软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2017,28(6):1455−1473 [doi: 10.13328/j.cnki.jos.005228]

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

# 一种半监督集成跨项目软件缺陷预测方法*

何吉元[1], 孟昭鹏[1], 陈 翔[2], 王 赞[1], 樊向宇[1]

[1](天津大学 软件学院 软件工程系,天津 300072)

[2](南通大学 计算机科学与技术学院,江苏 南通 226019)

通讯作者: 王赞, E-mail: wangzan@tju.edu.cn

**摘 要**: 软件缺陷预测方法可以在项目的开发初期,通过预先识别出所有可能含有缺陷的软件模块来优化测试资源的分配.早期的缺陷预测研究大多集中于同项目缺陷预测,但同项目缺陷预测需要充足的历史数据,而在实际应用中,可能需要预测项目的历史数据较为稀缺,或这个项目是一个全新项目.因此,跨项目缺陷预测问题成为当前软件缺陷预测领域内的一个研究热点,其研究挑战在于源项目与目标项目数据集间存在的分布差异性以及数据集内存在的类不平衡问题.受到基于搜索的软件工程思想的启发,提出了一种基于搜索的半监督集成跨项目软件缺陷预测方法 $S^3$EL.该方法首先通过调整训练集中各类数据的分布比例,构建出多个朴素贝叶斯基分类器;随后,利用具有全局搜索能力的遗传算法,基于少量已标记目标实例对上述基分类器进行集成,并构建出最终的缺陷预测模型.在 Promise 数据集及 AEEEM 数据集上与多个经典的跨项目缺陷预测方法(Burak 过滤法、Peters 过滤法、TCA+、CODEP 及 HYDRA)进行了对比.以 *F*1 值作为评测指标,结果表明:在大部分情况下,$S^3$EL 方法可以取得最好的预测性能.

**关键词**: 跨项目软件缺陷预测;半监督学习;集成学习;遗传算法;朴素贝叶斯

**中图法分类号**: TP311

中文引用格式: 何吉元,孟昭鹏,陈翔,王赞,樊向宇.一种半监督集成跨项目软件缺陷预测方法.软件学报,2017,28(6):1455−1473. http://www.jos.org.cn/1000-9825/5228.htm

英文引用格式: He JY, Meng ZP, Chen X, Wang Z, Fan XY. Semi-Supervised ensemble learning approach for cross-project defect prediction. Ruan Jian Xue Bao/Journal of Software, 2017,28(6):1455−1473 (in Chinese). http://www.jos.org.cn/1000-9825/5228.htm

## Semi-Supervised Ensemble Learning Approach for Cross-Project Defect Prediction

HE Ji-Yuan[1], MENG Zhao-Peng[1], CHEN Xiang[2], WANG Zan[1], FAN Xiang-Yu[1]

[1](Department of Software Engineering, School of Computer Software, Tianjin University, Tianjin 300072, China)

[2](School of Computer Science and Technology, Nantong University, Nantong 226019, China)

**Abstract**: Software defect prediction can help developers to optimize the distribution of test resources by predicting whether or not a software module is defect-prone. Most defect prediction researches focus on within-project defect prediction which needs sufficient training data from the same project. However, in real software development, a project which needs defect prediction is always new or without any historical data. Therefore cross-project defect prediction becomes a hot topic which uses training data from several projects and performs prediction on another one. The main research challenges in cross-project defect prediction are the variety of distribution from source project to target project and class imbalance problem among datasets. Inspired by search based software engineering, this paper proposes a search based semi-supervised ensemble learning approach $S^3$EL. By adjusting the ratio of distribution in training dataset,

* 基金项目: 国家自然科学基金(61202030, 61373012, 61202006, 71502125)
Foundation item: National Natural Science Foundation of China (61202030, 61373012, 61202006, 71502125)
收稿时间: 2016-07-28; 修改时间: 2016-10-11; 采用时间: 2016-12-22; jos 在线出版时间: 2017-02-20
CNKI 网络优先出版: 2017-02-20 15:37:43, http://www.cnki.net/kcms/detail/11.2560.TP.20170220.1537.035.html

several Naïve Bayes classifiers are built as the base learners, then a small amount of labeled target instances and genetic algorithm are used to combine these base classifiers as a final prediction model. S$^3$EL is compared with other up-to-date classical cross-project defect prediction approaches (such as Burak filter, Peters filter, TCA+, CODEP and HYDRA) on AEEEM and Promise dataset. Final results show that S$^3$EL has the best prediction performance in most cases under the $F1$ measure.

**Key words**: cross-project defect prediction; semi-supervised learning, ensemble learning; genetic algorithm; Naïve Bayes

软件内部隐藏的缺陷可能导致其在实际运行时产生不可预料的后果,严重影响软件质量甚至有时会危及到人们的生命安全.软件缺陷预测(software defect prediction)方法通过挖掘软件历史仓库(software historical repositories)、构建缺陷预测模型,从而对新的程序模块进行缺陷预测.该方法可以为软件开发过程提供决策支持,即:在项目的开发初期,通过预先识别出所有可能含有缺陷的软件模块,可以针对性地对这些程序模块设计测试用例,以确保充分测试,从而提高软件质量.程序模块根据实际测试需求可设置为包、文件、类或函数等.目前,缺陷预测模型在构建时常采用支持向量机[1]、Logistic 回归[2]、朴素贝叶斯[3]、决策树[4]、集成学习[5]等常见的机器学习方法.

软件缺陷预测是当前大数据技术在软件工程领域的重点研究方向之一[6],早期大多数研究都集中于同项目软件缺陷预测(within-project defect prediction),即:使用同一项目中的部分已标记程序模块来构建预测模型,并对项目中的剩余模块进行缺陷预测.同项目缺陷预测需要充足的历史数据进行训练,但在实际应用中,通常缺少足够的历史数据针对同一项目来构建缺陷预测模型,或需要对一个没有训练数据的全新项目进行缺陷预测.因此,研究者们开始关注跨项目软件缺陷预测(cross-project defect prediction)问题.跨项目软件缺陷预测使用其他项目(即源项目)的训练数据来构建预测模型,并对一个全新的项目(即目标项目)进行缺陷预测[7].跨项目缺陷预测的研究重点在于如何利用已有源项目的大量标记数据解决具有不同分布甚至不同度量特征的目标项目的缺陷预测问题.但已有的跨项目缺陷预测方法的性能通常要弱于同项目缺陷预测的性能,主要的原因是:(1) 大部分情况下,不同项目的度量元取值分布具有显著差异性;(2) 缺陷预测数据集的内部存在类不平衡问题.

由于不同项目的开发人员、应用领域、编程语言等并不相同,源项目与目标项目的数据集间具有较大的分布差异[8],不能满足独立同分布假设,导致根据源项目训练出的预测模型在目标项目上不能取得较好的预测性能[9,10].针对这一问题,研究人员一般借助迁移学习(transfer learning)来缓解数据取值分布的差异性[11,12].迁移学习是对源项目数据集进行转换、学习并获取与目标项目最为相关的知识来用于模型构建的方法.

类不平衡问题是缺陷预测数据集中普遍存在的问题,原因在于缺陷在软件模块中的分布大致符合帕累托原则[13],即,大部分(约 80%)的缺陷集中于很少(约 20%)的程序模块内[14].因此在搜集的数据集中,非缺陷模块(多数类)的数量要远超过缺陷模块(少数类)的数量,导致模型在预测时偏向多数类,而对少数类的预测精度较低.但在实际应用中,对缺陷模块的错误预测代价远高于对非缺陷模块的错误预测代价,因此,如何缓解源项目数据集中的类不平衡问题也成为跨项目缺陷预测研究中的一个重要问题.

基于搜索的软件工程(search based software engineering)是由 Harman 教授提出的概念[15],其将传统的软件工程问题转化为基于智能搜索的优化问题,并借助遗传算法等元启发式搜索算法(meta-heuristic search algorithms)在问题的所有可行解空间中搜索最优解或近似最优解的研究方法.受到该领域思想的启发,本文提出一种基于搜索的半监督集成跨项目缺陷预测方法 S$^3$EL(search based semi-supervised ensemble learning).该方法综合考虑了数据分布不一致问题及类不平衡问题对跨项目软件缺陷预测性能的影响,采用朴素贝叶斯分类器构建基分类器,并借助遗传算法的全局搜索能力进行集成学习,构建最终的预测模型.实验结果表明,该方法在多个公开数据集上均能取得较好的预测性能.本文的主要贡献总结如下.

(1) S$^3$EL 提出一种基分类器构建方法,即:根据各维特征均值对源项目数据进行划分,分别训练两个子分类器作为一个基分类器.借助该步骤,可以有效缓解类不平衡问题,并从训练数据角度提高基分类器的差异性.随后,借助具有全局搜索能力的遗传算法,对基分类器进行集成,即,基于目标项目中的少数已标记实例来搜索出基分类器的最优组合权重;

(2) 基于 AEEEM 与 Promise 数据集,将 S$^3$EL 方法与 Burak 过滤法[16]、Peters 过滤法[17]、TCA+方

法[10]、CODEP 方法[18]和 HYDRA 方法[2]进行了比较,并验证了论文所提方法的有效性.

本文第 1 节介绍跨项目缺陷预测的研究背景及相关方法.第 2 节描述 S$^3$EL 方法及其实现细节.第 3 节介绍本文的实证研究,包括实验设计、采用的评测数据集及评价指标以及实验结果分析等.第 4 节总结全文并对未来研究方向进行展望.

## 1 研究背景和相关工作

已有的跨项目软件缺陷预测方法可简单分为 3 类:(1) 有监督缺陷预测方法,即,存在源项目数据集及其类别标注信息,根据源项目构建缺陷预测模型,并对目标项目进行缺陷预测;(2) 无监督缺陷预测方法,直接根据目标项目中的未标记数据,借助聚类分析等方法进行缺陷预测;(3) 半监督缺陷预测方法,根据源项目及目标项目中的少数有类别标记的模块共同构建缺陷预测模型,并对目标项目中剩余未标注模块进行预测.本文提出的 S$^3$EL 方法属于半监督缺陷预测方法.

### 1.1 有监督缺陷预测方法

在有监督缺陷预测方法中,为缓解不同项目间数据的分布差异问题,Turhan 等人[16]提出了 Burak 过滤法,其针对每个待预测的目标实例,计算其与源数据实例的欧式距离,从中选取距离最近的 $k$ 个实例作为训练集进行模型构建,被重复选择的训练实例只保留一个.Burak 过滤法从目标项目出发,选取与其最为相似的源项目实例作为训练集,从而提升了跨项目缺陷预测的性能.结果表明:通过实例选择,能够显著提升跨项目缺陷预测的性能,但与同项目缺陷预测相比仍有一定差距.Peters 等人[17]从源数据集出发进行实例选择,提出了 Peters 过滤法.具体的做法是:针对源数据集中的每个实例,从目标项目中选取与之最为接近的实例进行标记,随后对每个有标记的目标实例,从源数据集中选取与之距离最近的实例构成训练集来构建缺陷预测模型.He 等人[19]从项目选择角度出发,首先计算出目标项目与源项目间的相似度,随后选出 $k$ 个最为相似的源项目作为候选训练集,接着,针对每个候选训练集中的源项目,移除掉部分与目标项目具有较大分布差异的特征,再借助 bagging 方法进行集成学习来得到最终的预测结果.实验发现,该方法在预测性能和计算开销上要优于 Burak 过滤法.

Nam 等人[10]从特征映射角度出发提出了 TCA(transfer component analysis)方法,将源项目和目标项目的特征映射到使其最为接近的潜在空间上,从而缓解了不同项目间的数据分布差异,但数据归一化方法的选择对最终的预测性能有较大影响.随后,他们在 TCA 方法的基础上提出了 TCA+[10],该方法可以通过对源项目和目标项目的特征进行分析,自动选出最为合适的数据归一化方法.

Malhotra 等人[20]使用 Android 系统的 6 个发布版本作为数据集,尝试了贝叶斯网络、多层感知器、决策树、支持向量机等 18 种机器学习方法进行跨项目缺陷预测,基于 AUC(area under ROC curve)指标,并借助统计检验方法对这些机器学习方法的预测性能进行了比较.结果表明:朴素贝叶斯、Adaboost、RBF 神经网络、Bagging、ADTree、多层感知器、随机森林、LogitBoost 等 8 种机器学习方法在 AUC 值上表现出了稳定且良好的性能.其中,朴素贝叶斯在整体上具有最好的预测性能.Lessmann 等人[21]在 NASA 数据集上比较了 22 种不同的机器学习方法,结果发现:包括朴素贝叶斯在内的 17 种机器学习方法表现出较好的预测性能,但它们之间的性能差异并不显著.随后,Ghotra 等人[22]在同一数据集上得出了相同的结论,但在去除噪音后的 NASA 数据集及 Promise 数据集上发现,不同机器学习方法之间存在显著的性能差异.因此,研究者们开始尝试对各种机器学习模型进行集成.

Panichella 等人[18]认为:不同的模型在跨项目缺陷预测中具有不同的表现能力,将不同的模型进行组合预测具有一定的合理性.因此,他们基于集成学习的 stacking 方法提出了 CODEP(combined defect predictor).具体来说:在训练数据集上分别构建 6 个基分类器(Logistic 回归、贝叶斯网络、RBF 网络、ADTree、多层感知器、判定表),将基分类器的输出作为新的输入向量,借助分类方法(Logistic 回归或贝叶斯网络)构建出集成分类器.Zhang 等人[23]在 CODEP 方法的基础上尝试了更多的集成方法,将 6 种基分类器进行组合,包括最大投票、Boosting、随机森林、平均投票、bagging 等,并采用 $F1$ 和 NofB20 指标,在 Promise 数据集上对这 6 种组合方法做出评价.实验结果表明:这些集成方法能够有效提高基分类器的预测能力,其中,最大投票法表现出了最好

的预测性能.

针对跨项目缺陷预测的类不平衡问题,Ryu 等人提出了 VCB-SVM[24]及 HISNN 方法[25].其中,VCB-SVM 方法根据特征分布识别出训练集中的离群点,并根据目标数据计算训练数据的相似度权值用于模型训练.HISNN 同样从源数据中移除离群点,具体的做法是:首先,移除训练集中影响其整体分布的离群点;其次,移除训练集中影响目标数据整体分布的离群点;最后,根据每个目标实例用最近邻过滤法选取 $k$ 个最近似实例加入训练集.完成训练集构建后,对于每个待测实例,根据其 $k$ 个最近邻训练实例的类别进行缺陷预测.

### 1.2 半监督缺陷预测方法

Turhan 等人[26]考虑混合使用源项目数据和目标项目数据进行缺陷预测.具体来说,若目标项目的训练数据较为充足,则不需要源项目实例参与模型训练;若目标项目中仅存在少数有类别标记的实例,则综合使用源项目数据共同构建缺陷预测模型.夏鑫等人[2]提出一种使用少量已标记目标数据构建混合缺陷预测模型的方法 HYDRA,并在 Promise 数据集上采用 $F1$ 和 PofB20 指标与 TCA+、Peters 过滤法、GP、MO 等经典方法做了对比实验.HYDRA 包含遗传算法及集成学习两个阶段.

- 在第 1 阶段中,基于每个候选源项目及少量已标记目标数据作为训练集构建多个基分类器;随后,借助遗传算法对分类器的最优组合权重进行全局搜索,得到一个 GA 模型;
- 集成学习阶段借助 Adaboost 的思想,每轮通过对实例的权重调整生成一个 GA 模型,将多轮迭代输出的 GA 模型做线性组合获得终分类器进行缺陷预测.

实证研究表明:HYDRA 与其他方法相比,在 Promise 数据集上有最好的预测性能.HYDRA 针对每个源项目构建基分类器的方法可以认为是某种程度的源项目选择,GA 过程搜索的组合阈值代表了某一源项目构建出的基分类器对目标实例的预测贡献度.当源项目数据集的项目个数过少(如 1 个)时,GA 阶段的结果会偏向仅使用少数已知类别的目标实例构建的基分类器,使得最终预测结果与不使用源项目进行训练时较为接近,不能达到充分利用源项目信息的目的.HYDRA 方法限定了跨项目缺陷预测过程必须有充足的源项目作为训练集,而在实际应用中,这样的训练集的选取和构建较为困难.

### 1.3 无监督缺陷预测方法

由于源项目与目标项目间具有较大的分布差异,有研究人员尝试仅借助目标数据本身的特征来构建预测模型.Zhong 等人[27]借助 $k$-means 及 neural-gas 方法对目标实例进行聚类并提取出每个簇的聚类中心,由专家对聚类中心进行类别标记,实验结果发现,neural-gas 的预测性能要优于 $k$-means,但该方法需要根据经验确定聚类个数并由专家参与最后的决策.Zhang 等人[28]提出一种基于谱聚类的无监督预测方法.谱聚类根据实体间的连通性进行数据集划分,实体间的连通性由特征值的相似度决定.将数据集划分为两类后,根据每一类的特征均值决定其是否可能包含缺陷,即,具有较高均值的聚类被标记为缺陷类.Nam 等人[29]提出了 CLA 和 CLAMI 方法.CLA 的具体做法是:计算各维度特征均值,对每个待预测实例,计算其高于均值的特征值个数标记为 $k$,并根据实例的 $k$ 值将全部数据集划分为 $n$ 类.最后,将前 $n/2$ 个聚类中的实例标记为缺陷类,其余则标记为非缺陷类.在 CLA 的基础上,CLAMI 增加了特征选择和实例选择阶段,该阶段将 CLA 标记的类别进行反向推导,移除违背 CLA 原则的特征和已标记实例,最后,在保留的特征和标记实例上,借助机器学习方法构建模型,并对未标记的实例进行缺陷预测.但已有实证研究表明,无监督缺陷预测方法通常在预测性能上弱于有监督及半监督缺陷预测方法[28].

总结相关研究工作,目前,大部分跨项目缺陷预测的方法都是基于有监督学习.然而,由于跨项目缺陷预测中,源项目数据与目标项目数据间的特征分布差异明显,基于源项目训练出来的预测模型具有较差的泛化能力.本文所提出的 $S^3$EL 方法基于半监督学习,借助少量已标注的目标项目实例构建模型训练的验证集,用于遗传算法阶段的搜索过程,能够有效提高预测模型的泛化能力.与从项目选择角度出发的 HYDRA 方法不同,$S^3$EL 不受参与模型训练的源项目个数的限制,从特征角度出发进行模型构建.本文的实证研究部分论证了 $S^3$EL 方法的有效性.

## 2 S³EL 方法

本文提出了一种基于搜索的半监督集成跨项目缺陷预测方法,该方法借助源项目和少量已标记的目标项目的实例进行模型构建,用于预测剩余未标记的目标项目实例.图 1 给出了方法的整体流程图,该方法主要包括如下 4 个阶段:(1) 度量元取值预处理;(2) 基于朴素贝叶斯的基分类器构造;(3) 基于遗传算法进行基分类器的集成;(4) 根据集成分类器对目标数据进行缺陷预测.

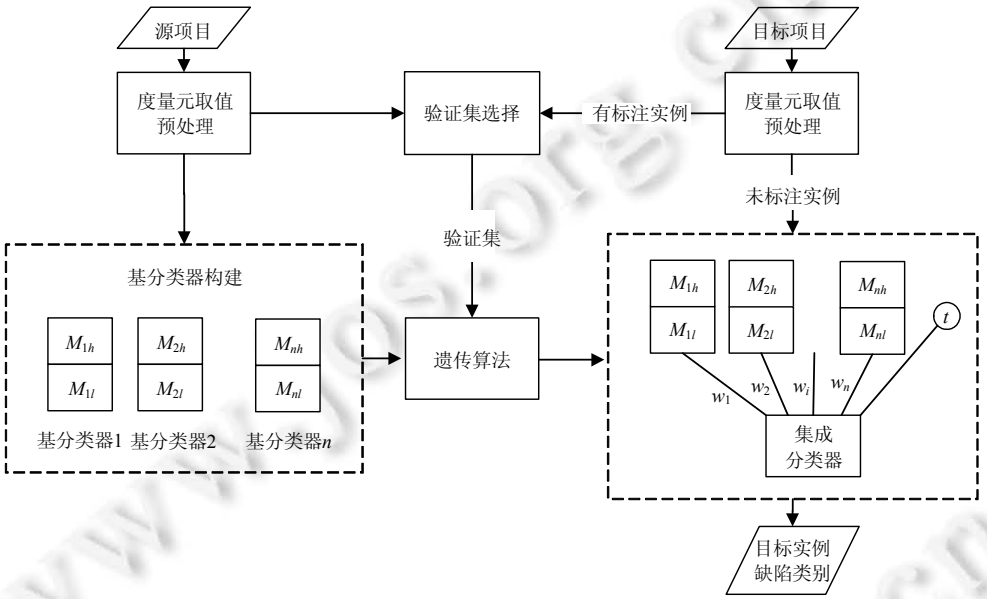


Fig.1 Flow chart of S³EL method

图 1 S³EL 方法的流程图

### 2.1 特征取值预处理

跨项目缺陷预测通常不能达到同项目缺陷预测效果的一个可能原因是:跨项目缺陷预测中,源项目与目标项目度量元的取值分布具有显著性差异[8−10].这种差异来自于不同软件所处的应用领域、采用的编码语言、项目开发流程及开发人员的技术水平等并不相同,源项目与目标项目间不能满足独立同分布假设,导致根据源项目训练出的预测模型在目标项目上并不十分适用.现有的跨项目缺陷预测方法主要从以下两方面对数据进行预处理:(1) 选择与目标项目具有相似分布的源项目或实例集合作为训练集来构建预测模型,代表方法有 Burak 过滤法[16]、Peters 过滤法[17]等;(2) 对源数据及目标数据度量元取值进行转换,使其具有更为接近的取值分布.研究发现,大多数度量元取值分布满足幂律分布而非正态分布[30],这会对缺陷预测模型的性能产生一定的影响.因此,研究人员多采用对数变换或 rank 变换对源数据及目标数据进行处理,以提高度量元取值的正态分布性[9,31−34].对数变换是一种数学算子,将原始特征值替换为其对数运算的结果.rank 变换将原始特征值替换为其对应的 rank 值.

S³EL 方法采用自然对数变换与 z-score 相结合的变换方法,对源项目数据及目标项目数据的各个度量元取值分别进行归一化处理.根据自然对数的定义,对数变换只能对大于 0 的数值进行转换,为处理特征值等于 0 的情况,通常给 $x$ 加上一个常数 1 再进行对数操作.因此,本文采用以下公式进行对数变换:

$$\log(x)=\ln(x+1) \tag{1}$$

其中,$x$ 为实例的任一度量元值.

研究发现:z-score 作为数据归一化方法,可以显著提高模型的预测能力[10].设原始数据是具有 $m$ 个实例、$n$ 个特征的矩阵 $D_{m\times n}$,$D_{ij}$ 表示第 $i$ 个实例的第 $j$ 个特征值,$v_j=[D_{1j},D_{2j},\ldots,D_{mj}]$表示第 $j$ 个特征对应的向量,$\bar{v}_j$ 表示 $v_j$

的均值,$s_j$ 表示 $v_j$ 的标准差,则对 $D_{m\times n}$ 中的每个元素取值可进行如下转化:

$$\hat{D}_{ij}=\frac{D_{ij}-\overline{v}_j}{s_j} \tag{2}$$

### 2.2 基于朴素贝叶斯的基分类器构造

影响跨项目缺陷预测性能的另一个主要因素是,缺陷预测数据集中普遍存在的类不平衡问题.类不平衡问题导致模型在训练时偏向训练数据中的多数类,而对少数类的预测精度较低.而在实际应用中,缺陷类(少数类)预测错误的代价远高于非缺陷类(多数类)的错误代价.针对这一问题,常见的解决方法主要分为以下两种:(1) 从调整算法角度出发的代价敏感学习方法,该方法在模型构建中,通过设置不同的代价函数,使模型更偏向预测错误代价更高的类数据;(2) 从样本角度出发的重采样方法,代表方法有过采样、欠采样和移动阈值等.过采样和欠采样分别通过增加少数类的样本数量和减少多数类的样本数量来改变数据间的类别分布,移动阈值则通过将决策平面向少数类方向移动,以减小类不平衡问题对决策产生的影响.本文的方法通过调整训练集中非缺陷类及缺陷类的实例数据比例,使训练得到的分类器对于某类数据具有一定的倾向性,以缓解类不平衡问题.基于开发经验,程序模块的内部结构越复杂,那么其含有缺陷的可能性越高[32,35−37].因此,缺陷实例通常比非缺陷实例具有更高的度量元取值[32,35,38−40].基于以上经验,我们提出了根据特征均值划分数据集的基分类器构建方法.具体的做法是:

(1) 设 $F=\{F_1,F_2,\ldots,F_n\}$ 代表数据集的 $n$ 维特征;

(2) 对每一维特征 $F_i$ 计算出其特征均值 $avg\_value[i]$;

(3) 对每一维特征 $F_i$:

a) 根据第 $i$ 维特征值的均值 $avg\_value[i]$,将训练集划分为大于等于均值和小于均值的两部分数据,分别记作 $T_{ih}$,$T_{il}$;

b) 分别在 $T_{ih}$ 和 $T_{il}$ 数据集上使用朴素贝叶斯分类器进行训练,构建出两个模型,记作 $M_{ih}$,$M_{il}$.将这两个子模型共同构建成一个基分类器,并记作($M_{ih}$,$M_{il}$).

(4) 最后,我们得到基分类器的集合$\{(M_{1h},M_{1l}),(M_{2h},M_{2l}),\ldots,(M_{nh},M_{nl})\}$.

基分类器构建流程如图 2 所示.

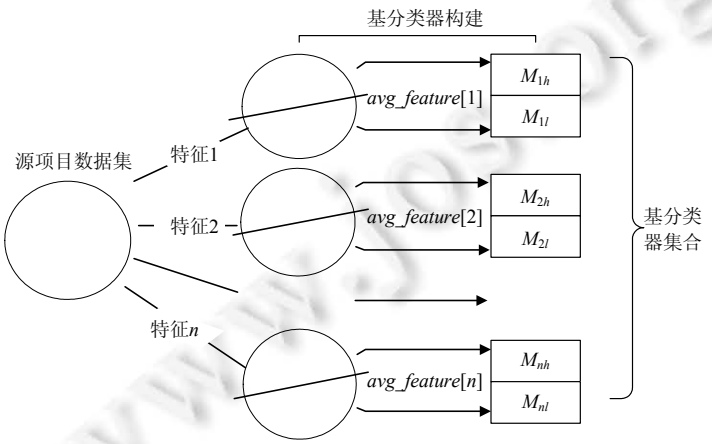


Fig.2 Flow chart of base learner building

图 2 基分类器构建流程

经过上述步骤,共构建出 $n$ 个基分类器.对一个待预测的实例 $ins$,使用第 $i$ 个基分类器进行预测时,根据待预测实例第 $i$ 维特征值所处的位置进行选择:若实例特征值大于等于该维度特征均值,选择 $M_{ih}$ 进行预测;否则,选择 $M_{il}$ 进行预测.

基分类器在构建时,采用朴素贝叶斯分类方法.朴素贝叶斯是一种基于贝叶斯定理的统计分类学习方法.设

输入空间 $S$ 为 $n$ 维向量的集合,$c_1,c_2,\ldots,c_j$ 为类标记集合,朴素贝叶斯法分类时,对给定的输入 $x$,根据全概率公式和条件概率的定义计算后验概率分布 $P(Y=c_k|X=x)$,并将后验概率最大的类作为 $x$ 的最终预测类型,其计算公式如下所示:

$$P(Y=c_k \mid X=x)=\frac{P(X=x \mid Y=c_k)P(Y=c_k)}{\sum_{k=1}^{j}P(X=x \mid Y=c_k)P(Y=c_k)} \tag{3}$$

朴素贝叶斯算法简单,具有坚实的数学基础及稳定的分类效率,已被证明在缺陷预测上能够取得较好的性能[3].但朴素贝叶斯模型是基于特征条件独立假设的,而实际应用中,该假设很难成立,尤其特征个数较多或特征间有较大冗余时,将造成朴素贝叶斯的分类效果较差.为了提高朴素贝叶斯的预测性能,我们以朴素贝叶斯分类器作为基分类器进行集成学习.集成学习通过某种方式组合多个性能较弱的基分类器,能够达到提高泛化能力及整体模型表现能力的目的.然而朴素贝叶斯模型本身非常稳定,在一些集成方法上,作为基分类器进行组合时并不能达到很好的效果(如 bagging).本文所采用的通过各维特征均值划分数据集进行模型训练的方法,能够一定程度上破坏朴素贝叶斯方法的稳定性,并提高其作为基分类器的差异性.

### 2.3 基于遗传算法的集成学习方法

集成学习(ensemble learning)是构建多个基分类器并通过一定方式将其组合成一个集成分类器进行分类的学习方法,包括以下两个步骤:(1) 用训练数据集训练若干个基分类器;(2) 融合不同基分类器的分类结果.基分类器可以是相同类型的,也可以是不同类型的.集成学习能够得到比基分类器更好的分类效果及泛化能力,不容易出现过拟合情况,因此适用于跨项目缺陷预测问题.但保证集成学习有效的前提是:(1) 每个基分类器有较好的性能;(2) 不同基分类器之间要具有一定的差异性.

单个基分类器具有较好的预测能力,通常要求其分类准确率不低于 0.5.基分类器之间具有一定差异性,表示不同分类器对同一实例的预测具有不同的表现能力.常见的集成学习方法包括 voting,boosting,bagging,stacking 等.S$^3$EL 方法正是基于遗传算法对上一节中构建的朴素贝叶斯基分类器进行组合,采用加权多数投票(weighted majority voting)规则进行集成学习.

#### 2.3.1 问题定义

基于遗传算法进行基分类器的组合过程,需要对各基分类器的权值及整体阈值进行搜索,根据评价指标选出最优种群个体.遗传算法是一种遵循生物进化思想,将求解问题的搜索空间映射到遗传空间的群体性操作过程.操作的对象为个体(或称为染色体),当前环境下的所有个体共同构成一个种群.根据个体相对于当前环境的适应能力对种群进行选择,保留适应力强的个体,淘汰适应力弱的个体.生存下来的个体之间通过交叉、变异等算子,产生适应力更强的全新个体,并构成新的种群.不断迭代这一过程,直至达到终止条件.遗传算法是一种全局优化的搜索算法,而其他的启发式搜索算法,如爬山法等,则容易出现局部收敛问题,从而无法找到全局最优解.因本问题中基分类器可能存在多种最优组合方式,我们选择遗传算法对集成学习阶段的各基分类器的权值及整体阈值进行搜索.遗传算法阶段的主要研究内容包括染色体的编码方式、适应度函数的定义、遗传算子的设定及迭代终止条件的定义等.

在本文的方法中,需要进行全局搜索的对象是每个基分类器的权重 $\omega_i$ 及集成分类器的阈值 $t$,即,向量 $[\omega_1,\omega_2,\ldots,\omega_n]$.基分类器的输出结果是各个实例被预测为缺陷类的概率 $p$,在朴素贝叶斯的判别过程中,若 $p\geqslant 0.5$,则预测结果为缺陷类;否则为非缺陷类.根据加权多数投票规则进行组合时各个基分类器的输出结果,经过加权组合后得到的结果若大于等于阈值 $t$,则最终判定结果为缺陷类(即取值为 1);否则为非缺陷类(即取值为 0).具体公式如下所示:

$$Label(i)=\begin{cases}1, & \text{if } C(i)\geqslant t\\ 0, & \text{if } C(i)<t\end{cases} \tag{4}$$

其中,$C(i)$对每个基分类器的输出结果(即 $f(M_{jh},M_{jl})$)进行加权求和,其计算公式为

$$C(i)=\sum_{j=1}^{N}\omega_j\times(M_{jh},M_{jl}) \tag{5}$$

2.3.2 染色体编码方式

根据问题定义,需要进行搜索的对象是 $n$(特征维数)个基分类的权重和集成分类器的阈值$[\omega_1,\omega_2,\ldots,\omega_n,t]$.权重代表按照某一维度特征均值对训练集进行划分后构建的基分类器在最终预测模型中的贡献度,阈值代表实例被判定为缺陷类的可能性,因此,各基分类器的权重之和应为 1(即 $\sum_{i=1}^{n}\omega_i=1$),而 $t$ 应为 0~1 之间的实数.在实际遗传算法的实现过程中,为了同时对 $\sum_{i=1}^{n}\omega_i=1$ 及 $t\in[0,1]$条件进行限制,计算适应度函数时,需对 $\omega_i$ 进行转换.

设源项目数据和目标项目数据的特征维数为 $n$,权重编码的取值空间为[0,1]之间的实数,令 $\omega=[\omega_1,\omega_2,\ldots,\omega_n]$ 为遗传算法的实际搜索结果,而在计算种群适应度时,采用以下方式对 $\omega_i$ 进行转换:

$$\hat{\omega}_i=\frac{\omega_i}{\sum_{i=1}^{n}\omega_i} \tag{6}$$

转化后的 $\hat{\omega}=[\hat{\omega}_1,\hat{\omega}_2,\ldots,\hat{\omega}_n]$ 是预测模型中各基分类器的实际组合权重,公式(6)保证了 $\sum_{i=1}^{n}\omega_i=1$ 的限制条件.由于概率 $p$ 的取值范围为 0~1,基分类器输出结果经过加权求和后得到一个[0,1]的实数,因此,$t\in[0,1]$的条件也得到了满足.将权值阈值表示为一个 $n$+1 维实数向量,即,染色体的定义为 $Chromosome=\{\omega_1,\omega_2,\ldots,\omega_n,t\}$,其中,每个参数的搜索空间均为[0,1],采用实数进行编码.

2.3.3 适应度函数的设定

$S^3EL$ 是一种半监督的集成学习方法,我们假设目标项目中有且仅有少数实例有类别标注,这些信息可以被学习并应用于剩余的多数未标注目标实例的分类上.在实际应用中,少数目标项目实例(如 5%)存在历史标注信息,或借助开发人员对某些实例进行人工标注,用于缺陷预测模型构建的方式是可行的.我们将目标项目中这些有标注的实例定义为验证集,验证集将用于遗传算法阶段以选择使目标数据预测能力最优的种群个体.遗传算法通过计算染色体在验证集上的适应度进行优胜劣汰,最终得到在验证集上具有最优适应度的个体,即为基分类器的最优组合权重及模型整体阈值.

$F1$ 指标是查准率(precision)和查全率(recall)的调和平均数,能够有效地平衡查准率和查全率,被广泛用于评价缺陷预测的性能,$F1$ 值越高,表示模型对缺陷类的预测能力越好.因此,方法使用模型在验证集上预测的 $F1$ 值作为染色体的适应度函数,其计算公式见第 3.3 节.

以 $F1$ 值作为适应度函数进行搜索时,若验证集中仅存在一类实例(即全为缺陷类或全为非缺陷类)时,会出现遗传算法的搜索结果使模型预测偏向验证集中唯一类别的情况,难以解决类不平衡问题.因此,在方法的实现过程中,我们通过在验证集中加入来自源项目的部分实例进行调和,防止上述不合理情况的发生,从而提高模型的泛化能力.具体的做法是:首先,按某一既定比例(例如 5%)对目标数据集进行抽样,设抽取的样本记为 $V_t$,样本大小为 $v$;随后,在源项目数据集中分别抽取 $v/2$ 个缺陷类实例和 $v/2$ 个非缺陷类实例,共同构成集合 $V_s$,其大小同样为 $v$.目标实例中的有标记实例集合 $V_t$ 和源项目的抽样集合 $V_s$ 共同构成遗传算法阶段使用的验证集 $V$($V_s$ 中的实例将不再参与基分类器构建阶段的模型训练).而在计算适应度函数时,由于目标项目集中的有标记实例的重要性要大于源项目集中的实例重要性,因此我们给 $V_t$ 的预测结果 $F1$ 值赋予更高的权重(即:将基于 $V_s$ 上计算出的 $F1$ 值的权重设置为 0.5,而将基于 $V_t$ 上计算出的 $F1$ 值的权重设置为 1).因此,染色体的适应度函数的计算公式如下所示:

$$Fitness=0.5\times F1(V_s)+F1(V_t) \tag{7}$$

2.3.4 遗传算子的设定

遗传算法中的染色体由基因构成,每个基因对应于所求解的一个参数(如某个基分类器的权值 $\omega_i$).初始时,遗传算法随机生成一个候选解集作为初始种群,种群由基因编码的一定数目染色体组成.进化过程中,通过按照

优胜劣汰的自然法则逐代产生新的个体,并保留适度度更高的个体进入下一代,成为新的种群.重复操作,直至终止条件.

遗传算法的种群进化过程包括 3 个操作算子:(1) 选择算子,根据适应度值对染色体进行选择,判断其是否能够保留至下一代;(2) 交叉算子,按一定的概率随机选择某两个染色体中的基因位置进行交换,以产生新的个体;(3)变异算子,按一定的概率在染色体的随机位置上进行基因突变,产生新的基因型,以确保种群的多样性.

本文使用轮盘赌法作为选择算子.轮盘赌法中,适应度值越高的个体被保留至下一代的可能性越大,即:染色体被选中的几率和适应度值成正比,而不保证适应度值最高的个体一定能进入下一代.轮盘赌法的具体做法是:(1) 计算种群中全部个体的适应度值总和 $F=\sum_{i=1}^{Popsize}F_i$;(2) 计算种群中每个个体被保留至下一代的概率 $F_i/F$;(3) 全部概率总和为 1,每个个体的概率值为 0-1 上的一个区间;(4) 产生一个 0-1 之间的随机数,随机数位于某一区间内的次数,即为该区间对应染色体被选中的次数.

交叉算子包括单点交叉、多点交叉、洗牌交叉等.本文的方法中,使用单点交叉算子,在产生新个体的过程中,以某一给定概率交换随机两个染色体之间的部分基因,交叉点基因位置随机产生.在交叉运算过程中,交叉的内容是父代染色体已有的基因,经过交叉得到新的基因构成即为新个体.但如果在某个基因位置上父代的基因型均相同,则子代就不可能产生新的基因型.因此,算法引入变异算子以确保种群的多样性.

变异运算对个体的某个或某些位置上的基因值按某一较小概率进行改变,以产生新的个体.实数编码中,常用的变异算子有均匀变异、边界变异、非均匀变异和高斯变异算子.本文的方法采用高斯变异.具体的做法是:变异操作中,使用均值为 $m$、方差为 $s^2$ 的正态分布的一个随机数来替换原有基因值.根据正态分布的特征可知,高斯变异重点搜索父代个体附近的局域区域.

每轮迭代中,父代种群经过选择、交叉、变异算子后形成新的种群,每次记录最优个体,直至达到终止条件,输出最优个体,即为我们所求的$[\omega_1,\omega_2,\ldots,\omega_n,t]$.

遗传算法阶段的伪代码见表 1.

**Table 1** Pseudocode in the phase of genetic algorithm

**表 1** 遗传算法阶段的伪代码

输入:基分类器向量$\{(M_{1h},M_{1l}),(M_{2h},M_{2l}),\ldots,(M_{nh},M_{nl})\}$,实例个数为 $m$,特征维度为 $n$ 的验证集 $V$,种群大小 *PopSize*,最大迭代次数 Max*Gen*;

输出:最优的 $n$ 维权重及阈值$[\omega_1,\omega_2,\ldots,\omega_n,t]$.

```
Pop=initial population of PopSize Chromosomes
for all V_i∈{V_1,V_2,…,V_m} do
    j=0
    while j<n do
        if V_ij≥0
            P_ij=M_jh(V_i)
        else
            P_ij=M_jl(V_i)
        endif
        j=j+1
    end while
end for
Use P_m×n to Calculate fitness value (Eq.7) of Pop and record the best individual
curGen=0, Pop'=Pop
Repeat
    curGen=curGen+1
    Pop'=select(Pop')
    Pop'=crossover(Pop')
    Pop'=mutation(Pop')
    Use P_m×n to Calculate fitness value (Eq.7) of Pop' and update the best individual
    curGen=curGen+1
until cunGen==MaxGen
return best individual
```

### 2.4 根据集成分类器对目标数据进行缺陷预测

对目标项目中的剩余未标记实例进行缺陷预测时,首先以相同的方式对目标项目中的特征取值进行预处理,对每个未标记实例,根据其各维特征值的大小选取对应基分类器的子分类器,借助遗传算法输出的最优个体权重及阈值,将各基分类器的输出结果进行组合,根据公式(4)得到最终的缺陷预测类型.

## 3 实证研究

本节我们将在多个公开数据集上对 $S^3$EL 方法的预测性能进行验证.首先,我们描述了实验设计、对比方法及两个需要回答的研究问题;其次,介绍了实验中选择的数据集的统计信息及评价指标;最后,对实验结果进行了总结,并借助假设检验方法进一步验证了我们算法的有效性.

### 3.1 实验设计

在跨项目缺陷预测中,根据源项目数据集的使用方式可以分为以下两种应用场景:一对一模式和多对一模式.其中,一对一模式使用单个源项目作为训练集对目标项目进行预测;多对一模式使用多个源项目共同构建预测模型,针对目标项目进行预测.为了验证算法的通用性及有效性,我们将选取两组不同的数据集进行跨项目缺陷预测,并分别考虑了一对一模式和多对一模式.我们将 $S^3$EL 方法与多个典型的跨项目缺陷预测方法进行对比,包括基于实例选择的 Burak 过滤法[16]($k$=10)、Peters 过滤法[17]、基于集成学习的 CODEP[18]、HYDRA[2]方法、基于特征映射的 TCA+方法[10]及作为基准方法的朴素贝叶斯方法.基准方法采用和 $S^3$EL 中基分类器一致的朴素贝叶斯模型,基于全部源项目实例进行训练.HYDRA 及 TCA+使用了作者提供的源代码,CODEP,Peters 过滤法及 Burak 过滤法借助 Weka[41]工具包进行了实现.需要特别指出,HYDRA 方法不适用于基于一对一模式的跨项目缺陷预测,因此在一对一模式下不进行比较.$S^3$EL 方法基于 Python 的机器学习工具 sklearn(http://neuro.debian.net/pkgs/python-sklearn.html)进行了实现,其中,遗传算法阶段借助了遗传编程框架 pyevolve(http://pyevolve.sourceforge.net)进行实现.

实验中的参数设置见表 2,其中,Max*Gen* 表示遗传算法的最大迭代次数,*PopSize* 表示每一代种群内含有的个体数,$P_c$ 表示两个染色体上相同位置的基因进行交叉的概率,$P_m$ 表示个体中染色体变异的概率,$V$%表示验证实例个数占目标实例总数的比例.为公平地与其他算法进行对比,验证集在目标数据集和源数据集上的的抽样均采用随机抽样.

**Table 2** Parameters setting

**表 2** 实验中参数设定

| Max*Gen* | *PopSize* | $P_c$ | $P_m$ | $V$% |
|---|---|---|---|---|
| 80 | 300 | 0.8 | 0.05 | 5% |

针对本次实证研究我们设计了两个研究问题.

- RQ1:算法在跨项目缺陷预测上,与经典方法相比是否能够表现出较优的预测性能?
- RQ2:算法和同项目缺陷预测相比,是否能够表现出稳定的预测性能?

### 3.2 评测数据集

使用 Promise 及 AEEEM 数据集对我们提出的方法进行评价.Promise 数据集最早由 Jureczko 等人收集完成,其包含来自 10 个不同开源项目共计 29 个版本[42].每个项目实例均包含 20 个静态代码特征及类别标注,程序模块的粒度设置为类.表 3 描述了 Promise 数据集的统计信息,通过汇总可以看出,缺陷模块所占比例为 41.3%.AEEEM 数据集由 D'Ambros 等人搜集,共包含 5 个不同的开源项目[42].每个项目实例均包含基于软件代码、开发过程、熵等信息的 61 维度量元及一维类别标注.表 4 描述了 AEEEM 数据集的统计信息,通过汇总可以看出,缺陷模块所占比例为 16.3%.由于 AEEEM 数据集包含的项目较少,而项目来源各不相同,因此选择 AEEEM 数据集进行一对一模式下的跨项目缺陷预测.而 Promise 数据集中包含项目较多,但其中存在多个项目的不同版本数据,因此适用于多对一模式下的跨项目缺陷预测.

**Table 3** Characteristic information of Promise dataset

**表 3** Promise 数据集的统计信息

| 项目名称 | 代码行数 | 实例数 | 缺陷实例数(所占比例) |
|---|---|---|---|
| ant-1.3 | 37 699 | 125 | 20(16%) |
| ant-1.4 | 54 195 | 178 | 40(22.5%) |
| ant-1.5 | 87 047 | 293 | 32(10.9%) |
| ant-1.6 | 113 246 | 351 | 92(26.2%) |
| ant-1.7 | 208 653 | 745 | 166(22.3%) |
| log4j-1.0 | 21 549 | 135 | 34(25.2%) |
| log4j-1.1 | 19 938 | 109 | 37(33.9%) |
| log4j-1.2 | 38 191 | 205 | 189(92.2%) |
| lucene-2.0 | 50 596 | 195 | 91(46.7%) |
| lucene-2.2 | 63 571 | 247 | 144(58.3%) |
| lucene-2.4 | 102 859 | 340 | 203(59.7%) |
| poi-1.5 | 55 428 | 237 | 141(59.5%) |
| poi-2.0 | 93 171 | 314 | 37(11.8%) |
| poi-2.5 | 119 731 | 385 | 248(64.4%) |
| poi-3.0 | 129 327 | 442 | 281(63.3%) |
| redaktor | 59 280 | 176 | 27(15.3%) |
| synapse-1.0 | 28 806 | 157 | 16(10.2%) |
| synapse-1.1 | 42 302 | 222 | 60(27.0%) |
| synapse-1.2 | 53 500 | 256 | 86(33.6%) |
| tomcat | 300 674 | 858 | 77(9.0%) |
| velocity-1.4 | 51 713 | 196 | 147(75.0%) |
| velocity-1.6 | 57 012 | 229 | 78(34.1%) |
| xalan-2.4 | 225 088 | 723 | 110(15.2%) |
| xalan-2.5 | 304 860 | 803 | 387(48.2%) |
| xalan-2.6 | 411 737 | 885 | 411(46.4%) |
| xalan-2.7 | 428 555 | 909 | 898(98.8%) |
| xerces-1.2 | 159 254 | 440 | 71(16.1%) |
| xerces-1.3 | 167 095 | 453 | 69(15.2%) |
| xerces-1.4 | 141 180 | 588 | 437(74.3%) |
| 汇总 | 3 626 257 | 11 196 | 4629(41.3%) |

**Table 4** Characteristic information of AEEEM dataset

**表 4** AEEEM 数据集的统计信息

| 项目名称 | 代码行数 | 实例数 | 缺陷实例数(所占比例) |
|---|---|---|---|
| EQ | 39 534 | 325 | 129(39.6%) |
| JDT | 224 055 | 997 | 206(20.7%) |
| LC | 73 184 | 399 | 39(9.3%) |
| ML | 156 102 | 1 862 | 245(13.2%) |
| PDE | 146 952 | 1 492 | 209(14.0%) |
| 汇总 | 639 827 | 5 075 | 828(16.3%) |

### 3.3 评价指标

软件缺陷预测可以看做是二分类问题,其中,有缺陷模块为正例,无缺陷模块为负例.每个实例的分类过程中可能出现以下 4 种情况:实际为缺陷类同时被划分为缺陷类(true positive,简称 TP)、实际为无缺陷类被错误划分为缺陷类(false positive,简称 FP)、实际为缺陷类被错误划分为无缺陷类(false negative,简称 FP)、实际为无缺陷类被正确划分为无缺陷类(true negative,简称 TN).查全率(recall)和查准率(precision)是二分类问题中常用的评价正例分类能力的指标.查全率(recall)指代所有缺陷实例被正确预测为缺陷类的比例,其计算公式为

$$recall = \frac{TP}{TP + FN} \tag{8}$$

查准率(precision)指代所有被预测为缺陷类的实例中真实缺陷类所占的比例,其计算公式为

$$precision = \frac{TP}{TP + FP} \tag{9}$$

但查全率和查准率本身存在一定限制,难以单独用于评价分类性能优劣[43].好的分类性能应同时具有较高的查全率和查准率.$F1$ 指标是查准率和查全率的调和平均数,能够有效地平衡查准率和查全率,被广泛应用于

评价跨项目缺陷预测的性能优劣[1,2,10]及其他软件缺陷分析问题中[44−46],其计算公式为

$$F1=\frac{2\times precision\times recall}{precision+recall} \tag{10}$$

$F1$ 值越高,表示模型对缺陷类的预测能力越好.因此,实验选择 $F1$ 指标评价算法的预测性能.

### 3.4 实验结果

#### 3.4.1 针对 RQ1 分析

为了验证 S$^3$EL 方法的跨项目缺陷预测性能,我们分别在 AEEEM 数据集和 Promise 数据集上进行一对一模式及多对一模式下的跨项目缺陷预测.由于 HYDRA 方法本身的限制,不适用于一对一模式下的跨项目缺陷预测,因此这一部分不进行比较.由于 S$^3$EL 方法内部具有一定的随机因素,因此每次跨项目缺陷预测时,该方法均需独立执行 3 次,并取 3 次运行结果的均值.在 AEEEM 数据集上,一对一模式下的跨项目预测结果 $F1$ 值见表 5,箱线图如图 3 所示.

**Table 5** Comparison of $F1$ on one-to-one cross-project defect prediction

**表 5** 一对一模式下的跨项目缺陷预测 $F1$ 值的对比

| 源项目→目标项目 | 基准方法 | Burak filter | Peters filter | TCA+ | CODEP | S$^3$EL |
|---|---|---|---|---|---|---|
| JDT→EQ | 0.40 | 0.46 | 0.47 | 0.60 | 0.56 | **0.62** |
| LC→EQ | 0.48 | 0.48 | 0.40 | 0.62 | 0.52 | **0.70** |
| ML→EQ | 0.53 | 0.48 | 0.54 | 0.56 | 0.55 | **0.60** |
| PDE→EQ | 0.49 | 0.49 | 0.52 | 0.60 | 0.58 | **0.62** |
| EQ→JDT | 0.39 | 0.43 | 0.42 | 0.54 | 0.38 | **0.55** |
| LC→JDT | 0.44 | 0.43 | 0.43 | **0.56** | 0.48 | 0.54 |
| ML→JDT | 0.44 | 0.55 | 0.47 | 0.43 | 0.43 | **0.59** |
| PDE→JDT | 0.44 | 0.45 | **0.53** | 0.48 | 0.44 | 0.51 |
| EQ→LC | **0.32** | 0.29 | 0.28 | 0.27 | 0.26 | 0.27 |
| JDT→LC | 0.31 | **0.42** | 0.35 | 0.31 | **0.42** | 0.33 |
| ML→LC | 0.37 | **0.39** | 0.32 | 0.25 | 0.20 | 0.34 |
| PDE→LC | **0.38** | 0.33 | 0.28 | 0.33 | 0.37 | 0.20 |
| EQ→ML | 0.23 | 0.22 | 0.22 | 0.23 | 0.23 | **0.28** |
| JDT→ML | 0.31 | 0.32 | 0.31 | **0.36** | 0.33 | 0.35 |
| LC→ML | 0.21 | 0.22 | 0.22 | 0.29 | 0.22 | **0.32** |
| PDE→ML | 0.22 | 0.23 | 0.22 | 0.29 | 0.22 | **0.36** |
| EQ→PDE | 0.31 | 0.32 | 0.32 | **0.33** | 0.29 | **0.33** |
| JDT→PDE | 0.33 | 0.36 | 0.35 | 0.38 | **0.40** | **0.40** |
| LC→PDE | 0.36 | 0.38 | **0.40** | 0.37 | 0.36 | 0.37 |
| ML→PDE | 0.37 | 0.36 | **0.39** | 0.37 | 0.33 | 0.36 |
| 均值 | 0.36 | 0.38 | 0.37 | 0.41 | 0.38 | **0.44** |

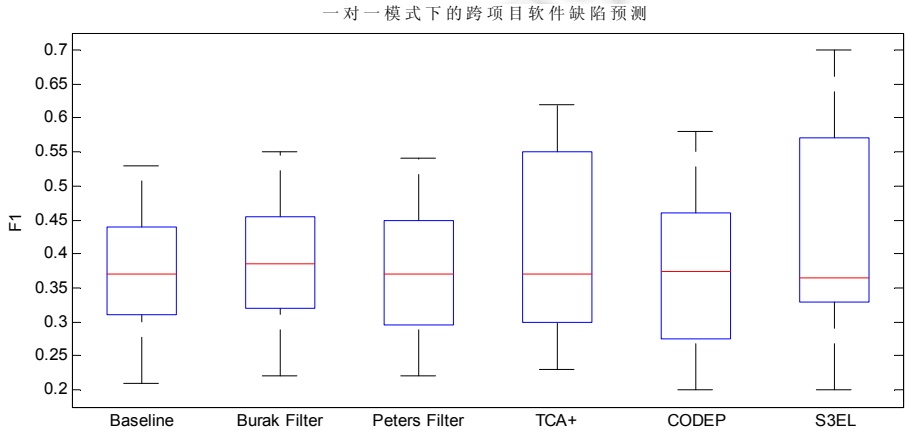


Fig.3 Box plot of one-to-one cross-project defect prediction

图 3 一对一跨项目缺陷预测结果箱线图

从实验结果可以看出:

(1) $S^3$EL 方法与 Burak 过滤法、Peters 过滤法、TCA+和 CODEP 方法相比,分别在 *F*1 指标的均值上提升了 16%,19%,7%和 16%;

(2) $S^3$EL 方法在一半以上(11 个)的预测结果中取得了最好的预测性能;

(3) 与基准方法相比,$S^3$EL 方法能够显著提升朴素贝叶斯的预测性能(22%).

通过对比其他方法,我们可以看出:基于实例选择的 Burak 过滤法和 Peters 过滤法的预测性能相差不大,并且与基准方法相比提升并不显著.可能的原因是:一对一模式下的跨项目缺陷预测中,训练实例与目标实例个数有限且通常相差不大,在这样的场景中进行实例选择,并不一定能够带来更好的结果,Burak 过滤法选取最近的 *k* 个实例和使用全集进行模型构建在样本数量上相差并不大.而 Peters 过滤法仅选取与每个有标记的目标实例距离最近的训练实例构建模型,导致训练集中的实例个数过少,信息不足,不能达到较好提升预测性能的结果.由此可见,基于实例选择的方法在实例数量相差不大的一对一模式下的跨项目缺陷预测中并不十分适用.

在 Promise 数据集上,多对一模式下的跨项目缺陷预测结果 *F*1 值见表 6,箱线图如图 4 所示.

从实验结果可以看出:

(1) $S^3$EL 方法对比 Burak 过滤法、Peters 过滤法、TCA+、CODEP 和 HYDRA 方法,分别在 *F*1 指标均值上提升了 38%,45%,35%,29%和 9%;

(2) $S^3$EL 方法在其中 12 个项目的缺陷预测上表现出最好的预测性能,HYDRA 方法在其中 10 个项目中表现出最优的预测性能;

(3) 与基准方法相比,$S^3$EL 方法在多对一模式下的跨项目缺陷预测中,能够大幅度提升 89%的预测性能.

**Table 6** Comparison of *F*1 on many-to-one cross-project defect prediction

**表 6** 多对一模式下的跨项目缺陷预测 *F*1 值对比

| 目标项目 | 基准方法 | Burak filter | Peters filter | TCA+ | CODEP | HYDRA | $S^3$EL |
|---|---|---|---|---|---|---|---|
| ant-1.3 | 0.35 | 0.28 | **0.44** | 0.28 | 0.36 | 0.42 | 0.39 |
| ant-1.4 | 0.21 | **0.37** | 0.31 | 0.35 | 0.35 | 0.25 | 0.36 |
| ant-1.5 | 0.42 | 0.26 | **0.44** | 0.21 | 0.35 | 0.14 | 0.41 |
| ant-1.6 | 0.45 | 0.38 | 0.51 | 0.39 | 0.53 | **0.62** | 0.61 |
| ant-1.7 | 0.50 | 0.38 | 0.51 | 0.33 | 0.47 | **0.52** | **0.52** |
| log4j-1.0 | 0.20 | 0.40 | **0.65** | 0.34 | 0.55 | 0.43 | 0.51 |
| log4j-1.1 | 0.27 | 0.51 | 0.53 | 0.49 | 0.63 | 0.53 | **0.65** |
| log4j-1.2 | 0.13 | 0.96 | 0.34 | 0.69 | 0.46 | 0.92 | **0.94** |
| lucene-2.0 | 0.18 | 0.58 | 0.47 | 0.54 | 0.53 | 0.62 | **0.67** |
| lucene-2.2 | 0.16 | 0.65 | 0.37 | 0.56 | 0.53 | 0.72 | **0.74** |
| lucene-2.4 | 0.31 | 0.67 | 0.39 | 0.58 | 0.59 | 0.71 | **0.74** |
| poi-1.5 | 0.26 | 0.73 | 0.43 | 0.55 | 0.72 | 0.71 | **0.75** |
| poi-2.0 | 0.24 | 0.14 | 0.28 | 0.22 | 0.20 | **0.36** | 0.31 |
| poi-2.5 | 0.24 | 0.16 | 0.22 | 0.61 | 0.59 | **0.81** | 0.79 |
| poi-3.0 | 0.28 | 0.50 | 0.23 | 0.61 | 0.49 | **0.83** | 0.81 |
| redaktor | 0.29 | 0.40 | 0.36 | 0.29 | 0.33 | **0.42** | 0.31 |
| synapse-1.0 | 0.38 | 0.18 | 0.20 | 0.21 | 0.24 | 0.17 | **0.39** |
| synapse-1.1 | 0.37 | 0.43 | **0.49** | 0.42 | **0.49** | 0.46 | 0.47 |
| synapse-1.2 | 0.42 | 0.50 | 0.57 | 0.44 | 0.57 | 0.22 | **0.61** |
| tomcat | 0.39 | 0.18 | 0.34 | 0.16 | 0.28 | 0.14 | **0.37** |
| velocity-1.4 | 0.11 | 0.19 | 0.15 | 0.66 | 0.33 | **0.86** | 0.76 |
| velocity-1.6 | 0.22 | 0.39 | 0.30 | 0.44 | **0.55** | 0.53 | 0.51 |
| xalan-2.4 | 0.36 | 0.28 | 0.38 | 0.25 | 0.34 | 0.37 | **0.39** |
| xalan-2.5 | 0.38 | 0.39 | 0.42 | 0.55 | 0.51 | **0.63** | 0.62 |
| xalan-2.6 | 0.44 | 0.39 | 0.57 | 0.51 | 0.55 | **0.68** | 0.63 |
| xalan-2.7 | 0.43 | 0.47 | 0.51 | 0.69 | 0.55 | 0.97 | **0.98** |
| xerces-1.2 | 0.24 | **0.28** | 0.22 | 0.23 | 0.23 | 0.27 | 0.27 |
| xerces-1.3 | 0.32 | 0.33 | **0.50** | 0.26 | 0.35 | 0.17 | 0.41 |
| xerces-1.4 | 0.25 | 0.85 | 0.39 | 0.67 | 0.40 | **0.88** | 0.87 |
| 均值 | 0.31 | 0.42 | 0.40 | 0.43 | 0.45 | 0.53 | **0.58** |

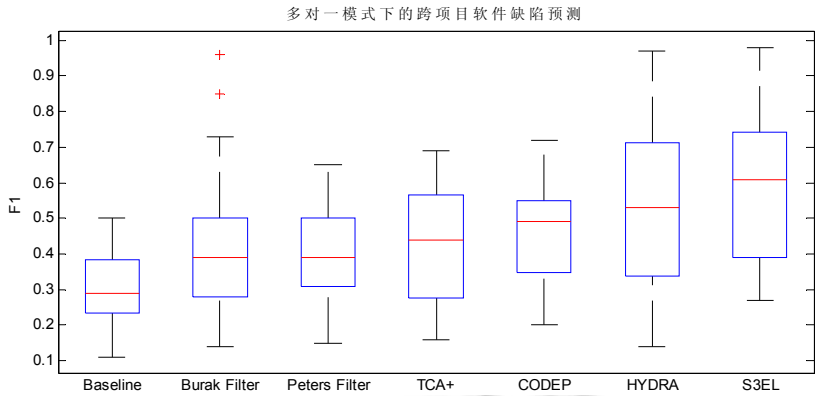


Fig.4 Box plot of many-to-one cross-project defect prediction

图 4 多对一跨项目缺陷预测结果箱线图

基于上述两组数据集的实验结果我们可以看出:与经典的跨项目缺陷预测方法相比,S$^3$EL 方法在多对一模式及一对一模式的预测场景中均能取得出平均最好的预测性能,因此初步验证了通过均值划分数据集的贝叶斯集成学习方法,能够显著提高朴素贝叶斯的预测能力.

3.4.2 针对 RQ2 的分析

为了验证 S$^3$EL 方法和同项目缺陷预测方法相比的性能优劣,本节在 Promise 数据集上分别用 5%和 90%的训练数据比例进行模型构建,并在剩余 95%和 10%的实例上进行缺陷预测.同项目预测模型同样使用朴素贝叶斯分类方法,随机抽样生成训练集,使用 *F*1 指标对预测性能进行评价.为了减小随机抽取训练集对结果的影响,方法均执行 3 次取平均 *F*1 值.表 7 给出了最终的实验结果.

**Table 7** Compared result of *F*1 value between our approach and other within-project defect prediction

**表 7** 方法与同项目缺陷预测方法的 *F*1 值对比

| 目标项目 | S$^3$EL | 5% | 90% |
|---|---|---|---|
| ant-1.3 | 0.39 | 0.21 | 0.33 |
| ant-1.4 | 0.36 | 0.25 | 0.17 |
| ant-1.5 | 0.41 | 0.34 | 0.5 |
| ant-1.6 | 0.61 | 0.52 | 0.59 |
| ant-1.7 | 0.52 | 0.39 | 0.65 |
| log4j-1.0 | 0.51 | 0.32 | 0.67 |
| log4j-1.1 | 0.65 | 0.49 | 0.86 |
| log4j-1.2 | 0.94 | 0.96 | 0.64 |
| lucene-2.0 | 0.67 | 0.49 | 0.67 |
| lucene-2.2 | 0.74 | 0.64 | 0.71 |
| lucene-2.4 | 0.74 | 0.73 | 0.45 |
| poi-1.5 | 0.75 | 0.67 | 0.38 |
| poi-2.0 | 0.31 | 0.16 | 0.3 |
| poi-2.5 | 0.79 | 0.78 | 0.74 |
| poi-3.0 | 0.81 | 0.81 | 0.42 |
| redaktor | 0.31 | 0.17 | 0.35 |
| synapse-1.0 | 0.39 | 0.11 | 0.36 |
| synapse-1.1 | 0.47 | 0.12 | 0.57 |
| synapse-1.2 | 0.61 | 0.09 | 0.62 |
| Tomcat | 0.37 | 0.25 | 0.4 |
| velocity-1.4 | 0.76 | 0.87 | 0.87 |
| velocity-1.6 | 0.51 | 0.39 | 0.4 |
| xalan-2.4 | 0.39 | 0.32 | 0.24 |
| xalan-2.5 | 0.62 | 0.41 | 0.21 |
| xalan-2.6 | 0.63 | 0.66 | 0.63 |
| xalan-2.7 | 0.98 | 0.99 | 0.99 |
| xerces-1.2 | 0.27 | 0.18 | 0.67 |
| xerces-1.3 | 0.41 | 0.37 | 0.45 |
| xerces-1.4 | 0.87 | 0.92 | 0.89 |
| 均值 | 0.58 | 0.47 | 0.55 |

结果表明:S$^3$EL 方法和 5%训练集的同项目缺陷预测相比,在指标均值上提升了 23%;和 90%训练集同项目缺陷预测相比,平均 *F*1 值相差不大,甚至略优,而半监督学习方法中仅用到 5%的样本做模型训练.可见,我们的方法跟与项目缺陷预测方法相比更具有应用价值.

3.4.3 实验结果的进一步分析

为了验证实验结论的可靠性,我们借助统计分析对实验结果进行方差检验.检验的假设是对比方法之间的平均预测性能 *F*1 没有显著差异,设置显著性水平为 0.05.我们分析一对一及多对一跨项目缺陷预测的 *F*1 指标,使用一种非参数的统计检验方法 Friedman 检验[47]来检验假设的可靠性.表 8 和表 9 分别显示了一对一模式及多对一模式下的缺陷预测实验的 Friedman 检验结果.*F* 值由组内平方和及组间平方和计算得到,*F* 值越大,表示拒绝原假设的可信度越高.*p* 值是 *F* 值对应的显著性水平,表 8 和表 9 中的结果 *p* 值都远小于显著性水平 0.05,所以原假设应该被拒绝,即一对一模式及多对一模式下的两组对比实验的结果均有显著差异.

**Table 8** Friedman test for one-to-one cross-project defect prediction

**表 8** 一对一模式下,实验结果的 Friedman 检验

| | 平方和 | 自由度 | 均方 | *F* 值 | *p* 值 |
|---|---|---|---|---|---|
| 组间误差 | 69.4 | 5 | 13.88 | 20.72 | 0.000 9 |
| 组内误差 | 265.6 | 95 | 2.795 8 | | |
| 总和 | 335 | 119 | | | |

**Table 9** Friedman test for many-to-one cross-project defect prediction

**表 9** 多对一模式下,实验结果的 Friedman 检验

| | 平方和 | 自由度 | 均方 | *F* 值 | *p* 值 |
|---|---|---|---|---|---|
| 组间误差 | 257.966 | 6 | 42.994 3 | 55.59 | 0.000 0 |
| 组内误差 | 549.534 | 168 | 3.271 | | |
| 总和 | 807.5 | 202 | | | |

Friedman 检验只能发现方法间的结果有显著差异,但是不能发现任意两种方法的具体差异性.因此,我们使用最小显著差异法 LSD(least-significant difference)来比较两种方法之间的差异.表 10 和表 11 显示了使用 LSD 方法[48]分别对一对一模式和多对一模式下的缺陷预测实验进行检验的结果.如果置信度下限和置信度上限之间没有包含 0(即同为正或同为负),则认为两种方法之间的差异是显著的.

**Table 10** LSD test for one-to-one cross-project defect prediction

**表 10** 一对一模式下,实验结果的 LSD 检验

| 方法 *x* | 方法 *y* | 置信下限 | 平均误差(*y*–*x*) | 置信上限 |
|---|---|---|---|---|
| **S$^3$EL** | **基准方法** | **0.550 613 739** | **2.2** | **3.849 386 261** |
| S$^3$EL | Burak filter | –0.124 386 26 | 1.525 | 3.174 386 261 |
| **S$^3$EL** | **Peters filter** | **0.025 613 739** | **1.675** | **3.324 386 261** |
| S$^3$EL | TCA+ | –0.999 386 26 | 0.65 | 2.299 386 261 |
| **S$^3$EL** | **CODEP** | **0.250 613 739** | **1.9** | **3.549 386 261** |
| 基准方法 | Burak filter | –2.324 386 26 | –0.675 | 0.974 386 261 |
| 基准方法 | Peters filter | –2.174 386 26 | –0.525 | 1.124 386 261 |
| 基准方法 | TCA+ | –3.199 386 26 | –1.55 | 0.099 386 261 |
| 基准方法 | CODEP | –1.949 386 26 | –0.3 | 1.349 386 261 |
| Burak filter | Peters filter | –1.499 386 26 | 0.15 | 1.799 386 261 |
| Burak filter | TCA+ | –2.524 386 26 | –0.875 | 0.774 386 261 |
| Burak filter | CODEP | –1.274 386 26 | 0.375 | 2.024 386 261 |
| Peters filter | TCA+ | –2.674 386 26 | –1.025 | 0.624 386 261 |
| Peters filter | CODEP | –1.424 386 26 | 0.225 | 1.874 386 261 |
| TCA+ | CODEP | –0.399 386 26 | 1.25 | 2.899 386 261 |

**Table 11** LSD test for many-to-one cross-project defect prediction

**表 11** 多对一模式下,实验结果的 LSD 检验

| 方法 $x$ | 方法 $y$ | 置信下限 | 平均误差($y$–$x$) | 置信上限 |
|---|---|---|---|---|
| **$S^3$EL** | **基准方法** | **2.004 447 527** | **3.672 413 793** | **5.340 380 059** |
| **$S^3$EL** | **Burak filter** | **0.866 516 493** | **2.534 482 759** | **4.202 449 025** |
| **$S^3$EL** | **Peters filter** | **0.521 688 906** | **2.189 655 172** | **3.857 621 438** |
| **$S^3$EL** | **TCA+** | **1.263 068 217** | **2.931 034 483** | **4.599 000 749** |
| **$S^3$EL** | **CODEP** | **0.194 102 7** | **1.862 068 966** | **3.530 035 231** |
| $S^3$EL | HYDRA | –0.616 242 13 | 1.051 724 138 | 2.719 690 404 |
| 基准方法 | Burak filter | –2.805 897 3 | –1.137 931 03 | 0.530 035 231 |
| 基准方法 | Peters filter | –3.150 724 89 | –1.482 758 62 | 0.185 207 645 |
| 基准方法 | TCA+ | –2.409 345 58 | –0.741 379 31 | 0.926 586 956 |
| **基准方法** | **CODEP** | **–3.478 311 09** | **–1.810 344 83** | **–0.142 378 56** |
| **基准方法** | **HYDRA** | **–4.288 655 92** | **–2.620 689 66** | **–0.952 723 39** |
| Burak filter | Peters filter | –2.012 793 85 | –0.344 827 59 | 1.323 138 68 |
| Burak filter | TCA+ | –1.271 414 54 | 0.396 551 724 | 2.064 517 99 |
| Burak filter | CODEP | –2.340 380 06 | –0.672 413 79 | 0.995 552 473 |
| Burak filter | HYDRA | –3.150 724 89 | –1.482 758 62 | 0.185 207 645 |
| Peters filter | TCA+ | –0.926 586 96 | 0.741 379 31 | 2.409 345 576 |
| Peters filter | CODEP | –1.995 552 47 | –0.327 586 21 | 1.340 380 059 |
| Peters filter | HYDRA | –2.805 897 3 | –1.137 931 03 | 0.530 035 231 |
| TCA+ | CODEP | –2.736 931 78 | –1.068 965 52 | 0.599 000 749 |
| **TCA+** | **HYDRA** | **–3.547 276 61** | **–1.879 310 34** | **–0.211 344 08** |
| CODEP | HYDRA | –2.478 311 09 | –0.810 344 83 | 0.857 621 438 |

根据显著性检验结果可以看出:在一对一模式下的跨项目缺陷预测中,我们的方法显著优于基准方法、Peters 过滤法及 CODEP 方法,略优于 Burak 过滤法及 TCA+方法,而其他几种方法之间差异并不显著.在多对一模式下的跨项目缺陷预测中,我们的方法显著优于基准方法、Burak 过滤法、Peters 过滤法、TCA+、CODEP 方法,略优于 HYDRA 方法.HYDRA 方法显著优于基准方法及 TCA+方法.CODEP 显著优于基准方法.其余方法之间的差异并不显著.

### 3.5 有效性影响因素分析

影响实证研究内部有效性的因素主要有:(1) $S^3$EL 方法实现及对比经典方法的重现过程中存在的隐藏缺陷;(2) 遗传算法中存在的随机因素对结果可能会产生一定的影响;(3) 不同的验证集分布可能导致结果差异较大.针对以上问题,本文在这些方法的实现过程中经过反复测试,以避免编码过程中可能产生的缺陷.将重现的经典方法的执行结果与之前文献中得到的研究结果进行比较,并确保基本一致.除此之外,还通过独立执行多次,借助取均值的方式降低方法内的随机因素可能给结果带来的影响.

影响实证研究外部有效性因素主要有:(1) 实证研究得到的结论是否具有一般性;(2) 实验中用到的数据集是否存在一定的质量问题.针对以上问题,本文的跨项目缺陷预测实验使用了两个不同的公开数据集 AEEEM 及 Promise,每个数据集包含不同的开源项目及不同的缺陷分布,并且已被相关研究者们多次使用[2,42,49].此外,跨项目缺陷预测分别在不同的数据集上考虑了多对一模式和一对一模式,从而确保了实验结论的可信度.

影响结论有效性的主要因素是使用的评测指标是否合理.针对缺陷预测问题,缺陷类的关注度要高于非缺陷类,$F1$ 指标能够平衡缺陷类预测性能的查全率及查准率,是跨项目缺陷预测领域最常用的重要评价指标之一[2,49−50].最后,我们还借助 Friedman 检验和 LSD 方法对一对一模式及多对一模式下的跨项目缺陷预测结果进行了显著性检验,进一步验证了结论的有效性

## 4 总结和展望

本文提出了一种基于搜索的半监督集成跨项目缺陷预测方法.实验结果表明:$S^3$EL 方法在多个公开数据集上与多种典型的跨项目缺陷预测方法相比均能获得较好的预测性能,并且能够有效提高朴素贝叶斯的预测能力.同时,$S^3$EL 方法也能够很好地适应多对一模式及一对一模式下的跨项目缺陷预测问题,因此具有很好的应

用价值.

我们认为,该方法仍有一些后续工作值得扩展,具体来说:(1) 我们将尝试进行验证集的选取实验,通过特定方式选取最能代表目标项目数据分布的少量实例作为验证集,借助相关代码人员的标注结果,达到更优更稳定的预测效果;(2) 尝试以特定的比例对数据划分的结果进行抽样,以进一步缓解类不平衡问题;(3) 在 $S^3EL$ 方法中进一步考虑特征选择方法[51,52],通过移除数据集中的冗余特征和无关特征来进一步提升跨项目缺陷预测的性能;(4) 考虑更多的数据集,对本文结论是否具有一般性进行验证.

**References**:


[1] Kim S, Whitehead EJ, Zhang Y. Classifying software changes: Clean or buggy? IEEE Trans. on Software Engineering, 2008,34(2): 181−196. [doi: 10.1109/TSE.2007.70773]

[2] 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. [doi: 10.1109/TSE.2016.2543218]

[3] Kim S, Zhang H, Wu R, Gong L. Dealing with noise in defect prediction. In: Proc. of the Int'l Conf. on Software Engineering. 2011. 481−490. [doi: 10.1145/1985793.1985859]

[4] Wang J, Shen B, Chen Y. Compressed C4.5 models for software defect prediction. In: Proc. of the Int'l Conf. on Quality Software. 2012. 13−16. [doi: 10.1109/QSIC.2012.19]

[5] Sun Z, Song Q, Zhu X. 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. [doi: 10.1109/TSMCC.2012.2226152]

[6] Zhou MH, Guo CG. New thinking of software engineering based on big data. Communications of the CCF, 2014,10(3):37−42 (in Chinese).

[7] Canfora G, Lucia AD, Penta MD, Oliveto R, Panichella A, Panichella S. Multi-Objective cross-project defect prediction. In: Proc. of the Int'l Conf. on Software Testing, Verification and Validation. 2013. 252−261. [doi: 10.1109/ICST.2013.38]

[8] Briand LC, Melo WL, Wust J. Assessing the applicability of fault-proneness models across object-oriented software projects. IEEE Trans. on Software Engineering, 2002,28(7):706−720. [doi: 10.1109/TSE.2002.1019484]

[9] Cruz AEC, Ochimizu K. Towards logistic regression models for predicting fault-prone code across software projects. In: Proc. of the Int'l Symp. on Empirical Software Engineering and Measurement. 2009. 460−463. [doi: 10.1109/ESEM.2009.5316002]

[10] Nam J, Pan SJ, Kim S. Transfer defect learning. In: Proc. of the Int'l Conf. on Software Engineering. 2013. 382−391.

[11] Pan SJ, Yang Q. A survey on transfer learning. IEEE Trans. on Knowledge and Data Engineering, 2010,22(10):1345−1359. [doi: 10.1109/TKDE.2009.191]

[12] Zhuang FZ, Ping L, Qing HE, 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]

[13] Pelayo L, Dick S. Evaluating stratification alternatives to improve software defect prediction. IEEE Trans. on Reliability, 2012, 61(2):516−525. [doi: 10.1109/TR.2012.2183912]

[14] Chen X, Gu Q, Liu WS, Liu SL, Ni C. 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]

[15] Harman M, Mansouri SA, Zhang YY. Search-Based software engineering: Trends, techniques and applications. ACM Computing Surveys, 2012,45(1):1−61. [doi: 10.1145/2379776.2379787]

[16] Turhan B, Menzies T, Bener AB, Di Stefano J. On the relative value of cross-company and within-company data for defect prediction. Empirical Software Engineering, 2009,14(5):540−578. [doi: 10.1007/s10664-008-9103-7]

[17] Peters F, Menzies T, Marcus A. Better cross company defect prediction. In: Proc. of the IEEE Working Conf. on Mining Software Repositories. 2013. 409−418. [doi: 10.1109/MSR.2013.6624057]

[18] Panichella A, Oliveto R, Lucia AD. Cross-Project defect prediction models: L'Union fait la force. In: Proc. of the IEEE Conf. on Software Maintenance, Reengineering and Reverse Engineering. 2014. 164−173. [doi: 10.1109/CSMR-WCRE.2014.6747166]

[19] He Z, Shu F, Yang Y, Li M, Wang Q. An investigation on the feasibility of cross-project defect prediction. Automated Software Engineering, 2011,19(2):167−199. [doi: 10.1007/s10515-011-0090-3]

[20] Malhotra R, Raje R. An empirical comparison of machine learning techniques for software defect prediction. In: Proc. of the Int'l Conf. on Bioinspired Information and Communications Technologies. 2014. 320−327. [doi: 10.4108/icst.bict.2014.257871]

[21] Lessmann S, Baesens B, Mues C, Pietsch S. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. IEEE Trans. on Software Engineering, 2008,34(4):485−496. [doi: 10.1109/TSE.2008.35]

[22] Ghotra B, McIntosh S, Hassan AE. Revisiting the impact of classification techniques on the performance of defect prediction models. In: Proc. of the Int'l Conf. on Software Engineering. 2015. 789−800. [doi: 10.1109/ICSE.2015.91]

[23] Zhang Y, Lo D, Xia X, Sun J. An empirical study of classifier combination for cross-project defect prediction. In: Proc. of the IEEE Computer Software and Applications Conf. 2015. 264−269. [doi: 10.1109/COMPSAC.2015.58]

[24] Ryu D, Choi O, Baik J. Value-Cognitive boosting with a support vector machine for cross-project defect prediction. Empirical Software Engineering, 2014,21(1):43−71. [doi: 10.1007/s10664-014-9346-4]

[25] Ryu D, Jang J, Baik J. A hybrid instance selection using nearest-neighbor for cross-project defect prediction. Journal of Computer Science and Technology, 2015,30(5):969−980. [doi: 10.1007/s11390-015-1575-5]

[26] Turhan B, Misirli AT, Bener A. Empirical evaluation of the effects of mixed project data on learning defect predictors. Information and Software Technology, 2013,55(6):1101−1118. [doi: 10.1016/j.infsof.2012.10.003]

[27] Zhong S, Khoshgoftaar TM, Seliya N. Unsupervised learning for expert-based software quality estimation. In: Proc. of the IEEE Int'l Symp. on High Assurance Systems Engineering. 2004. 149−155. [doi: 10.1109/HASE.2004.1281739]

[28] Zhang F, Zheng Q, Zou Y, Hassan AE. Cross-Project defect prediction using a connectivity-based unsupervised classifier. In: Proc. of the Int'l Conf. on Software Engineering. 2016. 309−320. [doi: 10.1145/2884781.2884839]

[29] Nam J, Kim S. CLAMI: Defect prediction on unlabeled datasets. In: Proc. of the Int'l Conf. on Automated Software Engineering. 2015. 452−463. [doi: 10.1109/ASE.2015.56]

[30] Concas G, Marchesi M, Pinna S, Serra N. Power-Laws in a large object-oriented software system. IEEE Trans. on Software Engineering, 2007,33(10):687−708. [doi: 10.1109/TSE.2007.1019]

[31] Jiang Y, Cukic B, Menzies T. Can data transformation help in the detection of fault-prone modules? In: Proc. of the Workshop on Defects in Large Software Systems. 2008. 16−20. [doi: 10.1145/1390817.1390822]

[32] Menzies T, Greenwald J, Frank A. Data mining static code attributes to learn defect predictors. IEEE Trans. on Software Engineering, 2007,33(1):2−13. [doi: 10.1109/TSE.2007.256941]

[33] Song Q, Jia Z, Shepperd M, Ying S, Liu J. A general software defect-proneness prediction framework. IEEE Trans. on Software Engineering, 2011,37(3):356−370. [doi: 10.1109/TSE.2010.90]

[34] Zhang F, Mockus A, Keivanloo I, Zou Y. Towards building a universal defect prediction model. In: Proc. of the Working Conf. on Mining Software Repositories. 2014. 182−191. [doi: 10.1145/2597073.2597078]

[35] Rahman F, Devanbu P. How, and why, process metrics are better. In: Proc. of the Int'l Conf. on Software Engineering. 2013. 432−441. [doi: 10.1109/ICSE.2013.6606589]

[36] Bacchelli A, D'Ambros M, Lanza M. Are popular classes more defect prone? Lecture Notes in Computer Science, 2010,6013: 59−73. [doi: 10.1007/978-3-642-12029-9_5]

[37] Nagappan N, Ball T. Use of relative code churn measures to predict system defect density. In: Proc. of the Int'l Conf. on Software Engineering. 2005. 284−292. [doi: 10.1109/ICSE.2005.1553571]

[38] Hassan AE. Predicting faults using the complexity of code changes. In: Proc. of the Int'l Conf. on Software Engineering. 2009. 78−88. [doi: 10.1109/ICSE.2009.5070510]

[39] Moser R, Pedrycz W, Succi G. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In: Proc. of the Int'l Conf. on Software Engineering. 2008. 181−190. [doi: 10.1145/1368088.1368114]

[40] Herzig K, Just S, Rau A, Zeller A. Predicting defects using change genealogies. In: Proc. of the Int'l Symp. on Software Reliability Engineering. 2013. 118−127. [doi: 10.1109/ISSRE.2013.6698911]

[41] Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA data mining software: An update. ACM SIGKDD Explorations Newsletter, 2009,11(1):10−18. [doi: 10.1145/1656274.1656278]

[42] Ambros MD, Lanza M, Robbes R. An extensive comparison of bug prediction approaches. In: Proc. of the IEEE Working Conf. on Mining Software Repositories. 2010. 31−41. [doi: 10.1109/MSR.2010.5463279]

[43] Agarwal S. Data mining: Data mining concepts and techniques. In: Proc. of the Int'l Conf. on Machine Intelligence and Research Advancement. 2013. 203−207. [doi: 10.1109/ICMIRA.2013.45]

[44] Nguyen AT, Nguyen TT, Nguyen HA, Nguyen TN. Multi-Layered approach for recovering links between bug reports and fixes. In: Proc. of the ACM SIGSOFT Int'l Symp. on the Foundations of Software Engineering. 2012. 1−11. [doi: 10.1145/2393596.2393671]

[45] Tian Y, Lawall J, Lo D. Identifying Linux bug fixing patches. In: Proc. of the Int'l Conf. on Software Engineering. 2012. 386−396. [doi: 10.1109/ICSE.2012.6227176]

[46] Wu R, Zhang H, Kim S, Cheung SC. ReLink: Recovering links between bugs and changes. In: Proc. of the ACM SIGSOFT Symp. and the European Conf. on Foundations of Software Engineering. 2011. 15−25. [doi: 10.1145/2025113.2025120]

[47] Friedman M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association, 1937,32(200):675−701. [doi: 10.1080/01621459.1937.10503522]

[48] Wilcoxon F. Individual comparisons by ranking methods. Biometrics, 1945,1(6):80−83. [doi: 10.2307/3001968]

[49] Cao Q, Sun Q, Cao Q, Tan H. Software defect prediction via transfer learning based neural network. In: Proc. of the Int'l Conf. on Reliability Systems Engineering. 2015. 1−10. [doi: 10.1109/ICRSE.2015.7366475]

[50] Xu Z, Xuan J, Liu J, Cui X. MICHAC: Defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering. In: Proc. of the IEEE Int'l Conf. on Software Analysis, Evolution, and Reengineering. 2016. 370−381. [doi: 10.1109/SANER.2016.34]

[51] Liu SL, Chen X, Liu WS, Chen JQ, Gu Q, Chen DX. FECAR: A feature selection framework for software defect prediction. In: Proc. of the Annual Int'l Computers, Software and Applications Conf. 2014. 426−435. [doi: 10.1109/COMPSAC.2014.66]

[52] Liu WS, Liu SL, Gu Q, Chen JQ, Chen X, Chen DX. Empirical studies of a two-stage data preprocessing approach for software fault prediction. IEEE Trans. on Reliability, 2016,65(1):38−53. [doi: 10.1109/TR.2015.2461676]

**附中文参考文献**:

[6] 周明辉,郭长国.基于大数据的软件工程新思维.计算机学会通讯,2014,10(3):37−42.

[12] 庄福振,罗平,何清,等.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]

[14] 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]

**何吉元**(1992－),女,江苏涟水人,硕士生,主要研究领域为软件测试,机器学习.

**孟昭鹏**(1962－),男,博士,教授,博士生导师,主要研究领域为机器学习,软件工程.

**陈翔**(1980－),男,博士,副教授,CCF 专业会员,主要研究领域为软件缺陷预测,软件缺陷定位,回归测试和组合测试.

**王赞**(1979－),男,博士,副教授,主要研究领域为软件工程,软件测试,机器学习,软件质量.

**樊向宇**(1992－),男,硕士生,CCF 学生会员,主要研究领域为软件测试,机器学习.