Journal of Computer Applications
计算机应用,2020,40(11):3273－3279
ISSN 1001-9081
CODEN JYIIDU
2020-11-10
http://www.joca.cn

文章编号:1001-9081(2020)11-3273-07
DOI:10.11772/j.issn.1001-9081.2020040464

软件缺陷预测中的数据预处理方法

潘春霞,杨秋辉*,谭武坤,邓惠心,伍　佳
(四川大学 计算机学院,成都 610065)
(*通信作者电子邮箱 yangqiuhui@scu.edu.cn)

摘　要:软件缺陷预测是软件质量保障领域的热点研究课题,缺陷预测模型的质量与训练数据有密切关系。用于缺陷预测的数据集主要存在数据特征的选择和数据类不平衡问题。针对数据特征选择问题,采用软件开发常用的过程特征和新提出的扩展过程特征,然后采用基于聚类分析的特征选择算法进行特征选择;针对数据类不平衡问题,提出改进的Borderline-SMOTE过采样方法,使得训练数据集的正负样本数量相对平衡且合成样本的特征更符合实际样本特征。采用bugzilla、jUnit等项目的开源数据集进行实验,结果表明:所采用的特征选择算法在保证模型F-measure值的同时,可以降低57.94%的模型训练时间;使用改进的Borderline-SMOTE方法处理样本得到的缺陷预测模型在Precision、Recall、F-measure、AUC指标上比原始方法得到的模型平均分别提高了2.36个百分点、1.8个百分点、2.13个百分点、2.36个百分点;引入了扩展过程特征得到的缺陷预测模型比未引入扩展过程特征得到的模型在F-measure值上平均提高了3.79%;与文献中的方法得到的模型相比,所提方法得到的模型在F-measure值上平均提高了15.79%。实验结果证明所提方法能有效提升缺陷预测模型的质量。

关键词:缺陷预测;数据预处理;开发过程特征;特征选择;类不平衡处理
中图分类号:TP311.5　　**文献标志码:**A

Data preprocessing method in software defect prediction

PAN Chunxia, YANG Qiuhui*, TAN Wukun, DENG Huixin, WU Jia
(*College of Computer Science, Sichuan University, Chengdu Sichuan* 610065, *China*)

Abstract: Software defect prediction is a hot research topic in the field of software quality assurance. The quality of defect prediction models is closely related to the training data. The datasets used for defect prediction mainly have the problems of data feature selection and data class imbalance. Aiming at the problem of data feature selection, common process features of software development and the newly proposed extended process features were used, and then the feature selection algorithm based on clustering analysis was used to perform feature selection. Aiming at the data class imbalance problem, an improved Borderline-SMOTE (Borderline-Synthetic Minority Oversampling Technique) method was proposed to make the numbers of positive and negative samples in the training dataset relatively balanced, and make the characteristics of the synthesized samples more consistent with the actual sample characteristics. Experiments were performed by using the open source datasets of projects such as bugzilla and jUnit. The results show that the used feature selection algorithm can reduce the model training time by 57.94% while keeping high F-measure value of the model; compared to the defect prediction model obtained by using the original method to process samples, the model obtained by the improved Borderline-SMOTE method respectively increase the Precision, Recall, F-measure, and AUC (Area Under the Curve) by 2.36 percentage points, 1.8 percentage points, 2.13 percentage points and 2.36 percentage points on average; the defect prediction model obtained by introducing the extended process features has an average improvement of 3.79% in F-measure value compared to the model without the extended process features; compared with the models obtained by methods in the literatures, the model obtained by the proposed method has an average increase of 15.79% in F-measure value. The experimental results prove that the proposed method can effectively improve the quality of the defect prediction model.

Key words: defect prediction; data preprocessing; development process feature; feature selection; class imbalance processing

0　引言

软件缺陷预测通过分析与缺陷有关的历史数据,比如源代码、缺陷报告、测试记录、缺陷修复记录等,发现缺陷在软件中的分布规律,构建缺陷预测模型,对新提交的软件版本进行缺陷分布的预测,以帮助软件测试人员有的放矢地进行软件测试,降低软件的测试成本。软件资源库中存储了与缺陷有关的历史数据,但从软件资源库里收集的数据集往往具有多维度的问题,数据特征很多;对于缺陷数据,还有不平衡问题,

收稿日期:2020-04-14;**修回日期:**2020-07-01;**录用日期:**2020-07-07。

作者简介:潘春霞(1995—),女,四川内江人,硕士研究生,主要研究方向:软件质量保证与测试;　杨秋辉(1970—),女,山东青岛人,副教授,博士,CCF会员,主要研究方向:软件工程、软件项目管理;　谭武坤(1990—),男,安徽亳州人,硕士,主要研究方向:软件测试、软件质量保证;　邓惠心(1996—),女,四川南充人,硕士研究生,主要研究方向:软件质量保证与测试;　伍佳(1996—),女,四川成都人,硕士研究生,主要研究方向:软件缺陷定位。

即有缺陷的数据是少数。数据特征的选择和数据的平衡处理问题是缺陷预测中数据预处理的关键。近年来，一些学者针对缺陷预测问题进行了相应的研究，如Ibrahim等[1]、Wang等[2]，但他们在特征选择后，仅使用随机过采样方法处理不平衡数据，随机复制有缺陷样本，易导致模型的过拟合问题。因此本文将特征选择算法和其他的不平衡处理算法结合，对缺陷类进行分析后再合成新缺陷样本，以更加有效提升模型的训练数据质量。

软件缺陷数据特征主要分为两大类：软件的静态特征和基于开发过程的软件动态特征。文献[3-5]表明：基于开发过程特征构建的软件缺陷预测模型的性能优于基于软件静态特征构建的软件缺陷预测模型。本文为提高缺陷预测模型训练数据的质量，提出在数据特征选取上采用软件开发过程特征数据，同时创新性地提出扩展过程特征，即软件演化版本间静态特征的变化值，将静态特征的变化值作为扩展过程特征，丰富了过程特征内涵；采用聚类分析的特征选择方法，去掉数据集中的非关键特征和冗余特征；针对缺陷数据类的不平衡问题，提出改进的Borderline-SMOTE（Borderline-Synthetic Minority Oversampling Technique），使训练数据集的数据相对平衡。

1 相关工作

软件缺陷预测中，使用哪些特征建立预测模型，对预测结果的质量有很大影响。文献[3-5]对使用过程特征和静态特征分别建立的缺陷预测模型的预测能力进行了对比，得到相同的结论：使用过程特征比使用静态特征建立的预测模型效率更高；文献[6-7]中也提到：用静态特征结合开发过程特征建立的缺陷预测模型具有更好的缺陷预测性能；文献[8]通过调查过程特征在缺陷预测中的作用，总结了部分常被一些文献使用的过程特征，其中包括不同提交者数量、修改行数、先前版本中修复缺陷个数等。本文参考上述工作，使用常规开发过程特征和扩展过程特征作为备选训练特征，将具有这些特征的数据构成了原始的高维数据。

特征选择是降低特征空间维数的重要手段。文献[9]通过对高维数据集使用不同的特征选择算法进行实验发现：大部分时候，缺陷预测模型只需要使用到原数据集的10%的特征；文献[10]使用了一种混合特征选择方法，通过考虑不同的特征排序评估方法和特征子集评估方法发现：移除85%的特征并不会大幅度降低预测性能；文献[11]提出了特征选择方法FECAR（FEature Clustering And feature Ranking），该方法通过对特征进行聚类和排序，能有效去除冗余特征和非关键特征。本文参考以上结论，使用基于聚类分析的特征选择算法，实现数据降维处理。

在解决数据类不平衡方面，文献[12]总结了一些常用的过采样方法：ADASYN（ADAptive SYNthetic sampling approach）、Borderline-SMOTE和Safe Level SMOTE。对于每个少数类，ADASYN[13]使用密度分布来自动确定需要合成的样本数量，不同的少数类样本采用不同的权重分布；Borderline-SMOTE算法[14]则关注少数类中处于边界的样本，针对这些样本进行重采样，从而合成新的样本；Safe Level SMOTE[15]则通过给每个少数类实例分配一个安全级别值，让合成新样本的安全级别更接近安全级别的最大值，从而达到合成新样本的目的；文献[16]通过在7个数据集上分别使用SMOTE、Borderline-SMOTE和ADASYN过采样方法，发现Borderline-SMOTE方法比其他方法具有更好的表现；文献[17]认为在Borderline-SMOTE方法中，非边界少数类样本无法得到充分考虑，因此提出了基于局部自适应距离的过采样方法LAD-SMOTE（Locally Adaptive Distance-Synthetic Minority Oversampling Technique），该方法建立的模型比Borderline-SMOTE方法建立的模型在F-measure值上平均提升了2.17个百分点。本文参考上述结论，采用Borderline-SMOTE算法，并对其提出改进，用以解决类不平衡问题。

2 开发过程特征的收集和选择

首先，以软件项目中的源代码文件为单位，从软件版本控制库中开发人员提交的日志文件中收集源代码文件的过程特征；然后，根据缺陷追踪系统的缺陷数据，对有缺陷的源代码文件进行缺陷标记；最后，在得到的原始数据集上，使用基于聚类分析的特征选择算法，去除原始数据集中的非关键特征和冗余特征，得到有效训练数据集。

2.1 收集软件开发过程特征

数据来源是软件资源库、软件版本控制库等记录软件开发过程的数据库。利用软件项目开发的日志信息，获取软件开发的过程特征数据。根据相关文献[3-8]，软件缺陷的产生与软件代码的修改、开发者数量、功能的增加、原缺陷数、代码的复杂度、操作符的变化等因素相关，本文共收集了55个与这些因素有关的过程特征（包含扩展过程特征），表1列出了部分收集的开发过程特征。

表1 与软件缺陷相关的部分软件开发过程特征

Tab. 1 Some software development process features related to software defects

序号	名称	描述	类别
1	NR	修改次数	常用过程特征
2	TNLA	添加的总行数	
3	TNLD	删除的总行数	
4	ALCA	平均添加行数	
5	ALCD	平均删除行数	
6	NML	修改行数	
7	NDC	开发者数量	
8	INM	是不是新模块	
9	HBR	先前版本bug出现概率	
10	PWMC	类中方法个数变化	扩展过程特征
11	PDIT	继承深度变化	
12	PNOOT	操作符变化	
13	PNOUT	唯一操作符变化	
14	PNOOD	操作数变化	
15	PNOUD	唯一操作数变化	

这些特征中，部分特征需要根据日志文件中的信息，通过编写统计分析程序获得，比如：平均添加行数（Average Lines of Code Added，ALCA）、先前版本bug出现概率（Historical Bug Rate，HBR）等。式(1)和式(2)分别是ALCA、HBR特征的计算公式：

$$I_{\mathrm{ALCA}}=\frac{\text{添加总行数}}{\text{修改次数}} \tag{1}$$

$$I_{\mathrm{HBR}}=\frac{\text{文件出现缺陷的版本数}}{\text{文件的版本数}} \tag{2}$$

扩展过程特征表示了某些静态特征的变化情况，可通过Understand工具获取每个文件不同版本的静态特征值，再通过式(3)计算获得扩展过程特征，fs_1和fs_2分别为同一个文件

相邻版本前后的静态特征值。

$$pfs = fs_2 - fs_1 \tag{3}$$

最后,再以软件源代码文件为单位,匹配日志中文件的缺陷信息和软件缺陷跟踪系统的历史缺陷信息,确定文件是否含有缺陷,并对文件进行人工缺陷标记,得到原始数据集。

2.2　基于聚类分析的开发过程特征选择

上述方式构建的原始数据集中过程特征总数(包含扩展过程特征)为55,数量较多,因此需要使用特征选择算法去除其中存在的非关键特征和冗余特征。Hall[18]证明了:好的特征子集由与类标高度相关但特征之间不相关的特征组成。例如:操作符的数量和操作数的数量两个特征都与类标高度相关,但二者成正比例关系,关联度较高,存在一定的冗余,故二者组成的子集不是质量好的子集。本文采用基于聚类分析的特征选择方法[11],主要分为2步:特征聚类和特征排序。

特征聚类　利用对称不确定性(Symmetric Uncertainty, SU)计算特征之间的关联度,对特征进行聚类,使类中特征的关联度高,类间特征的关联度低。在选取类的初始代表特征时,使用信息增益(Information Gain, IG)最高的前K个特征。

特征排序　使用信息增益IG计算每个特征与类标相关度,并对结果排序,选取每类中相关度最高的几个特征。图1描述了其流程。

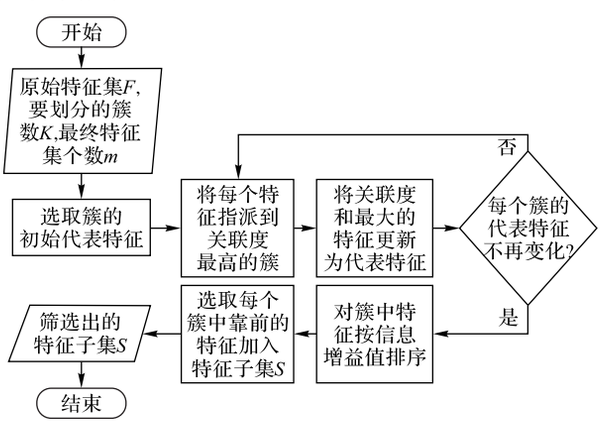


图1　基于聚类分析的特征选择算法流程

Fig. 1　Flowchart of feature selection algorithm based on clustering analysis

原始数据集经过基于聚类分析的特征选择后,能去除其中的非关键特征和冗余特征,缩减数据规模,减少模型的训练时间。

3　基于边界样本的类不平衡处理

在收集的数据集上,无缺陷的数据集一般都远大于有缺陷的数据集,存在数据不平衡问题,会严重影响缺陷分类器的性能。目前在机器学习领域,调整类不平衡问题可以从数据层面和算法层面入手。从数据层面解决包含过采样和欠采样方法。在一些小型项目和版本数量较少的项目中,缺陷样本数量很少,如果使用欠采样方法,将导致最后的训练样本数据量较少,不能得到好的预测模型。经过作者实验,在本文的数据集上使用过采样方法比使用欠采样方法建立的模型具有更好的性能表现,因此本文采用过采样方法。Borderline-SMOTE方法是一种常用的过采样方法,其主要实现流程是:首先,确定位于少数类和多数类之间的边界少数类样本,其样本数为$dnum$;然后,在每个边界少数类样本的$knum$个近邻少数类样本中随机选取$snum$($snum \leqslant knum$)个少数类样本,以合成新的少数类样本;最终,少数类的个数=原始少数类个数+$dnum*snum$,大部分情况下,不与多数类个数一致。

合成新样本时,首先计算边界缺陷样本$\boldsymbol{q}$和它邻近缺陷样本之间的差$\boldsymbol{dif}$,新样本$\boldsymbol{s}$合成公式为:$\boldsymbol{s}=\boldsymbol{q}$+random(0,1)*$\boldsymbol{dif}$,random(0,1)表示(0,1)的随机数。假设样本特征集是(修改频率,开发人员经验,…),其中一个有缺陷实例$\boldsymbol{q}_1$=(10, 100,…),$\boldsymbol{q}_1$的近邻有缺陷实例$\boldsymbol{q}_2$=(18,80,…),则$\boldsymbol{dif}=\boldsymbol{q}_1-\boldsymbol{q}_2$=(−8,20,…),假设random(0,1)是0.5,则新的有缺陷实例$\boldsymbol{s}=\boldsymbol{q}_1$+0.5*$\boldsymbol{dif}$=(6, 110,…)。在实际中,修改频率越大,文件出现缺陷的可能性应该越大,但新合成的实例$\boldsymbol{s}$的修改频率比$\boldsymbol{q}_1$和$\boldsymbol{q}_2$都小,更符合无缺陷情况。同时,$\boldsymbol{s}$的开发人员经验比$\boldsymbol{q}_1$和$\boldsymbol{q}_2$都高,也更符合无缺陷情况。因此,该方法合成的样本值不符合真实情况。

为解决上述问题,本文对Borderline-SMOTE方法做出改进,根据每个特征的具体特点,做如下修改:首先,对$\boldsymbol{dif}$中所有元素取绝对值;然后,对于每个特征,如果该特征的值越小,样本出现缺陷的概率越大,则该特征对应的$\boldsymbol{dif}$中的元素值取负值,否则保持对应特征值为正值,如:开发人员经验、类连续不含有缺陷的周期等特征对应的$\boldsymbol{dif}$中的元素值应取负值,修改次数、类中方法个数变化等特征对应值应取正值。通过分析具体特征和缺陷出现的关系,修改$\boldsymbol{dif}$以使其更加合理。改进的Borderline-SMOTE算法描述如算法1所示。

算法1　改进的Borderline-SMOTE算法。

输入　缺陷样本集$P=\{\boldsymbol{p}_1,\boldsymbol{p}_2,\cdots,\boldsymbol{p}_{pnum}\}$,无缺陷样本集$N=\{\boldsymbol{n}_1,\boldsymbol{n}_2,\cdots,\boldsymbol{n}_{nnum}\}$,有缺陷边界样本$DRANGER=\{\}$,缺陷样本的邻近样本个数$mnum$,$DRANGER$中边界缺陷样本的邻近缺陷样本个数$knum$,每个边界缺陷样本合成新的样本数$snum$($1 \leqslant snum \leqslant knum$),判断$\boldsymbol{dif}$元素值取正/负的数组$A=\{a[1],a[2],\cdots,a[m]\}$($m$为特征个数,$a[m]=0$或1)。

输出　新合成的缺陷类样本OS。

```
OS = ∅
for (i=1;i<=pnum;i++){
  mn_i=KNN(p_i,mnum).majorityNumber()
                //mn_i为p_i的mnum最近邻样本中的无缺陷样本数
  if(mn_i<mnum/2||mn_i==mnum)
    continue
  else
    DRANGER.add(p_i)                    //p_i为边界缺陷样本
}
for(t=1;t<=DRANGER.size;t++){
  q_t=DRANGER.get(t);
  K_t=KNN(q_t,knum)
            //K_t为边界缺陷样本q_t的knum最近邻缺陷样本集
  for(j=1;j<=snum;j++){
          //在K_t中随机选择snum个缺陷样本,以合成新样本
    pp = K_t.randomOne()
    K_t.remove(pp)
    dif_j = q_t - pp                    //计算缺陷样本之间的dif
    dif_j = |dif_j|                     //样本特征差值均取绝对值
    for(n=0;n<m;n++){
      if(a[n] == 0)
      //0代表特征值越小,样本有缺陷概率越大,对应特征值
      //取负
        dif_j[n] = -dif_j[n]
      else
```

```
            //1代表特征值越大,样本有缺陷概率越大,对应特征
            //值取正
            continue
        }
        r_j = Random(0, 1)
        synthetic_j = q_t + r_j × dif_j
        OS.add(synthetic_j)
    }
}
Return OS
```

使用改进的Borderline-SMOTE算法对数据集进行处理，能增加更符合实际样本特征的缺陷类样本实例，解决类不平衡问题，提升数据集质量。

4 实验验证

本文采用随机森林算法训练分类模型，使用精确率(Precision)、召回率(Recall)等指标评价模型性能，并设计了4个实验来验证本文数据预处理方法的有效性。

4.1 实验设计

4.1.1 分类模型

目前，软件缺陷预测领域较常用的分类算法为：朴素贝叶斯分类、决策树、随机森林、逻辑回归等[3,10,19]。通过作者的大量实验，发现随机森林分类算法在大多数数据集上有较好的表现。本文从bugzilla等项目的日志信息中获取原始数据集，进行数据预处理后，使用Weka机器学习库，选择随机森林分类算法，将预处理后的数据作为训练样本输入训练模型。

4.1.2 评价指标

数据的质量直接影响分类模型的性能。通过测评模型性能，可间接测评数据质量，若数据质量较好，则对应的模型性能表现也应更好。混淆矩阵是评价模型好坏的展示工具，对应缺陷预测主题，其内容见表2。

表2 混淆矩阵

Tab. 2 Confusion matrix

真实值	预测值	
	有缺陷	无缺陷
有缺陷	*TP*(True Positive)	*FN*(False Negative)
无缺陷	*FP*(False Positive)	*TN*(True Negative)

根据混淆矩阵衍生了各种评价指标，本文采用的评价指标为Precision、Recall、F-measure和ROC(Receiver Operating Characteristics)曲线下面积AUC(Area Under the Curve)。精确率表征预测模型识别的正类样本中正确的比率，其定义见式(4)。召回率表征预测模型对正类样本的识别程度，其定义见式(5)。F-measure是Precision和Recall的调和平均数，更能整体性地反映模型性能的好坏，其定义见式(6)。ROC曲线横坐标为假阳性率FPR(False Positive Rate)，纵坐标为Recall，假阳性率表征无缺陷实例被预测成有缺陷的概率，其定义见式(7)，ROC曲线越靠近左上方，面积AUC值越大，模型的性能越好。

$$Precision = TP/(TP + FP) \tag{4}$$

$$Recall = TP/(TP + FN) \tag{5}$$

$$F\text{-}measure = \frac{2*Precision*Recall}{Precision + Recall} \tag{6}$$

$$FPR = FP/(FP + TN) \tag{7}$$

4.1.3 实验内容

为了证明本文使用的数据预处理方法的有效性，本文设计了以下4个实验：1)基于聚类分析的特征选择算法的有效性验证；2)改进的Borderline-SMOTE算法的有效性验证；3)引入扩展过程特征有效性验证；4)本文方案与文献[19]方案的对比。

4.2 实验结果及分析

4.2.1 基于聚类分析的特征选择算法的有效性验证

为了证明特征选择算法的有效性，本实验分别使用原始数据和特征选择后的数据建立缺陷预测模型。使用的项目数据源是：bugzilla、columba、jdt、mozilla、platform、postgres。

在特征收集阶段，本文共得到初始过程特征55个，其中常用过程特征为29个，扩展过程特征为26个。经过多种m(m为最终选择的特征子集个数)、K(K为划分的簇数)值组合实验发现：当设置m=30、K=10时，能够保证使用选择的特征训练出的模型性能基本不变；当m<30时，训练出的预测模型的F-measure值开始明显降低。因此，使用特征选择算法后，共筛选出30个过程特征，其中常用过程特征为17个，扩展过程特征为13个。筛选出的30个特征如表3所示。

表3 特征选择算法筛选出的特征

Tab. 3 Features selected by feature selection algorithm

序号	名称	描述	类别
1	NLAL	最后一次添加行数	
2	ALCA	平均添加的行数	
3	ALCD	平均删除行数	
4	CCC	复杂度变化	
5	MNLA	添加的最大行数	
6	MNLD	删除的最大行数	
7	AR	添加行数/总行数	
8	DR	删除行数/总行数	常用
9	NML	修改行数	过程
10	MNML	最大修改行数	特征
11	IN	是不是新文件	
12	MVN	模块版本数	
13	MBFP	最大无bug周期	
14	NDC	提交者数量	
15	DE	开发者平均经验	
16	RDE	开发者最近平均经验	
17	HBR	先前版本bug出现概率	
18	PRFC	调用方法数变化	
19	PNOOT	操作符变化	
20	PNOUT	唯一操作符变化	
21	PNOOD	操作数变化	
22	PNOUD	唯一操作数变化	
23	PCN	循环数变化	扩展
24	PIN	If判断数变化	过程
25	PNLV	局部变量数变化	特征
26	PNOPA	公共属性数变化	
27	PWMC	类中方法个数变化	
28	PCBO	类的耦合数变化	
29	PRFC	调用方法数变化	
30	PNEH	异常处理块数变化	

实验结果显示，在多数情况下特征选择后建立的模型的F-measure值与选择前模型的F-measure值相差不大，如表4所示。实验结果说明了非关键特征和冗余特征的存在。

特征选择降低了训练模型时间，特征选择前，使用随机森

林算法在多个项目上训练模型的平均时间为79.63 s,而特征选择后,训练平均时间仅为33.49 s,时间缩短效率为57.94%。在保证模型F-measure值的同时,极大地缩短了模型的训练时间。

表4　特征选择前后模型的F-measure值对比

Tab. 4　Comparison of F-measure values of models before and after feature selection

项目	特征选择前	特征选择后	项目	特征选择前	特征选择后
bugzilla	0.762	0.761	mozilla	0.687	0.685
columba	0.594	0.586	platform	0.680	0.666
jdt	0.735	0.735	postgres	0.646	0.651

4.2.2　改进的Borderline-SMOTE算法的有效性验证

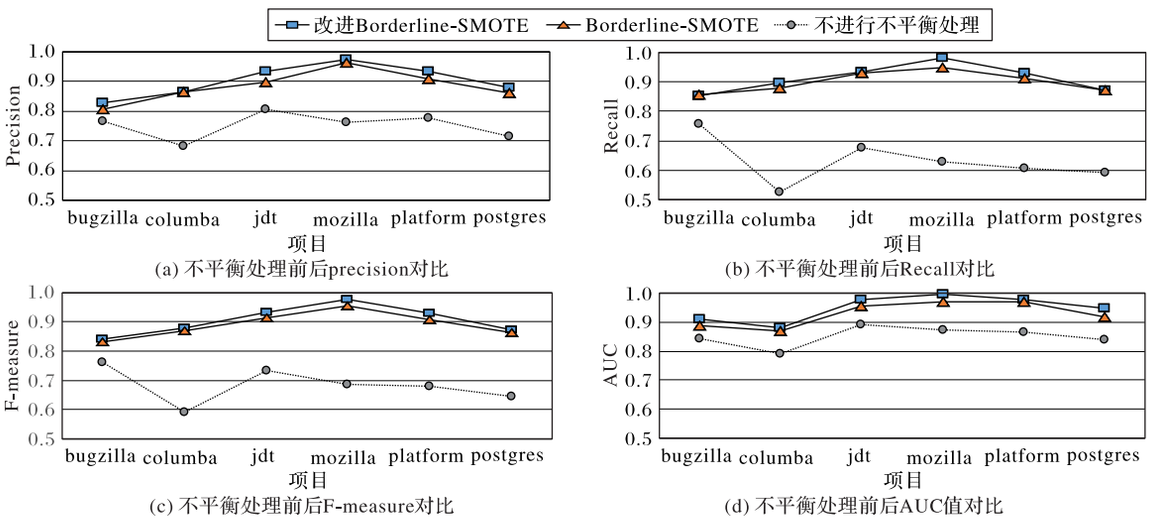
为了证明本文提出的改进的Borderline-SMOTE算法的有效性,本实验分别使用原始和改进的Borderline-SMOTE方法进行数据不平衡处理,实验使用的项目分别是:bugzilla、columba、jdt,mozilla、platform、postgres,处理前后的样本数量见表5。改进的Borderline-SMOTE方法与原始方法相比,只会更改新合成样本的特征值,使特征值更符合真实情况,并不会改变新合成样本的数量,因此使用两种方法处理后的缺陷样本数量一致。

表5　数据不平衡处理前后样本数量

Tab. 5　Samples before and after data imbalance processing

项目	无缺陷样本数	缺陷样本数	原始/改进Borderline-SMOTE方法处理后缺陷样本数
bugzilla	2 924	1 666	3 392
columba	3 094	1 361	4 623
jdt	30 297	5 089	30 534
mozilla	88 126	10 149	97 682
platform	54 798	9 452	56 712
postgres	15 312	5 119	15 357

图2比较了在采用随机森林分类算法时,不同项目在不平衡处理前后对应模型的精确率、召回率、F-measure和*AUC*。从图2可以看出使用原Borderline-SMOTE方法和改进后的Borderline-SMOTE方法处理不平衡样本,都能明显提升各项模型评价指标,但改进后的Borderline-SMOTE方法在各项指标上的表现都优于原始方法。相比原始Borderline-SMOTE方法,使用改进后的Borderline-SMOTE方法进行数据不平衡处理后,所建立的模型在Precision、Recall、F-measure、AUC上都有所提升,平均分别提高了2.36、1.8、2.13、2.36个百分点,证明了改进的Borderline-SMOTE算法处理数据不平衡问题的有效性。



(a) 不平衡处理前后precision对比　(b) 不平衡处理前后Recall对比

(c) 不平衡处理前后F-measure对比　(d) 不平衡处理前后AUC值对比

图2　样本不平衡处理前后预测效果对比

Fig. 2　Comparison of prediction effects before and after sample imbalance processing

4.2.3　引入扩展过程特征有效性验证

为了验证本文提出的扩展过程特征的有效性,本实验在完成了特征选择和不平衡处理的6个数据集上,对比了在特征集中是否引入扩展过程特征所建立的模型的F-measure值,表6是对比结果。

表6　引入扩展过程特征前后模型的F-measure值对比

Tab. 6　Comparison of F-measure values of models before and after the introduction of extended process features

项目	未引入扩展过程特征	引入扩展过程特征	项目	未引入扩展过程特征	引入扩展过程特征
bugzilla	0.791	0.840	mozilla	0.961	0.976
columba	0.822	0.879	platform	0.906	0.931
jdt	0.903	0.932	postgres	0.856	0.874

由表6可以看出:在所有项目中,引入了扩展过程特征后所建立的模型都比未引入扩展过程特征所建立的模型性能表现更好,F-measure值平均提高了3.79%。其中,bugzilla和columba两个项目的提升效果最为明显,提升率分别为6.19%和6.93%,证明了本文提出的扩展过程特征的有效性。

4.2.4　本文方案与文献[19]方案的对比

为了证明本文方案的有效性,将本文提出方案与文献[19]提出方案进行对比。文献[19]方案为:使用混合特征,即静态特征和动态特征,不对原始特征集进行降维处理,使用随机欠采样方法进行类不平衡处理。实验采用文献[19]中的项目:Android Universal I. L、ANTLR v4、Elasticsearch、jUnit等。文献[19]比较了不同的分类算法构建的模型,其中随机森林算法构建的模型在F-measure指标上表现最好,因此本实验只与其随机森林算法建立模型结果进行比较。文献[19]给出了预测模型结果的F-measure值和bug覆盖率,在bug覆盖率指标上,文献[19]方案和本文方案都取得了较好的结果,因此本

文仅展示两种方案模型的F-measure值对比结果，见图3。

可以看到：所有项目中，本文方案的模型具有更高的F-measure值，平均提高率为15.79%，其中jUnit项目提升最大，本文模型的F-measure值为0.890 9，而文献[19]模型的F-measure值为0.659 1，提高了35.17%。在MapDB项目上，两者F-measure值相差不大，本文模型仅提高了0.88%。查看样本数据发现jUnit项目的有缺陷样本和无缺陷样本数量相差较大，而MapDB项目的有缺陷样本和无缺陷样本数量之差比其他项目更小，本文数据不平衡处理方案在类不平衡明显的样本上表现较好。对于原始数据基本平衡情况，经过本文不平衡处理方法后，对应的预测模型性能可能仅微小提高，此时可以不再使用不平衡处理方法。

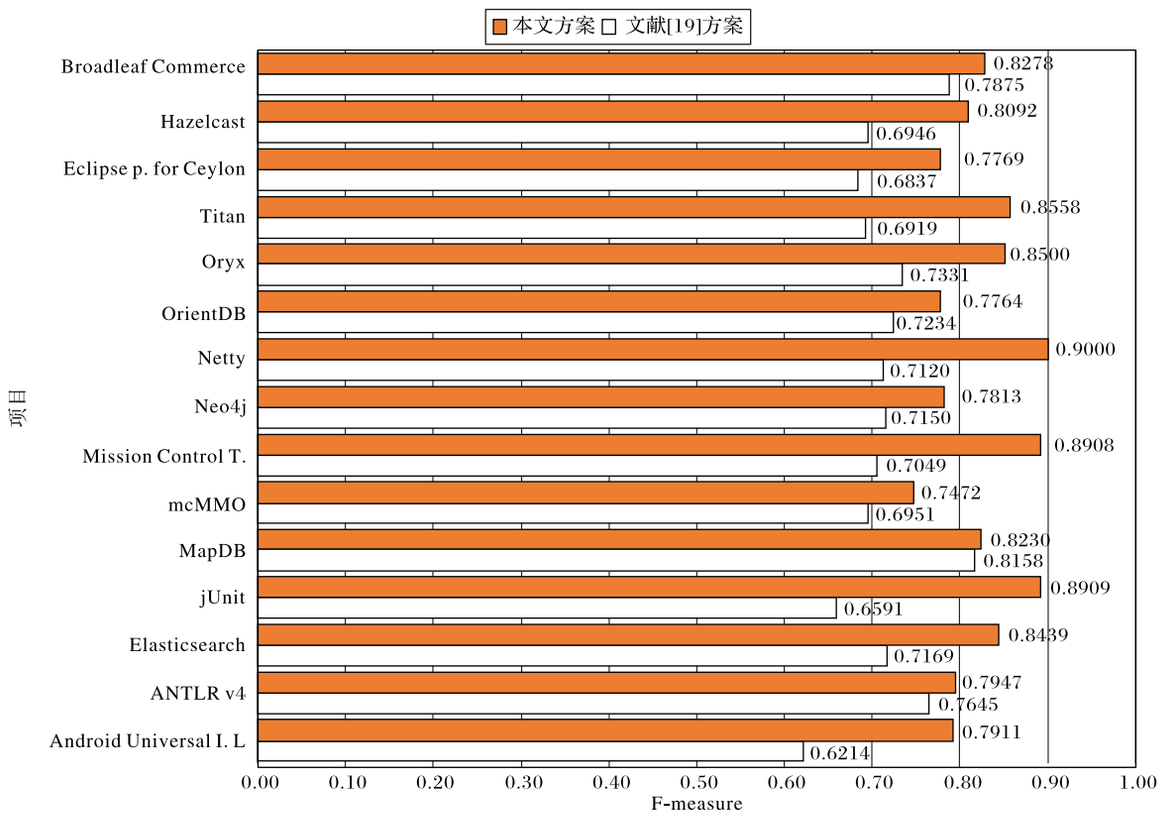


图3 文献[19]方案和本文方案建立模型的F-measure对比

Fig. 3 Comparison of F-measure of the models established by the solution in literature[19] and the solution in this paper

5 结语

本文使用过程特征作为缺陷预测模型的训练特征，分别对特征选择和类不平衡处理方法进行了研究。通过软件资源库获取开发常用过程特征和扩展过程特征，构建原始数据集，采用基于聚类分析的特征选择方法删除非关键特征和冗余特征，使用改进的Borderline-SMOTE过采样方法合成新的缺陷样本，以解决数据分类不平衡问题。实验采用随机森林分类算法构建缺陷预测模型。通过实验，证明了本文数据预处理方法的有效性。

本文旨在解决软件缺陷预测中数据质量问题，但还存在诸多不足之处。比如：本文采用的数据集拥有大量与缺陷有关的历史数据，对于某些新的项目或者历史信息很少的项目，本文方法并不适用，未来可考虑将本文方法与跨项目缺陷预测进行结合。另外，研究收集的缺陷样本中可能会存在噪声数据，比如开发人员忘记声明修复bugid，将会导致该样本被标记为无缺陷。如何有效地发现并去除噪声数据也是未来的研究工作之一。

参考文献（References）

[1] IBRAHIM D R, GHNEMAT R, HUDAIB A. Software defect prediction using feature selection and random forest algorithm[C]// Proceedings of the 2017 International Conference on New Trends in Computing Sciences. Piscataway: IEEE, 2017: 252-257.

[2] WANG F, AI J, ZOU Z. A cluster-based hybrid feature selection method for defect prediction[C]// Proceedings of the IEEE 19th International Conference on Software Quality, Reliability and Security. Piscataway: IEEE, 2019: 1-9.

[3] MOSER R, PEDRYCZ W, SUCCI G. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction[C]// Proceedings of the 30th International Conference on Software Engineering. New York: ACM, 2008: 181-190.

[4] JURECZKO M. Significance of different software metrics in defect prediction[J]. Software Engineering: An International Journal, 2011, 1(1): 86-95.

[5] ÖZTÜRK M M. Which type of metrics are useful to deal with class imbalance in software defect prediction?[J]. Information and Software Technology, 2017, 92: 17-29.

[6] XIA Y, YAN G, ZHANG H. Analyzing the significance of process metrics for TT&C software defect prediction[C]// Proceedings of the IEEE 5th International Conference on Software Engineering and Service Science. Piscataway: IEEE, 2014: 77-81.

[7] MADEYSKI L, JURECZKO M. Which process metrics can significantly improve defect prediction models? An empirical study

[J]. Software Quality Journal, 2015, 23(3): 393-422.

[8] JURECZKO M, MADEYSKI L. A review of process metrics in defect prediction studies [J]. Metody Informatyki Stosowanej, 2011, 30(5): 133-145.

[9] SHIVAJI S, WHITEHEAD E J, AKELLA R, et al. Reducing features to improve code change-based bug prediction [J]. IEEE Transactions on Software Engineering, 2013, 39(4): 552-569.

[10] GAO K, KHOSHGOFTAAR T M, WANG H, et al. Choosing software metrics for defect prediction: an investigation on feature selection techniques [J]. Software: Practice and Experience, 2011, 41(5): 579-606.

[11] 刘望舒,陈翔,顾庆,等. 软件缺陷预测中基于聚类分析的特征选择方法[J]. 中国科学:信息科学, 2016, 46(9):1298-1320. (LIU W S, CHEN X, GU Q, et al. A cluster-analysis-based feature-selection method for software defect prediction [J]. SCIENTIA SINICA Informationis, 2016, 46(9):1298-1320.)

[12] GOSAIN A, SARDANA S. Handling class imbalance problem using oversampling techniques: a review [C]// Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics. Piscataway: IEEE, 2017: 79-85.

[13] HE H, BAI Y, GARCIA E A, et al. ADASYN: adaptive synthetic sampling approach for imbalanced learning [C]// Proceedings of the 2008 IEEE International Joint Conference on Neural Networks. Piscataway: IEEE, 2008: 1322-1328.

[14] HAN H, WANG W, MAO B. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning [C]// Proceedings of the 2005 International Conference on Intelligent Computing, LNCS 3644. Berlin: Springer, 2005: 878-887.

[15] BUNKHUMPORNPAT C, SINAPIROMSARAN K, LURSINSAP C. Safe-level-SMOTE: safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem [C]// Proceedings of the 2009 Pacific-Asia Conference on Knowledge Discovery and Data Mining, LNCS 5476. Berlin: Springer, 2009: 475-482.

[16] CAHYANA N, KHOMSAH S, ARIBOWO A S. Improving imbalanced dataset classification using oversampling and gradient boosting [C]// Proceedings of the 5th International Conference on Science in Information Technology. Piscataway: IEEE, 2019: 217-222.

[17] WANG H, HUANG H. LAD-SMOTE: a new oversampling method based on locally adaptive distance [C]// Proceedings of the 9th International Conference on Intelligent Control and Information Processing. Piscataway: IEEE, 2018: 305-311.

[18] HALL M A. Correlation-based feature selection for discrete and numeric class machine learning [C]// Proceedings of the 17th International Conference on Machine Learning. San Francisco: Morgan Kaufmann, 2000: 359-366

[19] TÓTH Z, GYIMESI P, FERENC R. A public bug database of github projects and its application in bug prediction [C]// Proceedings of the 2016 International Conference on Computational Science and Its Applications, LNCS 9789. Cham: Springer, 2016: 625-638.

PAN Chunxia, born in 1995, M. S. candidate. Her research interests include software quality assurance and testing.

YANG Qiuhui, born in 1970, Ph. D., associate professor. Her research interests include software engineering, software project management.

TAN Wukun, born in 1990, M. S. His research interests include software testing, software quality assurance.

DENG Huixin, born in 1996, M. S. candidate. Her research interests include software quality assurance and testing.

WU Jia, born in 1996, M. S. candidate. Her research interests include software defect location.