用于软件缺陷预测的集成模型

胡梦园[1] 黄鸿云[2] 丁佐华[3]
(浙江理工大学理学院 杭州 310018)[1] (浙江理工大学图书馆多媒体大数据中心 杭州 310018)[2]
(浙江理工大学信息学院 杭州 310018)[3]

摘 要 软件缺陷预测的目的是有效地识别出有缺陷的模块。对于类别平衡数据,传统的分类器具有较好的预测效果,但当数据类别比例分布不均衡时,传统的分类器往往偏向于多数类,易使得少数类模块被误分。但是,真实的软件缺陷预测中的数据往往是类别不平衡的。为了处理软件缺陷中的这种类别不平衡问题,文中提出了基于改进的类权自适应、软投票与阈值移动的集成模型,该模型在不改变原始数据集的情况下,从训练阶段和决策阶段同时考虑处理类别不平衡的问题。首先,在类权值学习阶段,通过类权自适应学习得到不同类的最优权值;然后,在训练阶段,使用前一步得到的最优权值训练 3 个基分类器,并通过软集成的方法组合 3 个基分类器;最后,在决策阶段,根据阈值移动模型来做出决策,以得到最终预测类别。为了证明所提方法的有效性,实验采用 NASA 软件缺陷标准数据集和 Eclipse 软件缺陷标准数据集进行预测,并在相同的数据集上将其与近年提出的几种软件缺陷预测方法在召回率值 *Pd*、假正例率值 *Pf* 和 *F*1 度量值 *F-measure* 方面进行了对比。实验结果表明,所提方法的召回率 *Pd* 平均提高了 0.09,在 *F*1 度量值 *F-measure* 上平均提高了 0.06。因此,文中提出的处理软件缺陷预测中类别不平衡问题的方法的整体性能优于其他软件缺陷预测方法,具有较好的预测效果。

关键词 软件缺陷预测,类权自适应,软投票,集成学习,软集成,阈值移动
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/jsjkx.180901685

Ensemble Model for Software Defect Prediction

HU Meng-yuan[1] HUANG Hong-yun[2] DING Zuo-hua[3]
(School of Science, Zhejiang Sci-Tech University, Hangzhou 310018, China)[1]
(Center of Multimedia Big Data of Library, Zhejiang Sci-Tech University, Hangzhou 310018, China)[2]
(School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China)[3]

Abstract Software defect prediction aims to identify defective modules effectively. Traditional classifiers have good predictive effect on class-balanced data, but when the proportion of data classes is unbalanced, the traditional classifiers incline to majority classes, easily leading to the misclassification of minorityclass module. In reality, the data in software defect prediction are often unbalanced. In order to deal with this kind of class imbalance problem in software defects, this paper proposed an integrated model based on improved class weight self-adaptation, soft voting and threshold moving. This model considers the class imbalance problem in the training stage and decision stage without changing the original data sets. Firstly, in class weight learning stage, the optimal weights of different classes are obtained through class weight adaptive learning. Then, in the training stage, three base classifiers are trained by using the optimal weights obtained in the previous step, and the three base classifiers are combined by soft ensemble method. Finally, in the decision stage, the decision is made according to the threshold moving model to get the final prediction category. In order to prove the validity of the proposed method, the NASA software defect standard data sets and the Eclipse software defect standard data sets are used for prediction, and the proposed method is compared with the results of several software defect prediction methods proposed in recent years on the recall rate *Pd*, false positive rate *Pf* and *F*1 measurement *F-measure*. The experimental results show that the recall rate *Pd* and *F*1 measurement *F-measure* of the proposed method improves by 0.09 and 0.06 on average respectively. Therefore, the overall performance of proposed method for dealing with class imbalance in software defect prediction is superior to other software defect prediction methods, and it has better prediction effect.

到稿日期:2018-09-09 返修日期:2018-12-22 本文受国家自然科学基金项目(61751210,61572441)资助。

胡梦园(1993—),女,硕士生,主要研究方向为软件测试与可靠性建模;**黄鸿云**(1977—),女,硕士生,主要研究方向为软件工程;**丁佐华**(1964—),男,博士,教授,CCF 高级会员,主要研究方向为软件需求建模与分析、软件测试与可靠性评估、在线软件失效预测、程序自动修复、智能软件系统与服务机器人,E-mail:zouhuading@hotmail.com(通信作者)。

Keywords　Software defect prediction，Class weighted self-adaptation，Soft voting，Ensemble learning，Soft ensemble，Threshold-moving

1　引言

软件缺陷预测在软件测试中是一个重要的研究课题，在保证软件质量方面发挥着重要的作用，受到工业界和学术界的广泛关注[1]。软件缺陷预测技术主要是根据软件的基本属性，以及软件模块中的历史数据等信息，来预测新开发的软件模块中是否存在潜在缺陷，具体来讲就是通过统计分析来发现软件缺陷的分布规律，并将其应用在实际的软件缺陷预测中[2]。有效的软件缺陷预测有助于合理地安排测试资源，评估软件质量。

软件模块通常可以分为两类，即有缺陷模块（正例）和无缺陷模块（负例）。为了预测软件中的缺陷，许多经典的数据挖掘算法被提出，如决策树（C4.5）[3]、支持向量机（SVM）[4]和神经网络（Neural Networks）[5]等。然而，真实的软件缺陷预测中的数据往往是类别不平衡的[6-7]，这些传统的方法并不能有效地处理此类数据[8]。鉴于此，研究者提出了重采样（Resampling）[9]、代价敏感学习（Cost-Sensitive）[10]和集成学习（Ensemble Learning）[11]等方法。重采样方法虽然简单有效，但该方法改变了原始数据的真实性。代价敏感学习分类问题中的代价矩阵是人为赋值的，并没有为代价赋予严格的现实意义，因此缺乏事实支撑[12]。集成学习需要通过组合多个基分类器来获得比单一分类器更好的泛化性能，但选择的基分类器要求满足“好而不同”（即基分类器要有一定的“准确性”和“多样性”）的特点，否则集成会起负作用[13]。

为了解决软件缺陷中的类不平衡问题，本文提出了一个新的集成模型，即基于改进的类权自适应、软投票与阈值移动的集成模型。该模型的优点是不改变原始数据集，并从训练阶段和决策阶段同时处理类不平衡问题。首先，在不改变原有数据类别比例的情况下（训练集和测试集上的类别不平衡率相等），对每类样本进行类权自适应学习，从而得到每类样本的权值；然后，使用该权值训练 3 个基分类器，同时计算每个基分类器的置信度（即权值），在预测（测试）阶段，用测试集计算各个基分类器的分类概率；最后，根据软投票的方法对基分类器结果进行集成投票，使用阈值移动方法得出最终的预测类别。

为了说明模型的可行性和有效性，在公开的 NASA 数据集和 Eclipse 数据集上进行实验，并与近年其他研究者提出的预测软件缺陷的分类方法进行比较。实验结果表明，本文提出的基于改进的类权自适应、软投票与阈值移动的集成模型，可以较好地解决软件缺陷预测中的类不平衡问题。

2　本文的模型框架

本节概述了本文的基本方法，图 1 给出了本文模型的基本框架。其中，第一部分是类权值学习阶段（Class Weight Learning Stage），该阶段通过类权自适应方法得到每类样本在当前数据集上的最优权值；第二部分是软集成分类阶段（Soft Ensemble Classification Stage），该阶段独立地训练 3 个类加权分类器，并将结果进行软集成；最后一部分是决策阶段，该阶段使用阈值移动模型得到最终的预测结果。

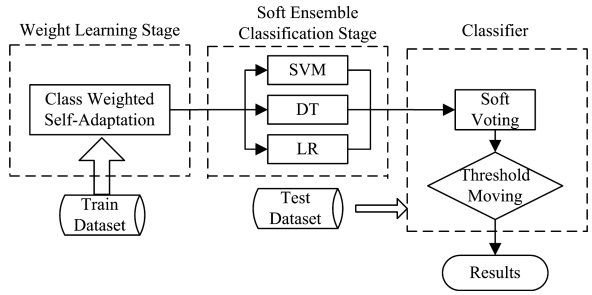

图 1　模型框架

Fig. 1　Model framework

根据本文提出的集成模型在 NASA 的公开数据集和 Eclipse 的公开数据集上计算召回率值、假正例率值、F1 度量值，并将这些值与 SVM[14]，C4.5[3]，ROCUS[15]，CBNN[5]，NSGLP[16]，CDDL[17]方法计算出的相应值进行比较。

3　集成模型

假设我们的数据集为：

$$D = D_{\text{train}} + D_{\text{test}} = \{(x_{1,i}, y_1), \cdots, (x_{m^+,i}, y_{m^+}), (x_{m^++1,i}, y_{m^++1}), \cdots, (x_{m^++m^-,i}, y_{m^++m^-})\} \tag{1}$$

其中，D_{train} 表示训练数据集，D_{test} 表示测试数据集。

$$x_{m^++m^-,i} = (x_{m^++m^-,1}, x_{m^++m^-,2}, \cdots, x_{m^++m^-,n}) \tag{2}$$

其中，$x_{m^++m^-,i}$ 表示每个样本的属性，$y_{m^++m^-}$ 表示的是样本标签，m^+ 表示正例类数目，m^- 表示负例类数目。

下面介绍本文方法的具体实现步骤。

3.1　类权值学习阶段

通过给类赋权值使得分类器在训练过程中对不同的类别赋予不同的关注度。由于各个数据集的类别不平衡率不同、误分类代价不同，因此不能简单地基于数据集类别分布来选择权值[18]。文献[19]提出了一种自适应类权计算方法，受该文献的启发，本文提出了一种基于交叉验证的类权自适应学习方法。

该方法中正例的权值如式(3)所示：

$$W = \arg\max_h [Pd(h) - Pf(h)] \tag{3}$$

使用交叉验证的方法计算出最大的 W 值，并将其与 w_h 中的数值进行匹配，把匹配后的值赋予 W_1，依据文献[19]，把 w_h 中的值设为 1，1.3，1.5，1.7，2，2.5，3，3.5，4，5，7，9，10，20，50，其中 $h=[1,15]$，$Pd(h)$ 表示召回率，$Pf(h)$ 表示假正例率。该方法中令负例权值为 $W_0=1$。

由类权学习阶段可分别获得正例和负例的权值 W_1，W_0。

3.2　软集成分类阶段

分类器集成的使用在机器学习和统计领域已经得到了广泛的认可，各个分类器应尽可能的多样化，众所周知的 bagging 和 boosting 集成技术，通过操纵训练实例来产生多个假设以实现这种多样性[20]。在 3.1 节得到了正例和负例的权

值 W_1,W_0 之后,由于个体分类器应该尽可能多样化,我们选取了 3 个经典的分类器来学习,分别为 SVM、DT 和逻辑回归(LR)。将类加权与 3 个分类器进行集成,得到类加权支持向量机(CW-SVM)、类加权决策树(CW-DT)、类加权逻辑回归(CW-LR)3 个类加权基分类器。

在 D_{train} 上求出每个分类器的置信度 c_i(即权重),其中 $i=1,2,3$,计算方法如式(4)所示:

$$c_i=\frac{\text{the number of correctly predict in i-th classifier}}{\text{total number of training set}} \tag{4}$$

在 D_{test} 中计算每个样本 x_j 在各个分类器上的分类概率。

$$pro_{ij}=(p_{0ij},p_{1ij}) \tag{5}$$

其中,$j=[1,m^++m^-]$,p_{0ij} 表示第 j 个样本在第 i 个分类器上属于负例类的概率,p_{1ij} 表示第 j 个样本在第 i 个分类器上属于正例类的概率,求出 pro_{ij} 后,使用改进的软投票法[21]对基分类器进行组合,通过式(6)求出样本 x_j 最终的分类概率 P_j,公式如下:

$$P_j=\sum_{i=1}^{3}pro_{ij}\cdot c_i \tag{6}$$

其中,$c_i\geqslant 0$,$\sum_{i=1}^{3}c_i=1$。

从软集成分类阶段中可以获得每个样本的最终分类概率 P_j。

3.3 决策阶段(Classifier)

此阶段用 3.2 节计算出的每个样本的最终分类概率 P_j 进行决策。为了更好地调整分类器的决策规则,阈值移动[21]尝试向低成本的模块移动输出,使得具有高成本的模块变得难以被错误分类。在数据集 D 中,观测几率为 m^+/m^-,经典分类器假设训练数据集是真实数据总体的无偏采样,因此观测几率就近似代表真实几率。于是可以得出,只要分类器的预测几率高于观测几率,就把此样本判定为正例,即若:

$$\frac{P_j}{1-P_j}>\frac{m^+}{m^-} \tag{7}$$

则将此样本预测为正例,否则预测为负例。

4 实验数据与评价指标

4.1 数据集规模

为了证明本文所提模型的性能,我们使用 NASA 数据库中随机挑选出的 9 个软件缺陷数据集和 2 个 Eclipse 软件缺陷数据集进行实验,数据集的具体内容如表 1 所列。表 1 列出了数据集名称、属性的数量、有缺陷模块和无缺陷模块的数量以及不平衡率(无缺陷模块数量与有缺陷模块数量的比值)。

表 1 数据集

Table 1 Datasets

数据集	属性	缺陷数量	无缺陷数量	不平衡率
MC2	39	44	81	1.84
KC3	39	36	158	4.39
JM1	21	1672	6110	3.65
MW1	37	27	226	8.37
CM1	37	42	285	6.79
PC1	37	61	644	10.56
PC3	37	134	943	7.04
PC4	37	178	1280	7.19
PC5	38	516	16670	32.31
Eclipse2.0	202	975	5754	5.90
Eclipse3.0	202	1568	9025	5.76

4.2 评价指标

软件缺陷预测是一个二分类问题,在预测的过程中会得到 4 种不同的结果,定义二值分类问题的混淆矩阵(Confusion Matrix)如表 2 所列,其中 TP 表示被正确分类的有缺陷模块数量,FN 表示被错误分类的有缺陷模块数量,FP 表示被错误分类的无缺陷模块数量,TN 表示被正确分类的无缺陷模块数量。

表 2 混淆矩阵

Table 2 Confusion matrix

真实情况	预测结果	
	有缺陷模块	无缺陷模块
有缺陷模块	TP	FN
无缺陷模块	FP	TN

本文的软件缺陷预测的评价指标如下:召回率 Pd、假正例率 Pf、查准率 P 和 $F1$ 度量值 F-measure。

$$Pd=\frac{TP}{TP+FN} \tag{8}$$

$$Pf=\frac{FP}{FP+TN} \tag{9}$$

$$P=\frac{TP}{TP+FP} \tag{10}$$

$$F\text{-measure}=\frac{2\times P\times Pd}{P+Pd} \tag{11}$$

召回率 Pd 表示正确识别出有缺陷模块的概率,即正确预测出的有缺陷模块的数量与真实的有缺陷模块的数量之比。

假正例率 Pf,也称误报率,表示错误识别出有缺陷模块数与实际的无缺陷模块数之比。

查准率 P,也称精度,是预测正确的有缺陷模块数量与所有预测为有缺陷的模块数量之比。

$F1$ 度量值 F-measure 是综合衡量 Pd 值和 Pf 值的调和平均值。由表 2 可知,一个好的预测模型希望达到较高的 Pd 值和较低的 Pf 值,但在实际中这两个目标难以同时实现,提高 Pd 值就意味着需要正确预测出更多的有缺陷模块,然而当更多的软件模块被识别为有缺陷模块时,其中难免会包含一些误分类的无缺陷模块,因此 Pf 值会随之升高。反之,为使 Pf 值降低,势必会以放弃预测出更多的有缺陷模块为代价,因此需要综合衡量 Pd 值和 Pf 值。

上述评价指标的取值范围均在 0 到 1 之间,好的预测模型具有较高的 Pd 值、F-*measure* 值和 P 值,且具有较低的 Pf 值。

5 实验结果与分析

本文实验采用的数据是 NASA 和 Eclipse 公开的数据集,基本信息如表 1 所列,以 Pd 值、Pf 值和 F-measure 值作为评价指标。

本文实验对比了如下 3 种方法:

(1)基于经典机器学习分类器的软件缺陷预测方法,即 SVM[14],C4.5[3]。

(2)基于传统处理类不平衡的软件缺陷预测方法,即 RO-CUS[15],CBNN[5],CDDL[17]。

(3)基于半监督学习的软件缺陷预测方法,即 NS-GLP[16]。

对于每个数据集，所有方法的实验结果都是取 20 次独立运行的平均值。实验结果如表 3 所列。

表 3　实验结果

Table 3　Experimental results

Datasets	Measure	SVM	C4.5	ROCUS	CBNN	NSGLP	CDDL	Our Method
MC2	*Pd*	0.51	0.64	0.72	0.79	0.83	0.83	**0.89**
	Pf	0.24	0.49	0.32	0.54	0.29	0.29	**0.21**
	F-measure	0.52	0.48	0.56	0.56	0.62	0.63	**0.79**
KC3	*Pd*	0.33	0.41	0.55	0.51	0.70	0.71	**0.88**
	Pf	**0.08**	0.16	0.31	0.25	0.35	0.34	0.24
	F-measure	0.38	0.38	0.39	0.38	0.43	0.44	**0.52**
JM1	*Pd*	0.53	0.37	0.67	0.54	0.73	0.68	**0.89**
	Pf	0.45	**0.17**	0.32	0.29	0.36	0.35	0.35
	F-measure	0.29	0.34	0.42	0.38	**0.46**	0.40	**0.46**
MW1	*Pd*	0.21	0.29	0.32	0.61	0.67	0.79	**0.84**
	Pf	**0.04**	0.09	0.15	0.25	0.28	0.25	0.23
	F-measure	0.27	0.27	0.25	0.33	0.36	0.38	**0.50**
CM1	*Pd*	0.15	0.26	0.61	0.59	0.76	0.74	**0.86**
	Pf	**0.04**	0.11	0.32	0.29	0.36	0.37	0.36
	F-measure	0.20	0.25	0.35	0.33	0.38	0.38	**0.50**
PC1	*Pd*	0.66	0.38	0.38	0.54	0.69	0.86	**0.92**
	Pf	0.19	0.15	0.17	0.17	0.31	0.29	**0.12**
	F-measure	0.35	0.32	0.34	0.32	0.41	0.41	**0.49**
PC3	*Pd*	0.64	0.34	0.44	0.65	0.74	0.77	**0.88**
	Pf	0.41	0.08	0.23	0.25	0.31	0.28	0.26
	F-measure	0.28	0.29	0.35	0.38	0.42	0.42	**0.53**
PC4	*Pd*	0.72	0.49	0.61	0.66	0.78	0.89	**0.91**
	Pf	0.16	0.17	0.27	0.18	0.31	0.28	**0.14**
	F-measure	0.47	0.49	0.49	0.46	0.55	0.55	**0.65**
PC5	*Pd*	0.71	0.50	0.59	0.79	0.82	0.84	**0.98**
	Pf	0.22	**0.02**	0.24	0.08	0.30	0.06	0.09
	F-measure	0.16	0.48	0.49	0.37	**0.59**	**0.59**	0.41
Eclipse2.0	*Pd*	0.52	0.46	0.60	0.57	0.77	0.75	**0.80**
	Pf	0.19	0.23	0.26	0.27	0.33	0.24	**0.16**
	F-measure	0.31	0.24	0.44	0.29	0.39	**0.50**	0.49
Eclipse3.0	*Pd*	0.44	0.45	0.62	0.69	0.77	0.78	**0.82**
	Pf	**0.17**	0.23	0.33	0.25	0.29	0.19	0.19
	F-measure	0.38	0.27	0.39	0.49	0.50	0.54	**0.55**

注：粗体数据为每个数据集上各个数据中的最优结果

由表 3 可知，在所有数据集的 3 个评价指标上，本文提出的方法优于其他预测方法，尤其是 *Pd* 值和*F-measure* 值。在 11 个数据集上，本文提出的方法获得了 11 个最优的 *Pd* 值，在数据 PC5 上获得了最优的 *Pd* 值，高达 98%。而文本提出的方法获得了 8 个较好的 *F-measure* 值，另外，1 个 *F-measure* 值在 JM1 数据集上与 NSGLP 中的 *F-measure* 值持平，2 个 *F-measure* 值在 PC5 数据集和 Eclipse2.0 数据集上低于 NSGLP 和 CDDL 中的 *F-measure* 值。但对于 *Pf* 值，本文所提方法的 *Pf* 值比 SVM 和 C4.5 的值略高，具体分析后发现这些方法属于机器学习经典分类方法，并没实际考虑软件缺陷的基本特点，它们的预测性能是有限的。ROCUS 和 CBNN 中的所有实验结果均处于中间水平。

根据表 3 中的结果，可以计算出该方法的平均 *Pd* 值高于其他相关方法，平均有0.09 的提高，*F-measure* 值平均有 0.06 的提高。

综上所述，虽然 SVM 方法和 C4.5 方法的部分 *Pf* 值的效果较优，但是它们的 *Pd* 值和 *F-measure* 值与本文方法的 *Pd* 值和 *F-measure* 值相比较低，*Pd* 值表示正确识别出有缺陷模块的概率，*Pd* 值越高，该方法的预测性能就越好。由此可见，本文方法在软件缺陷预测中的效果更优，更适合类不平衡数据样本的预测。

6　相关工作

Zheng[5] 提出了 3 种代价敏感 boosting 神经网络，第一种基于阈值移动的算法，将阈值移向负例类易发生的模块，从而使得更多的正例类易发生模块被正确分类；另外两种是将代价敏感加入 boosting 过程中，使得算法在样本上增加更多的权重。但代价敏感学习分类问题中的代价矩阵都是人为赋值的，大多数的代价矩阵是未知的，代价并没有赋予严格的现实意义，缺乏事实支撑。代价敏感学习在两类任务中相对容易，而在多类任务上则比较困难。Jing 等[17] 提出了一种代价敏感判别字典学习的方法。该方法根据样本中是否包含缺陷，将软件模块标记为有缺陷模块和无缺陷模块；然后使用复杂性属性作为模块特征，再利用监督字典学习技术，充分挖掘历史数据的类别信息，并在字典学习过程中，对缺陷模块被错误分类为无缺陷的模块增加误分类的惩罚，但是软件缺陷预测技术主要基于大量的历史数据，然而大量的标注是一项耗时的工作，在实际场景中，软件项目中存在许多未标注的软件模块。Zhang 等[16] 提出了一种基于非负稀疏图的标签传播方法，该方法用于软件缺陷预测中的半监督学习，针对缺陷预测中数据的特点，使用少量标记数据和大量未标记数据，并利用 Laplacian 评分抽样和稀疏表示构造类不平衡训练数据集，学

习稀疏标签传播图，最后采用半监督分类方法进行分类预测。但当缺陷训练数据集包含少量样本，并且需要足够多的标记数据时，利用有效的半监督学习算法进行软件缺陷预测将无法得到很好的实现。

本文提出了基于改进的类权自适应、软投票与阈值移动的集成模型，该模型首先在原始数据上进行类权学习，取得了不同类别的最优权值，然后使用上一步获得的最优权值独立训练 3 个加权分类器，同时引入软投票来对训练结果进行集成，通过在测试阶段使用阈值移动模型进行测试，得出最终的预测结果。本文提出的方法同时在训练阶段和决策阶段处理类不平衡问题，在训练阶段不仅考虑了类权自适应学习，而且还运用了 3 个“好而不同”的基分类器进行集成，决策阶段用训练好的分类器将阈值移动模型嵌入到决策过程中，从而获得较好的分类效果。

结束语 本文基于改进的类加权自适应方法和软投票阈值移动，提出了一种用于预测软件缺陷的集成模型。实验数据显示，该集成模型对软件缺陷的预测效果最为明显，Pd 值平均提高 9%，F-measure 值平均提高 6%。由此可见，本文提出的集成模型更适合软件中的类别不平衡样本的缺陷预测。

未来的工作包含 3 个方面：1)本文的研究对象只是两类标签问题，对于多类标签还需要进一步研究；2)本文只是在有限的历史数据上进行预测，未来需要在更多的数据集中验证所提方法的有效性，从而能在实际中真正地用于对软件缺陷的预测；3)本文提出的是一种处理类别不平衡的方法，在以后的工作中可以用于处理其他更多类不平衡问题。

参 考 文 献

[1] BISHNU P S, BHATTACHERJEE V. Software fault prediction using quad tree-based k-means clustering algorithm[J]. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(6): 1146-1150.

[2] HALL T, BEECHAM S, BOWES D, et al. A Systematic Literature Review on Fault Prediction Performance in Software Engineering[J]. IEEE Transactions on Software Engineering, 2012, 38(6): 1276-1304.

[3] WANG J, SHEN B, CHEN Y. Compressed C4.5 Models for Software Defect Prediction [C]// International Conference on Quality Software. Xi An China. IEEE, 2012: 13-16.

[4] XING F, GUO P. Support vector regression for software reliability growth modeling and prediction[C]// International Conference on Advances in Neural Networks. Chongqing China. Springer-Verlag, 2005: 925-930.

[5] ZHENG J. Cost-sensitive boosting neural networks for software defect prediction[J]. Expert Systems with Applications, 2010, 37(6): 4537-4543.

[6] GAO K, KHOSHGOFTAAR T M, NAPOLITANO A. A Hybrid Approach to Coping with High Dimensionality and Class Imbalance for Software Defect Prediction [C]// International Conference on Machine Learning and Applications. Atlanta, GA, USA, IEEE, 2013: 281-288.

[7] WANG S, YAO X. Using Class Imbalance Learning for Software Defect Prediction[J]. IEEE Transactions on Reliability, 2013, 62(2): 434-443.

[8] YU Q, JIANG S J, ZHANG Y M, et al. The Impact Study of Class Imbalance on the Performance of Software Defect Prediction Models[J]. Chinese Journal of Computer, 2018, 41(4): 809-822. (in Chinese)
于巧，姜淑娟，张艳梅，等. 分类不平衡对软件缺陷预测模型性能的影响研究[J]. 计算机学报，2018，41(4)：809-822.

[9] MARUF ÖZTURK M, ZENGIN A. HSDD: A hybrid sampling strategy for class imbalance in defect prediction data sets[C]// Eleventh International Conference on Digital Information Management. Fukuoka, Japan. IEEE, 2017: 60-69.

[10] ZHOU Z H, LIU X Y. Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem[J]. IEEE Transactions on Knowledge & Data Engineering, 2006, 18(1): 63-77.

[11] WANG S, CHEN H, YAO X. Negative correlation learning for classification ensembles[C]// International Joint Conference on Neural Networks. San Jose, California: IEEE, 2011: 1-8.

[12] MIAO L, LIU M, ZHANG D. Cost-sensitive feature selection with application in software defect prediction[C]// 2012 21st International Conference on Pattern Recognition (ICPR). Portland, Oregon: IEEE, 2012: 967-970.

[13] GALA R, FERNANDE Z, BARRENECHE A, et al. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches[J]. IEEE Transactions on Systems Man & Cybernetics Part C Applications & Reviews, 2012, 42(4): 463-484.

[14] ELISH K O, ELISH M O. Predicting defect-prone software modules using support vector machines[J]. Journal of Systems & Software, 2008, 81(5): 649-660.

[15] JIANG Y, LI M, ZHOU Z H. Software Defect Detection with Rocus[J]. Journal of Computer Science & Technology, 2011, 26(2): 328-342.

[16] ZHANG Z W, JING X Y, WANG T J. Label propagation based semi-supervised learning for software defect prediction[J]. Automated Software Engineering, 2016, 24(1): 1-23.

[17] JING X Y, YING S, ZHANG Z W, et al. Dictionary learning based software defect prediction[C]// Proceedings of the 36th International Conference on Software Engineering. ACM, 2014: 414-423.

[18] LU Q, JU C. Research on Credit Card Fraud Detection Model Based on Class Weighted Support Vector Machine[J]. Journal of Convergence Information Technology, 2011, 6(1): 62-68.

[19] MÖHLE S, BRÜNDL M, BEIERLE C. Modeling a System for Decision Support in Snow Avalanche Warning Using Balanced Random Forest and Weighted Random Forest[C]// International Conference on Artificial Intelligence: Methodology, Systems, and Applications. Varna, Bulgaria, Springer/LNAI, 2014: 80-91.

[20] ZHANG Y, ZHANG H, CAI J, et al. A Weighted Voting Classifier Based on Differential Evolution[J]. Abstract and Applied Analysis, 2014, 2014(2): 1-6.

[21] ZHOU Z H. Ensemble Methods: Foundations and Algorithms [M]. London: Taylor & Francis, 2012.