity in this paper. STSE consists of two stages: the deep learning stage and the ensemble learning stage. In the deep learning stage, stacked denoising autoencoders are used to extract deep representations from software metrics. In the ensemble learning stage, three strong classifier, i.e., bagging, random forest, and AdaBoost, are used as base learners. The weighted average probabilities are then utilized as the combining rule of base learners. The details of the combination approach can be found in Sect. 3 from Tong et al. (2018).

5 Experimental results

In this section, based on the experimental results, we study three research questions.

5.1 *RQ1*: Which feature-weighting technique is more suitable for alleviating the equal importance assumption in WNB?

In the application of the WNB stage, we use the WNB method to alleviate the equal importance assumption of NB. To identify which feature-weighting technique performs better, we conduct experiments to compare the six feature-weighting techniques, i.e., CS, IG, GR, SU, RF, and IF. We conduct 10×10 -fold cross-validation experiments for each feature-weighting technique on each data set. In each run of the 10×10 -fold cross-validation

Table 11 Student's t test for Poi-1.5

	IF × NB	IF × CS	IF × IG	IF × GR	IF × SU	IF × RF
p value	0.0002	0.0874	0.0789	0.0142	0.0373	0.0098

Table 12 Student's t test for Poi-2.5

	IG × NB	IG × CS	IG × GR	IG × SU	IG × RF	IG × IF
p value	0.0059	0.1045	0.0182	0.0902	0.0963	0.0063

Table 13 Student's *t* test for Velocity-1.6

	IG × NB	IG × CS	IG × GR	IG × SU	IG × RF	IG × IF
<i>p</i> value	0.0001	0.0568	0.0000	0.0018	0.0978	0.0086

experiment, we first calculate the weight for each feature on the basis of nine training bins and then apply the test on the one testing bin by running the WNB algorithm with these feature weights. Finally, after determining the performance indices of each feature-weighting technique, we perform Student's *t* test at a confidence level of 95% to determine the statistical significance and thus the better feature-weighting method based on the *F-measure*. Table 7 shows the experimental results.

From Table 7, we observe that all six WNB methods perform better than standard NB with, on average, a higher *F-measure*. This finding demonstrates that WNB methods can alleviate the equal importance assumption of NB for SDP. We can also find that the WNB(IG) performs better than the other five feature-weighting techniques with, on average, the highest *F-measure*. To further verify this result, we conduct Student's *t* test at a confidence level of 95% for each release. The *F-measure* values derived from different feature-weighting techniques for each release are used to implement the tests. The test results are shown in Tables 8, 9, 10, 11, 12, 13, 14, 15, 16, and 17. In each row of Table 7, values in boldface have no significant difference. Taking Ivy-1.1 as an example, since WNB(RF) has the highest *F-measure*, 0.7347, as shown in Table 7, we conduct Student's *t* test between performances of WNB(RF) and other methods, as shown in Table 8. Table 8 shows that the *p* value of RF × IG is 0.0913, while the remainder are all less than 0.05. The results reveal no significant difference between the performances of WNB(RF) and WNB(IG), and the performance of WNB(RF) is significantly different from the others. Table 7 and Tables 8, 9, 10, 11, 12, 13, 14, 15, 16, and 17 demonstrate that WNB(IG) has, on average, the highest *F-measure*, and although the *F-measure* of WNB (IG) is not the highest in Ivy-1.1, Lucene-2.4, Poi-1.5, Xalan-2.6, and CM1, it consistently shows no significant difference from the highest one. Therefore, we select IG as the best feature-weighting technique in WNB. Table 18 shows the average feature weights measured by IG for java data sets in the 10 × 10-fold cross-validation experiments. The feature weights of the c and c++ projects are shown in Table 19. The data set CM1 has 22 features (21 code metric features and a dependent feature defective) shown in boldface in Table 6, while the data sets MW1 and PC4 have 38 features (37 code metric features and a dependent feature defective). Therefore, the weights of the features in Table 19 that do not belong to CM1 are empty.

Table 14 Student's *t* test for Xalan-2.6

	GR × NB	GR × CS	GR × IG	GR × SU	GR × RF	GR × IF
<i>p</i> value	0.0000	0.0114	0.0897	0.0918	0.0095	0.0658

Table 15 Student's *t* test for CM1

	IF × NB	IF × CS	IF × IG	IF × GR	IF × SU	IF × RF
<i>p</i> value	0.0000	0.0003	0.0729	0.0097	0.0203	0.0156

Table 16 Student's *t* test for MW1

	IG × NB	IG × CS	IG × GR	IG × SU	IG × RF	IG × IF
<i>p</i> value	0.0000	0.0071	0.0752	0.0000	0.0062	0.0297

For ease of readers' understanding, we have plotted the distribution of cumulative feature weights on the java projects in Fig. 3, from which we observe that features 'RFC', 'LOC', 'LCOM3', 'AMC', 'WMC', and 'NPM' have the highest 6 cumulative weights. Thus, they have the highest degree of importance and the most discriminative power according to the principle of IG. In contrast, the features 'NOC', 'IC', 'CA', 'MOA', 'MAX_CC', and 'AVG_CC' have the lowest six cumulative weights, i.e., they have the lowest degree of importance in the classification.

These experiments demonstrate that WNB can effectively alleviate the equal importance assumption of NB, and IG is selected as the most appropriate feature-weighting technique.

5.2 RQ2. How can IDM appropriately alleviate the normal distribution assumption to improve the performance of NB for SDP?

To investigate the effectiveness of IDM in alleviating the normal distribution assumption of NB, we first calculate the probability density for each feature by using the diffusion function of

Table 17 Student's *t* test for PC4

	IG × NB	IG × CS	IG × GR	IG × SU	IG × RF	IG × IF
<i>p</i> value	0.0000	0.0042	0.0579	0.0015	0.0097	0.0140

Table 18 Average feature weights of java data sets measured by IG in the 10 × 10-fold cross-validation experiments

Features	Ivy-1.1	Lucene-2.2	Lucene-2.4	Poi-1.5	Poi-2.5	Velocity-1.6	Xalan-2.6
WMC	1.4426	1.4606	1.4864	1.3549	1.2884	1.5286	0.9640
DIT	0.4517	0.6565	0.4717	0.8260	1.0537	0.3735	0.6139
NOC	0.5919	0.5319	0.4428	0.2281	0.0771	0.6985	0.3048
CBO	1.3210	1.3054	1.4866	0.3349	0.6262	0.7656	1.2381
RFC	1.8848	1.5741	1.6713	1.8232	1.1987	3.0263	1.4297
LCOM	1.4659	1.1551	1.1083	1.5423	1.1470	0.5241	1.1011
CA	0.6670	0.7841	0.5990	0.3429	0.4820	0.0915	0.5219
CE	1.7173	1.4051	1.5314	0.8905	0.9121	0.9290	0.5639
NPM	0.8751	1.7586	1.6410	1.5802	1.5136	0.6802	1.2404
LCOM3	1.5040	1.1977	0.6283	2.0047	1.6345	1.1223	2.0836
LOC	1.9398	1.0918	1.1059	1.4773	1.1719	2.5189	2.8353
DAM	0.1679	1.2089	1.4765	1.1480	1.8305	1.3718	0.3397
MOA	0.3816	0.8806	0.2374	0.6084	0.2168	0.8847	0.3895
MFA	1.2148	1.0247	1.0370	1.1167	1.3794	0.5734	0.4881
CAM	0.9968	0.5434	1.3593	0.7986	0.9423	1.3281	1.5591
IC	0.3741	0.2887	0.7001	0.5556	0.7882	0.0072	0.3472
CBM	0.4808	0.8373	0.9931	1.2894	1.8900	0.3184	0.4841
AMC	1.3280	0.9683	0.5757	1.3247	1.1581	1.9468	2.4725
MAX_CC	0.2278	0.9896	0.8076	0.2830	0.5205	0.4224	0.5718
AVG_CC	0.9672	0.3374	0.6407	0.4706	0.1691	0.8886	0.4513

Table 19 Average feature weights of c and c++ projects measured by IG in the 10×10 -fold cross-validation experiments

Features	CM1	MW1	PC4
LOC_BLANK	0.6010	1.6930	1.3290
BRANCH_COUNT	0.9768	1.2531	0.2022
CALL_PAIRS	—	1.6202	0.7116
LOC_CODE_AND_COMMENT	0.2423	0.0543	4.3327
LOC_COMMENTS	1.5130	1.2080	2.1292
CONDITION_COUNT	—	1.1657	0.1513
CYCLOMATIC_COMPLEXITY	0.7125	1.6037	0.1642
CYCLOMATIC_DENSITY	—	0.2951	1.1045
DECISION_COUNT	—	1.4136	0.1766
DECISION_DENSITY	—	0.2270	3.4084
DESIGN_COMPLEXITY	1.2966	1.1673	0.2157
DESIGN_DENSITY	—	0.7325	0.5535
EDGE_COUNT	—	1.8287	0.3410
ESSENTIAL_COMPLEXITY	0.5406	0.9236	0.3128
ESSENTIAL_DENSITY	—	0.0357	0.3143
LOC_EXECUTABLE	1.0711	1.3468	1.3261
PARAMETER_COUNT	1.0326	0.0899	1.0035
HALSTEAD_CONTENT	1.9619	1.2677	0.1690
HALSTEAD_DIFFICULTY	0.8527	0.2919	0.7822
HALSTEAD_EFFORT	0.5329	0.6896	0.3436
HALSTEAD_ERROR_EST	—	1.5399	0.5302
HALSTEAD_LENGTH	1.1381	1.3542	1.0685
HALSTEAD_LEVEL	0.5795	0.2743	0.8698
HALSTEAD_PROG_TIME	0.7364	0.6642	0.2603
HALSTEAD_VOLUME	1.0747	1.3727	0.3925
MAINTENANCE_SEVERITY	—	0.3650	1.2879
MODIFIED_CONDITION_COUNT	—	1.2830	0.2091
MULTIPLE_CONDITION_COUNT	—	1.1291	0.2078
NODE_COUNT	—	1.8288	0.3799
NORMALIZED_CYLOMATIC_COMPLEXITY	—	0.2106	2.2320
NUM_OPERANDS	1.0010	1.1410	0.5750
NUM_OPERATORS	1.4678	1.5571	0.8877
NUM_UNIQUE_OPERANDS	1.4436	1.7972	0.2279
NUM_UNIQUE_OPERATORS	1.6598	0.5383	1.1002
NUMBER_OF_LINES	—	1.2299	1.1467
PERCENT_COMMENTS	—	0.4649	4.2054
LOC_TOTAL	1.1497	1.3427	2.3478

IDM and then apply this new probability density in NB. We name it NB-ID in this experiment. We then conduct comparison experiments with standard NB using 10×10 -fold cross-validation on the 10 data sets.

Table 20 and Figs. 4, 5, 6, 7, 8, 9, 10, 11, 12, and 13 show the experimental results. Each value in Table 20 represents the mean of 100 experimental results. “R”, “P”, and “F” in Figs. 4, 5, 6, 7, 8, 9, 10, 11, 12, and 13 refer to *Recall*, *Precision*, and *F-measure*, respectively. Each figure in Figs. 4, 5, 6, 7, 8, 9, 10, 11, 12, and 13 depicts the *Recall*, *Precision*, and *F-measure* of 100 experimental results based on their quartiles. From the bottom to the top of a standardized box plot: minimum, first quartile, median, third quartile, and maximum. Any outlier data are plotted as a small cross. Taking Fig. 4 as an example, considering the red median line, we find that NB-ID performs better than NB in *Recall*, *Precision*, and *F-measure*. Compared with standard NB, NB-ID performs better in *Recall*, *Precision*, and *F-measure* in

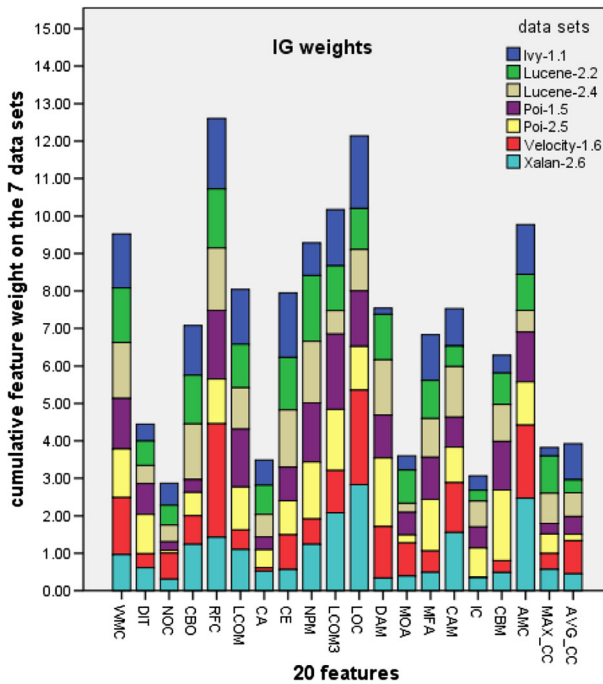


Fig. 3 The distribution of cumulative feature weight on java data sets

Table 20 Comparison between NB-ID and standard NB on 10 data sets

Release	Predictor	Recall	Precision	F-measure
Ivy-1.1	NB	0.6324	0.7345	0.6750
	NB-ID	0.7864	0.7499	0.7627
Lucene-2.2	NB	0.6510	0.6717	0.6582
	NB-ID	0.8364	0.6381	0.7228
Lucene-2.4	NB	0.6640	0.7456	0.7002
	NB-ID	0.8559	0.7132	0.7766
Poi-1.5	NB	0.7588	0.6909	0.7215
	NB-ID	0.8611	0.7546	0.8035
Poi-2.5	NB	0.7935	0.7906	0.7912
	NB-ID	0.8966	0.7937	0.8411
Velocity-1.6	NB	0.6233	0.8207	0.7056
	NB-ID	0.9338	0.7495	0.8304
Xalan-2.6	NB	0.6560	0.6521	0.6527
	NB-ID	0.7437	0.7779	0.7595
CM1	NB	0.4297	0.2272	0.2964
	NB-ID	0.6285	0.2395	0.3403
MW1	NB	0.5956	0.2599	0.3604
	NB-ID	0.6667	0.2793	0.3898
PC4	NB	0.6451	0.2995	0.4082
	NB-ID	0.6734	0.3706	0.4726

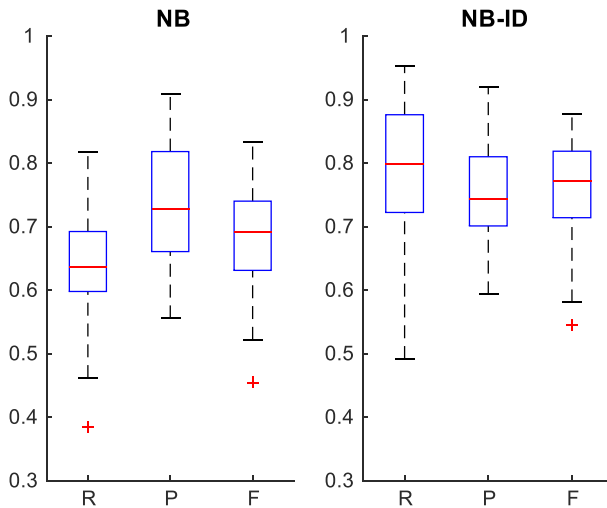


Fig. 4 The standardized boxplots of the performances of NB-ID and standard NB on the data set Ivy-1.1 From the bottom to the top of a standardized box plot: minimum, first quartile, median, third quartile, and maximum. Any outlier data are plotted as a small cross. “R”, “P”, and “F” refer to *Recall*, *Precision*, and *F-measure*, respectively

the data sets of Ivy-1.1, Poi-1.5, Poi-2.5, Xalan-2.6, CM1, MW1, and PC4. In the data sets of Lucene-2.2, Lucene-2.4, and Velocity-1.6, NB-ID performs better in *Recall* and *F-measure*, while standard NB performs better in *Precision*. In brief, NB-ID can always obtain better *F-measure* statistics than the standard NB, demonstrating the ability of IDM to alleviate the normal distribution assumption of NB, considering that the *F-measure* integrates *Recall* and *Precision* in a single indicator.

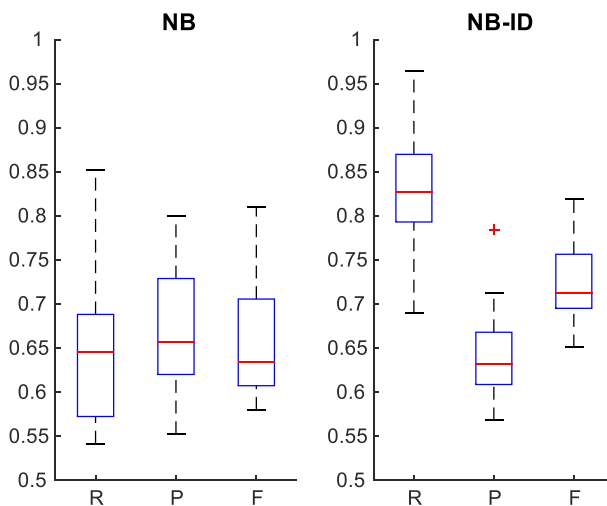


Fig. 5 The standardized boxplots of the performances of NB-ID and standard NB on the data set Lucene-2.2

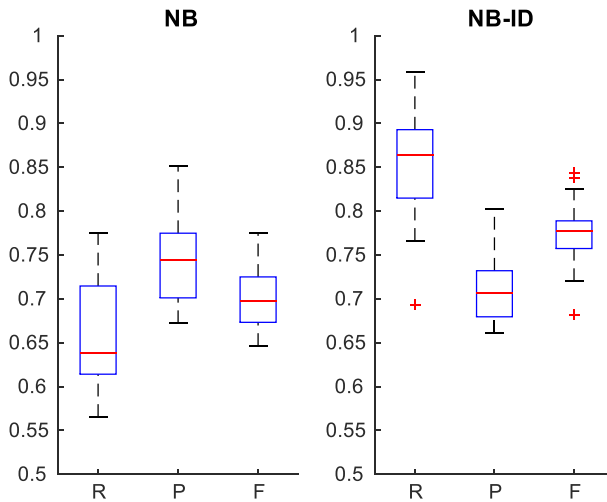


Fig. 6 The standardized boxplots of the performances of NB-ID and standard NB on the data set Lucene-2.4

5.3 RQ3: What is the defect prediction performance of WNB-ID compared with other prediction methods?

To demonstrate the overall effectiveness of WNB-ID, we combine WNB and IDM and make comparison among NB, SVM, LR, RT, and WNB-ID on 10 data sets. Please note that we use the best feature-weighting technique IG in WNB, which has been obtained in RQ1. We also make comparison among WNB (IG), NB-ID, STSE, and WNB-ID, which have been demonstrated to be effective in RQ1, RQ2, and Tong et al. (2018), respectively.

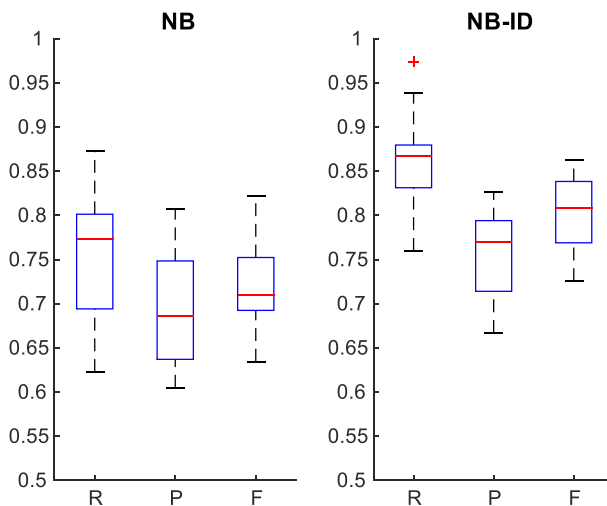


Fig. 7 The standardized boxplots of the performances of NB-ID and standard NB on the data set Poi-1.5

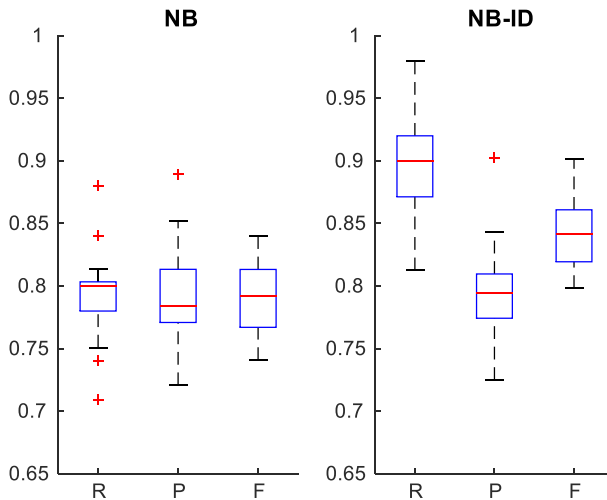


Fig. 8 The standardized boxplots of the performances of NB-ID and standard NB on the data set Poi-2.5

Table 21 and Figs. 14, 15, 16, 17, 18, 19, 20, 21, 22, and 23 show the experimental results of the comparison among NB, SVM, LR, RT, and WNB-ID. WNB-ID performs better than the others, especially obviously in Ivy-1.1, Lucene-2.2, Poi-1.5, and Velocity-1.6. Although WNB-ID is not the best in *Recall* or *Precision* in Lucene-2.4, Poi-2.5, Xalan-2.6, CM1, MW1, and PC4, the aggregative indicator *F-Measure* is consistently better than the others in these data sets.

To further verify this result, we conduct Student's *t* test at a confidence level of 95% for each release to determine whether WNB-ID shows significant difference in performance from the other methods. Considering that *F-measure* integrates *Recall* and *Precision* in a single

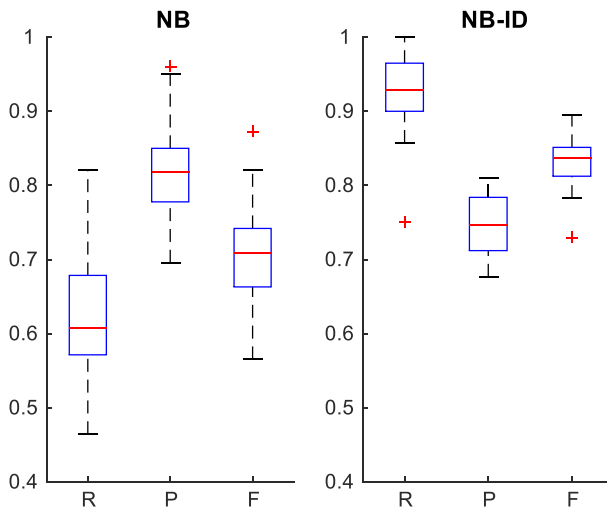


Fig. 9 The standardized boxplots of the performances of NB-ID and standard NB on the data set Velocity-1.6

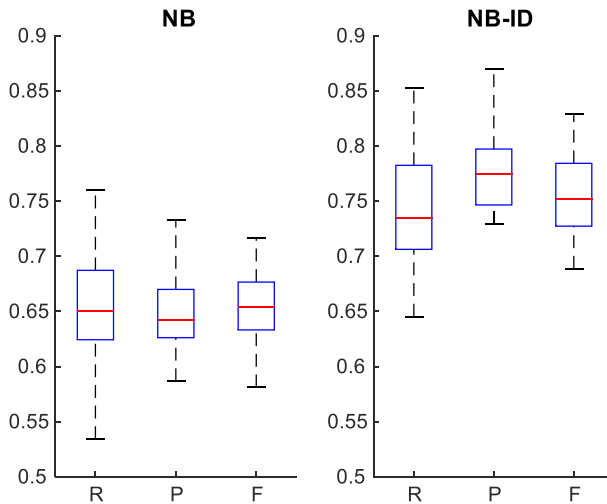


Fig. 10 The standardized boxplots of the performances of NB-ID and standard NB on the data set Xalan-2.6

indicator, *F-measure* values derived from different classifiers for each release are used to implement the tests. The test results are listed in Table 22. We can see that the p values are all less than 0.05. The test results reveal that the performances of WNB-ID and other classic methods are significantly different. Therefore, we think that WNB-ID is superior to NB, SVM, LR, and RT.

Table 23 and Figs. 24, 25, 26, 27, 28, 29, 30, 31, 32, and 33 show the experimental results of the comparison among WNB (IG), NB-ID, STSE, and WNB-ID. We find that WNB-ID

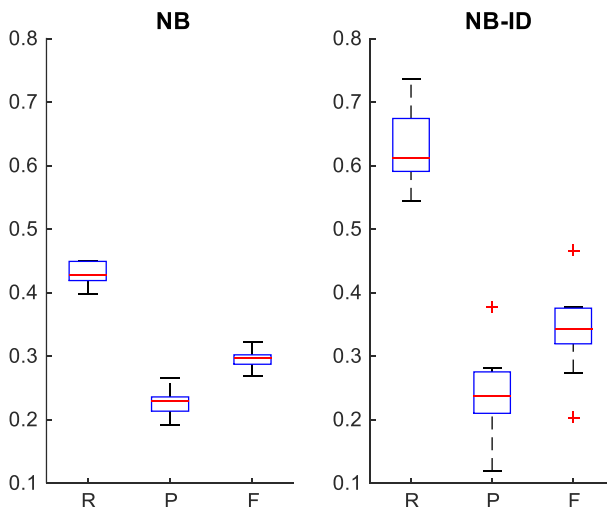


Fig. 11 The standardized boxplots of the performances of NB-ID and standard NB on the data set CM1

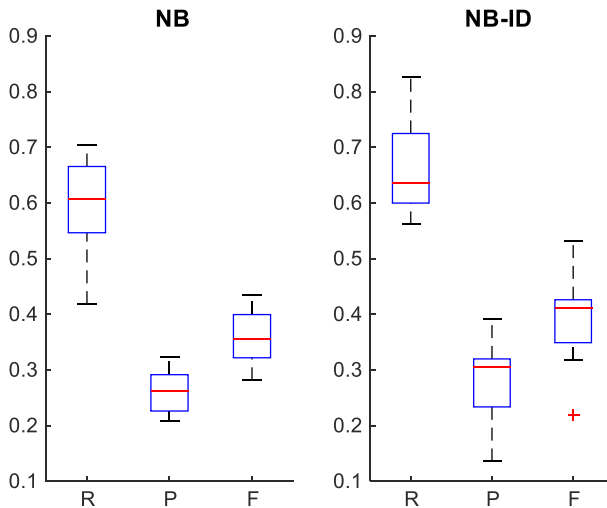


Fig. 12 The standardized boxplots of the performances of NB-ID and standard NB on the data set MW1

performs better than the others obviously in the data sets of Ivy-1.1, Poi-1.5, and Poi-2.5. Although WNB-ID is not the best in *Recall* or *Precision* in Lucene-2.2, Lucene-2.4, Velocity-1.6, Xalan-2.6, CM1, and MW1, the aggregative indicator *F-Measure* is consistently better than the others in these data sets. In data set PC4, the *F-measure* of WNB-ID is less than STSE, but it is also comparable to STSE.

We also conduct Student's *t* test at a confidence level of 95% for each release. The test results are listed in Table 24. We can see that the *p* values between WNB-ID and STSE in data

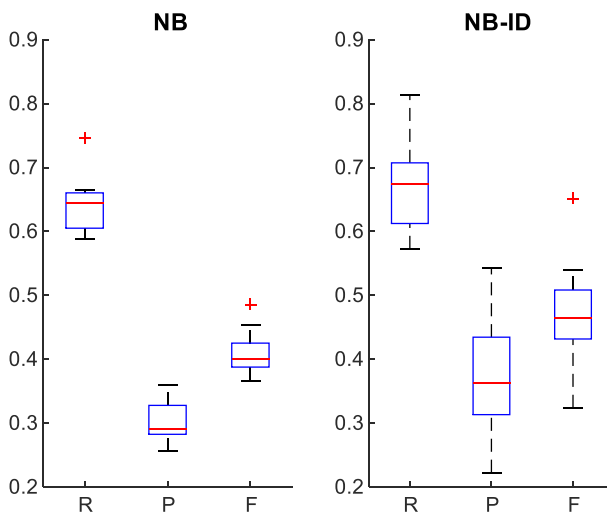


Fig. 13 The standardized boxplots of the performances of NB-ID and standard NB on the data set PC4

Table 21 Comparisons among NB, SVM, LR, RT, and WNB-ID on the 10 data sets

Release	Predictor	Recall	Precision	F-measure
Ivy-1.1	NB	0.6324	0.7345	0.6750
	SVM	0.5989	0.7265	0.6579
	LR	0.6873	0.7224	0.6944
	RT	0.6579	0.6782	0.6602
	WNB-ID	0.8177	0.7748	0.7907
Lucene-2.2	NB	0.6510	0.6717	0.6582
	SVM	0.5510	0.6609	0.6068
	LR	0.7276	0.6522	0.6853
	RT	0.6538	0.6410	0.6454
	WNB-ID	0.8275	0.6735	0.7412
Lucene-2.4	NB	0.6640	0.7456	0.7002
	SVM	0.6637	0.7841	0.7157
	LR	0.7737	0.7432	0.7565
	RT	0.7353	0.7041	0.7173
	WNB-ID	0.8424	0.7418	0.7872
Poi-1.5	NB	0.7588	0.6909	0.7215
	SVM	0.4523	0.7729	0.7232
	LR	0.8100	0.7297	0.7662
	RT	0.6908	0.7237	0.7038
	WNB-ID	0.9094	0.7827	0.8406
Poi-2.5	NB	0.7935	0.7906	0.7912
	SVM	0.6195	0.8126	0.7490
	LR	0.8539	0.8141	0.8327
	RT	0.8330	0.8343	0.8322
	WNB-ID	0.9170	0.8234	0.8669
Velocity-1.6	NB	0.6233	0.8207	0.7056
	SVM	0.6670	0.8008	0.7198
	LR	0.8321	0.7629	0.7937
	RT	0.7927	0.8036	0.7954
	WNB-ID	0.8947	0.8472	0.8684
Xalan-2.6	NB	0.6560	0.6521	0.6527
	SVM	0.2427	0.8656	0.4226
	LR	0.6647	0.7254	0.6930
	RT	0.6808	0.7120	0.6949
	WNB-ID	0.7676	0.7651	0.7651
CM1	NB	0.4297	0.2272	0.2964
	SVM	0.5757	0.2105	0.3071
	LR	0.0767	0.5827	0.1336
	RT	0.2557	0.5120	0.3367
	WNB-ID	0.6411	0.2840	0.3901
MW1	NB	0.5956	0.2599	0.3604
	SVM	0.7395	0.2605	0.3840
	LR	0.5027	0.3063	0.3919
	RT	0.4417	0.3281	0.3801
	WNB-ID	0.6124	0.3793	0.4641
PC4	NB	0.6451	0.2995	0.4082
	SVM	0.8933	0.2414	0.3791
	LR	0.2790	0.4420	0.3426
	RT	0.4775	0.4260	0.4470
	WNB-ID	0.6460	0.4497	0.5243

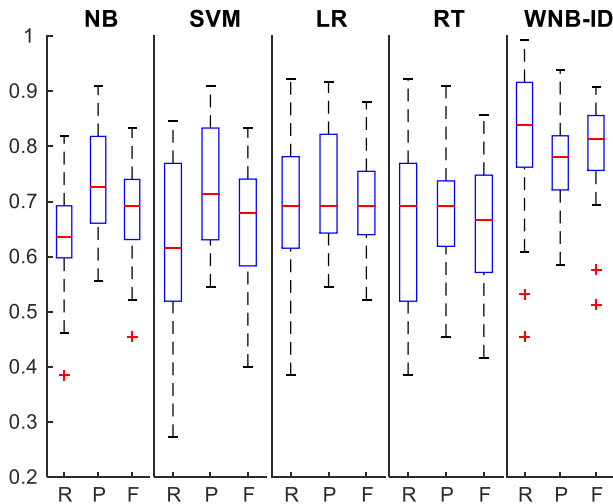


Fig. 14 The standardized boxplots of the performances of NB, SVM, LR, RT, and WNB-ID on the data set Ivy-1.1. From the bottom to the top of a standardized box plot: minimum, first quartile, median, third quartile, and maximum. Any outlier data are plotted as a small cross. “R”, “P”, and “F” refer to *Recall*, *Precision*, and *F-measure*, respectively

sets Poi-1.5, Poi-2.5, and Xalan-2.6 are greater than 0.05. The test results reveal that the performances of WNB-ID and STSE are not significantly different in these data sets. As shown in Table 24, the performances of WNB-ID and NB-ID also show no significant differences in the data set Xalan-2.6.

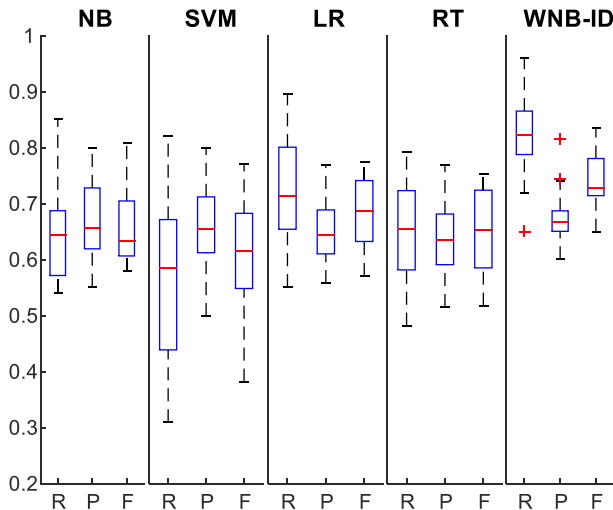


Fig. 15 The standardized boxplots of the performances of NB, SVM, LR, RT, and WNB-ID on the data set Lucene-2.2

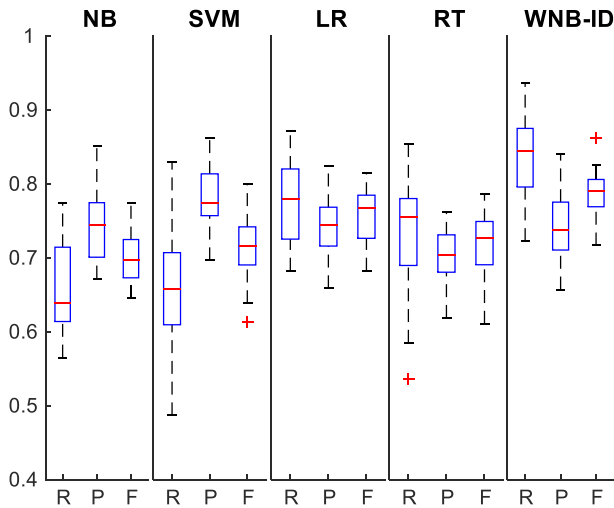


Fig. 16 The standardized boxplots of the performances of NB, SVM, LR, RT, and WNB-ID on the data set Lucene-2.4

In summary, considering the aggregative indicator *F-measure*, WNB-ID performs better than WNB(IG) in all data sets. WNB-ID shows no significant difference from NB-ID in the data set Xalan-2.6, and it performs better than NB-ID in the other data sets. WNB-ID performs better than STSE in the 6 data sets, shows no significant differences from STSE in the three data sets, and is not as effective as STSE in PC4. Therefore, we think that WNB-ID is superior to WNB (IG), NB-ID, and STSE.

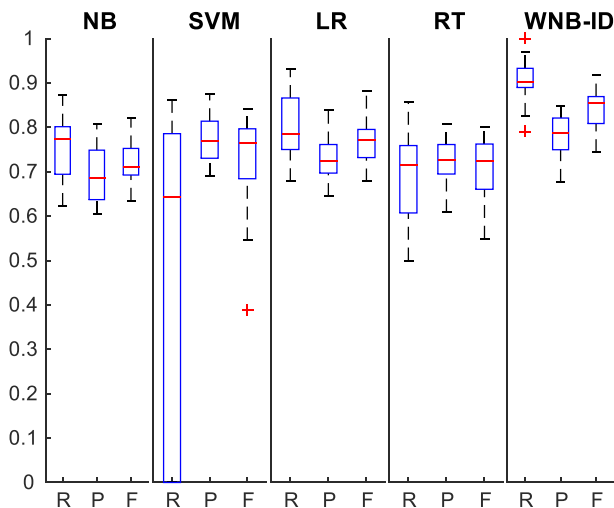


Fig. 17 The standardized boxplots of the performances of NB, SVM, LR, RT, and WNB-ID on the data set Poi-1.5

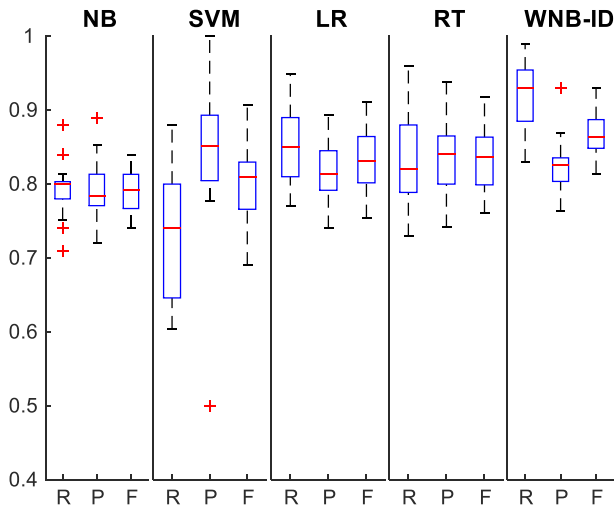


Fig. 18 The standardized boxplots of the performances of NB, SVM, LR, RT, and WNB-ID on the data set Poi-2.5

Additionally, based on Table 23, our new method achieves poorer results than the others in some situations. The main reasons comprise three aspects. (1) WNB(IG) and NB-ID are the two components of WNB-ID. As suggested by Tong et al. (2018), the construction of an SDP model that is more complex than it used to be may cause over fitting, which can affect the effectiveness of the new method. This may be the explanation why the *Recall* or *Precision* of WNB-ID are not superior in some releases. (2) Although WNB-ID achieves better

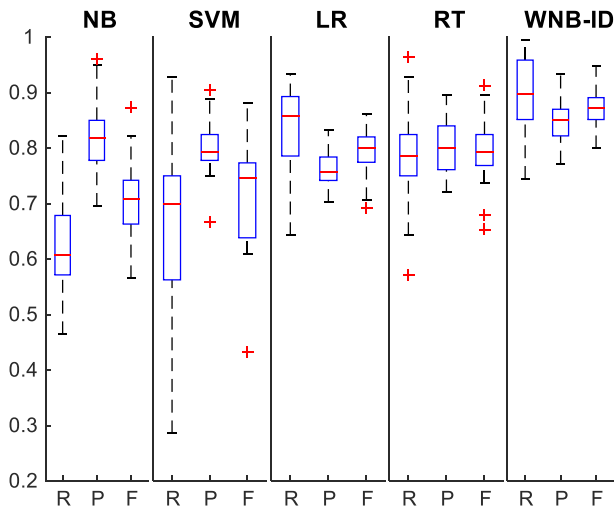


Fig. 19 The standardized boxplots of the performances of NB, SVM, LR, RT, and WNB-ID on the data set Velocity-1.6

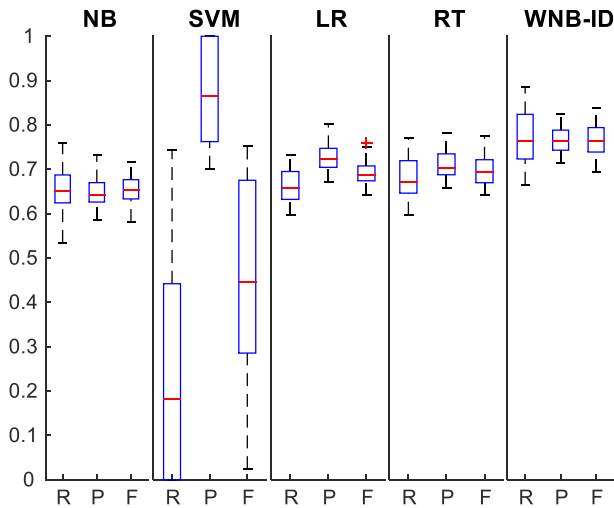


Fig. 20 The standardized boxplots of the performances of NB, SVM, LR, RT, and WNB-ID on the data set Xalan-2.6

performance in most releases, both balanced and imbalanced, ensemble methods may be more immune to class imbalance problems in some conditions, potentially explaining the superior performance of STSE than WNB-ID in PC4. (3) The quantity of software modules in PC4 clearly exceeds the others, while NB is effective at dealing with data sets having a small data quantity (Zaidi et al. 2013). This feature may be another reason why STSE performs better than WNB-ID in PC4.

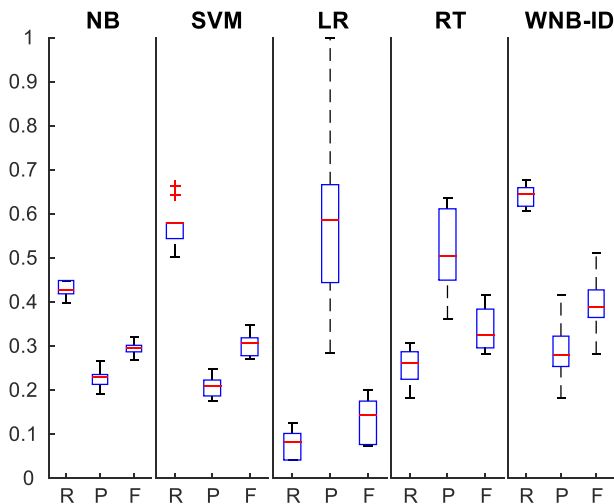


Fig. 21 The standardized boxplots of the performances of NB, SVM, LR, RT, and WNB-ID on the data set CM1

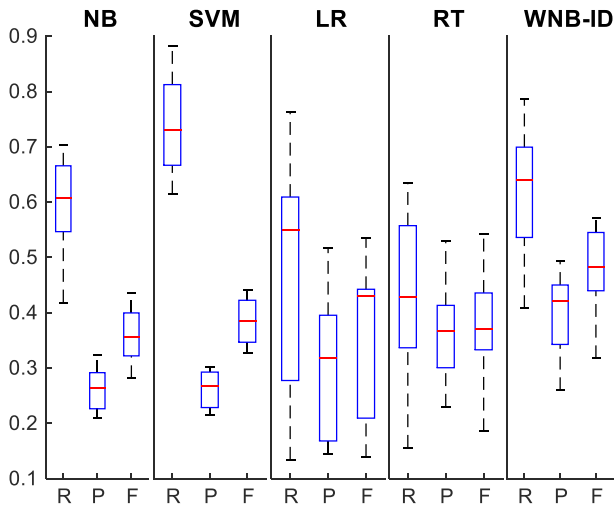


Fig. 22 The standardized boxplots of the performances of NB, SVM, LR, RT, and WNB-ID on the data set MW1

6 Threats to validity

This paper focuses on how to boost the potential of NB with the feature-weighting technique and IDM for SDP. One possible threat to the validity of the proposed WNB-ID method may be the parameter settings. In feature-weighting techniques, these settings influence the prediction

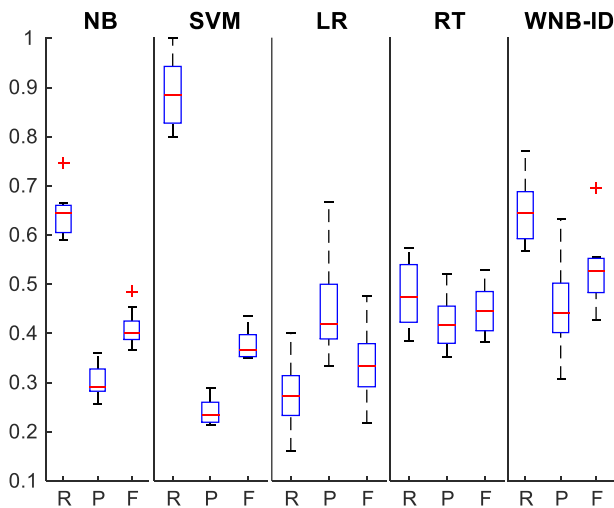


Fig. 23 The standardized boxplots of the performances of NB, SVM, LR, RT, and WNB-ID on the data set PC4

Table 22 Student's *t* test between WNB-ID and NB, SVM, LR, and RT

Release	NB	SVM	LR	RT
Ivy-1.1	0.0000	0.0000	0.0000	0.0000
Lucene-2.2	0.0001	0.0000	0.0002	0.0000
Lucene-2.4	0.0000	0.0000	0.0000	0.0035
Poi-1.5	0.0000	0.0021	0.0052	0.0000
Poi-2.5	0.0000	0.0000	0.0003	0.0001
Velocity-1.6	0.0000	0.0000	0.0000	0.0000
Xalan-2.6	0.0000	0.0000	0.0000	0.0000
CM1	0.0000	0.0000	0.0000	0.0108
MW1	0.0000	0.0000	0.0000	0.0000
PC4	0.0000	0.0000	0.0000	0.0047

Table 23 Comparisons among WNB (IG), NB-ID, STSE, and WNB-ID on the 10 data sets

Release	Predictor	Recall	Precision	F-measure
Ivy-1.1	WNB (IG)	0.7110	0.7617	0.7289
	NB-ID	0.7864	0.7499	0.7627
	STSE	0.7594	0.7707	0.7601
	WNB-ID	0.8177	0.7748	0.7907
Lucene-2.2	WNB (IG)	0.7108	0.6947	0.7006
	NB-ID	0.8364	0.6381	0.7228
	STSE	0.7607	0.6950	0.7060
	WNB-ID	0.8275	0.6735	0.7412
Lucene-2.4	WNB (IG)	0.7413	0.7712	0.7542
	NB-ID	0.8559	0.7132	0.7766
	STSE	0.7878	0.7413	0.7617
	WNB-ID	0.8424	0.7418	0.7872
Poi-1.5	WNB (IG)	0.8276	0.7224	0.7703
	NB-ID	0.8611	0.7546	0.8035
	STSE	0.8770	0.7822	0.8251
	WNB-ID	0.9094	0.7827	0.8406
Poi-2.5	WNB (IG)	0.8423	0.8193	0.8298
	NB-ID	0.8966	0.7937	0.8411
	STSE	0.8975	0.8208	0.8563
	WNB-ID	0.9170	0.8234	0.8669
Velocity-1.6	WNB (IG)	0.7631	0.8254	0.7906
	NB-ID	0.9338	0.7495	0.8304
	STSE	0.8065	0.7719	0.7856
	WNB-ID	0.8947	0.8472	0.8684
Xalan-2.6	WNB (IG)	0.7951	0.6763	0.7303
	NB-ID	0.7437	0.7779	0.7595
	STSE	0.7471	0.7801	0.7620
	WNB-ID	0.7676	0.7651	0.7651
CM1	WNB (IG)	0.4603	0.2714	0.3366
	NB-ID	0.6285	0.2395	0.3403
	STSE	0.2280	0.4289	0.2885
	WNB-ID	0.6411	0.2840	0.3901
MW1	WNB (IG)	0.6062	0.2912	0.3899
	NB-ID	0.6667	0.2793	0.3898
	STSE	0.4683	0.4864	0.4368
	WNB-ID	0.6124	0.3793	0.4641
PC4	WNB (IG)	0.6067	0.3826	0.4642
	NB-ID	0.6734	0.3706	0.4726
	STSE	0.5576	0.5871	0.5599
	WNB-ID	0.6460	0.4497	0.5243

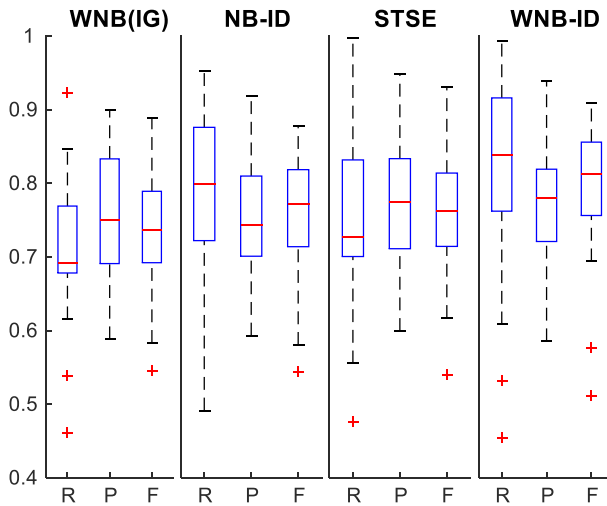


Fig. 24 The standardized boxplots of the performances of WNB (IG), NB-ID, and WNB-ID on the data set Ivy-1.1

performance of WNB-ID. Appropriate values can enhance the efficiency of the algorithm, but how to choose these values is not the emphasis of this study. Hence, we use the acquiescent values to compare the results of WNB-ID on different data sets, which means that the

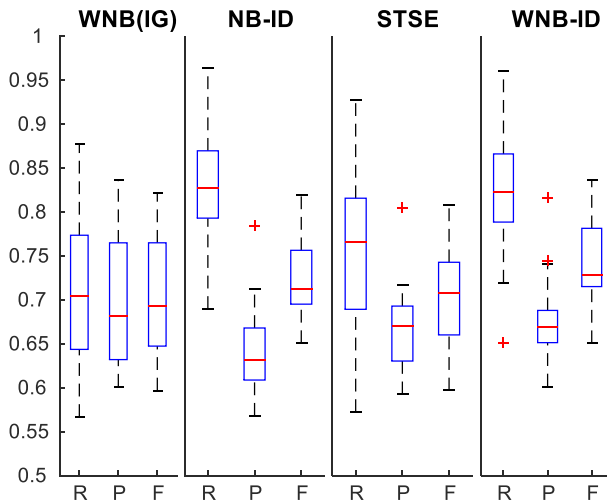


Fig. 25 The standardized boxplots of the performances of WNB (IG), NB-ID, and WNB-ID on the data set Lucene-2.2

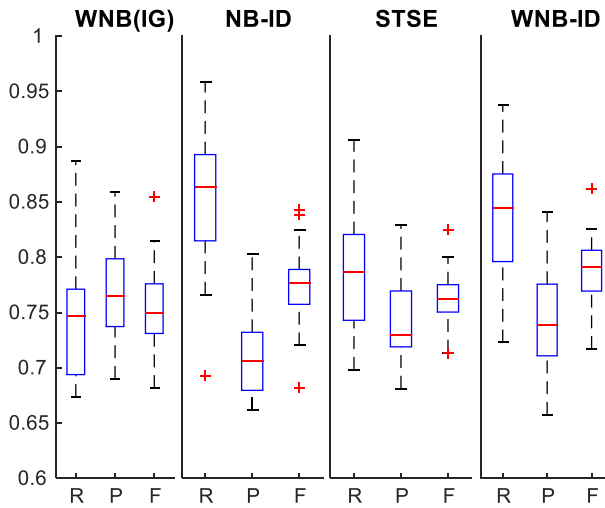


Fig. 26 The standardized boxplots of the performances of WNB (IG), NB-ID, and WNB-ID on the data set Lucene-2.4

performance of WNB-ID is not ideal for some data sets. The second threat to validity may be the distributions of values of the features. The experiments show that the distributions of values of features have a significant influence on the performance of WNB-ID. If the

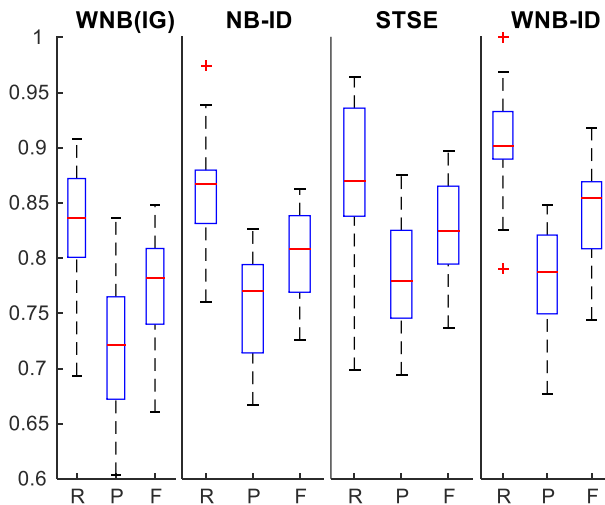


Fig. 27 The standardized boxplots of the performances of WNB (IG), NB-ID, and WNB-ID on the data set Poi-1.5

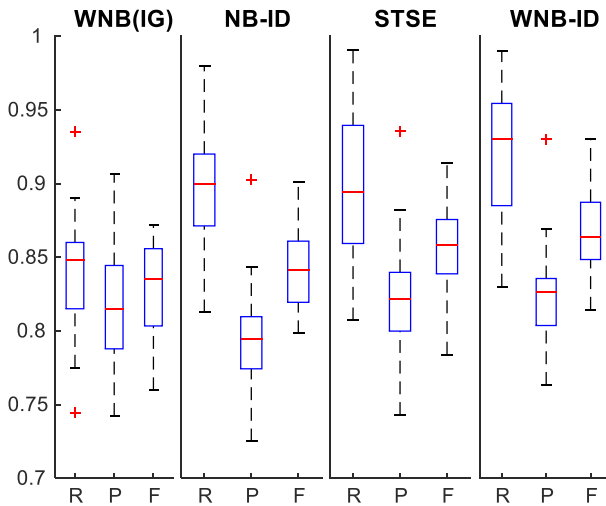


Fig. 28 The standardized boxplots of the performances of WNB (IG), NB-ID, and WNB-ID on the data set Poi-2.5

distributions of values of features follow a normal distribution, the proposed method WNB-ID may perform as WNB(IG). The third threat to validity may be the limited data sets. The data sets used in this paper are from the PROMISE repository. Different conclusions may be drawn by using different data sets. However, most of these data sets have been widely used in many other SDP studies, providing us and other researchers with the possibility of comparing results.

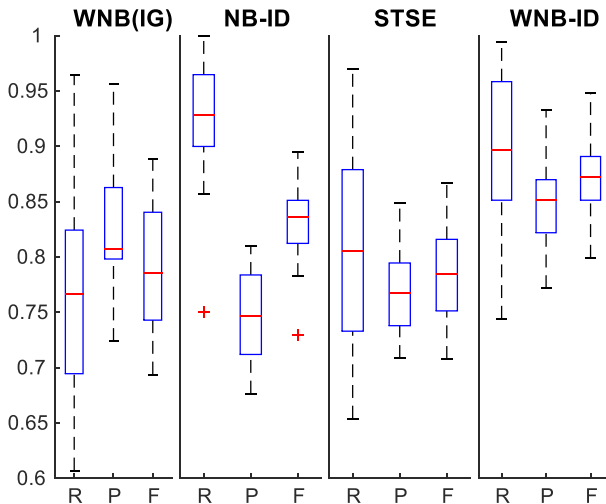


Fig. 29 The standardized boxplots of the performances of WNB (IG), NB-ID, and WNB-ID on the data set Velocity-1.6

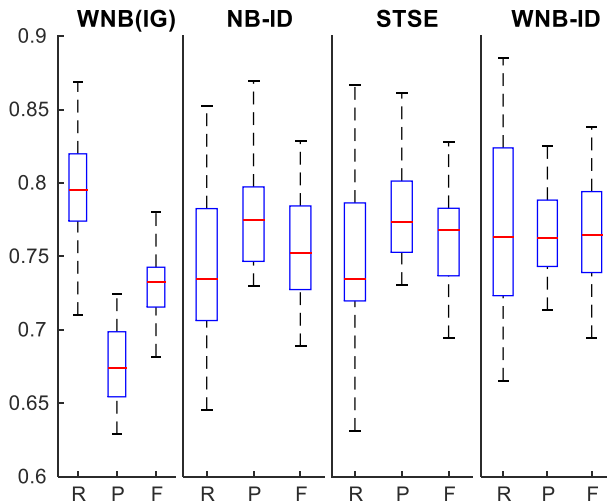


Fig. 30 The standardized boxplots of the performances of WNB (IG), NB-ID, and WNB-ID on the data set Xalan-2.6

7 Discussion

In this study, feature-weighting techniques and IDM are combined to improve NB in SDP for the first time. The advantage of the proposed method is that the equal importance and normal distribution assumptions of NB, which are often violated in common practice, are alleviated by applying feature-weighting techniques and IDM, respectively. Through a series of experiments, the proposed methods are shown to be effective in improving NB for SDP:

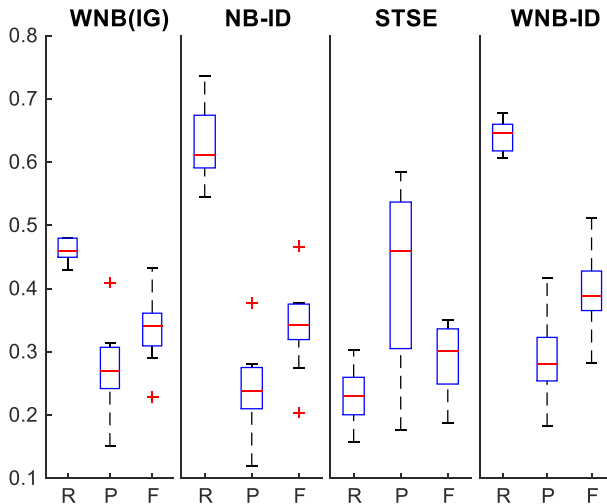


Fig. 31 The standardized boxplots of the performances of WNB (IG), NB-ID, STSE, and WNB-ID on the data set CM1

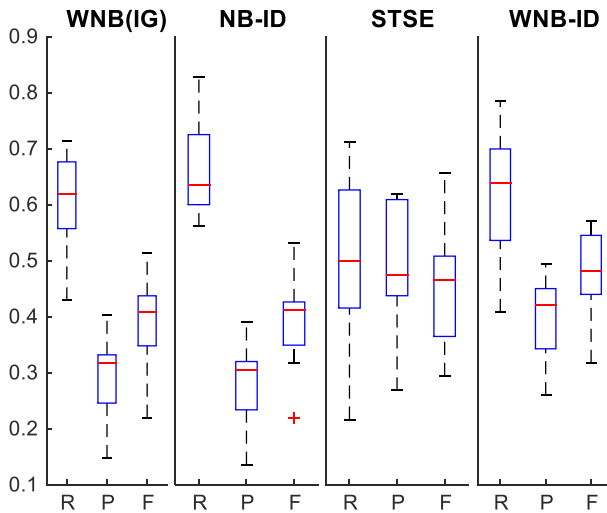


Fig. 32 The standardized boxplots of the performances of WNB (IG), NB-ID, STSE, and WNB-ID on the data set MW1

- For the application of feature-weighting techniques, we find that WNB can effectively alleviate the equal importance assumption of NB, and the feature-weighting method IG performs best in WNB.
- For the application of IDM, after comparison with standard NB, we find that it is sufficiently competent when the feature values of defect data sets are non-normally distributed.

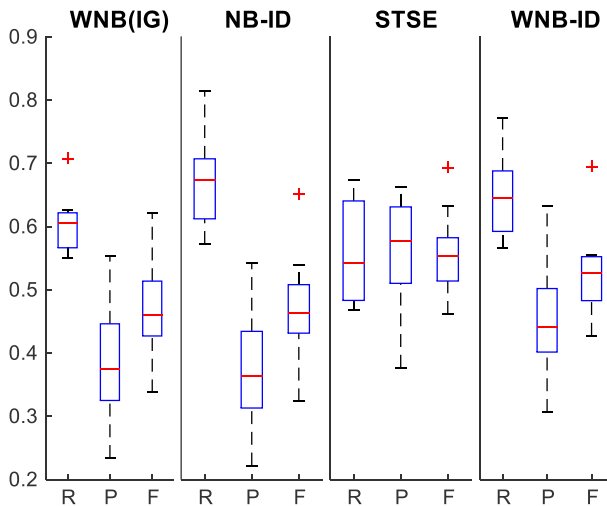


Fig. 33 The standardized boxplots of the performances of WNB (IG), NB-ID, STSE, and WNB-ID on the data set PC4

Table 24 Student's *t* test between WNB-ID and WNB(IG), NB-ID, and STSE

Release	WNB(IG)	NB-ID	STSE
Ivy-1.1	0.0065	0.0302	0.0215
Lucene-2.2	0.0108	0.0203	0.0154
Lucene-2.4	0.0000	0.0139	0.0097
Poi-1.5	0.0000	0.0168	0.0637
Poi-2.5	0.0017	0.0092	0.0603
Velocity-1.6	0.0000	0.0000	0.0000
Xalan-2.6	0.0000	0.0653	0.0710
CM1	0.0000	0.0000	0.0000
MW1	0.0000	0.0000	0.0315
PC4	0.0137	0.0165	0.0420

- For the proposed classification model WNB-ID, we observe that it is better than the classic classification models, i.e., NB, SVM, LR, and RT, and it is also superior to WNB(IG), NB-ID, and the state-of-the-art.

8 Conclusion

In this paper, aiming to alleviate the equal importance and normal distribution assumptions of NB, we design and implement a new NB method, WNB-ID, which includes the application of WNB and of IDM. For the application of WNB, we investigate six feature-weighting techniques to establish the feature weights to alleviate the equal importance assumption of NB. For the application of IDM, we use IDM to compute the probability density of each feature instead of the acquiescent probability density function of the normal distribution to alleviate the normal distribution assumption of NB. To validate the effectiveness and practicality of our approach, we systematically conduct verification experiments. Several classic methods and state-of-the-art are also included for comparison. The final experimental results demonstrate the potential of our approach in improving the performance of SDP. In future work, we will examine more feature-weighting techniques and optimize information diffusion models to further improve the potential of NB in SDP. In addition, we plan to apply this method to cross-project defect prediction, which is now a research hotspot in the field of SDP (Yu et al. 2017; Herbold et al. 2017).

Acknowledgements The authors would like to thank the anonymous reviewers for their constructive comments.

Funding information This work is supported by the National Natural Science Foundation of China (Grant No. 61702544) and the Natural Science Foundation of Jiangsu Province of China (Grant No. BK20160769).

Compliance with ethical standards

Conflict of interest The authors declare that there are no conflict of interests regarding the publication of this paper.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

- Aman, H., Amasaki, S., Sasaki, T., Kawahara, M. (2015). Lines of comments as a noteworthy metric for analyzing faultproneeness in methods. *IEICE transactions on Information & Systems*, vol. E98.D, no. 12, pp. 2218–2228.
- Arar, Ö. F., & Ayan, K. (2017). A feature dependent naive Bayes approach and its application to the software defect prediction problem. *Applied Soft Computing*, 59, 197–209.
- Boetticher, G., Menzies, T., Ostrand, T. J. (2007). The promise repository of empirical software engineering data. [online]. Available: <http://openscience.us/repo>.
- Bai, C., Hong, M., Wang, D., Zhang, R., & Qian, L. (2014). Evolving an information diffusion model using a genetic algorithm for monthly river discharge time series interpolation and forecasting. *Journal of Hydrometeorology*, 15(6), 2236–2249.
- Bai, C. Z., Zhang, R., Hong, M., Qian, L., & Wang, Z. (2015). A new information diffusion modeling technique based on vibrating string equation and its application in natural disaster risk assessment. *International Journal of General Systems*, 44(5), 601–614.
- Bai, C., Zhang, R., Qian, L., & Wu, Y. (2017). A fuzzy graph evolved by a new adaptive Bayesian framework and its applications in natural hazards. *Natural Hazards Journal of the International Society for the Prevention & Mitigation of Natural Hazards*, 87, 899–918.
- Bai, C., Zhang, R., Bao, S., Liang, X. S., & Guo, W. (2018). Forecasting the tropical cyclone genesis over the northwest pacific through identifying the causal factors in the cyclone-climate interactions. *Journal of Atmospheric & Oceanic Technology*, 35(2), 247–259.
- Bicer, M.S., Diri, B. (2015). Predicting defect prone modules in web applications. 21st international conference on information and software technologies (ICIST).
- Bicer, M. S., & Diri, B. (2016). Defect prediction for cascading style sheets. *Applied Soft Computing*, 49, 1078–1084.
- Bowes, D., Hall, T., Harman, M. et al. (2016). Mutation-aware fault prediction. International symposium on software testing and analysis, pp. 330–341.
- Chen, X., Zhao, Y., Wang, Q., & Yuan, Z. (2018). MULTI: Multi-objective effort-aware just-in-time software defect prediction. *Information and Software Technology*, 93, 1–13.
- Ghotra, B., McIntosh, S., & Hassan, A. E. (2015). Revisiting the impact of classification techniques on the performance of defect prediction models. In *Proc. 37th international conference on software engineering* (pp. 789–800).
- Hall, T., Zhang, M., Bowes, D., & Sun, Y. (2014). Some code smells have a significant but small effect on faults. *ACM Transactions on Software Engineering and Methodology*, 23(4), 1–39.
- Halstead, M. H. (1977). *Elements of software science*. NewYork: Elsevier.
- Huang, C. (1997). Principle of information diffusion. *Fuzzy Sets and Systems*, 91, 69–90.
- Hand, D. J., & Yu, K. (2001). Idiot's Bayes: Not so stupid after all? *International Statistical Review*, 69(3), 385–398.
- Herbold, S., Trautsch, A., & Grabowski, J. (2017). Global vs. local models for cross-project defect prediction a replication study. *Empirical software engineering*, 22(4), 1866–1902.
- He, P., Li, B., Liu, X., Chen, J., & Ma, Y. (2015). An empirical study on software defect prediction with a simplified metric set. *Information and Software Technology*, 59, 170–190.
- Hosseini, S., Turhan, B., & Mäntylä, M. (2018). A benchmark study on the effectiveness of search-based data selection and feature selection for cross project defect prediction. *Information and Software Technology*, 95, 296–312.
- Huang, C. (2002). An application of calculated fuzzy risk. *Information Sciences*, 142(1–4), 37–56.
- Huang, C., Shi, Y. (2012). Towards efficient fuzzy information processing: Using the principle of information diffusion. Vol. 99:Physica.
- Jaganathan, G., Pillaipakkammatt, K., & Wright, R. N. (2009). A practical differentially private random decision tree classifier. In *IEEE international conference on data mining workshops* (pp. 114–121).
- Jin, C., & Liu, J. A. (2010). Applications of support vector machine and unsupervised learning for predicting maintainability using object-oriented metrics. In *Second international conference on multimedia and information technology* (pp. 24–27).
- Kamei, Y., et al. (2013). A large-scale empirical study of just-in-time quality assurance. *IEEE Transactions on Software Engineering*, 39(6), 757–773.
- Kaufman, A., Augustson, E. M., & Patrick, H. (2011). Unraveling the relationship between smoking and weight: The role of sedentary behavior. *Journal of Obesity*, 2012, 1–12.
- Kim, S., & Zhang, Y. (2008). Classifying software changes: Clean or buggy. *IEEE Transactions on Software Engineering*, 34(2), 181–196.
- Kira, K., Rendell, L. A. (1992). A practical approach to feature selection. *Proc. 9th international workshop on machine learning*, pp. 249–256.
- Khoshgoftaar, T. M., Seliya, N. (2002). Tree-based software quality estimation models for fault prediction. *Proc. 8th IEEE symposium software metrics*, pp. 203–214.

- Kononenko, I. (1994) Estimating attributes: Analysis and extensions of relief. Proc. European conference on machine learning on Machine Learning, pp.171–183.
- Lee, T., Nam, J., Han, D., Kim, S., & In, H. P. (2016). Developer micro interaction metrics for software defect prediction. *IEEE Transactions on Software Engineering*, 42(11), 1015–1035.
- Li, H. (2012). *Statistical learning method*. Tsinghua University press.
- Liang, X. S. (2014). Unraveling the cause-effect relation between time series. *Physical Review E Statistical Nonlinear & Soft Matter Physics*, 90(5–1), 052150.
- Lenz, A. R., Pozo, A., & Vergilio, S. R. (2013). Linking software testing results with a machine learning approach. Pergamon press. *Inc*, 26(5–6), 1631–1640.
- Ma, W., Chen, L., Yang, Y., Zhou, Y., & Xu, B. (2016a). Empirical analysis of network measures for effort-aware fault-proneness prediction. *Information & Software Technology*, 69(c), 50–70.
- Macias, D., Garcia-Gorri, E., & Stips, A. (2016). The seasonal cycle of the Atlantic jet dynamics in the alboran sea: Direct atmospheric forcing versus Mediterranean thermohaline circulation. *Ocean Dynamics*, 66(2), 1–15.
- McCabe, T. J. (1976). A complexity measure. *IEEE Transactions on Software Engineering*, 2(4), 308–320.
- Menzies, T., Greenwald, J., & Frank, A. (2007). Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering*, 33(1), 2–13.
- Malhotra, R. (2015). A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing Journal*, 27(c), 504–518.
- Ma, Y., Liang, S., Chen, X., & Jia, C. (2016b). The approach to detect abnormal access behavior based on naive Bayes algorithm. In *International conference on innovative Mobile and internet Services in Ubiquitous Computing*, IEEE (pp. 313–315).
- Miholca, D., Czibula, G., & Czibula, I. G. (2018). A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks. *Information Sciences*, 441, 152–170.
- Plackett, R. L. (1983). Karl Pearson and the chi-squared test. *International Statistical Review*, 51(1), 59–72.
- Pelayo, L., Dick, S. (2007). Applying novel resampling strategies to software defect prediction. NAFIPS 2007–2007 annual meeting of the north American fuzzy information processing society, pp. 69–72.
- Quinlan, J. R. (1993). C4.5: Programs for machine learning.
- Olague, H. M., Gholston, S., Quattlebaum, S. (2007). 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, 402–419.
- Robnikšikonja, M., & Kononenko, I. (2003). Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning*, 53(1/2), 23–69.
- Rathore, S. S., & Kumar, S. (2017). Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems. *Knowledge-Based Systems*, 119, 232–256.
- Razali, N. M., & Wah, Y. B. (2011). Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics*, 2(1), 21–33.
- Song, Q., Jia, Z., Shepperd, M., Ying, S., Liu, J. (2011). A general software defect-proneness prediction framework. *IEEE Transactions on Software Engineering*, vol.37, no.3, pp.356–370.
- Shirakawa, M., Nakayama, K., Hara, T., & Nishio, S. (2015). Wikipedia-based semantic similarity measurements for Noisy short texts using extended naive Bayes. *IEEE Transactions on Emerging Topics in Computing*, 3(2), 205–219.
- Tang, B., He, H., Baggenstoss, P., & Kay, S. (2016). A Bayesian classification approach using class-specific features for text categorization. *IEEE Transactions on Knowledge & Data Engineering*, 28(6), 1602–1606.
- Tantithamthavom, C., McIntosh, S., Hassan, A., & Matsumoto, K. (2017). An empirical comparison of model validation techniques for defect prediction models. *IEEE Transactions on Software Engineering*, 43(1), 1–18.
- Tong, H., Liu, B., & Wang, S. (2018). Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning. *Information and Software Technology*, 96, 94–111.
- Turhan, B., & Bener, A. (2007). Software defect prediction: Heuristics for weighted Naïve Bayes. In *Proceedings of the second international conference on software and data technologies* (pp. 244–249).
- Turhan, B., Menzies, T., Bener, A. B., & Di Stefano, J. (2009). On the relative value of cross-company and within-company data for defect prediction. *Empirical Software Engineering*, 14(5), 540–578.
- Turhan, B., & Bener, A. (2009). Analysis of naive bayes' assumptions on software fault data: An empirical study. *Data & Knowledge Engineering*, 68(2), 278–290.
- Vitello, G., Sorbello, M., & F., G. I. M., Conti, V., Vitabile, S. (2014). A novel technique for fingerprint classification based on fuzzy C-means and naive Bayes classifier. In *Eighth international conference on complex* (pp. 155–161).
- Witten, L. H., Frank, E., & Hell, M. A. (2011). Data mining: Practical machine learning tools and techniques (third edition). In *Acm Sigsoft software engineering notes*, 90–99. Burlington: Morgan Kaufmann.
- Wong, T. T. (2012). A hybrid discretization method for naive Bayesian classifiers. *Pattern Recognition*, 45(6), 2321–2325.

- Wu, Y., Huang, S., Ji, H., Zheng, C., & Bai, C. (2018). A novel Bayes defect predictor based on information diffusion function. *Knowledge-Based Systems*, 144, 1–8.
- Xia, X., Lo, D., Pan, S. J., Nagappan, N., & Wang, X. (2016). HYDRA: Massively compositional model for cross-project defect prediction. *IEEE Transactions on Software Engineering*, 42(10), 977–998.
- Yang, X., Lo, D., Xia, X., & Sun, J. (2017). TLEL: A two-layer ensemble learning approach for just-in-time defect prediction. *Information and Software Technology*, 87, 206–220.
- Yang, X., Tang, K., & Yao, X. (2015). A learning-to-rank approach to software defect prediction. *IEEE Transactions on Reliability*, 64(1), 234–246.
- Yang, T., Qian, K., & Dan, C. T. L. (2016). Improve the prediction accuracy of Naïve Bayes classifier with association rule mining. In *International conference on big data security on cloud*, IEEE (pp. 129–133).
- Yu, Q., Jiang, S., & Zhang, Y. (2017). A feature matching and transfer approach for cross-company defect prediction. *Journal of Systems and Software*, 132, 366–378.
- Yu, L., & Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Twentieth international conference on international conference on machine learning* (pp. 856–863).
- Zaidi, N. A., Cerquides, J., Carman, M. J., & Webb, G. I. (2013). Alleviating naive Bayes attribute independence assumption by attribute weighting. *Journal of Machine Learning Research*, 14(1), 1947–1988.
- Zhang, H., & Sheng, S. (2005). Learning weighted naive Bayes with accurate ranking. In *IEEE international conference on data mining* (pp. 567–570).
- Zhao, Y., Yang, Y., Lu, H., Zhou, Y., Song, Q., & Xu, B. (2015). An empirical analysis of package-modularization metrics: Implications for software fault-proneness. *Information & Software Technology*, 57(1), 186–203.
- Zhao, Y., Yang, Y., Lu, H., Liu, J., Leung, H., Wu, Y., Zhou, Y., & Xu, B. (2017). Understanding the value of considering client usage context in package cohesion for fault-proneness prediction. *Automated Software Engineering*, 24(2), 393–453.
- Zheng, F., Webb, G. I. (2005). A comparative study of semi-naïve Bayes methods in classification learning. *Proc. 4th Australasian data mining conference*, pp. 141–156.
- Zheng, J. (2010). Cost-sensitive boosting neural networks for software defect prediction. *Expert Systems with Applications*, 37(6), 4537–4543.
- Zhou, L., Li, R., Zhang, S., & Wang, H. (2017). Imbalanced data processing model for software defect prediction. *Wireless Pers Commun*, 6, 1–14.



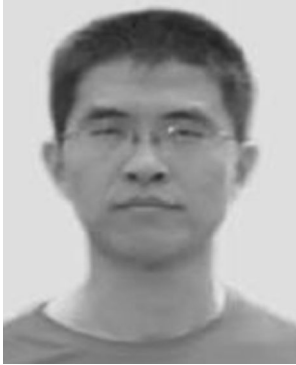
Haijin Ji received his B.A.S.C. in Information Science from Zhongyuan University of Technology, Zhengzhou, China, and M.A.S.C. in Information Science from Jiangnan University, Wuxi, China. He is a member of software engineering research group of Huaiyin Normal University and a Doctor candidate in Army Engineering University of PLA. His current research interests include software testing, defects prediction and fuzzy information fusion. Contact him at 94662837@qq.com.



Song Huang received Ph. D degree from PLA university of Science and Technology. He is a member of CCF and ACM. He is currently a professor of software engineering at Software Testing and Evaluation Center from Army Engineering University of PLA. He is a member of the advisory boards of Journal of Systems and Software, IEEE Transactions on Reliability, etc. He has contributed more than 100 journal articles to professional journals. His current research interests include software testing, quality assurance, data mining and empirical software engineering. Contact him at hs0317@163.com.



Yaning Wu received her B.A.SC. in Information Science from Army Engineering University of PLA, Nanjing, China, M.A.SC. in Information Science from the Institute of Information System, Nanjing, China. Her current research interests include software testing and defects prediction. Contact her at muyaxin@126.com.



Zhanwei Hui received his B.A.SC. in Information Science from Army Engineering University of PLA, Nanjing, China, M.A.SC. in Information Science from the Institute of Information System, Nanjing, China. His current research interests include software testing and defects prediction. Contact him at hzw_1983821@163.com.



Changyou Zheng received his B.A.SC. in Information Science from Army Engineering University of PLA, China, M.A.SC. in Information Science from the Institute of Information System, Nanjing, China. His current research interests include software testing, defects prediction, linguistic preference modeling and fuzzy information fusion. Contact him at zheng_chy@163.com.