Empir Software Eng
https://doi.org/10.1007/s10664-018-9638-1

Balancing the trade-off between accuracy and interpretability in software defect prediction

Toshiki Mori [1,2] · **Naoshi Uchihira** [2]

Abstract

Context Classification techniques of supervised machine learning have been successfully applied to various domains of practice. When building a predictive model, there are two important criteria: predictive accuracy and interpretability, which generally have a trade-off relationship. In particular, interpretability should be accorded greater emphasis in the domains where the incorporation of expert knowledge into a predictive model is required.

Objective The aim of this research is to propose a new classification model, called superposed naive Bayes (SNB), which transforms a naive Bayes ensemble into a simple naive Bayes model by linear approximation.

Method In order to evaluate the predictive accuracy and interpretability of the proposed method, we conducted a comparative study using well-known classification techniques such as rule-based learners, decision trees, regression models, support vector machines, neural networks, Bayesian learners, and ensemble learners, over 13 real-world public datasets.

Results A trade-off analysis between the accuracy and interpretability of different classification techniques was performed with a scatter plot comparing relative ranks of accuracy with those of interpretability. The experiment results show that the proposed method (SNB) can produce a balanced output that satisfies both accuracy and interpretability criteria.

Conclusions SNB offers a comprehensible predictive model based on a simple and transparent model structure, which can provide an effective way for balancing the trade-off between accuracy and interpretability.

Communicated by: Tim Menzies

✉ Toshiki Mori
toshiki1.mori@toshiba.co.jp; toshiki.mori@gmail.com

Naoshi Uchihira
uchihira@jaist.ac.jp

[1] Corporate Software Engineering & Technology Center, Toshiba Corporation, Kawasaki, Kanagawa, Japan

[2] School of Knowledge Science, Japan Advanced Institute of Science and Technology (JAIST), Nomi, Ishikawa, Japan

Published online: 25 July 2018

Keywords Software defect prediction · Predictive accuracy · Interpretability · Trade-off analysis · Naive Bayes classifier · Weights of evidence · Ensemble learning · Model approximation

1 Introduction

In recent years, classification techniques of supervised machine learning have been successfully applied to various domains, i.e., computer vision, speech recognition, information retrieval, marketing, finance, manufacturing, bioinformatics, and medical diagnosis (Bishop 2006; Hastie et al. 2009). Consequently, a variety of predictive models have been developed so far in order to support or even replace human decision making in these domains. When building a predictive model, there are two important criteria: predictive accuracy and interpretability (Fayyad et al. 1996; Freitas 2004, 2014). Although accuracy is an essential property of a predictive model, interpretability should be also taken into account particularly in the domains where the incorporation of expert knowledge into a predictive model is required. Predictive accuracy and interpretability in general have a trade-off relationship, and their proper balance may be different for each domain (Freitas 2004, 2014).

Among these domains, software engineering has been one of the promising application areas for machine learning. In order to deal with the increasing complexity of software development, machine learning techniques have been extensively applied to software defect prediction (Hall et al. 2012; Malhotra 2015), software development effort estimation (Dejaeger et al. 2012; Wen et al. 2012), and software project risk analysis (Choetkiertikul et al. 2015; Hu et al. 2013; Mori et al. 2013). While most studies in this area have been mainly concerned with the accuracy of various predictive models, rather less attention has been paid to their interpretability.

In this paper, we propose a new classification model, called superposed naive Bayes (SNB; Mori 2015), which uses a two-step approach, i.e., firstly builds a naive Bayes ensemble via stochastic boosting, and then transforms it into a simple naive Bayes model by linear approximation. The two-step approach decomposes the problem of balancing the trade-off between accuracy and interpretability into two sub-problems: improving the predictive accuracy of a naive Bayes ensemble, followed by obtaining a better linear approximation to gain the interpretability. In order to evaluate the accuracy and interpretability of the proposed method, we conduct a comparative study using well-known classification techniques such as rule-based learners, decision trees, regression models, support vector machines, neural networks, Bayesian learners, and ensemble learners, over 13 real-world public datasets.

In this study, we set the following research questions:

(RQ1) How accurate are the current defect predictors?

In order to compare the accuracy of the proposed method (SNB) with those of the current defect predictors, we adopt a similar approach to Ghotra et al. (2015) that uses multiple repetition of k-hold cross-validation and a double Scott-Knott test (Jelihovschi et al. 2014). Our findings indicate that SNB is able to achieve a relatively higher rank of accuracy among the benchmark predictors.

(RQ2) How can we assess the interpretability of the predictors?

To our knowledge, there are few studies comparing the interpretability of different classification techniques. We perform a qualitative assessment of the interpretability based on Lipton (2016)'s criteria, which is comprised of model transparency (simulatability), component transparency (decomposability), and algorithmic transparency.

(RQ3) How can we balance the trade-off between accuracy and interpretability?

In order to analyze the trade-off between accuracy and interpretability of the predictors, we combine the experimental results measuring accuracy with the assessment results of interpretability. Our findings confirm a generally accepted assumption that the higher the predictor's accuracy the lower its interpretability. Nevertheless, the proposed method (SNB) is able to produce balanced outputs that satisfied both accuracy and interpretability criteria.

The remainder of this paper is organized as follows: Section 2 briefs background and related work. Section 3 describes the proposed method. Section 4 deals with experimental settings. Section 5 discusses the experimental results and answers the research questions. Section 6 presents threats to validity. Finally, concluding remarks are shown in Section 7.

2 Background and Related Work

2.1 Software Defect Prediction

Software defect prediction is the process to predict where defects might appear in the future and to quantify the relative importance of various factors used to build a model (Kamei and Shihab 2016). Existing studies in software defect prediction are generally characterized by what combination of software metrics and machine learning techniques are used in the predictive models.

Guo et al. (2004) evaluated the predictive performance of a software defect predictor using random forest (Breiman 2001) by five datasets from the NASA Metrics Data Program (MDP). Menzies et al. (2007) explored the possibility of data mining static code metrics for software defect prediction by demonstrating the effectiveness of using a naive Bayes classifier with logarithmic filtering. Lessmann et al. (2008) conducted a large-scale empirical comparison of 22 classification techniques over 10 MDP datasets and showed that the selection of a classification technique had less impact on predictive accuracy. Ghotra et al. (2015) revisited the findings of Lessmann et al. (2008) with non-biased MDP datasets (Shepperd et al. 2013), thus confirming statistical distinction between classification techniques. Zimmermann et al. (2007) investigated the relationship between complexity metrics and post-release defects in Eclipse files and packages with logistic regression analysis. Zimmermann et al. (2009) conducted a large-scale experiment of cross-project defect prediction over 12 real-world applications. Kim et al. (2008) proposed change level defect prediction, i.e., Just-in-Time (JIT) defect prediction, which determines whether a new software change is more similar to prior

buggy changes or clean changes. Kamei et al. (2013) performed an empirical study of JIT defect prediction on a variety of open source and commercial projects from multiple domains.

Whereas the above studies mainly focus on only predictive accuracy, there have been also several studies concerning interpretability as well. Fenton et al. (2008) proposed a causal model using Bayesian networks (BN) for early life cycle defect prediction. Dejaeger et al. (2013) compared the predictive accuracy and comprehensibility of 15 different BN classifiers. Vandecruys et al. (2008) used a comprehensible classification technique based on ant colony optimization (ACO) for predicting fault-prone software modules. To the best of our knowledge, few studies have performed the trade-off analysis between accuracy and interpretability of different classification techniques.

2.2 Interpretability

A predictive model built for software engineering applications should be both accurate and interpretable to users. Software engineering data collected from actual development process is often noisy and non-stationary, which only captures limited aspects of the software development activities (Briand et al. 1992; Fenton and Neil 1999). In order to exploit a predictive model for practical use, we need to carefully interpret the prediction results and understand the mechanism behind the data. The importance of interpretability may further increase in a situation where the output is used for crucial decision-making.

Despite the increasing attention to interpretability, there seems to be no clear consensus on its definition. In fact, interpretability has been defined in various ways in literature, sometimes referred to by different terms such as comprehensibility, understandability, or explainability. Martens et al. (2011) defined interpretability (comprehensibility) as the mental fit of a predictive model, whereas the data fit corresponds to predictive accuracy. Kulesza et al. (2015) described interpretability (explainability) as the ability to accurately explain the learning system's reasons for each prediction to the end user. Ribeiro et al. (2016) defined interpretability as the ability to provide qualitative understanding between the input variables and the response. Lipton (2016) considered that interpretability is not a monolithic concept but reflects several distinct ideas, where the properties of interpretable models are classified into two categories: transparency (i.e., *how does the model work?*) and post-hoc interpretability (i.e., *what else can the model tell me?*). Transparency is the opposite concept of opacity or blackbox-ness that connotes some sense of understanding the mechanism by which the model works, which can be further decomposed into three sub-categories: model transparency (simulatability), component transparency (decomposability), and algorithmic transparency. Conversely, post-hoc interpretability focuses on extracting useful information from a learned model after-the-fact without elucidating precisely how the model works. One example of such post-hoc interpretability is partial dependence plots (PDP; Friedman 2001), which highlight the average partial relationship between a set of predictors and the predicted response (Goldstein et al. 2015). Another example is Local Interpretable Model-agnostic Explanations (LIME; Ribeiro et al. 2016) that provides model-agnostic explanations by locally approximating the model around a given prediction.

The evaluation of interpretability is, in general, a difficult task. If all the classification models under consideration are of the same type, the model size, such as the number of terms, rules, or tree nodes, could be used for an interpretability measure. However, comparing the interpretability of different types of classification models is more challenging, because the size-based interpretability measures are generally specific to each model type. There have been few studies of user-based experiments directly comparing the interpretability among different model types. Huysmans et al. (2011) designed an end-user experiment to test the accuracy, response time, and answer confidence for a set of problem-solving tasks and revealed that decision tables performed significantly better on all the criteria. Allahyari and Lavesson (2011) conducted a quantitative survey to examine the understandability of decision trees and rule lists induced from two small datasets and analyzed the results by using the analytic hierarchy process (AHP; Saaty 1990). Such experiments comparing the users' preferences for different model types are very important, but note that the results of such experiments largely depend on the characteristics of datasets and are naturally biased by the background of the users (Freitas 2014).

3 Proposed Method

In this paper, we propose a new classification model that uses a two-step approach, where simple naive Bayes models (Section 3.1) or tree augmented naive Bayes models (Section 3.2) are firstly combined into an ensemble model (Section 3.3) and then transformed back into a simple naive Bayes model by linear approximation (Section 3.4). We focus on binary classification problems, because multiclass classification problems can be converted into binary problems by either one-versus-all (OVA), all-versus-all (AVA), error-correcting output-coding (ECOC), or generalized coding (Aly 2005).

3.1 Simple Naive Bayes

The naive Bayes classifier (Domingos and Pazzani 1997; Lewis 1998) is a simple probabilistic classifier based on Bayes' theorem along with a strong independence assumption. Let Y denote a binary dependent variable that equals either 1 (true) or 0 (false), and let X_1 through X_d denote independent variables, either categorical or numeric. Based on Bayes' theorem and the independence assumption, a posterior probability $P(Y=1|X_1, \cdots, X_d)$ is represented as the multiplication of a prior probability $P(Y=1)$ and likelihoods $P(X_i|Y=1)$ divided by a constant $P(X_1, \cdots, X_d)$, as the following form:

$$P(Y=1|X_1,\cdots,X_d) = \frac{P(Y=1)P(X_1,\cdots,X_d|Y=1)}{P(X_1,\cdots,X_d)} = \frac{P(Y=1)\prod_{i=1}^{d}P(X_i|Y=1)}{P(X_1,\cdots,X_d)} \quad (1)$$

A posterior probability $P(Y=0|X_1, \cdots, X_d)$ is also expressed as

$$P(Y=0|X_1,\cdots,X_d) = \frac{P(Y=0)P(X_1,\cdots,X_d|Y=0)}{P(X_1,\cdots,X_d)} = \frac{P(Y=0)\prod_{i=1}^{d}P(X_i|Y=0)}{P(X_1,\cdots,X_d)} \quad (2)$$

Note that missing values in the dataset are ignored when calculating the probability estimates. From Eqs. 1 and 2, we obtain the log-odds in favor of $Y=1$ as follows:

$$\underbrace{log\frac{P(Y=1|X_1,\cdots,X_d)}{P(Y=0|X_1,\cdots,X_d)}}_{W(X)}=\underbrace{log\frac{P(Y=1)}{P(Y=0)}}_{W_0}+\sum_{i=1}^{d}\underbrace{log\frac{P(X_i|Y=1)}{P(X_i|Y=0)}}_{W_i(X_i)} \tag{3}$$

Equation 3 is a linear additive model, each term of which is called the weight of evidence (WoE; Good 1985; Madigan et al. 1997). A positive $W_i(X_i)$ indicates that the state of X_i is evidence in favor of the hypothesis that $Y=1$, and a negative $W_i(X_i)$ is evidence for $Y=0$. The total sum of weights of evidence $W(X)$ determines a raw score, the higher of which indicates a higher posterior probability of $Y=1$.

From a practical point of view, several issues should be addressed. The first issue is the zero-frequency problem, which arises when a dependent variable value and an independent variable value never occur together in the training data, and results in an infinite or indefinite weight of evidence in Eq. 3. A principled solution to this problem is to incorporate a small-sample correction into all the likelihoods. This technique is called Laplace correction (Cestnik 1990). If we consider a contingency table shown in Table 1, for example, then Laplace correction replaces the likelihood $P(X_i=x|\ Y=y)=n_{(i,y,x)}/n_{(i,*,x)}$ with $(n_{(i,y,x)}+l)/(n_{(i,*,x)}+2l)$, where l is a positive value, such as 1.

The second issue is the handling of numeric variables. There are three approaches for dealing with numeric variables in naive Bayes classifiers: parametric approaches, non-parametric approaches, and discretization (Bouckaert 2004). Parametric approaches approximate the distribution of the numeric variables using a parameterized distribution such as the Gaussian. In contrast, non-parametric approaches use a sum of kernel functions that are centered around data points. John and Langley (1995) proposed to use a normal distribution for the kernel where the mean is the data point and the variance is estimated as the inverse of the root of the number of instances. Unlike these continuous approaches, discretization is performed as a pre-processing step that converts the continuous variables into discrete variables prior to the learning process.

Commonly used discretization techniques include equal width discretization (EWD; Yang and Webb 2009), equal frequency discretization (EFD; Yang and Webb 2009), and the minimum description length criterion (MDL; Bouckaert 2004; Fayyad and Irani 1993; Yang and Webb 2009). In particular, the MDL criterion is one of the most popular discretization techniques that start with a single interval containing all data, and recursively splits intervals as long as the information gain exceeds a given threshold. In this paper, we propose to alternatively use Akaike's information criterion (AIC; Akaike 1973; Sakamoto and Akaike 1978) or Bayesian information criterion (BIC; Schwarz 1978) for discretization of continuous variables. A detailed description is provided in Appendix 1.

Table 1 A contingency table for a binary dependent variable Y and an independent variable X_i with k values

	$X_i=v_1$	$\cdots$	$X_i=v_k$	Total
$Y=1$	$n_{(i,1,1)}$	$\cdots$	$n_{(i,1,k)}$	$n_{(i,1,*)}$
$Y=0$	$n_{(i,0,1)}$	$\cdots$	$n_{(i,0,k)}$	$n_{(i,0,*)}$
Total	$n_{(i,*,1)}$	$\cdots$	$n_{(i,*,k)}$	$n_{(i,*,*)}$

The discretization method based on AIC or BIC enables us to concurrently calculate the importance of the independent variables that can be used for variable selection.

The third issue is the post-processing of raw scores. Platt (1999) proposed to use the sigmoid functions for the calibration of raw scores to posterior probabilities in support vector machines (SVMs). Zadrozny and Elkan (2002) applied the sigmoid calibration to naive Bayes classifiers, resulting in the significant improvement of the mean squared error (MSE). Accordingly, class membership probability estimates for naive Bayes classifiers, denoted as $P_{nb}(X)$, is obtained with the sigmoid calibration of raw scores $W(X)$, which is given by

$$P_{nb}(X) = sigmoid(c_0 + c_1 W(X)) = \frac{1}{1 + exp(-(c_0 + c_1 W(X)))} \tag{4}$$

where the parameters c_0 and c_1 are searched to maximize the log-likelihood of the data. Furthermore, given a discrimination threshold s for the probability estimates, binary class prediction $C_{nb}(X|\, s)$ is made as follows:

$$C_{nb}(X|s) = \begin{cases} 1\ (true), & if\ \ P_{nb}(X) \geq\ s \\ 0\ (false), & if\ \ P_{nb}(X) < s \end{cases} \tag{5}$$

The model is fitted to the data by finding the discrimination threshold s such that the weighted sum error for misclassified points is minimized.

A naive Bayes classifier is stable, robust, and computationally efficient. Although the conditional independence assumption is often unrealistic, naive Bayes classifiers have been found to perform surprisingly well in many real-world applications (Domingos and Pazzani 1997; Menzies et al. 2007; Rish 2001; Zhang 2004). Rish (2001) used Monte Carlo simulations to confirm that the accuracy of naive Bayes is not directly correlated with the degree of variable dependencies. Zhang (2004) proved that no matter how strong the dependences among variables are, naive Bayes can still be optimal if the dependences distribute evenly in classes, or if the dependences cancel each other out.

A naive Bayes classifier is also easy to interpret, because its probabilistic explanations appear to replicate users' way of diagnosing (Kononenko 1993; Kotsiantis et al. 2007). In particular, the weight-of-evidence approach has been preferably applied in the domains where interpretability is highly valued, such as medical diagnosis (Goodman 1999; Heckerman et al. 1991; Madigan et al. 1997; Spiegelhalter and Knill-Jones 1984) and geological survey (Agterberg et al. 1990; Bonham-Carter et al. 1988; Carranza 2004; Dahal et al. 2008).

3.2 Tree Augmented Naive Bayes (TAN)

A simple naive Bayes classifier may not be optimal unless the dependences distribute evenly in classes, and unless the dependences cancel each other out. Thus, tree augmented naive Bayes (TAN; Friedman et al. 1997) was proposed as an extension of simple naive Bayes that relaxes the conditional independence assumption among variables by employing a tree structure, in which each independent variable only depends on a dependent variable and one other independent variable. Figure 1 shows examples of various Bayesian network structures. Simple naive Bayes (Fig. 1a) has a star-like

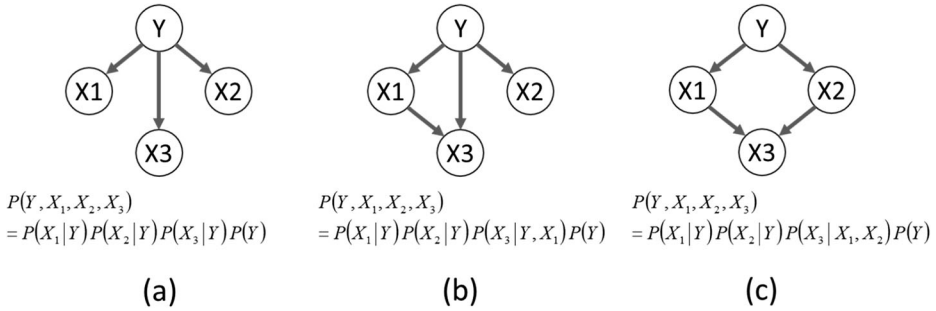


Fig. 1 Examples of Bayesian network structures. **a** Simple naive Bayes. **b** Tree augmented naive Bayes (TAN). **c** General Bayesian network

structure, where each independent variable (X_1, X_2, X_3) only depends on the dependent variable c. In contrast, TAN (Fig. 1b) allows for a tree-structured dependence among independent variables in addition to the dependence on Y. On the other hand, the model described in Fig. 1c is a general Bayesian network, because an independent variable X_3 depends on *multiple* independent variables X_1 and X_2. The structure of TAN can be determined by building a maximum weighted spanning tree using the pairwise mutual information as the weights of the arcs in the tree. Friedman et al. (1997) reported that TAN outperformed simple naive Bayes in predictive accuracy, yet at the same time maintained the computational simplicity and robustness.

If we use TAN instead of simple naive Bayes as a base learner of the proposed method, a further preprocessing step is required. A TAN model can be easily transformed into a simple naive Bayes model by utilizing the peculiar treatment of missing values when calculating the probability estimates in naive Bayes, i.e., missing values are ignored in the calculation of weights of evidence. For example, Fig. 2a represents a TAN model in which X_3 depends on Y and X_1. The preprocessing step decomposes X_3 into two separate variables X_3' and X_3'' as defined in Eq. 6, and generates a simple naive Bayes model as shown in Fig. 2b.

$$X_3' = \begin{cases} X_3, & if\ X_1 = x_{11} \\ NA, & if\ X_1 \neq x_{11} \end{cases}, X_3'' = \begin{cases} X_3, & if\ X_1 = x_{12} \\ NA, & if\ X_1 \neq x_{12} \end{cases}, \tag{6}$$

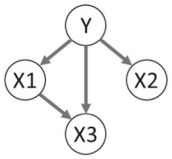
This transformation would be particularly useful in the superposition procedure described later in Section 3.4. Note that a TAN model of the original structure can be easily reconstructed from the transformed naive Bayes model, if necessary, even after the superposition of weights of evidence.

3.3 Naive Bayes Ensemble (NBE)

The proposed method uses a two-step approach of model construction, where the first step is to build an ensemble model of naive Bayes classifiers. We assume that the first step mainly contributes to the accuracy part of the predictor. An ensemble model is defined as a collection of learning models for obtaining better predictive performance as compared to individual model separately. Among various types of ensemble models, the most popular techniques are bootstrap aggregating (bagging; Breiman 1996) and boosting (Freund and Schapire 1997).

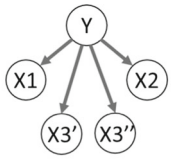
X_1	X_2	X_3	Y
x_{11}	x_{21}	x_{31}	1
x_{11}	x_{22}	x_{31}	1
x_{11}	x_{21}	x_{31}	0
x_{11}	x_{22}	x_{32}	0
x_{12}	x_{21}	x_{32}	1
x_{12}	x_{22}	x_{32}	1
x_{12}	x_{21}	x_{32}	0
x_{12}	x_{22}	x_{31}	0

X_1	X_2	X_3'	X_3''	Y
x_{11}	x_{21}	x_{31}	NA	1
x_{11}	x_{22}	x_{31}	NA	1
x_{11}	x_{21}	x_{31}	NA	0
x_{11}	x_{22}	x_{32}	NA	0
x_{12}	x_{21}	NA	x_{32}	1
x_{12}	x_{22}	NA	x_{32}	1
x_{12}	x_{21}	NA	x_{32}	0
x_{12}	x_{22}	NA	x_{31}	0



$$P(Y, X_1, X_2, X_3) = P(X_1|Y)P(X_2|Y)P(X_3|Y, X_1)P(Y)$$

(a)



$$P(Y, X_1, X_2, X_3) = P(X_1|Y)P(X_2|Y)P\left(X_3'|Y\right)P\left(X_3''|Y\right)P(Y)$$

(b)

Fig. 2 An example of the transformation of TAN into simple naive Bayes. **a** A TAN model. **b** The corresponding simple naive Bayes model, where NAs indicate missing values

Bagging (Breiman 1996) is a technique for building an ensemble model of weak learners (typically decision trees) that fit to different subsets of training data generated by bootstrap sampling, i.e., sampling with replacement. The combined results are obtained by voting or averaging the individual results. Random forest (Breiman 2001) is an important extension of bagging that selects a subset of independent variables at random out of the total and uses the best split variable from the subset to split each node in a tree, unlike in bagging where all the variables are considered for splitting a node. Bagging and random forest are a kind of variance reduction techniques suitable for *unstable* learners with low-bias and high-variance such as decision trees to help avoid overfitting and improve predictive accuracy. However, naive Bayes classifiers are *stable* learners with high-bias and low-variance, and are thus unable to benefit from the bagging-based ensemble (Bauer and Kohavi 1999).

Boosting is another popular technique that incrementally builds an ensemble model by reweighting each sample of training data based on previous classification results: samples that are misclassified gain weight and samples that are classified correctly lose weight. AdaBoost is the first practical boosting algorithm proposed by Freund and Schapire (1997), which is based on a multiplicative weight-update technique that does not require any prior knowledge of the weak learner. AdaBoost is known to often dramatically improve performance but sometimes overfit. Hence, Friedman (2002) proposed an extended boosting algorithm called stochastic gradient boosting that uses a gradient descent approach at each iteration of which a component classifier should be fit on a subsample of the training set drawn at random without replacement. While bagging is mainly a variance reduction technique, boosting can reduce both bias and variance (Webb 2000). This implies that boosting may complement the weakness of naive Bayes classifiers, which are *stable* learners with high-bias and low-variance.

1. Initialize sample weights $w_i = 1/n$, $i = 1, 2, \cdots, n$.
2. Build a naive Bayes classifier $nb_1(X)$ from $X = \{x_1, x_2, \cdots, x_n\}$.
3. Set $\beta_1 \leftarrow 1$.
4. For $t = 2$ to m:
   a. Draw a random subsample X_a of size ηn $(0 < \eta < 1)$ from X without replacement.
   b. Build $nb_t(X_a)$ with a threshold s_t so as to minimize $e_t = \frac{\sum_{x_i \in X_a} w_i I\left(y_i \neq C_{nb}^{(t)}(x_i|s_t)\right)}{\sum_{x_i \in X_a} w_i}$.
   c. Compute $\alpha_t = \nu \log((1 - e_t)/e_t)$, where ν $(0 < \nu < 1)$ is a shrinkage parameter.
   d. Compute $\beta_t = \nu \log((1 - e_{oob})/e_{oob})$, where $e_{oob} = \frac{\sum_{x_i \in X - X_a} w_i I\left(y_i \neq C_{nb}^{(t)}(x_i|s_t)\right)}{\sum_{x_i \in X - X_a} w_i}$.
   e. Set $w_i \leftarrow w_i \, exp\left(\alpha_t \, I\left(y_i \neq C_{nb}^{(t)}(x_i|s_t)\right)\right)$, $i = 1, 2, \cdots, n$.
5. Output $P_{nbe}(x) = \frac{\sum_{t=1}^{m} \beta_t P_{nb}^{(t)}(x)}{\sum_{t=1}^{m} \beta_t}$.

Fig. 3 Naive Bayes ensemble (NBE) algorithm that extends AdaBoost with a stochastic gradient descent method

Therefore, in order to improve the predictive accuracy of simple naive Bayes, we propose a new naive Bayes ensemble (NBE) algorithm that extends AdaBoost with a stochastic gradient descent method. Figure 3 shows the proposed algorithm, where there are several differences from the original AdaBoost: stochastic subsampling in step 4a; gradient descent in step 4c; out-of-bag estimates (Breiman 2001) in step 4d.

3.4 Superposed Naive Bayes (SNB)

The second step of the proposed method is to transform a naive Bayes ensemble model into a simple naive Bayes model by linear approximation. We assume that the second step mainly contributes to the interpretability part of the predictor. The transformation is based on the superposition of weights of evidence. Figure 4 illustrates an image of the

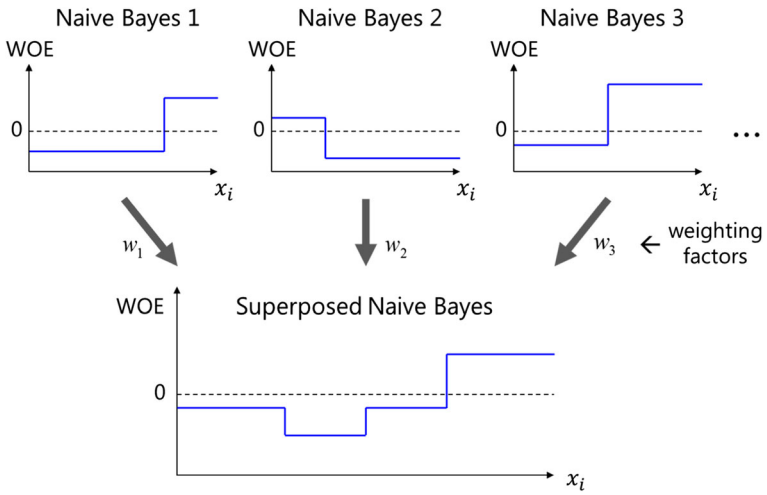


Fig. 4 An image of the superposition of weights of evidence (WoE)

superposition procedure, where all the weights of evidence of naive Bayes classifiers are accumulated with weighting factors w_k ($k = 1, 2, 3, \ldots$) across the range of the variable X_i.

Referring to Eqs. 3, 4, and Fig. 4, the output of NBE is represented as

$$P_{nbe}(X) = \frac{\sum_{t=1}^{m} \beta_t P_{nb}^{(t)}(X)}{\sum_{t=1}^{m} \beta_t} = \frac{\sum_{t=1}^{m} \beta_t \, sigmoid\left(c_0^{(t)} + c_1^{(t)} W^{(t)}(X)\right)}{\sum_{t=1}^{m} \beta_t} = \frac{\sum_{t=1}^{m} \beta_t \, sigmoid\left(c_0^{(t)} + c_1^{(t)} \sum_{i=0}^{d} W_i^{(t)}(X)\right)}{\sum_{t=1}^{m} \beta_t} \tag{7}$$

Assume a linear approximation such that $sigmoid\left(c_0^{(t)} + c_1^{(t)} W^{(t)}(X)\right) \cong a_t + b_t W^{(t)}(X)$, Eq. 7 can be transformed into the following equation, which is the output of SNB.

$$P_{snb}(X) = \underbrace{\frac{\sum_{t=1}^{m} \beta_t \left(a_t + b_t W_0^{(t)}(X)\right)}{\sum_{t=1}^{m} \beta_t}}_{W_0'} + \sum_{i=1}^{d} \underbrace{\frac{\sum_{t=1}^{m} \beta_t b_t W_i^{(t)}(X)}{\sum_{t=1}^{m} \beta_t}}_{W_i'} \tag{8}$$

In approximating the sigmoid function, we use Taylor expansion. The first order approximation of $sigmoid\left(c_0^{(t)} + c_1^{(t)} W^{(t)}(X)\right)$ is derived from Taylor expansion around $c_0^{(t)}$ as follows:

$$sigmoid\left(c_0^{(t)} + c_1^{(t)} W^{(t)}(X)\right) = \sum_{n=0}^{\infty} \frac{sigmoid^{(n)}\left(c_0^{(t)}\right)}{n!} \left(c_1^{(t)} W^{(t)}(X)\right)$$

$$\cong \underbrace{\frac{1}{1 + exp\left(-c_0^{(t)}\right)}}_{a_t} + \underbrace{\frac{c_1^{(t)} exp\left(-c_0^{(t)}\right)}{\left(1 + exp\left(-c_0^{(t)}\right)\right)^2}}_{b_t} W^{(t)}(X) \tag{9}$$

It is noteworthy that SNB expressed in Eq. 8 is equivalent to a *simple* naive Bayes classifier with the superposed weights of evidence $W_i^{'}$ ($i = 1, 2, \cdots, d$), where the features of individual naive Bayes classifiers are integrated.

Our approach (SNB) was inspired by the work of Ridgeway et al. (1998), in which a derivation of boosted naive Bayes classifier using Taylor expansion was proposed. The main difference is that Ridgeway's model applies Taylor expansion to the combining function of boosted naive Bayes, whereas our approach employs Tayler approximation for the sigmoid calibration of component classifiers in the boosting process. Furthermore, Ridgeway's model uses the original AdaBoost algorithm, whereas we adopt an extended AdaBoost using a stochastic gradient descent method and out-of-bag estimates to obtain better predictive performance.

4 Experimental Settings

4.1 Datasets

In our experiments, we use 13 real-world public datasets from two different data sources. One is the cleaned version of the NASA MDP datasets, created by Shepperd et al. (2013) and provided by Menzies et al. (2016), where the problem data (e.g., cases with either conflicting feature values or implausible values) and the useless data (e.g., the features with constant values and either identical or inconsistent cases) are removed. The other is the JIT datasets provided by Kamei et al. (2013), which contain various change-level metrics extracted from well-known open source projects.

From the first data source, i.e., the MDP datasets, we select 11 datasets: MC2, KC3, MW1, CM1, PC1, PC2, PC3, PC4, PC5, MC1, and JM1. They were originally collected from various NASA systems such as spacecraft instruments, flight software for earth orbiting satellite, real-time predictive ground system, and storage management system for ground data. Table 2 shows an overview of the MDP datasets. Each dataset defines a binary classification problem of software modules with numeric independent variables of characteristic code attributes, such as LOC counts, Halstead metrics (Halstead 1977) that estimate reading complexity by counting operators and operands in a software module, cyclomatic complexity (McCabe 1976) derived from a module's flow graph, and other miscellaneous attributes of software modules. Detailed descriptions of these metrics are found in Jiang et al. (2008b).

From the second data source, i.e., the JIT datasets, we select 2 datasets: Bugzilla and Columba. Bugzilla is a web-based bug tracking system originally developed by the Mozilla project. Columba is an open source email client written in Java. Table 3 shows an overview of the JIT datasets. Each dataset contains a binary dependent variable and 14 numeric independent variables, the latter of which are all change-level metrics comprised of 4 Diffusion metrics (NS, ND, NF, and Entropy), 3 Size metrics (LAn, LDn, and LTn), a Purpose metric (FIX), 3 History metrics (NDEV, AGE, and NUCn), and 3 Experience metrics (EXP, REXP, and SEXP). Note that the values of LAn, LDn, LTn, and NUCn are all normalized so as not to induce high correlation among independent variables. The rationale of each change-level metric is listed in Table 3. Further detailed descriptions and discussions are found in Kamei et al. (2013).

4.2 Defect Predictors

Table 4 summarizes the defect predictors that are compared in this study, including the proposed methods. The first 14 are popular defect predictors of various categories, which are available in Weka machine learning toolkit (Witten et al. 2011). The details of our parameter settings in Weka are listed in Appendix 2.

- OneR (Holte 1993) is a simple rule-based learner that uses the minimum-error variable for prediction.
- JRip is the Weka implementation of RIPPER (Cohen 1995) that is a propositional rule learner based on incremental reduced error pruning (IREP).
- J48 is the Weka implementation of C4.5 (Quinlan 1993) that is a popular decision tree algorithm using information entropy.

Table 2 An overview of the NASA MDP datasets (cleaned version). A *defective* module is defined as the module that contains one or more errors

		NASA MDP datasets (cleaned version)										
		MC2	KC3	MW1	CM1	PC1	PC2	PC3	PC4	PC5	MC1	JM1
LOC counts	LOC_total	X	X	X	X	X	X	X	X	X	X	X
	LOC_blank	X	X	X	X	X		X	X	X	X	X
	LOC_code_and_comment	X	X	X	X	X	X	X	X	X	X	X
	LOC_comments	X	X	X	X	X	X	X	X	X	X	X
	LOC_executable	X	X	X	X	X	X	X	X	X	X	X
	Number_of lines	X	X	X	X	X	X	X	X	X	X	
Halstead attributes	content	X	X	X	X	X	X	X	X	X	X	X
	difficulty	X	X	X	X	X	X	X	X	X	X	X
	effort	X	X	X	X	X	X	X	X	X	X	X
	error_est	X	X	X	X	X	X	X	X	X	X	X
	length	X	X	X	X	X	X	X	X	X	X	X
	level	X	X	X	X	X	X	X	X	X	X	X
	prog_time	X	X	X	X	X	X	X	X	X	X	X
	volume	X	X	X	X	X	X	X	X	X	X	X
	num_operands	X	X	X	X	X	X	X	X	X	X	X
	num_operators	X	X	X	X	X	X	X	X	X	X	X
	num_unique_operands	X	X	X	X	X	X	X	X	X	X	X
	mum_unique_operators	X	X	X	X	X	X	X	X	X	X	X
McCabe attributes	cyclomatic_complexity	X	X	X	X	X	X	X	X	X	X	X
	cyclomatic_density	X	X	X	X	X	X	X	X	X	X	
	design_complexity	X	X	X	X	X	X	X	X	X	X	X
	essential_complexity	X	X	X	X	X	X	X	X	X	X	X
Miscellaneous	branch_count	X	X	X	X	X	X	X	X	X	X	X
	call_pairs	X	X	X	X	X	X	X	X	X	X	
	condition_count	X	X	X	X	X	X	X	X	X	X	
	decision_count	X	X	X	X	X	X	X	X	X	X	
	decision_density	X	X	X	X	X	X	X	X			
	design_density	X	X	X	X	X	X	X	X	X	X	
	edge_count	X	X	X	X	X	X	X	X	X	X	
	essential_density	X	X	X	X	X	X	X	X	X	X	
	global_data_complexity	X	X							X	X	

Table 2 (continued)

	NASA MDP datasets (cleaned version)										
	MC2	KC3	MW1	CM1	PC1	PC2	PC3	PC4	PC5	MC1	JM1
global_data_density	X	X							X	X	
maintenance_severity	X	X	X	X	X	X	X	X	X	X	
modified_condition_count	X	X	X	X	X	X	X	X	X	X	
multiple_condition_count	X	X	X	X	X	X	X	X	X	X	
node_count	X	X	X	X	X	X	X	X	X	X	
normalized cyclomatic_compl.	X	X	X	X	X	X	X	X	X	X	
parameter_count	X	X	X	X	X	X	X	X	X	X	
percent_comments	X	X	X	X	X	X	X	X	X	X	
Number of code attributes	39	39	37	37	37	36	37	37	38	38	21
Number of modules	125	194	253	327	705	745	1077	1287	1711	1988	7782
Number of defective modules	44	36	27	42	61	16	134	177	471	46	1672
Percentage of defective modules	35.2%	18.6%	10.7%	12.8%	8.7%	2.1%	12.4%	13.8h	27.5%	2.3%	21.5%

#1. *LOC_total=LOC_executable+LOC_code_and_comment*

#2. *Number_of _lines=LOC_total+LOC_blank+LOC_comments*

Table 3 An overview of the JIT datasets containing Bugzilla (period: 08/1998-12/2006) and Columba (period: 11/2002-07/2006)

Dimension	Name	Definition	Rationale	JIT datasets	
				Bugzilla	Columba
Diffusion	NS	Number of modified subsystems	Change modifying many subsystems are more likely to be defect-prone.	X	X
	ND	Number of modified directories	Changes that modify many directories are more likely to be defect-prone.	X	X
	NF	Number of modified files	Changes touching many files are more likely to be defect-prone.	X	X
	Entropy	Distribution of modified code across each file	Changes with high entropy are more likely to be defect-prone, because a developer will have to recall and track large numbers of scattered changes across each file.	X	X
Size	LAn	Lines of code added (normalized)	The more lines of code added, the more likely a defect is introduced.	X	X
	LDn	Lines of code deleted (normalized)	The more lines of code deleted, the higher the chance of a defect.	X	X
	LTn	Lines of code in a file before the change (normalized)	The larger a file, the more likely a change might introduce a defect.	X	X
Purpose	FIX	Whether or not the change is a defect fix	Fixing a defect means that an error was made in an earlier implementation, therefore it may indicate an area where errors are more likely.	X	X
History	NDEV	The number of developers that changed the modified file	The larger the NDEV, the more likely a defect is introduced, because files revised by many developers often contain different design thoughts and coding styles.	X	X
	AGE	The average time interval between the last and the current change	The lower the AGE (i.e., the more recent the last change), the more likely a defect will be introduced.	X	X
	NUCn	The number of unique changes to the modified files (normalized)	The larger the NUC, the more likely a defect is introduced, because a developer will have to recall and track many previous changes.	X	X
Experience	EXP	Developer experience	More experienced developers are less likely to introduce a defect.	X	X
	REXP	Recent developer experience	A developer that has often modified the files in recent months is less likely to introduce a defect, because he/she will be more familiar with the recent developments in the system.	X	X
	SEXP	Developer experience on a subsystem	Developer that are familiar with the subsystems modified by a change are less likely to introduce a defect.	X	X
Number of metrics				14	14
Number of changes				4620	4455
Number of defective changes				1696	1361
Percentage of defective changes				36.7%	30.5%

- NBTree (Kohavi 1996) is a hybrid algorithm that constructs decision trees with naive Bayes models at the leaves.

Table 4 A summary of the defect predictors compared in this study, where "Abbr." represents the abbreviation and "Impl." gives the implementation platform or language

Category	Technique	Abbr.	Impl.
Rule-based learners	OneR classifier	OneR	Weka
Rule-based learners	RIPPER	JRip	Weka
Decision trees	C4.5	J48	Weka
Decision trees	NBTree	NBTree	Weka
Regression models	Ridge logistic regression	RLR	Weka
Support vector machines	Support vector machine	SVM	Weka
Neural networks	Multilayer perceptron	MLP	Weka
Bayesian learners	Gaussian naive Bayes	NBc	Weka
Bayesian learners	Discrete naive Bayes	NBd	Weka
Bayesian learners	Tree-augmented naive Bayes	TAN	Weka
Bayesian ensemble learners	Averaged one-dependence estimators	AODE	Weka
Bayesian learners	Hidden naive Bayes	HNB	Weka
Decision tree ensemble learners	AdaBoost	AdaBst	Weka
Decision tree ensemble learners	Random forest	RF	Weka
Bayesian learners	Discrete naive Bayes	NBd2	Java, R
Bayesian ensemble learners	Naive Bayes ensemble	NBE	Java, R
Bayesian ensemble learners	Superposed naive Bayes	SNB	Java, R
Bayesian learners	Tree-augmented naive Bayes	TAN2	Java, R
Bayesian ensemble learners	Naive Bayes ensemble + TAN	NBE2	Java, R
Bayesian ensemble learners	Superposed naive Bayes + TAN	SNB2	Java, R

- Ridge logistic regression (RLR; Le Cessie and Van Houwelingen 1992) is a multinomial logistic regression model with a ridge estimator.
- Support vector machine (SVM; Burges 1998) is a binary classifier that finds the maximum margin hyperplane separating two classes of data.
- Multilayer perceptron (MLP; Jain et al. 1996) is a multilayer feed-forward neural network in which each computational unit employs either the thresholding function or the sigmoid function.
- Simple naive Bayes classifier (Domingos and Pazzani 1997; Lewis 1998) is the simplest form of Bayesian networks, which includes Gaussian naive Bayes (NBc; naive Bayes classifier based on Gaussian distribution) and discrete naive Bayes (NBd; naive Bayes classifier based on discretization).
- Tree-augmented naive Bayes (TAN; Friedman et al. 1997) is an extension of simple naive Bayes that relaxes the conditional independence assumption among variables.
- Averaged one-dependence estimators (AODE; Webb et al. 2005) is a Bayesian ensemble learner that averages the estimates of all the one dependence estimators.
- Hidden naive Bayes (HNB; Jiang et al. 2009) is a Bayesian learner in which a hidden parent is created for each variable that combines the influences from all other variables.
- AdaBoost (AdaBst; Freund and Schapire 1997) is a popular implementation of boosting that uses a multiplicative weight-update technique.
- Random forest (RF; Breiman 2001) is an advanced extension of bagging that uses a random subset of predictors to define the best split at each node.

The last 6 are the proposed methods including intermediate models, implemented in Java and R.

- Our implementation of discrete naive Bayes (NBd2; see Section 3.1) is firstly combined into naive Bayes ensemble (NBE; see Section 3.3) and then transformed into superposed naive Bayes (SNB; see Section 3.4).
- Our implementation of TAN (TAN2; see Section 3.2) is firstly combined into naive Bayes ensemble (NBE2; see Section 3.3) and then transformed into superposed naive Bayes (SNB2; see Section 3.4).

The parameter settings of the proposed methods are as follows: the number of iterations $N = 100$; threshold for discretization $BIC \leq -2$; subsampling ratio $\eta = 0.5$; shrinkage parameter $\nu = 0.1$. Note that in our experiments we use the same number of iterations (100 iterations) for all ensemble learners, i.e., AODE, AdaBst, RF, NBE, SNB, NBE2, and SNB2.

4.3 Evaluation of Predictive Accuracy

In order to evaluate the predictive accuracy of defect predictors, we adopt a similar approach to Ghotra et al. (2015). Figure 5 shows an overview of model construction and performance evaluation in our experiments.

The predictive models are evaluated using the 5-fold cross-validation, which divides a dataset into five folds of equal size. Of the five folds, four folds (80%) are used for the training data, and one fold (20%) is left for the testing data. The training data is used to train the models using different defect predictors, whereas the testing data is used to analyze model performance. This process is repeated five times, using each fold as the testing data once. Furthermore, the entire 5-fold cross-validation is also repeated five times (25 total iterations) with different random data partition. Appendix 3 lists the random seeds used to generate different data partitions in the experiments with Weka. Note that the same data partitions were also used in the experiments with R.

We use the area under the receiver operating characteristic (ROC) curve (AUC; Fawcett 2006) as a performance measure. A ROC curve plots the false positive rate against the true positive rate. AUC ranges from 0 to 1, where larger AUC indicates better performance. AUC less than 0.5 is equivalent to random guessing.

We also adopt the Scott-Knott test (Jelihovschi et al. 2014) to partition the predictors into statistically distinct ranks ($\alpha = 0.05$). The Scott-Knott test uses a hierarchical cluster analysis to

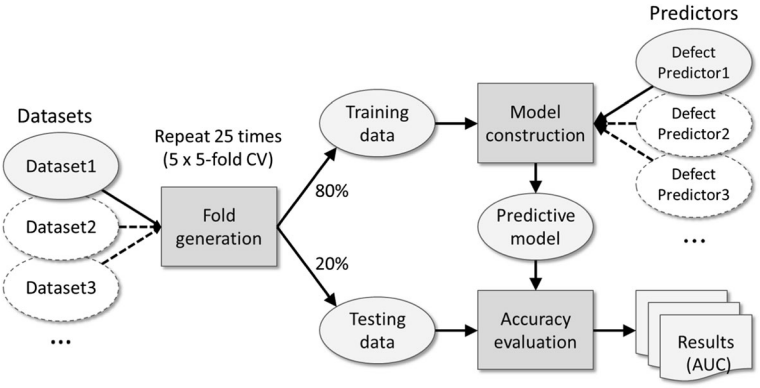


Fig. 5 An overview of model construction and accuracy evaluation in our experiments, which is a similar approach to Ghotra et al. (2015)

find the homogeneous groups. Figure 6 shows the adopted approach for ranking predictors by using a double Scott-Knott test. The first run of the Scott-Knott test is performed on each individual dataset, and then the second run is performed on the outputs obtained by the first run to find the final statistically distinct ranks of the predictors.

The output ranks are transformed into reversed fractional ranks (RFRs), i.e., the fractional ranks (Bury and Wagner 2008) in reverse order divided by the total number of items. Table 5 shows a computational example of RFRs; a fractional rank *FR* is given by the mean of the ordinal ranks in the same group, and the corresponding reversed fractional rank *RFR* is derived as follows:

$$RFR = (N - FR + 1)/N \tag{10}$$

where N is the total number of items. The items in the same group have equal RFRs. It is evident from the definition that RFR ranges from 0 to 1, where larger RFR indicates ranked higher, and the total sum of RFRs is constantly equal to $(N+1)/2$.

4.4 Evaluation of Interpretability

In order to evaluate the interpretability of defect predictors, we need to clarify interpretability measures. However, comparing the interpretability of different classification techniques is a difficult task, because typical size-based measures such as the number of terms, rules, or tree nodes are all model-specific. Freitas (2014) addressed that the interpretability of different types of classification models cannot be measured in a fully objective way. Even user-based experiments can be easily biased by various factors, such as the characteristics of datasets and the background of the users. Therefore, in this paper, we adopt a qualitative approach to assess the interpretability of different defect predictors based on multiple criteria.

We use Lipton (2016)'s categories of interpretability for the assessment criteria. We focused on the transparency (i.e., *how does the model work?*), which is comprised of model transparency (simulatability), component transparency (decomposability), and algorithmic

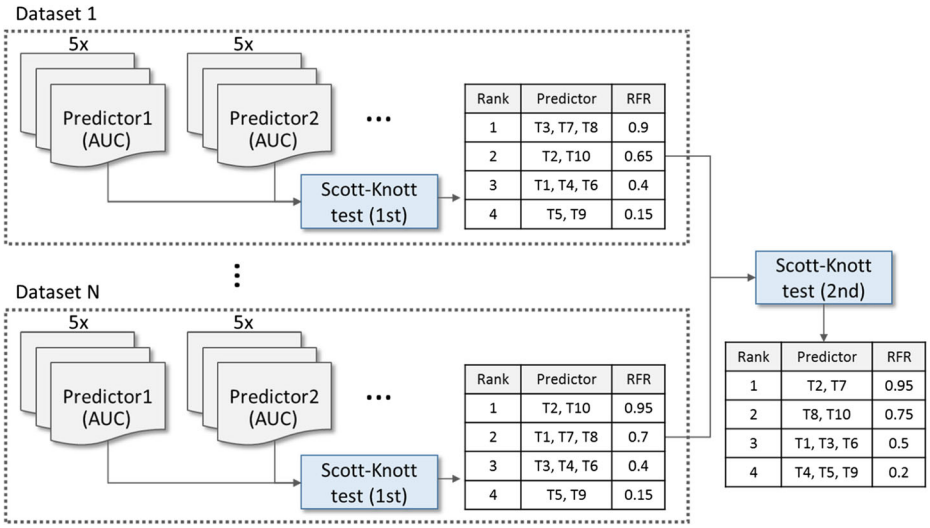


Fig. 6 A double Scott-Knott test for ranking defect predictors, where RFR represents reversed fractional rank

Table 5 A computational example of reversed fractional ranks (RFRs), where the items in the same group have equal ranks

Group	Item	Ordinal rank	Fractional rank	Reversed fractional rank (RFR)
Group 1	A	1	(1+2+3) / 3 = 2	(10-2+1) / 10 = 0.9
	B	2	2	0.9
	C	3	2	0.9
Group 2	D	4	(4+5) / 2 = 4.5	(10-4.5+1) / 10 = 0.65
	E	5	4.5	0.65
Group 3	F	6	(6+7+8) / 3 = 7	(10-7+1) / 10 = 0.4
	G	7	7	0.4
	H	8	7	0.4
Group 4	I	9	(9+10) / 2 = 9.5	(10-9.5+1) / 10 = 0.15
	J	10	9.5	0.15

transparency. A model transparency is defined as whether a user can contemplate the entire model at once. A component transparency is defined as whether each part of the model admits an intuitive explanation such as a plain text description. An algorithmic transparency is defined as whether a user can understand how the learning algorithm behaves. Based on these criteria, we set the following questions to assess the interpretability of predictors:

Q1. Is the entire model simple enough to be fully understood by a user?
Q2. Is each part of the model (each input, parameter, and calculation) intuitively explainable?
Q3. Is the algorithm deterministic (non-stochastic) without using any random numbers?

In our assessment, Q1 through Q3 are applied to each instance of the predictors, without any use of post-hoc explanation techniques such as partial dependence plots (PDP; Friedman 2001) or Local Interpretable Model-agnostic Explanations (LIME; Ribeiro et al. 2016).

5 Results and Discussion

5.1 (RQ1) How Accurate are the Current Defect Predictors?

We compared the accuracy of the proposed method (SNB) with those of the current defect predictors based on a similar approach to Ghotra et al. (2015), as described in Section 4.3. Table 6 shows the mean (upper number) and the standard deviation (lower number) of AUC values in five repetitions of 5-fold cross-validation for 20 defect predictors over 13 real-world public datasets. The top three highest mean AUC values for a particular dataset are highlighted in boldface. The second last column gives the mean across all the datasets, and the last column gives the ranking of the mean AUC values. As can be seen in Table 6, the AUC values vary considerably among predictors, among datasets, and even among repetitions of cross-validation, thus supporting our approach for ranking defect predictors by using a double Scott-Knott test.

Table 6 The mean (upper number) and the standard deviation (lower number) of AUC values in five repetitions of 5-fold cross-validation for 20 defect predictors over 13 real-world public datasets

	MC2	KC3	MW1	CM1	PC1	PC2	PC3	PC4	PC5	MC1	JM1	bugzilla	columba	Mean	Rank
OneR	0.583	0.554	0.536	0.534	0.548	0.499	0.546	0.626	0.571	0.508	0.540	0.576	0.549	0.552	20
	(0.024)	(0.024)	(0.044)	(0.018)	(0.017)	(0.000)	(0.006)	(0.008)	(0.007)	(0.005)	(0.003)	(0.005)	(0.004)	(0.013)	
Jrip	0.586	0.620	0.587	0.508	0.578	0.483	0.5151	0.700	0.617	0.560	0.559	0.690	0.650	0.591	19
	(0.033)	(0.054)	(0.021)	(0.018)	(0.022)	(0.005)	(0.024)	(0.014)	(0.013)	(0.013)	(0.005)	(0.007)	(0.013)	(0.019)	
J48	0.608	0.569	0.503	0.567	0.679	0.544	0.635	0.745	0.678	0.587	0.624	0.748	0.669	0.627	18
	(0.021)	(0.037)	(0.039)	(0.040)	(0.068)	(0.061)	(0.042)	(0.034)	(0.014)	(0.058)	(0.008)	(0.006)	(0.011)	(0.034)	
NBTree	0.645	0.569	0.594	0.647	0.783	0.689	0.763	0.874	0.704	0.739	0.654	0.772	0.713	0.704	16
	(0.034)	(0.079)	(0.072)	(0.050)	(0.043)	(0.050)	(0.012)	(0.016)	(0.012)	(0.032)	(0.004)	(0.002)	(0.005)	(0.031)	
RLR	0.627	0.668	0.633	0.688	0.826	0.732	**0.820**	0.900	0.741	0.726	0.673	0.751	0.724	0.732	11
	(0.048)	(0.064)	(0.027)	(0.045)	(0.019)	(0.047)	(0.005)	(0.004)	(0.003)	(0.029)	(0.001)	(0.000)	(0.001)	(0.023)	
SVM	0.606	**0.684**	0.705	0.673	0.805	0.710	0.735	0.896	0.707	0.617	0.653	0.707	0.688	0.707	14
	(0.025)	(0.020)	(0.008)	(0.052)	(0.021)	(0.036)	(0.032)	(0.003)	(0.001)	(0.009)	(0.001)	(0.005)	(0.004)	(0.017)	
MLP	**0.719**	0.612	0.609	0.614	0.756	0.683	0.772	0.881	0.722	0.702	0.653	0.729	0.722	0.705	15
	(0.040)	(0.057)	(0.023)	(0.043)	(0.027)	(0.016)	(0.013)	(0.006)	(0.007)	(0.020)	(0.003)	(0.006)	(0.006)	(0.021)	
NBc	0.707	0.649	0.695	0.665	0.750	0.714	0.737	0.814	0.694	0.691	0.633	0.676	0.661	0.699	17
	(0.008)	(0.013)	(0.027)	(0.022)	(0.013)	(0.053)	(0.012)	(0.014)	(0.006)	(0.030)	(0.003)	(0.003)	(0.006)	(0.016)	
NBd	0.627	0.607	0.692	0.689	0.803	0.792	0.766	0.811	0.726	0.729	0.668	0.731	0.732	0.721	13
	(0.023)	(0.045)	(0.013)	(0.025)	(0.015)	(0.013)	(0.009)	(0.004)	(0.001)	(0.016)	(0.002)	(0.001)	(0.003)	(0.013)	
TAN	0.615	0.606	0.687	0.716	0.837	**0.815**	0.794	0.881	0.747	0.801	0.671	**0.774**	0.743	0.745	9
	(0.041)	(0.043)	(0.025)	(0.035)	(0.005)	(0.018)	(0.008)	(0.002)	(0.006)	(0.013)	(0.004)	(0.002)	(0.007)	(0.016)	
AODE	0.643	0.608	0.703	**0.721**	0.840	**0.818**	0.795	0.875	0.728	0.810	0.670	0.765	0.745	0.748	8
	(0.027)	(0.045)	(0.007)	(0.014)	(0.008)	(0.012)	(0.007)	(0.006)	(0.004)	(0.013)	(0.003)	(0.001)	(0.004)	(0.011)	
HNB	0.618	0.605	0.678	**0.719**	**0.845**	0.810	0.799	0.874	0.747	0.795	**0.681**	0.768	0.741	0.745	10
	(0.030)	(0.052)	(0.026)	(0.014)	(0.006)	(0.012)	(0.008)	(0.006)	(0.011)	(0.020)	(0.002)	(0.003)	(0.005)	(0.015)	
AdaBst	0.688	0.671	**0.709**	**0.717**	0.838	0.762	0.806	**0.917**	0.738	0.832	**0.675**	**0.779**	0.747	0.760	4
	(0.028)	(0.037)	(0.035)	(0.031)	(0.015)	(0.016)	(0.009)	(0.003)	(0.006)	(0.022)	(0.003)	(0.002)	(0.003)	(0.016)	
RF	0.706	**0.722**	**0.719**	0.702	**0.868**	0.785	**0.831**	**0.936**	**0.792**	**0.883**	**0.695**	**0.819**	**0.785**	**0.788**	1
	(0.016)	(0.015)	(0.025)	(0.029)	(0.009)	(0.020)	(0.005)	(0.002)	(0.006)	(0.035)	(0.005)	(0.002)	(0.002)	(0.013)	
NBd2	0.677	0.658	0.703	0.676	0.781	0.770	0.776	0.811	0.728	0.759	0.670	0.724	0.744	0.729	12
	(0.012)	(0.024)	(0.005)	(0.010)	(0.007)	(0.013)	(0.004)	(0.004)	(0.001)	(0.015)	(0.001)	(0.002)	(0.001)	(0.008)	
NBE	0.678	0.681	0.700	0.709	0.835	0.805	0.803	0.891	0.735	**0.839**	0.674	0.768	**0.764**	0.760	3
	(0.013)	(0.020)	(0.016)	(0.010)	(0.007)	(0.015)	(0.005)	(0.003)	(0.001)	(0.029)	(0.001)	(0.002)	(0.001)	(0.010)	

Table 6 (continued)

	MC2	KC3	MW1	CM1	PC1	PC2	PC3	PC4	PC5	MC1	JM1	bugzilla	columba	Mean	Rank
SNB	0.678 (0.013)	0.673 (0.025)	**0.707** (0.014)	0.706 (0.004)	0.827 (0.008)	**0.815** (0.012)	0.801 (0.005)	0.885 (0.004)	0.735 (0.001)	0.821 (0.025)	0.674 (0.001)	0.770 (0.002)	**0.765** (0.002)	0.758 (0.009)	6
TAN2	0.691 (0.017)	0.683 (0.022)	0.703 (0.016)	0.688 (0.017)	0.822 (0.009)	0.760 (0.007)	0.798 (0.007)	0.886 (0.005)	0.751 (0.005)	0.784 (0.016)	0.672 (0.004)	0.757 (0.004)	0.753 (0.005)	0.750 (0.010)	7
NBE2	**0.714** (0.020)	**0.685** (0.020)	0.704 (0.024)	0.712 (0.018)	**0.851** (0.011)	0.777 (0.020)	0.808 (0.006)	**0.903** (0.006)	**0.754** (0.008)	**0.846** (0.016)	0.674 (0.004)	0.772 (0.004)	0.758 (0.007)	**0.766** (0.013)	2
SNB2	**0.707** (0.022)	0.679 (0.020)	0.696 (0.025)	0.716 (0.014)	0.839 (0.006)	0.778 (0.014)	**0.809** (0.008)	0.899 (0.006)	**0.752** (0.008)	0.796 (0.034)	0.674 (0.004)	0.772 (0.004)	0.756 (0.007)	0.760 (0.013)	5

The top three highest mean AUC values for a particular dataset are highlighted in boldface

Table 7 summarizes the results of the 1st Scott-Knott test. Each value in the table indicates a reversed fractional rank (RFR) of predictive accuracy. The predictor yielding the best RFR for a particular dataset is highlighted in boldface. The second last column gives the mean RFRs across all the datasets, and the last column gives the ranking of the mean RFRs. As shown in Table 7, the predictors with statistically insignificant differences in the AUC values are given the same RFR, but considerable variations among datasets still exist.

Figure 7 gives the result of the second Scott-Knott test on the RFRs of the first Scott-Knott tests, where predictors are sorted according to the mean RFR in descending order. After the second run, the defect predictors are clustered into 4 groups: the first group containing RF, NBE2, NBE, SNB2, AdaBst, and SNB, the second group containing HNB, TAN, TAN2, AODE, and RLR, the third group containing NBd2, NBd, SVM, NBc, MLP, and NBTree, and the fourth group containing J48, JRip, and OneR, where the RFR of each group is computed as 0.875, 0.6, 0.325, and 0.1, respectively. It is noteworthy that the proposed methods and the intermediate models, i.e., superposed naive Bayes classifiers (SNB and SNB2) and naive Bayes ensembles (NBE and NBE2) are all graded as the first group (RFR = 0.875), together with random forest (RF) and AdaBoost (AdaBst). In particular, compared with popular Bayesian learners such as Gaussian naive Bayes (NBc), discrete naive Bayes (NBd), tree-augmented naive Bayes (TAN), averaged one-dependence estimators (AODE), and hidden naive Bayes (HNB), the proposed methods (SNB and SNB2) are shown to achieve better predictive accuracy with statistical significance.

5.2 (RQ2) How Can We Assess the Interpretability of the Predictors?

As described in Section 4.4, we perform a qualitative assessment of interpretability based on Lipton (2016)'s criteria. Table 8 summarizes the assessment results on the interpretability of various defect predictors, where three assessment questions related to model transparency, component transparency, and algorithmic transparency are applied. We assume that the interpretability of the predictors would be ranked by the number of "Yes" responses to the assessment questions. Hence, the reversed fractional rank (RFR) of interpretability for each predictor is calculated based on the "Yes" counts, as shown in Table 8.

In the following, we explain the rationales of our assessment for each predictor.

- Rule-based learners such as OneR and JRip are considered transparent in all three levels.
- J48 would be transparent at component level, as each node can correspond to a plain text description. However, it would not always be transparent at model level, because the size of the model may grow much larger than that a user can understand. For example, J48 models generated by Weka for Bugzilla and Columba have 241 nodes and 245 nodes, respectively.
- NBTree would be model transparent, as the entire model looks compact. However, we assess the component transparency as "not enough" because of the difficulty in qualitative understanding between the input variables and the response.
- As RLR is a type of linear regression models, it is considered transparent in all three levels.
- SVM and MLP are both regarded as black-box in all three levels. They are not ensemble models, but random numbers may be used for the parameter tuning inside the algorithms.

Table 7 A summary of the results of the 1st Scott-Knott test, where each value indicates a reversed fractional rank (RFR) of predictive accuracy

	MC2	KC3	MW1	CM1	PC1	PC2	PC3	PC4	PC5	MC1	JM1	bugzilla	columba	Mean	Rank
OneR	0.125	0.1	0.075	0.075	0.05	0.075	0.075	0.05	0.05	0.05	0.05	0.05	0.05	0.067	20
Jrip	0.125	0.325	0.225	0.075	0.1	0.075	0.075	0.1	0.1	0.125	0.1	0.15	0.1	0.129	19
J48	0.125	0.1	0.075	0.15	0.15	0.15	0.15	0.15	0.15	0.125	0.15	0.425	0.2	0.162	18
NBTree	0.375	0.1	0.225	0.4	0.375	0.225	0.375	0.5	0.275	0.425	0.3	0.75	0.3	0.356	17
RLR	0.375	**0.75**	0.225	0.4	**0.75**	0.35	**0.975**	0.8	0.825	0.425	0.775	0.425	0.375	0.573	11
SVM	0.125	**0.75**	**0.675**	0.4	0.375	0.35	0.225	0.8	0.275	0.2	0.3	0.2	0.25	0.379	14
MLP	**0.9**	0.325	0.225	0.2	0.225	0.225	0.375	0.5	0.425	0.275	0.3	0.325	0.375	0.360	16
NBc	**0.9**	**0.75**	**0.675**	0.4	0.225	0.35	0.225	0.25	0.2	0.275	0.2	0.1	0.15	0.362	15
NBd	0.375	0.325	**0.675**	0.4	0.375	0.6	0.375	0.25	0.425	0.425	0.5	0.325	0.45	0.423	13
TAN	0.375	0.325	**0.675**	**0.8**	**0.75**	**0.9**	0.7	0.5	0.825	0.675	0.5	0.75	0.6	0.644	8
AODE	0.375	0.325	**0.675**	**0.8**	**0.75**	**0.9**	0.7	0.5	0.425	0.675	0.5	0.75	0.6	0.613	10
HNB	0.375	0.325	**0.675**	**0.8**	**0.75**	**0.9**	0.7	0.5	0.825	0.675	0.95	0.55	0.6	0.663	7
AdaBst	0.65	**0.75**	**0.675**	**0.8**	**0.75**	0.6	0.7	0.95	0.6	0.9	0.775	0.95	0.6	0.746	5
RF	**0.9**	**0.75**	**0.675**	**0.8**	**0.75**	**0.6**	**0.975**	**1**	**1**	**1**	**1**	**1**	**1**	**0.881**	1
NBd2	0.65	**0.75**	**0.675**	0.4	0.375	0.6	0.375	0.25	0.425	0.425	0.5	0.25	0.6	0.483	12
NBE	0.65	**0.75**	**0.675**	**0.8**	**0.75**	**0.9**	0.7	0.8	0.6	0.9	0.775	0.75	0.925	0.767	3
SNB	0.65	**0.75**	**0.675**	**0.8**	**0.75**	**0.9**	0.7	0.5	0.6	0.675	0.775	0.75	0.925	0.727	6
TAN2	0.65	**0.75**	**0.675**	0.4	**0.75**	0.6	0.7	0.5	0.825	0.675	0.5	0.5	0.8	0.640	9
NBE2	**0.9**	**0.75**	**0.675**	**0.8**	**0.75**	0.6	0.7	0.8	0.825	0.9	0.775	0.75	0.8	0.771	2
SNB2	**0.9**	**0.75**	**0.675**	**0.8**	**0.75**	0.6	0.7	0.8	0.825	0.675	0.775	0.75	0.8	0.754	4

The highest RFR for a particular dataset is highlighted in boldface

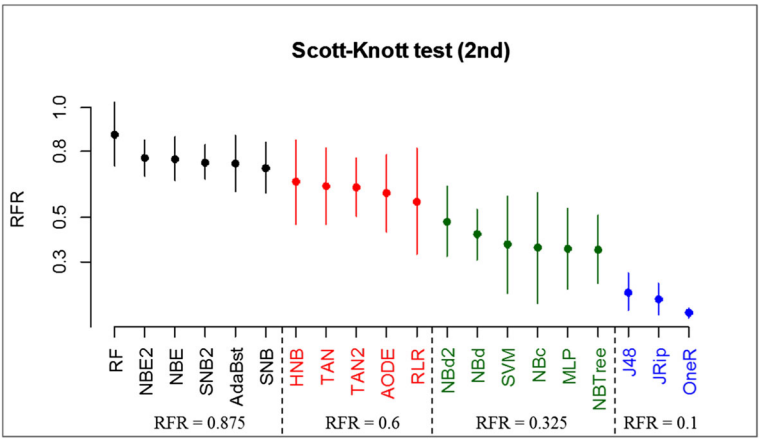


Fig. 7 The result of the 2nd Scott-Knott test on the RFRs of the 1st Scott-Knott tests, where predictors are sorted according to the mean RFR in descending order

- NBc and NBd are both considered transparent. They are only different in the assumed probability distributions, i.e., NBc uses Gaussian distributions, while NBd uses step functions.
- TAN would be model transparent, as the entire model looks compact. However, we assess the component transparency as "not enough" because of the difficulty in qualitative understanding between the input variables and the response.

Table 8 A summary of the assessment results on the interpretability of various defect predictors

	Model Transparency (Simulatability) Q1. Is the entire model simple enough to be fully understood by a user?	ComponentTransparency (Decomposability) Q2. Is each part of the model (each input, parameter, and calculation) intuitively explainable?	Algorithmic Transparency Q3. Is the algorithm deterministic (non-stochastic) without using any random numbers?	# of Yes	RFR
OneR	Yes	Yes	Yes	3	0.875
JRip	Yes	Yes	Yes	3	0.875
J48		Yes	Yes	2	0.6
NBTree	Yes		Yes	2	0.6
RLR	Yes	Yes	Yes	3	0.875
SVM				0	0.18
MLP				0	0.18
NBc	Yes	Yes	Yes	3	0.875
NBd	Yes	Yes	Yes	3	0.875
TAN	Yes		Yes	2	0.6
AODE			Yes	1	0.4
HNB			Yes	1	0.4
AdaBst				0	0.175
RF				0	0.175
NBd2	Yes	Yes	Yes	3	0.875
NBE				0	0.175
SNB	Yes	Yes		2	0.6
TAN2	Yes		Yes	2	0.6
NBE2				0	0.175
SNB2	Yes			1	0.4

- AODE and HNB seem black-box in model and component levels, but rather transparent in algorithm level. Although AODE is a Bayesian ensemble learner, the algorithm itself is deterministic (not using random numbers).
- AdaBst and RF are both considered black-box in all three levels. Note that post-hoc explanation techniques such as partial dependence plots (PDP) are not used in the assessment.
- Since NBd2 is a different implementation of NBd, the assessment result would be the same as that of NBd, i.e., transparent in all three levels.
- Since TAN2 is a different implementation of TAN, the assessment result would be the same as that of TAN, i.e, model and algorithmic transparent.
- NBE and NBE2 are both regarded as black-box in all three levels, as are AdaBst and RF.
- Since SNB have the same structure as NBd2, the model and component transparency would be inherited from NBd2. However, the algorithmic transparency would be low, because SNB is an ensemble learner.
- Since SNB2 have the same structure as TAN2, the model and component transparency would be inherited from TAN2. However, the algorithmic transparency would be low, because SNB2 is an ensemble learner.

5.3 (RQ3) How Can We Balance the Trade-Off Between Accuracy and Interpretability?

In order to analyze the trade-off between predictive accuracy and interpretability of defect predictors, we combine the experimental results measuring predictive accuracy with the assessment results of interpretability. Figure 8 shows a scatter plot comparing RFRs of predictive accuracy (as shown in Fig. 7) to those of interpretability (as shown in Table 7). The diagonal line $y = 1 - x$ illustrates a generally accepted assumption that the higher the classifier's predictive accuracy the lower its interpretability.

As can be seen, superposed naive Bayes, SNB and SNB2, are positioned in the upper-right region, which means higher accuracy with higher interpretability. Each of dotted lines connecting {NBd2, NBE, and SNB} and {TAN2, NBE2, and SNB2} represents the chain of model transformations. NBd2 moves to NBE along the diagonal line by using the ensemble algorithm (described in Section 3.3) and then moves upward to SNB by the superposition of weights of evidence (described in Section 3.4). Similarly, TAN2 moves to NBE2 along the diagonal line and then moves upward to SNB2. Our experiment results show that the proposed methods, i.e., SNB and SNB2, can produce balanced outputs that satisfy both accuracy and interpretability criteria.

5.4 Detailed Comparison of RLR, SNB, and RF + PDP

In this section, we present a detailed comparison of ridge logistic regression (RLR), superposed naive Bayes (SNB), and random forest (RF) with partial dependence plots (PDP), mainly from the viewpoint of interpretability. We select RFR because its position in Fig. 8 indicates the highest rank of interpretability with moderate predictive accuracy. We also select RF because it achieved the highest accuracy in most of the datasets and its interpretability could be largely enhanced by the use of PDP.

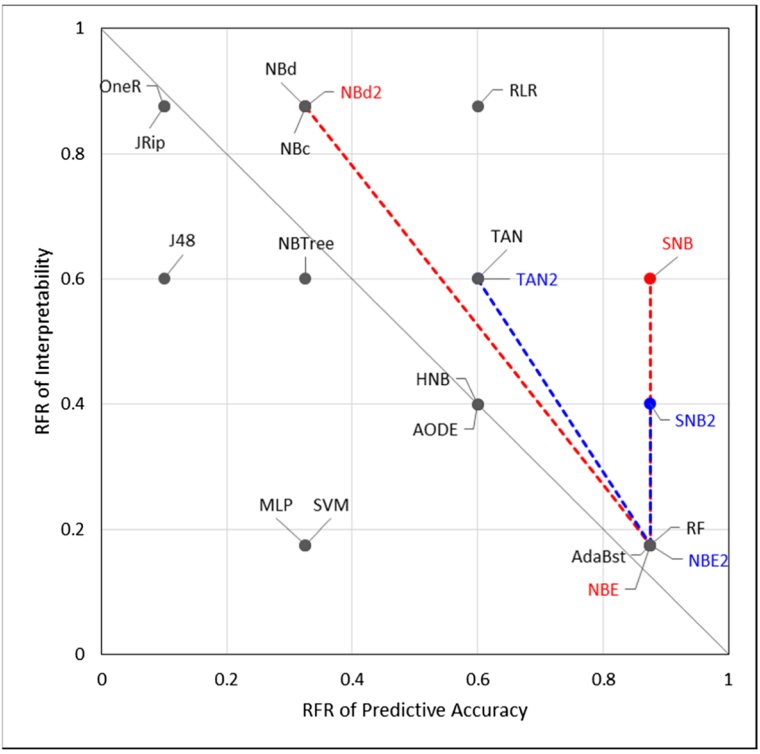


Fig. 8 A scatter plot comparing RFRs of the predictive accuracy to those of the interpretability of defect predictors

Table 9 gives a summary of the interpretation of RLR, SNB, and RF+PDP models generated from Bugzilla and Columba datasets. The summary table includes the importance rank and the expected and observed direction of each independent variable, where the direction is classified into four types, i.e., risk-increasing (+), risk-decreasing (-), non-monotonic (+/-), or constant (0). The expected directions of the variables are determined based on the rationales of the JIT metrics shown in Table 3, whereas the importance ranks and the observed directions are dependent on each predictor, as follows.

Firstly, in a RLR model, the variable importance is calculated as the absolute value of the natural logarithm of the odds ratio ($abs(log_e OR)$) with the standardized data, where the odds ratio (OR) is the probability of $Y = 1$ divided by that of $Y = 0$. The sign of the coefficient determines the observed direction of the corresponding variable, i.e., a positive or negative coefficient means a risk-increasing or risk-decreasing factor, respectively.

Secondly, in a SNB model, the variable importance is provided as the reversal of BIC (see Appendix 1) The observed direction of an independent variable is determined by the evaluation of the weights of evidence (WoE) plot that illustrates the relationship between the value of the independent variable X_i and the corresponding WoE, where a positive WoE indicates that the value of X_i is evidence in favor of the hypothesis that $Y = 1$.

Table 9 A summary of the interpretation of RLR, SNB, and RF+PDP models generated from Bugzilla and Columba datasets, where "Rank" gives the importance rank and "Dir." represents the direction of each independent variable

Dimension	Metrics (expected Dir.)		Bugzilla						Columba					
			RLR		SNB		RF+PDP		RLR		SNB		RF+PDP	
			Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.
Diffusion	NS	(+)	12	+	7	+	14	+	13	−	13.5	0	14	+
	ND	(+)	6	+	5	+	13	+	1	+	4	+	11	+
	NF	(+)	3	+	3	+	9	+	7	−	2	+	7	+
	Entropy	(+)	7	+	4	+/−	10	+/−	14	−	6	+/−	10	+/−
Size	LAn	(+)	11	+	2	+	1	+	11	+	7	+	5	+
	LDn	(+)	2	−	9	+	4	0	8	+	8	+	6	+
	LTn	(+)	8	−	6	+/−	3	+/−	10	+	1	+	1	+
Purpose	FIX	(+)	4	+	14	0	11	+	12	+	13.5	0	13	+
History	NDEV	(+)	14	−	12	+	8	+/−	9	+	11	+	12	+
	AGE	(−)	1	−	1	−	2	−	4	−	10	+/−	8	+
	NUCn	(+)	13	+	8	−	12	−	2	−	3	−	9	−
Experience	EXP	(−)	10	−	10	−	5	+/−	3	−	5	+/−	2	+/−
	REXP	(−)	5	−	11	−	7	+/−	6	+	12	+/−	3	+/−
	SEXP	(−)	9	+	13	+/−	6	+/−	5	+	9	+/−	4	+/−

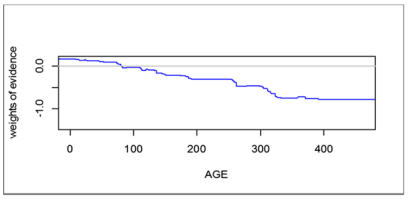
Direction: increasing (+), decreasing (−), non-monotonic (+/−), constant (0)

Thirdly, in a RF+PDP model, the variable importance is given as the mean decrease Gini. The observed direction of an independent variable is determined by the evaluation of the partial dependence plot (PDP) that visualizes the average partial relationship between the independent variable and the predicted response, where a positive value means that the positive class ($Y = 1$) is more likely for that values of the independent variable.

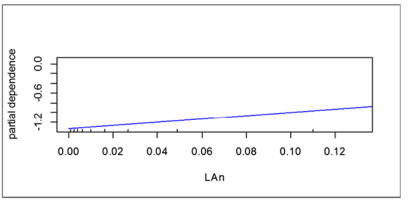
Figure 9 shows examples of WoE plots in a SNB model and PDPs in a RF model, both generated from Bugzilla dataset. The top three most important variables in the SNB model are AGE (Fig. 9a), LAn (Fig. 9b), and NF (Fig. 9c), whose observed directions are risk-decreasing (-), risk-increasing (+), and risk-increasing (+), respectively. On the other hand, the top three most important variables in the RF model are LAn (Fig. 9d), AGE (Fig. 9e), and LTn (Fig. 9f), whose observed directions are risk-increasing (+), risk-decreasing (-), and non-monotonic (+/-), respectively.

Tables 10, 11, 12, 13, 14 and 15 give summaries of the interpretation of RLR, SNB, and RF+PDP models generated from the MDP datasets, i.e., MC2, KC3, MW1, CM1, PC1, PC2, PC3, PC4, PC5, MC1, and JM1 datasets, which includes the importance rank and the expected and observed direction of each independent variable. The expected directions of the variables are determined based on the definitions of the MDP metrics, referring to Jiang et al. (2008b). The importance ranks and the observed directions are dependent on each predictor, as described above in this section.

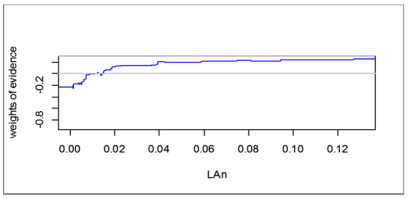
As can be seen, although the importance ranks and the observed directions vary considerably among predictors and datasets, there are several findings to be addressed in our results. The first finding is about the correlation of importance ranks. It can be observed that a rank vector of SNB generally has a high correlation with that of RF+PDP, but little similarity to that of RLR, particularly in the MDP datasets. For example, *Halstead_content* and *Halstead_effort* have relatively higher ranks in SNB and RF+PDP, but not in RLR. Conversely,



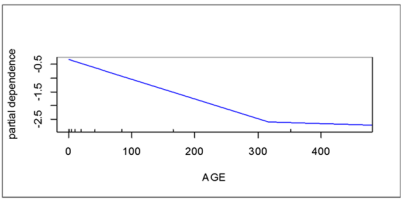
(a) SNB: AGE (Rank = 1, Dir. = −)



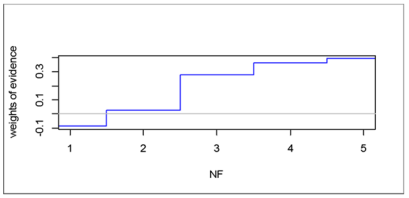
(d) RF+PDP: LAn (Rank = 1, Dir. = +)



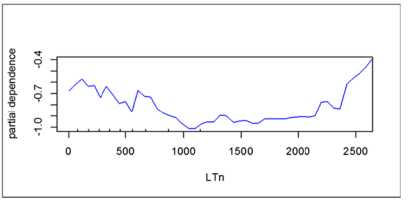
(b) SNB: LAn (Rank = 2, Dir. = +)



(e) RF+PDP: AGE (Rank = 2, Dir. = −)



(c) SNB: NF (Rank = 3, Dir. = +)



(f) RF+PDP: LTn (Rank = 3, Dir. = + / −)

Fig. 9 Weights of evidence (WoE) plots of top three most important variables (**a** AGE. **b** LAn. **c** NF.) in a SNB model, and partial dependence plots (PDPs) of top three most important variables (**d** LAn. **e** AGE. **f** LTn.) in a RF model, both generated from Bugzilla dataset

condition_count, *modified_condition_count*, and *multiple_condition_count* have relatively higher ranks in RLR, but not in SNB and RF+PDP. This difference may imply that the results of RLR are more affected by the multicollinearity among variables, because *condition_count*, *modified_condition_count* and *multiple_condition_count* have strong correlations of more than 0.99.

The second finding relates to the non-monotonic (+/-) directions of variables. In our results, non-monotonic directions are most frequently found in RF+PDP models, which account for more than 40 percent of all the PDPs. Non-monotonic directions are also observed in about 20 percent of the WoE plots in SNB, whereas RLR produces only monotonic directions, i.e., risk-increasing (+) or risk-decreasing (-). An example of the non-monotonic direction that seems difficult to interpret is shown in Fig. 9f. Thus, from the viewpoint of interpretability, non-monotonic directions should be avoided as much as possible, even though they are effective in improving the predictive accuracy. It is true that PDPs could complement the interpretability of RF to some extent, but it should be also noted that PDPs may not work correctly if there are substantial interaction effects between variables (Goldstein et al. 2015). A RF model itself is still black-box, even with the help of PDPs.

Table 10 A summary of the interpretation of RLR, SNB, and RF+PDP models generated from MC2 and KC3 datasets, where "Rank" gives the importance rank and "Dir." represents the direction of each independent variable

	Metrics (expected Dir.)		MC2						KC3					
			RLR		SNB		RF + PDP		RLR		SNB		RF + PDP	
			Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.
LOC counts	LOC_total	(+)	33	−	28	+	27	+/−	11	−	16	+	12	+/−
	LOC_blank	(+)	14	+	24	+	9	+	18	−	3	+	10	+
	LOC_code_and_comment	(+)	28	−	34	+	33	+/−	24	+	1	+	1	+
	LOC_comments	(+)	35	+	9	+	4	+	15	−	34	+	20	+
	LOC_executable	(+)	37	−	29	+	29	+/−	10	−	15	+	22	+/−
	Number_of_lines	(+)	13	−	19	+	5	+/−	6	+	10	+	3	+/−
Halstead attributes	content	(+)	20	+	35	+	18	+/−	16	+	7	+	6	+/−
	difficulty	(+)	23	−	1	+	1	+	26	−	27	+	18	+/−
	effort	(+)	10	+	2.5	+	2	+	31	−	5.5	+/−	4	+/−
	error_est	(+)	1	+	30	+	30	+	3	−	17	+	28	+
	length	(+)	22	−	25	+	25	+/−	21	−	11	+	23	+
	level	(+)	32	−	8	−	20	−	28	+	36	−	25	+
	prog_time	(+)	11	+	2.5	+	6	+	30	−	5.5	+/−	5	+
	volume	(+)	2	−	26	+	24	+/−	1	+	9	+	14	+/−
	num_operands	(+)	17	+	23	+	19	+	13	−	19	+	9	+/−
	num_operators	(+)	12	−	22	+	22	+	32	−	13	+	19	+/−
	num_unique_operands	(+)	15	−	32	+	38	+/−	14	−	18	+	7	+
	num_unique_operators	(+)	16	+	18	+	10	+	25	+	23	+	26	+/−
McCabe attributes	cyclomatic_complexity	(+)	8	−	10	+	15	+	8	+	14	+	17	+
	cyclomatic_density	(+)	36	−	38	+	17	+/−	29	+	22	−	16	+/−
	design_complexity	(+)	18	+	21	+	26	+	39	+	25	+	30	+
	essential_complexity	(+)	19	−	17	+	32	+	37	−	32	+	34	+
Miscellaneous	branch_count	(+)	5	+	16	+	23	+	7	−	12	+	8	+
	call_pairs	(+)	24	−	12	+	12	+	19	−	8	+	11	+
	condition_count	(+)	6	−	13.5	+	35	+	5	+	28.5	+	35	+/−
	decision_count	(+)	4	−	20	+	31	+	9	+	33	+	27	+
	decision_density	(+)	34	−	33	+	39	+/−	38	−	35	0	39	0
	design_density	(+)	29	−	37	+/−	21	+/−	35	+	37	−	33	−

Table 10 (continued)

Metrics (expected Dir.)		MC2						KC3					
		RLR		SNB		RF + PDP		RLR		SNB		RF + PDP	
		Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.
edge_count	(+)	21	+	5	+	3	+	17	+	20	+	24	+/−
essential_density	(+)	39	−	6	+	11	+	23	+	38	+	38	+
global_data_complexity	(+)	25	−	27	+	37	+	12	−	24	+	36	+
global_data_density	(+)	38	+	11	−	16	−	27	+	26	+	32	+/−
maintenance_severity	(+)	26	+	7	+	13	+/−	22	−	30	−	15	+/−
modified_condition_count	(+)	7	−	15	+	36	+	4	+	31	+	31	+/−
multiple_condition_count	(+)	3	+	13.5	+	34	+	2	−	28.5	+	29	+
node_count	(+)	9	+	4	+	14	+	34	+	21	+	21	+/−
normalized_cyclomatic_compl.	(+)	31	+	36	+/−	28	+/−	20	−	4	−	13	+/−
parameter_count	(+)	27	+	39	+	8	+	36	+	39	0	37	+/−
percent_comments	(−)	30	+	31	+	7	+	33	+	2	+	2	+

Direction: increasing (+), decreasing (−), non-monotonic (+/−), constant (0)

Table 11 A summary of the interpretation of RLR, SNB, and RF+PDP models generated from MW1 and CM1 datasets, where "Rank" gives the importance rank and "Dir." represents the direction of each independent variable

	Metrics (expected Dir.)		MW1						CM1					
			RLR		SNB		RF + PDP		RLR		SNB		RF + PDP	
			Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.
LOC counts	LOC_total	(+)	27	−	13	+	10	+/−	17	+	11	+	3	+
	LOC_blank	(+)	37	+	17	+	16	+/−	21	+	18	+/−	5	+/−
	LOC_code_and_comment	(+)	5	−	36.5	0	37	−	33	−	21	+/−	7	+/−
	LOC_comments	(+)	13	+	9	+	9	+	12	+	1	+	1	+
	LOC_executable	(+)	29	+	12	+	18	+/−	16	+	10	+	9	+
	Number_of_lines	(+)	36	+	10	+	5	+/−	9	−	5	+	10	+/−
Halstead attributes	content	(+)	35	+	1	+	1	+/−	18	+	2	+	6	+
	difficulty	(+)	23	−	20	+/−	20	+/−	20	−	24	+/−	18	+/−
	effort	(+)	33	+	15.5	+	8	+/−	22	−	15.5	+	19	+
	error_est	(+)	4	+	11	+	12	+	2	+	12	+	27	+/−
	length	(+)	15	−	8	+	13	+/−	24	−	13	+	23	+/−
	level	(+)	26	−	29	+/−	24	+/−	32	+	34	−	26	+
	prog_time	(+)	32	+	15.5	+	19	+/−	23	−	15.5	+	21	+
	volume	(+)	6	−	2	+	11	+/−	3	−	6	+	12	+
	num_operands	(+)	7	−	14	+	21	+	14	+	20	+	17	+/−
	num_operators	(+)	16	+	5	+	7	+	13	−	8	+	20	+
	num_unique_operands	(+)	14	+	4	+	6	+	10	−	4	+	11	+
	num_unique_operators	(+)	28	−	32	+	22	+	11	+	7	+	13	+
McCabe attributes	cyclomatic_complexity	(+)	22	−	27	+	33	+/−	8	+	29	+	30	+
	cyclomatic_density	(+)	10	−	21	−	15	+/−	19	−	19	−	24	−
	design_complexity	(+)	19	+	18	+	26	+/−	31	+	26	+	28	+
	essential_complexity	(+)	17	−	33	+	35	+/−	36	−	37	−	37	+
Miscellaneous	branch_count	(+)	9	−	19	+	31	+/−	7	−	28	+	33	+/−
	call_pairs	(+)	34	+	3	+	4	+	37	−	14	+/−	16	+
	condition_count	(+)	3	+	22	+	28	+/−	5	+	31	+	32	+/−
	decision_count	(+)	8	+	28	+	27	+	6	+	30	+	29	+
	decision_density	(+)	12	−	34	+	30	0	34	+	35	+/−	34	+/−
	design_density	(+)	21	−	31	+/−	29	+/−	30	+	15	+	2	+/−

Table 11 (continued)

Metrics (expected Dir.)		MW1						CM1					
		RLR		SNB		RF + PDP		RLR		SNB		RF + PDP	
		Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.
edge_count	(+)	25	+	7	+/−	3	+/−	29	+	25	+/−	25	+/−
essential_density	(+)	24	+	36.5	0	36	+	26	+	33	−	31	+/−
global_data_complexity	(+)												
global_data_density	(+)												
maintenance_severity	(+)	30	−	30	−	23	+/−	27	−	27	−	22	+/−
modified_condition_count	(+)	2	+	23	+	25	+/−	4	+	36	+	35	+
multiple_condition_count	(+)	1	−	24	+	32	+/−	1	−	32	+	36	+/−
node_count	(+)	18	+	6	+	2	+/−	15	−	22	+/−	14	+/−
normalized_cyclomatic_compl.	(+)	11	+	25	−	17	+/−	28	+	9	−	15	+/−
parameter_count	(+)	31	−	35	−	34	+/−	35	−	23	−	8	−
percent_comments	(−)	20	−	26	+/−	14	+/−	25	+	3	+	4	+/−

Direction: increasing (+), decreasing (−), non-monotonic (+/−), constant (0)

Table 12 A summary of the interpretation of RLR, SNB, and RF+PDP models generated from PC1 and PC2 datasets, where "Rank" gives the importance rank and "Dir." represents the direction of each independent variable

	Metrics (expected Dir.)		PC1						PC2					
			RLR		SNB		RF + PDP		RLR		SNB		RF + PDP	
			Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.
LOC counts	LOC_total	(+)	22	−	8	+	8	+	13	−	13	+	29	+
	LOC_blank	(+)	23	−	3	+	2	+						
	LOC_code_and_comment	(+)	28	+	24	+	6	+	11	−	15	+	10	+
	LOC_comments	(+)	33	−	19	+/−	1	+	10	−	7	+	8	+
	LOC_executable	(+)	21	−	9	+	11	+	21	+	33	−	14	−
	Number_of_lines	(+)	10	+	2	+	5	+	6	+	12	+	17	+/−
Halstead attributes	content	(+)	25	−	1	+	3	+	18	−	2	+	2	+
	difficulty	(+)	8	−	17	+/−	12	−	35	+	11	+	6	+
	effort	(+)	18	−	10.5	+	15	−	34	+	3.5	+	18	+
	error_est	(+)	5	−	15	+	22	+	3	+	14	+	16	+
	length	(+)	32	+	12	+	19	+/−	20	+	6	+	9	+
	level	(+)	37	−	28	−	29	+	31	−	18	−	24	+/−
	prog_time	(+)	19	−	10.5	+	23	−	36	+	3.5	+	12	+/−
	volume	(+)	3	+	7	+	17	+	2	−	5	+	3	+
	num_operands	(+)	13	+	14	+	14	+/−	22	−	9	+	13	+
	num_operators	(+)	20	−	13	+	21	+/−	15	+	8	+	4	+/−
	num_unique_operands	(+)	11	−	5	+	4	+	12	+	10	+	7	+
	num_unique_operators	(+)	16	+	22	+	20	+/−	25	−	22	+	15	−
McCabe attributes	cyclomatic_complexity	(+)	4	+	30	+	28	+/−	5	−	30	+	32	+
	cyclomatic_density	(+)	24	−	6	−	10	−	29	−	16	−	20	−
	design_complexity	(+)	35	−	27	+/−	26	+	16	+	35	+	26	0
	essential_complexity	(+)	26	+	37	−	37	+/−	24	−	34	+	35	+
Miscellaneous	branch_count	(+)	34	−	29	+/−	31	+/−	8	+	28	+	31	0
	call_pairs	(+)	31	−	20	+/−	24	+	27	−	24	+	27	−
	condition_count	(+)	9	−	33	+/−	27	+	7	+	25	+	30	+
	decision_count	(+)	17	−	35	+/−	36	+/−	9	+	29	+	33	+
	decision_density	(+)	30	−	32	+/−	34	+	28	−	32	+	22	+/−
	design_density	(+)	29	+	25	+/−	25	+/−	26	−	19	−	21	−

Table 12 (continued)

Metrics (expected Dir.)		PC1						PC2					
		RLR		SNB		RF + PDP		RLR		SNB		RF + PDP	
		Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.
edge_count	(+)	1	−	18	+/−	13	−	17	−	21	+	23	+/−
essential_density	(+)	12	+	31	+/−	32	+	32	+	36	0	36	+
global_data_complexity	(+)												
global_data_density	(+)												
maintenance_severity	(+)	14	−	23	+/−	18	+/−	33	−	23	−	25	+/−
modified_condition_count	(+)	7	−	36	+	33	+/−	4	+	27	+	34	+
multiple_condition_count	(+)	6	+	34	+/−	30	−	1	−	26	+	28	0
node_count	(+)	2	+	21	+/−	16	+/−	19	+	20	+	11	−
normalized_cyclomatic_compl.	(+)	15	−	4	−	9	−	23	+	17	−	19	−
parameter_count	(+)	27	−	26	−	35	0	30	−	31	+	5	+/−
percent_comments	(−)	36	−	16	+	7	+/−	14	+	1	+	1	+

Direction: increasing (+), decreasing (−), non-monotonic (+/−), constant (0)

Table 13 A summary of the interpretation of RLR, SNB, and RF+PDP models generated from PC3 and PC4 datasets, where "Rank" gives the importance rank and "Dir." represents the direction of each independent variable

	Metrics (expected Dir.)		PC3						PC4					
			RLR		SNB		RF + PDP		RLR		SNB		RF + PDP	
			Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.
LOC counts	LOC_total	(+)	26	+	9	+	20	+	13	+	19	+	7	+
	LOC_blank	(+)	18	+	5	+	1	+	32	+	3	+/−	5	+
	LOC_code_and_comment	(+)	36	−	18	+	9	+	17	+	1	+	1	+
	LOC_comments	(+)	21	+	19	+	5	+	23	−	13	−	4	+/−
	LOC_executable	(+)	25	+	14	+/−	11	+/−	14	+	26	+/−	18	+/−
	Number_of_lines	(+)	12	−	3	+	6	+	12	−	7	+	8	+/−
Halstead attributes	content	(+)	13	−	1	+	2	+	19	−	6	+	6	+
	difficulty	(+)	30	−	16	+/−	7	+/−	35	−	12	+/−	15	+/−
	effort	(+)	24	+	12.5	+	10	+	30	−	9.5	+/−	13	+/−
	error_est	(+)	1	−	10	+	26	+	10	−	27	+/−	26	−
	length	(+)	7	+	7	+	17	+	20	−	17	+/−	20	+/−
	level	(+)	34	−	24	+/−	29	+/−	24	+	24	+	30	+
	prog_time	(+)	27	+	12.5	+	13	+	29	−	9.5	+/−	14	−
	volume	(+)	2	+	4	+	16	+	8	+	8	+/−	9	+
	num_operands	(+)	20	+	6	+	8	+	28	−	20	+/−	12	+/−
	num_operators	(+)	4	+	8	+	15	+	16	−	21	+/−	11	+/−
	num_unique_operands	(+)	5	+	2	+	3	+	15	−	25	+/−	19	+/−
	num_unique_operators	(+)	19	−	23	+	23	+/−	36	+	28	+/−	21	+/−
McCabe attributes	cyclomatic_complexity	(+)	10	+	31	+	35	+/−	5	+	36	−	35	+/−
	cyclomatic_density	(+)	32	+	21	−	12	+/−	11	−	4	−	3	−
	design_complexity	(+)	22	−	29	+	28	+/−	25	−	30	−	33	+/−
	essential_complexity	(+)	23	−	36	−	37	+/−	21	−	32	−	36	+
Miscellaneous	branch_count	(+)	9	−	30	+	34	+/−	9	−	37	−	34	+/−
	call_pairs	(+)	35	−	26	+	21	+/−	34	+	22	+/−	24	+/−
	condition_count	(+)	8	−	32	+	31	+/−	6	+	14	+	16	+
	decision_count	(+)	6	−	37	+	33	+	7	+	18	+	25	+
	decision_density	(+)	31	+	22	+	18	+	31	+	11	+	17	+
	design_density	(+)	29	+	27	+/−	24	+/−	37	+	33	+/−	31	+/−

Table 13 (continued)

Metrics (expected Dir.)		PC3						PC4					
		RLR		SNB		RF + PDP		RLR		SNB		RF + PDP	
		Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.
edge_count	(+)	33	+	17	+	22	+/−	1	−	29	−	28	+/−
essential_density	(+)	16	+	35	+/−	36	+/−	26	+	34	−	37	+/−
global_data_complexity	(+)												
global_data_density	(+)												
maintenance_severity	(+)	15	−	25	−	25	+/−	18	−	35	−	27	+/−
modified_condition_count	(+)	11	−	34	+	32	+	4	+	15	+	29	+
multiple_condition_count	(+)	3	+	33	+	27	+/−	3	−	16	+	22	+
node_count	(+)	17	−	20	+	19	+/−	2	+	31	−	23	+/−
normalized_cyclomatic_compl.	(+)	14	−	15	−	14	+/−	27	+	5	−	10	−
parameter_count	(+)	37	−	28	−	30	+	33	−	23	−	32	+
percent_comments	(−)	28	+	11	+	4	+/−	22	+	2	+	2	+/−

Direction: increasing (+), decreasing (−), non-monotonic (+/−), constant (0)

Table 14 A summary of the interpretation of RLR, SNB, and RF+PDP models generated from PC5 and MC1 datasets, where "Rank" gives the importance rank and "Dir." represents the direction of each independent variable

	Metrics (expected Dir.)		PC5						MC1					
			RLR		SNB		RF + PDP		RLR		SNB		RF + PDP	
			Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.
LOC counts	LOC_total	(+)	9	−	5	+	8	+	16	+	16	+	12	+/−
	LOC_blank	(+)	14	−	21	+	23	+	29	−	20	+	22	+/−
	LOC_code_and_comment	(+)	17	+	27	+	10	+	21	+	10	+	2	+
	LOC_comments	(+)	12	−	31	+	19	+/−	34	−	24	+	11	+/−
	LOC_executable	(+)	8	−	11	+	21	+	17	+	11	+/−	15	+/−
	Number_of_lines	(+)	6	+	12	+	11	+	18	+	7	+	3	+/−
Halstead attributes	content	(+)	27	−	19	+	4	−	22	−	5	+	4	+/−
	difficulty	(+)	35	+	7	+	3	+/−	19	+	9	+/−	13	+/−
	effort	(+)	29	+	1.5	+	1	−	7	−	3.5	+/−	5	+
	error_est	(+)	2	−	8	+	12	+	2	−	21	+/−	26	+/−
	length	(+)	16	−	6	+	16	+	13	−	6	+/−	7	+/−
	level	(+)	20	−	15	−	28	+/−	30	+	19	+/−	25	+/−
	prog_time	(+)	32	+	1.5	+	9	+	6	−	3.5	+/−	9	+
	volume	(+)	1	+	3	+	7	+	1	+	2	+/−	6	+
	num_operands	(+)	13	+	4	+	6	+	15	−	13	+/−	21	+
	num_operators	(+)	11	−	9	+	15	+	11	−	8	+/−	8	+/−
	num_unique_operands	(+)	34	+	14	+	20	+	32	−	15	+	14	+/−
	num_unique_operators	(+)	18	+	17	+	2	+/−	37	+	29	+/−	20	+/−
McCabe attributes	cyclomatic_complexity	(+)	37	+	28	+	34	+	25	+	30	+/−	29	+
	cyclomatic_density	(+)	21	−	30	−	17	+/−	28	−	14	+/−	19	+/−
	design_complexity	(+)	38	−	18	+	26	+	14	−	33	+	32	+
	essential_complexity	(+)	28	−	32	+	35	+	26	−	32	+/−	34	+/−
Miscellaneous	branch_count	(+)	5	+	26	+	29	+	9	−	26	+	31	+/−
	call_pairs	(+)	36	−	16	+	22	+/−	31	+	31	+	10	+
	condition_count	(+)	10	−	22	+	30	+	12	−	23	+/−	27	+/−
	decision_count	(+)	15	−	25	+	33	+	3	−	28	+/−	30	+
	decision_density	(+)												
	design_density	(+)	19	+	37	+/−	27	+/−	24	−	36.5	0	37	−

Table 14 (continued)

Metrics (expected Dir.)		PC5						MC1					
		RLR		SNB		RF + PDP		RLR		SNB		RF + PDP	
		Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.	Rank	Dir.
edge_count	(+)	3	−	13	+	18	+	8	−	12	+/−	18	+
essential_density	(+)	31	+	36	+	37	+/−	33	+	36.5	0	36	+
global_data_complexity	(+)	26	+	20	+	25	+	36	+	27	+/−	33	+
global_data_density	(+)	23	+	34	+	24	+/−	23	−	36.5	0	35	+
maintenance_severity	(+)	24	−	33	−	32	+/−	27	+	36.5	0	38	+
modified_condition_count	(+)	7	−	23	+	36	+	10	+	25	+/−	28	+
multiple_condition_count	(+)	22	−	24	+	31	+	5	+	22	+/−	23	+
node_count	(+)	4	+	10	+	14	+	4	+	17	+/−	16	+
normalized_cyclomatic_compl.	(+)	33	−	35	−	13	+/−	35	+	18	+/−	17	+/−
parameter_count	(+)	25	−	38	−	38	+	38	+	34	+/−	24	+/−
percent_comments	(−)	30	−	29	+/−	5	+/−	20	+	1	+	1	+/−

Direction: increasing (+), decreasing (−), non-monotonic (+/−), constant (0)

Table 15 A summary of the interpretation of RLR, SNB, and RF+PDP models generated from JM1 dataset, where "Rank" gives the importance rank and "Dir." represents the direction of each independent variable

	Metrics (expected Dir.)		JM1					
			RLR		SNB		RF + PDP	
			Rank	Dir.	Rank	Dir.	Rank	Dir.
LOC counts	LOC_total	(+)	1	+	2	+	2	+
	LOC_blank	(+)	15	−	1	+	8	+
	LOC_code_and_comment	(+)	20	+	20	+	21	+
	LOC_comments	(+)	13	−	19	+	14	+
	LOC_executable	(+)	2	−	3	+	4	+
Halstead attributes	content	(+)	17	+	9	+	3	+/−
	difficulty	(+)	10	−	15	+	7	+/−
	effort	(+)	19	+	10	+	6	−
	error_est	(+)	8	−	8	+	17	+
	length	(+)	9	+	7	+	12	+/−
	level	(+)	16	+	18	−	19	+/−
	prog_time	(+)	18	+	11	+	5	−
	volume	(+)	3	−	4	+	1	+
	num_operands	(+)	12	+	12	+	11	+
	num_operators	(+)	6	+	5	+	9	+/−
	num_unique_operands	(+)	11	+	6	+	10	+/−
	num_unique_operators	(+)	7	+	13	+	13	+/−
McCabe attributes	cyclomatic_complexity	(+)	5	−	16	+	18	+
	design_complexity	(+)	21	+	14	+	16	+
	essential_complexity	(+)	14	−	21	+	20	+
Miscellaneous	branch_count	(+)	4	+	17	+	15	+

Direction: increasing (+), decreasing (−), non-monotonic (+/−), constant (0)

The third finding is about the consistency of the expected directions with the observed directions. We assume that the consistency ratio of each predictor between the expected and observed directions indicates its predictability of variable directions. Referring to the summary tables, the consistency ratios of RLR, SNB, and RF+PDP are calculated as 45 percent, 59 percent, and 43 percent, respectively. This implies that SNB could predict the most accurate direction of change among these predictors, because SNB would be less affected by the multicollinearity or non-monotonic effects in independent variables.

Overall, even if any classification technique is used, we believe that the prediction results should be checked from various perspectives with the incorporation of expert knowledge. In this sense, SNB would offer a comprehensible predictive model based on a simple and transparent model structure, which can provide an effective way for balancing the trade-off between accuracy and interpretability.

6 Threats to Validity

As any empirical study, this study has limitations that must be considered when interpreting its results. This section discusses three primary sources of threat to the validity of the results obtained.

6.1 Construct Validity

Construct validity is the degree to which dependent and independent variables are accurately measured by the measurement instruments used in the study.

The data quality of the datasets used in empirical studies may influence threats to construct validity. In our experiments, we employed 13 well-known public datasets that have been used in various other studies. It is assumed that the use of well-known public datasets improves the reliability of the empirical results.

We adopted the AUC as a performance measure. The AUC is one of the most commonly used metrics in software fault prediction studies (Malhotra 2015). It is known to perform well in practice and is often used when a general measure of predictability is desired (Fawcett 2006). Nevertheless, there are many other performance measures such as accuracy, sensitivity, specificity, precision, recall, G-mean, and F-measure, each of which has both strengths and shortcomings. Jiang et al. (2008a) suggest that performance measures should be selected based on the model evaluation criteria specific to each project.

Comparing the interpretability of different classification techniques is a difficult task. In this paper, we adopted a qualitative approach that assessed the interpretability of different defect predictors based on multiple criteria. We believe that the assessment criteria would be reasonable and acceptable, but it would be still subjective. A well-controlled user-based experiment is needed in future studies.

6.2 Internal Validity

Internal validity defines the degree of confidence in a cause-effect relationship between factors of interest and the observed results. In many cases, threats to internal validity are likely related to confounding and selection bias, which lead to the potential limitations of the experiments.

The randomness involved in the experiments (e.g., fold generation in cross-validation and performance evaluation of ensemble models) could possibly introduce some bias. In order to strengthen the robustness and generalization of our results, we performed highly-controlled experiments by referring to Ghotra et al. (2015).

The tuning of the configuration parameters of predictors may also influence our results. Our experiments basically followed the default parameter settings of WEKA's implementation, but a more controlled tuning of the configuration parameters may be required.

6.3 External Validity

External validity is the degree to which the research results can be generalized to the population under study and to other research settings.

Our experiments are based on only 13 real-world public datasets for defect prediction studies in the software engineering domain. Although the datasets vary widely in size and distribution, this could be a potential threat to external validity.

The selection of classification techniques could be another source of threat. To reduce it, our experiments included a wide variety of defect predictors, such as rule-based learners, decision trees, regression models, support vector machines, neural networks, Bayesian learners, and ensemble learners.

7 Conclusion

In this paper, we have proposed a new classification model, called superposed naive Bayes (SNB), which uses a two-step approach to decompose the problem of balancing accuracy and interpretability into two sub-problems: improving the predictive accuracy of a naive Bayes ensemble, followed by obtaining a better linear approximation to gain the interpretability. The first step builds a naive Bayes ensemble via the extended AdaBoost using a stochastic gradient descent method, and the second step transforms it into a simple naive Bayes model by Tayler approximation of the sigmoid calibration of component classifiers in the boosting process.

In order to evaluate the accuracy and interpretability of the proposed method, we conducted a comparative study using well-known classification techniques such as rule-based learners, decision trees, regression models, support vector machines, neural networks, Bayesian learners, and ensemble learners, over 13 real-world public datasets. For the evaluation of predictive accuracy, we adopt a similar approach to Ghotra et al. (2015), which uses multiple repetition of k-fold cross-validation and a double Scott-Knott test. For the evaluation of interpretability, we use a qualitative approach that assessed the interpretability of different types of classification techniques based on Lipton (2016)'s categories, i.e., model transparency (simulatability), component transparency (decomposability), and algorithmic transparency.

The results of our experiments show that SNB can produce a balanced output that satisfies both accuracy and interpretability criteria. A scatter plot comparing reversed fractional ranks (RFRs) of predictive accuracy with those of interpretability would be particularly useful to evaluate and analyze the trade-off between accuracy and interpretability of different classification techniques. A detailed comparison also reveals that SNB offers a comprehensible predictive model based on a simple and transparent model structure, which can provide an effective way for balancing the trade-off between accuracy and interpretability.

There are several issues to be considered for future work. First, how SNB improves the performance of a simple naive Bayes model during iterations is not yet fully understood. Although we assumed that the random subsampling and shrinkage control in constructing a naive Bayes ensemble (NBE) may substantially affect the performance of SNB, further studies are needed to clarify the mechanism. The second issue is the measurement of interpretability. In this paper, a qualitative assessment approach was adopted, but it would be still subjective. A more accurate and objective way to measure the interpretability, such as a highly-controlled user-based experiment, will be required. Finally, the study has only explored a limited range of application areas, i.e., defect prediction in the software engineering domain. However, we believe that the proposed method can be extensively applied to various domains where both predictive accuracy and interpretability are required. Additional studies are needed to further explore the potential of the proposed method.

Acknowledgments The authors would like to thank Prof. William Riley Holden of Japan Advanced Institute of Science and Technology (JAIST) for his helpful comments and advice on the manuscript. The authors also would like to thank the anonymous reviewers who gave us invaluable suggestions.

Appendix 1

The AIC for the contingency table of Table 1 is derived as follows (Sakamoto and Akaike 1978):

$$AIC_1^{(i)} = (-2)\sum_{1\le y\le 2}\sum_{1\le x\le k}\left\{n_{(i,y,x)}\log\left(n_{(i,y,x)}/n_{(i,*,*)}\right)\right\} + 2(2k-1) \tag{11}$$

If mutual independence between Y and X_i is assumed, the AIC changes like the following:

$$AIC_0^{(i)} = (-2)\sum_{1\le y\le 2}\sum_{1\le x\le k}\left\{n_{(i,y,x)}\log\left(n_{(i,y,*)}n_{(i,*,x)}/n_{(i,*,*)}^2\right)\right\} + 2k \tag{12}$$

The difference between the AIC of the *dependence* model ($AIC_1^{(i)}$) and that of the *independence* model ($AIC_0^{(i)}$) is given by

$$\begin{aligned}\Delta AIC^{(i)} &= AIC_1^{(i)} - AIC_0^{(i)} \\ &= (-2)\sum_{1\le y\le 2}\sum_{1\le x\le k}\left\{n_{(i,y,x)}\log\frac{n_{(i,*,*)}n_{(i,y,x)}}{n_{(i,y,*)}n_{(i,*,x)}}\right\} + 2(k-1)\end{aligned} \tag{13}$$

Similarly to the MDL criterion (Fayyad and Irani 1993), the discretization method using AIC starts with a single interval containing all data, and recursively splits intervals. If $\Delta AIC^{(i)}$ is greater than a given (negative) threshold, the *independence* model will be selected as a model with a smaller AIC, so that no more splitting is needed. The total sum of $\Delta AIC^{(i)}$ for all the splittings is $AIC^{(i)}$, the reversal of which indicates the importance of the variable X_i. The variables that have positive AIC can be ignored, because the AIC of a single interval variable is equal to zero, as derived from Eq. 11.

The discretization process using BIC is almost same as that based on AIC. $\Delta BIC^{(i)}$, i.e., the difference between the BIC of a *dependence* model ($BIC_1^{(i)}$) and that of a *independence* model ($BIC_0^{(i)}$), is obtained by simply replacing the last term $2(k-1)$ in Eq. 13 with $\log n(k-1)$, where n is the number of instances.

$$\begin{aligned}\Delta BIC^{(i)} &= BIC_1^{(i)} - BIC_0^{(i)} \\ &= (-2)\sum_{1\le y\le 2}\sum_{1\le x\le k}\left\{n_{(i,y,x)}\log\frac{n_{(i,*,*)}n_{(i,y,x)}}{n_{(i,y,*)}n_{(i,*,x)}}\right\} + \log n(k-1)\end{aligned} \tag{14}$$

The total sum of $\Delta BIC^{(i)}$ for all the splittings is $BIC^{(i)}$, the reversal of which indicates the importance of the variable X_i.

Appendix 2

Abbr.	Parameter Settings in Weka machine learning toolkit
OneR	OneR -B 6
JRip	JRip -F 3 -N 2.0 -O 2 -S 1
J48	J48 -C 0.25 -M 2
NBTree	NBTree

LR	Logistic -R 1.0E-8 -M -1
SVM	SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -M -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"
MLP	MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
NBc	NaiveBayes
NBd	NaiveBayes –D
TAN	BayesNet -D -Q weka.classifiers.bayes.net.search.local.TAN -- -S BAYES -E weka.classifiers.bayes.net.estimate SimpleEstimator -- -A 0.5
AODE	FilteredClassifier -F "weka.filter.supervised.attribute.Discretize -R first-last" -W weka.classifiers.bayes.AODE -- -F 1
HNB	FilteredClassifier -F "weka.filter.supervised.attribute.Discretize -R first-last" -W weka.classifiers.bayes.HNB
AdaBst	AdaBoostM1 -P 100 -S 1 -I 100 -W weka.classifiers.trees.DecisionStump
RF	RandomForest -I 100 -K 0 -S 1

Appendix 3

Dataset	Random Seeds used in Weka				
	1st CV	2nd CV	3rd CV	4th CV	5th CV
MC2	869	280	820	326	961
KC3	309	433	717	708	321
MW1	154	509	564	850	493
CM1	325	629	602	391	467
PC1	111	501	663	151	200
PC2	474	303	537	936	614
PC3	298	566	445	44	381
PC4	711	652	811	516	832
PC5	543	569	505	684	978
MC1	292	604	61	154	746
JM1	172	283	92	966	398
Bugzilla	559	776	599	493	383
Columba	782	155	941	517	882

References

Agterberg FP, Bonham-Carter GF, Wright DF (1990) Statistical pattern integration for mineral exploration. Computer Applications in Resource Estimation Prediction and Assessment for Metals and Petroleum, pp 1–21

Akaike H (1973) Information theory and an extension of the maximum likelihood principle. In: Second international symposium on information theory, pp 267–281

Allahyari H, Lavesson N (2011) User-oriented assessment of classification model understandability. In: 11th Scandinavian conference on artificial intelligence, pp 11–19

Aly M (2005) Survey on multiclass classification methods. Neural Netw 19:1–9

Bauer E, Kohavi R (1999) An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Mach Learn 36(1-2):105–139

Bishop CM (2006) Pattern recognition and machine learning. Springer

Bonham-Carter GF, Agterberg FP, Wright DF (1988) Integration of geological datasets for gold exploration in Nova Scotia. Digital Geologic and Geographic Information Systems, pp 15–23

Bouckaert RR (2004) Naive bayes classifiers that perform well with continuous variables. In: Australasian joint conference on artificial intelligence, pp 1089–1094

Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
Breiman L (2001) Random forests. Mach Learn 45(1):5–32
Briand LC, Basili VR, Thomas WM (1992) A pattern recognition approach for software engineering data analysis. IEEE Trans Softw Eng 18(11):931–942
Burges CJC (1998) A tutorial on support vector machines for pattern recognition. Data Min Knowl Disc 2(2): 121–167
Bury H, Wagner D (2008) Group judgement with ties. Distance-based methods. In: Aschemann H (ed) New approaches in automation and robotics. IntechOpen, London, pp 153–172
Carranza EJM (2004) Weights of evidence modeling of mineral potential: a case study using small number of prospects, Abra, Philippines. Nat Resour Res 13(3):173–187
Cestnik B (1990) Estimating probabilities: a crucial task in machine learning. In: Proceedings of the 9th European conference on artificial intelligence, ECAI '90, pp 147–149
Choetkiertikul M, Dam HK, Tran T, Ghose A (2015) Characterization and prediction of issue-related risks in software projects. In: Proceedings of the 12th working conference on mining software repositories, pp 280–291
Cohen WW (1995) Fast effective rule induction. In: Proceedings of the twelfth international conference on machine learning, ICML'95, pp 115–123
Dahal RK, Hasegawa S, Nonomura A, Yamanaka M, Masuda T, Nishino K (2008) GIS-based weights-of-evidence modelling of rainfall-induced landslides in small catchments for landslide susceptibility mapping. Environ Geol 54(2):311–324
Dejaeger K, Verbeke W, Martens D, Baesens B (2012) Data mining techniques for software effort estimation: a comparative study. IEEE Trans Softw Eng 38(2):375–397
Dejaeger K, Verbraken T, Baesens B (2013) Toward comprehensible software fault prediction models using bayesian network classifiers. IEEE Trans Softw Eng 39(2):237–257
Domingos P, Pazzani M (1997) On the optimality of the simple Bayesian classifier under zero-one loss. Mach Learn 29(2):103–130
Fawcett T (2006) An introduction to ROC analysis. Pattern Recogn Lett 27(8):861–874
Fayyad U, Irani K (1993) Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the 13th international joint conference on artificial intelligence, IJCAI'93, pp 1022–1029
Fayyad U, Piatetsky-Shapiro G, Smyth P (1996) From data mining to knowledge discovery in databases. AI Mag 17(3):37–54
Fenton NE, Neil M (1999) A critique of software defect prediction models. IEEE Trans Softw Eng 25(5): 675–689
Fenton N, Neil M, Marsh W, Hearty P, Radliński Ł, Krause P (2008) On the effectiveness of early life cycle defect prediction with Bayesian nets. Empir Softw Eng 13(5):499–537
Freitas AA (2004) A critical review of multi-objective optimization in data mining: a position paper. ACM SIGKDD Explorations Newsletter 6(2):77–86
Freitas AA (2014) Comprehensible classification models: a position paper. ACM SIGKDD Explorations Newsletter 15(1):1–10
Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139
Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Ann Stat 29(5):1189–1232
Friedman JH (2002) Stochastic gradient boosting. Comput Stat Data Anal 38(4):367–378
Friedman N, Geiger D, Goldszmidt M (1997) Bayesian network classifiers. Mach Learn 29(2-3):131–163
Ghotra B, McIntosh S, Hassan AE (2015) Revisiting the impact of classification techniques on the performance of defect prediction models. In: Proceedings of the 37th international conference on software engineering. ICSE'15, pp 789–800
Goldstein A, Kapelner A, Bleich J, Pitkin E (2015) Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. J Comput Graph Stat 24(1):44–65
Good IJ (1985) Weight of evidence: a brief survey. In: Bernardo JM, DeGroot MH, Lindley DV, Smith AFM (eds) Bayesian Statistics 2: Proceedings of the second valencia international meeting: September 6/10, 1983, New York: North Holland, pp 249–269 (including discussion)
Goodman SN (1999) Toward evidence-based medical statistics. 2: the Bayes factor. Ann Intern Med 130(12): 1005–1013
Guo L, Ma Y, Cukic B, Singh H (2004) Robust prediction of fault-proneness by random forests. In: IEEE 15th international symposium on software reliability engineering, ISSRE, pp 417–428
Hall T, Beecham S, Bowes D, Gray D, Counsell S (2012) A systematic literature review on fault prediction performance in software engineering. IEEE Trans Softw Eng 38(6):1276–1304
Halstead MH (1977) Elements of software science. Elsevier
Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning, 2nd edn. Springer

Heckerman DE, Horvitz EJ, Nathwani BN (1991) Toward normative expert systems: the Pathfinder project. Methods Inf Med 31(2):90–105
Holte RC (1993) Very simple classification rules perform well on most commonly used datasets. Mach Learn 11(1):63–90
Hu Y, Zhang X, Ngai EWT, Cai R, Liu M (2013) Software project risk analysis using Bayesian networks with causality constraints. Decis Support Syst 56:439–449
Huysmans J, Dejaeger K, Mues C, Vanthienen J, Baesens B (2011) An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. Decis Support Syst 51(1):141–154
Jain AK, Mao J, Mohiuddin KM (1996) Artificial neural networks: a tutorial. IEEE Comput 29(3):31–44
Jelihovschi E, Faria JC, Allaman IB (2014) ScottKnott: a package for performing the Scott-Knott clustering algorithm in R. Trends in Applied and Computational Mathematics 15(1):003–017
Jiang Y, Cukic B, Ma Y (2008a) Techniques for evaluating fault prediction models. Empir Softw Eng 13(5):561–595
Jiang Y, Cukic B, Menzies T, Bartlow N (2008b) Comparing design and code metrics for software quality prediction. In: Proceedings of the 4th international workshop on Predictor models in software engineering, pp 11–18
Jiang L, Zhang H, Cai Z (2009) A novel Bayes model: hidden naive Bayes. IEEE Trans Knowl Data Eng 21(10): 1361–1371
John GH, Langley P (1995) Estimating continuous distributions in Bayesian classifiers. In: Proceedings of the 11th conference on uncertainty in artificial intelligence, pp 338–345
Kamei Y, Shihab E (2016) Defect prediction: accomplishments and future challenges. In: 23rd International conference on software analysis, evolution, and reengineering, SANER, vol 5, pp 33–45
Kamei Y, Shihab E, Adams B, Hassan AE, Mockus A, Sinha A, Ubayashi N (2013) A large-scale empirical study of just-in-time quality assurance. IEEE Trans Softw Eng 39(6):757–773
Kim S, Whitehead EJ Jr, Zhang Y (2008) Classifying software changes: clean or buggy? IEEE Trans Softw Eng 34(2):181–196
Kohavi R (1996) Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In: Proceedings of the 2nd international conference on knowledge discovery and data mining. KDD96, pp 202–207
Kononenko I (1993) Inductive and Bayesian learning in medical diagnosis. Appl Artif Intell 7(4):317–337
Kotsiantis SB, Zaharakis I, Pintelas P (2007) Supervised machine learning: a review of classification techniques. Informatica 31:249–268
Kulesza T, Burnett M, Wong WK, Stumpf S (2015) Principles of explanatory debugging to personalize interactive machine learning. In: Proceedings of the 20th International Conference on Intelligent User Interfaces, pp 126–137
Le Cessie S, Van Houwelingen JC (1992) Ridge estimators in logistic regression. Appl Stat 191–201
Lessmann S, Baesens B, Mues C, Pietsch S (2008) Benchmarking classification models for software defect prediction: a proposed framework and novel findings. IEEE Trans Softw Eng 34(4):485–496
Lewis DD (1998) Naive (Bayes) at forty: the independence assumption in information retrieval. In: European conference on machine learning, pp 4–15
Lipton ZC (2016) The mythos of model interpretability. In: 2016 ICML workshop on human interpretability in machine learning. WHI 2016
Madigan D, Mosurski K, Almond RG (1997) Graphical explanation in belief networks. J Comput Graph Stat 6(2):160–181
Malhotra R (2015) A systematic review of machine learning techniques for software fault prediction. Appl Soft Comput 27:504–518
Martens D, Vanthienen J, Verbeke W, Baesens B (2011) Performance of classification models from a user perspective. Decis Support Syst 51(4):782–793
McCabe TJ (1976) A complexity measure. IEEE Trans Softw Eng 2(4):308–320
Menzies T, Greenwald J, Frank A (2007) Data mining static code attributes to learn defect predictors. IEEE Trans Softw Eng 33(1):2–13
Menzies T, Krishna R, Pryor D (2016) The Promise Repository of Empirical Software Engineering Data; http://openscience.us/repo. North Carolina State University, Department of Computer Science bibtex
Mori T (2015) Superposed Naive Bayes for Accurate and Interpretable Prediction. In: Proceedings of the 14th IEEE international conference on machine learning and applications. ICMLA 2015, pp 1228–1233
Mori T, Tamura S, Kakui S (2013) Incremental estimation of project failure risk with Naive Bayes classifier. In: Proceedings of 7th ACM/IEEE international symposium on empirical software engineering and measurement. ESEM 2013, pp 283–286
Platt JC (1999) Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Advances in Large Margin Classifiers, 10(3):61–74
Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann, San Mateo

Ribeiro MT, Singh S, Guestrin C (2016) Why should I trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1135–1144

Ridgeway G, Madigan D, Richardson T, O'Kane J (1998) Interpretable boosted Naive Bayes classification. In: Proceedings of the 4th international conference on knowledge discovery and data mining. KDD98, pp 101–104

Rish I (2001) An empirical study of the naive Bayes classifier. In: IJCAI 2001 workshop on empirical methods in artificial intelligence, 3(22):41–46

Saaty TL (1990) How to make a decision: the analytic hierarchy process. Eur J Oper Res 48(1):9–26

Sakamoto Y, Akaike H (1978) Analysis of cross classified data by AIC. Ann Inst Stat Math 30(1):185–197

Schwarz G (1978) Estimating the dimension of a model. Ann Stat 6(2):461–464

Shepperd M, Song Q, Sun Z, Mair C (2013) Data quality: some comments on the NASA software defect datasets. IEEE Trans Softw Eng 39(9):1208–1215

Spiegelhalter DJ, Knill-Jones RP (1984) Statistical and knowledge-based approaches to clinical decision-support systems, with an application in gastroenterology. Journal of the Royal Statistical Society. Series A (General):35–77

Vandecruys O, Martens D, Baesens B, Mues C, De Backer M, Haesen R (2008) Mining software repositories for comprehensible software fault prediction models. J Syst Softw 81(5):823–839

Webb GI (2000) Multiboosting: a technique for combining boosting and wagging. Mach Learn 40(2):159–196

Webb GI, Boughton JR, Wang Z (2005) Not so naive Bayes: aggregating one-dependence estimators. Mach Learn 58(1):5–24

Wen J, Li S, Lin Z, Hu Y, Huang C (2012) Systematic literature review of machine learning based software development effort estimation models. Inf Softw Technol 54(1):41–59

Witten IH, Frank E, Hall MA, Pal CJ (2011) Data mining: practical machine learning tools and techniques, 3rd edn. Morgan Kaufmann

Yang Y, Webb GI (2009) Discretization for naive-Bayes learning: managing discretization bias and variance. Mach Learn 74(1):39–74

Zadrozny B, Elkan C (2002) Transforming classifier scores into accurate multiclass probability estimates. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp 694–699

Zhang H (2004) The optimality of naive Bayes. In: Proceedings of the 17th Florida artificial intelligence research society conference. FLAIRS2004, pp 562–567

Zimmermann T, Premraj R, Zeller A (2007) Predicting defects for eclipse. In: Proceedings of the third international workshop on predictor models in software engineering, IEEE Computer Society, pp 9

Zimmermann T, Nagappan N, Gall H, Giger E, Murphy B (2009) Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In: Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, pp 91–100

Toshiki Mori is a chief specialist of Corporate Software Engineering & Technology Center at Toshiba Corporation. Before he joined Toshiba in 1992, he received B.E. degree in precision mechanical engineering in 1990 and M.S. degree in information engineering in 1992, both from the University of Tokyo. He was a Visiting Industrial Associate from 1996 to 1999 at the Center for Design Research and the Manufacturing Modeling Laboratory of Stanford University. He is currently a Ph.D. student of School of Knowledge Science, Japan Advanced Institute of Science and Technology. His current research interests include the application of data mining and knowledge science to software engineering and project management.

Naoshi Uchihira is a professor of School of Knowledge Science and a director of Education for Working Professionals-Tokyo Satellite at Japan Advanced Institute of Science and Technology. He received the B.S. and Dr. Eng. degrees in Information Science and Engineering from Tokyo Institute of Technology, and Ph.D. (Knowledge Science) from Japan Advanced Institute of Science and Technology in 1982, 1997 and 2010, respectively. He worked at the corporate R&D center of Toshiba Corporation from 1982 to 2013. His current research interests include project management and IoT innovation design.