分类不平衡对软件缺陷预测模型性能的影响研究

于 巧[1] 姜淑娟[1),2)] 张艳梅[1),3)] 王兴亚[1] 高鹏飞[1] 钱俊彦[2]

[1)](中国矿业大学计算机科学与技术学院 江苏 徐州 221116)

[2)](桂林电子科技大学广西可信软件重点实验室 广西 桂林 541004)

[3)](南京大学计算机软件新技术国家重点实验室 南京 210023)

摘 要 分类不平衡是指不同类别间样本数量分布不均衡的现象.在软件缺陷预测中,传统预测模型的性能可能会因数据集分类不平衡而受到影响.为了探究分类不平衡对软件缺陷预测模型性能的影响程度,该文提出一种分类不平衡影响分析方法.首先,设计一种新数据集构造算法,将原不平衡数据集转化为一组不平衡率依次递增的新数据集.然后,选取不同的分类模型作为缺陷预测模型,分别对构造的新数据集进行预测,并采用 *AUC* 指标来度量不同预测模型的分类性能.最后,采用变异系数 $C \cdot V$ 来评价各个预测模型在分类不平衡时的性能稳定程度.在8种典型的预测模型上进行实验验证,结果表明 C4.5、RIPPER 和 SMO 这3种预测模型的性能随着不平衡率的增大而下降,而代价敏感学习和集成学习能够有效提高它们在分类不平衡时的性能和性能稳定程度.与上述3种模型相比,逻辑回归、朴素贝叶斯和随机森林等模型的性能更加稳定.

关键词 分类不平衡;软件缺陷预测;预测模型;不平衡率;代价敏感学习;集成学习

中图法分类号 TP311 **DOI号** 10.11897/SP.J.1016.2018.00809

The Impact Study of Class Imbalance on the Performance of Software Defect Prediction Models

YU Qiao[1] JIANG Shu-Juan[1),2)] ZHANG Yan-Mei[1),3)] WANG Xing-Ya[1]

GAO Peng-Fei[1] QIAN Jun-Yan[2]

[1)](*School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, Jiangsu* 221116)

[2)](*Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, Guangxi* 541004)

[3)](*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210023)

Abstract Class imbalance refers to that the number of samples in different classes is unbalanced. In the process of software defect prediction, the performance of traditional prediction models may be affected by the class imbalance problem of datasets. In order to explore the impact of class imbalance on the performance of software defect prediction models, this paper presents an approach to analyzing the impact of class imbalance. First, an algorithm is designed to construct new datasets, which could convert an original imbalanced dataset into a set of new datasets with imbalance ratio increased one by one. Second, different classification models are selected as the defect prediction models to predict on these new constructed datasets respectively. Moreover, *AUC* metric is used to measure the classification performance of different prediction models.

收稿日期:2016-04-14;**在线出版日期**:2016-07-05.**本课题得到国家自然科学基金**(61673384,61502497,61562015)、**广西可信软件重点实验室研究课题**(kx201530)、**南京大学计算机软件新技术国家重点实验室开放课题**(KFKT2014B19)、**江苏省普通高校研究生科研创新计划项目**(KYLX15_1443)、**国家级大学生创新项目**(201510290001)**资助**.于 巧,女,1989年生,博士研究生,中国计算机学会(CCF)会员,主要研究方向为软件分析与测试、机器学习.E-mail:yuqiao@cumt.edu.cn.姜淑娟(通信作者),女,1966年生,博士,教授,博士生导师,中国计算机学会(CCF)会员,主要研究领域为编译技术、软件工程.E-mail:shjjiang@cumt.edu.cn.张艳梅,女,1982年生,博士,讲师,中国计算机学会(CCF)会员,主要研究方向为软件分析与测试.王兴亚,男,1990年生,博士研究生,主要研究方向为软件分析与测试.高鹏飞,男,1995年生,本科生,主要研究方向为软件分析与测试.钱俊彦(通信作者),男,1973年生,博士,教授,中国计算机学会(CCF)会员,主要研究领域为软件工程、模型检验和程序验证.E-mail:qjy2000@gmail.com.

Finally, Coefficient of Variation ($C \cdot V$) is applied to evaluate the performance stability of each prediction model with class imbalance. The empirical study is conducted on eight typical prediction models. The results show that the performance of three prediction models, C4.5, RIPPER and SMO, are decreased with the increasing of imbalance ratio. However, cost-sensitive learning and ensemble learning could improve their performance and performance stability with class imbalance. Compared with the above three models, the performance of Logistic Regression, Naive Bayes and Random Forest models are more stable.

Keywords class imbalance; software defect prediction; prediction models; imbalance ratio; cost-sensitive learning; ensemble learning

1 引 言

近几年,随着机器学习和数据挖掘领域的不断发展,分类不平衡问题也逐步成为目前的研究热点.一般地,分类不平衡是指不同类别间样本数量分布不均衡的现象.例如,在二分类问题中,当两个类别的样本数量相差很大时,就表现出分类不平衡问题.在实际应用中,分类不平衡问题也是普遍存在的,如文本分类[1]、欺诈检测[2]和医学诊断[3]等.然而,在处理分类不平衡问题时,传统的分类模型可能会变得低效[4].

软件缺陷预测是软件工程领域的研究热点之一,也是保证软件质量的重要工作.软件缺陷预测是一个典型的二分类问题,其目的是将软件模块分为有缺陷模块和无缺陷模块,主要对历史缺陷信息进行统计分析,挖掘历史缺陷的分布规律并构建模型,从而对新的软件模块进行预测.

一般地,在缺陷数据集中,有缺陷模块属于少数类,而无缺陷模块属于多数类.在这种情况下,正确识别少数类比正确识别多数类要更加重要,然而,传统的分类模型往往以最大化总体的分类准确率为目标,却忽略了对少数类的分类[5].例如,一个缺陷数据集中包含 100 个样本,其中仅有 1 个有缺陷样本和 99 个无缺陷样本,若某一分类模型将所有样本均预测为无缺陷样本,可达到 99% 的总体分类准确率,但有缺陷样本的分类准确率却为 0,而这样的预测结果并没有任何价值.因此,分类不平衡会对缺陷预测结果产生一定的影响,同时对传统分类模型的有效性也提出了新的挑战.

目前,解决分类不平衡问题的方法有很多,主要分为 3 类[6]:第 1 类是采样法[7-8],包括过采样和欠采样两种,它们分别通过增加少数类样本和减少多数类样本得到分类相对平衡的新数据集.第 2 类是代价敏感学习[9-10],在分类不平衡问题中,正确识别少数类比正确识别多数类更有价值,即错分少数类比错分多数类要付出更大的代价,但传统的分类模型认为所有类别的错分类代价是相同的[11].因此,代价敏感学习通过为不同类别赋予不同的错分类代价来提高少数类的分类性能.第 3 类是集成学习[5,12],通过聚集多个模型的预测结果来提高分类性能.一般地,集成模型的性能要优于单个模型的性能.虽然集成学习并不是为了解决分类不平衡问题而提出的,但其在处理分类不平衡问题时却能够取得较好的效果.

然而,上述 3 种方法在解决分类不平衡问题时,往往需要结合特定的预测模型(即分类模型)或者在某些预测模型下进行验证[5,13],具体该如何选择合理的预测模型?另外,预测模型本身也可能会受到分类不平衡的影响,哪些预测模型的性能更稳定?假如能够掌握不同预测模型在分类不平衡时的性能稳定程度,则可在实际应用中有针对性地选择合理的预测模型,从而更好地指导软件缺陷预测工作.

基于此,本文提出一种分类不平衡影响分析方法,用于评价分类不平衡对软件缺陷预测模型性能的影响程度.该方法通过在原不平衡数据集上构造一组不平衡率依次递增的新数据集,然后选取 8 种典型的分类模型作为缺陷预测模型,分别对构造的新数据集进行预测,并采用 ROC(Receiver Operating Characteristic)曲线下方的面积——*AUC*(Area Under the Curve)[14-15]指标来评价不同预测模型的分类性能,同时采用变异系数 $C \cdot V$(Coefficient of Variation)[16]来评价不同预测模型在分类不平衡时的性能稳定程度.实验结果表明,C4.5[17]、RIP-

PER[18]和 SMO[19]这 3 种预测模型的性能随着不平衡率的增大而下降，说明这 3 种模型的性能极易受到分类不平衡的影响，而代价敏感学习和集成学习能够有效提高它们在分类不平衡时的性能和性能稳定程度.

本文的主要贡献如下：

(1) 提出一种分类不平衡影响分析方法，将原不平衡数据集转化为一组不平衡率依次递增的新数据集，并选取不同的预测模型对新数据集进行预测，从而评价不同预测模型在分类不平衡时的性能稳定程度.

(2) 在 8 种典型的预测模型上进行实验验证，结果表明 C4.5、RIPPER 和 SMO 这 3 种预测模型的性能极易受到分类不平衡的影响，而逻辑回归、朴素贝叶斯和随机森林等模型的性能更加稳定.

(3) 通过本文工作可以掌握不同预测模型在分类不平衡时的性能稳定程度，从而在实际应用中有针对性地选择合理的预测模型，对于软件缺陷预测研究具有一定的指导作用.

2　预测模型介绍

在软件缺陷预测中，研究者往往需要选取一些预测模型进行实验，如决策树、逻辑回归和朴素贝叶斯等.为了使待评价的预测模型更具有代表性，对常用的预测模型进行统计，初步统计结果如表 1 所示，包括预测模型的名称(缩写)、简介以及引用文献.其中，引用文献是指这些文献均采用了相应的预测模型进行实验.下面对这些预测模型作进一步介绍.

表 1　预测模型统计结果

编号	预测模型	简介	引用文献
1	C4.5	一种决策树算法	[5,13,20-24]
2	K-Nearest Neighbor(KNN)	K 近邻算法	[5,24-26]
3	Logistic Regression(LR)	逻辑回归	[21-23,25,27]
4	MultiLayer Perceptron(MLP)	多层感知器	[25-26,28]
5	Naive Bayes(NB)	朴素贝叶斯	[5,13,20-26,28]
6	Random Forest(RF)	随机森林	[5,13,20,23,27-28]
7	RIPPER	命题规则学习	[5,20,24]
8	SMO	SVM 序列最小优化算法	[5,24]

(1) C4.5

C4.5 算法[17]是一种典型的决策树算法，由 Quinlan 教授于 1993 年对 ID3 算法[29]进行改进而提出的，C4.5 算法采用信息增益率进行属性选择，避免了 ID3 算法采用信息增益选择属性时偏向选择取值多的属性这一问题.

(2) K-Nearest Neighbor(KNN)

K 近邻算法[30]是一种基于实例的分类算法，其基本思想是一个样本应与其特征空间中最近邻的 K 个样本中的多数样本属于同一个类别.然而，该方法认为每个近邻样本对分类的影响是相同的，这使得算法对 K 的取值较敏感.为了降低 K 值影响，可以按照与近邻样本的距离加权，从而使得最近邻的样本对分类的贡献最大.

(3) Logistic Regression(LR)

逻辑回归[31]是在线性回归模型的基础上引入一个逻辑回归函数得到的，是一种非线性回归方法，可以用于处理大规模数据.其中，逻辑回归函数如式(1)所示：

$$g(z)=\frac{1}{1+e^{-z}},\ z\in R,\ g(z)\in(0,1) \quad (1)$$

(4) MultiLayer Perceptron(MLP)

多层感知器[32]是一种前馈人工神经网络模型，通常由输入层、隐藏层和输出层三部分组成，它能够将多组输入数据映射为一组适当的输出数据，可用于解决线性不可分问题.

(5) Naive Bayes(NB)

朴素贝叶斯[33]是一种基于贝叶斯定理的概率模型，也是目前使用最广泛的一种分类模型.假设样本 $\boldsymbol{x}=\{f_1,f_2,\cdots,f_d\}$，包含 d 个特征，类别集合 $c=\{c_1,c_2,\cdots,c_l\}$，则样本 $\boldsymbol{x}$ 属于类别 $c_i(i=1,2,\cdots,l)$的概率如式(2)所示：

$$P(c_i|f_1,f_2,\cdots,f_d)=\frac{P(f_1,f_2,\cdots,f_d|c_i)P(c_i)}{P(f_1,f_2,\cdots,f_d)} \quad (2)$$

其中，$P(c_i|f_1,f_2,\cdots,f_d)$中的最大值所属类别即为样本 $\boldsymbol{x}$ 的类别.对不同类别 c_i 而言，式(2)中的分母 $P(f_1,f_2,\cdots,f_d)$为常量，因此只需最大化其分子即可.另外，朴素贝叶斯认为不同特征 $f_1,f_2,\cdots,f_d$ 之间是相互独立的，由此可进一步将式(2)中的分子简化为式(3)：

$$P(f_1,f_2,\cdots,f_d|c_i)P(c_i)=$$

$$P(f_1|c_i)P(f_2|c_i),\cdots,P(f_d|c_i)P(c_i)=P(c_i)\prod_{j=1}^{d}P(f_j|c_i) \quad (3)$$

(6) Random Forest(RF)

随机森林[34]是一种由多棵决策树组成的集成学习模型,对于分类问题,随机森林的输出是由多棵决策树投票得到的,而对于回归问题则是依据多棵决策树的平均值得到的.

(7) RIPPER

RIPPER[18]是一种基于规则的分类算法,由 Cohen 对 IREP 进行优化而提出的.该方法是通过对命题规则进行重复增量地修剪使得产生的错误最少,其能够有效处理噪声数据.

(8) SMO

SMO[19]是一种实现支持向量机(Support Vector Machine,SVM)[35]的序列最小优化(Sequential Minimal Optimization)算法,该算法将二次型求解问题转化为多个优化子问题,并采用启发式搜索策略进行迭代求解,加快了算法的收敛速度.

3 分类不平衡对软件缺陷预测模型性能的影响

3.1 基本概念

在软件缺陷预测中,分类不平衡是指数据集中的无缺陷样本数远远高于有缺陷样本数.假设一个数据集 $\boldsymbol{D}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_n\}$,$\boldsymbol{x}_i\in R^d(i=1,2,\cdots,n)$,即该数据集包括 n 个样本,每个样本含有 d 个特征,另外还包括一个类别特征来标记样本的类别,即有缺陷或无缺陷.根据类别特征可将数据集 $\boldsymbol{D}$ 分为有缺陷类 c_1 和无缺陷类 c_2,样本个数分别为 n_1 和 n_2,且 $n=n_1+n_2$.由此,数据集 $\boldsymbol{D}$ 的不平衡率 IR(Imbalance Ratio)[12]为无缺陷样本数 n_2 与有缺陷样本数 n_1 的比值,即 n_2/n_1,这里对其向下取整,如式(4)所示:

$$IR=\lfloor n_2/n_1\rfloor \quad (4)$$

一般地,$n_2>n_1$,即 $IR>1$. IR 值越大,说明数据集的分类不平衡程度越高,反之则越低.

3.2 分类不平衡影响分析方法

为了探究分类不平衡对软件缺陷预测模型性能的影响程度,本文提出一种分类不平衡影响分析方法,该方法可分为两个阶段:新数据集构造和预测模型评价.具体流程如图 1 所示.

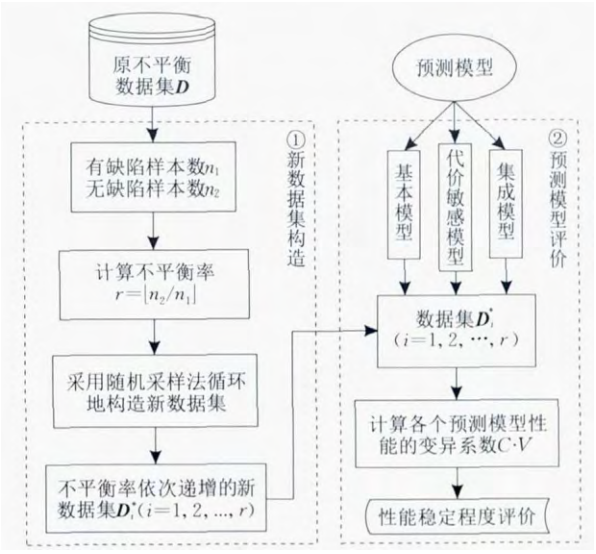

图 1 方法流程

① 新数据集构造.对于一个原不平衡数据集 $\boldsymbol{D}$,首先,统计其有缺陷样本数 n_1 和无缺陷样本数 n_2,计算其不平衡率 $r=\lfloor n_2/n_1\rfloor$;然后,采用随机采样法循环地构造新数据集;最后,将原不平衡数据集 $\boldsymbol{D}$ 转化为一组不平衡率依次递增的新数据集 $\boldsymbol{D}_i^*(i=1,2,\cdots,r)$.

② 预测模型评价.采用表 1 中的 8 种预测模型分别对每个新数据集 $\boldsymbol{D}_i^*(i=1,2,\cdots,r)$ 进行预测,并计算各个预测模型性能的变异系数 $C\cdot V$,变异系数 $C\cdot V$ 值越大,说明预测模型的性能越不稳定,由此来评价各个预测模型在分类不平衡时的性能稳定程度.

特别地,本文将预测模型记为基本模型、代价敏感模型和集成模型.基本模型是指表 1 中给出的预测模型,它们均在 Weka 工具[①]上实现,且考虑到大部分研究者往往采用默认参数进行实验,因此上述模型的运行参数均取 Weka 中的默认值.代价敏感模型是指采用代价敏感学习得到的预测模型,而集成模型则是采用集成学习方法得到的预测模型.

本文主要探究各个基本模型在分类不平衡情况下的性能稳定程度,对于性能不稳定的预测模型,再进一步探究它们的代价敏感模型和集成模型的性能稳定程度.

3.3 新数据集构造

为了探究分类不平衡对预测模型性能的影响,即在分类不平衡情况下各个预测模型性能的变化情况,需要一组具有不同不平衡率的数据集.因此,本文设计一种新数据集构造算法,将原不平衡数据集

① Weka. http://www.cs.waikato.ac.nz/ml/weka/

转化为一组不平衡率依次递增的新数据集，具体过程如算法 1 所示.

算法 1.　新数据集构造算法.

输入：*DataSet*–原不平衡数据集

输出：*newDataSet*–新数据集

1. 根据类别特征将 *DataSet* 分为 *DefectSet* 和 *NonDefectSet*；
2. 有缺陷样本数 $n_1=DefectSet.size()$；
3. 无缺陷样本数 $n_2=NonDefectSet.size()$；
4. 不平衡率 $r=Math.floor(n_2/n_1)$；
5. 数据集 $newDataSet=DefectSet$；
6. 数据集 $restNonDefectSet=NonDefectSet$；
7. WHILE $restNonDefectSet$!=null
8. 　对 $restNonDefectSet$ 进行随机化处理；
9. 　IF $restNonDefectSet.size()>=2n_1$
10. 　　从 $restNonDefectSet$ 中随机选取 n_1 个样本，并保存至 $newDataSet$ 中；
11. 　　同时将选取的样本从 $restNonDefectSet$ 中移除；
12. 　ELSE
13. 　　将 $restNonDefectSet$ 的剩余样本保存至 $newDataSet$；
14. 　　$restNonDefectSet$ =null；
15. 　END IF
16. 　保存新数据集 $newDataSet$；
17. END WHILE
18. RETURN 所有新数据集 $newDataSet$；

Step1. 根据类别特征将原不平衡数据集 *DataSet* 分为有缺陷数据集 *DefectSet* 和无缺陷数据集 *NonDefectSet*，样本数分别为 n_1 和 n_2（行 1～行 3），同时计算数据集 *DataSet* 的不平衡率 r（行 4）.

Step2. 创建两个数据集 *newDataSet* 和 *restNonDefectSet*（行 5～行 6）.

Step3. 对数据集 *restNonDefectSet* 进行随机化处理，从中随机选取 n_1 个样本并保存到数据集 *newDataSet* 中构成新的数据集，同时把这些样本从数据集 *restNonDefectSet* 中移除，从而保证选取的样本不重复（行 8～行 11）.

Step4. 保存构造的新数据集 *newDataSet*（行 16）.

Step5. 循环执行 Step3 和 Step4，直至数据集 *restNonDefectSet* 为空（行 7～行 17）.

Step6. 返回所有新数据集 *newDataSet*（行 18），且它们的不平衡率由 1 到 r 依次递增.

实际上，算法 1 是通过欠采样法来实现的，初始的有缺陷样本数为 n_1，首先从所有的无缺陷样本（n_2 个）中随机选取 n_1 个与有缺陷样本合并，得到新数据集 $\boldsymbol{D}_1^*(IR=1)$；然后再从剩余的无缺陷样本（n_2-n_1 个）中随机选取 n_1 个与 $\boldsymbol{D}_1^*$ 合并，得到新数据集 $\boldsymbol{D}_2^*(IR=2)$；依次循环，直至取完所有的无缺陷样本，最后即可得到一组不平衡率依次递增的新数据集 $\boldsymbol{D}_i^*(i=1,2,\cdots,r)$. 具体计算过程如图 2 所示.

	有缺陷样本数	无缺陷样本数	样本总数	新数据集
初始	n_1	0	n_1	/
第1次	n_1	n_1	$2n_1$	→ $\boldsymbol{D}_1^*(IR=1)$
第2次	n_1	$2n_1$	$3n_1$	→ $\boldsymbol{D}_2^*(IR=2)$
⋮	⋮	⋮	⋮	⋮
第$r-1$次	n_1	$(r-1)n_1$	rn_1	→ $\boldsymbol{D}_{r-1}^*(IR=r-1)$
第r次	n_1	n_2	n_1+n_2	→ $\boldsymbol{D}_r^*(IR=r)$

图 2　不平衡率计算过程

特别地，当剩余的无缺陷样本数小于 n_1 时，我们将这些剩余样本一起合并到 $\boldsymbol{D}_r^*$ 中，所以图 2 中第 r 次循环的无缺陷样本数为 n_2，即此时 $\boldsymbol{D}_r^*$ 与原不平衡数据集 $\boldsymbol{D}$ 所包含的样本总数是相同的.

若利用过采样法来构造新数据集，则是在原不平衡数据集 $\boldsymbol{D}$ 上依次增加有缺陷样本，得到的新数据集的不平衡率分别为 $\lfloor n_2/(n_1+1)\rfloor$，$\lfloor n_2/(n_1+2)\rfloor$，…，但它们的值可能是不连续的，从而影响评价结果的精确性. 因此，本文采用欠采样法来构造新数据集，并假设原不平衡数据集与新数据集具有相同的样本分布.

3.4　预测模型评价

在预测过程中，首先需要选取合理的指标来评价各个预测模型的性能. 本文采用 ROC 曲线下方的面积——*AUC*[14-15] 指标进行评价，这里先介绍最基本的性能评价指标.

软件缺陷预测是一个二分类问题，其预测过程会产生 4 种不同的结果，如表 2 所示，其中有缺陷为正例，无缺陷为负例. 行表示实际类别，列表示预测类别.

表 2　预测结果

		预测类别	
		有缺陷	无缺陷
实际类别	有缺陷	真正例 *TP*	假负例 *FN*
	无缺陷	假正例 *FP*	真负例 *TN*

真正例率 *TPR*（True Positive Rate）是指被正确预测为正例数与实际正例数的比率[36]，即被正确预测为有缺陷样本数与实际有缺陷样本数的比率，如式(5)所示：

$$TPR=\frac{TP}{TP+FN} \tag{5}$$

假正例率 *FPR*（False Positive Rate）是指被错误预测为正例数与实际负例数的比率[36]，即被错误

预测为有缺陷样本数与实际无缺陷样本数的比率，如式(6)所示：

$$FPR=\frac{FP}{FP+TN} \tag{6}$$

基于此，ROC 曲线则是描述分类模型真正例率 TPR 和假正例率 FPR 之间关系的一种图形化方法[36]，如图 3 所示，横坐标表示假正例率，纵坐标表示真正例率.

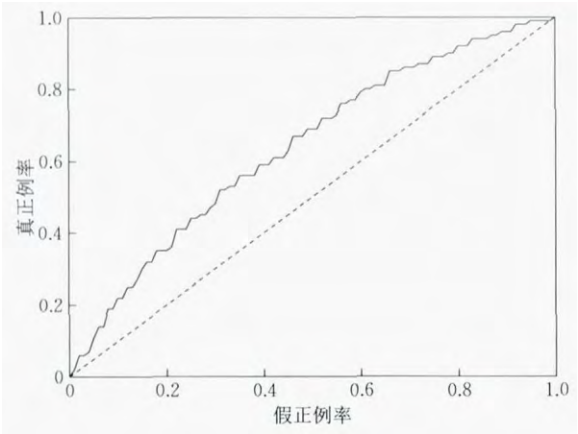

图 3 ROC 曲线

对于一个特定的预测模型和训练数据集，其预测结果对应于 ROC 曲线上的一个点，通过调整该模型的阈值即可得到一条经过(0，0)和(1，1)的曲线，曲线下方的面积即为 AUC 的值[15]. 特别地，AUC 的取值范围为 0～1，当 AUC 为 0.5 时，表示随机猜测模型的性能，如图 3 中的虚线所示. AUC 值越大，说明该模型的性能越好. 因此，好的预测模型应尽可能地靠近坐标系的左上角.

实际上，在软件缺陷预测中，用于评价预测模型性能的指标有很多，如精确率 P、召回率 R 以及 F-Measure 等[37-38]. 与这些指标相比，AUC 指标具有一个很好的特性，即当训练集中不同类别的样本分布变化时，它能够保持相对稳定，所以说在分类不平衡情况下，AUC 指标会更加准确可靠. 同时，Jiang 等人[39]也通过实验证明了 AUC 指标的误差更小且更稳定，在实证研究中被广泛应用于评价预测模型的性能[13,22,24,26-27,40-41]. 因此，本文也采用 AUC 指标来评价预测模型的性能.

接下来，选取某个预测模型对构造的新数据集进行预测，得到一组不同不平衡率下的 AUC 值，记为集合 $S=\{AUC_1,AUC_2,\cdots,AUC_r\}$，然后通过这组 AUC 值的分布来评价预测模型在不同不平衡率下的性能稳定程度.

特别地，在同一数据集上，不同预测模型的平均性能是有差异的，所以不能直接用标准差来衡量各个预测模型性能的离散程度. 因此，本文采用标准差与平均值的比值，即变异系数 $C\cdot V$[16]来评价不同不平衡率下 AUC 值的离散程度，从而消除了平均值差异对数据离散程度比较的影响.

式(7)和式(8)分别表示集合 S 中所有 AUC 值的平均值 μ 和标准差 σ，式(9)则表示变异系数 $C\cdot V$. 具体计算如下所示：

$$\text{平均值}\quad \mu=\frac{\sum_{i=1}^{r}AUC_i}{r} \tag{7}$$

$$\text{标准差}\quad \sigma=\sqrt{\frac{\sum_{i=1}^{r}(AUC_i-\mu)^2}{r}} \tag{8}$$

$$\text{变异系数}\quad C\cdot V=\frac{\sigma}{\mu}\times 100\% \tag{9}$$

变异系数 $C\cdot V$ 值越大，说明 AUC 值的离散程度越大. 由式(9)可知，变异系数 $C\cdot V$ 值不仅受到标准差 σ 的影响，还受到平均值 μ 的影响.

同时，变异系数 $C\cdot V$ 值越大，说明该预测模型的性能越不稳定，即分类不平衡对该预测模型性能的影响程度越大，从而评价不同预测模型在分类不平衡时的性能稳定程度.

4 实证研究

在分类不平衡情况下，运用本文提出的分类不平衡影响分析方法来评价不同预测模型的性能稳定程度. 实验环境：Windows Server 2012，32 GB RAM；Open JDK 1.8，Weka 3.7.

4.1 实验对象

实验选取了 8 个分类不平衡数据集，基本信息如表 3 所示，这些数据集均为 PROMISE 库①中的缺陷数据集. 第 1 列表示数据集名称，第 2 列表示数据集程序的开发语言，包括 C、C++和 Java；第 3 列表示数据集所包含的特征数，即特征维度；第 4 列表示数据集中的样本总数，描述了数据集规模的大小，其中包括小规模(几百)、中规模(上千)和大规模(上万)；第 5～7 列分别表示数据集中的有缺陷样本数、无缺陷样本数和缺陷率；第 8 列表示数据集的不平衡率，其计算如式(4)所示，即无缺陷样本数与有缺陷样本数比值的下取整数，其值越大，说明数据集的分类越不平衡.

① The Promise Repository of Empirical Software Engineering Data. http://openscience.us/repo/

表 3　实验数据集的基本信息

数据集	开发语言	特征数	样本总数	有缺陷样本数	无缺陷样本数	缺陷率/%	不平衡率
MC1	C++	38	1988	46	1942	2.31	42
PC1	C	37	705	61	644	8.65	10
PC2	C	36	745	16	729	2.15	45
PC3	C	37	1077	134	943	12.44	7
PC4	C	37	1458	178	1280	12.21	7
PC5	C++	38	17186	516	16670	3.00	32
Jedit-4.3	Java	20	492	11	481	2.24	43
Tomcat	Java	20	858	77	781	8.97	10

特别地，MC1、PC1、PC2、PC3、PC4 和 PC5 均为美国国家航空航天局（NASA）发布的缺陷数据集，其中的特征均为方法级，包括 McCabe 度量[42]和 Halstead 度量[43]．McCabe 度量是对程序内部结构的复杂度分析，认为程序的复杂度（如控制流）越高，则越容易存在缺陷；Halstead 度量则是基于程序中的操作符和操作数来度量程序的复杂性，并认为操作符和操作数越多则程序越复杂，包含缺陷的可能性也就越高．另外，Jedit-4.3 和 Tomcat 为开源数据集，它们所包含的特征为类级的 CK 度量[44]，该度量综合考虑了面向对象程序中的继承性、耦合性和内聚性等特征，从而更有效地度量软件特征与缺陷的相关性．

在软件缺陷预测中，预测结果是由数据集和预测模型共同决定的．对于某一数据集，其中包括很多软件特征，在未进行特征选择或预测模型本身不具有特征选择能力时，所有特征均被用于训练预测模型；或者先通过特征选择方法选取与类别相关度较高的特征子集，然后再根据选出的特征子集训练预测模型．特别地，在本文实验中，将上述数据集中的所有特征均用于训练预测模型．

4.2　实验设计

为了验证本文方法的有效性，提出了如下 3 个问题：

问题 1．在分类不平衡情况下，哪些预测模型的性能更加稳定，哪些不稳定？

为了探究不同预测模型在分类不平衡情况下的性能稳定性差异，运用本文方法对表 1 中各个预测模型的性能稳定程度进行评价，各个预测模型均在 Weka 工具上实现，且所有模型的运行参数均取 Weka 中的默认值．设计实验见 4.3.1 节．

问题 2．对于性能不稳定的预测模型，代价敏感学习是否能够提高其性能稳定程度？

代价敏感学习是解决分类不平衡的有效途径之一．针对上述实验得出的性能不稳定的预测模型，评价其代价敏感模型在分类不平衡时的性能稳定程度．设计实验见 4.3.2 节．

问题 3．对于性能不稳定的预测模型，集成学习是否能够进一步提高其性能稳定程度？

同样地，集成学习也可以解决分类不平衡问题．针对实验 4.3.1 节得出的性能不稳定的预测模型，进一步评价其集成模型在分类不平衡时的性能稳定程度．设计实验见 4.3.3 节．

本文实验均采用 AUC 指标来评价预测模型的性能，且采用 10 次 10 折交叉验证．同时，为了降低算法 1 中随机采样法对新数据集构造产生的差异，AUC 值取 100 次重复实验的平均值，从而使得实验结果更加准确有效．

4.3　实验结果与分析

4.3.1　不同预测模型的性能稳定程度比较

利用表 1 中的预测模型和表 3 中的数据集进行组合实验．首先，选取一个数据集，通过算法 1 将该数据集转化为一组不平衡率依次递增（即 $IR=1,2,\cdots,r$）的新数据集；然后，采用表 1 中的预测模型对该组新数据集分别进行预测，得到一组不同的不平衡率下的 AUC 值，记为集合 $S=\{AUC_1,AUC_2,\cdots,AUC_r\}$；最后，通过这组 AUC 值的变异系数 $C\cdot V$ 来评价不同预测模型在不同的不平衡率下的性能稳定程度．

表 4 给出了各个预测模型在不同数据集上的实验结果，包括平均值 μ、标准差 σ 和变异系数 $C\cdot V$．变异系数 $C\cdot V$ 值越大，说明该预测模型的性能越不稳定，即说明分类不平衡对预测模型性能的影响程度越大．

为了便于比较，采用“加粗”来标记变异系数 $C\cdot V$ 值大于 5.00% 的数据．由表 4 可得，C4.5、RIPPER 和 SMO 这 3 种预测模型在大部分数据集上的 $C\cdot V$ 值相对较高，说明这 3 种模型的性能极易受到分类不平衡的影响，而 LR、NB 和 RF 等模型的性能保持相对稳定．另外，不同数据集的样本分布差异也

表 4 各个预测模型在不同数据集上的实验结果

预测模型	MC1			PC1			PC2			PC3		
	μ	σ	$C\cdot V/\%$	μ	σ	$C\cdot V/\%$	μ	σ	$C\cdot V/\%$	μ	σ	$C\cdot V/\%$
C4.5	0.6260	0.0387	**6.18**	0.7218	0.0232	3.21	0.5879	0.0663	**11.28**	0.6584	0.0238	3.61
*K*NN	0.6726	0.0110	1.64	0.7026	0.0155	2.21	0.5657	0.0434	**7.67**	0.6503	0.0233	3.58
LR	0.7328	0.0087	1.19	0.8198	0.0214	2.61	0.7127	0.0221	3.10	0.8156	0.0063	0.77
MLP	0.6981	0.0135	1.93	0.8068	0.0140	1.74	0.7152	0.0157	2.20	0.7838	0.0038	0.48
NB	0.6989	0.0058	0.83	0.7652	0.0100	1.31	0.7398	0.0111	1.50	0.7534	0.0047	0.62
RF	0.8963	0.0124	1.38	0.8689	0.0080	0.92	0.7976	0.0078	0.98	0.8256	0.0036	0.44
RIPPER	0.5867	0.0381	**6.49**	0.6341	0.0534	**8.42**	0.5254	0.0524	**9.97**	0.6371	0.0593	**9.31**
SMO	0.5126	0.0391	**7.63**	0.5629	0.0704	**12.51**	0.5084	0.0332	**6.53**	0.5666	0.0914	**16.13**

预测模型	PC4			PC5			Jedit-4.3			Tomcat		
	μ	σ	$C\cdot V/\%$	μ	σ	$C\cdot V/\%$	μ	σ	$C\cdot V/\%$	μ	σ	$C\cdot V/\%$
C4.5	0.7889	0.0220	2.79	0.8191	0.0421	**5.14**	0.5115	0.0707	**13.82**	0.6520	0.0176	2.70
*K*NN	0.7463	0.0135	1.81	0.9367	0.0045	0.48	0.6952	0.0166	2.39	0.6561	0.0144	2.19
LR	0.9108	0.0046	0.51	0.9523	0.0005	0.05	0.6106	0.0268	4.39	0.7883	0.0118	1.50
MLP	0.8914	0.0025	0.28	0.9403	0.0025	0.27	0.5424	0.0728	**13.42**	0.7607	0.0049	0.64
NB	0.8329	0.0033	0.40	0.8419	0.0137	1.63	0.5920	0.0110	1.86	0.7988	0.0034	0.43
RF	0.9391	0.0063	0.67	0.9763	0.0009	0.09	0.7865	0.0080	1.02	0.8223	0.0041	0.50
RIPPER	0.7963	0.0408	**5.12**	0.8329	0.0607	**7.29**	0.5114	0.0470	**9.19**	0.6567	0.0481	**7.32**
SMO	0.6583	0.0837	**12.71**	0.6838	0.1273	**18.62**	0.5103	0.0201	3.94	0.5765	0.0818	**14.19**

会在一定程度上影响预测模型的性能，因此可能会出现同一模型在个别数据集上的性能与在其他数据集上的性能表现不同的情况.例如，*K*NN 模型仅在 PC2 数据集上表现出稍微不稳定（$C\cdot V$ 值为 7.67%），而 MLP 模型仅在 Jedit-4.3 数据集上表现出明显的不稳定情况（$C\cdot V$ 值为 13.42%），原因是这两个数据集中的有缺陷样本数太少，如 PC2 中含有 16 个有缺陷样本，而 Jedit-4.3 中仅含有 11 个有缺陷样本，这使得初始构造的新数据集的规模过小，从而影响了预测模型的性能稳定程度.

为了更清楚地比较不同预测模型的性能稳定程度，采用盒形图来描述，如图 4 所示，横坐标表示各个预测模型，纵坐标表示 AUC 值，虚线表示随机猜测模型的性能，即 AUC 值为 0.5.

在图 4 中，每个盒形代表某一预测模型在一组不平衡率分别为 $1,2,\cdots,r$ 的新数据集下的 AUC 值分布.若一组 AUC 值的变异系数 $C\cdot V$ 值越大，说明这组 AUC 值分布的越离散，则其盒形图也越大，即说明预测模型的性能越不稳定；反之，则说明预测模型的性能相对稳定.然而，盒形图可以描述数据的高低和稳定程度，但却不能描述数据的变化趋势.因此，我们用“阴影”盒形标记某一预测模型的性能随着不平衡率的增大而下降，即代表该模型的性能不稳定，同时该模型在表 4 中的变异系数 $C\cdot V$ 值也相对较高（大于 5.00%）.“无阴影”盒形则表示随着不平衡率的增大，预测模型的性能保持相对稳定，相应地在表 4 中的变异系数 $C\cdot V$ 值也相对较低.

同样地，比较图 4 中的各个子图可以看出，在大部分数据集上，随着不平衡率的增大，C4.5、RIPPER 和 SMO 这 3 种预测模型的性能均明显下降.由此说明，这 3 种模型的性能极易受到分类不平衡的影响，即它们对分类不平衡问题较敏感，且预测模型本身不能有效地处理该问题，因此表现出性能下降的情况.另外还可以看出，随着不平衡率的增大，LR、NB 和 RF 等模型的性能仍然保持相对稳定.

4.3.2 代价敏感模型在分类不平衡时的性能稳定程度

代价敏感学习是机器学习领域中的一种新方法，它认为在分类中不同的错分类结果应该产生不同的代价.因此，代价敏感学习更加注重避免代价较高的错分类，从而使得总的错分类代价最低[45]，这也更加符合于现实应用中的分类问题.例如，在金融交易中，将“欺诈交易”误认为“正常交易”的代价也要高于将“正常交易”误认为“欺诈交易”的代价；在医学诊断中，将“病人”误诊为“健康人”的代价要高于将“健康人”误诊为“病人”的代价[46].同样地，在软件缺陷预测中，将“有缺陷模块”预测为“无缺陷模块”的代价也要远远高于将“无缺陷模块”预测为“有缺陷模块”的代价[11].

在代价敏感学习中，不同的分类代价可以表示为一个二阶矩阵，即为表 2 中 TP、FN、FP 和 TN 这 4 种分类结果赋予不同的代价因子，而不同的代价因子可能会得到不同的预测结果.一般地，FN 的代价要高于 FP 的代价.依照文献[47]，我们可以根据实际情况设置不同的代价因子，这里设置 FN 的代价为 FP 的 10 倍，而 TP 和 TN 的代价均为 0，得到的代价矩阵 $\boldsymbol{CM}$（Cost Matrix）如式（10）所示：

$$\boldsymbol{CM}=\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}=\begin{bmatrix} 0 & 10 \\ 1 & 0 \end{bmatrix} \tag{10}$$

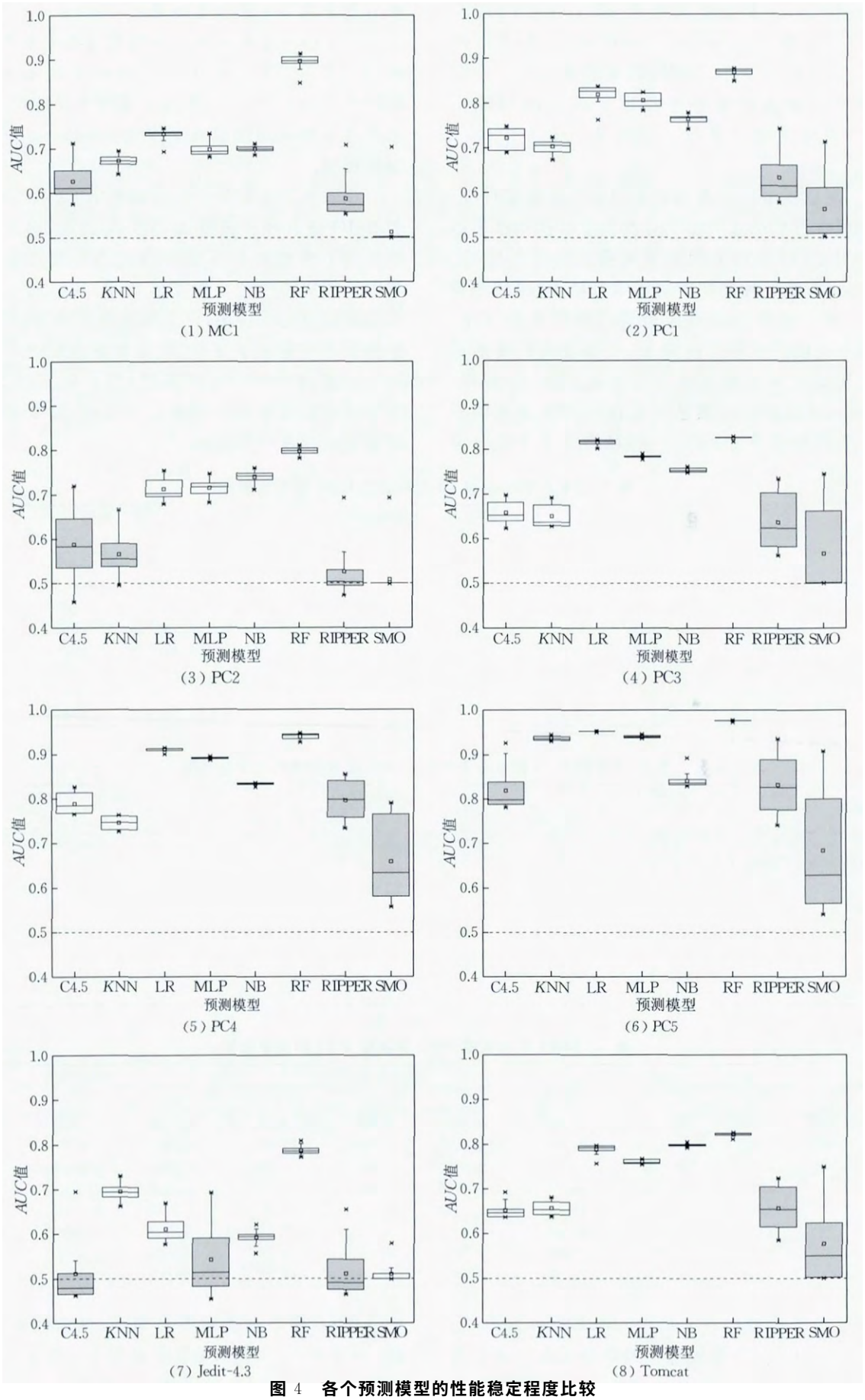

图 4　各个预测模型的性能稳定程度比较

针对 4.3.1 节实验得出的 C4.5、RIPPER 和 SMO 这 3 种性能不稳定的预测模型，分别采用代价敏感学习方法得到 3 种代价敏感模型，并验证这 3 种代价敏感模型在分类不平衡时的性能稳定程度.具体实验结果与分析在 4.3.3 节中统一说明.

4.3.3 集成模型在分类不平衡时的性能稳定程度

集成学习是通过对多个分类模型的预测结果进行某种组合，以达到比单个分类模型更好的性能. AdaBoost[48]是一种典型的集成学习算法，也称为提升算法，每一轮提升都增加错误分类样本的权重，同时减少正确分类样本的权重，从而使得后续的迭代过程更加注重于那些难以分类的样本.一般地，AdaBoost 算法使用投票法或加权法为分类准确率高的分类模型赋予更高的权重，最终将多个弱分类模型集成为一个强分类模型.

为了进一步探究集成模型在分类不平衡时的性能稳定程度，我们采用 AdaBoost 算法实现 C4.5、RIPPER 和 SMO 这 3 种预测模型的集成模型，也就是将这 3 种预测模型分别作为 AdaBoost 算法的基分类模型.

在表 5、表 6 和表 7 中，分别列出了 C4.5、RIPPER 和 SMO 这 3 种预测模型的实验结果，每个表均包括 4.3.1 节的基本模型、4.3.2 节的代价敏感模型和本节的集成模型 3 组实验结果.比较每个表中的实验结果可初步得出，在大部分数据集上，采用代价敏感学习和集成学习后，各个预测模型的平均性能有一定提高；同时，变异系数 $C \cdot V$ 值有所下降，说明采用代价敏感学习和集成学习后，这 3 种模型的性能稳定程度得到提高.

表 5 C4.5、代价敏感 C4.5 和集成 C4.5 的实验结果

数据集	C4.5			代价敏感 C4.5			集成 C4.5		
	μ	σ	$C \cdot V/\%$	μ	σ	$C \cdot V/\%$	μ	σ	$C \cdot V/\%$
MC1	0.6260	0.0387	6.18	0.7052	0.0234	3.32	0.8542	0.0125	1.46
PC1	0.7218	0.0232	3.21	0.7267	0.0230	3.16	0.8295	0.0066	0.80
PC2	0.5879	0.0663	11.28	0.5871	0.0460	7.84	0.7044	0.0329	4.67
PC3	0.6584	0.0238	3.61	0.7069	0.0319	4.51	0.7822	0.0063	0.81
PC4	0.7889	0.0220	2.79	0.8381	0.0192	2.29	0.9205	0.0028	0.30
PC5	0.8191	0.0421	5.14	0.7862	0.0946	12.03	0.9505	0.0077	0.81
Jedit-4.3	0.5115	0.0707	13.82	0.6768	0.0425	6.28	0.7382	0.0210	2.84
Tomcat	0.6520	0.0176	2.70	0.6779	0.0389	5.74	0.7565	0.0061	0.81

表 6 RIPPER、代价敏感 RIPPER 和集成 RIPPER 的实验结果

数据集	RIPPER			代价敏感 RIPPER			集成 RIPPER		
	μ	σ	$C \cdot V/\%$	μ	σ	$C \cdot V/\%$	μ	σ	$C \cdot V/\%$
MC1	0.5867	0.0381	6.49	0.7453	0.0353	4.74	0.8244	0.0118	1.43
PC1	0.6341	0.0534	8.42	0.7652	0.0237	3.10	0.8098	0.0054	0.67
PC2	0.5254	0.0524	9.97	0.6320	0.0713	11.28	0.6777	0.0344	5.08
PC3	0.6371	0.0593	9.31	0.7304	0.0354	4.85	0.7692	0.0060	0.78
PC4	0.7963	0.0408	5.12	0.8477	0.0182	2.15	0.9108	0.0031	0.34
PC5	0.8329	0.0607	7.29	0.9263	0.0115	1.24	0.9551	0.0053	0.55
Jedit-4.3	0.5114	0.0470	9.19	0.7101	0.0509	7.17	0.7676	0.0141	1.84
Tomcat	0.6567	0.0481	7.32	0.7012	0.0584	8.33	0.7496	0.0128	1.71

表 7 SMO、代价敏感 SMO 和集成 SMO 的实验结果

数据集	SMO			代价敏感 SMO			集成 SMO		
	μ	σ	$C \cdot V/\%$	μ	σ	$C \cdot V/\%$	μ	σ	$C \cdot V/\%$
MC1	0.5126	0.0391	7.63	0.6342	0.0587	9.26	0.7487	0.0060	0.80
PC1	0.5629	0.0704	12.51	0.6996	0.0845	12.08	0.8173	0.0109	1.33
PC2	0.5084	0.0332	6.53	0.5673	0.0609	10.74	0.7361	0.0089	1.21
PC3	0.5666	0.0914	16.13	0.6796	0.0980	14.42	0.7844	0.0034	0.43
PC4	0.6583	0.0837	12.71	0.7423	0.0489	6.59	0.8757	0.0036	0.41
PC5	0.6838	0.1273	18.62	0.9003	0.0194	2.15	0.9412	0.0041	0.44
Jedit-4.3	0.5103	0.0201	3.94	0.5604	0.0350	6.25	0.6884	0.0174	2.53
Tomcat	0.5765	0.0818	14.19	0.6755	0.0965	14.29	0.7687	0.0086	1.12

为了更清晰地比较 C4.5、RIPPER 和 SMO 这 3 种基本模型及其代价敏感模型和集成模型在分类不平衡时的性能变化情况，我们采用折线图描述，如图 5 所示，横坐标表示不平衡率，纵坐标表示 AUC 值，同样地，虚线表示随机猜测模型的性能.特别地，由表 3 可知，这 8 个数据集的不平衡率是不同的，分

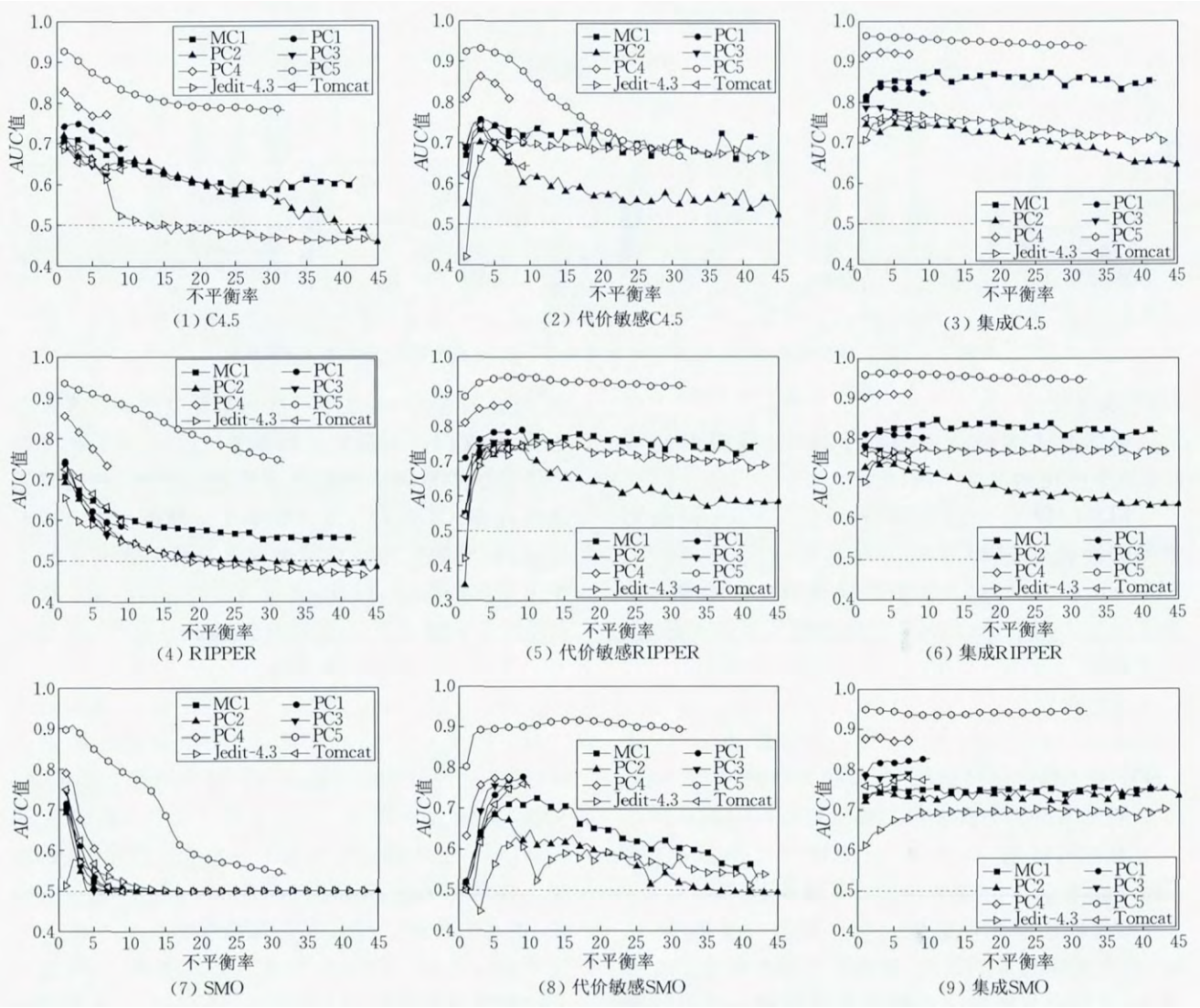

图 5　C4.5、RIPPER、SMO 与其代价敏感模型和集成模型的性能变化比较

别为{42,10,45,7,7,32,43,10}，因此，图 5 中的各条折线表现出长短不一的情况，且折线上是每两个点作一个标记.

首先，比较图 5 中的子图(1)(4)(7)可以看出，随着不平衡率的增大，C4.5、RIPPER 和 SMO 这 3 种模型的性能均明显下降，有的甚至低于随机猜测模型的性能.特别地，从图 5 中的子图(7)可以看出，当不平衡率为 1 时，SMO 模型的性能相对较高；当不平衡率大于 1 时，在大部分数据集上，AUC 值迅速下降并逐渐趋于 0.5，几乎与随机猜测模型性能一致.由此可得，此时 SMO 模型的性能极易受到分类不平衡的影响.

然后，比较图 5 中的子图(1)(2)(3)可以得出，从性能高低和性能稳定程度来看，代价敏感 C4.5 模型略优于 C4.5 模型，而集成 C4.5 模型明显优于 C4.5 模型，而且也优于代价敏感 C4.5 模型.同样地，分别比较图 5 中的子图(4)(5)(6)和子图(7)(8)(9)也可以得出，代价敏感学习和集成学习均能够在一定程度上提高预测模型的性能及性能稳定程度.另外还可以看出，所有预测模型在 PC5 这个规模最大的数据集上的性能保持最高.

为了进一步比较它们的性能稳定程度差异，我们采用柱状图描述，如图 6 所示，横坐标表示不同数据集，纵坐标表示变异系数 $C \cdot V$，其值越大，说明预测模型的性能越不稳定，对应的柱状也越高.从整体来看，在大部分数据集上，代价敏感学习和集成学习能够有效提高预测模型的性能稳定程度；与代价敏感模型相比，集成模型的性能更加稳定，即集成模型处理分类不平衡问题的能力相对更强.原因是，集成模型是通过聚集多个模型的结果得到的，如投票法或加权法等，从而使得其预测结果更加稳定可靠.

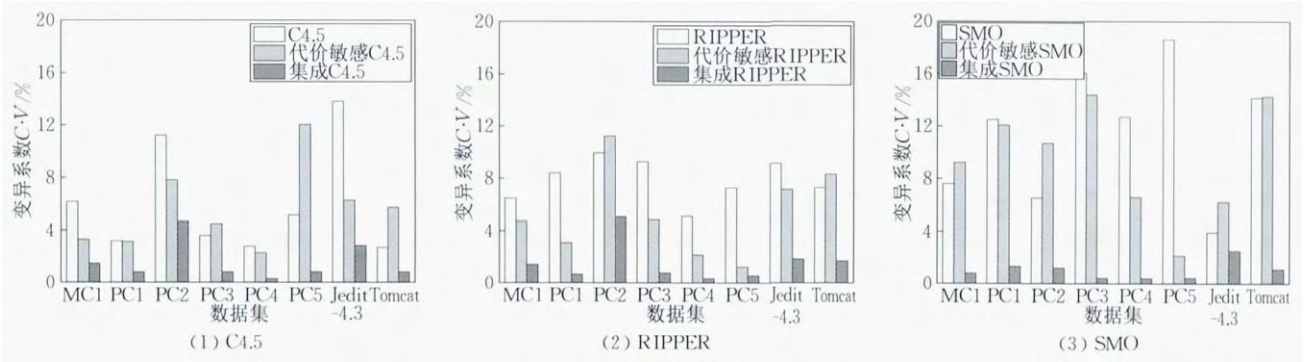

图 6 C4.5、RIPPER、SMO 与其代价敏感模型和集成模型的变异系数 $C \cdot V$ 比较

4.4 实验结论

根据上述实验可得出如下结论，同时可回答 4.2 节中提出的 3 个问题：

回答问题 1. 在分类不平衡情况下，不同预测模型的性能稳定程度确实存在一定差异，C4.5、RIPPER 和 SMO 这 3 种预测模型的性能随着不平衡率的增大而下降，即极易受到分类不平衡的影响. 相对而言，逻辑回归、朴素贝叶斯和随机森林等模型的性能更加稳定.

回答问题 2. 对于 C4.5、RIPPER 和 SMO 这 3 种性能不稳定的预测模型，采用代价敏感学习后，各个模型的性能和性能稳定程度均有一定提高.

回答问题 3. 对于 C4.5、RIPPER 和 SMO 这 3 种性能不稳定的预测模型，采用集成学习后，这 3 种模型的性能和性能稳定程度均有明显提高. 同时，与代价敏感模型相比，集成模型的性能稳定程度更高，可以说集成学习能够更好地处理分类不平衡问题.

本文方法旨在评价不同预测模型在分类不平衡时的性能稳定程度. 根据评价结果，对性能不稳定的预测模型分别采用代价敏感学习和集成学习，验证这两种方法是否能够提高这些模型的性能稳定程度. 通过上述实验，一方面说明了本文方法评价不同预测模型的性能稳定程度的有效性，另一方面也验证了代价敏感学习和集成学习处理分类不平衡问题的有效性.

从整体来看，本文提出的分类不平衡影响分析方法可以有效地评价不同预测模型在分类不平衡时的性能稳定程度. 在评价过程中，对预测模型本身没有任何特殊的要求. 所以说，本文方法对不同预测模型而言是普遍适用的.

通过本文工作可以掌握不同预测模型在分类不平衡情况时的性能稳定程度，从而在实际应用中可以有针对性地选择合理的预测模型. 比如说，在软件缺陷预测中，要探究某一因素对预测模型性能的影响，在这种情况下，应尽可能地避免其他因素带来的干扰. 如文献[13]探究了数据集规模、度量指标和特征选择技术对缺陷预测模型性能的影响，并选取了随机森林、C4.5 和朴素贝叶斯等预测模型进行实验验证，而根据本文实验结果，C4.5 模型容易受到分类不平衡的影响，因此，文献[13]中选用 C4.5 模型可能会影响其实验结果的有效性和准确性，而选用随机森林和朴素贝叶斯是相对合理的.

因此，为了避免分类不平衡的影响，在选取预测模型进行类似上述的实验研究时，应选择朴素贝叶斯、逻辑回归和随机森林等性能相对稳定的模型；然而，在集成学习中，可以选择 C4.5、RIPPER 和 SMO 等性能不稳定的模型作为基分类模型，从而验证集成学习方法的有效性，例如文献[5,20]. 所以说，本文工作对于研究者在预测模型选择上具有一定的指导作用. 另外，在本文工作的基础上，可进一步探究导致 C4.5、RIPPER 和 SMO 这 3 种模型性能不稳定的原因，从而进一步提高它们在处理分类不平衡时的性能及性能稳定程度.

4.5 有效性分析

从以下三个方面来分析影响本文实验有效性的因素：

首先是构建因素，如新数据集构造是影响本文实验有效性的关键因素. 本文方法将原不平衡数据集转化为一组不平衡率依次递增的新数据集，且每个新数据集中均包含所有的有缺陷样本，并采用欠采样法来循环增加无缺陷样本. 然而，这可能会使得构造出的新数据集之间存在一定的随机差异. 因此，为了尽量降低这种差异，本文实验结果均取 100 次重复实验的平均值. 特别地，本文假设原不平衡数据集与新数据集具有相同的样本分布，即不考虑不同数据集间的样本分布差异.

其次是内部因素，预测模型的运行参数可能会对实验结果产生一定的影响. 考虑到大部分研究者往往采用默认参数进行实验，在本文实验中，所有预

测模型的运行参数均取 Weka 中的默认值.然而,对于不同的运行参数对预测模型性能和性能稳定程度的影响,还需进一步验证.

最后是外部因素,如数据集质量可能会影响本文方法对预测模型性能稳定程度的评价.因此,本文在不同规模、不同缺陷率以及不同不平衡率等数据集上进行了较充分的实验,且所有实验数据集都是缺陷预测中的真实数据集,也是最常用的缺陷数据集,从而保证了预测结果的有效性和可靠性.

5　相关工作

近几年,分类不平衡问题在机器学习和数据挖掘领域引起了广泛关注.同样地,在软件缺陷预测中,分类不平衡问题也是普遍存在的,并对传统分类模型的有效性提出了新的挑战.

目前,解决分类不平衡问题的方法主要有 3 种:采样法、代价敏感学习和集成学习[6].例如,针对数据集的分类不平衡问题,Chawla 等人[7]提出一种合成少数类的过采样技术——SMOTE,并证明将 SMOTE 技术与欠采样技术相结合,会取得比单一欠采样技术更好的分类性能.Tahir 等人[8]提出一种逆随机欠采样方法来解决分类不平衡问题,并应用到多分类问题中.然而,采样法在解决分类不平衡问题的同时也可能会产生一些其他问题,如过采样可能会产生过拟合问题,欠采样可能会丢失一些有效的分类信息[49].Zhou 和 Liu[9]结合采样法和阈值移动法来训练代价敏感的神经网络算法,从而有效地解决二分类中的不平衡问题.Siers 和 Islam[10]提出一种代价敏感的决策森林技术,并证明该技术能够有效处理分类不平衡问题.Zheng[50]比较了基于阈值移动和两种权重更新的代价敏感神经网络算法,并证明基于阈值移动的代价敏感算法的性能更优,尤其是针对面向对象程序中的缺陷预测问题的.Sun 等人[5]提出先将不平衡数据集转化为多个平衡的数据集,然后构建多个预测模型,最后采用集成学习方法得到最终的预测结果,且该方法能够很好地处理分类不平衡问题.

在软件缺陷预测中,分类不平衡问题带来的影响也日益突显.为了探究分类不平衡对软件缺陷预测及其相关技术的影响,研究者们也开展了大量的实证性研究.

例如,Japkowicz 和 Stephen[51]通过实验发现,C5.0 模型的性能对分类不平衡问题较敏感,而本文实验得出 C4.5 模型在分类不平衡时的性能极不稳定,实际上 C5.0 模型和 C4.5 模型类似,都属于决策树模型.Galinac Grbac 等人[52]针对不同分类不平衡级别对缺陷预测技术性能稳定性的影响进行了研究,并得出特征选择技术在高不平衡率下性能很不稳定,同时他们还探究了分类不平衡对不同评价指标的影响.针对二分类中的不平衡问题,Galar 等人[12]结合数据处理技术,对不同的集成学习方法 Bagging,Boosting 和 Hybrid 进行比较,实验结果表明,结合了随机欠采样技术的 Bagging 和 Boosting 集成方法的性能更高.Wang 等人[53]探究了影响特征排序技术稳定性的因素,并得出数据集的平衡程度、干扰程度、特征子集大小和排序方法均会影响特征排序技术的稳定性,而 Seiffert 等人[54]通过实验得出,噪声数据和不平衡数据会影响采样技术的性能.

另外,Hall 等人[55]研究了不同建模技术对缺陷预测模型性能的影响,实验结果表明,贝叶斯或逻辑回归等简单建模技术的性能相对更高.Wang 和 Yao[56]比较了重采样、阈值移动的代价敏感学习和集成学习 3 种技术,并证明了集成学习在处理分类不平衡问题时的性能更优.同样地,根据本文方法的评价结果,集成学习的性能稳定程度也高于代价敏感学习.类似地,Rodriguez 等人[57]比较了采样法、代价敏感方法、集成方法以及混合方法在处理分类不平衡时的性能差异,并证明集成方法的性能要优于采样法和代价敏感方法,但集成方法的代价也相对更高.

与上述研究方法不同的是,本文方法针对分类不平衡对缺陷预测模型性能的影响程度进行了探究,并提出一种分类不平衡影响分析方法.通过在原不平衡数据集上构造一组不平衡率依次递增的新数据集,然后选取不同的预测模型对新数据集进行预测和评价.实验结果表明,在软件缺陷预测中,C4.5、RIPPER 和 SMO 这 3 种预测模型的性能随着不平衡率的增大而下降,而逻辑回归、朴素贝叶斯和随机森林等模型的性能比较稳定.同时,与上述相关工作相比,本文方法可以有效评价不同预测模型在处理分类不平衡时的稳定程度,且对于不同预测模型而言,本文方法是普遍适用的.

实际上,除了分类不平衡会影响缺陷预测模型的性能外,数据集规模或噪声数据等也是影响预测模型性能的主要因素.如 Catal 和 Diri[13]针对数据集规模、度量指标和特征选择技术对软件缺陷预测模型性能的影响进行了研究,并选取了随机森林、C4.5 和朴素贝叶斯等多种预测模型进行实验.实验

结果表明,在不同度量指标下,随机森林模型在大规模数据集上的性能最优,而朴素贝叶斯在小规模数据集上的性能最优,当采用特征选择技术后,随机森林模型的性能保持最优.Zhou 等人[28]通过实验发现,类规模对面向对象度量与易错性之间的关系存在潜在的混杂影响,并提出一种线性回归方法来消除这种潜在的影响,从而提高了缺陷预测模型的性能.另外,数据集中的噪声数据也会对分类结果产生影响,如 Kim 等人[58]提出一种检测噪声数据的方法,实验结果表明,该方法能够有效识别噪声数据,有利于提高分类精确率.

综上所述,影响软件缺陷预测性能的因素有很多,如数据集规模、噪声数据和分类不平衡等,不同因素会对预测结果产生不同程度的影响.因此,只有充分探究和挖掘影响分类性能的因素,才能全面高效地提高软件缺陷预测的性能.

6 总结与展望

本文以软件缺陷预测为研究背景,探究了分类不平衡对不同预测模型性能的影响程度,并提出了一种分类不平衡影响分析方法.该方法通过将原不平衡数据集转化为一组不平衡率依次递增的新数据集,然后选取典型的预测模型对新数据集分别进行预测和评价.实验结果表明,随着不平衡率的增大,C4.5、RIPPER 和 SMO 这 3 种预测模型的性能明显下降,说明这 3 种模型的性能极易受到分类不平衡的影响,而代价敏感学习和集成学习能够有效提高它们在分类不平衡时的性能和性能稳定程度.另外,与上述 3 种不稳定的预测模型相比,逻辑回归、朴素贝叶斯和随机森林等模型的性能比较稳定.因此,通过本文方法可以有效评价不同预测模型在分类不平衡时的性能稳定程度,在实际应用中可根据评价结果有针对性地选择合理的预测模型,从而更好地指导软件缺陷预测工作.

特别地,在本文实验中,所有预测模型的参数均取默认值,因此,接下来将探究不同运行参数对预测模型性能的影响,尤其是对上述 3 种性能不稳定的预测模型的影响.另外,本文探究了分类不平衡对软件缺陷预测的影响,即对二分类问题的影响,而在多分类问题中分类不平衡现象也同样存在,因此,分类不平衡对多分类问题的影响还需要进一步探究.

致 谢 感谢各位审稿专家对本文工作提出的宝贵意见和建议!

参 考 文 献

[1] Liu Y, Loh H T, Sun A. Imbalanced text classification: A term weighting approach. Expert Systems with Applications, 2009, 36(1): 690-701

[2] Phua C, Alahakoon D, Lee V. Minority report in fraud detection: Classification of skewed data. ACM SIGKDD Explorations Newsletter, 2004, 6(1): 50-59

[3] Mena L, Gonzalez J A. Symbolic one-class learning from imbalanced datasets: Application in medical diagnosis. International Journal on Artificial Intelligence Tools, 2009, 18(2): 273-309

[4] Peng L, Zhang H, Yang B, et al. A new approach for imbalanced data classification based on data gravitation. Information Sciences, 2014, 288: 347-373

[5] Sun Z, Song Q, Zhu X, et al. A novel ensemble method for classifying imbalanced data. Pattern Recognition, 2015, 48(5): 1623-1637

[6] López V, Fernández A, García S, et al. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. Information Sciences, 2013, 250: 113-141

[7] Chawla N V, Bowyer K W, Hall L O, et al. SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 2002, 16: 321-357

[8] Tahir M A, Kittler J, Yan F. Inverse random under sampling for class imbalance problem and its application to multi-label classification. Pattern Recognition, 2012, 45(10): 3738-3750

[9] Zhou Z H, Liu X Y. Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Transactions on Knowledge and Data Engineering, 2006, 18(1): 63-77

[10] Siers M J, Islam M Z. Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem. Information Systems, 2015, 51: 62-71

[11] Arar Ö F, Ayan K. Software defect prediction using cost-sensitive neural network. Applied Soft Computing, 2015, 33: 263-277

[12] Galar M, Fernández A, Barrenechea E, et al. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. IEEE Transactions on Systems Man and Cybernetics, Part C: Applications and Reviews, 2012, 42(4): 463-484

[13] Catal C, Diri B. Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. Information Sciences, 2009, 179(8): 1040-1058

[14] Bradley A P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition, 1997, 30(7): 1145-1159

[15] Huang J, Ling C X. Using AUC and accuracy in evaluating learning algorithms. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(3): 299-310

[16] Forkman J. Estimator and tests for common coefficients of variation in normal distributions. Communications in Statistics-Theory and Methods, 2009, 38(2): 233-251

[17] Quinlan J R. C4. 5: Programs for Machine Learning. Burlington, Massachusetts: Morgan Kaufmann, 1993

[18] Cohen W W. Fast effective rule induction//Proceedings of the 12th International Conference on Machine Learning. Tahoe City, USA, 1995: 115-123

[19] Platt J C. Fast training of support vector machines using sequential minimal optimization//Schölkopf B, Burges C, Smola A eds. Advances in Kernel Methods-Support Vector Learning. Cambridge, MA: MIT Press, 1999: 185-208

[20] Sun Z, Song Q, Zhu X. Using coding-based ensemble learning to improve software defect prediction. IEEE Transactions on Systems Man and Cybernetics, Part C: Applications and Reviews, 2012, 42(6): 1806-1817

[21] He Z, Shu F, Yang Y, et al. An investigation on the feasibility of cross-project defect prediction. Automated Software Engineering, 2012, 19(2): 167-199

[22] He P, Li B, Liu X, et al. An empirical study on software defect prediction with a simplified metric set. Information and Software Technology, 2015, 59: 170-190

[23] Nam J, Kim S. CLAMI: Defect prediction on unlabeled datasets//Proceedings of the 30th International Conference on Automated Software Engineering. Lincoln, USA, 2015: 452-463

[24] Ghotra B, McIntosh S, Hassan A E. Revisiting the impact of classification techniques on the performance of defect prediction models//Proceedings of the 37th International Conference on Software Engineering. Florence, Italy, 2015, 1: 789-800

[25] Gao K, Khoshgoftaar T M, Wang H, et al. Choosing software metrics for defect prediction: An investigation on feature selection techniques. Software: Practice and Experience, 2011, 41(5): 579-606

[26] Khoshgoftaar T M, Gao K, Napolitano A, et al. A comparative study of iterative and non-iterative feature selection techniques for software defect prediction. Information Systems Frontiers, 2014, 16(5): 801-822

[27] Laradji I H, Alshayeb M, Ghouti L. Software defect prediction using ensemble learning on selected features. Information and Software Technology, 2015, 58: 388-402

[28] Zhou Y, Xu B, Leung H, et al. An in-depth study of the potentially confounding effect of class size in fault prediction. ACM Transactions on Software Engineering and Methodology, 2014, 23(1): 10:1-10:51

[29] Quinlan J R. Induction of decision trees. Machine Learning, 1986, 1(1): 81-106

[30] Aha D W, Kibler D, Albert M K. Instance-based learning algorithms. Machine Learning, 1991, 6(1): 37-66

[31] Le Cessie S, Van Houwelingen J C. Ridge estimators in logistic regression. Applied Statistics, 1992, 41(1): 191-201

[32] Attali J G, Pagès G. Approximations of functions by a multilayer perceptron: A new approach. Neural Networks, 1997, 10(6): 1069-1081

[33] John G H, Langley P. Estimating continuous distributions in Bayesian classifiers//Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence. Montreal, Canada, 1995: 338-345

[34] Breiman L. Random forests. Machine Learning, 2001, 45(1): 5-32

[35] Cortes C, Vapnik V. Support-vector networks. Machine Learning, 1995, 20(3): 273-297

[36] Fawcett T. An introduction to ROC analysis. Pattern Recognition Letters, 2006, 27(8): 861-874

[37] Malhotra R. A systematic review of machine learning techniques for software fault prediction. Applied Soft Computing, 2015, 27: 504-518

[38] Li M, Zhang H, Wu R, et al. Sample-based software defect prediction with active and semi-supervised learning. Automated Software Engineering, 2012, 19(2): 201-230

[39] Jiang Y, Lin J, Cukic B, et al. Variance analysis in software fault prediction models//Proceedings of the 20th International Symposium on Software Reliability Engineering. Mysuru, India, 2009: 99-108

[40] Erturk E, Sezer E A. A comparison of some soft computing methods for software fault prediction. Expert Systems with Applications, 2015, 42(4): 1872-1879

[41] Zhang F, Zheng Q, Zou Y, et al. Cross-project defect prediction using a connectivity-based unsupervised classifier //Proceedings of the 38th International Conference on Software Engineering. Austin, USA, 2016: 309-320

[42] McCabe T J. A complexity measure. IEEE Transactions on Software Engineering, 1976, SE-2(4): 308-320

[43] Halstead M H. Elements of Software Science. New York: Elsevier, 1977

[44] Chidamber S R, Kemerer C F. A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 1994, 20(6): 476-493

[45] Yan Ming-Song, Zhou Zhi-Hua. An empirical comparative study of cost-sensitive classification algorithms. Pattern Recognition and Artificial Intelligence, 2005, 18(5): 628-635(in Chinese)
(闫明松，周志华. 代价敏感分类算法的实验比较. 模式识别与人工智能, 2005, 18(5): 628-635)

[46] Qiao X, Liu Y. Adaptive weighted learning for unbalanced multicategory classification. Biometrics, 2009, 65(1): 159-168

[47] Jiang Y, Cukic B, Menzies T. Cost curve evaluation of fault prediction models//Proceedings of the 19th International Symposium on Software Reliability Engineering. Seattle, USA, 2008: 197-206

[48] Freund Y, Schapire R E. Experiments with a new boosting algorithm//Proceedings of the 13th International Conference on Machine Learning. Bari, Italy, 1996: 148-156

[49] Li Xiong-Fei, Li Jun, Dong Yuan-Fang, Qu Cheng-Wei. A new learning algorithm for imbalanced data-PCBoost. Chinese Journal of Computers, 2012, 35(2): 202-209(in Chinese)
(李雄飞, 李军, 董元方, 屈成伟. 一种新的不平衡数据学习算法 PCBoost. 计算机学报, 2012, 35(2): 202-209)

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

[51] Japkowicz N, Stephen S. The class imbalance problem: A systematic study. Intelligent Data Analysis, 2002, 6(5): 429-449

[52] Galinac Grbac T, Mauša G, Dalbelo Bašić B. Stability of software defect prediction in relation to levels of data imbalance //Proceedings of the 2nd Workshop on Software Quality Analysis, Monitoring, Improvement and Applications. Novi Sad, Serbia, 2013: 1-10

[53] Wang H, Khoshgoftaar T M, Napolitano A. An empirical study on the stability of feature selection for imbalanced software engineering data//Proceedings of the 11th International Conference on Machine Learning and Applications. Boca Raton, USA, 2012, 1: 317-323

[54] Seiffert C, Khoshgoftaar T M, Van Hulse J, et al. An empirical study of the classification performance of learners on imbalanced and noisy software quality data. Information Sciences, 2014, 259: 571-595

[55] Hall T, Beecham S, Bowes D, et al. A systematic literature review on fault prediction performance in software engineering. IEEE Transactions on Software Engineering, 2012, 38(6): 1276-1304

[56] Wang S, Yao X. Using class imbalance learning for software defect prediction. IEEE Transactions on Reliability, 2013, 62(2): 434-443

[57] Rodriguez D, Herraiz I, Harrison R, et al. Preliminary comparison of techniques for dealing with imbalance in software defect prediction//Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. London, United Kingdom, 2014, 43:1-43:10

[58] Kim S, Zhang H, Wu R, et al. Dealing with noise in defect prediction//Proceedings of the 33rd International Conference on Software Engineering. Honolulu, USA, 2011: 481-490

YU Qiao, born in 1989, Ph.D. candidate. Her research interests include software analysis and testing, machine learning.

JIANG Shu-Juan, born in 1966, Ph. D., professor, Ph.D., supervisor. Her research interests include compilation techniques and software engineering.

ZHANG Yan-Mei, born in 1982, Ph.D., lecturer. Her research interests include software analysis and testing.

WANG Xing-Ya, born in 1990, Ph.D. candidate. His research interests include software analysis and testing.

GAO Peng-Fei, born in 1995, undergraduate. His research interests include software analysis and testing.

QIAN Jun-Yan, born in 1973, Ph.D., professor. His research interests include software engineering, model checking and program verification.

Background

In recent years, class imbalance has gradually become a crucial problem in machine learning and data mining. Generally, class imbalance refers to that the number of samples in different classes is unbalanced. In practice, class imbalance is widespread in real-world applications, such as text classification, fraud detection and medical diagnosis.

Software defect prediction is the research focus of software engineering. In reality, there are a lot of traditional prediction models or classification models applied in software defect prediction, such as Logistics Regression, Naive Bayes and Random Forest. However, the performance of these defect prediction models may be affected by the factor of class imbalance. At present, many solutions have been presented to solve this problem, including sampling, cost-sensitive learning and ensemble learning. Unfortunately, these solutions did not fully take into account the impact of class imbalance on prediction models themselves.

In order to explore the impact of class imbalance on the performance of software defect prediction models, this paper presents an approach to analyzing the impact of class imbalance on typical prediction models. The experimental results show that the proposed approach could effectively evaluate the performance stability of different prediction models with class imbalance. Based on our study, we can choose reasonable prediction models in practice, which can guide the research of software defect prediction to a certain extent.

This work is partly supported by the National Natural Science Foundation of China (Nos. 61673384, 61502497, 61562015), the Guangxi Key Laboratory of Trusted Software (No.kx201530), the State Key Laboratory for Novel Software Technology at Nanjing University (No.KFKT2014B19), the Scientific Research Innovation Project for Graduate Students of Jiangsu Province (No.KYLX15_1443) and the National Innovative Project for College Students (No.201510290001).