软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(10):3090−3114 [doi: 10.13328/j.cnki.jos.005790]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

软件缺陷预测技术研究进展*

宫丽娜[1,2,3], 姜淑娟[1,2], 姜 丽[1,2]

[1](中国矿业大学 计算机科学与技术学院,江苏 徐州 221116)

[2](矿山数字化教育部工程研究中心,江苏 徐州 221116)

[3](枣庄学院 信息科学与工程学院,山东 枣庄 277160)

通讯作者: 姜淑娟, E-mail: shjjiang@cumt.edu.cn

摘 要: 随着软件规模的扩大和复杂度的不断提高,软件的质量问题成为关注的焦点,软件缺陷是软件质量的对立面,威胁着软件质量,如何在软件开发的早期挖掘出缺陷模块成为一个亟需解决的问题.软件缺陷预测通过挖掘软件历史仓库,设计出与缺陷相关的内在度量元,然后借助机器学习等方法来提前发现与锁定缺陷模块,从而合理地分配有限的资源.因此,软件缺陷预测是软件质量保证的重要途径之一,近年来已成为软件工程中一个非常重要的研究课题.汇总近 8 年(2010 年~2017 年)国内外的缺陷预测技术的研究成果,并以缺陷预测的形式为主线进行分析,首先介绍了软件缺陷预测模型的框架;然后从软件缺陷数据集、构建模型的方法及评价指标这 3 个方面对已有的研究工作进行分类归纳和比较;最后探讨了软件缺陷预测的未来可能的研究方向、机遇和挑战.

关键词: 软件缺陷预测;软件度量;数据预处理;机器学习;性能评价指标

中图法分类号: TP311

中文引用格式: 宫丽娜,姜淑娟,姜丽.软件缺陷预测技术研究进展.软件学报,2019,30(10):3090−3114. http://www.jos.org.cn/1000-9825/5790.htm

英文引用格式: Gong LN, Jiang SJ, Jiang L. Research progress of software defect prediction. Ruan Jian Xue Bao/Journal of Software, 2019,30(10):3090−3114 (in Chinese). http://www.jos.org.cn/1000-9825/5790.htm

Research Progress of Software Defect Prediction

GONG Li-Na[1,2,3], JIANG Shu-Juan[1,2], JIANG Li[1,2]

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

[2](Engineering Research Center of Mine Digitalization of Ministry of Education, Xuzhou 221116, China)

[3](College of Information Science and Engineering, Zaozhuang University, Zaozhuang 277160, China)

Abstract: With the improvement of the scale and complexity of software, software quality problems become the focus of attention. Software defect is the opposite of software quality, threatening the software quality. How to dig up defect modules in the early stages of software development has become a urgent problem that needs to be solved. Software defect prediction (SDP) designs the internal metrics related defects by mining software history repositories, and then in advance finds and locks the defect modules with the aid of machine learning methods, so as to allocate the limited resources reasonably. Therefore, SDP is one of the important ways of software quality assurance (SQA), which has become a very important research subject in software engineering in recent years. Based on the form of defect perfection, this research offers a systematic analysis of the existing research achievements of the domestic and foreign researchers in recent eight years (2010~2017). First, the research framework of SDP is given.Then the existing research achievements are classified

* 基金项目: 国家自然科学基金(61673384)

Foundation item: National Natural Science Foundation of China (61673384)

本文由“面向 DevOps 的软件工程新技术”专题特约编辑荣国平、白晓颖、岳涛推荐.

收稿时间: 2018-08-31; 修改时间: 2018-10-31; 采用时间: 2018-12-14; jos 在线出版时间: 2019-08-09

CNKI 网络优先出版: 2019-08-12 12:08:00, http://kns.cnki.net/kcms/detail/11.2560.TP.20190812.1207.004.html

and compared from three aspects, including datasets of SDP, the methods models and the evaluation indicators. Finally, the possible research directions are pointed out.

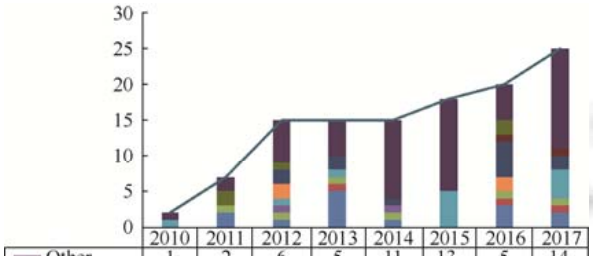
Key words: software defect prediction (SDP); software metrics; data preprocessing; machine learning; performance evaluation criteria

随着软件规模的扩大和复杂度的不断提高,软件的质量问题成为了关注的焦点,软件缺陷是软件质量的对立面,威胁着软件质量.因此,如何在软件开发的早期挖掘出缺陷模块成为一个亟需解决的问题.软件缺陷预测技术能够在软件开发过程中使用软件模块的度量元数据来提前发现与锁定缺陷模块,从而合理地分配有限资源,为软件质量提供保障.

从20世纪70年代以来,软件缺陷预测技术一直是软件研究人员和开发者非常关注的研究热点之一[1].研究者从软件仓库出发,基于软件度量数据,利用统计学和机器学习等方法来研究缺陷预测技术,并且取得了丰富的研究成果.

目前已有一些研究人员从不同角度对缺陷预测的相关研究工作进行了总结[2−5],重点讨论了静态缺陷预测方法、数据驱动缺陷预测方法等.特别地,Li 等人以数据来源为主线对版本内数据、跨版本数据和跨项目数据进行了归纳和分析[2],Chen 等人对静态软件缺陷预测技术[5]和跨项目缺陷预测技术[3]分别进行了归纳和总结.在此基础上,本文收集了近 8 年的缺陷预测技术的相关文献,以缺陷预测形式为主线,从数据集、构建方法、评价指标等多个角度对相关技术进行了汇总和分析.

本文使用 defect prediction、software defect prediction、fault prediction 及 software fault prediction 等搜索关键词,在 ACM Digital Library、Springer Link online Library、IEEE Xplore Digital Library、Elsevier ScienceDirect 及 CNKI 等在线数据库中全面搜索了 2010 年~2017 年在期刊和会议中发表的相关文献,截止到 2018 年 1 月进行人工的筛选和过滤,选择出与缺陷预测技术相关的高质量论文 117 篇,其中,期刊论文 85 篇,会议论文 30 篇,其他 2 篇.其发表出处的分布情况如图 1 所示,发表的文献每年呈增长趋势.其中不乏在软件工程领域的国际顶级期刊和顶级会议上的研究论文.例如:TSE 期刊(14 篇)、FSE/ESEC 期刊(3 篇)、ICSE 会议(6 篇)、ASE 会议(2 篇)、IST(12 篇)和 ESE(12 篇).基于上述分析不难发现,软件缺陷预测技术一直是软件工程中一个重要的研究课题.



	2010	2011	2012	2013	2014	2015	2016	2017
Other	1	2	6	5	11	13	5	14
ESEM	0	2	1	0	0	0	2	0
JSS	0	0	0	0	0	0	1	1
ESE	0	0	2	2	1	0	5	2
ASE(J)	0	0	2	0	0	0	2	0
IST	1	0	1	1	0	5	0	4
ASE(C)	0	0	1	0	1	0	0	0
ICSE	0	1	1	1	1	0	1	1
FSE/ESEC	0	0	0	1	0	0	1	1
TSE	0	2	1	5	1	0	3	2
总量	2	7	15	15	15	18	20	25

Fig.1 The source distribution of the literature

图 1 文献发表出处分布

本文第 1 节对软件缺陷预测模型的框架进行描述.第 2 节系统分析已有软件缺陷数据集及数据集的预处理过程.第 3 节对软件缺陷预测模型的构建方法进行总结和分析.第 4 节总结分析目前软件缺陷预测技术的相关性能评价指标.最后总结全文,并对未来值得关注的研究方向进行探讨.

1 软件缺陷预测模型框架

软件缺陷预测技术通过使用软件度量元数据来对软件模块的缺陷倾向性、缺陷数或者缺陷严重度等进行预测.根据模型的构建方法,软件缺陷预测模型可以用数学符号表示如下.

假设给定包含 n 个样本的数据集 $D=\{(x_1,y_1),(x_2,y_2),\ldots,(x_i,y_i),,,(x_n,y_n)\}$,其中,$x_i=(a_1,a_2,\ldots,a_j,\ldots,a_d)\in R^d$ 表示软件模块 i 的度量属性向量,a_j 表示模块的第 j 个度量元的值,d 表示度量元的个数;$y_i\in Y$ 表示软件模块 i 的标记,Y 表示所有标记的集合,如若是对缺陷倾向性进行预测,则 $Y=\{$有缺陷,无缺陷$\}$,如若是对缺陷数进行预测,则 $Y=N$,如若是对缺陷严重度进行预测,则 $Y=\{Z|Z\geqslant 0\}$,根据 Y 值对程序模块严重度进行排序.软件缺陷预测模型 $f:R^d\rightarrow Y$,表示了从软件模块的内部度量元属性到软件外部缺陷的映射.

具体的缺陷预测过程由 4 部分组成[2−5]:收集软件缺陷数据、提取度量元、构建软件缺陷预测模型的方法和性能评价指标.图 2 给出了软件缺陷预测模型的框架.

(1) 从包含软件项目开发周期的全部数据的软件仓库里收集软件缺陷数据;

(2) 提取与软件缺陷(缺陷数、缺陷倾向性、缺陷严重性)相关的不同软件度量元,并对软件模块进行标记;

(3) 在提取的数据集上利用统计学和机器学习的技术构建软件缺陷预测模型;

(4) 最后通过性能评价指标(比如精确率、召回率、AUC 等)对构建的预测模型进行评价.

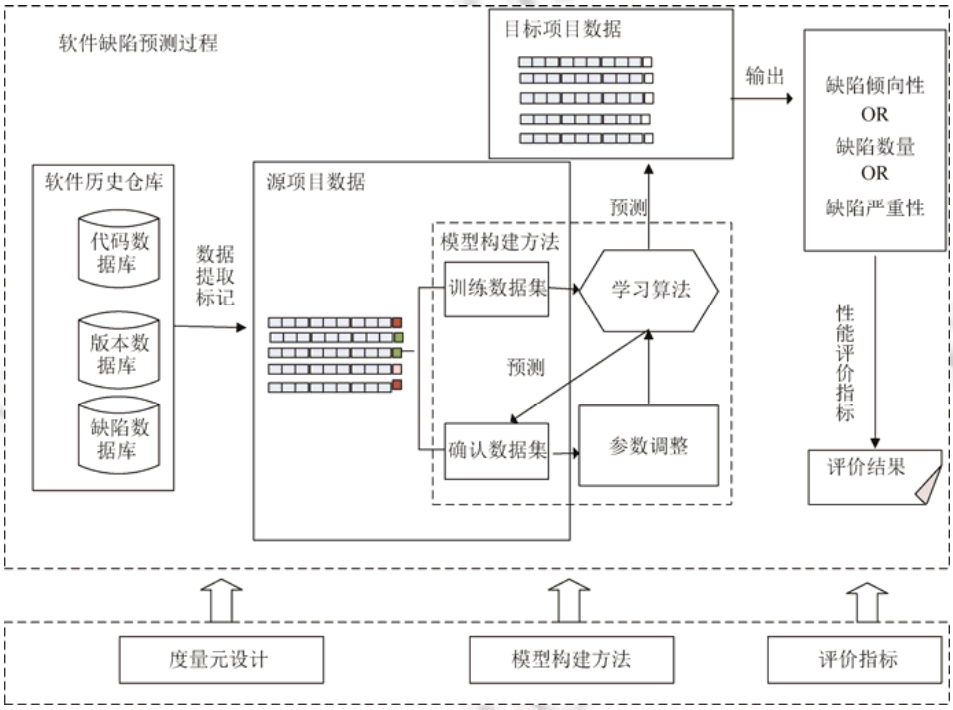


Fig.2 The framework of software defect prediction model

图 2 软件缺陷预测模型框架

在上述流程中,第(2)步和第(3)步的处理直接决定着软件缺陷预测模型的性能.因此,本文重点关注软件缺陷数据集及构建方法的研究进展,在总结常用的数据集及数据预处理方法的基础上,对构建缺陷预测模型的关键技术进行了综述,最后给出经常使用的评价指标.接下来的章节将从这几方面分别展开详细的叙述.

2 软件缺陷数据集

软件缺陷预测技术需要大量的缺陷数据来挖掘模块内部的度量元与模块外部的缺陷之间的某种关系,因

此获取有效的缺陷数据集及度量元是构建软件缺陷预测模型的前提.软件缺陷数据仓库中包含大量项目的缺陷数据集,但是存在着异常值、高维度、类不平衡等问题,所以,在使用前需要对这些数据集中的数据进行预处理.使用的软件缺陷数据仓库、度量元及数据预处理技术直接关系到预测模型的性能.

接下来,我们分别从软件缺陷数据仓库、缺陷度量元以及数据处理这 3 个方面进行详细的总结和分析.

2.1 缺陷数据仓库

缺陷数据仓库收集了软件项目从需求、设计、开发、测试等各个生命期阶段的缺陷数据.缺陷库对于能否构建一个有效的预测模型起到关键作用.不同的缺陷库具有不同的特征信息,其中,用于缺陷预测的信息主要包括数据来源、项目成熟度、开发环境及粒度等.

(1) 数据来源

随着越来越多的大型开源软件的公开,缺陷数据仓库经历了从私有数据集到公开数据集的发展阶段.缺陷预测模型对不同来源的缺陷库会产生不同的性能.到目前为止,用于软件缺陷预测的数据仓库大体上可以分为私人/商业仓库、部分公共/免费仓库和公共仓库这 3 种类型[4].

- 私人/商业仓库:缺陷库只能公司内部使用,以这些数据集为依据进行的实验无法重现.
- 部分公共/免费仓库:缺陷库中无度量元信息,源代码和错误信息公开使用.
- 公共仓库:缺陷库中度量元及错误信息都可使用(如:NASA、PROMISE 等公共库).

本文针对收集和选出的117篇论文进行分析,发现预测模型使用的数据来源的分布如图3所示,公共仓库的数据集(NASA、PROMISE、AEEEM 及 ReLink)占绝大多数,越来越多的研究者使用公共缺陷数据集,这样有利于缺陷预测模型的重现,实现各个模型的比较分析.

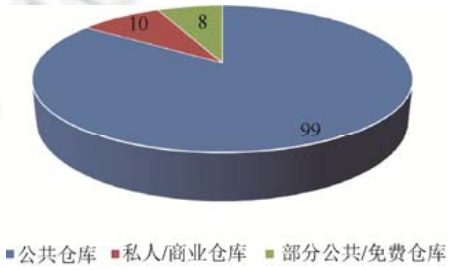


Fig.3 The distribution of the repository

图 3 数据仓库的分布情况

(2) 项目成熟度

项目成熟度主要是指项目进化的版本数.通常,一个版本对应于项目的一个缺陷库,而不同版本的缺陷库会对缺陷预测模型产生不同影响.因此,部分研究者提出了跨版本的缺陷预测模型,研究使用之前版本数据对缺陷预测模型的预测能力.

(3) 开发环境

开发环境指的是软件项目开发过程中使用的软件工具,比如:Java、C++等.同样,使用不同开发语言的缺陷库会对缺陷预测模型产生不同的影响.因此,在选取训练集时需要考虑该项目的开发环境.

(4) 粒度

粒度指的是缺陷库中每条样本的项目粒度.其中可以分为类粒度、文件粒度或者包粒度等.使用不同粒度的缺陷库会对缺陷预测模型产生不同的影响.因此,部分研究者从不同粒度提取不同的度量元,研究它们对缺陷预测模型的影响.

综上所述,由于数据来源、项目成熟度、开发环境及粒度等的不同会形成不同类型的缺陷库.但是,目前大部分的研究在建立预测模型前很少考虑这些缺陷库的特征,而且对于这些特征对缺陷预测模型的影响的研究

也很少.因此,在后续研究中可以讨论这些特征对缺陷预测模型的不同影响程度.

2.2 软件缺陷数据度量元

软件缺陷预测技术可以使项目开发人员重点关注那些有缺陷的模块,但是要挖掘出有缺陷的模块,需要找出与缺陷外在表现有关系的内在属性,这些属性就是软件度量数据,即度量元.通常,每个度量都与软件项目的一些功能属性相关(比如耦合、内聚、继承、代码更改等)[6].目前,现有的度量方法可以分为软件产品属性度量和软件过程度量两大类,图 4 表明了文献中使用到的度量方法分类情况.本文在文献[5]的基础上,对度量方法作了进一步的总结和分析.

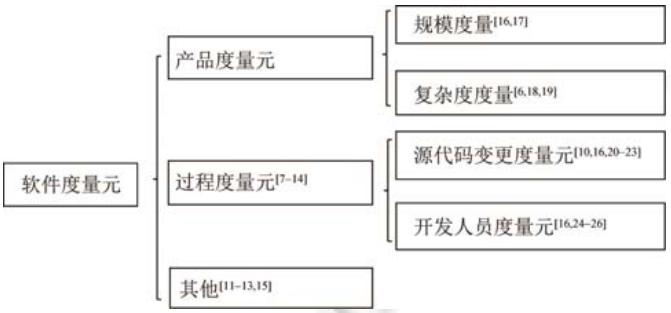


Fig.4 The literature overview of software metrics

图 4 软件度量元文献分类总览图

2.2.1 软件产品度量

软件产品度量是通过分析软件代码来设计度量元,描述软件产品的特征,用于产品评估和决策.软件产品度量重点关注的是代码的规模和内在复杂度等属性,软件产品度量包括规模度量、复杂度度量等.

(1) 规模度量

软件产品的规模度量是以代码行数、对象点、特征点或功能点数等来衡量的.常用的度量方法主要有:代码行度量(lines of code,简称 LOC)方法[16]、功能点分析法、面向对象软件的对象点方法及构造性成本模型(constructive cost model,简称 COCOMO).

(2) 复杂度度量

软件产品的复杂度度量是用于确定程序控制流或软件系统结构的复杂程度的指标.主要有面向过程的传统度量和面向对象的度量.

传统度量是在软件工程初期,针对面向过程程序设计的方法级别进行的度量.常用的是 Halstead[6]和 McCabe[18]的度量.

- Halstead 度量法:依据程序中可执行代码行的操作数和操作符的数量来计算程序的复杂性.其值越大,程序结构就越复杂.
- McCabe 度量法:依据控制流的复杂性来计算程序的复杂性,顺序结构单一,则程序最简单,而选择和循环构成的环路越多,程序越复杂.

随着面向对象开发方法成为主流方法,研究者依据代码结构中的继承、耦合、内聚等关系计算代码的复杂度,形成了面向对象度量指标[27].主要的面向对象指标(OO)包括 C&K 度量[28]和 MOOD 度量[29].

2.2.2 软件过程度量

软件过程(process)度量是对软件开发过程的各个因素进行度量.过程度量与软件开发流程密切相关,对软件开发过程的各阶段行为进行目标管理,为软件开发的过程评价及控制提供定量性度量.例如,代码变更度量、开发人员度量等过程度量元被提了出来.

(1) 源代码变更度量

在软件项目开发过程中,源代码是不断变更的,代码在不同的版本中会增加、删除或者修改,从而有可能导

致软件缺陷.因此,一些研究者借助源代码的变更,设计出新的源代码变更度量元进行软件缺陷预测.Madeyski等人[10]通过实验确定过程度量元与缺陷预测数量的关系,设计出了NR(number of revisions)、NDC(number of distinct committers)、NML(number of modified lines)、NDPV(number of defects in previous version)4个过程度量元,并通过实验发现NDC、NML对软件缺陷预测有很大的影响.Feng等人[28]提出了PL(programming language)、IT(issue tracking)、TLOC(total lines of code)、TNF(total number of files)、TNC(total number of commits)及TND(total number of developers)这6种度量元,并使用这6种度量元对项目内和跨项目缺陷预测模型进行分析.

(2) 基于开发人员度量

软件开发过程中人是不可忽视的重要因素,基于开发人员度量就是对人进行度量.Bhattacharya等人[24]引入社交网络概念,提取变更信息和开发者关系信息构建开发者交互关系图.Lee等人[25]对开发人员的交互行为进行分析,提出了MIMS(micro interaction metrics)度量元,通过实验验证MIMS对缺陷预测性能的影响.Ekanayake等人[26]提出了开发人员的数量的度量元,通过实验验证得出:该度量元对预测的质量有影响.Jiang等人[7]根据个人开发者的表现作为度量元进行软件修改的缺陷预测,为每一个开发者建立一个预测模型PCC(personalized change classification).

2.2.3 度量元分析

近几年,随着开源项目以及缺陷公共数据集的增多,部分研究者开始把工作转移到哪些度量元可以构建更好的缺陷预测模型研究中.Muhammed等人[9]通过实验对比静态度量元和过程度量元哪个是更影响类不平衡的数据,实验结果表明,过程度量元要比静态度量元产生更好的分类效果.Shin等人[8]同时使用复杂性、代码变更和开发人员活动度量元对软件缺陷进行预测,实验发现,使用这些度量元的组合能够得到较好的预测性能.Shepperd等人[11]分析了分类器、数据集、度量元及研究团队4个因素对缺陷预测模型性能的影响,实验发现,研究团队对缺陷预测模型的影响最大.接着,Chakkrit等人[12]对Shepperd的研究加以重现,发现研究人员与使用的数据集及度量元有很大的关系,建议使用更广泛的数据集.Okutan等人[14]设计了开发者人数(number of developers,简称NOD)和源代码质量(lack of coding quality,简称LOCQ)两个度量元,并使用贝叶斯网络分析各种度量元与错误倾向性之间的关系,发现RFC(response for class)、LOC(lines of code)及LOCQ是最有效的度量元,而CBO(coupling between objects)、WMC(weighted method perclass)及LCOM(lack of cohesion of methods)对缺陷倾向性的影响较小,NOD和缺陷倾向性预测呈正相关的关系.

本文针对收集的117篇论文进行分析,发现预测模型使用的度量元的分布如图5所示.大部分研究者使用混合的度量元集(如:OO+LOC、OO+Process),其次是面向对象和过程度量元.软件产品度量用来描述软件产品的特征,针对最终软件产品进行度量,而过程度量是对软件开发过程的各个质量指标进行度量,软件过程质量直接影响软件产品质量.因此,同时考虑这两种度量方法可以进一步提高缺陷预测模型的性能.

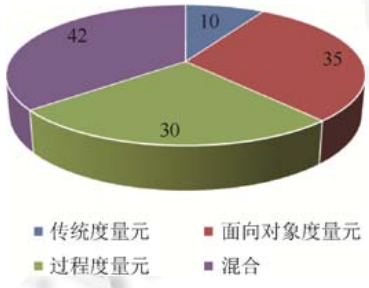


Fig.5 The distribution of the metric

图5 度量元的分布情况

2.3 数据质量相关问题处理

软件缺陷数据集从包含软件项目开发周期的软件仓库里收集.然而,在挖掘软件历史仓库的过程中,数据集中包含一些降低预测模型性能的不必要的信息,如异常值、高维度、类不平衡、数据差异等.因此,在使用这些

数据集之前,需要处理数据集中存在质量问题的数据.本文对文献进行了汇总分析,图 6 所示为具体数据质量处理的文献分类总览图.

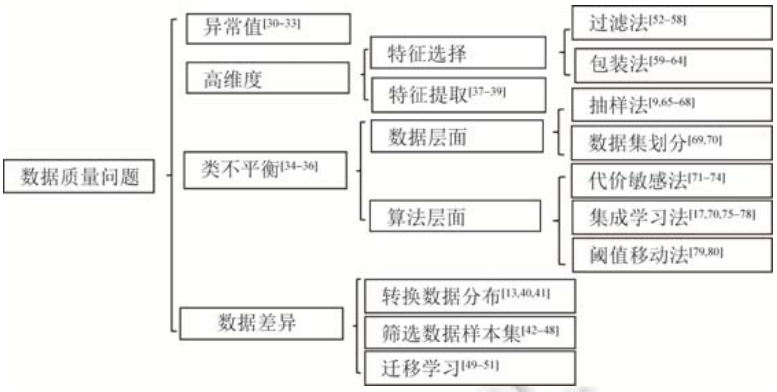


Fig.6 The literature overview of data processing

图 6 数据处理文献分类总览图

2.3.1 异常值

在挖掘软件仓库的过程中,对代码类型标记和软件度量等操作可能产生部分数据值不一致或一些实例重复等异常值[31]问题,这些异常值影响到预测模型的性能.Gray 等人[30]的实验结果表明,异常值降低了缺陷预测模型的性能.所以,在构建预测模型前要处理离群的异常值,研究者在异常值检测及处理方面作了相应的研究.

(1) 检测异常值

数据集中的异常值可以通过箱线图进行检测,箱线图的结构如图 7 所示,分别由最小值、中位数、最大值和两个四分位数这 5 部分组成.

(2) 处理异常值

对异常值的处理在缺陷预测技术中主要通过删除异常值的方法来解决.Ryu 等人[32]针对跨项目缺陷数据集中的异常数据进行处理,结合实例中的异常属性个数超过某个范围就剔除该实例的方法,提出了 VAB-SVM (value aware boosting with support vector machine)预测方法.Cao 等人[33]对缺陷数据集中的异常值也进行了研究,通过四分位距(inter quartile range,简称 IQR)去除源项目和目标项目数据集中的异常数据,实验结果表明,去除了异常值后,缺陷预测模型的性能得到提升.

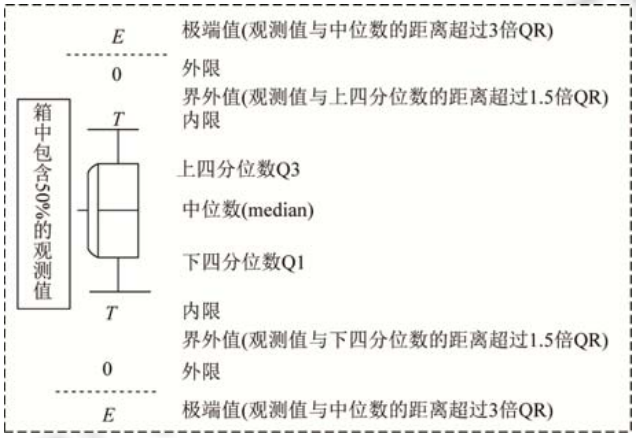


Fig.7 The structure of boxplot

图 7 箱线图结构

2.3.2 高维度数据

在程序模块度量中,不同的度量方法(如规模度量 LOC、过程度量、传统度量和面向对象度量)被应用在缺陷预测中,文献[8]的实验发现,使用度量元的组合能够得到较好的预测结果.大部分研究者使用的是不同度量方法组合的度量元集,比如 OO+LOC、OO+Process+LOC 等.然而,同时考虑所有的度量元会导致数据集的高维度数据问题,需要设计有效的方法来识别出冗余的度量元和无关度量元.He 等人[53]认为,对于大量的特征进行特征选择是一种合理的方法,处理后的特征子集有助于提升缺陷预测模型的性能.因此,对缺陷数据集进行高维度数据处理能够促进数据可视化和数据的可理解,缩短训练时间,提高预测模型的性能.应用于软件缺陷预测模型的高维度数据处理方法主要包括特征选择和特征提取这两种.

(1) 特征选择

特征选择是从原始的缺陷度量元数据集中得到最“好”的度量元子集,在选择度量元子集时,需要一个评价函数进行评估,根据评价函数的工作原理,主要分为过滤和包装两种方法[52],其工作流程如图 8 所示.过滤法是对每一个度量元计算 Chi-squared test(CS)或 information gain(IG)或 correlation coefficient scores 或 Pearson correlation 等的值,根据这些值进行特征选择,选择的过程与后续的学习器无关[63].相反地,包装方法是根据学习器的性能来作为最佳特征子集选择的评价准则.包装法是将学习器的性能作为评价函数[61],所以,包装法的性能要优于过滤法的性能.但是,包装法的时间开销较大,而过滤式方法与学习器无关,开销较小,所以泛化能力强于包装法.

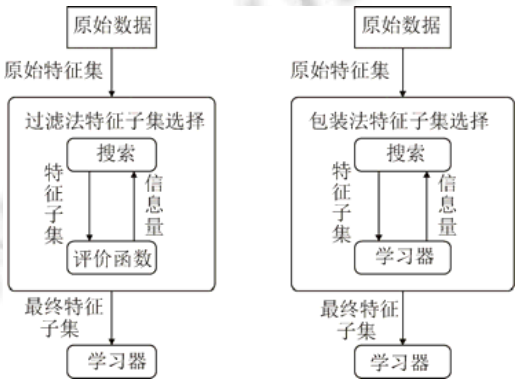


Fig.8 Process of filter and wrapper

图 8 过滤法和包装法的工作流程

He 等人[53]提出了一种 Top_k 的特征选择方法.首先使用过滤法对缺陷数据集进行特征选择,然后根据特征在所有的缺陷数据集中出现的次数进行排序,并且引入覆盖索引来决定最优的 k 值.Sadhana 等人[54]提出了使用局部选择和全局优化的特征选择方法.首先,利用基于 IG 的过滤法(mutual information maximization,简称 MIM)和基于 correlation coefficient 的过滤法(sequential backward search,简称 SBS)得到两个初始的特征子集,然后,将这些子集进行组合,将全局优化技术(GA(genetic algorithm)、DEFS(differential evolution-cased feature selection)或 PSO(particle swarm optimization))分别应用到组合的子集中,将方差作为停止标准,从而得到有效的特征子集.Chao 等人[55]提出了基于聚类的特征选择方法 FeSCH(feature selection using clusters of hybrid-data)以解决跨项目缺陷预测问题.该方法分为两个阶段.第 1 阶段使用基于密度的 DPC(density peaks clustering)聚类方法将原始特征集划分为多个簇,第 2 阶段分别使用局部密度策略(local density of features,简称 LDF)、相似特征分布策略(similarity of feature distributions,简称 SFD)及信息增益策略(feature class relevance,简称 FCR)对每个簇的特征进行排序,然后选择前 Top_k 的特征.

Yu 等人[58]针对缺陷数的预测提出了结合谱聚类及特征排序的 FSCR(feature spectral clustering and feature ranking)特征选择方法.首先,根据每两个特征之间的相关性,利用谱聚类方法对原始特征集进行聚类.然后,利用 ReliefF 算法计算每个特征与缺陷数之间的相关性.最后,从每个聚类里选择 Top_k 的特征组成特征子集.

Chen 等人[62]首次将多目标优化的特征选择方法应用于软件缺陷预测中,提出了 MOFES(multi-objective feature selection)方法.主要考虑了两个目标,一个是尽可能地减少特征子集的数量,另一个是最大化缺陷预测模型的性能.其中,优化算法使用NSGA-II方法,染色体的编码模式根据特征集决定,若数据集有*n*个特征,则编码为*n*位的串,*i-th*位的值为1,表明该*i-th*特征被选择.

除此之外,Wang等人[64]首次研究了不同的性能指标对基于Wrapper特征选择的预测模型性能的影响.该研究以逻辑回归学习器作为基础分类器,分别使用Accuracy、AUC、Area Under the Precision-Recall Curve(PRC)、Best Geometric Mean(BGM)及 Best Arithmetic Mean(BAM)作为性能评价指标,实验结果表明,基于 BAM 的Wrapper 方法是最好的 Wrapper 特征选择方法.Khoshgoftaar 等人[52]使用 16 个软件数据集进行实验,对比了 7个基于过滤的特征排序技术.这些技术包括 chi-squared(CS)、信息增益(IG)、增益比(gain rate,简称 GR)、对称不确定性(symmetric uncertainty,简称 SU)和 ReliefF(两个变体:RF 和 RFW).

(2) 特征提取

特征提取是根据缺陷度量元之间的关系,将原始特征集映射到一个新特征集中.用于软件缺陷预测的特征提取方法主要有主成分分析(principal component analysis,简称 PCA)方法、判别分析(linear discriminant analysis,简称 LDA)方法和多维标度测量(multidimensional scaling,简称 MDS)方法.PCA 和 MDS 都是无监督的特征提取方法,而 LDA 是有监督的特征提取方法.当数据规模较小时,PCA 优于 LDA,但当数据规模较大时,LDA 优于PCA.

Öztürk 等人[37]比较哪种度量元对类不平衡的缺陷预测模型影响最大时,使用主成分分析 PCA 方法进行特征提取.Lu 等人[38]使用半监督学习方法 FTcF(fitting the confident fits)对软件缺陷进行预测,在使用该方法前首先使用 MDS 方法进行特征提取.Jing 等人[69]使用改进的 LDA 方法对项目内和跨项目的数据集进行特征提取.

另外,有些研究者对特征选择和特征提取方法进行了比较.Lu 等人[39]比较了特征选择的 IG 方法和特征提取的 MDS 方法.实验结果表明,MDS 方法要优于 IG 方法.但是特征选择后的特征是原来特征的一个子集,而特征提取则形成了新的特征空间.相比较而言,特征选择能够对缺陷预测模型有更好的理解和解释,能够分辨出哪些度量元对缺陷更为重要,有利于指导软件的研发.

2.3.3 类不平衡问题

通常,软件产品大致符合二八原则,80%的缺陷集中在 20%的程序模块中,缺陷数据分布不均匀,有缺陷的模块要远少于无缺陷模块,即类不平衡问题.类不平衡严重影响了分类模型的性能,无缺陷的实例支配着数据样本,学习分类器会偏向于无缺陷的实例,因而,分类器对于有缺陷的实例不会产生好的分类结果[66].而在软件测试中,有缺陷的模块被误报为无缺陷模块,会给软件公司带来更大的损失.

类不平衡学习是当前机器学习中的一个热点研究领域.在软件缺陷预测中也有不少研究,形成了若干类不平衡的学习方法,用于缺陷预测的类不平衡调整方法可大体上分为数据层面和算法层面两类.

(1) 数据层面

数据层面决策是从数据准备阶段入手,对原始不平衡的缺陷数据进行调整,从而达到平衡的数据.主要包括抽样方法及数据集划分方法.

(a) 抽样方法

主要采用随机欠抽样或者随机过抽样技术将不平衡的缺陷数据集转化成平衡数据集.随机过抽样技术是随机地复制小类(有缺陷)的实例,但却使模型训练难度加大,容易导致模型的过拟合问题.针对此问题,可以使用合成少数类过采样技术(synthetic minority over-sampling,简称 SMOTE),对少数类样本进行分析并根据少数类样本人工合成新的样本.相反地,随机欠抽样技术是随机剔除大类(有缺陷)的实例,但却带来一些重要信息的缺失.针对此问题,可以采用迭代的思想,如 BalanceCascade,是通过迭代思想不断更新大类的实例.

在软件缺陷预测模型中,不少研究者考虑到了类不平衡问题并采用抽样技术解决这个问题.Zhang 等人[67]提出了一种改进的采样算法,被称为扩散的基于相似性的分类算法(expanded dissimilarity based classification algorithm,简称 EDBC),实验结果表明,改进算法优于随机欠采样和随机过采样,但其性能不如集成学习方法.

Muhammed[9]首先利用主成分分析对数据进行特征提取,然后利用 SOMTE、Virtual 和 HSDD(hybrid sampling for defect data sets)方法进行过抽样,利用 Bayes、Naïve Bayes(NB)、Random Forest(RF)、J48 和 LIB-SVM 进行训练,发现不同抽样方法的性能差别不大.

(b) 数据集划分方法

数据集划分法的主要思想是将不平衡的缺陷数据集按照一定的规则划分为多个平衡的缺陷数据集,然后在平衡的缺陷数据集上训练多个分类器,按照一定的学习方法集成在一起.

在软件缺陷预测模型中,有部分研究者采用数据集划分方法解决类不平衡问题.Xia 等人[70]设计了 ELBlocker 方法来预测阻碍性错误.首先,将训练集随机分为多个互斥集合,对每个互斥集合构建一个分类器.然后,集成这些分类器,并能自动确定恰当的不平衡决策边界来区分阻碍性错误和非阻碍性错误.若一个错误的似然分数大于决策边界值,即使它的似然值很低,也会被分类为阻碍性错误.Jing 等人[69]提出了改进的子类判别分析(improved subclass discriminant analysis,简称 ISDA)方法以解决类不平衡问题,把每个类的数据分成若干个子类,对每个子类进行判别分析学习分类能力强的特征,从而解决类不平衡问题.

抽样方法和数据集划分方法都是从数据层面上解决缺陷预测的类不平衡问题.抽样方法简单,但容易出现重要数据丢失或过拟合的现象.数据集划分方法需要与集成方法一起使用,相对比抽样方法要复杂,但是可以在一定程度上避免抽样方法的问题.

(2) 算法层面

除了从数据层面降低类不平衡的影响外,也可以直接从算法层面进行改进,设计出适用于类不平衡数据的缺陷预测分类算法.主要包括代价敏感学习方法、集成学习方法及分类阈值移动方法.

(a) 代价敏感学习方法

该方法对数据集中的有/无缺陷实例赋予不同的误分类代价,其中,误分类代价可以分为基于类别的代价和基于样本的代价两种.在软件缺陷预测模型中,基于类别的误分代价使用得比较多.Ryu 等人[72]考虑了跨项目的类不平衡问题,提出了基于代价敏感的迁移学习方法 TCSBoost(transfer cost-sensitive boosting).该方法基于分布特征及类不平衡特征,对正确和错误分类的样本赋予不同的权值.Ömer 等人[73]针对类不平衡问题提出了代价敏感的神经网络缺陷预测方法.为了把人工神经网络转化为代价敏感学习器,将假阳性(FP)及假阴性(FN)引入到误差函数中,使用人工蜂群算法对人工神经网络进行训练.Liu 等人[71]提出了两阶段的代价敏感学习方法(two-stage cost-sensitive,简称 TSCS),代价敏感不仅使用于分类阶段,而且使用在特征选择阶段.在特征选择阶段,将代价敏感信息引入到传统的特征选择方法中,设计了 3 种新的代价敏感特征选择算法,即代价敏感方差评分(cost-sensitive variance score,简称 CSVS)、代价敏感拉普拉斯评分(cost-sensitive Laplacian score,简称 CSLS)以及代价敏感约束评分(cost-sensitive constraint score,简称 CSCS).

(b) 集成学习方法

集成学习方法是结合多个弱监督分类器以得到一个更好的强监督分类器,比较常用的集成学习方法有 Bagging、Boosting 和 Stacking.在缺陷预测模型中,部分学者使用集成学习方法解决类不平衡问题.

Sun 等人[76]针对缺陷预测问题提出了基于编码的集成学习(coding based ensemble learning,简称 CEL)方法,该方法首先将无缺陷的实例分成几个箱子,每个箱子样本的数量与有缺陷样本的数量相同,然后用一个特定的编码方案在多类数据集上构建缺陷预测模型.Wang 等人[77]将多核学习方法和集成学习方法相结合,提出了多核集成学习方法(multiple kernel ensemble learning,简称 MKEL).首先,通过集成学习的 boosting 方法反复学习多个内核分类器,然后,根据它们的组合权重集成起来,从而得到最终的多核集成分类器.Ryu 等人[78]针对跨项目的类不平衡问题,提出了 VCB-SVM(value-cognitive boosting with a support vector machine)方法.该模型的相似性权值根据分布特征及非对称误分类代价得到.同时,采用 Boosting-SVM 和重新采样技术来解决类不平衡问题.

(c) 分类阈值移动

该方法的主要思想是在算法分类过程中,阈值的设定不再是 0.5,而且根据有/无缺陷样本的比例对阈值进行移动,根据改变后的阈值在分类器上进行预测.部分学者使用分类阈值移动方法解决缺陷预测的类不平衡问

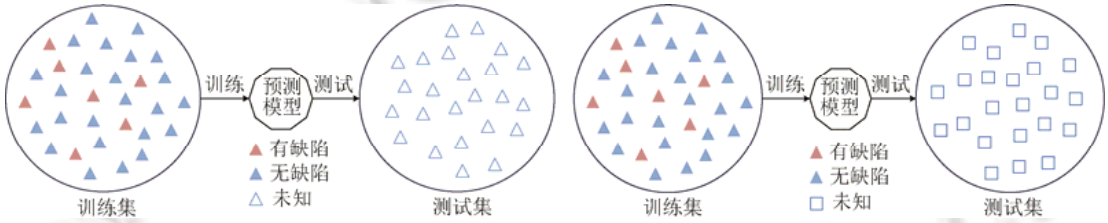
题.Zheng[80]提出了使用神经网络算法解决缺陷预测的类不平衡问题.在分类过程中,使用阈值移动的方法将分类阈值移动到无缺陷的模块,这样可使更多的有缺陷模块得到预测.与代价敏感学习法相比,实验结果表明,阈值移动方法要优于代价敏感方法.Wang 等人[79]在 10 个不同的数据集上对采样方法、阈值移动方法、代价敏感方法和集成学习方法进行了研究,使用 *G*-means 和 AUC 进行评估,集成学习方法表现最佳.

除此之外,部分研究者还将高维度问题和类不平衡问题相结合来构建软件缺陷预测模型.Ömer 等人[65]使用随机抽样方法解决类不平衡问题,特征选择使用 correlation-based feature selection(CFS)过滤方法.Kehan 等人[68]使用抽样的方法解决类不平衡问题,特征选择使用 Filter 方法.Wan 等人[74]使用代价敏感解决类不平衡问题,特征提取使用稀疏编码方式,采用字典学习方法进行缺陷预测.Issam 等人[75]将集成学习方法和特征选择相结合来解决类不平衡问题,该方法同时使用了静态代码度量元和过程度量元.Gao 等人[81]提出了两种结合抽样方法的集成学习方法:selectRUSBoost 和 selectSMOTEBoost.

但是,大部分研究者集中于对项目内的类不平衡问题进行研究,针对跨项目类不平衡问题的研究还比较少,这是今后进一步的研究方向.

2.3.4 数据差异性问题

通常,实际软件开发中,当要预测的项目没有充足的历史数据时,需要使用其他项目的数据.但是不同项目开发过程、使用的语言及应用领域不同,导致训练数据集与测试数据集分布不一致的问题,即存在数据差异性.在缺陷预测领域,数据差异性包括同构(度量元集相同但分布不同)及异构(度量元不同)两种情况,其预测模型如图 9 所示[82].数据差异性的问题会导致无法直接使用传统学习器进行预测,因为传统学习器要求训练数据集和测试数据集同分布,所以需要对训练集和测试集的分布进行转变.在缺陷预测领域,研究者主要通过转换数据分布,筛选相关数据集,迁移学习方式来解决差异性问题.



(a) 同构缺陷预测模型 (b) 异构缺陷预测模型

Fig.9 The prediction model of data difference

图 9 数据差异的缺陷预测模型

(1) 转换数据分布

在实际软件开发过程中,若新项目没有足够的标签数据,可以使用同公司其他的历史项目数据集,即存在度量元相同但数据分布不同的情况.针对该情况,可以在建立模型前对数据的分布进行调整,使得训练集与测试集具有相似的分布.Feng 等人[40]首先研究了不同的数据转换方法(log、rank 及 Box-Cox)对跨项目缺陷预测的影响,发现 3 种方法的影响是相似的,然后整合这 3 种数据转换方法后提出了多数据转换方法 MT(multiple transformations),更进一步地,使用 Box-Cox 转换的参数来确定每个测试集最适合的训练集,即 MT+方法.Li 等人[41]通过对数据进行标准化和正态化(log)处理来解决数据差异性问题,然后使用无监督学习方法 SNN(shared nearest neighbors)提取静态代码度量元的相似模式.

(2) 筛选相关数据集

针对特征集相同但数据分布不同的情况,在建立模型前,部分研究者通过算法对训练数据集进行实例筛选,筛选出与被测项目相似性高的数据集进行训练.Seyedrebvar 等人[43]提出了基于搜索的实例选择方法 GIS(genetic instance selection),使用遗传算法进行搜索,并且以收敛目标来指导实例的选择过程,从而更精确地匹配测试集的分布特征.每代的适应度函数通过 NN-filter 得到的确认集进行评估.而且基于这些确认集,GIS 方

法能够同时解决数据中的潜在噪音和实例选择两个问题.Steffen 等人[42]和 Tim 等人[45]比较了全局模型和局部模型对跨项目缺陷预测的影响,在局部模型中使用 EM 和 WHERE 等聚类方法对训练项目集进行聚类,实验结果表明,聚类后的数据集有更好的性能.He 等人[46]研究了各种实例选择方法,发现可以通过数据集的分布特征进行训练实例的选择.然后根据数据集的分布特征提出了一种三步法的自动实例选择方法.Peters 等人[47]针对数据差异性问题,提出了 Peters filter 方法选择训练样本集.该方法假设训练集中包含了比较小的测试集的缺陷信息,因此,该方法能够为每个训练集发现距离最近的测试集实例.最后每个测试实例会选择离它最近的实例作为筛选训练集的候选.Herbold[48]根据数据集的分布,提出了两种基于距离的实例选择方法,即 EM-Clustering 和 K-nearest neighbor 策略.

(3) 迁移学习技术

迁移学习是用一个环境中学到的知识来帮助新环境中的学习任务,正好可以解决缺陷预测中数据分布差异问题[49,51].根据被测项目和源项目之间的度量元空间是否相同,迁移学习可以分为同构迁移学习和异构迁移学习,其中,同构迁移又可分为基于实例的迁移学习及基于特征的迁移学习等[83].

- 基于实例的迁移学习.根据被测项目中部分标签的缺陷数据,对源域项目实例的权值进行调整,使得源域与目标域进行匹配.Ma 等人[49]提出了 TNB(transfer Naïve Bayes)模型,首先计算测试集每个特征的最大值和最小值,然后依据数据引力计算训练集实例的权重,利用贝叶斯网络对这些加权重的训练集进行训练,实验设计与直接使用所有的训练集和使用 NN-filter 方法进行比较,实验结果表明,TNB 提高了模型的性能.Ryu 等人[72]提出了基于代价敏感的迁移学习方法,对于源项目中与目标项目有相同分布的实例分类错误时增加权重,分类正确的实例减少权重;对于分布不同的实例分类错误时减少一定的权重,分类正确的实例减少更多的权重.Chen 等人[17]提出了双迁移 Boosting(double transfer boosting,简称 DTB)方法来减少消极的数据实例.首先利用 NN 和 SMOTE 对源项目数据集进行处理,然后利用数据引力对选择的数据集赋予权重,利用 Boosting 方法消减小数据集的权重.
- 基于特征的迁移学习.主要是通过特征变换,将源项目和被测项目的度量元集变换到相同的空间.缺陷预测模型中最常用的是迁移主成分分析(transfer component analysis,简称 TCA).Cao 等人[33]使用迁移主成分分析(TCA)转换训练数据集数据,使得训练数据集具有与测试数据集相似的数据分布.
- 异构迁移学习.这是源项目和被测项目之间的度量元空间不同,而且特征的维度也可能不同的学习方法.Yu 等人[50]针对跨项目缺陷预测模型的异构问题,首先对源项目和目标项目进行特征选择,然后利用特征匹配迁移匹配特征,消除数据集的差异性.Feng 等人[84]采用无监督学习中的谱聚类方法进行跨项目缺陷预测,实验结果表明能取得较好的结果.

这 3 种方法的假设都是目标任务和源任务的目标都是对缺陷的倾向性进行预测.其中,基于实例的迁移学习和基于特征的迁移学习的假设是源项目和被测项目的度量元集相同,数据分布不同,而异构迁移学习的前提是源项目和被测项目的度量元是不同的.在使用这 3 种方法时,如果假设得不到满足,则这类方法的有效性得不到保障.在特征迁移和异构迁移方法中,源项目向被测项目迁移,或者被测项目向源项目迁移,与源项目和被测项目迁向共同的空间之间存在着差异性.Li 等人[85]提出了多核的集成学习方法解决异构缺陷预测,实验发现,源项目向被测项目转换的性能优于后两个的性能.基于上述研究工作,在后续的研究中需要深入地分析全局和局部对预测模型性能的影响.

3 缺陷预测模型构建方法

本文通过对收集的 117 篇文献进行研究和分析,根据预测形式可以将缺陷预测模型大体上分为缺陷倾向性、缺陷严重度及缺陷数量 3 种类型,具体的分类方法及文献分布如图 10 所示.

3.1 缺陷倾向性预测

缺陷倾向性预测是把需要预测的软件模块分类为有缺陷或者无缺陷,是一个二分类问题,也是构建最多的缺陷预测模型.大量的研究者已经使用了不同的分类技术来构建缺陷倾向性预测模型,主要包括统计学方法和

机器学习方法.

3.1.1 机器学习方法

大部分的研究都是基于机器学习方法来进行缺陷倾向性预测.具体可以分为有监督缺陷预测、无监督缺陷预测和半监督缺陷预测,下面我们分别介绍应用到缺陷倾向性预测中的方法.

(1) 有监督学习方法

该方法是利用已有标签的有/无缺陷样本集优化分类器的参数,使其达到最佳预测效果的过程.在 117 篇文献中,应用到的有监督学习主要有:贝叶斯[14,16,65,86,87]、决策树[39,88]、集成学习[17,34−36,57,72,77,89−92]、神经网络[33,93−95]、支持向量机[59,96,97]、逻辑回归[64]及字典学习[66]等.接下来在文献[5]的基础上,对近两年的有监督的学习方法进一步地加以总结和分析.

Issam 等人[34]将特征选择和集成学习相结合,提出了平均概率集成(average probability ensemble,简称 APE)学习方法,其中包含了 Random Forests(RF)、Gradient Boosting(GB)、Stochastic Gradient Descent(SGD)、Weighted SVMs(W-SVMs)、Logistic Regression(LR)、Multinomial Naïve Bayes(MNB)及 Bernoulli Naïve Bayes(BNB)这 7 种分类器.在 APE 模型里,每个分类器都预测出软件模块属于缺陷的概率,然后对这 7 个概率取平均值作为最后的输出.

Yang 等人[35]提出了结合决策树和集成学习的双层集成学习方法 TLEL(two-layer ensemble learning)来构建预测动态缺陷预测模型.在内层使用基于决策树的 bagging 集成学习方法来构建随机森林模型,在外层使用随机抽样的方法来训练不同的随机森林模型,然后按照 stacking 方式来集成这些不同的随机森林模型.

Bowes 等人[89]提出了多样性选择的集成学习方法.该方法首先选择来自 4 个不同家族的分类器 NB、C4.5、*K*-NN 及序列最小化作为基础分类器,并且设计了加权精度多样性 WAD(weighted accuracy diversity),然后依据 WAD 使用 stacking 技术集成一个强分类器.

Xia 等人[92]针对跨项目的缺陷预测问题提出了混合重构方法 HYDRA(hybrid model reconstruction approach).该方法包括两个阶段:遗传算法阶段和集成学习阶段.在遗传算法阶段,首先将每个带标签的源训练集与带标签的测试集结合构建多个分类器,然后利用遗传算法,为每个分类器分配不同的权值构建 GA 分类器.在集成学习阶段,即多次迭代 GA 阶段,每次迭代产生一个 GA 分类器,然后根据每个 GA 分类器的错误率分配相应的权重.经过两阶段后就建立了大量的分类器,然后将这些分类器集成进行测试集的缺陷预测.

(2) 半监督学习方法

该方法是将有监督学习与无监督学习相结合的一种学习方法.该方法在使用大量的未标记缺陷数据的同时使用了标记的缺陷数据,来进行缺陷预测.Zhang 等人[98]提出了基于图的半监督学习方法.首先根据拉普拉斯算子得分从无缺陷的训练集中抽样样本,构建一个类平衡的标签训练集.然后用非负稀疏算法来计算关系图的非负稀疏权值作为聚类指标.最后在非负稀疏图上使用一个标签传播算法来迭代预测未标记的软件模块的标签.Wu 等人[99]使用半监督结构字典学习方法 SSDL(semi-supervised dictionary learning)构建项目内和跨项目的缺陷预测总体框架.该方法学习了总字典,以及有缺陷、无缺陷、未标记模型 3 个子字典,从而充分提取了缺陷数据的有用信息.

He 等人[100]提出了半监督学习方法 extRF.该方法通过自我训练模式扩展了有监督的随机森林算法.首先从数据集中抽取小部分数据作为有标签的数据集,使用这一部分数据集训练随机森林分类器,根据训练好的分类器来预测未标记的数据集,然后选择最自信的样本加入到训练集中,并进行训练初始模型,以此类推,最后形成精炼模型.

(3) 无监督学习方法

当被测项目为无任何标记的有/无缺陷数据集时,可采用无监督学习方法.该方法没有导师,需要学习器自身形成(form)和评价(evaluate)概念.Feng 等人[84]采用无监督学习中的谱聚类方法进行跨项目缺陷预测.并且,将该方法和 5 个经常使用的有监督学习器 RF、NB、LR、DT 及逻辑模型树进行比较,发现使用谱聚类的无监督学习方法在跨项目和项目内场景下都有可比较性.Fu 等人[101]重新验证了无监督学习对动态缺陷预测模型的影响,发现并不是所有的无监督学习都优于有监督学习,提出了 OneWay 方法来自动选择潜在的最好的方法.

Rodrigo 等人[102]提出了结合粒子群算法的基于聚类集成方法的缺陷预测模型.Park 等人[103]使用无监督学习方法(EM 和 Xmeans)方法进行缺陷倾向性预测.Muhammed 等人[104]针对网络软件系统,使用无监督学习的 *K*-means++进行缺陷预测.Bishnu 等人[105]使用基于四叉树的 *K*-means 聚类算法进行缺陷倾向性预测.

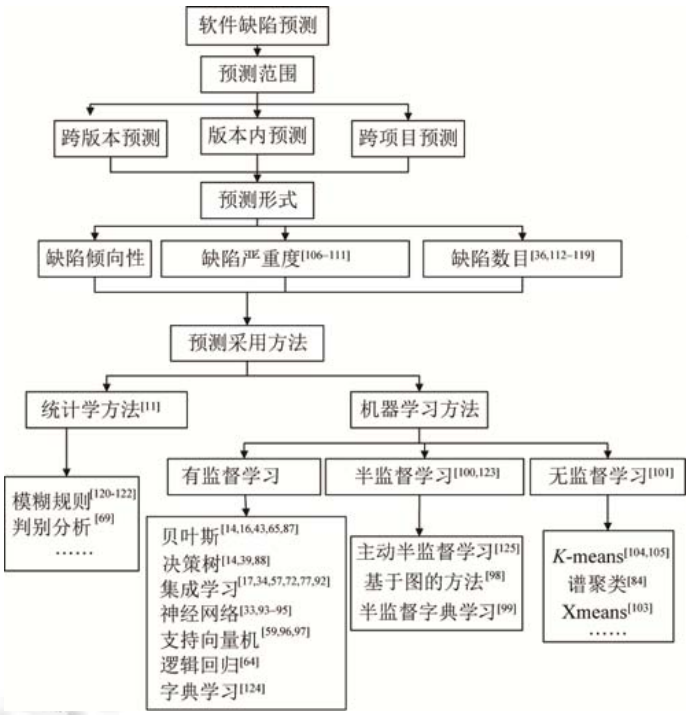


Fig.10 The overview of the learners

图 10 学习器分布总览

3.1.2 统计学方法

统计学方法是收集、整理、分析和解释统计数据,并给出一定结论的方法.在缺陷倾向性预测中统计学方法也有一定研究.Shepperd 等人[11]使用统计学的方法分析了分类器、数据集、输入度量元以及研究团队 4 个因素对缺陷预测模型的性能影响,实验结果表明,研究团队的影响较大.Pradeep 等人[120]针对缺陷倾向性提出了模糊规则,根据使用的特征比例选择有用的特征.Chang 等人[121]提出了基于行为的缺陷模型,利用行为修正建议(action correction recommendation,简称 ACR)构建低缺陷行为和高缺陷行为,预测算法使用负关联规则挖掘技术.Zhang 等人[122]提出了代价效益规则,利用规则进行缺陷倾向性预测,并提出了 $FN/(FN+TN)$的新的性能评价指标.

一般地,当有标签的缺陷数据集充足时,使用有监督学习要优于无监督学习和半监督学习;当有部分标签缺陷数据集时,可以使用半监督学习;当无任何缺陷标签数据集时,使用无监督学习.除此之外,有些研究者使用不同的方法,比较分类器的不同分类效果,但是,由于项目内在特征及评价指标的不同,没有哪个分类器对所有的数据集都能产生好的结果[25,37,40,41,44,46,47,53,60,67,76,107,126-128],但是,在比较的时候,对于分类器的参数一般采用的是默认值,所以,在后续的工作中需要深入地分析分类器方法中参数取值对缺陷预测模型性能的影响.

3.2 缺陷数预测

软件缺陷数预测是通过软件历史数据对软件模块的缺陷数量进行预测,一般地,程序模块针对的是类级别或文件级别.近年来,一些研究者在这个领域进行了研究,大部分研究者使用的是逻辑回归算法和简单的神经网络算法进行预测.

Bell 等人[118]分析了开发人员对缺陷预测模型的影响,使用标准的负二项回归分类器预测文件缺陷的数量.Andreou 等人[119]提出了随机信念泊松回归网络(stochastic belief Poisson regression network,简称 SBPRN)来计算缺陷数,通过引入双随机齐次泊松过程模型,受 SBN 公式的启发,在每个时间点的失效 log-rate 参数进行建模.Santosh 等人[36]提出将以前版本的数据作为依据,利用集成学习技术预测现在版本的软件模块的缺陷数.首先,把数据集按照 6、2、2 分成训练集、确认集和测试集,之后,利用 meta-training 重新训练 3 个分类器(线性规划、遗传算法、多层感知器),用训练好的分类器对确认集进行训练,得到相应的值,然后,利用整合函数得到最后的缺陷数.Tanvi 等人[114]利用人工神经网络预测缺陷的数量,并且与模糊逻辑回归方法进行比较,实验结果表明,该方法的性能有很大提高.Ratgore 等人[112]比较了 6 个常用的预测缺陷数量的技术,包括归零校正"泊松回归"和负二项回归,结果表明,DTR(decision tree regression)、GP(genetic programming)、多层感知器以及 LR 在所有考虑的项目中都能得到较好的性能.

3.3 缺陷严重度预测

研究者近年来对缺陷严重性预测进行了一定的研究,大多采用逻辑回归方法进行预测.You 等人[106]提出了采用多逻辑回归方法(multiple linear regression,简称 MLR)和随机梯度下降法来预测缺陷的严重度,进而以此对软件模块进行排序,指导测试资源的合理分配.Nguyen 等人[107]针对缺陷的严重度提出了一种基于排序的方法,根据软件模块的缺陷数来对软件模块的严重度排序,通过实验表明,该方法提高了缺陷预测的性能.Yang 等人[108]将 LTR(learning to rank)方法引入到缺陷预测模型中,组合不同的评估指标来优化缺陷的严重度,实验结果表明了方法的有效性.Yadav 等人[109]使用模糊逻辑提取规则,预测缺陷的严重度,其中,严重度的确定根据缺陷的数量来判断.You 等人[110]针对跨项目缺陷预测的严重度问题,提出了利用单目标的多线性回归方法,分类器使用 Linear Regression(LR)、RankSVM、RankBoost,严重度根据 bug 的个数进行排序.

3.4 其 他

大部分的研究者主要对缺陷倾向性、缺陷数和缺陷严重度进行了建模,除此之外,部分研究者还研究了其他预测模型.Borg 等人[129]利用 *K*-means 聚类方法预测缺陷修复时间.Parizy 等人[130]基于硬件设计资源库(私有项目)的缺陷预测,来预测缺陷的等级.Gupta 等人[131]针对缺陷预测的类重叠和噪音问题,首先将 54 个软件项目整理为 327 个二类的数据集,然后利用数据复杂度对数据集进行重叠度计算,并且对 207 个重叠度高的数据集计算实例的重叠度,最后在 327 个 Promise 数据集上利用 *K*NN、SVM、NB 和 C4.5 分别进行有重叠实例和没有重叠实例的计算,实验结果表明,去除重叠的数据集有更好的预测性能.Rathore 等人[132]提出了一个推荐系统,该系统考虑了影响缺陷性能的 10 种因素,从 10 种因素提取相应的规则,依据项目的实际需求选择合适的技术进行缺陷预测.

综上所述,研究者采用机器学习的方法对程序模块的缺陷倾向性进行了大量研究,但对于程序模块粗粒度的缺陷数和缺陷严重度的研究较少,而缺陷数和缺陷严重度的研究更能指导实际的软件开发和测试工作,所以,在后续的工作中需进一步加强对缺陷数或者缺陷严重度模型的研究.

4 性能评价指标

性能评价指标是对构建的缺陷预测模型的性能进行评价,以此来判断构建的缺陷预测模型的好坏.针对不同的缺陷预测模型可以分为不同的评价指标,大体上可以分为缺陷倾向性评价指标、缺陷数分析评价指标和缺陷严重度评价指标.但是目前还没有一套通用的评价标准来衡量不同的缺陷预测模型的性能,大部分研究者使用多种性能指标进行评估.

4.1 缺陷倾向性评价指标

软件缺陷倾向性预测是预测软件模块为有缺陷模块还是无缺陷模块,是一个二分类问题.因此,可以使用机器学习中分类问题的模型评价指标,混淆矩阵是评价分类模型好坏的展示工具,具体见表 1.矩阵的每一行表示实例的真实情况,矩阵的每一列表示预测模型预测的实例情况.

矩阵元素的具体含义分别是:

True Positive(TP):表示实例的真实类别是有缺陷模块,预测模型的预测结果也是有缺陷模块.

False Negative(FN):表示实例的真实类别是有缺陷模块,预测模型的预测结果是无缺陷模块.

Table 1 Confusion matrix

表 1 混淆矩阵

	预测为有缺陷	预测为无缺陷
真实有缺陷	True positive	False negative
真实无缺陷	False positive	True negative

False Positive(FP):表示实例的真实类别是无缺陷模块,预测模型的预测结果是有缺陷模块.

True Negative(TN):表示实例的真实类别是无缺陷模块,预测模型的预测结果是无缺陷模块.

根据混淆矩阵衍生了各种评价指标,其中,用于缺陷预测的主要有 Accuracy、Precision、Recall、*F*-measure、*G*-mean、AUC、MCC 等.

(1) Accuracy:表示预测模型的精确度,是预测模型预测正确的个数与数据集中实例总个数的比值.公式为

$$Accuracy=\frac{TP+TN}{TP+FN+FP+TN} \tag{1}$$

在软件缺陷预测中,精确度表示被正确分类的软件模块的比例,但是,这个度量指标有些含糊不清,没有说明有/无缺陷的错误分类的信息.

(2) Precision:表示查准率,是预测模型预测为有缺陷实例中真实类别为有缺陷所占的比例.公式为

$$Precision=\frac{TP}{TP+FP} \tag{2}$$

缺陷预测中的精确度表示了正确分类的缺陷模块与预测为有缺陷模块的比例,一般情况下,Precision 越高,说明预测模型的效果越好.

(3) Recall:表示召回率,是预测模型预测为有缺陷的实例的数量占真实有缺陷的实例数的比例.公式为

$$Recall=\frac{TP}{TP+FN} \tag{3}$$

Recall 对软件缺陷预测非常重要,越早地找到有缺陷的特别是高危的模块,对软件越好.一般情况下,Recall 越高,说明有更多的有缺陷的模块被模型预测正确,预测模型的效果越好,但是,较高的召回率往往是以降低精确度为代价的.

(4) pf:表示假阳率,是真实为无缺陷模块被预测为有缺陷模块占真实无缺陷模块的比例.其公式表示为

$$pf=\frac{FP}{TN+FP} \tag{4}$$

当然,pf 值越小,表明模型的性能越好.因为 pf 的值过大,表明越多的无缺陷模块被误分为有缺陷,则需要投入很多的资源到无缺陷的模型进行测试,造成资源的浪费.

(5) *F*-measure:是一个综合评价指标,提供了召回率和精确度之间的权衡,具体公式如下:

$$F\text{-}measure=\beta\times\frac{Precision\times Recall}{Precision+Recall} \tag{5}$$

(6) *G*-mean:几何平均数,能够评价类不平衡数据的表现.对于软件缺陷数据来说,有缺陷的模块占少数,无缺陷的模块占多数,所以有类不平衡的问题.公式如下:

$$G\text{-}mean=\sqrt{Precision\times Recall} \tag{6}$$

(7) AUC:表示 ROC 曲线下的面积.ROC 曲线叫作接受者工作特征曲线,其横坐标为假阳性率,纵坐标为 Recall.一般情况下,AUC 值越大,说明缺陷预测模型的性能越好[95].

(8) MCC(Matthews correlation coefficient):表示实际分类与预测分类之间的相关系数.该指标同时考虑了 TP、FN、FP 及 TN,是一个相对均衡的指标,能够评价类不平衡数据的表现[133].公式如下:

$$MCC=\frac{TP\times TN-FP\times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{7}$$

一般地,Accuracy、Precision、Recall、*F*-measure、*G*-mean、AUC 的取值范围在[0,1],而 MCC 的取值范围是[–1,1].在缺陷数据的类不平衡情况下,Accuracy 指标有很大的缺陷(如有缺陷的模块只占 2%时,Accuracy 等于 98%,但无意义);Precision 是检索出来的缺陷模块有多少是准确的,Recall 是所有有缺陷的模块有多少被检索出来, *F*-measure 和 *G*-mean 是用来权衡 Precision 及 Recall 两个指标的,属于综合评价指标,Precision、Recall、*F*-measure 及 *G*-mean 对类分布的改变较敏感;AUC 不受类分布的影响,适合评估类分布不平衡的数据集;MCC 是一个较均衡的指标,即使缺陷数据集类分布严重不平衡,也可以使用 MCC.

4.2 缺陷数评价指标

缺陷数的预测正好与缺陷倾向性相反,评价的是连续的数值.具体的评价指标有:AAE、ARE、Measure of Completeness.

(1) AAE(average absolute error):平均绝对误差,表示预测值与实际值之间的距离,在缺陷预测中表示预测的值与实际值之前的差值,其被定义为

$$AAE=\frac{1}{n}\sum_{i=1}^{n}|f_i-y_i| \tag{8}$$

(2) ARE(average relative error):平均相对误差,表示预测的值与被测的实际值的比值,首先需要确定绝对误差来计算相对误差,具体的公式为

$$ARE=\frac{1}{n}\sum_{i=1}^{n}|f_i-y_i|/(y_i+1)| \tag{9}$$

(3) Measure of Completeness:完整性测量,用来表示模型在预测过程中的完整程度,在软件系统中,它是测量实际的缺陷值与预测缺陷值的一个比值.具体公式为

$$Completeness=\frac{f_i}{y_i} \tag{10}$$

如果该值接近 100%,表示这个模型可以接受,如果高于 100%,表示该预测模型具有很高的假阳性率,需要额外的努力来修复错误,如果值小于 100%,表示预测模型没有预测到一些错误.

4.3 缺陷严重度评价指标

缺陷严重度预测主要是给开发人员反馈一个缺陷严重度的推荐列表,具体的评价指标有:MAP、FPA、CE.

(1) MAP(mean acerage precision):平均精度均值,表示每个类别的平均准确率的平均值.预测出来的严重度高的缺陷模块越靠前,MAP 就越高.具体公式为

$$MAP=\frac{1}{2}\sum_{q=0}^{1}AP(q) \tag{11}$$

其中,q 表示类别,0 表示无缺陷,1 表示有缺陷.$AP(q)$表示不同查全率的点上的正确率的平均.

(2) FPA(fault percentile average):平均缺陷百分比,表示 Top_m 模块中实际缺陷占整个系统缺陷的比例的平均值[122].FPA 的值越大,表明排序越好.具体公式为

$$FPA=\frac{1}{k}\sum_{m=1}^{k}\frac{1}{n}\sum_{i=k-m+1}^{k}n_i \tag{12}$$

其中,k 表示模块的数量,n_i 表示根据预测的排序 i 模块的实际缺陷数,n 表示 k 个模型总的缺陷数.

(3) CE(cost effectiveness):成本效益,该指标是基于源代码行数的 Alberg 图的概念.在 Alberg 图中,X 轴表示从排序模块中选择模块的累积代码行数百分比,Y 轴表示选择模块发现缺陷数的累积百分比[134].$CE_\pi(m)$的取值范围为[–1,1],该值越大,代表该模型效果越好.其计算公式为

$$CE_{\pi}(m)=\frac{Area_{\pi}(m)-Area_{\pi}(random\ model)}{Area_{\pi}(optimal\ model)-Area_{\pi}(random\ model)} \tag{13}$$

其中,$Area_{\pi}(m)$是被评价模型的曲线面积,$Area_{\pi}$(random model)是随机选择方法的曲线面积,$Area_{\pi}$(optimal model)是根据实际项目排序的曲线面积.

本文对收集的 117 篇文献中的性能评价指标的使用次数进行了统计,得到的结果见表 2,从表中可以看出,对于缺陷倾向性预测,经常使用的性能评价指标是:*F*-measure、Precision、Recall 及 AUC.其中,Recall 为预测的实际有缺陷模块的比例,当该值很高时,表明实际有缺陷模块被挖掘;*F*-measure 及 AUC 为综合评价指标,能够权衡 Precision 及 Recall 之间的关系,当二者都很高时,表明这个模型性能好,所以使用比较多.MCC 是同时考虑了 4 个指标的评价指标,所以在后期的实验比较中,加入 MCC 更具有说服力.

Table 2 The statistics of performance measures

表 2 性能评价指标统计

指标	累计使用次数
Accuracy	54
Precision	108
pf	15
F-measure	94
G-mean	5
Recall	109
AUC	112
MCC	4
AAE	9
ARE	9
Measure of completeness	1
MAP	1
FPA	3
CE	2

5 总 结

软件缺陷预测(SDP)在测试过程之前预测软件缺陷,从而有助于更好地利用测试资源,保证软件质量(SQA).因此,软件缺陷预测近年来吸引了研究人员的注意.本文总结了软件缺陷预测的相关工作,如缺陷度量元、数据集问题、缺陷预测方法及性能评价指标.从整理分析的文献中可以看出,大部分的工作都集中在使用公共数据集的面向对象度量元和过程度量元上,而且主要是利用机器学习和统计方法来进行缺陷倾向性预测;在数据质量问题上,高维度和类不平衡问题得到了广泛的研究;大部分的工作是使用 *F*-measure、Precision、Recall 及 AUC 等指标来评估模型的性能.当然,软件缺陷预测模型还存在着大量值得进一步研究的问题.

(1) 基于演化的缺陷预测模型值得关注

目前,大多数的软件缺陷预测模型都是基于需求、设计等的文档和代码等相关的度量元,但是,软件产品为了适应需求的变化,不断演化.在每次演化的过程中,一方面可能由于新增需求而新增加类代码,另一方面可能由于需求变化而变动已有的类代码,这些变更都会相应地引入新的缺陷.因此,需要根据软件产品的演化轨迹,挖掘出一些与演化有关的度量元,完善演化度量元集,并应用在缺陷预测模型中.

一方面,可以分析不同粒度的演化度量元对演化缺陷预测模型的影响.粗粒度一般考虑的是事务或者文件级别的变更,通常针对多个语句进行修改,所以,在该级别预测出引入的缺陷,无法准确定位.细粒度一般是类或者语句级别,能够更精确地定位缺陷,缩小测试和审查范围.

另一方面,提取演化度量元时可以考虑开发人员的经验.有研究者指出,开发人员对软件项目的熟悉程度关系到变更的质量[135,136].因此,提取演化度量元时需要关注人的因素,需要分析开发人员的经验对演化缺陷预测模型的影响.

(2) 缺陷数预测或缺陷严重度预测模型值得关注

大部分研究者对软件缺陷倾向性预测进行了大量研究,只有少数研究人员专注于缺陷数或者缺陷严重度

度量.但是,从软件开发者的角度来看,他们更想知道哪个模块有更多的软件缺陷,而不是这个模块有没有缺陷.这样,他们就可以尽早地快速识别大部分的缺陷.所以,需要构建更多的缺陷数、缺陷严重度的预测模型,并对这些模型提供合理的评价指标.

一方面,目前的研究大都集中在使用逻辑回归的方法进行预测.深度学习方法具有拟合任何复杂函数的特点,可以进行非常复杂的非线性映射.因此,可以将多种深度学习方法进一步引入到缺陷数或严重度的预测中,发现软件内部度量元与缺陷数或严重度的非线性映射,从而形成具有良好效果的预测模型.

另一方面,目前的缺陷预测模型的评价指标多集中在缺陷倾向性模型的评价上,缺陷数和缺陷严重度的评价指标较少,而且目前的评价指标大多是从缺陷的个数加以考虑,因此,设计出更合理的综合评价指标更有意义.

(3) 异构缺陷预测模型值得关注

传统的缺陷预测模型需要源项目和目标项目具体同分布,即只能进行同项目之间的预测.但在公司的早期,没有这么多的历史数据,所以需要使用不同项目或者不同公司的历史数据来构建异构缺陷预测模型.

一方面,需要进一步引入异构迁移学习领域内的最新研究成果,转换特征集,缩小两者之间的差异,构建效果好的异构缺陷预测模型.

另一方面,研究不同的转换方向对异构缺陷预测模型的影响.在进行转换时可以是源域向目标域靠近,也可以是源域和目标域转变到共同的特征空间集,具体哪种方式可以得到更好的性能需要进一步加以研究.

(4) 私有/商业项目的数据脱敏问题

目前,大部分的缺陷预测模型的研究是基于公共数据库的,因为私有/商业项目涉及到公司内部隐私问题,不能公开,但是公共数据库和各个公司内部的某些度量元可能会不同,所以,可以对某些私有/商业的项目进行去隐私化处理,这样既能进行缺陷预测,也能不泄露内部机密.Peters 等人[126]在这方面已进行了一些研究,提出了 CLIFF+MORPH 的方法来去除隐私化.目前对该问题的研究工作较少,若能对该问题进行深入研究,则可以给商业项目提供更好的缺陷预测模型.

致　谢:在此,我们对审稿人和编辑表示感谢,对给本文提出建议的同行表示感谢.

References:

[1] Wang Q, Wu SJ, Li MS. Software defect prediction. Ruan Jian Xue Bao/Journal of Software, 2008,19(7):1565–1580 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/1565.htm [doi: 10.3724/SP.J.1001.2008.01565]

[2] Li Y, Huang ZQ, Wang Y, Fang BW. Survey on data driven software defects prediction. Acta Electronica Sinica, 2017, 45(4):982–988 (in Chinese with English abstract).

[3] Chen X, Wang LP, Gu Q, Wang Z. A survey on cross-project software defect prediction methods. Chinese Journal of Computers, 2018,(1):254–274 (in Chinese with English abstract).

[4] Rathore SS, Sandeep K. A study on software fault prediction techniques. Artificial Intelligence Review, 2017,5:1–73.

[5] Chen X, Gu Q, Liu WS, Liu SL, Ni C. Survey of static software defect prediction. Ruan Jian Xue Bao/Journal of Software, 2016,27(1):1–25 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4923.htm [doi: 10.13328/j.cnki.jos.004923]

[6] Halstead MH. Elements of Software Science (Operating and Programming Systems Series). New York: Elsevierence, 1977.

[7] Jiang T, Tan L, Sunghun K. Personalized defect prediction. In: Proc.of the Automated Software Engineering. Piscataway, 2014, 19(17):279–289.

[8] Shin Y, Williams L. Can traditional fault prediction models be used for vulnerability prediction. Empirical Software Engineering, 2013,18(1):25–59.

[9] Öztürk MM. Which type of metrics are useful to deal with class imbalance in software defect prediction. Information and Software Technology, 2017,92(12):17–29.

[10] Madeyski L, Marian J. Which process metrics can significantly improve defect prediction models: An empirical study. Software Quality Journal, 2015,23(3):393–422.

[11] Shepperd M, Bowes D, Hall T. Researcher bias: The use of machine learning in software defect prediction. IEEE Trans. on Software Engineering, 2014,42(11):1092–1094.

[12] Chakkrit T, Shane M, Hassan AE, Kenichi M. Comments on “Researcher bias: The use of machine learning in software defect prediction”. IEEE Trans. on Software Engineering, 2016,42(11):1092–1094.

[13] Shin Y, Meneely A, Williams L, Osborne JA. Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. IEEE Trans. Software Engineering, 2011,37(6):772–787.

[14] Okutan A, Yıldız OT. Software defect prediction using bayesian networks. Empirical Software Engineering, 2014,19(1):154–181.

[15] Feng Z, Hassan AE, Shane M, Ying Z. The use of summation to aggregate software metrics hinders the performance of defect prediction models. IEEE Trans. on Software Engineering, 2017,43(5):476–491.

[16] Feng Z, Audris M, Keivanloo I, Ying Z. Towards building a universal defect prediction model. In: Proc.of the Mining Software Repositories. 2014. 182–191.

[17] Chen L, Fang B, Shang Z, Tang Y. Negative samples reduction in cross-company software defects prediction. Information and Software Technology, 2015,62(1):67–77.

[18] Mccabe TJ. A complexity measure. IEEE Trans. on Software Engineering, 1976,se-2(4):308–320.

[19] Rodriguez D, Ruiz R, Riquelme JC, Harrison R. A study of subgroup discovery approaches for defect prediction. Information and Software Technology, 2013,55(10):1810–1822.

[20] Kamei Y, Shihab E, Adams B, Hassan AE, Mockus A, Sinha A, Ubayashi N. A large-scale empirical study of just-in-time quality assurance. IEEE Trans. on Software Engineering, 2013,39(6):757–773.

[21] Herzig K, Sascha J, Andreas Z. The impact of tangled code changes on defect prediction models. Empirical Software Engineering, 2016,21(2):303–336.

[22] Kamei Y, Fukushima T, Mcintosh S, *et al*. Studying just-in-time defect prediction using cross-project models. Empirical Software Engineering, 2016,21(5):2072–2106.

[23] Caglayan B, Misirli AT, Bener AB, Miranskyy A. Predicting defective modules in different test phases. Software Quality Journal, 2015,23(2):205–227.

[24] Bhattacharya P, Iliofotou M, Neamtiu I, *et al*. Graph-based analysis and prediction for software evolution. In: Proc.of the Software Engineering. Zurich, 2012. 419–429.

[25] Lee T, Nam J, Han D, Kim S, *et al*. Developer micro interaction metrics for software defect prediction. IEEE Trans. on Software Engineering, 2016,42(11):1015–1035.

[26] Ekanayake J, Tappolet J, Gall HC, Bernstein A. Time variance and defect prediction in software projects: Towards an exploitation of periods of stability and change as well as a notion of concept drift in software projects. Empirical Software Engineering, 2012,17(4-5):348–389.

[27] Bansiya J, Davis C. A hierarchical model for object-oriented design quality assessment. IEEE Trans. on Software Engineering, 2015,28(28):4–17.

[28] Chidamber SR, Kemerer C. A metrics suite for object oriented design. IEEE Trans. on Software Engineering, 1994,20(20): 476–493.

[29] Abreu FB, Carapuca R. Candidate metrics for object-oriented software within a taxonomy framework. Journal of Systems and Software, 1994,26(1):87–96.

[30] Gray D, Bowes D, Davey N, Sun Y, Christianson B. The misuse of the nasa metrics data program datasets for automated software defect prediction. In: Proc. of the Evaluation and Assessment in Software Engineering (EASE 2011). Durham, 2011. 96–103.

[31] Aggarwal CC. Outlier Analysis. Springer Publish Company, Incorporated, 2015. 75–99.

[32] Ryu D, Okjoo C, Jongmoon B. Improving prediction robustness of VAB-SVM for cross-project defect prediction. In: Proc. of the Computational Science and Engineering. Chengdu, 2015. 994–999.

[33] Cao QM, Sun Q, Cao QH, Huobin T. Software defect prediction via transfer learning based neural network. In: Proc. of the Reliability Systems Engineering. Beijing, 2016. 1–10.

[34] Laradji IH, Alshayeb M, Ghouti L. Software defect prediction using ensemble learning on selected features. Information and Software Technology, 2015,58(2):388–402.

[35] Yang X, Lo D, Xia X, Sun J. TLEL: A two-layer ensemble learning approach for just-in-time defect prediction. Information and Software Technology, 2017,87(7):206–220.

[36] Rathore SS, Kumar S. Towards an ensemble based system for predicting the number of software faults. Expert Systems with Applications, 2017,82(10):357–382.

[37] Öztürk MM. Which type of metrics are useful to deal with class imbalance in software defect prediction. Information and Software Technology, 2017,92(12):17–29.

[38] Lu HH, Bojan C, Mark C. Software defect prediction using semi-supervised learning with dimension reduction. In: Proc. of the Automated Software Engineering. Essen, 2012. 314–317.

[39] Lu HH, Ekrem K, Bojan C. Defect prediction between software versions with active learning and dimensionality reduction. In: Proc. of the Software Reliability Engineering. Naples, 2014. 312–322.

[40] Feng Z, Iman K, Zou Y. Data transformation in cross-project defect prediction. Empirical Software Engineering, 2017,22(3): 186–3218.

[41] Li LF, Hareton L. Mining static code metrics for a robust prediction of software defect-proneness. In: Proc. of the Empirical Software Engineering and Measurement. Banff, 2011. 207–214.

[42] Steffen H, Alexander T, Jens G. Global vs. local models for cross-project defect prediction: A replication study. Empirical Software Engineering, 2017,(22):1866–1902.

[43] Seyedrebvar H, Burak T, Mika M. A benchmark study on the effectiveness of search-based data selection and feature selection for cross project defect prediction. Information and Software Technology, 2017,1–17.

[44] Yu Q, Jiang SJ, Qian JY. Which is more important for cross-project defect prediction: Instance or feature. In: Proc. of the Software Analysis, Testing and Evolution. Kunming, 2016. 90–95.

[45] Tim M, Andrew B, David C, Andrian M, Lucas L. Local versus global lessons for defect prediction and effort estimation. IEEE Trans. on Software Engineering, 2013,39(6):822–834.

[46] He Z, Shu F, Yang Y, Li M, Wang Q. An investigation on the feasibility of cross-project defect prediction. Automated Software Engineering, 2012,19(2):167–199.

[47] Peters F, Marcus MT. Better cross company defect prediction. In: Mining Software Repositories. 2013. 409–418.

[48] Herbold S. Training data selection for cross-project defect prediction. In: Proc. of the Predictive Models in Software Engineering. Baltimore, 2013. 1–10.

[49] Ma Y, Luo GH, Zeng X, Chen AG. Transfer learning for cross-company software defect prediction. Information and Software Technology, 2012,(54):248–256.

[50] Yu Q, Jiang S, Zhang Y. A feature matching and transfer approach for cross-company defect prediction. Journal of Systems and Software, 2017,132(10):366–378.

[51] He Q, Li B, Shen B, Yong X. Cross-project software defect prediction using feature-based transfer learning. In: Proc. of the Asia-pacific Symp. on Internetware. Wuhan, 2015. 74–82.

[52] Khoshgoftaar TM, Gao K, Napolitano A. An empirical study of feature ranking techniques for software quality prediction. Int'l Journal of Software Engineering and Knowledge Engineering, 2012,22(2):161–183.

[53] He P, Li B, Liu X, Chen J, Ma YT. An empirical study on software defect prediction with a simplified metric set. Information and Software Techology, 2014,59(C):170–190.

[54] Sadhana T, Birmohan S, Manpreet K. An approach for feature selection using local searching and global optimization techniques. Neural Computing & Applications, 2017,28(10):1–16.

[55] Chao N, Liu WS, Chen X, Gu Q, Chen DX, Huang QG. A cluster based feature selection method for cross-project software defect prediction. Journal of Computer Science and Technology, 2017,32(6):1090–1107.

[56] Gabriela C, Zsuzsanna M, Istvan GC. Software defect prediction using relational association rule mining. Information Sciences, 2014,264(183):260–278.

[57] Wang HJ, Tag HM, Khoshgofta, Jason VH, Kegab G. Metric selection for software defect prediction. Int'l Journal of Software Engineering and Knowledge Engineering, 2011,21(2):237–257.

[58] Yu X, Ma ZY, Ma CX, Gu Y, Liu RQ, Zhang Y. FSCR: A feature selection method for software defect prediction. In: Proc. of the Software Engineering and Knowledge Engineering. 2017. 351–356.

[59] Piao YJ, Piao MH, Cheng HJ, Ho SS. A new ensemble method with feature space partitioning for high-dimensional data classification. Mathematical Problems in Engineering, 2015,2(10):1–12.

[60] Song QB, Jia ZH, Shepperd M, Ying S, Liu J. A general software defect-proneness prediction framework. IEEE Trans. on Software Engineering, 2011,37(3):356–370.

[61] Marco DA, Michele L, Romain R. Evaluating defect prediction approaches: A benchmark and an extensive comparison. Empirical Software Engineering, 2012,17(4-5):531–577.

[62] Chen X, Shen YX, Cui ZQ, Ju XL. Applying feature selection to software defect prediction using multi-objective optimization. In: Proc. of the Annual Computer Software and Applications. Turin, 2017. 54–59.

[63] Juan MM, Marcelo J. A software defect-proneness prediction framework: A new approach using genetic algorithms to generate learning schemes. In: Proc. of the Software Engineering and Knowledge Engineering. 2015. 445–450.

[64] Wang HJ, Taghi M, Khoshgoftaar, Amri N. Choosing the best classification performance metric for wrapper-based software metric selection for defect. In: Proc. of the Software Engineering and Knowledge Engineering. 2014. 540–545.

[65] Ömer FA, Kürs, Ayan A. A feature dependent naïve Bayes approach and its application to the software defect prediction problem. Applied Soft Computing, 2017,59(10):197–209.

[66] Moreno JG, Raeder T, Alaiz R, Chawla NV, Herrera F. A unifying view on dataset shift in classification. Pattern Recognition, 2012,45(1):521–530.

[67] Zhang X, Song Q, Wang G, Zhang K, He L, Jia X. A dissimilarity-based imbalance data classification algorithm. Applied Intelligence, 2015,42(3):544–565.

[68] Gao K, Khoshgoftaar T. Software defect prediction for high-dimensional and class-imbalanced data. In: Proc. of the Software Engineering and Knowledge Engineering. Miami Beach, 2011. 89–94.

[69] Jing XY, Wu F, Dong XW, Xu BW. An improved SDA based defect prediction framework for both within-project and cross-project class-imbalance problems. IEEE Trans. on Software Engineering, 2017,43(4):321–339.

[70] Xia X, Lo D, Shihab E, Wang X, Yang X. ELBlocker: Predicting blocking bugs with ensemble imbalance learning. Information and Software Technology, 2015,61(5):93–106.

[71] Liu M, Miao L, Zhang D. Two-stage cost-sensitive learning for software defect prediction. IEEE Trans. on Reliability, 2014, 63(2):676–686.

[72] Ryu D, Jang JI, Jongmoon B. A transfer cost-sensitive boosting approach for cross-project defect prediction. Software Quality Journal, 2017,25(1):1–38.

[73] Ömer FA, Kürs A. Software defect prediction using cost-sensitive neural network. Applied Soft Computing, 2015,(33):263–277.

[74] Wan HY, Wu GQ, Cheng M, Huang Q, Wang R, Yuan MT. Software defect prediction using dictionary learning. In: Proc. of the Software Engineering and Knowledge Engineering. 2017. 335–340.

[75] Laradji IH, Mohammad A, Lahouari G. Software defect prediction using ensemble learning on selected features. Information and Software Technology, 2015,58(2):388–402.

[76] Sun ZB, Song QB, Zhu XY. Using coding based ensemble learning to improve software defect prediction. IEEE Trans. on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012,42(6):1806–1817.

[77] Wang T, Zhang Z, Jing X, Zhang L. Multiple kernel ensemble learning for software defect prediction. Automated Software Engineering, 2016,23(4):1–22.

[78] Ryu D, Choi O, Baik J. Value-cognitive boosting with a support vector machine for cross-project defect prediction. Empirical Software Engineering, 2016,21(1):43–71.

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

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

[81] Gao KH, Khoshgoftaar TM, Amri N. The use of ensemble-based data preprocessing techniques for software defect prediction. Int'l Journal of Software Engineering and Knowledge Engineering, 2014,24(9):1229–1253.

[82] Yu Q. Research on software defect prediction based on machine learning [Ph.D. Thesis]. Xuzhou: China University of Mining and Technology, 2017. 1−142 (in Chinese with English abstract).

[83] Zhuang FZ, Luo P, He Q, Shi ZZ. Survey on transfer learning research. Ruan Jian Xue Bao/Journal of Software, 2015,26(1):26−39 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4631.htm [doi: 10.13328/j.cnki.jos.004631]

[84] Feng Z, Quan Z, Ying Z, Ahmed EH. Cross project defect prediction using a connectivity-based unsupervised classifier. In: Proc. of the Software Engineering. Austin, 2016. 309–320.

[85] Li Z, Jing X Y, Zhu X, *et al*. Heterogeneous defect prediction through multiple kernel learning and ensemble learning. In: Proc. of the Software Maintenance and Evolution. IEEE, 2017. 91–102.

[86] Dejaeger K, Verbraken T, Baesens B. Toward comprehensible software fault prediction models using Bayesian network classifiers. IEEE Trans. on Software Engineering, 2013,39(2):237–257.

[87] Santosh S, Rathore, Sandeep K. A decision tree logic based recommendation system to select software fault prediction techniques. Computing, 2017,99(3):255–285.

[88] Shivaji S, Whitehead EJ, Akella R, Kim S. Reducing features to improve code change-based bug prediction. IEEE Trans. on Software Engineering, 2013,39(4):552–569.

[89] Bowes D, Hall T, Christianson B, Baddoo N. Building an ensemble for software defect prediction based on diversity selection. In: Proc. of the Empirical Software Engineering and Measurement. Ciudad Real, 2016. 46–55.

[90] Yang Y, Zhou Y, Liu J, Zhao Y, Lu H, Xu L, Xu B, Leung H. Effort-aware just-in-time defect prediction: Simple unsupervised models could be better than supervised models. In: Proc. of the Foundations of Software Engineering. Seattle, 2016. 157–168.

[91] Hu Y, Feng B, Mo XZ, Zhang XZ, Ngai EWT, Fan M, Liu M. Cost-sensitive and ensemble-based prediction model for outsourced software project risk prediction. Decision Support Systems, 2015,72(C):11–23.

[92] Xia X, Lo D, Pan SJ, Nagappan N, Wang X. HYDRA: Massively compositional model for cross-project defect prediction. IEEE Trans. on Software Engineering, 2016,42(10):977–998.

[93] Vipul V, Manohar L, Sureshchandar GS. A framework for software defect prediction using neural networks. Journal of Software Engineering and Applications, 2015,8(8):384–394.

[94] Elham P, Michael MR, Guenther R. Defect prediction using case-based reasoning: An attribute weighting technique based upon sensitivity analysis in neural networks. Int'l Journal of Software Engineering and Knowledge Engineering, 2012,22(6):747–768.

[95] Li J, He PJ, Zhu JM, Michael RL. Software defect prediction via convolutional neural network. In: Proc. of the Software Quality, Reliability and Security. Prague, 2017. 318–328.

[96] Miao L, Liu M, Zhang D. Cost-sensitive feature selection with application in software defect prediction. In: Proc. of the Pattern Recognition. Tsukuba, 2013. 967–970.

[97] Gao K, Khoshgoftaar TM. Software defect prediction for high dimensional and class-imbalanced data. In: Proc. of the Software Engineering and Knowledge Engineering. Eden Roc Renaissance, 2011. 89–94.

[98] Zhang ZW, Jing XY, Wang J. Label propagation based semi-supervised learning for software defect prediction. Automated Software Engineering, 2016,24(1):1–23.

[99] Wu F, Jing XY, Dong X, Cao J, Xu M. Cross-project and within-project semi-supervised software defect prediction problems study using a unified solution. In: Proc. of the Software Engineer Companion. Buenos Aires, 2017. 195–197.

[100] He Q, Shen B, Chen Y. Software defect prediction using semi-supervised learning with change burst. In: Proc. of the Annual Computer Software and Applications. Atlanta, 2016. 113–122.

[101] Fu W, Menzies T. Revisiting unsupervised learning for defect prediction. In: Proc. of the Software Engineering. Paderborn, 2017. 72–83.

[102] Coelho RA, Fabrício R, Guimarães N, Ahmed A, Esmin A. Applying swarm ensemble clustering technique for fault prediction using software metrics. In: Proc. of the Machine Learning and Applications. Detroit, 2014. 356–361.

[103] Park M, Hong E. Software fault prediction model using clustering algorithms determining the number of clusters automatically. Int'l Journal of Software Engineering and its Applications, 2014,8(7):199–204.

[104] Muhammed M, Unal C, Ahmet Z. A novel defect prediction method for Web pages using *K*-means++. Expert Systems with Applications, 2015,(42):6496–6506.

[105] Bishnu PS, Bhattacherjee V. Software fault prediction using quad tree-based *K*-means clustering algorithm. IEEE Trans. on Knowledge and Data Engineering, 2012,24(6):1146–1151.

[106] You G, Wang F, Ma YT. An empirical study of ranking-oriented cross-project software defect prediction. Int'l Journal of Software Engineering and Knowledge Engineering, 2017,26(9-10):1511–1538.

[107] Nguyen TT, An TQ, Hai VT, Phuong TM. Similarity-based and rank-based defect prediction. In: Proc. of the Advanced Technologies for Communications (ATC 2014). Hanoi, 2014. 321–325.

[108] Yang X, Tang K, Yao X. A learning-to-rank approach to software defect prediction. IEEE Trans. on Reliability, 2015,64(1): 234–246.

[109] Yadav HB, Yadav DK. A fuzzy logic based approach for phase-wise software defects prediction using software metrics. Information and Software Technology, 2015,63(7):44–57.

[110] You G, Ma Y. A ranking-oriented approach to cross-project software defect prediction: An empirical study. In: Proc. of the Software Engineering & Knowledge Engineering. San Francisco, 2015. 278–287.

[111] Verma D, Kumar S. Prediction of defect density for open source software using repository metrics. Journal of Web Engineering, 2017,61(4):293.

[112] Rathore SS, Kumar S. An empirical study of some software fault prediction techniques for the number of faults prediction. Soft Computing, 2016,7:1–18.

[113] Rathore SS, Kumar S. Predicting number of faults in software system using genetic programming. In: Proc. of the Soft Computing and Software Engineering. Berkeley, 2015. 303–311.

[114] Gagandeep TS. Improved approach for software defect prediction using artificial neural networks. In: Proc. of the Reliability, Infocom Technologies and Optimization. Noida, 2016. 7–9.

[115] Osman H, Ghafari M, Nierstrasz O. Automatic feature selection by regularization to improve bug prediction accuracy. In: Proc. of the Machine Learning Techniques for Software Quality Evaluation. Klagenfurt, 2017. 2–32.

[116] Chen X, Zhao Y, Wang Q, Yuan Z. MULTI: Multi-objective effort-aware just-in-time software defect prediction. Information and Software Technology, 2018,1–13.

[117] Tunnell J, Anvik J. Using time series models for defect prediction in software release planning. In: Proc. of the Software Engineering and Knowledge Engineering. 2015. 451–454.

[118] Bell RM, Ostrand TJ, Weyuker EJ. The limited impact of individual developer data on software defect prediction. Empirical Software Engineering, 2013,18(3):478–505.

[119] Andreou AS, Chatzis SP. Software defect prediction using doubly stochastic poisson processes driven by stochastic belief networks. Journal of Systems and Software, 2016,122:72–82.

[120] Singh P, Pal N, Verma S, Vyas OP. Fuzzy rule-based approach for software fault prediction. IEEE Trans. on Systems Man and Cybernetics Systems, 2017,47(5):8826–837.

[121] Chang CP. Integrating action-based defect prediction to provide recommendations for defect action correction. Int'l Journal of Software Engineering and Knowledge Engineering, 2013,23(2):147–172.

[122] Zhang H, Cheung SC. A cost-effectiveness criterion for applying software defect prediction models. In: Proc. of the Software Engineering. Saint Petersburg, 2013. 643–646.

[123] Lu H, Cukic B, Culp M. A semi-supervised approach to software defect prediction. In: Proc. of the Annual Computer Software and Applications. Vasteras, 2014. 416–425.

[124] Jing XY, Ying S, Zhang ZW, Wu SS, Liu J. Dictionary learning based software defect prediction. In: Proc. of the Software Engineering. Hyderabad, 2014. 414–423.

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

[126] Peters F, Menzies T, Liang G, Hongyu Z. Balancing privacy and utility in cross-company defect prediction. IEEE Trans. on Software Engineering, 2013,39(8):1054–1068.

[127] Panichella A, Oliveto R, Lucia AD. Cross-project defect prediction models: L'union fait la force. In: Proc. of the Software Maintenance, Reengineering and Reverse Engineering. Antwerp, 2014. 164–173.

[128] Fukushima T, Kamei Y, Mcintosh S, Yamashita K, Ubayashi N. An empirical study of just-in-time defect prediction using cross-project models. In: Proc. of the Mining Software Repositories. Hyderabad, 2014. 172–181.

[129] Borg M, Pfahl D. Using text clustering to predict defect resolution time: A conceptual replication and an evaluation of prediction accuracy. Empirical Software Engineering, 2016,21(4):1437–1475.

[130] Parizy M, Takayama K, Kanazawa Y. Software defect prediction for LSI designs. In: Proc. of the Software Maintenance and Evolution. Victoria, 2014. 565–568.

[131] Gupta S, Gupta A. A set of measures designed to identify overlapped instances in software defect prediction. Computing, 2017,99(9):1–26.

[132] Rathore SS, Kumar S. A decision tree logic based recommendation system to select software fault prediction techniques. Computing, 2017,99(3):255–285.

[133] Jing X, Wu F, Dong X, *et al*. Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning. In: Proc. of the Software Engineering. Bergamo, 2015. 496–507.

[134] Yang Y, Zhou Y, Lu H, *et al*. Are slice-based cohesion metrics actually useful in effort-aware post-release fault-proneness prediction: An empirical study. IEEE Trans. on Software Engineering, 2015,41(4):331–357.

[135] Eyolfson J, Tan L, Lam P. Do time of day and developer experience affect commit bugginess. In: Proc. of the Mining Software Repositories, MSR. DBLP, 2011. 153–162.

[136] Rahman F. Ownership, experience and defects: A fine-grained study of authorship. In: Proc. of the Software Engineering. IEEE, 2011. 491–500.

附中文参考文献:

[1] 王青,伍书剑,李明树.软件缺陷预测技术.软件学报,2008,19(7):1565–1580. http://www.jos.org.cn/1000-9825/19/1565.htm [doi: 10.3724/SP.J.1001.2008.01565]

[2] 李勇,黄志球,王勇,房丙午.数据驱动的软件缺陷预测研究综述.电子学报,2017,45(4):982–988.

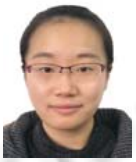
[3] 陈翔,王莉萍,顾庆,王赞,倪超,刘望舒,王秋萍.跨项目软件缺陷预测方法研究综述.计算机学报,2018,(1):254–274.

[5] 陈翔,顾庆,刘望舒,刘树龙,倪超.静态软件缺陷预测方法研究.软件学报,2016,27(1):1–25. http://www.jos.org.cn/1000-9825/4923.htm [doi: 10.13328/j.cnki.jos.004923]

[82] 于巧.基于机器学习的软件缺陷预测方法研究[博士学位论文].徐州:中国矿业大学,2017.1–142.

[83] 庄福振,罗平,何清,史忠植.迁移学习研究进展.软件学报,2015,26(1):26–39. http://www.jos.org.cn/1000-9825/4631.htm [doi: 10.13328/j.cnki.jos.004631]

宫丽娜(1985－),女,山东东平人,硕士,CCF 学生会员,主要研究领域为软件分析与测试,机器学习.



姜丽(1994－),女,学士,主要研究领域为软件分析与测试,机器学习.

姜淑娟(1966－),女,博士,教授,博士生导师,CCF 专业会员,主要研究领域为软件分析与测试,编译技术.