FSDNP:针对软件缺陷数预测的特征选择方法

李叶飞[1,2],官国飞[2],葛崇慧[2],陈 翔[3],倪 超[1],钱柱中[1]
1.南京大学 计算机科学与技术系,南京 210023
2.江苏方天电力技术有限公司,南京 210000
3.南通大学 计算机科学与技术学院,江苏 南通 226019

摘 要:软件缺陷预测先前的研究工作主要关注软件缺陷分类问题,即判断一个软件模块是否含有缺陷。如何量化一个软件模块中含有软件缺陷的数量问题还未被很好地研究。针对该问题,提出了一种两阶段的软件模块缺陷数预测特征选择方法FSDNP:特征聚类阶段和特征选择阶段。在特征聚类阶段中,使用基于密度峰聚类的算法将高度相关的特征进行聚类;在特征选择阶段,设计了三种启发式的排序策略从簇中删除冗余的和无关的特征。在PROMISE数据集上,使用平均错误率和平均相对错误率指标,与6个经典的方法进行了比较。实验结果表明,FSDNP能够有效移除冗余的和无关的特征,构建高效的软件缺陷数预测模型。

关键词:软件质量保障;软件缺陷数预测;特征选择;实证研究
文献标志码:A **中图分类号:**TP311.5 doi:10.3778/j.issn.1002-8331.1811-0103

李叶飞,官国飞,葛崇慧,等.FSDNP:针对软件缺陷数预测的特征选择方法.计算机工程与应用,2019,55(14):61-68.
LI Yefei, GUAN Guofei, GE Chonghui, et al. FSDNP: feature selection method for software defect number prediction. Computer Engineering and Applications, 2019, 55(14):61-68.

FSDNP:Feature Selection Method for Software Defect Number Prediction

LI Yefei[1,2], GUAN Guofei[2], GE Chonghui[2], CHEN Xiang[3], NI Chao[1], QIAN Zhuzhong[1]
1.Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China
2.Jiangsu Frontier Electric Technology Co., Ltd., Nanjing 210000, China
3.College of Computer Science & Technology, Nantong University, Nantong, Jiangsu 226019, China

Abstract: Previous works in software defect prediction mainly focused on classifying software modules as defect-prone or not. How to quantify the number of defects in a software module has rarely been investigated. To address such issue, the paper proposes a two-stage feature selection approach for software defect number prediction FSDNP: feature clustering phase and feature selection phase. The feature clustering phase clusters highly correlated features using a density-based clustering method, and the feature selection phase removes irrelevant and redundant features from each cluster using three heuristic ranking strategies. FSDNP compares with six state-of-the-art baseline approaches using average absolute error and average relative error on PROMISE dataset. The results show FSDNP can effectively remove irrelevant and redundant features and build effective software defect number prediction model.

Key words: software quality assurance; software defect number prediction; feature selection; empirical studies

基金项目:国家自然科学基金(No.61472181);江苏省自然科学基金(No.BK20151392)。

作者简介:李叶飞(1979—),男,工程师,研究领域为计算机应用、信息安全方面研究研究,E-mail:imliyf@163.com;官国飞(1978—),男,高级工程师,研究领域为电力信息技术和信息安全;葛崇慧(1985—),男,工程师,研究领域为电力信息技术和信息安全;陈翔(1980—),男,博士,研究领域为智能化软件运维、软件缺陷分析、机器学习;倪超(1990—),通讯作者,男,博士,研究领域为智能化软件运维、软件缺陷分析、机器学习;钱柱中(1980—),男,博士,副教授,研究领域为机器学习、智能化运维、云计算。

收稿日期:2018-11-09 **修回日期:**2019-01-07 **文章编号:**1002-8331(2019)14-0061-08
CNKI **网络出版:**2019-03-18, http://kns.cnki.net/kcms/detail/11.2127.TP.20190315.0904.006.html

1 引言

软件缺陷(software defect)产生于开发人员的编码过程,软件需求理解不正确、软件开发过程不合理或开发人员的经验不足,均有可能引入软件缺陷。含有缺陷的软件在部署后可能会发生运行故障,产生意料之外的结果或行为,严重的时候会给企业造成巨大的经济损失,甚至会威胁到人们的生命安全。在软件项目的开发生命周期中,检测出内在缺陷的时间越晚,修复该缺陷的代价也越高。尤其在软件发布后,检测和修复缺陷的代价将大幅度增加。因此,项目主管借助智能化软件质量保障新技术,希望能够在软件部署前尽可能多地检测出内在缺陷,从而避免因软件运行故障而造成的巨大损失。但是若关注所有的程序模块会消耗大量的人力物力,因此,软件质量保障部门主管希望能够预先识别出潜在缺陷程序模块,并随后对其分配足够的测试资源。

软件缺陷预测(Software Defect Prediction,SDP)[1-7]技术是避免软件运行故障的一种可行方法,可以有效提高智能化软件运维的效率。软件缺陷预测通过挖掘软件仓库(例如,版本控制系统和缺陷跟踪系统),分析软件历史开发数据以及已发现的运行故障等缺陷信息,借助机器学习等智能化方法来构建软件缺陷预测模型以预测新的软件模块是否含有缺陷。因此,软件缺陷预测能够在软件测试阶段识别出软件项目内的潜在软件缺陷模块,从而可以最优化地、最经济地分配软件测试资源。

目前,大量研究工作被提出用于软件缺陷预测,避免软件在运行阶段发生故障,减少损失。这些方法通常使用多种多样的人工智能技术,如统计学习方法或者机器学习方法。这些工作中,绝大部分都在关注能否有效地判断一个软件模块是否含有缺陷会导致运行故障。这意味着之前的研究工作将软件模块中连续性的缺陷数据进行了离散化处理,并将其转化为二分类问题,即有缺陷或者没有缺陷[8]。但是,实际应用中导致运行故障的缺陷在软件模块中的重要性并不一样。例如,在一个项目中,有三个软件模块M1,M2和M3。这三个模块分别含有2,10和3个缺陷可能导致运行故障。显然,这三个软件模块在软件质量管理过程中应具有不同的重要性。而在软件缺陷预测中,对数据进行简单的离散化处理一方面会导致重要信息的损失,另一方面对于含有不同缺陷数量的软件模块将被赋予相同的重要性。因此,有效地预测出软件各模块中可能导致运行故障的缺陷数量将有助于对软件各模块的重要性排序,从而可以将有限的软件测试资源分配给含有更多缺陷的软件模块,优化软件运维的效率。

之前的研究工作主要关注对软件模块进行有、无缺陷会导致运行故障的分类问题,只有很少的一些工作提出并且评估了基于回归模型的软件缺陷数预测的方法[9-11],这类方法将预测出软件模块中有多少个缺陷会导致运行故障。但是,软件缺陷预测数据集中存在维度灾难问题。其中,冗余的和无关的特征会导致较差的预测性能、较高的模型复杂度和较多的训练时间[12],从而降低智能化运维的效率。而特征选择方法是一种能有效地减轻维度灾难问题并提高预测性能的方法[13-20]。针对该问题,为了有效地移除无关特征和冗余特征,本文提出了一个包含两阶段软件缺陷数预测的方法FSDNP(Feature selection for Software Defect Number Prediction):特征选择阶段和模型构建阶段。具体而言,在特征聚类阶段中,使用基于密度峰聚类的算法将高度相关的特征进行聚类;在特征选择阶段,设计了三种启发式的排序策略从簇中删除冗余的和无关的特征。为了验证FSDNP方法的有效性,在15个真实软件项目上进行了经验性的研究,并与6个经典的方法进行了比较。此外,使用平均绝对错误率(Average Absolute Error,AAE)和平均相对错误率(Average Relative Error,ARE)指标来评估FSDNP的性能。实验结果表明,在绝大多数情况下,FSDNP能够获得更好的性能。这表明FSDNP能够有效移除冗余的和无关的特征,并在构建软件缺陷数预测模型方面具有一定的优越性。

本文的主要贡献可总结如下:(1)在软件缺陷数预测方法上,首次提出了基于特征选择的两阶段方法:FSDNP,该方法能够有效地移除无关的特征和冗余的特征。(2)基于真实软件项目对FSDNP方法的有效性进行了经验性研究。研究结果表明FSDNP方法相对经典的软件缺陷数预测方法能够取得实质性的提高。

2 研究背景和相关工作

2.1 研究背景

软件缺陷预测是智能化软件工程领域中的一个研究热点,其通过挖掘软件历史仓库来构建缺陷预测模型,从而避免软件在运行时发生故障,其框架如图1所示。缺陷预测可以在软件发布之前尽可能多地预测出潜在缺陷程序模块,以便于合理分配测试资源,从而最终提高软件质量。其主要包括两个阶段:模型构建阶段和模型应用阶段。模型构建阶段主要包括如下三个步骤:(1)挖掘软件历史仓库。目前可供挖掘与分析的软件历史仓库包括项目所处的版本控制系统(例如CVS、SVN或Git等),缺陷跟踪系统(例如Bugzilla、Mantis、Jira或Trac等)或团队开发人员间相互发送的电子邮件等。程序模块的粒度根据实际应用场景的需要,可以设置为包、文件、类、函数或代码修改等。(2)程序模块的度量和标记。通过分析软件代码复杂度或开发过程特征,可以设计出与软件缺陷存在相关性的度量元(metrics)[21]。通过这些度量元可以对程序模块进行度量,随后通过分析缺陷报告可以完成对这些程序模块中含有导致运行故障的缺陷数的标记。(3)模型的构建。基于搜集的缺

陷预测数据集，首先进行必要的数据预处理（例如特征选择、数据取值归一化、噪音移除等）[22-23]，随后可以基于特定的机器学习方法（例如，线性回归、决策树回归等）完成模型的构建。而在模型应用阶段，当面对新的程序模块时，在完成对该模块的软件度量后，基于已构造出的缺陷数预测模型和具体度量元取值，可以完成对该模块的预测，即预测软件模块中可能多少个缺陷会导致软件运行故障。

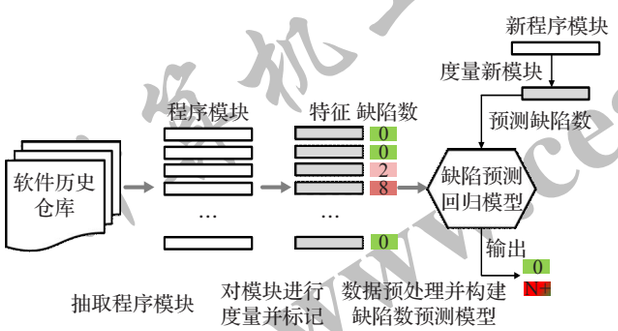

图1　软件缺陷数预测框架

2.2　相关工作

在软件质量管理过程中，预测软件模块中含有导致软件运行故障的缺陷数目可以有效地对这些模块进行排序。含有较多缺陷的软件模型排名靠前，不含或含有少量缺陷的软件模块排名靠后。通过这种方法，可以将更多的测试资源分配给排名靠前的软件模块，因为这些模块可能含有更多的缺陷。因而，有限的测试资源被最优化的、最经济的方式用于检测和修复更多的软件缺陷，避免软件在运行中发生故障，造成巨大损失。

之前的研究工作主要探究基于回归模型来解决软件缺陷数预测问题。Graves等人[9]采用了通用的线性回归（Linear Regression，LR）方法，并且在大型的电信系统项目上进行了经验性研究。他们发现，在软件模块年龄、对软件模型的改变以及改变的年龄这三个因素之间存在重要的相关性。Ostrand等人[10]使用负二项回归（Negative Binomial Regression，NBR）方法进行软件缺陷数预测的研究。Wang和Zhang[24]提出名为BugState方法，该方法主要基于缺陷状态的转移模型来解决软件缺陷数预测问题。Janes等人[25]则在五个实时电信系统项目上对三个方法（例如：负二项回归，零膨胀负二项回归和泊松回归）进行了研究。他们发现，零膨胀负二项回归在这三个方法中表现最好。随后，Gao和Khoshgoftaar[26]在两大商业软件项目上进一步进行了经验性研究，并且他们的研究结果再一次证实了零膨胀负二项回归能够取得更好的性能。

Chen等人[27]对六个回归方法进行了经验性的研究。他们发现在项目内缺陷数据预测和项目间缺陷数预测这两种场景下使用决策树回归（Decision Tree Regression，DTR）模型能够获得最好的性能。Yu等人[28]则同时考虑了类间不平衡问题和集成学习方法。在处理类不平衡时，他们采用两种经典的重采样方法：SMOTEND和RUSND。在处理集成预测模型时，他们采用被广泛使用的AdaBoost.R2方法。通过实证研究发现，两种混合的方法（例如：SMOTENDBoost和RUSNDBoost）能够取得更好的性能。

Rathore和Kumar[29]在两种不同的研究场景下（例如：版本内缺陷数预测和跨版本缺陷数预测）探究决策树回归模型的预测能力。他们[30]也比较了六种用于软件缺陷数预测的方法，例如，遗传编程（genetic programming）、多层感知器（multi-layer perceptron）、线性回归（linear regression）、决策树回归（decision tree regression）、零膨胀泊松回归（zero-inflated Poisson regression）和负二项回归（negative binomial regression）。最近，他们[31-32]又深入研究了集成学习方法在软件缺陷数预测方面的性能表现。

基于以上分析，以前的研究工作都是从构建软件缺陷数模型角度出发，使用不同的模型进行经验性实验。不同于之前的研究工作，本文从数据预处理角度出发，提出一种两阶段的缺陷数预测方法FSDNP：特征聚类和选择阶段和模型构建阶段。在特征聚类阶段中，使用基于密度峰聚类的算法将高度相关的特征进行聚类；在特征选择阶段，从簇中删除冗余的和无关的特征。

3　FSDNP

为了减少数据集中无关特征和冗余特征对软件缺陷数模型性能的影响，提出一个基于密度峰聚类的特征选择方法FSDNP，其流程如图2所示。

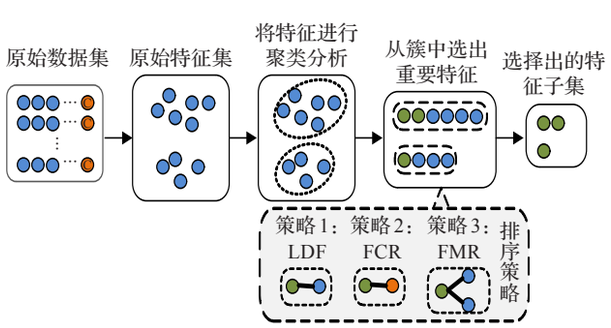

图2　FSDNP方法的总体流程

具体而言，FSDNP包含两个阶段：特征聚类阶段和特征选择阶段。在前一个阶段中，使用基于密度峰的聚类方法[33]将原始数据集中的特征集合根据特征之前的相关性划分到多个簇中。高度相关的特征将被划分到相同的簇中，而彼此不相关的特征将被划分到不同的簇中。在后一个阶段中，使用不同的排序策略对每一个簇中的所有特征进行排序，然后根据簇的大小占原始特征数的比例从中挑选出排名靠前的特征。

经过FSDNP处理之后，原始的数据集合将被过滤、

清洗,从而得到高质量特征,并在此基础之上构建回归模型。由于已有研究工作证实了决策数回归模型的出色性能[27,29],因此,在本文中,采用被广泛使用的决策树回归模型作为缺省模型。需要说明的是,经过FSDNP方法处理之后的数据集可以适用于任何基于回归模型的方法以构建软件缺陷数预测器。

3.1 特征聚类

在这一阶段中,使用DPC方法[33]对除类标以外的原始特征集合进行聚类,并且DPC算法不需要事先指定簇的个数。在DPC算法中,一个簇中心应同时具备两大特征:高密度和大距离。其中,高密度要求一个簇中心应被很多较低密度的特征包围,而大距离要求一个簇中心应当远离比其密度更大的簇中心。具体操作涉及如下三个步骤。

第一步,对于每一个特征 f_i,需要计算其自身密度 ρ_i 和自身距离 δ_i。如果用 d_{ij} 表示特征 f_i 和特征 f_j 之间的欧式距离,则 d_{ij} 计算如下:

$$d_{ij}=\sqrt{\sum_{k=1}^{m}(x_{i_k}-x_{j_k})^2} \tag{1}$$

其中,m 表示数据集中含有的实例个数,X 则表示一个实例对应的特征值。特征 f_i 的自身密度 ρ_i 表示与其距离小于指定距离的所有特征个数,其定义如下:

$$\rho_i=\sum_j \chi(d_{ij}-d_c) \tag{2}$$

其中 χ 是计数函数,即如果 $x<0$,那么 $\chi(x)=1$;否则,$\chi(x)=0$。d_c 为截断距离,其必须要确保每一个特征至少含有原始特征总数的2%个特征作为其邻居。

特征 f_i 的自身距离 δ_i 是通过与特征 f_j 的欧式距离 d_{ij} 计算而来。特征 f_j 必须具备如下两点要求:(1)与特征 f_i 的欧式距离最近;(2)自身密度 ρ_j 大于 ρ_i。其定义如下:

$$\delta_i=\min_{j:\rho_j>\rho_i}(d_{ij}) \tag{3}$$

对于具有最大自身密度的特征而言,其自身距离是最大可能的欧式距离。其定义如下:

$$\delta_i=\max_j(d_{ij}),\ \text{如果}f_i\ \text{有最大的}\ \rho_i \tag{4}$$

第二步,使用每个特征已经计算好的自身密度和自身距离来确定每一个簇的簇中心。首先,对于特征 f_i,计算其可能成为簇中心的指标 γ_i 如下:

$$\gamma_i=z(\rho_i)\times z(\delta_i) \tag{5}$$

其中,$z(\cdot)$ 是 z -score 归一化函数。其次,根据每个特征的 γ_i 值进行降序排序,将其表示为 $\{\gamma\}_i^n$。其中,上标 n 表示除类标特征以外的特征总数。研究表明,$\{\gamma\}_i^n$ 服从幂律分布(power law distribution)[34]。因此,基于 $\{\gamma\}_i^n$,可以选取其单拐点作为截断点 γ_c。这表明,对于特征的簇中心指标 γ_i 大于 γ_c 的所有特征将被选取作为聚簇中心。

第三步,将所有非簇中心特征进行簇的指派。假设有 k 中心被指定,那么将所有剩下的非簇中心特征依次指派到比其自身密度大并且距离最近的簇中心去。通过使用DPC,所有特征被指派到各个簇中仅仅需要迭代一轮,并且无需事先指定簇的个数。

3.2 特征选择

本文中设计了3种启发式的排序策略用于对簇中的特征进行降序排序。由于每个簇含有特征的个数不同,并且簇的个数也不是事先指定的,因此最终从每个簇中挑选出来的特征个数应当与簇的大小成比例,即应从较大的簇中挑选较多的特征。各种排序策略定义如下:

策略1 特征自身密度策略(Self Density of Features,SDF)。在这种策略中,从各个簇中被选取出来的特征是根据其自身的密度来排序的。特征的自身密度意味着包围该特征的邻居特征个数。因此,排名靠前的特征将被更多的邻居特征包围。这在一定程序上反映了该特征的重要性和代表性。

策略2 特征与类标相关性策略(Feature-Class Relevance,FCR)。该策略的基本假设是,如果普通特征与类标特征之间具有强相关性,那么这些特征对构建模型具有重要意义。因此,这些特征是值得被挑选出来的。本文通过皮尔逊相关系数(Pearson Correlation Coefficient)来计算特征与缺陷数类别之间的相关性。对于特征 f_i 与类特征 C 之间的皮尔逊相关系数为例,假设训练集中共有 M 个实例,其中 $f_{i,m}$ 表示特征 f_i 在第 m 个实例中对应的取值,C_m 表示第 m 个实例对应的缺陷数,$\bar{f}_i$ 表示特征 f_i 在所有实例中的均值,而 $\bar{C}$ 表示缺陷数特征 C 在所有实例中的均值。其计算公式为:

$$PR_{f_i,C}=\frac{\sum_{m=1}^{M}(f_{i,m}-\bar{f}_i)(C_i-\bar{C})}{\sqrt{\sum_{m=1}^{M}(f_{i,m}-\bar{f}_i)^2}\sqrt{\sum_{m=1}^{M}(C_i-\bar{C})^2}} \tag{6}$$

其中,$PR_{f_i,C}$ 可以有效计算出特征 f_i 与类特征 C 之间的相关性,其取值区间为 $[-1,1]$,取值越大,表示特征与类间的相关性越高。

策略3 簇中最具代表性的特征策略(Feature Maximization Representation,FMR)。该策略在每个簇中,计算各个特征与其他所有特征的特征相关性之和,根据相关性之和进行降序排序,选择排名靠前的特征作为最终选择的特征。特征与特征之间的相关性计算仍采用皮尔逊相关系数。

3.3 时间复杂度分析

算法1详细描述了在软件缺陷数预测场景下本文

提出的特征选择方法。其输入为训练数据集 $D^{m\times n}$，m 和 n 分别表示数据集中含有的实例个数和原始特征个数(除类标特征外)，n_t 表示最终需要保留的特征个数，$strat$ 表示在特征排序时使用的排序策略。因此，其时间复杂度为 $O(n_t^2\times m)$。

在算法1中，第1~8行描述了特征聚类阶段的操作步骤，第9~15行描述了特征选择阶段的操作步骤。在第14行中，算法根据每个簇的大小从对应的簇中等比例地选择出需要保留的特征。例如，假设训练数据集含有 n 个特征，并且希望最终保留 n_t 个特征。那么，对于一个簇 C，其中含有 $|C|$ 个特征，算法将从中选择前 $\left\lceil\frac{|C|\times n_t}{n}\right\rceil$ 个特征。

算法1 特征选择方法FSDNP

输入：

$D^{m\times n}$：训练数据集；

n_t：需要保留的特征个数；

$strat$：排序策略；

输出：

$Subset$：最终选择出来的特征子集。

/******特征聚类阶段******/

1. for D 中的每一个特征向量 f_i do
2. 计算特征的自身密度 ρ_i 和自身距离 δ_i；
3. 计算特征的簇中心指标 $\gamma_i=\rho_i\cdot\delta_i$；
4. end for
5. 对 $\{\gamma\}_i^n$ 降序排序；
6. 基于排序后的 $\{\gamma\}_i^n$ 计算截断点 γ_c；
7. 将大于 γ_c 的特征作为簇中心；
8. 将小于 γ_c 的特征指派到各个簇中；

/******特征选择阶段******/

9. for 簇集合中的每个簇 C do
10. 根据排序策略 $strat$ 对簇中的特征进行排序；
11. end for
12. $Subset\leftarrow\phi$；
13. for 簇集合中的每个簇 C do
14. 选取簇 C 中排名靠前的 $\left\lceil\frac{|C|\times n_t}{n}\right\rceil$ 个特征，将其加入到 $Subset$；
15. end for
16. return $Subset$.

4　实证研究

本文在实证研究中，重点回答如下两个问题。

(1)RQ1：FSDNP方法是否优于经典的软件缺陷数预测方法？

(2)RQ2：不同的排序策略如何影响FSDNP的性能？

4.1　评测对象

在本文的实证研究中，使用了15个来自PROMISE数据集的项目。这些项目在之前的研究工作[27-28,31-32,35-36]中被广泛使用并可从PROMISE(http://openscience.us/repo/)免费获取。

表1详细列出了这些项目的统计特征，包括项目名称、模块数、缺陷数、缺陷比例和软件模块中含有的最大缺陷数。每个项目都含有很多类级别的特征，这些特征都是基于代码复杂度和面向对象特性进行设计的。

表1　数据集特征

项目名称	模块数	缺陷数	缺陷比例/%	最大缺陷数
ant-1.3	125	33	16.00	3
ant-1.4	178	47	22.47	3
ant-1.5	293	35	10.92	2
ant-1.6	351	184	26.21	10
ant-1.7	745	338	22.28	10
ivy-1.1	111	233	56.76	36
ivy-1.4	241	18	6.64	3
ivy-2.0	352	56	11.36	3
jedit-3.2	272	382	33.09	45
jedit-4.0	306	226	24.51	23
jedit-4.1	312	217	25.32	17
jedit-4.2	367	106	13.08	10
xalan-2.4	723	156	15.21	7
xalan-2.5	803	531	48.19	9
xalan-2.6	885	625	46.44	9

4.2　评测指标

为了评估FSDNP在软件缺陷数预测中的性能，本文使用平均绝对误差(Average Absolute Error，AAE)和平均相对误差(Average Relative Error，ARE)两个性能评价指标。

平均绝对误差(AAE)。AAE测量一组预测数据中所有误差的平均大小。它表明了预测结果和真实结果之间的差距。其定义如下：

$$AAE=(1/n)\sum_{i=1}^{n}\left|\bar{Y}_i-Y_i\right| \tag{7}$$

平均相对误差(ARE)。ARE表示当整体目标项目被评估时，绝对误差有多大。其定义如下：

$$ARE=(1/n)\sum_{i=1}^{n}\left|\bar{Y}_i-Y_i\right|/(Y_i+1) \tag{8}$$

在公式(7)和公式(8)中，$\bar{Y}_i$ 表示模型在一个特定软件模块上输出的预测结果(例如：软件模块中含有的缺陷数)，Y_i 表示该模块真实含有的缺陷数，n 表示模块数。

对于评价指标ARE而言，由于 Y_i 可能为零，即模块不含有缺陷。为了缓解这种非法定义情况，本文如同文献[26]一样，在分母中对 Y_i 进行了加一操作，以使得该公式在计算时始终有意义。值得注意的是，这两个评价指标越低，表明其对应的预测模型性能越好[37]。

为了进一步比较两个方法之间的性能，本文使用

Win/Draw/Loss分析。在特定评价指标上,方法1 vs 方法2的Win/Draw/Loss结果分析共有三种情况:即方法1的性能好于、相似或者差于方法2的性能的次数。

4.3 实验设置

在本文的经验性实验中,为了评估FSDNP方法的有效性,本文将FSDNP方法与6个经典的基准方法进行比较(例如LR、NNR、SVR、DTR、BRR和DBR)。为了减少在实现缺陷预测模型中人工引入的错误,同时也为了方便重现本文的实验,采用被广泛使用的机器学习包sklearn提供的方法。sklearn包含这6个经典方法。需要注意的是,sklearn中包含的这6个方法的超参数均使用缺省参数设置,即没有对这些方法进行额外的优化操作。这6个基准方法的简要介绍如下:

(1)线性回归(Linear Regression,LR):它通常被用于求解一个或多个自变量和一个因变量之间的线性关系的最小二乘函数。

(2)最近邻回归(Nearest Neighbors Regression,NNR):该方法是基于K-近邻算法。通过其最近邻居的加权平均来计算实例的回归值。此外,权重的大小与实例及其邻居之间距离的倒数成正比。

(3)支持向量回归(Support Vector Regression,SVR):该方法是广泛使用的支持向量机的扩展方法。由于构建SVR模型的成本函数会忽略与模型预测结果接近的训练数据,所以SVR方法仅仅依赖于原始训练数据的部分数据集。

(4)决策树回归(Decision Tree Regression,DTR):该方法学习简单的决策规则来近似拟合给定的训练数据,然后预测目标变量。

(5)贝叶斯岭回归(Bayesian Ridge Regression,BRR):该方法类似于经典的岭回归方法。这种方法的超参数是通过先验概率引入的,然后基于概率模型的最大化边缘对数似然去评估。

(6)梯度提升回归(Gradient Boosting Regression,GBR):该方法是弱预测模型的一种集成版本。为了提高单个模型的预测精度,通常将多个基模型与给定的学习算法相结合。

FSDNP方法首先对特征进行聚类而无需事先指定簇的个数。FSDNP选择的特征个数为$40\%\times M$,其中M为除类别特征外的所有特征总数。在回归模型选择上,本文使用软件缺陷数预测领域中被广泛使用的Decision Tree Regression[27,29-30]作为默认回归模型。

此外,在实验中,采用10折交叉检验评估6种基准方法和FSDNP。具体而言,将原始的数据集等分成10份,使用其中的9份作为训练数据,剩下的1份作为测试数据。如此反复10次,确保每份数据都被恰好作为测试集1次。为了尽可能避免因数据集划分而引起的偏差,本文将10折交叉检验随机进行了10 次,并将平均性能作为模型的最终性能。

本文实验在以下配置的台式机上运行:操作系统:Windows 7,64位;CPU:Intel® Core™ i5-4590 CPU @ 3.30 GHz×2;内存:16 GB。

4.4 结果分析

4.4.1 针对RQ1的结果分析

本文在PROMISE数据集上对FSDNP方法的整体性能进行了实证研究,使用决策树回归作为基分类器,选择的特征个数为$40\%\times M$,特征排序策略为FCR,并将FSDNP方法与6种基准方法进行了比较。表2和表3分别给出了这些方法在AAE和ARE指标上取得的平均结果。

表2 基于AAE评测指标,不同软件缺陷数预测模型的平均性能

项目名称	SDNP	LR	NNR	SVR	DTR	BRR	GBR
ant-1.3	0.41	0.43	**0.36**	0.44	0.46	0.38	0.45
ant-1.4	**0.37**	0.40	0.38	0.42	0.42	0.41	0.41
ant-1.5	**0.16**	0.22	0.19	0.26	0.18	0.21	0.18
ant-1.6	**0.43**	0.58	0.50	0.68	0.55	0.55	0.51
ant-1.7	**0.45**	0.49	0.48	0.61	0.58	0.48	0.50
ivy-1.1	**0.09**	0.68	0.65	0.93	0.96	0.39	0.50
ivy-1.4	**0.11**	0.15	0.12	0.21	**0.11**	0.13	0.13
ivy-2.0	**0.21**	0.26	0.23	0.31	0.27	0.23	0.23
jedit-3.2	2.00	0.54	0.56	0.54	0.82	**0.46**	0.60
jedit-4.0	0.77	0.83	0.88	0.90	**0.02**	0.81	0.87
jedit-4.1	**0.66**	0.70	0.77	0.86	0.83	0.69	0.78
jedit-4.2	**0.32**	0.38	0.41	0.45	0.43	0.39	0.36
xalan-2.4	0.31	0.32	**0.30**	0.37	0.33	0.31	**0.30**
xalan-2.5	**0.52**	0.63	0.59	0.61	0.64	0.64	0.59
xalan-2.6	**0.51**	0.62	0.56	0.62	0.58	0.64	0.53
Win/Draw/Loss	—	13/0/2	12/0/3	14/0/1	12/1/2	12/1/2	13/0/2

表3 基于ARE评测指标,不同软件缺陷数预测模型的平均性能

项目名称	SDNP	LR	NNR	SVR	DTR	BRR	GBR
ant-1.3	0.27	0.31	**0.23**	0.29	0.31	0.26	0.32
ant-1.4	**0.27**	0.30	**0.27**	0.31	0.30	0.29	0.31
ant-1.5	**0.12**	0.18	0.14	0.21	0.13	0.17	0.13
ant-1.6	**0.28**	0.38	0.30	0.38	0.31	0.36	0.31
ant-1.7	**0.28**	0.32	0.29	0.34	0.34	0.31	0.30
ivy-1.1	**0.55**	0.74	0.61	0.60	0.70	0.64	0.61
ivy-1.4	0.10	0.10	**0.08**	0.17	**0.08**	0.09	0.10
ivy-2.0	**0.16**	0.19	0.18	0.22	0.20	0.18	0.18
jedit-3.2	**0.09**	0.97	0.75	0.48	0.83	0.91	0.81
jedit-4.0	**0.34**	0.52	0.47	0.38	0.58	0.52	0.51
jedit-4.1	**0.36**	0.42	0.41	0.39	0.42	0.41	0.44
jedit-4.2	**0.23**	0.26	0.26	0.26	0.27	0.27	**0.23**
xalan-2.4	0.24	0.22	**0.20**	0.26	0.22	0.21	**0.20**
xalan-2.5	**0.37**	0.42	0.38	0.38	0.39	0.43	0.39
xalan-2.6	**0.31**	0.40	0.34	0.37	0.34	0.41	0.33
Win/Draw/Los	—	13/1/1	11/1/3	15/0/0	13/0/2	12/0/3	12/2/1

在表2和表3中，第1列表示项目的名称，剩下的7列分别表示本文提出的FSDNP方法和6种经典的软件缺陷数预测算法在不同的性能指标下取得的平均结果（例如：AAE和ARE）。在每一个行中，最好的结果（即最小的结果）被加粗显示。最后一行表示FSDNP方法与各个基准方法之间的Win/Draw/Loss比较结果。需要说明的是，这些方法所取得结果是进行100次独立实验后的平均值。

在表2中，对于AAE而言，FSDNP在大多数情况下能够取得较好性能。具体而言，FSDNP在ant项目、ivy项目、jedit项目和xalan 项目上分别能取得4/5、3/3、2/4和2/3的胜算。其他6种基准算法，在15个项目中最好情况下只获得了2次最好性能（如：NNR和BRR）。

在表3中，对于ARE而言，FSDNP在大多数情况下也能够取得较好性能。具体而言，FSDNP在ant项目、ivy项目、jedit项目和xalan项目上分别能取得4/5、2/3、4/4和2/3的胜算。其他6种基准算法，在15个项目中最好情况下能够获得4次最好性能（如：NNR）。

基于上述分析，在软件缺陷数预测中，冗余特征和无关特征会影响软件缺陷数预测模型的性能。而FSDNP可以有效地移除这些特征，并提高模型的预测性能。

4.4.2　针对RQ2的结果分析

在FSDNP方法中，本文设计了3种启发式的排序策略（例如：SDF，FCR和FMR）用于从簇中选取代表性的特征。表4列出了当使用不同的排序策略，FSDNP在PROMISE 数据集上的性能表现。在表4中，第1列表示项目名称，第2~4列表示FSDNP在使用AAE评价指标下，不同排序策略的性能表现。第5~7列表示FSDNP在使用ARE评价指标下，不同排序策略的性能表现。需要说明的是，对于这3种排序策略，FSDNP选择原项目特征总数的40%。

表4　基于AAE和ARE评测指标，FSDNP使用不同排序策略的平均性能

项目名称	SDF		FCR		FMR	
	AAE	ARE	AAE	ARE	AAE	ARE
ant-1.3	0.42	0.29	0.41	0.27	**0.40**	**0.26**
ant-1.4	0.43	0.31	**0.37**	**0.27**	0.38	**0.27**
ant-1.5	**0.16**	0.14	**0.16**	**0.12**	0.19	**0.12**
ant-1.6	0.52	0.32	**0.43**	**0.28**	0.54	0.32
ant-1.7	0.49	0.29	**0.45**	**0.28**	0.53	0.32
ivy-1.1	**2.02**	0.93	2.09	**0.55**	2.25	1.00
ivy-1.4	0.14	**0.10**	**0.11**	**0.10**	0.14	0.17
ivy-2.0	0.24	0.17	**0.21**	**0.16**	0.24	0.19
jedit-3.2	2.00	1.09	2.00	1.09	**1.98**	**1.08**
jedit-4.0	0.85	0.43	**0.77**	**0.34**	0.87	0.45
jedit-4.1	0.99	0.60	**0.66**	**0.36**	0.86	0.56
jedit-4.2	0.41	0.25	**0.32**	0.23	0.41	**0.22**
xalan-2.4	0.36	0.25	**0.31**	**0.24**	0.35	**0.24**
xalan-2.5	0.67	0.44	**0.52**	**0.37**	0.63	0.42
xalan-2.6	**0.51**	0.35	**0.51**	**0.31**	0.57	0.36

基于AAE，FCR vs SDF和FCR vs FMR的Win/Draw/Loss分别为11/3/1和13/0/2。基于ARE，FCR vs SDF和FCR vs FMR的Win/Draw/Loss分别为13/2/0和9/3/3。总体而言，FCR排序策略在各个项目上表现性能较好，这可能是因为该策略挑选的特征是与类别特征高度相关。其次，FMR策略表现较好，最后是SDF排序策略。

4.5　有效性影响因素分析

这一节主要分析可能影响到本文实证研究结论有效性的影响因素。具体来说：（1）内部有效性主要涉及到可能影响到实验结果正确性的内部因素。最主要的有效性影响因素是实验代码的实现是否正确，为了减少重新实现各种基准方法过程中引入的人为因素的影响，使用了第三方提供的成熟框架，例如来自sklearn中的机器学习包。（2）外部有效性主要涉及到实验研究得到的结论是否具有一般性。为了确保实证研究结论的一般性，选择了软件缺陷预测问题研究中被广泛使用的PROMISE数据集，该数据集包含了15个具有一定代表性的开源项目，同时这些项目也覆盖了不同类型的应用领域，可以确保研究结论具有一定的代表性。（3）结论有效性主要涉及到使用的评测指标是否合理。重点考虑了AAE和ARE指标，该指标在软件缺陷数预测领域被广泛使用[27,30]，因此可以更好地评估模型的性能。

5　总结与展望

提出一种基于特征选择的软件缺陷数预测FSDNP，该方法主要包含特征聚类阶段和特征选择阶段。在特征聚类阶段中，使用基于密度峰聚类的算法将高度相关的特征进行聚类；在特征选择阶段，设计了三种启发式的排序策略从簇中删除冗余的和无关的特征。

本文仍存在一些问题值得探讨，未来工作主要包括以下方面：（1）计算相关性的时候，考虑其他类型的相关性计算方法，如Relief相关性；（2）研究更有效的特征排序策略，提高性能；（3）探究被挑选的特征所属的类别，为以后研究软件缺陷数预测工作提供指导。

参考文献：

[1] 陈翔，顾庆，刘望舒，等.静态软件缺陷预测方法研究[J].软件学报，2016，27（1）：1-25.

[2] 王青，伍书剑，李明树.软件缺陷预测技术[J].软件学报，2008，19（7）：1565-1580.

[3] 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.

[4] Kamei Y，Shihab E.Defect prediction：accomplishments and future challenges[C]//IEEE International Conference on Software Analysis，Evolution，and Reengineering，2016：33-45.

[5] Radjenović D，Heričko M，Torkar R，et al.Software fault

prediction metrics:a systematic literature review[J].Information & Software Technology,2013,55(8):1397-1418.
[6] 何吉元,孟昭鹏,陈翔,等.一种半监督集成跨项目软件缺陷预测方法[J].软件学报,2017,28(6):1455-1473.
[7] 陈翔,王莉萍,顾庆,等.跨项目软件缺陷预测方法研究综述[J].计算机学报,2018,41(1):254-274.
[8] Rajbahadur G K,Wang S,Kamei Y,et al.The impact of using regression models to build defect classifiers[C]//IEEE International Conference on Mining Software Repositories,2017:135-145.
[9] Graves T L,Karr A F,Marron J S,et al.Predicting fault incidence using software change history[J].IEEE Transactions on Software Engineering,2000,26(7):653-661.
[10] Ostrand T J,Weyuker E J,Bell R M.Predicting the location and number of faults in large software systems[J].IEEE Trans on Software Engineering,2005,31(4):340-355.
[11] Yu Liguo.Using negative binomial regression analysis top predict software faults:a study of apache ant[J].International Journal of Information Technology & Computer Science,2012,4(8):63-70.
[12] Liu S,Chen X,Liu W,et al.FECAR:a feature selection framework for software defect prediction[C]//Computer Software and Applications Conference,2014:426-435.
[13] Gao K,Khoshgoftaar T M,Wang H,et al.Choosing software metrics for defect prediction:an investigation on feature selection techniques[J].Software Practice & Experience,2011,41(5):579-606.
[14] Liu W,Liu S,Gu Q,et al.Empirical studies of a two-stage data preprocessing approach for software fault prediction[J].IEEE Transactions on Reliability,2016,65(1):38-53.
[15] Ni Chao,Liu Wangshu,Chen Xiang,et al.A cluster based feature selection method for cross-project software defect prediction[J].Journal of Computer Science and Technology,2017,32(6):1090-1107.
[16] 刘望舒,陈翔,顾庆,等.一种面向软件缺陷预测的可容忍噪声的特征选择框架[J].计算机学报,2018,41(3):506-520.
[17] 陈翔,赵英全,顾庆,等.基于文件粒度的多目标软件缺陷预测方法实证研究[J].软件学报,2019.
[18] 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(9).
[19] 陈翔,沈宇翔,孟少卿,等.基于多目标优化的软件缺陷预测特征选择方法[J].计算机科学与探索,2018,12(9):1420-1433.
[20] Ni C,Liu W,Gu Q,et al.FeSCH:a feature selection method using clusters of hybrid-data for cross-project defect prediction[C]//Computer Software and Applications Conference,2017:51-56.
[21] Radjenović D,Heričko M,Torkar R,et al.Software fault prediction metrics:a systematic literature review[J].Information & Software Technology,2013,55(8):1397-1418.
[22] Menzies T,Greenwald J,Frank A.Data mining static code attributes to learn defect predictors[J].IEEE Transactions on Software Engineering,2006,33(1):2-13.
[23] Yu X,Liu J,Yang Z,et al.Learning from imbalanced data for predicting the number of software defects[C]//IEEE International Symposium on Software Reliability Engineering,2017:78-89.
[24] Wang Jue,Zhang Hongyu.Predicting defect numbers based on defect state transition models[C]//IEEE International Symposium on Empirical Software Engineering and Measurement,2012:191-200.
[25] Janes A,Scotto M,Pedrycz W,et al.Identification of defect-prone classes in telecommunication software systems using design metrics[J].Information Sciences,2006,176(24):3711-3734.
[26] Gao K,Khoshgoftaar T M.A comprehensive empirical study of count models for software fault prediction[J].IEEE Transactions on Reliability,2007,56(2):223-236.
[27] Chen M,Ma Y.An empirical study on predicting defect numbers[C]//International Conference on Software Engineering and Knowledge Engineering,2015.
[28] Yu X,Liu J,Yang Z,et al.Learning from imbalanced data for predicting the number of software defects[C]//IEEE International Symposium on Software Reliability Engineering,2017:78-89.
[29] Rathore S S,Kumar S.A decision tree regression based approach for the number of software faults prediction[J].ACM SIGSOFT Software Engineering Notes,2016,41(1):1-6.
[30] Rathore S S,Kumar S.An empirical study of some software fault prediction techniques for the number of faults prediction[J].Soft Computing,2017,21(24):7417-7434.
[31] Rathore S S,Kumar S.Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems[J].Knowledge-Based Systems,2016.
[32] Rathore S S,Kumar S.Towards an ensemble based system for predicting the number of software faults[J].Expert Systems with Applications,2017,82.
[33] Rodriguez A,Laio A.Clustering by fast search and find of density peaks[J].Science,2014,344(6191):1492.
[34] Clauset A,Shalizi C R,Newman M E J.Power-law distributions in empirical data[J].SIAM Review,2009,51(4):661-703.
[35] Yan M,Fang Y,Lo D,et al.File-level defect prediction:unsupervised vs. supervised models[C]//ACM/IEEE International Symposium on Empirical Software Engineering and Measurement,2017:344-353.
[36] He Peng,Li Bing,Liu Xiao,et al.An empirical study on software defect prediction with a simplified metric set[J].Information & Software Technology,2015,59(C):170-190.
[37] Conte S D,Dunsmore H E,Shen V Y.Software engineering metrics and models[M].[S.l.]:Benjamin/Cummings Publishing Company,Inc,1986.