工作量感知软件缺陷预测中偏斜分布的影响及测试评估方法

郭育晨，朱晓燕
（西安交通大学计算机科学与技术学院，710049，西安）

摘要：针对工作量感知软件缺陷预测中传统模型测试评估方法存在偏差这一问题，采用偏斜分布的偏度作为数值特征，研究了3种主要测试评估方法的测试集在工作量偏度的偏差和与其对应的估计误差，并基于偏度偏差较小的采样余量方法，提出一种改进方法——后采样方法，所提后采样方法能够保持测试集的类标签比例以避免生成无效测试集。研究结果表明：最常用的十折交叉验证方法偏度偏差最大，其估计误差也最大；与十折交叉验证相比，改进方法性能估计误差减少约4.9%～26.9%；与采样余量方法相比，改进方法不会产生无效测试集，并证明了减小测试集偏度偏差以减少估计误差的有效性。所提后采样方法为工作量感知软件缺陷预测提供了一种更可靠的测试评估方法，能够更准确地评估模型性能。

关键词：软件缺陷预测；工作量感知；偏斜分布；测试评估方法
中图分类号：TP311.5 **文献标志码**：A
DOI：10.7652/xjtuxb202407019 **文章编号**：0253-987X(2024)07-0203-11

Impact of Skewed Effort Distribution in Effort-Aware Software Defect Prediction and Its Validation Techniques

GUO Yuchen, ZHU Xiaoyan
(School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: To handle the bias in traditional model validation methods for effort-aware software defect prediction, the skewness of skewed distribution is used as numeric feature to investigate the bias of effort skewness and estimation bias in the datasets of three major validation techniques. Based on out-of-sample method which is less biased in effort skewness, a refined hold-to-sample method is proposed, which maintains the class label proportion of the testing data to avoid generating invalid testing data. Results show that the commonly used 10-fold cross-validation method has the greatest bias in effort skewness and the greatest estimation bias. In comparison with this method, our refined method reduces the evaluation bias by 4.9%—26.9%; in comparison with the out-of-sample, it does not generate invalid testing data. Experiments confirm the effectiveness of reducing the evaluation bias by decreasing bias of skewness in testing data. The hold-to-sample method provides a more reliable validation method for effort-aware software defect prediction and can evaluate model performance more accurately.

Keywords: software defect prediction; effort-aware; skewed effort; validation method

收稿日期：2024-01-31。 作者简介：郭育晨(1990—)，男，博士生；朱晓燕(通信作者)，女，副教授，博士生导师。 基金项目：国家自然科学基金资助项目(92046009)。
网络出版时间：2024-04-17 网络出版地址：https://link.cnki.net/urlid/61.1069.T.20240416.1001.002

在软件开发和维护过程中,与软件缺陷相关的工作量占据了软件工程成本的很大一部分。据估计,定位、修复软件缺陷(defect,也称为 bug)及其相关工作占总开发成本的 40%~80%[1]。软件缺陷预测希望通过引入机器学习技术,对历史开发数据进行数据挖掘,从而实现预测和预警潜在缺陷,帮助软件开发人员合理分配开发资源和工作量,以节省工程成本。

传统软件缺陷预测研究把软件缺陷预测看作一个分类问题[2],应用机器学习中的分类算法对软件开发过程中的一些特征数据进行学习,如软件代码模块的结构和特征信息、软件版本管理仓库的提交特征数据、缺陷报告等。分类算法在软件特征数据上训练之后,便可以对潜在的软件缺陷发出预警,预测新的软件模块(module)是否含有缺陷。然而,这样二分类的预测预警并没有考虑到在实际中的效率问题,因此有了工作量感知(effort-aware)软件缺陷预测。

工作量感知缺陷预测要求使用更少的工作量来发现更多的软件缺陷。这不仅要求预测是否有缺陷,还要求根据检查潜在缺陷工作量的工效比进行排序[3],以此提高找到软件缺陷的效率。比如文献[4]中提出的工作量感知线性回归(EALR)方法,用 20%的工作量就可以找到 35%的缺陷。文献[5]中提出的双层集成学习(TLEL)方法,只用 20%的工作量就能找到 70%的缺陷。可见,只要根据预测模型给出的排序来检查软件模块,就可以用有限的测试资源找到更多潜在的软件缺陷,从而提高软件缺陷预测的效率和实用性。

近些年来,学者们对工作量感知缺陷预测的关注度逐渐提升[6],更多的软件缺陷预测研究在评价模型预测性能时使用了工作量感知性能指标[6],一些工作量感知缺陷预测模型被提出[3-5, 7-11],但在单个数据集上评估软件缺陷预测模型性能时,仍然沿用机器学习领域中的经典测试评估方法,如十折交叉验证[12]。这是因为经典测试评估方法常用来评估分类性能。文献[13]研究了多种测试评估方法对软件缺陷预测模型分类性能估计的准确度和偏差,并推荐采样余量(out-of-sample)方法以得到更好的综合评估准确度。根据文献[6,12]的统计,经典的交叉验证(cross-validation)是使用最多的测试评估方法。

源自机器学习领域的测试评估方法并未考虑工作量感知缺陷预测的偏斜分布问题。本文在研究偏斜分布对工作量感知预测模型性能的影响时发现,十折交叉验证从原始数据集上生成的测试集的工作量数值分布,往往和原始数据的工作量数值分布偏差较大。这种分布上的差异,将会导致对工作量感知缺陷预测模型的测试评估结果不够准确。文献[14]证实了偏斜分布对模型预测性能有不小的影响。数值化这种影响的大小,正是本文的研究动机。

现有研究较少考虑到不同测试评估方法生成的测试集,导致其工作量的偏差会有很大不同。从这一点出发,本文用偏度(skewness)作为工作量偏斜分布的特征,数值化研究测试集和原始数据集的偏差。本文假设当测试集工作量的偏度与原始数据集工作量的偏度相差很大,那么在测试集上对模型排序性能的测试评估将会不准确。基于这个假设,开展了研究并在真实数据集上验证本文的假设。

1 工作量的偏斜分布

1.1 真实数据集工作量的分布

在工作量感知软件缺陷预测中,处理每一个缺陷预警的工作量(effort)是不同的。如果把工作量数值分布可视化,就会发现工作量数值分布是一个天然向左的偏斜分布,如图 1 所示。这是因为大部分软件模块并不需要很多的工作量,只有少数软件模块需要非常多的工作量。

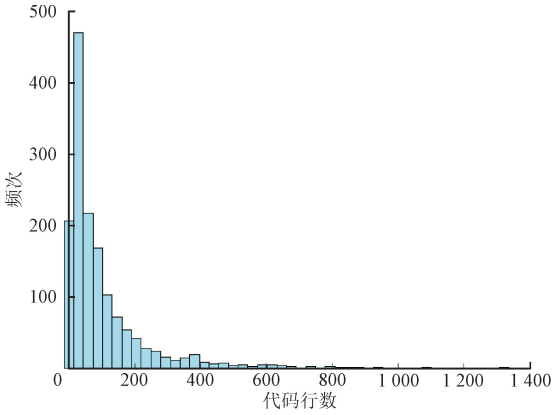

图 1 工作量的直方图

Fig. 1 Histogram of effort

在软件缺陷预测中一般用软件模块尺寸,即代码行数近似估计工作量,这是基于越大的软件模块需要更多的工作量这一假设。文献[15]研究了复杂模型计算的工作量,并用代码行数近似相比,发现应用于工作量感知软件缺陷预测中结果是相似的。因此工作量感知软件缺陷预测的研究[3-5, 8-11, 13, 15-17]基

本都采用代码行数(line of codes, LOC)或者修改行数(churn)来估计工作量。

从图 1 可以观察到，大部分模块比较小，需要检查的代码在 180 行以内，聚集在整体分布的左侧，整个分布就向左偏斜。采用统计学中的偏度来度量工作量分布的偏斜程度[18]，计算公式如下

$$S_{sk}=\frac{1}{N}\sum_{i=1}^{N}\frac{(x_i-\bar{x})^3}{\sigma} \tag{1}$$

式中：N 是工作量数据个数；$x_1,x_2,\cdots,x_N$ 是工作量数据；$\bar{x}$ 是工作量的均值；σ 是工作量的标准差；S_{sk} 表示偏度计算值。

由式(1)计算可得图 1 分布的偏度是 3.2，偏度为正，表示向左偏斜。偏度越大，偏斜程度越高；偏斜程度越高，软件模块的分布就在左侧越集中。

表 1 统计了本文使用的数个软件缺陷数据集工作量的偏度。从表 1 可以看到，缺陷数据集中工作量分布的偏斜程度差异很大，偏度取值从 1.4 到 141.3，中位数为 9.7。可见，大多数缺陷预测数据集中工作量的数值分布是高度向左偏斜的偏斜分布。

表 1　缺陷数据集中工作量偏度的统计信息

Table 1　Statistics of effort skewness in defect datasets

统计量	偏度
最小值	1.4
较小四分位数	4.1
中位数	9.7
较大四分位数	28.4
最大值	141.3

1.2　测试集工作量的偏斜分布

在单个数据集上测试评估预测模型性能时，要从原数据集上划分或者采样生成训练集和测试集，故测试集的生成方法是由具体的测试评估方法决定的。测试集工作量的分布是否靠近原始数据集也是由具体的测试评估方法决定的。虽然文献[13]研究了更多的测试评估方法，但是那些方法在软件缺陷预测的研究中并不常见。本文采用工作量感知软件缺陷预测中最常见的 3 种测试评估方法——十折交叉验证、五折交叉验证和采样余量方法进行研究。

根据文献[12]的统计，经典的十次十折交叉验证(10x10-fold cross-validation)是使用最多的测试评估方法。十折交叉验证将原始数据集均匀划分为 10 个子集，将每个子集轮流作为测试集，剩下的 9 个子集组成训练集来测试评估性能。一轮 10 次，再重复 10 轮，就是最常用的十次十折交叉验证。最终估计的性能是 100 次性能得分的平均值。

近几年也开始有研究[19]使用文献[13]推荐的采样余量测试评估方法，因为其在评估分类性能时的综合准确度最好[13]。采样余量基于重复抽样法(boostrap)，通过对原始数据集进行有放回的采样生成训练集，而测试集则是由未被采样到、不在训练集中的数据条目构成的，因此测试集的规模不固定，平均规模为原始数据集的 36.8%[13]。

除了在单个数据集上测试评估预测性能，参数调整是测试评估方法的另一个应用场景。比如文献[20]使用网格搜索(grid search)全自动参数优化方法，具体采用五折交叉验证来评估性能和优化参数[21]。五折交叉验证和十折交叉验证类似，将原始数据集均匀划分为 5 个子集，每个子集都会作为测试集一次，所以测试集的规模是原始数据集的 20%。类似地，文献[17]在调整参数时使用采样余量方法，验证评估方法的准确度将会直接影响模型参数的设置。

本文研究十折交叉验证、五折交叉验证和采样余量 3 种方法生成的测试集的偏度，其测试集的规模分别是原始数据集的 10%、20%和大约 36.8%。分别用 3 种测试评估方法在一个真实数据集上各自生成 100 个测试集，3 种方法的 100 测试集的偏度分布密度见图 2。图中的竖线是原数据集的工作量偏度，即 7.5。只有当测试集的偏度靠近竖线时，测试集的工作量偏度才会接近原数据集的工作量偏度。

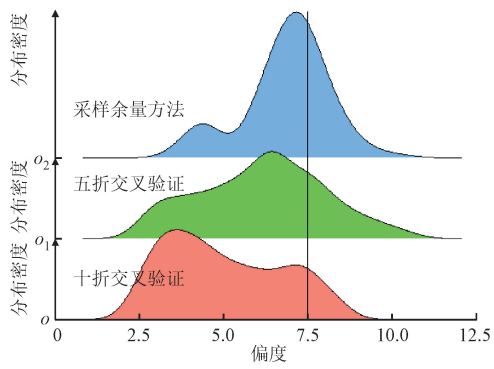

图 2　测试数据中工作量偏度的分布

Fig. 2　Distribution of effort skewness in testing data

图 2 中红色分布是十折交叉验证，可以看到其测试集的工作量偏度在 2.5～5.0 这个区间分布密度最高，但是这个区间没有包含原始数据集的偏度 7.5，说明十折交叉验证测试集偏斜程度的差异较

大。绿色是五折交叉验证,和十折交叉验证相比更接近 7.5 的竖线,但其密度分布较为扁平。蓝色的是采样余量方法,在竖线左边附近有较高的峰,说明采样余量方法生成的测试集最靠近原始数据集工作量的偏斜分布。3 种方法的测试集在工作量偏度这个指标上有明显区别,说明研究测试集的偏度有其理论指导意义。

1.3 改进的采样办法

本文在实验中发现,虽然采样余量方法的测试集工作量的偏斜程度和原始数据集相差较小,但是采样余量方法有概率生成无效测试集。无效测试集是指测试集完全不包含缺陷数据,缺陷数目为 0,此时无法计算缺陷的召回率等性能指标。因此本文提出一种改进方法,称为后采样(hold-to-sample)方法。

在采样余量方法中,通过对原始数据集进行有放回的采样来生成训练集,训练集规模和原数据集一致;测试集由未被采样到、不在训练集中的数据条目构成。由于测试集是被动生成的,因此规模不固定(平均规模 36.8%[13]),同时缺陷数据的比例也不稳定。由于软件缺陷预测数据集的类标签本身是不均衡的[22],缺陷数据是少数类,因此当测试集的缺陷数据条目数减少到 0 时,对工作量感知缺陷预测来说就是无效数据集。本文实验发现,采样余量方法有一定概率产生无效数据集。

针对此问题,本文做出改进:把测试集的生成方式从被动生成改为主动划分。先主动划分测试集,然后采样生成与采样余量方法同样的训练集。本文称这种方法为后采样的拟重复抽样法(boostrap)方法,具体步骤如下。

步骤 1 按照 36.8%的规模,从原数据集中提前划分测试集,同时保持类标签比例和原始数据集一致,这样不仅可以避免生成无效数据集,还会让测试结果更准确。

步骤 2 对剩下的 63.2%的数据通过有放回的采样进行补充,补充到和原始数据集规模一致。

在采样余量方法中,训练集的规模和原始数据集规模一致,由大约 63.2%(非重复)的数据条目和重复条目组成。本文提出的后采样方法的训练集由 63.2%的非重复数据条目和补充的重复条目组成。两个方法的训练集基本保持一致。

2 实验方法

2.1 数据集

本文使用的软件缺陷预测数据集来自多个研究和数据集仓库,分别是 PROMISE 数据集[23]、AEEEM 数据集[24]、文献[25]中 Tian 等提供的 JiT 数据集、文献[4]中 Kamei 等提供的 JiT 数据集、文献[16]中 Ni 等提供的 JavaScript 数据集,共计 120 个数据集。实验中,PROMISE 数据集有部分数据集非常小,仅为 10%规模的测试集,数据条目数不足 10 条,作为测试数据样本量太少。按照数据集规模要求,本文排除了数据条目不足 100 条数据的数据集。另外,在有监督框架实验中,bugzilla、pdf.js、postgres、Mylyn 共 4 个数据集无法生成满足条件的划分,因此未在有监督框架实验中使用,最终使用了 95 个数据集。

2.2 工作量感知性能指标

本文使用工作量感知缺陷预测中常用的 20%工作量召回率和归一化工效曲线下面积作为实验研究的工作量感知性能指标。其中,20%工作量召回率度量的是 20%工作量能够找到的缺陷模块占总缺陷模块数的占比,记作 R_{20}

$$R_{20} = k/K \tag{2}$$

式中:k 是用 20% 工作量发现的缺陷模块数;K 是缺陷模块的总数。R_{20} 越高说明预测性能越好,能找到更多的缺陷。R_{20} 有时也被记作 Acc 性能指标[4,8]。文献[3]用到的性能指标 20%工作量缺陷检出率(PofB20)和 R_{20} 类似,计算的是缺陷数的召回率,适用于一个模块含多个缺陷的情况。如果每个模块最多含有一个缺陷,那么 PofB20 和 R_{20} 是相等的。但是有的数据集只有缺陷与否的类标签数据,缺陷数不是所有数据集都有,为保持一致,本文选择使用 R_{20}。

归一化工效曲线下面积计算的是工效曲线的曲线下面积。图 3 是一个工效曲线的例子,图中蓝色的线代表模型预测排序的工效曲线,红色的是最差解,绿色的是最优解。当模型给出排序预测之后,按照预测顺序检查软件模块,随着工作量的增加,被找到的缺陷也会增加,将这个过程在坐标轴上画成曲线即为工效曲线。

归一化工效曲线下面积(P_{opt})的计算过程为:首先计算预测模型工效曲线的曲线下面积 S_{m},然后分别计算最优解和最差解的曲线下面积 S_{opt} 和 S_{worst},最后用如下公式计算得到

$$P_{\text{opt}} = 1 - \frac{S_{\text{opt}} - S_{\text{m}}}{S_{\text{opt}} - S_{\text{worst}}} \tag{3}$$

可以看到,P_{opt} 是一个归一化的性能指标,取值 0～1。

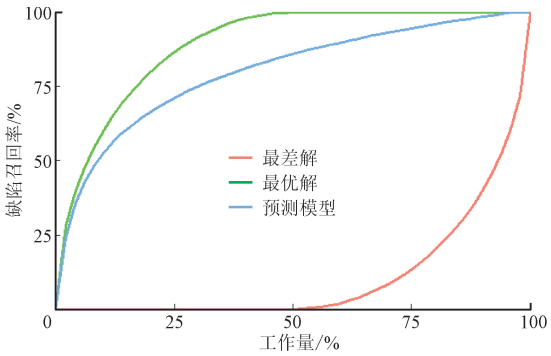

图 3　工效曲线

Fig. 3　Cost-effectiveness curve

2.3　工作量感知缺陷预测的基线模型

本文使用无监督基线模型 ManualUP[9] 和有监督基线模型[3,11] 先分类后排序方法（classify-before-sorting，CBS+）作为被评估的预测模型。这两个模型因为易于实现和性能优异，在近年的研究[3,15] 中仍然是最主要的基线模型。准确评估这两个基线模型的性能，是比较其他工作量感知软件缺陷预测模型的基础。

无监督模型 ManualUP 根据代码行数对软件模块进行排序和预测。文献[16]使用的无监督基线模型 Churn 可以视作 ManualUP 的等价模型。当软件模块粒度为提交（commit）时，提交的代码行数指的是修改行数。在预测时，两个模型用同样的公式按照代码行数给出排序。

有监督模型 CBS+是文献[11]提出的方法，在分类结果的基础上对软件模块进行排序预测。文献[11]中使用逻辑回归（LR）算法训练分类器，文献[3]提出用随机森林（RF）算法训练分类器性能更好，本文使用的是 CBS+RF 版本。

文献[11]中提到无监督模型会受到工作的偏斜分布的影响，文献[14]发现有监督模型的预测性能也会受到工作的偏斜分布的影响。因此不论有监督模型还是无监督模型，工作量感知的缺陷预测模型都会受工作量偏斜分布的影响。

2.4　实验设计

无监督实验框架只适用于评估无监督模型 ManualUP，其不需要训练数据，而是将整个数据集用作测试集，因此可以排除训练数据和目标数据不同所引入的误差，是理想化的实验框架。

有监督实验框架参考了文献[12]的实验设计，更符合实际使用场景，可用于评估有监督模型和无监督模型。有监督实验框架设计如图 4 所示。将软件缺陷数据集划分为训练数据和目标数据，在训练数据上用评测估计方法对模型性能进行估计，在目标数据上测试预测模型的实际性能，对比估计性能和实际性能即可计算估计误差。

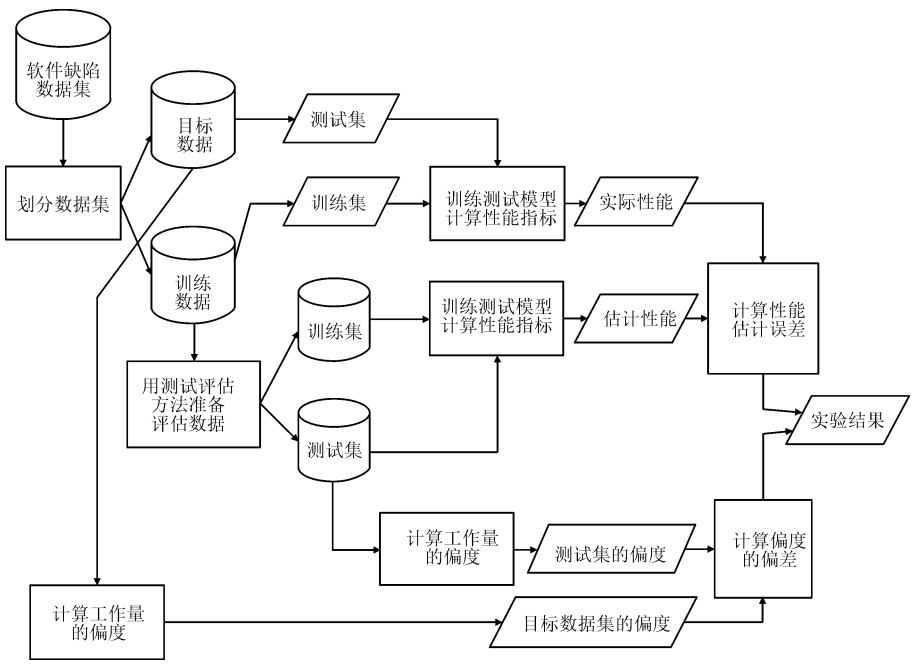

图 4　有监督实验框架

Fig. 4　Framework of supervised experiment

有监督框架划分训练数据和目标数据的方法为：在保持类标签比例不变的情况下，随机将软件缺陷数据集分成 50%规模的两个部分。此时，为了控制实验误差，限制划分的训练数据和目标数据偏度的偏差不大于 6 或者原数据集偏度的 30%，否则重新划分数据集。如果不能在有限次划分生成满足条件的数据集，则原数据集不纳入实验。不同于有监督框架，无监督框架的训练数据和目标数据都是原数据集，没有开始的划分步骤。

在应用评测估计方法在训练数据上评估基线模型性能时，两种框架实验都进行同样的单组实验。单组实验模拟实际应用的场景，如常用的十次十折交叉验证，会生成共计 100 个测试集，对模型性能的估计值则是这 100 个数据集上性能得分的平均值。本文实验对标实际的使用场景，用 4 种测试评估方法分别生成 100 个测试集，作为一组实验，具体配置见表 2。

表 2 单组实验测试评估方法的使用设置

Table 2 Setup of validation methods

测试评估方法	单组重复次数	生成测试集数
十折交叉验证	10	100
五折交叉验证	20	100
采样余量方法	100	100
后采样方法	100	100

由于无监督实验框架和有监督实验框架都有随机划分的误差，因此两种实验都对单组配置进行 100 组重复，而实验结果则是对 100 组的估计误差取平均值。

2.5 误差计算和统计方法

本文一共计算 3 种估计误差：偏度的偏差、R_{20} 的误差和 P_{opt} 的误差。偏度是工作量偏斜分布的偏斜特征值，计算公式如下

$$E_{bias\_sk} = |S_{sk\_o} - S_{sk\_test}| \tag{4}$$

式中：S_{sk_o} 是预测目标数据工作量的偏度；S_{sk_test} 是测试集工作量的偏度。偏度的误差是两个偏度之差的绝对值。

类似地，性能指标 R_{20} 的误差计算公式如下

$$E_{bias\_R20} = |R_{20\_o} - \overline{R_{20\_test}}| \tag{5}$$

式中：R_{20_o} 是预测目标数据的 R_{20} 性能得分；$\overline{R_{20\_test}}$ 是在一组实验即 100 个测试集上 R_{20} 得分的平均值。与偏度的误差不同，这里取平均值是和实际应用保持一致，使用一组实验的均值作为性能得分的估计值，此时 R_{20} 的误差计算的是估计值和真实值之差的绝对值。

同理，P_{opt} 的误差计算公式如下

$$E_{bias\_Popt} = |P_{opt\_o} - \overline{P_{opt\_test}}| \tag{6}$$

式中：P_{opt_o} 是预测目标数据的 P_{opt} 得分；$\overline{P_{opt\_test}}$ 是在一组实验上 P_{opt} 得分的平均值。此时误差计算的是估计值和真实值之差的绝对值。

统计分析方面，使用胜平负计数在每个数据集上进行成对比较，即用提出的后采样方法和其他 3 种方法相比，误差更小为胜，误差相等为平，误差更大为负。采用 Wilcoxon 符号秩检验[26]对观察到的结果进行统计检验。为了降低多次统计检验而累积的统计风险，对表格中所有检验的 p 值应用 Benjamini-Hochberg(BH)方法[27]进行矫正，以控制和减少假阳性率。

3 结果与分析

3.1 测试集工作量的偏差

将无监督框架和有监督框架的实验各自进行 100 组来评估 4 个测试评测方法的偏差和估计误差。每个测试方法在 100 组实验生成 1 万个测试集，测试集工作量偏度的偏差结果如图 5 的箱线图所示。箱线图中长方形的上、下边分别代表上四分位数和下四分位数，中线是中位数，长方形与向上、下延伸的线表示最大值到最小值的范围，其余点表示离群值。实验结果按照数据集分成 5 组，组名是其数据来源。

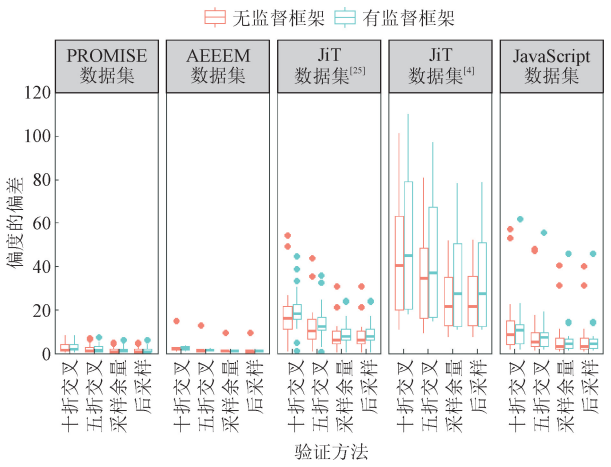

图 5 测试集偏度的偏差

Fig. 5 Bias of skewness in testing dataset

从图 5 可以观察到，红色的无监督框架实验结果和蓝色的有监督框架实验结果十分相近，在 5 个分组都呈现了相同的规律：十折交叉验证的箱图更高，偏差最大；五折交叉验证次之；采样余量和后采样方法的箱图最低，偏差最小。

无监督框架实验计算的是测试集和原数据集的偏差，有监督框架实验计算的是测试集和预测目标数据工作量的偏差。从图 5 可以看到，二者偏差相近、结论相同。这是因为训练数据（原数据）和目标数据偏度比较接近，实际应用中也建议选取和目标数据接近的训练数据。有监督框架实验结果的统计量见表 3。

表 3　偏度的平均偏差

Table 3　Mean bias of skewness

统计值	十折交叉方法	五折交叉方法	采样余量方法	后采样方法
平均偏差	10.147	7.863	5.601	5.587
胜平负计数	95/0/0	95/0/0	75/0/20	
检验 p 值	<0.001	<0.001	<0.001	

表 3 的计算结果与图 5 一致，偏度的偏差由大到小分别是十折交叉验证、五折交叉验证、采样余量方法和后采样方法。其中，偏差最高的十折交叉验证的平均偏差为 10.147，约为后采样方法偏差的 1.8 倍。胜平负计数是把后采样方法和其他 3 种方法在每个数据集上一一对比。从表 3 可以看到，后采样方法在 95 个数据集上的误差都比十折交叉验证和五折交叉验证小（胜代表误差更小）。而对比采样余量方法，在 95 个数据集中的 75 个数据集上后采样方法偏差更小。统计检验的 p 值均小于 0.001，说明有统计显著性。

总之，十折交叉验证生成的测试集工作量分布的偏差最大，五折交叉验证次之，采样余量方法和后采样方法的偏差最小。

3.2　无效测试集的问题

实验发现，采样余量方法生成的部分测试集无法计算性能指标 R_{20}，因为测试集包含有 0 个缺陷，按照式(2)计算召回率会发生错误。在无监督框架实验中，采样余量方法一共产生了 86 个没有缺陷数据的无效测试集。

在有监督框架实验中，除了采样余量方法，十折交叉验证也生成了无效测试集。从原理上来说，如果数据集含有的缺陷数少于 10 个，那么必然不能保证十折中的每一折都有缺陷数据。PROMISE 数据仓库中的 camel-1.0、ivy-1.4、jedit-4.3、synapse-1.0 这 4 个数据集的缺陷数目全部少于 20 个。因此，在有监督框架实验中，划分 50% 规模的数据集包含少于 10 个缺陷，以保证十折交叉验证产生不含缺陷的无效测试集。

表 4 为 4 种测试评估方法生成的无效测试集。根据表 4，本文更推荐五折交叉验证和后采样方法。

表 4　测试评估方法的无效测试集

Table 4　Invalid testing data of validation techniques

评测方法	有无生成无效测试集
十折交叉验证	有监督框架实验
五折交叉验证	实验中未遇到
采样余量方法	无监督框架和有监督框架实验
后采样方法	实验中未遇到

3.3　对基线模型 ManualUP 的估计误差

本文对基线模型 ManualUP 分别进行了无监督框架实验和有监督框架实验。4 种评测估计方法对性能指标 R_{20} 的估计误差如图 6 所示。

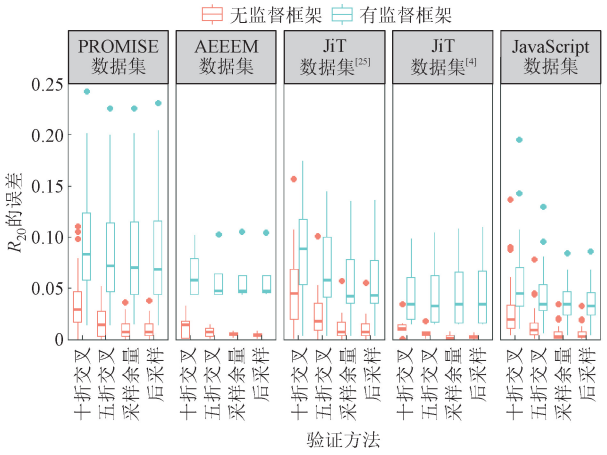

图 6　模型 ManualUP 的 R_{20} 性能的误差

Fig. 6　Bias in performance R_{20} of model ManualUP

从图 6 可见，无监督框架实验的误差总体比有监督框架实验低很多，这是因为无监督框架排除了训练数据和目标数据不同所引入的误差。无监督框架更能理想化、结果更纯净，而有监督框架更符合实际应用场景，对比二者可以更清楚地观察到由偏度偏差造成的估计误差。

图 6 中红色箱图是无监督框架的结果。在全部 5 个分组中，十折交叉验证的箱图最高，五折交叉验证次之，采样余量和后采样方法最低。十折交叉验证的误差平均比后两种方法高 0.02 左右。

图 6 中蓝色箱图是有监督框架实验的结果。在除了 JiT 数据集[4] 的 4 个数据集上，十折交叉验证的误差最高，五折交叉验证次之；采样余量和后采样方法误差最低。在 JiT 数据集[4] 上 4 种方法区别较小，十折交叉验证的中位数线稍高，但是上四分位数线更低。这可能是因为 4 种方法在 JiT 数据集[4] 上偏度的偏差都很大（平均大于 20）造成的。在其他 4 个数据集上，十折交叉验证的误差最大，比采样余量

和后采样方法的估计误差高 0.02 左右，这个差值和无监督框架实验结果相近。以实际应用为准，计算了有监督框架实验的统计量，包括误差的均值、胜平负计数和无效测试集的数目，结果见表 5。

表 5 模型 ManualUP 的 R_{20} 性能的平均误差

Table 5 Mean bias in performance R_{20} of model ManualUP

统计值	十折交叉验证	五折交叉验证	采样余量方法	后采样方法
平均误差	0.084 8	0.069 7	0.065 8	0.065 9
胜平负计数	79/0/16	59/0/36	42/0/53	
检验 p 值	<0.001	<0.001	0.775	

从表 5 可知，十折交叉验正的误差最大，平均为 0.084 8，后采样方法的误差和采样余量方法相近，平均为 0.065 9。这个结果和从图 6 一致。从表 5 的胜平负计数可以看到，后采样方法和十折交叉验证、五折交叉验证相比，分别在 79、59 个数据集上胜出，说明在超过半数以上的数据集上误差更小。统计检验的 p 值在矫正前和矫正后也均小于 0.001，说明有显著差异；采样余量方法相比后采样方法 p 值较大，可以认为没有统计意义。后采样方法对 R_{20} 的估计误差和十折交叉验证相比误差减少了约 22.2%。

图 7 为 4 个测试评估方法对性能指标 P_{opt} 的估计误差。可以观察到和图 6 类似的情况：①无监督框架实验的误差绝对值比有监督框架实验误差低很多；②4 种方法的误差从高到低为十折交叉验证、五折交叉验证、采样余量方法和后采样方法；③在 JiT 数据集[4]上，有监督框架实验 4 种方法差别较小；④无监督框架和有监督框架实验中，采样余量方法和后采样方法相比十折交叉验证的误差低 0.01 左右。以实际应用为准，表 6 列出了有监督框架实验的结果。

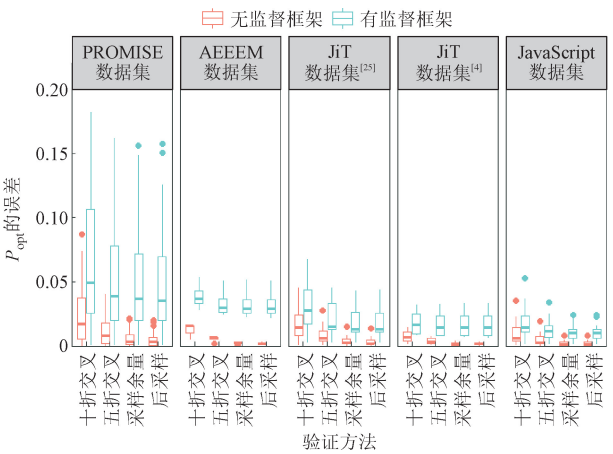

图 7 模型 ManualUP 的 P_{opt} 性能的误差

Fig. 7 Bias in performance P_{opt} of model ManualUP

表 6 模型 ManualUP 的 P_{opt} 性能的平均误差

Table 6 Mean bias in performance P_{opt} of model ManualUP

统计值	十折交叉验证	五折交叉验证	采样余量方法	后采样方法
平均误差	0.045 4	0.036 1	0.033 4	0.033 2
胜平负计数	88/0/7	82/0/13	52/0/43	
检验 p 值	<0.001	<0.001	0.087	

从表 6 可以看出，十折交叉验证的误差最大，平均为 0.045 4；后采样的误差小，平均为 0.033 2，相比十折交叉验证，误差减少大约 26.9%。在一对一对比中，后采样方法和其他 3 种方法相比，分别在 88、82 和 52 个数据集上胜出，说明后采样方法在超过半数以上的数据集上误差更小。后采样方法和十折交叉验证、五折交叉验证相比，统计检验的 p 值在矫正前和矫正后均小于 0.001，说明有显著差异。

综合来看，测试集偏度的偏差最大的十折交叉验证，其对无监督基线模型 ManualUP 性能估计的误差也是最大的。这验证了本文的假设，即如果测试集偏度的偏差很大，那么在测试集上对模型排序性能的测试评估将会不准确。后采样方法和十折交叉验证相比，在 R_{20} 和 P_{opt} 性能指标上的误差分别减少了 22.2%和 26.9%。

3.4 对基线模型 CBS+的估计误差

有监督模型 CBS+是工作量感知缺陷预测中另一个主要的基线模型。不同于 ManualUP 模型，有监督模型 CBS+仅适用于有监督框架的实验。

图 8 展示了 4 个测试评估方法对模型 CBS+的 R_{20} 性能的估计误差。实验结果的箱线图被按照数据集分成 5 个组，可以观察到 5 个组的结果有着相同的规律：十折交叉验证的误差最大，五折交叉验证次之，采样余量方法和后采样方法的误差最小。这个顺序与偏度偏差的结果一致，说明偏差较大时，在

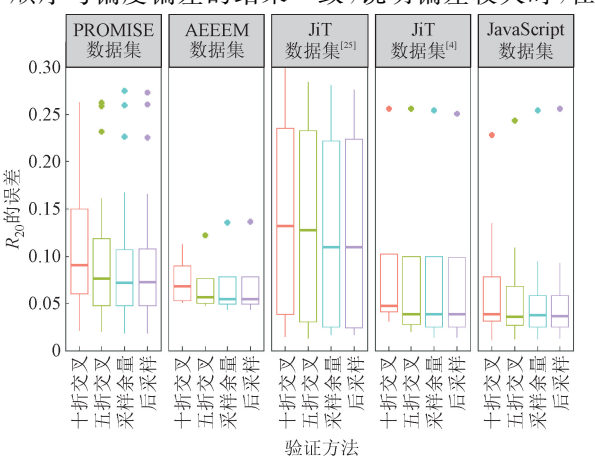

图 8 模型 CBS+的 R_{20} 性能的误差

Fig. 8 Bias in performance R_{20} of model CBS+

性能指标 R_{20} 上的估计误差是更大的。与图 8 对应的统计结果见表 7。

表 7　模型 CBS+ 的 R_{20} 性能的平均误差

Table 7　Bias in performance R_{20} of model CBS+

统计值	十折交叉验证	五折交叉验证	采样余量方法	后采样方法
平均误差	0.103 3	0.090 4	0.086 1	0.085 6
胜平负计数	74/0/21	77/0/18	56/0/39	
检验 p 值	$<$0.001	$<$0.001	0.018	

表 7 的统计结果与图 8 一致。十折交叉验证的误差最大，平均为 0.103 3；后采样的误差最小，平均为 0.085 6，相比十折交叉验证，误差减少大约 17.1%。在一对一胜平负计数中，后采样方法和其他 3 种方法相比，分别在 74、77 和 56 个数据集上胜出，都超过了半数。统计检验的 p 值在矫正前和矫正后全部小于 0.05，有统计显著性。

图 9 展示了对 P_{opt} 性能估计误差的箱图。可以看到，在 PROMISE 和 JiT 数据集[4]上，4 种方法的估计误差仍然符合前面结果的的规律；在 AEEEM 数据集中 4 种方法差别很小；在 JiT 数据集[25]中，后采样和采样余量方法和中位数高度接近。在 JiT 数据集[25]和 JavaScript 数据集中，十折交叉验证误差的上四分位数更高。总的来说，十折交叉验证的误差更大，其他 3 种方法则相差不大，这可能是因为 CBS+ 模型在性能指标 P_{opt} 上的误差更高更不稳定造成的。对比同样是 P_{opt} 误差的图 7 和图 9 可以发现，图 9 的误差要比图 7 高，同时箱体更长。更高的误差，意味着和偏差相关的误差在总体误差中所占比例相对减小，结果差异不再显著。此时 4 种方法的结果相近，表 8 中的统计量证明了这一点。

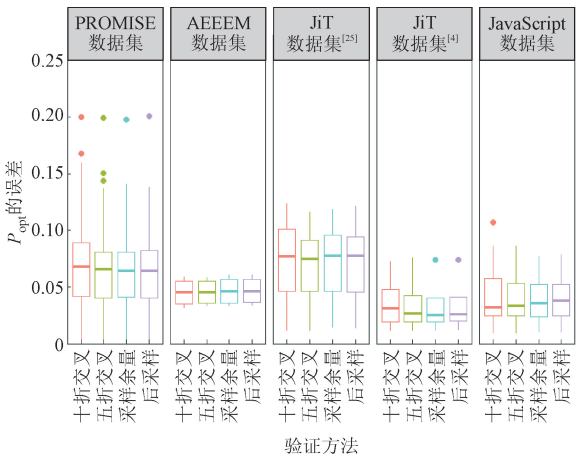

图 9　模型 CBS+ 的 P_{opt} 性能的误差

Fig. 9　Bias in performance P_{opt} of model CBS+

表 8　模型 CBS+ 的 P_{opt} 性能的平均误差

Table 8　Mean Bias in performance P_{opt} of model CBS+

统计值	十折交叉验证	五折交叉验证	采样余量方法	后采样方法
平均误差	0.062 9	0.060 2	0.059 7	0.059 8
胜平负计数	51/0/44	46/0/49	47/0/48	
检验 p 值	0.028	0.775	0.785	

从表 8 可以观察到，十折交叉验证的误差最大，p 值在矫正前后都小于 0.05，有统计显著性，说明十折交叉验证的误差比后采样方法的误差大。后采样方法和其他两种方法相比 p 值较大，没有明显差异。后采样方法和十折交叉验证相比，均值误差减少约 4.9%。总的来说，偏差最大的十折交叉验证，估计误差仍然最大。

4　结　论

工作量感知缺陷预测的性能会受到工作量数值分布的影响，而传统的测试评估方法在划分或者产生测试集时并未考虑到测试的工作量分布，因此最常用的十折交叉验证方法划分的测试集往往和原始数据集在工作量的数值分布差异较大。

本文假设，如果测试集工作量的偏度与原始数据集的工作量的偏度相差很大，那么在测试集上对模型排序性能的测试评估将会不准确。为了验证这一点，设计了无监督框架实验和有监督框架实验，用来评测估计基线模型 ManualUP 和 CBS+ 的预测性能，以计算测试集工作量的偏差和对模型性能的估计误差。实验比较了 4 种测试评估方法，包括十折交叉验证、五折交叉验证、采样余量方法和本文提出的后采样方法。实验结果发现，最常用的十折交叉验证的测试集偏差最大，同时估计误差也是最大的，而采样余量方法和后采样方法的测试集偏差较小，综合来看估计误差也是更小的。这个结果验证了本文的假设。本文提出的后采样方法和十折交叉验证相比，误差减少约 4.9%～26.9%；和采样余量方法相比，不会产生无效数据集。

本文结果对实际应用工作量感知软件缺陷预测有以下 3 点贡献：①实验验证了最常用的十折交叉验证在评估主要基线模型 ManualUP 和 CBS+ 的工作量感知预测性能时，估计误差较大；②发现文献[12]推荐的采样余量方法会生成无效测试集；③所提后采样方法能够更准确地估计基线模型的性能，可作为进一步比较其他工作量感知软件缺陷预测模型的基础。

参考文献：

[1] HAMILL M, GOSEVA-POPSTOJANOVA K. Analyzing and predicting effort associated with finding and fixing software faults [J]. Information and Software Technology, 2017, 87: 1-18.

[2] LESSMANN S, BAESENS B, MUES C, et al. Benchmarking classification models for software defect prediction: a proposed framework and novel findings [J]. IEEE Transactions on Software Engineering, 2008, 34(4): 485-496.

[3] YU Xiao, RAO Jiqing, LIU Lei, et al. Improving effort-aware defect prediction by directly learning to rank software modules [J]. Information and Software Technology, 2024, 165: 107250.

[4] KAMEI Y, SHIHAB E, ADAMS B, et al. A large-scale empirical study of just-in-time quality assurance [J]. IEEE Transactions on Software Engineering, 2013, 39(6): 757-773.

[5] YANG Xinli, LO D, XIA Xin, et al. TLEL: a two-layer ensemble learning approach for just-in-time defect prediction [J]. Information and Software Technology, 2017, 87: 206-220.

[6] 刘旭同，郭肇强，刘释然，等. 软件缺陷预测模型间的比较实验:问题、进展与挑战 [J]. 软件学报，2023，34(2)：582-624.
LIU Xutong, GUO Zhaoqiang, LIU Shiran, et al. Comparing software defect prediction models: research problem, progress, and challenges [J]. Journal of Software, 2023, 34(2): 582-624.

[7] MENDE T, KOSCHKE R. Effort-aware defect prediction models [C]//2010 14th European Conference on Software Maintenance and Reengineering. Piscataway, NJ, USA: IEEE, 2010: 107-116.

[8] YANG Yibiao, ZHOU Yuming, LIU Jinping, et al. Effort-aware just-in-time defect prediction: simple unsupervised models could be better than supervised models [C]//Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. New York, USA: Association for Computing Machinery, 2016: 157-168.

[9] ZHOU Yuming, YANG Yibiao, LU Hongmin, et al. How far we have progressed in the journey?: an examination of cross-project defect prediction [J]. ACM Transactions on Software Engineering and Methodology, 2018, 27(1): 1-51.

[10] NI Chao, XIA Xin, LO D, et al. Revisiting supervised and unsupervised methods for effort-aware cross-project defect prediction [J]. IEEE Transactions on Software Engineering, 2022, 48(3): 786-802.

[11] HUANG Qiao, XIA Xin, LO D. Revisiting supervised and unsupervised models for effort-aware just-in-time defect prediction [J]. Empirical Software Engineering, 2019, 24(5): 2823-2862.

[12] 蔡亮，范元瑞，鄢萌，等. 即时软件缺陷预测研究进展 [J]. 软件学报，2019，30(5)：1288-1307.
CAI Liang, FAN Yuanrui, YAN Meng, et al. Just-in-time software defect prediction: literature review [J]. Journal of Software, 2019, 30(5): 1288-1307.

[13] TANTITHAMTHAVORN C, MCINTOSH S, HASSAN A E, et al. An empirical comparison of model validation techniques for defect prediction models [J]. IEEE Transactions on Software Engineering, 2017, 43(1): 1-18.

[14] GUO Yuchen, SHEPPERD M, LI Ning. Poster: bridging effort-aware prediction and strong classification-a just-in-time software defect prediction study [C]//2018 IEEE/ACM40th International Conference on Software Engineering: Companion (ICSE-Companion). Piscataway, NJ, USA: IEEE, 2018: 325-326.

[15] SHIHAB E, KAMEI Y, ADAMS B, et al. Is lines of code a good measure of effort in effort-aware models? [J]. Information and Software Technology, 2013, 55(11): 1981-1993.

[16] NI Chao, XIA Xin, LO D, et al. Just-in-time defect prediction on JavaScript projects: a replication study [J]. ACM Transactions on Software Engineering and Methodology, 2022, 31(4): 76.

[17] LI Fuyang, YANG Peixin, KEUNG J W, et al. Revisiting 'revisiting supervised methods for effort-aware cross-project defect prediction' [J]. IET Software, 2023, 17(4): 472-495.

[18] KING R S, JULSTROM B. Applied statistics using the computer [M]. Palo Alto, CA, USA: Distributed by Mayfield Pub. Co., 1982.

[19] GONG Lina, ZHANG Haoxiang, ZHANG Jingxuan, et al. A comprehensive investigation of the impact of class overlap on software defect prediction [J]. IEEE Transactions on Software Engineering, 2023, 49(4): 2440-2458.

[20] ZHANG Tanghaoan, YU Yue, MAO Xinjun, et al. FENSE: a feature-based ensemble modeling approach to cross-project just-in-time defect prediction [J]. Empirical Software Engineering, 2022, 27(7): 162.

[21] PEDREGOSA F, VAROQUAUX G, GRAMFORT A, et al. Scikit-learn: machine learning in python [J]. The Journal of Machine Learning Research, 2011,

12：2825-2830.

[22] SONG Qinbao，GUO Yuchen，SHEPPERD M. A comprehensive investigation of the role of imbalanced learning for software defect prediction [J]. IEEE Transactions on Software Engineering，2019，45(12)：1253-1269.

[23] BOETTICHER G. The PROMISE repository of empirical software engineering data [EB/OL]. [2024-01-01]. http://promisedata.org/repository.

[24] D'AMBROS M，LANZA M，ROBBES R. Evaluating defect prediction approaches：a benchmark and an extensive comparison [J]. Empirical Software Engineering，2012，17(4)：531-577.

[25] TIAN Yuli，LI Ning，TIAN J，et al. How well just-in-time defect prediction techniques enhance software reliability? [C]//2020 IEEE 20th International Conference on Software Quality，Reliability and Security (QRS). Piscataway，NJ，USA：IEEE，2020：212-221.

[26] HOLLANDER M，WOLFE D A. Nonparametric statistical methods [M]. New York：Wiley，1973：503.

[27] BENJAMINI Y，HOCHBERG Y. Controlling the false discovery rate：a practical and powerful approach to multiple testing [J]. Journal of the Royal Statistical Society：Series B　Methodological，1995，57(1)：289-300.

（编辑　亢列梅）