软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.000000]

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

基于特征迁移和实例迁移的跨项目缺陷预测方法*

倪 超[1], 陈 翔[1,2], 刘望舒[1,3], 顾 庆[1], 黄启国[1], 李 娜[1]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210023)

[2](计算机科学与技术学院(南通大学),江苏 南通 226019)

[3](计算机科学与技术学院(南京工业大学),江苏 南京 211816)

通讯作者: 顾庆，陈翔，E-mail: guq@nju.edu.cn，xchencs@ntu.edu.cn

摘 要: 在实际软件开发中，需要进行缺陷预测的项目可能是一个新启动项目，或这个项目的历史训练数据较为稀缺。一种解决方案是利用其他项目（即源项目）已搜集的训练数据来构建模型，并完成对当前项目（即目标项目）的预测。但不同项目的数据集间会存在较大的分布差异性。针对该问题，论文首次同时从特征迁移和实例迁移角度出发，提出了一种两阶段跨项目缺陷预测方法 FeCTrA。具体来说，在特征迁移阶段，该方法借助聚类分析，选出源项目与目标项目之间具有高分布相似度的特征。在实例迁移阶段，该方法基于 TrAdaBoost 方法，借助目标项目中的少量已标注实例，从源项目中选出与这些已标注实例分布相近的实例。为了验证 FeCTrA 方法的有效性，我们选择 Relink 数据集和 AEEEM 数据集作为评测对象，以 F1 作为评测指标。首先我们发现 FeCTrA 方法的预测性能要优于仅考虑特征迁移阶段或实例迁移阶段的单阶段方法。其次与经典的跨项目缺陷预测方法 TCA+、Peters 过滤法、Burak 过滤法以及 DCPDP 法相比，FeCTrA 方法的预测性能在 Relink 数据集上可以分别提升 23%、7.2%、9.8%和 38.2%，在 AEEEM 数据集上可以分别提升 96.5%、108.5%、103.6% 和 107.9%。最后我们分析了 FeCTrA 方法内的影响因素对预测性能的影响，从而为有效使用 FeCTrA 方法提供了指南。

关键词: 软件质量保障;软件缺陷预测;跨项目缺陷预测;迁移学习;特征迁移;实例迁移

中图法分类号: TP311

中文引用格式:xxxxx

英文引用格式: xxxxx

Cross-Project Defect Prediction Method Based on Feature Transfer and Instance Transfer

NI Chao[1], CHEN Xiang[1,2], LIU Wang-Shu[1,3], GU Qing[1], Huang Qi-Guo[1], Li Na[1]

[1](State Key Laboratory for Novel Software Technology(Nanjing University), Nanjing 210023, China)

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

[3](School of Computer Science and Technology, Nanjing Tech University, Nanjing 211816, China)

Abstract: In real software development, a project, which needs defect prediction, may be a new project or maybe has less training data. A simple solution is to use training data from other projects (i.e., source projects) to construct the model, and use the trained model to perform prediction on the current project (i.e., target project). However, datasets between different projects may have large distribution difference. To solve this problem, we first propose a novel two phase cross-project defect prediction method FeCTrA, which considers

* 基金项目: 国家自然科学基金（61373012, 61202006, 91218302, 61321491）; 南京大学计算机软件新技术国家重点实验室开放课题（KFKT2016B18, KFKT2018B17）Foundation item: National Natural Science Foundation of China (61373012, 61202006, 91218302,61321491); The Open Project of State Key Laboratory for Novel Software Technology at Nanjing University(KFKT2016B18); 江苏省自然科学基金资助项目(No.BK20180695); 中国国家留学基金管理委员会(No.201806190172).

收稿时间: 0000-00-00; 修改时间: 0000-00-00; 采用时间: 0000-00-00; jos 在线出版时间: 0000-00-00

CNKI 在线出版时间: 0000-00-00

both feature transfer and instance transfer. In the feature transfer phase, FeCTrA uses cluster analysis to select features, which have high distribution similarity between the source project and the target project. In the instance transfer phase, FeCTrA utilizes TrAdaBoost, which selects relevant instances from the source project when give some labeled instances in the target project. To verify the effectiveness of FeCTrA, we choose Relink and AEEEM datasets as our experimental subjects and F1 as our performance measure. First, we find FeCTrA outperforms single phase methods, which only consider feature transfer or instance transfer. Then after comparing with state-of-the-art baseline methods (i.e., TCA+, Peters filter, Burak filter, and DCPDP), the performance of FeCTrA improves 23%, 7.2%, 9.8%, and 38.2% on Relink dataset and the performance of FeCTrA improves 96.5%, 108.5%, 103.6%, and 107.9% on AEEEM dataset. Finally, we analyze the influence of factors in FeCTrA and provide a guideline to effectively use this method.

Key words: software quality assurance; software defect prediction; cross-project defect prediction; transfer learning; feature transfer; instance transfer

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

软件缺陷预测（software defect prediction，简称 SDP）[1-3] 是其中一种可行方法，根据软件历史开发数据以及已发现的缺陷，借助机器学习等方法来预测出软件项目内的潜在缺陷程序模块。目前，大部分研究工作关注项目内的软件缺陷预测（within-project defect prediction，简称 WPDP）问题，即基于同一项目的数据完成软件缺陷预测模型的构建和预测。通常，构建一个性能较好的缺陷预测模型需要大量的训练实例。但是，现实中收集并标注足够多的训练数据相当困难。一方面，一些新开发项目只含有较少的训练数据，另一方面，标记数据需要耗费大量的人力和物力，且这个过程容易出错。于是，跨项目软件缺陷预测（cross-project defect prediction，简称 CPDP）[4-11] 问题被研究人员提出，其基于其他项目（即源项目）的历史数据来构建软件缺陷预测模型，然后在当前项目（即目标项目）上进行缺陷预测。然而，由于源项目和目标项目之间的数据分布在大部分时候都具有很大的差异性，这导致了在源项目上构建的模型在目标项目上很难具有良好的预测性能。因此，缩小源项目和目标项目之间的数据分布差异性是设计跨项目缺陷预测方法需要重点考虑的问题。

针对该问题，论文首次同时从特征迁移和实例迁移角度出发，提出了一种两阶段跨项目缺陷预测方法 FeCTrA（cross-project software defect prediction using Feature Clustering and TrAdaBoost）。具体来说，在特征迁移阶段，该方法借助聚类分析，选出源项目与目标项目之间具有高分布相似度的特征。在实例迁移阶段，该方法借助 TrAdaBoost 方法，基于目标项目中的少量已标注实例，从源项目中选出与这些已标注实例分布相近的实例。为了验证 FeCTrA 方法的有效性，我们选择具有代表性的、开源的、真实项目的 Relink 数据集和 AEEEM 数据集作为评测对象，以 F1 作为评测指标。首先我们发现 FeCTrA 方法的预测性能要优于仅考虑特征迁移阶段或实例迁移阶段的单阶段方法。其次与经典的跨项目缺陷预测方法 TCA+[12]、Peters 过滤法[13]、Burak 过滤法[14]以及 DCPDP 法[15] 相比，FeCTrA 方法的预测性能在 Relink 数据集上可以分别提升 23%，7.2%，9.8%和 38.2%，在 AEEEM 数据集上可以分别提升 96.5%、108.5%、103.6%和 107.9%。基于上述分析，我们发现 FeCTrA 方法不仅可以有效移除冗余特征和无关特征，而且可以从源项目中选出与目标项目分布更为相似的实例。此外，实证研究结果也初步验证了 FeCTrA 方法的有效性。

论文的主要贡献可总结如下：

（1）首次提出了基于特征迁移和实例迁移的跨项目软件缺陷预测方法 FeCTrA，该方法可以有效地移除冗余特征和无关特征，并且可以从源项目中选出与目标项目分布相似的实例，从而有效减小源项目和目标项目之间的数据分布差异性。

（2）在实证研究中考虑了基于实际项目的 Relink 和 AEEEM 数据集，深入分析了不同影响因素对 FeCTrA

方法性能的影响，为更好地使用 FeCTrA 方法提供了指导性建议。

（3）通过将 FeCTrA 方法与 Nam 等人提出的 TCA+[12]、Peters 等人提出的 Peters 过滤法[13]、Turhan 等人提出的 Burak 过滤法[14]以及 Zimmermann 等人提出 DCPDP 法[15]进行比较，发现 FeCTrA 方法在预测性能上具有显著优势，这说明通过融合多种不同类型的迁移学习方法来解决 CPDP 问题值得关注。

论文剩余内容结构安排如下：第 1 节首先介绍软件缺陷预测的研究背景和相关研究工作。第 2 节介绍 FeCTrA 方法及其具体实现细节。第 3 节介绍论文的实证研究，包括研究问题、评测对象、评测指标、显著性检验方法、实验流程及其方法参数设置。第 4 节对实证研究结果进行了详细分析。最后总结全文并对下一步研究工作进行了展望。

1 研究背景与相关工作

1.1 软件缺陷预测

软件缺陷预测[1-3, 16] 是当前软件工程数据挖掘领域的一个研究热点[11, 17-22]。缺陷预测可以在软件发布之前尽可能多的预测出潜在缺陷程序模块，以便于合理分配测试资源，从而最终提高软件质量。其主要包括两个阶段：模型构建阶段和模型应用阶段。模型构建阶段主要包括如下三个步骤：（1）挖掘软件历史仓库。目前可供挖掘与分析的软件历史仓库包括项目所处的版本控制系统（例如 CVS、SVN 或 Git 等），缺陷跟踪系统（例如 Bugzilla、Mantis、Jira 或 Trac 等）或团队开发人员间相互发送的电子邮件等。程序模块的粒度根据实际应用场景的需要，可以设置为包、文件、类、函数或代码修改等。（2）程序模块的度量和标记。通过分析软件代码复杂度或开发过程特征，可以设计出与软件缺陷存在相关性的度量元（metrics）[23]。通过这些度量元可以对程序模块进行度量，随后通过分析缺陷报告可以完成对这些程序模块的标记。（3）模型的构建。基于搜集的缺陷预测数据集，首先进行必要的数据预处理（例如特征选择、数据取值归一化、噪音移除等）[17, 24-28]，随后可以基于特定的机器学习方法（例如 Logisitic 回归、随机森林、支持向量机等）完成模型的构建。而在模型应用阶段，当面对新的程序模块时，在完成对该模块的软件度量后，基于已构造出的缺陷预测模型和具体度量元取值，可以完成对该模块的预测，即预测为潜在缺陷模块或无缺陷模块。

1.2 跨项目软件缺陷预测

目前，绝大部分软件缺陷预测方法基于同一项目上的数据进行模型训练和模型预测，这类方法被称为项目内缺陷预测方法，而充足的标记数据是保障这类方法取得良好预测性能的前提。然而，对于刚开发的项目或者历史遗留项目，可能难以获取足够多的标记数据，因此难以构建出具有良好预测性能的模型。针对这种情况，跨项目缺陷预测[12, 15, 29]应运而生。跨项目缺陷预测基于其他项目（即源项目）的标记数据进行模型训练，并在当前项目（即目标项目）上进行缺陷预测。然后，由于源项目和目标项目之间存在较大的数据分布差异性，因此在源项目上训练的模型在目标项目上未必能够取得良好的预测性能[15]。因此，设计新颖高效的跨项目软件缺陷预测方法具有一定的研究挑战性，研究人员针对该问题提出了多种解决方法。

Zimmermann 等人[15] 首先针对 CPDP 的可行性展开了大规模的实证研究，然而研究结果并不乐观。Turhan 等人[14] 提出了基于 k 近邻的 Burak 过滤法。具体而言，该方法首先计算出目标项目中实例与候选源项目中实例之间的欧式距离，然后为目标项目中的每一个未标注实例从源项目中选出距离其最近的 k（k 为 10）个实例，最后将所有从源项目中选出的实例构成训练集。实验结果表明 Burak 过滤法虽然可以提升 CPDP 的性能，但是仍然低于 WPDP 方法。不同于 Turhan 等人从目标项目出发进行实例选择, Peters 等人[13] 则认为源项目中的数据包含更多的信息。因此他们提出 Peters 过滤法。具体而言，首先针对源项目中的每个实例，从目标项目中识别出与之距离最近的实例并进行标记。随后对于目标项目中已标记的实例，从源项目中选出与之距离最近的实例并添加到最终训练集中。Ma 等人[30] 则提出 TNB（Transfer naive bayes）方法，它们认为应该为源项目中与目标项目中实例相似的实例赋予更高的权重。Wang 等人[31] 则基于深度学习，从模块代码中自动学出语义表示。Chen 等人[32]提出的方法则考虑从源项目中删除带有负面影响的实例。Xia 等人

[6]提出了两阶段框架 Hydra，该框架引入了遗传算法和集成学习以获取源项目和目标项目之间的共有信息。

一些研究人员针对异构 CPDP 问题展开了深入研究，该问题尝试在源项目和目标项目间具有不同特征空间的情况下，进行跨项目缺陷预测。针对该问题，Turhan 等人[14] 提出了一种简单方法，即在构建 CPDP 模型的时候，仅考虑源项目和目标项目之间共有的特征。显然，这种方法存在两点不足，首先源项目和目标项目之间共有的特征通常较少，甚至有时候没有；其次仅考虑共有的特征，会遗漏掉大量其他有用的信息。针对上述不足，He 等人[33] 首先提出了 CPDP-IFS 方法。该方法将每一个实例视为向量，并计算其分布特征指标的取值，从而可以将目标项目和源项目中的实例映射到一个潜在空间，该潜在空间由实例的分布特征指标构成，从而确保 CPDP 可以在同一个特征空间内进行。Nam 和 Kim[34] 提出 HDP（heterogeneous defect prediction）方法，该方法包含特征选择阶段和特征映射阶段。Jing 等人[20] 提出了 UMR（unified metric representation）表示，随后借助典型相关分析（canonical correlation analysis）来减少源项目和目标项目间的数据分布差异性。

除此之外，一些研究人员考虑使用无监督方法来解决 CPDP 问题。Zhong 等人[35]使用 k-means 和 NeuralGas 方法对程序模块进行聚类分析，然后他们从每一个簇中选出典型模块，并将一些统计信息（如特征的均值、最大值、最小值等）提供给专家，最终交由专家完成对簇的标记。Nam 和 Kim [22] 提出了一种自动方法 CLA。CLA 方法会依据每个特征的中位数统计该实例含有的异常特征数，然后将具有相同异常数的实例划分到同一个簇中，随后将高于一定异常特征数的实例标记为有缺陷实例，否则标记为无缺陷实例。在 CLA 方法的基础上，Nam 等人又通过特征选择和实例选择来移除数据集中噪声，并提出 CLAMI 方法。Zhang 等人[36]借助谱聚类（spectral clustering）方法，基于程序模块间的连通性完成对数据集的划分。最近，Yang 等人[37]借助代价感知的评测指标进行性能评估时，意外的发现一些简单的无监督方法比有监督方法可以取的更好的预测性能。随后，Zhou 等人[38]针对无监督方法（即 ManualDown 和 ManualUp 方法）展开了更大规模的实证研究。

1.3 迁移学习

近些年来，迁移学习（transfer learning）引起了广泛的研究和关注[39-43]。迁移学习是运用已存在的知识对不同但相关的领域问题进行求解的一种学习方法，该方法放宽了传统机器学习中两个基本假设：（1）用于学习的训练样本与新的测试样本需要满足独立同分布的假设；（2）必须有足够的已标记样本才能训练出一个好的分类模型。该方法旨在迁移已有的知识来解决目标领域中仅有少量有标记样本数据甚至有时候没有的学习问题。跨项目软件缺陷预测可以视为迁移学习在软件缺陷预测领域中的一个重要应用。

目前迁移学习方法可以从两个角度进行分类。基于“迁移什么”角度，已有方法可以细分为四类：基于特征的迁移学习、基于实例的迁移学习、基于参数的迁移学习和基于相关知识的迁移学习。基于“如何迁移”角度，已有方法可以细分为三类：（1）归纳式迁移学习：目标领域中有少量标注样本；（2）直推式迁移学习：只有源领域中有标签样本；（3）无监督迁移学习：源领域和目标领域都没有标签样本。表 1 总结了传统机器学习方法和不同类型的迁移学习方法间的关系。

Table 1 The relationship between traditional machine learning methods and different types of transfer learning methods

表 1 传统机器学习方法和不同类别迁移学习方法的关系

机器学习方法	源领域数据和目标领域数据	源领域任务和目标领域任务
传统机器学习方法	相同	相同
归纳迁移学习	相同/不同但相关	不同但相关
直推式迁移学习	不同但相关	相同
无监督迁移学习	相同/不同但相关	不同但相关

目前，跨项目缺陷预测主要从“迁移什么”的角度展开研究，其中“基于特征的迁移学习”和“基于实例的迁移学习”是研究人员关注较多的方法，这也是我们主要考虑的角度。基于特征迁移学习的方法尝试寻找在源领域和目标领域之间具有相同或者相似性质的特征，从而达到从源领域将知识迁移到目标领域。一些跨项目缺陷预测法方法[7, 8, 12, 14, 20, 33, 34] 从这个角度展开研究。而基于实例的迁移学习方法则根据目标领域中数据的部分知识从源领域中挑选有价值的实例。这类方法通常对源领域中的实例进行选择或权重设置。一些跨项目缺陷预测方法[6, 13, 14, 30, 32] 从这个角度展开研究。但上述研究工作仅从单一角度将源项目中的信息迁移到目标项目中。据我们所知，论文首次将基于特征的迁移学习和基于实例的迁移学习相结合，提出基于直推式迁移学习方式的 FeCTrA 方法。

2 基于特征迁移和实例迁移的缺陷预测方法 FeCTrA

本节首先介绍基于特征迁移和实例迁移的跨项目预测方法 FeCTrA 的研究动机，然后对 FeCTrA 方法的整体框架进行描述，最后对框架内的特征间相关性和特征分布相似性的度量方法进行介绍。

2.1 研究动机

软件缺陷预测基于项目的历史数据构建模型，然后对新的软件模块进行缺陷预测。但在实际软件开发过程中，新开发的项目可能没有充足的训练数据，或者一些遗留项目因为特殊原因而无法获得足够多的训练数据。因此构建一个具有良好性能的缺陷预测模型变得异常困难。于是迁移学习被引入到软件缺陷预测研究中，借助源项目中的历史标记数据以解决目标项目训练实例过少的问题。然而，由于源项目和目标项目之间存在较大的数据分布差异，使得在源项目数据上构建的缺陷预测模型，并不能保证在目标项目上取得良好的预测性能。一方面，从特征角度而言，源项目中并不是所有的特征都与目标项目具有相似的分布，只有分布相似的特征才能辅助目标项目构建性能良好的缺陷预测模型。另一方面，从实例角度而言，源项目数据和目标项目数据本身源于不同的项目，因此通常情况下，源项目中的数据与目标项目数据的分布具有差异性。针对该问题，研究人员分别从特征迁移或者实例迁移的角度展开研究[6-8, 12-14, 30, 32]，并取得了一定成果，这充分证明了从特征迁移和实例迁移角度尝试缩小源项目与目标项目之间数据分布差异性的可行性。然而，目前取得的效果并不令人十分满意。为此，我们提出了两阶段跨项目缺陷预测方法 FeCTrA，该方法通过同时考虑特征迁移和实例迁移，旨在缩小两个项目的数据分布差异性。在特征迁移阶段中，通过聚类分析，可以识别并移除无关特征和冗余特征。在实例迁移阶段中，依据目标项目中仅有的少量标记数据，借助 TrAdaBoost 方法[41] 从源项目中选出与目标项目数据分布更相近的实例，从而解决训练数据不足的问题。

2.2 方法框架

FeCTrA 方法的框架如图 1 所示，该方法包含两个阶段：特征迁移阶段和实例迁移阶段。在特征迁移阶段，为了能够迁移有效的特征信息，FeCTrA 方法首先移除源项目数据中的类标信息，然后将源项目和目标项目数据进行合并。随后，对合并后的数据集基于特征进行聚类分析，从而把高度相关的特征聚集到同一个簇中。然后，计算每一个特征在源项目数据和目标项目数据之间分布的相似性，以此作为排序依据，并对每一个簇中的特征进行降序排列，最后从每一个簇中选取排名靠前的特征作为最终需要迁移的特征。在实例迁移阶段，删除源项目和目标项目中不必要的特征，仅保留特征迁移阶段选出的特征和源项目的类标特征。然后，利用 TrAdaBoost 方法[41]，从源项目中选出与目标项目分布相似的数据来构建训练数据集，并通过 Boost 方法不断迭代，增强基分类器的分类能力，得到若干基分类器，从而构成一个基于集成学习方式的跨项目缺陷预测模型。

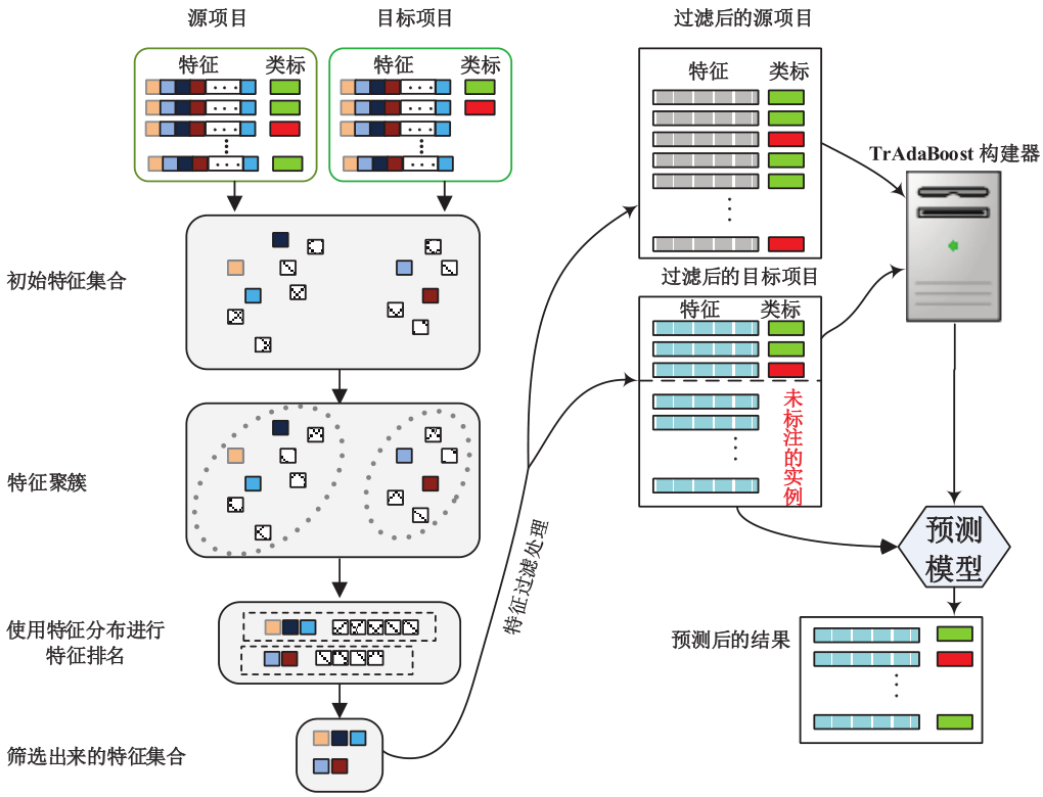

Fig.1 Cross-project software defect prediction framework based on feature transfer and instance transfer

图 1 基于特征迁移和实例迁移的跨项目软件缺陷预测框架

2.2.1 特征迁移阶段

为了更好地描述该阶段，首先给出两个定义。

定义 1 (特征间相关性(Inter-Feature Correlation，简称 IFC)). $IFC(f_i, f_j)$ 表示特征 f_i 和特征 f_j 之间的相关性，其中 i 和 j 不相同。

$IFC(f_i, f_j)$ 的取值范围是[0,1]。$IFC(f_i, f_j)$ 取值越高，表明 f_i 和 f_j 之间的相关性越高。其中当 $IFC(f_i, f_j)=0$ 时，表明特征 f_i 和特征 f_j 之间完全独立；当 $IFC(f_i, f_j)=1$，表明特征 f_i 和特征 f_j 之间完全相关。论文中使用的特征间相关性度量方法如 2.2.3 节所述。

定义 2 (特征分布相似性(Similarity of Feature Distribution，简称 SFD)). $SFD(f_i)$ 表明特征 f_i 在源项目和目标项目上分布的相似性。

$SFD(f_i)$ 的取值范围是 $[0,+\infty)$。$SFD(f_i)$ 取值越大，表明该特征在两个数据集上分布越相似。显然，$SFD(f_i)$ 取值越大越好。其中，$SFD(f_i)=0$ 表明该特征在两个数据集上分布完全不相似。论文中使用的特征分布相似性度量方法如 2.2.4 节所述。

算法 1 中给出了特征迁移阶段的详细描述。该算法是对 K -medoids 算法的扩展，主要目标是找出 K 个

最具有代表性的特征，这 K 个特征能够使得各个簇之间的距离最大化并且簇内特征之间的距离最小化。在算法 1 中，其输入是源项目数据 S 、目标项目数据 T 、需要选出的特征数 m 以及簇的个数 k 。算法的输出是最终需要迁移的特征集合 FS。

算法 1：Feature Transfer Phase

Input:

S: Source project
T: Target project
m: The number of selected features
k: The number of clusters

Output:

FS: The selected feature subset

1: Remove the class label from both S and T;
2: Combine S and T as *DatasetAll* which obtains feature set $F = \{ f_1, f_2, f_3, \cdots, f_n \}$
3: Compute $IFC(f_i, f_j)$ between each two different features;
4: Calculate $SFD(f_i)$ for each feature between S and T ;
5: Cluster all feature into K cluster with $IFC(f_i, f_j)$ and $SFD(f_i)$;
6: Sort features in each cluster in the descending order according to $SFD(f_i)$;
7: Select top $\left\lceil |C_i| \times \frac{m}{n} \right\rceil$ in each cluster, note that n is the number of features without the class feature, $|C_i|$ is the size of the i -th cluster;
8: Put all selected features into *FS*.
9: **RETURN** *FS*

在算法 1 中，首先移除源项目和目标项目数据中的类标特征（Line 1）。然后将过滤掉类标特征的源项目数据和目标项目数据进行合并，形成一个完整的数据集（Line 2）。随后计算任意两个不同特征之间的相关性 $IFC(f_i, f_j)$ 以及同一个特征在两个项目之间的分布相似性 $SFD(f_i)$（Lines 3-4）。计算完这两项指标之后执行特征聚类、特征排序和特征选择过程。在特征聚类过程中（Line 5），$IFC(f_i, f_j)$ 指标被用来完成特征聚类，特征聚类的目的就是将初始的特征集合分配到 k 个簇中。在这些簇中，每一个簇里的任意两个特征 f_i 和 f_j $(i \neq j)$都是高度相关的，但是两个不同簇之间的特征相关度却很低。整个聚类过程如下：首先，$SFD(f_i)$ 指标将被用于完成 k 个簇中心的初始化。在此过程中，为了避免选择不相关的特征作为初始化阶段中的簇中心并提高聚类过程的收敛速度，我们首先选择在源项目和目标项目中分布相似度最高的前 k 个特征，然后根据特征与这 k 个簇中心之间的相关度，依次指派每一个特征到与之相关度最高的簇中，直至所有特征都属于一个特定的簇。接着重新计算各个簇的中心，按照相同的操作过程更新簇中心并重新指派各个特征到最相关的簇中，重复此过程，直至簇中心不在发生变化为止。在特征排序和特征选择过程中（Lines 6-7），仅使用 $SFD(f_i)$ 指标完成。在该过程中，对于每一个簇中的所有特征进行排序后，从各个簇中选取特定的特征作为最终需要保留的特征，即最终需要迁移的特征。由于每一个簇的大小不尽相同，因此我们考虑了每一个簇的规模，并根据簇的大小从对应的簇中选出相应比例的特征。也就是说，从含有特征越多的簇中选取越多的特征，反之则越少。因此，将每个簇中特征根据 $SFD(f_i)$ 取值进行降序排序后，从每个簇中选取 $\left\lceil |C_i| \times \frac{m}{n} \right\rceil$ 个特征。其中，m 表示最终需要选择的特征个数，n 表示除类标以外的所有特征个数，$|C_i|$ 表示某个簇中含有的特征的个数。最后，汇集从所有簇中选择的特征到 FS 中（Lines 8-9）。

2.2.2 实例迁移阶段

在实例迁移阶段，FeCTrA 方法使用 TrAdaBoost[41] 来完成，即从源项目数据中挑选实例以便有足够多的数据用于构建在目标项目上具有良好预测性能的模型。TrAdaBoost 尝试从源项目中选出与目标项目中已经标注实例分布相近的实例，从而完成实例的迁移。TrAdaBoost 是对 Adaboost 的一个改进，它是一个用于对源项目中的训练实例设置权重的框架，与目标项目中实例分布相近的实例会被赋予更高的权重，反之则更小，从而可以在源项目中找到与目标项目中实例分布相似的数据。

算法 2 中给出了实例迁移的详细描述。算法的输入是经过特征迁移过滤后的源项目数据 *FSource*、经过特征迁移过滤后的目标项目数据 *FTarget*、基分类器 *Learner* 以及最大迭代次数 *N*。算法的输出是含有多个分类器的集成预测模型。

在算法 2 中，主要借助 TrAdaBoost 完成实例的迁移。TrAdaBoost 需要使用目标项目中少量的标记实例，为了操作和表述方便，将经过特征过滤后的 *FSource* 记为 T_d，经过特征过滤后的 $FTarget_{labeled}$ 记为 T_s，经过特征过滤后的 $FTarget_{unlabeled}$ 为 S（Line 1）。其中，$|T_d|=n$ 并且 $|T_s|=m$。然后，为每一个已经标注的实例赋予初始权重（Line 2）。随后进入 TrAdaBoost 的循环阶段（Lines 3-10）。在循环内部，设当前迭代的轮次为 t，首先对 T_d 和 T_s 中实例的权重 w^t 做归一化操作（Line 4）。其次，在 T_d 和 T_s 数据上，调用分类器 *Learner* 构建一个分类假设 h_t（Line 5）。然后，用当前得到的分类假设去预测目标项目中已经标注的实例 T_s，并计算预测错误率（Lines 6-7）。当用得到的分类假设 h_t 去预测实例 T_s 时，如果实例被预测正确的话，这意味着这个实例和目标项目中的实例具有相同的分布，否则表明他们具有不同的分布。随后，在 TrAdaboost 中，根据错误率来更新调整训练实例权重的因子（Line 8）。最后，分别更新源项目数据和目标项目数据中已标注的实例的权重。这个方式可以减小分类错误的实例对训练模型的影响，从而可以构建一个具有良好性能的跨项目缺陷预测模型。重复此过程 *N* 次可以得到多个分类假设并进行集成。因此，该模型充分利用了所有基分类器的分类能力来对未知的实例进行预测（Line 11）。

算法 2：Instance Transfer Phase

Input:

$FSource^{n\times c}$: data from the source project filtered by feature transfer;

$FTarget_{labeled}^{m\times c}$: labeled data from the target project filtered by feature transfer;

$FTarget_{unlabeled}^{m'\times c}$: unlabeled-data from the target project filtered by feature transfer;

Learner : the base learning algorithm learner;

N : the maximum number of iterations;

Output:

1: Use T_d, T_s and S to represent *FSource*, $FTarget_{labeled}$ and $FTarget_{unlabeled}$, respectively;

2: Initial the weight for each labeled instances, that $w^1=(w_1^1,...,w_{n+m}^1)$;

3: **for** *t*=1, ···, *N* **do**

4: Set $p^t=w^t/(\sum_{i=1}^{n+m}w_i^t)$;

5: Call *Learner*, providing it with T_d+T_s and the distribution p^t over S;

6: Get back a hypothesis $h_t:X\to Y$ which can be predicted on the T_s;

7: Calculate the error of h_t on T_s: $\epsilon_t=\sum_{i=n+1}^{n+m}\frac{w_i^t\cdot|h_t(x_i)-c(x_i)|}{\sum_{i=n+1}^{n+m}w_i}$;

8: Set $\beta_t=\frac{\epsilon_t}{1-\epsilon_t}$ and $\beta=\frac{1}{1+\sqrt{2ln(n/N)}}$;

9: Update the weight for each instance: $w_i^{t+1} = \begin{cases} w_i^t \beta^{|h_t(x_i)-c(x_i)|}, & 1 \le i \le n \\ w_i^t \beta_t^{-|h_t(x_i)-c(x_i)|}, & n+1 \le i \le n+m \end{cases}$

10: **end for**

11: Output the hypothesis $h_f(x) = \begin{cases} 1, & \prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-h_t(x)} \ge \prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-\frac{1}{2}} \\ 0, & otherwise \end{cases}$

2.2.3 特征之间相关性的度量方法

特征相关性度量是用来度量两个特征 f_i 和 f_j 之间的关联性。定义 1 中的 $IFC(f_i, f_j)$ 是用于衡量两个特征之间的关联性。已有研究[44] 表明特征之间并不一定满足线性关系。因此，使用非线性的度量方法来衡量两个特征之间的关系更加合理。对称不确定性（Symmetric Uncertainty， 简称 SU）是非线性度量中的一种典型的方法[45]，论文在实验中使用 *SU* 来计算两个特征之间的关联性的，即 $IFC(f_i, f_j) = SU(f_i, f_j)$。值得注意的是，本文提出的框架适用于可以计算两个特征之间相关性的任何方法，因此具有可扩展性。

SU 借助信息论中的熵，通过计算特征 f_i 和 f_j 之间的分布差异性来衡量彼此之间的关联性。通过计算特征 f_i 和 f_j 的互信息然后进行归一化可得到 *SU* 。计算公式如下所示：

$$SU(f_i, f_j) = 2\left[\frac{IG(f_i \mid f_j)}{H(f_i) + H(f_j)}\right] \tag{1}$$

其中：

（1） $H(f_i)$ 表示特征的不确定度（即熵），其定义如下：

$$H(f_i) = -\sum p(f'_i) \times \log_2 p(f'_i) \tag{2}$$

$p(f_i')$ 表示特征 f_i 取一个特定值的先验概率。

（2） $IG(f_i \mid f_j)$ 表示信息增益率。表示在给定特征 f_j 的情况下，特征 f_i 减少的量。

其计算公式如下所示：

$$IG(f_i \mid f_j) = H(f_i) - H(f_i \mid f_j) = H(f_j) - H(f_j \mid f_i) \tag{3}$$

$H(f_i \mid f_j)$ 表示特征 f_j 确定的情况下，特征 f_i 的熵。其计算公式如下：

$$H(f_i \mid f_y) = -\sum_{f'_y \in f_y} p(f'_j) \sum_{f'_i \in f_i} p(f'_i \mid f'_y) \log_2 p(f'_i \mid f'_y) \tag{4}$$

$p(f_j')$ 表示 f_j 取一个特定值的先验概率。由上述公式可知，信息增益率具有对称性，即两个特征在公式中出现的顺序不会影响它们之间的信息增益率的计算结果。

2.2.4 特征之间分布相似性的度量方法

特征分布的相似性是用来度量一个具体的特征 f_i 在两个不同的数据集上的分布的相似性程度。定义 2 中的 $SFD(f_i)$ 用于衡量一个具体特征在源项目和目标项目数据集上的分布相似度。我们使用 $K-S$ (Kolmogorov-Smirnov) 检验来验证两组数据分布是否相似，这两组数据是指两个不同的数据集上相同特征对应的数据。$K-S$ 检验是一种非参检验，主要通过计算两组数据之间是否具有显著的差异来反应特征分布的相似度。在本方法中，我们将 $K-S$ 检验出来没有显著差异的情况定义为某特征在源项目和目标项目数据上分布相似程度。值得注意的是，本文提出的框架适用于可以计算一个具体特征在两组数据上分布相似性的任何方法，因此具有可扩展性。

$K-S$ 检验是一种拟合度检验，用来判断样本的实际分布值与指定理论的分布是否吻合。当然，$K-S$ 也可以用来检测两个样本分布是否具有显著差异，也就是说两样本的分布是否相同。该方法通过两个样本的累计频次数分布是否相当接近来判断原假设 H_0 是否为真。假如两个样本间的累计概率具有较大的分布差异，那么可以肯定两个样本取自不同的总体，则拒绝 H_0。

$K-S$ 检验首先提出两个假设：$H_0: S_1(x)=S_2(x)$，$H_1: S_1(x)\neq S_2(x)$。如果令 $S_1(x)$、$S_2(x)$ 分别表示第一和第二个样本观察值的累计概率分布函数，那么得到 $K-S$ 两个样本的双尾检测统计量为：$D=max|S_1(x)-S_2(x)|$。如果对于每一对样本值，$S_1(x)$ 和 $S_2(x)$ 都能十分接近的话，则表明两个分数的拟合程度很高，则有理由认为两个样本数据来自于相同的分布函数。这里，通常将显著性水平 p -value 设置为 0.05。因此, 当计算出来的值大于 0.05 时，则接受原假设 H_0，否则拒绝。

3 实验设计

本节将对 FeCTra 方法的有效性展开实证研究，首先提出研究问题，从不同角度来探讨 FeCTrA 方法在各类场景下的性能优劣。然后，介绍实证研究中使用的数据集、性能评测指标和显著性检验方法。最后，介绍实验流程及其方法参数设定。

3.1 研究问题

FeCTrA 方法的目的是在目标项目内没有足够多的训练数据前提下完成缺陷预测的任务。FeCTrA 方法从分布的差异性入手，结合特征迁移和实例迁移来构造出对目标项目有用的训练数据。一方面，特征迁移主要从源项目中选择与目标项目中分布相似的特征，从特征角度降低分布差异较大对数据集质量的影响；另一方面，实例迁移主要使用 TrAdaBoost 方法，尝试从源项目中选出与目标项目数据服从相似分布的实例，从实例角度降低分布差异较大对数据集质量的影响。为了验证 FeCTrA 方法的有效性，论文提出如下四个研究问题：

RQ1：FeCTrA 方法是否优于已有经典的跨项目缺陷预测方法？

目前研究人员针对跨项目缺陷预测问题已经提出了多种方法，我们重点考虑 TCA+[12]、Peters 过滤法[13]、Burak 过滤法[14] 和 Zimmermann 等人[15] 提出的方法。除此之外，我们还同时考虑了仅基于特征迁移的 FeCTrA 方法和仅基于实例迁移的 FeCTrA 方法。在 FeCTrA 方法中会存在很多影响因素，例如特征迁移阶段的特征选择比例、实例迁移阶段的目标项目中标记实例比例以及方法考虑的分类器。在该研究问题中，我们基于 RQ2 到 RQ4 的分析结果为这些影响因素设置最优取值。

RQ2：在 FeCTrA 方法的特征迁移阶段，特征选择比例对 FeCTrA 方法的性能影响如何？

在特征迁移阶段，FeCTrA 方法尝试通过聚类分析, 从源项目中迁移与目标项目分布相似的特征。因此在该 RQ 中，我们想分析不同的特征选择比例是否会对 FeCTrA 方法的性能产生影响。

RQ3：在 FeCTrA 方法的实例迁移阶段，目标项目中标记实例比例对 FeCTrA 方法的性能影响如何？

在实例迁移阶段，FeCTrA 方法需要使用少量目标项目中已经标注的实例。因此在该 RQ 中，我们想分析使用不同目标项目中的标记实例比例是否会对 FeCTrA 方法的性能产生影响。

RQ4：使用不同的分类器对 FeCTrA 方法的性能影响如何？

FeCTrA 方法内部需要提供分类器以完成预测模型的构建。在软件缺陷预测领域中，被广泛使用的分类器主要包括如下类型：基于概率的分类器，基于决策树的分类器，基于函数式的分类器以及基于集成学习的分类器等。不同类型的分类器在不同数据集上的预测性能并不相同。因此，我们想分析不同类型的分类器是否会对 FeCTrA 方法的性能产生影响。

3.2 评测对象

在我们的实证研究中，使用了在软件缺陷预测领域中被研究人员广泛使用的数据集（即 Relink 数据集和 AEEEM 数据集）[12, 46-48]。表 2 和表 3 列出了这两个数据集的统计特征。

Table 2 The Statistical characteristics of Relink datasets

表 2 Relink 数据集的统计特征

项目	项目类型	模块数	缺陷模块数(比例)	特征数	模块粒度
Apache	Web Server	194	98(50.52%)	26	文件
Safe	Security	56	22(39.29%)	26	文件
ZXing	Bar-code reader library	399	118(29.57%)	26	文件

Table 3 The Statistical characteristics of AEEEM datasets

表 3 AEEEM 数据集的统计特征

项目	项目类型	模块数	缺陷模块数(比例)	特征数	模块粒度
EQ	OSGI frameword	325	129(39.69%)	61	类
JDT	Development	997	206(20.66%)	61	类
LC	Text search engine library	399	64(9.26%)	61	类
ML	Task management	1862	245(13.16%)	61	类
PDE	Development	1492	209(14.01%)	61	类

Relink 数据集是由 Wu 等人[48] 搜集整理的，并且借助手工方式对数据集中的缺陷信息进行了确认。他们使用 Understand 工具[1]分析三个项目(例如: Apache、Safe 和 ZXing), 从源代码中抽取出重要的软件特征指标。Relink 数据集有 26 个复杂度特征，这些特征主要基于代码的复杂度和抽象语法树，总体可以分为两个大类：基于程序复杂度的特征和基于数量的特征。表 4 仅仅列举了这 26 个特征中的 7 个特征并对其含义进行描述。Understand 网站可以查询到每个特征的具体含义。

Table 4 The Description of Some Feactures in Relink Dataset

表 4 Relink 数据集中部分特征的描述

特征名称	描述
AvgCyclomatic	所有嵌套的函数方法的平均圈复杂度
AvgLine	所有嵌套的函数或方法的平均行数
CountLine	所有嵌套函数或方法的所有行的数量
CountStmt	语句数量
MaxCyclomatic	所有嵌套的函数或方法的最大圈复杂度
RaticCommentToCode	注释占所有代码的比例
SumCyclomatic	所有嵌套的函数或方法的圈复杂度之和

AEEEM 数据集是由 D’Ambros 等人搜集整理[49]。AEEEM 中的每一个项目都包含 61 个特征，其中 17 个属于与源代码相关的特征，5 个属于与之前预测相关的特征，5 个属于与代码变更熵相关的特征，17 个属于与源代码熵相关的特征以及 17 个属于与源代码衰退相关的特征。更具体地说，AEEEM 数据集包含线性衰减熵（LDHH）和权值衰退（WCHU）。LDHH 和 WCHU 已经被证实了对于缺陷预测是非常有用的。表 5 仅仅列出了 AEEEM 中的部分特征及其具体含义。

[1] https://scitools.com

Table 5 The Description of Some Feactures in AEEEM Dataset

表 5 AEEEM 数据集中部分特征的描述

类别	特征名称	描述
源代码指标	ck oo cho	对象之间的耦合度
	ck oo numberOfLinesOfCode	代码行数
之前预测的指标	numberOfBugsFoundUntil	所有已发现的缺陷行数
	numberOfCriticalBugsFoundUntil	严重度是重要的缺陷数量
变更熵指标	CvsLogEntropy	CvsEntropy 的对数衰减
源代码熵指标	LDHH cbo	ck oo cbo 的线性衰减
	LDHH numberOfLinesOfCode	ck oo numberOfLinesOfCode 的线性衰减
源代码衰退指标	WCHU cbo	ck oo cbo 的加权衰退
	WCHU numberOfLinesOfCode	ck oo numberOfLinesOfCode 的加权衰退

本文仅考虑同构类型的跨项目缺陷预测问题，即源项目和目标项目考虑了相同的特征集合。因此 FeCTrA 方法仅在同一个数据集内部的项目间进行跨项目预测研究。例如, 在 Relink 数据集中，FeCTrA 方法可以使用 Apache 作为源项目，使用 Safe 或者 ZXing 作为目标项目，即 $Apache \to Safe$ 或者 $Apache \to ZXing$。但是，对于 Relink 数据集中任一项目做源项目，AEEEM 数据集中任意一项目的做目标项目这种情况， FeCTrA 方法无法处理。例如，以 Apache 为例， $Apache \to \{EQ, JDT, LC, ML, PDE\}$ 在 FeCTrA 方法中是不支持的。

3.3 评测指标

对于目标项目中任一实例经缺陷预测模型后会有四种可能的输出结果：当一个含有缺陷的实例被预测为有缺陷实例，记为 TP（True Positive）；当一个不含有缺陷的实例被预测为有缺陷实例，记为 FP（False Positive）；当一个含有缺陷的实例被预测为无缺陷的实例，记为 FN（False Negative）；当一个不含有缺陷的实例被预测为无缺陷的实例，记为 TN（True Negative）。基于以上这些可能的输出结果，可以定义查准率(Precision)、查全率(Recall)以及 F1 度量(F1-measure)。

查准率：在所有被预测为有缺陷的实例中，真正含有缺陷的实例所占的比例，

$$Precision = TP / (TP + FP) \tag{5}$$

查全率：在所有真正含有缺陷的实例中，被正确预测为有预测的实例所占的比例，

$$Recall = TP / (TP + FN) \tag{6}$$

F1 度量又称为 F1-Score，是综合考虑查准率和查全率两个指标的指标，其定义如下：

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{7}$$

通常，在查准率和查全率之间有一个折衷，一般来说，查准率高时，查全率往往偏低；而查全率高时，查准率往往偏低。然而，折衷的方案很难与仅使用查准率或者查全率作为评价指标的预测模型进行比较[50]。而 F1 是通过查准率和查全率的调和平均数计算而来，其综合考量各方法在查全率和查准率上的整体性能表现，可以全面的反应方法实际性能的优劣。鉴于此，论文使用 F1 评价各方法的性能。

3.4 显著性检验方法

为了检验不同方法之间的性能差异是否显著，论文考虑了被广泛使用的 Wilcoxon 符号秩检验[51]。该检验是无参的统计假设检验，也是 t-检验的替代方案，其核心是忽略数据的符号，对两个不同分类器在每一个数据集上取得的性能结果之间的差异进行排序，同时比较正向差异和反向差异的排名。

假设，两个分类器在特定的实验方案下产生了 N 组实验数据。以 d_i 表示这两个分类器在第 i 组数据上的性能差异。这 N 组差异将会根据差异的绝对值进行排序。如果两者无差异，即 $d_i = 0$ 则将它们两者差异排名的平均值作为它们各自的排名。使用 R^+ 表明第二个算法优于第一个算法对应数据的排名总和，那么 R^- 表示第二个算法劣于第一个算法对应数据的排名总和。当 $d_i = 0$ 时，则取所有 d_i 为零的数据对应的排名之和的一半。如果他们的个数是奇数，那么其中一个数据将会被忽略掉。 R^+， R^- 的定义如下：

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \quad (8)$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \quad (9)$$

记 T 为 R^+、R^- 这两者的最小值，即 $T = min(R^+, R^-)$。对于数据量大于 25 数据集，下面的统计公式表示分布是接近于正态分布的。

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \quad (10)$$

当给定的显著因子 $\alpha = 0.05$ 时，如果 $z < -1.96$，则空假设将被拒绝，即表明被比较的两个分类器的性能是存在显著差异的。关于 Wilcoxon 的更加详细的描述可参考[51, 52]。

为了进一步比较两个方法之间的性能，我们使用“Win/Draw/Loss”分析。在特定评价指标上，“方法 1 vs 方法 2” 的“Win/Draw/Loss”结果分析共有三种情况：即“方法 1” 的性能显著好于、相似或者显著差于“方法 2”的性能的次数。

3.5 实验流程及其方法参数设定

为了评价 FeCTrA 方法在跨项目缺陷预测中的性能，我们比较了六种经典的跨项目软件缺陷预测方法：（1）只包含特征迁移阶段的 FeCTrA 方法，记为 FeCTrA(FT)；（2）只包含实例迁移阶段的 FeCTrA 方法，记为 FeCTrA(IT)；（3）Nam 等人[12]提出的 TCA+方法；（4）Peters 等人[13]提出的 Peters 过滤法；（5）Turhan 等人[14] 提出的 Burak 过滤法；以及（6）Zimmermann 等人[15] 提出的方法, 为了后续描述的方便，将其记为 DCPDP(Directly Cross-project Defect Prediction) 方法。

FeCTrA 方法主要包括特征迁移和实例迁移两个阶段。在特征迁移阶段中，需要对特征进行聚类。而簇的个数和选择的特征个数会对预测性能产生重要影响。已有研究工作[53, 54] 表明：比较理想的簇的个数为 $\lceil log_2 M / 2 \rceil$，其中 M 表示原始特征的个数。缺省情况下，迁移的特征个数为 40%×M。在实例迁移阶段，需要目标项目中提供部分已标记的数据。缺省情况下，FeCTrA 方法从目标项目中选出 10%的实例作为已经标注的实例。除此之外，TrAdaboost 是一个不断迭代的过程，研究结果表明[41]，100 次迭代可以使得模型的性能收敛。因此, FeCTrA 方法将实例迁移阶段的迭代次数设置为 100。为了减少随机性对实验结果造成的影响，论文重复执行 FeCTrA 方法 100 次并取均值作为最终结果。此外，当计算两个特征的分布相似度时，论文使用 R 语言提供的 K-S 检验。

TCA+方法是对 TCA 方法的一个扩展，主要包含两个阶段：正规化方法自动选择阶段和 TCA 应用阶段。DCPDP 方法直接在源项目上构建缺陷预测模型，然后在目标项目上进行预测。Peters 过滤法和 Burak 过滤法都是经典的基于实例迁移的跨项目缺陷预测方法。此外，在分类器的选择上，本文使用软件缺陷预测领域中被广泛使用的 Naive Bayes[7, 53, 55] 作为默认分类器。FeCTrA 方法和所有基准方法均基于 weka 软件包编程实现。

在跨项目缺陷预测的实验中，采取“一对一”的方式，即每次只选择一个项目作为源项目，选择另一个项目作为目标项目。以 Relink 数据集中的 Apache 项目为例，其对应的源项目可以是 Safe，或者是 ZXing, 但并不允许是两个项目的融合。此外，对于 FeCTrA 方法，需要同时使用源项目数据和目标项目中少量已经标注的数据作为训练集。本文采用反转的十折交叉检验处理方式。对于某一折划分，采用如下的处理方法：使用源项目的所有标记数据和目标项目中一折（即 10%）已标记数据作为最终的训练集，然后在此数据集上构建一个跨项目的缺陷预测模型，最后对目标项目中未标记的九折（即 90%）数据进行预测。依次与目标项目中每一折数据结合构建训练集，预测剩下的九折，从而得到十次结果。重复该过程 10 次，即可得到 100 次实验结果。此外，为了确保所有项目的在相同的测试集上进行性能的公平评估，对于其他基准方法采取了相同

的反转十折处理方法。但是不同于 FeCTrA 方法，基准方法构建的训练集并不包含目标项目中少量的标记数据（缺省 10%），直接在把源项目数据作为其总体训练集，然后预测目标项目中的未标记数据（缺省 90%）。

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

4 实证研究结果的分析

4.1 针对RQ1 的结果分析

论文在 Relink 数据集和 AEEEM 数据集上分别对 FeCTrA 方法的整体性能进行了实证研究，使用 Naïve Bayes 作为基分类器，迁移的特征比例为 40%，选择目标项目中标注的实例比例为 10%，并将 FeCTrA 方法与六种基准方法进行了比较。表 6 和表 7 分别给出了这些方法在两个数据集上取得的 F1 均值。

在表 6 和表 7 中，第一列表示跨项目缺陷预测的具体场景，比如在表 7 中，*JDT* ⇒ *EQ* 表示使用 JDT 作为源项目，EQ 作为目标项目。接下来的三列表示论文提出的 FeCTrA 方法,由于 FeCTrA 方法包含两个阶段，因此我们使用 FeCTrA、FeCTrA(FT) 和 FeCTrA(IT)分别表示同时使用特征迁移和实例迁移、仅使用特征迁移和仅使用实例迁移。最后四列表示跨项目缺陷预测研究中的已有经典基准方法，即 TCA+, Peters 过滤法, Burak 过滤法和 DCPDP。两个表格的最后一行给出了每一种方法的整体平均性能。表格中每一行的最大值进行加粗表示。

从两个表格的最后一行可以看出，基于 Relink 和 AEEEM 两个数据集，与基准方法相比较，论文提出的 FeCTrA 方法能够取得更好的预测性能。

与 FeCTrA(FT)方法和 FeCTrA(IT)方法相比，FeCTrA 方法在绝大部分的跨项目缺陷预测场景中都能取得更好的预测性能。例如，对于 Relink 数据集上的 Safe ⇒ Apache，FeCTrA 方法取得的 F1 均值为 0.672，而 FeCTrA(FT) 方法和 FeCTrA(IT)方法分别获得了 0.646 和 0.583，因此 FeCTrA 方法与这两种方法相比，其性能提升分别为 4% 和 15.3%。对于 AEEEM 数据集上的 LC ⇒ EQ，FeCTrA 方法取得的 F1 均值为 0.724，相对于 FeCTrA(FT)方法(0.532) 和 FeCTrA(IT) 方法(0.593)，其性能提升分别为 36.1%和 22.1%。实验结果表明，在跨项目缺陷预测中，将特征迁移和实例迁移进行结合，与仅考虑单个阶段的方法相比，其能够获得更好的预测性能。

Table 6 Comparison of F1 Among FeCTrA and Six Baseline Methods on Relink

表 6 基于 F1 评价指标，FeCTrA 与六种基准方法在 Relink 数据集上的平均性能比较

Source⇒Target	FeCTra	FeCTra(FT)	FeCTra(IT)	TCA+	Peters 过滤法	Burak 过滤法	DCPDP
Safe⇒Apache	0.672	0.646	0.583	0.545	**0.685**	0.633	0.167
Zxing⇒Apache	0.654	**0.662**	0.655	0.162	0.364	0.331	0.241
Apache⇒Safe	0.568	0.536	0.536	0.727	0.731	**0.740**	0.731
Zxing⇒Safe	**0.695**	0.692	0.562	0.436	0.401	0.395	0.277
Apache⇒Zxing	0.581	0.580	0.591	0.598	0.690	0.703	**0.709**
Safe⇒Zxing	0.654	0.661	0.655	0.642	**0.693**	0.677	0.638
Avg	**0.637**	0.630	0.597	0.518	0.594	0.580	0.461

Table 7 Comparison of F1 Among FeCTrA and Six Baseline Methods on AEEEM

表 7 基于 F1 评价指标，FeCTrA 与六种基准方法在 AEEEM 数据集上的平均性能比较

Source⇒Target	FeCTra	FeCTra(FT)	FeCTra(IT)	TCA+	Peters 过滤法	Burak 过滤法	DCPDP

JDT⇒EQ	**0.709**	0.509	**0.709**	0.424	0.419	0.414	0.424
LC⇒EQ	**0.724**	0.532	0.593	0.269	0.271	0.272	0.279
ML⇒EQ	**0.720**	0.570	0.604	0.285	0.217	0.214	0.219
PDE⇒EQ	**0.689**	0.549	0.590	0.289	0.334	0.319	0.322
EQ⇒JDT	**0.805**	0.686	0.692	0.667	0.523	0.509	0.429
LC⇒JDT	**0.788**	0.723	0.691	0.336	0.301	0.375	0.402
ML⇒JDT	**0.780**	0.726	0.673	0.332	0.322	0.360	0.313
PDE⇒JDT	**0.792**	0.711	0.684	0.312	0.382	0.356	0.343
EQ⇒LC	**0.815**	0.723	0.724	0.693	0.462	0.487	0.460
JDT⇒LC	**0.854**	0.755	0.733	0.423	0.455	0.432	0.426
ML⇒LC	**0.798**	0.764	0.681	0.286	0.271	0.220	0.226
PDE⇒LC	**0.842**	0.757	0.717	0.289	0.386	0.384	0.368
EQ⇒ML	**0.800**	0.714	0.700	0.683	0.528	0.511	0.513
JDT⇒ML	**0.831**	0.729	0.711	0.400	0.513	0.589	0.570
LC⇒ML	0.798	**0.833**	0.551	0.266	0.249	0.390	0.405
PDE⇒ML	0.826	0.724	**0.828**	0.289	0.391	0.395	0.348
EQ⇒PDE	**0.734**	0.700	0.619	0.657	0.557	0.539	0.484
JDT⇒PDE	**0.817**	0.721	0.700	0.470	0.450	0.450	0.465
LC⇒PDE	**0.801**	0.718	0.682	0.316	0.289	0.287	0.338
ML⇒PDE	**0.805**	0.727	0.684	0.318	0.217	0.223	0.223
Avg	**0.786**	0.693	0.678	0.400	0.377	0.386	0.378

与 TCA+方法相比，在两个数据集上，FeCTrA 方法的性能要好于 TCA+方法。TCA+方法借助特征映射完成特征迁移，这与 FeCTrA 方法的第一阶段比较相似，即与 FeCTrA(FT)方法相类似。从实验结果可以看出，绝大部分情况下，论文提出的特征迁移方法 FeCTrA(FT) 要好于 TCA+方法。例如，在 Relink 和 AEEEM 数据集上，FeCTrA(FT)方法分别获得了 0.630 和 0.693 的性能，而 TCA+方法仅获得了 0.518 和 0.400。这充分体现了在跨项目缺陷预测中特征迁移阶段的重要性，有利于排除无关特征对实验结果产生的影响。此外，相对于 TCA+方法，FeCTrA(FT)方法的性能更为稳定。例如，在 Apapche ⇒Safe 场景中，TCA+方法能够获得令人满意的性能（即 0.727), 而在 LC ⇒ ML 场景中，TCA+方法则难以获得令人满意的性能（仅 0.266），因此在不同的场景下，TCA+方法的性能波动较大。

与 Peters 过滤法和 Burak 过滤法相比，在 AEEEM 数据集上，FeCTrA 方法在所有的跨项目缺陷预测场景中都取得了最好的性能。而在 Relink 数据集上，Peters 过滤法和 Burak 过滤法在部分场景下表现较好，如 Safe ⇒ Apache，Apache ⇒ Safe 和 Safe ⇒ ZXing 这三个场景上。其可能原因如下：在 Relink 数据集上，各个项目的实例普遍偏少，而 FeCTrA 方法仅仅借助了目标项目中 10% 的实例，因此可用的信息较少，而 Peters 过滤法和 Burak 过滤法选出的实例较多，因此包含的信息也更多。但从整理来说，FeCTrA 方法相对于 Peters 过滤法和 Burak 过滤法，其性能分别提高了 7.2% 和 9.8%。

与 DCPDP 方法相比，FeCTrA 方法在两个数据集上也几乎取得更好的预测性能。总体而言，FeCTrA 方法在 Relink 数据集上，其性能提升了 38.2%，在 AEEEM 数据集上，其获得的性能是 DCPDP 方法的两倍,这些结果表明在跨项目缺陷预测中，直接使用源项目数据中的所有特征和实例并不能保证可以得到更好的预测效果，而移除冗余特征、无关特征以及分布不相似的实例会显著提升模型的性能。然而，在 Apache ⇒ ZXing 场景下，DCPDP 能够获得更好的预测结果，这可能是因为 Apache 和 ZXing 两个项目本身分布较为相似，所以 DCPDP 方法能够取得更好的性能。

Table 8 Win/Draw/Loss of FeCTrA Compared with Six Baselines on Both Datasets
表 8 FeCTrA 方法与六种基准方法在两个数据集上的 WIN/DRAW/LOSS 比较结果

Target	Against(Win/Draw/Loss)					
	FeCTra(FT)	FeCTra(IT)	TCA+	Peters 过滤法	Burak 过滤法	DCPDP
Apache	1/0/1	1/0/1	2/0/0	1/0/1	2/0/0	2/0/0
Safe	2/0/0	2/0/0	1/0/1	1/0/1	1/0/1	1/0/1
Zxing	0/0/2	1/0/1	1/0/1	0/0/2	0/0/2	1/0/1
Total	**3/0/3**	**4/0/2**	**4/0/2**	**2/0/4**	**3/0/3**	**4/0/2**
EQ	3/1/0	4/0/0	4/0/0	4/0/0	4/0/0	4/0/0
JDT	4/0/0	4/0/0	4/0/0	4/0/0	4/0/0	4/0/0
LC	4/0/0	4/0/0	4/0/0	4/0/0	4/0/0	4/0/0
ML	3/0/1	3/0/1	4/0/0	4/0/0	4/0/0	4/0/0
PDE	4/0/0	4/0/0	4/0/0	4/0/0	4/0/0	4/0/0
Total	**18/1/1**	**19/0/1**	**20/0/0**	**20/0/0**	**20/0/0**	**20/0/0**

表 8 列出了 FeCTrA 方法与六种基准方法之间的 Win/Draw/Loss 比较结果。表格分为上下两个部分，分别表示 Relink 数据集中的项目和 AEEEM 数据集中的项目。表中的每一行表示，以当前项目作为目标项目，其他的项目为源项目。例如以 EQ 为例，则表示将 EQ 设置为目标项目，由剩下项目（即 JDT, LC,ML 和 PDE）中可以选一个作为源项目。因为 AEEEM 数据集总共含有 5 个项目，因此会总共产生 20 个跨项目缺陷预测场景。从表 8 中不难看出，在 Relink 数据集上，FeCTrA 方法最低可以取得 33.3%（2/6）的胜算, 即与 Peters 过滤法比较; 在 AEEEM 数据集上，FeCTrA 方法最低可以取得 90%（18/20）的胜算。在大部分数据集上, FeCTrA 方法优于仅考虑特征迁移或仅考虑实例迁移的方法。

Table 9 p-Value of the Wilconxon Signed-Rank Test Among Baseline Methods and FeCTrA
表 9 FeCTrA 和基准方法间的显著性检验结果

Dataset	FeCTra vs FeCTra(FT)	FeCTra vs FeCTra(IT)	FeCTra vs TCA+	FeCTra vs Peters 过滤法	FeCTra vs Burak 过滤法	FeCTra vs DCPDP
Relink	4.66E-03	1.18E-04	3.41E-04	1.25E-01	9.44E-03	3.50E-02
AEEEM	2.20E-16	2.20E-16	2.20E-16	2.20E-16	2.20E-16	2.20E-16
Both datasets	2.20E-16	2.20E-16	2.20E-16	2.20E-16	2.20E-16	3.76E-16

此外,为了验证 FeCTrA 方法与基准方法间的性能差异是否具有显著性,论文对实验结果进行了 Wilcoxon 符号秩检验，并设置显著性水平 α 为 0.05，具体结果如表 9 所示。基于表 9，可以发现：“FeCTrAvs FeCTrA(FT)”、“FeCTrA vs FeCTrA(IT)”、“FeCTrA vs TCA+”、“FeCTrA vs Peters 过滤法”、“FeCTrA vs Burak 过滤法”和“FeCTrA vs DCPDP” 的 p 值都小于 0.05，这表明：基于显著性分析，FeCTrA 方法的预测性能要显著优于其他六种基准方法。

基于上述分析，在跨项目缺陷预测中，冗余特征、无关特征以及分布不同的实例均会影响跨项目缺陷预测模型的性能，而论文提出的 FeCTrA 方法通过同时考虑特征迁移和实例迁移，可以取的更好的预测性能。

4.2 针对RQ2 的结果分析

为了分析特征迁移阶段中特征选择比例对 FeCTrA 方法性能的影响，我们将特征选择比例从 10%逐步增长到 100%，步长设置为 10%。图 2 和图 3 分别显示了基于 Relink 数据集和 AEEEM 数据集上，特征选择比

例对 FeCTrA 方法性能的影响。

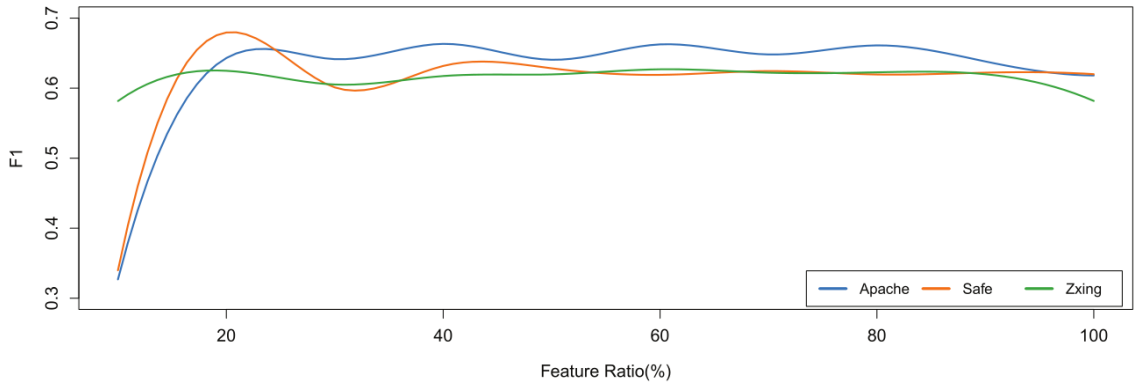

Fig.2 The Impact on FeCTra by Varying Feature Transfer Ratio on Relink

图 2 在 Relink 数据集上，迁移不同比例的特征对 FeCTrA 性能的影响

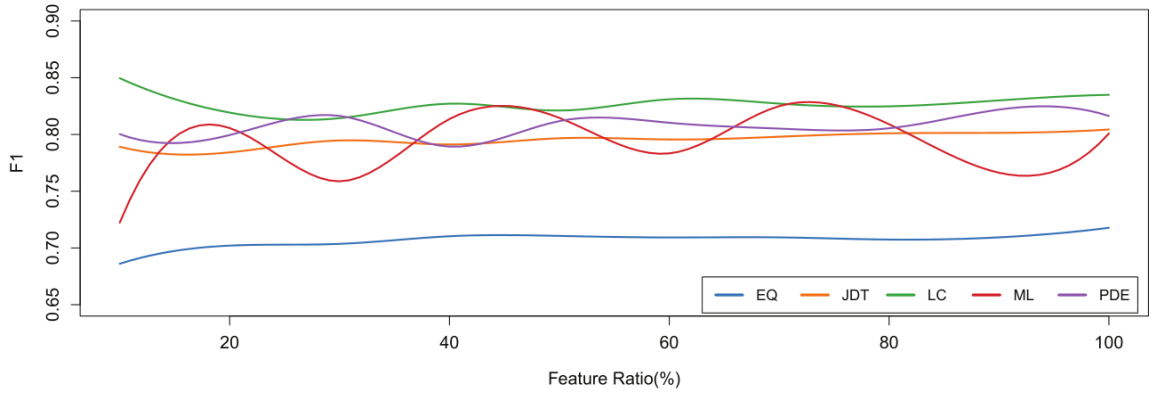

Fig.3 The Impact on FeCTra by Varying Feature Transfer Ratio on AEEEM

图 3 在 AEEEM 数据集上，迁移不同比例的特征对 FeCTrA 性能的影响

在图 2 和图 3 中，x 轴表示从源项目中迁移的特征数量占所有特征数量的比例，其取值从 10%逐步增长到 100%，y 轴表示 FeCTrA 方法基于该特征选择比例取得的 F1 均值。我们对数据折线做了平滑处理。图中，每一条曲线表示一个具体的项目，为了便于区分，使用了不同颜色对曲线进行绘制。

从图 2 中可以发现，当特征选择比例从 10% 增长到 20% 时，FeCTrA 在各个项目上取得的预测性能在不断提高。但是当特征选择比例从 20%增加到 30%的时候，FeCTrA 方法在各个项目上的预测性能都出现了不同程度的下降，其中在 Safe 项目上，性能下降最为严重。这可能是由于选出的这些特征在目标项目上并不起到重要作用，而这些特征被挑选中可能仅仅是因为在数据分布上与源项目更接近而已。当特征选择比例从 30% 增加到 40%的时候，FeCTrA 方法在各个项目上的预测性能又开始提升。随后，当特征选择比例不断提高时，FeCTrA 方法的预测性能并没有持续提高，而是趋于稳定。甚至当选择全部特征的时候，FeCTrA 方法的预测性能反而出现下降。这说明:（1）从源项目中迁移所有的特征，并不能保证在目标项目上具有很好的泛化能力，这可能是冗余特征和无关特征的存在所引起。因此，在迁移特征时，识别并移除上述两类特征

很有必要。（2）迁移 40%的特征能够使得 FeCTrA 方法在 Relink 数据集上可以达到较高的预测性能。

从图 3 中可以发现，随着特征选择比例的变化，FeCTrA 在各个项目上的性能表现也在不断变化。例如，在 EQ 和 JDT 项目上，FeCTrA 方法的预测性能随着特征选择比例的增加而不断提高，但是提高的幅度不大。这说明，在这两个项目上，当迁移的特征比例为 20%~40% 的时候，FeCTrA 方法已经能够挑选出最重要的特征。在 LC 项目上，当迁移的特征比例为 10%的时候，FeCTrA 方法取得的预测性能最高，随着迁移特征比例的增加，FeCTrA 方法开始出现下降，直至迁移的特征比例为 40% 的时候，FeCTrA 方法的性能才趋于稳定。而对于 ML 和 PDE 项目，随着迁移的特征比例不断增加，FeCTrA 方法的性能出现波动现象。具体来说，在 PDE 项目上，FeCTrA 方法在迁移的特征比例为 30%的时候首次达到了最优效果，而在迁移的特征比例为 40%以后，其最好性能与最差性能的波动情况趋于稳定。在 ML 项目上，当迁移的特征比例大于 20%的时候，FeCTrA 方法的最好性能与最差性能几乎保持不变，并且在迁移的特征比例为 40%的时候，首次达到最优性能。因此，在 AEEEM 项目上，迁移的特征比例设置为 40%是理想的选择。

基于上述分析，在 FeCTrA 方法的特征迁移阶段，从源项目中迁移 40%的特征比较理想。

4.3 针对RQ3 的结果分析

为了分析目标项目中标注实例比例对 FeCTrA 方法预测性能的影响，论文主要假设目标项目中存在 5%、10% 和 20%的标注实例。选择以上三种不同的标注实例比例主要有以下两个原因：（1）标注实例是一个耗时耗力、成本高昂并且容易出错的工作，但是使用有限的成本去标注少部分的实例是切实可行的，这也是论文研究 FeCTrA 方法的前提。但在目标项目中，标注的实例不宜过多，论文在实验中将标注实例的比例上限设置为 20%；（2）论文在模型性能评估时基于交叉验证的方式，因此选择以上三种标注实例的比例可以保证更好的进行交叉检验（即 20 折交叉验证、10 折交叉验证以及 5 折交叉验证）。图 4 和图 5 展示了不同的标注实例比例对 FeCTrA 方法预测性能的影响。

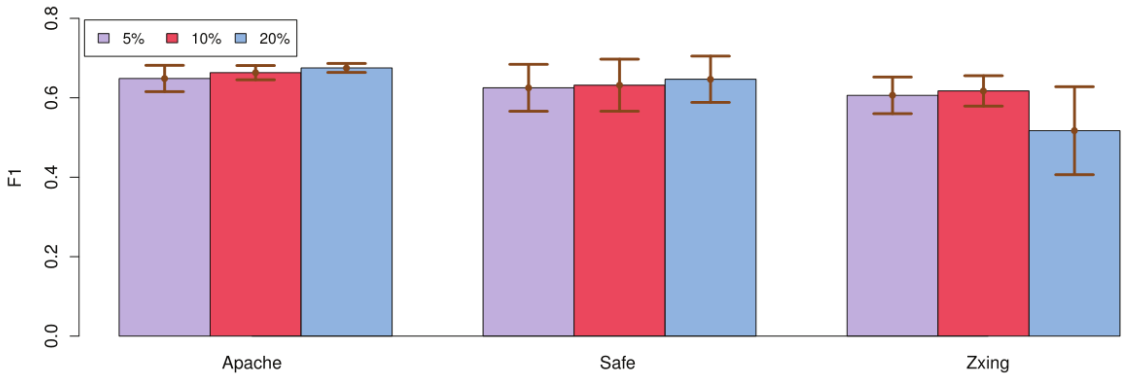

Fig.4 The Impact on FeCTrA by Varying Labeled Instance Ratio in Target Project on Relink

图 4 在 Relink 数据集上，目标项目中不同标注实例比例对 FeCTrA 方法预测性能的影响

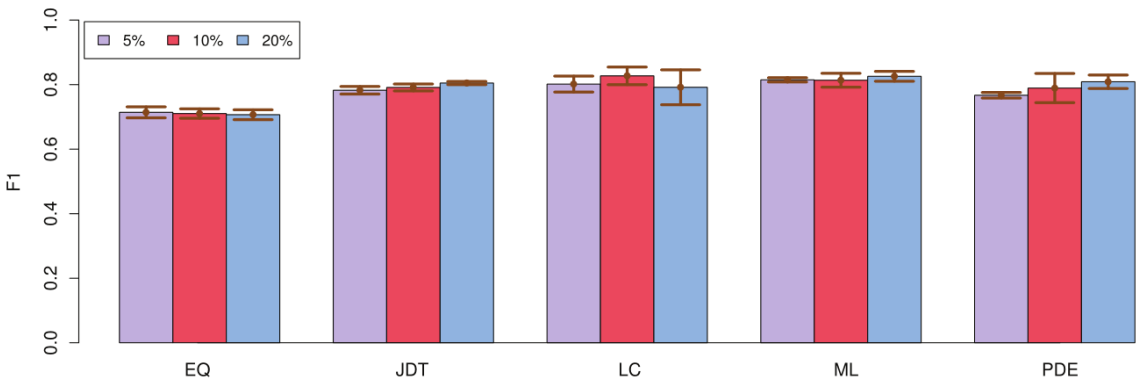

Fig.5 The Impact on FeCTrA by Varying Labeled Instance Ratio in Target Project on AEEEM

图 5 在 AEEEM 数据集上，目标项目中不同标注实例比例对 FeCTrA 方法预测性能的影响

在图 4 和图 5 中，横轴表示不同的项目，纵轴表示以某一项目为目标项目时，所有跨项目缺陷预测场景下得到的F1性能均值和标准差，为了便于区分，使用不同颜色表示目标项目中不同的标注实例比例。以 Apache 项目为例，当选择目标项目中 5%的实例作为已标注实例时，Apache 可以被等分成 20 份，即可以执行 20 折交叉验证。当选择目标项目中 20% 的实例作为已标注实例时，Apache 可以被等分成 5 份，即可以执行 5 折交叉验证。因此，每一个实验结果是基于 $\frac{M \times 10}{P}$ 个数据所获得。其中，P 表示目标项目中标注实例的比例（例如 P=5%），M 表示可以作为源项目的个数（例如当 Apache 为目标项目时，M=2），最后的 10 表示该交叉检验会重复执行 10 次。

从图 4 中可以看出，在 Apache 和 Safe 项目上，随着目标项目中标记数据的增加，FeCTrA 方法性能的均值在不断提高, 然后增加的幅度并不是很大。其原因可能是：在 Relink 数据集上，各个项目内含有的实例数目普遍较少。例如，Apache 项目仅有 194 个实例，Safe 项目仅有 56 个实例，ZXing 项目仅有 399 个实例。因此增加 5%~10%的实例比例并不会增加太多的标注信息。所以，FeCTrA 在各个项目的性能表现相对稳定。而在 ZXing 项目上，当标注实例的比例是 10% 时，FeCTrA 方法获得了最好的预测性能, 当比例增加到 20%时，性能反而有所下降。从图 5 中可以看出，在 JDT，ML 和 PDE 项目上，随着目标项目中标注实例的增加，FeCTrA 方法的性能也逐渐提高。这主要是因为 JDT, LC 和 ML 这三个项目里含有的实例数较多。例如，JDT 项目含有 997 个实例，ML 项目含有 1862 个实例, PDE 项目含有 1497 实例。因此随着标注实例比例的增加，可以被 FeCTrA 方法利用的实例信息就越多，性能自然越来越高。而 EQ 项目中含有的实例数较少，因此性能几乎保持不变。在 LC 项目中，当标注实例的比例为 10%时，FeCTrA 方法获得了最好的性能, 当比例增加到 20% 时，性能也有所下降。其原因一方面是数据集本身含有的实例较少（仅 399 个实例），另一方面可能是由于数据集本身质量不高所导致的。

基于上述分析，在 FeCTrA 方法的实例迁移阶段，从目标项目中的选择 10% 的标注实例比较理想。

4.4 针对RQ4 的结果分析

为了研究不同类型的分类器对 FeCTrA 方法的影响，论文考虑了软件缺陷预测研究经常使用的分类器。其中 J48 属于基于决策树的分类器，LR（Logistic Regression）和 SVM（Support Vector Machine）属于基于函数式的分类器，NB（Naive Bayes）属于基于概率的分类器，RF（Random Forest）属于基于集成学习的分类器。图 6 和图 7 显示了不同分类器对 FeCTrA 方法的影响。

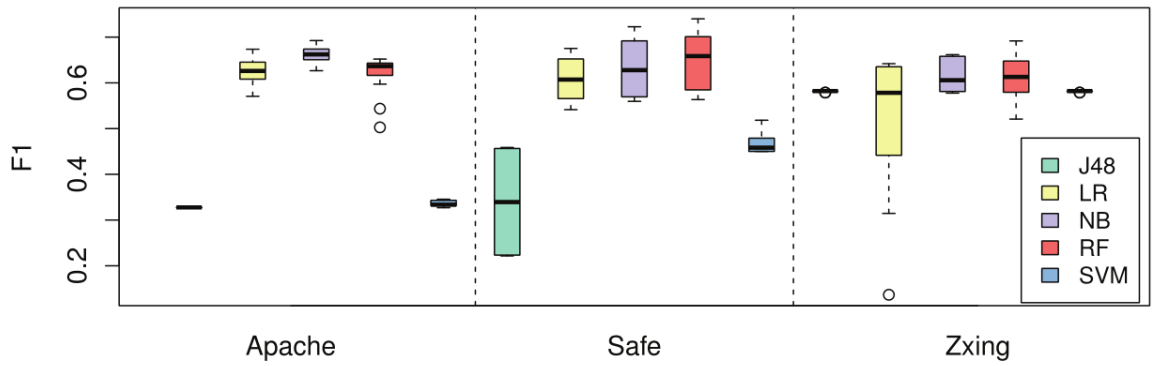

Fig.6 The Impact on FeCTrA by Using Different Basic Classifier on Relink

图 6 在 Relink 数据集上，不同分类器对 FeCTrA 方法性能的影响

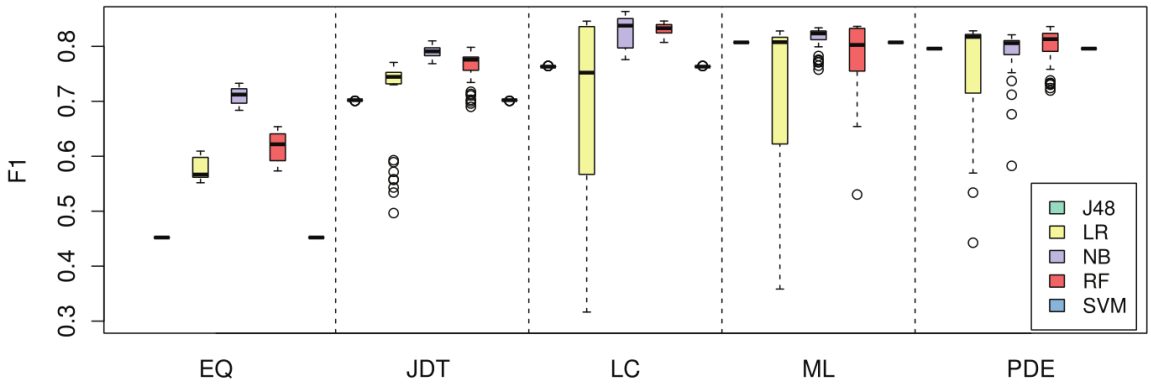

Fig.7 The Impact on FeCTrA by Using Different Basic Classifier on AEEEM

图 7 在 AEEEM 数据集上，不同分类器对 FeCTrA 方法性能的影响

在图 6 和图 7 中，横轴表示数据集中不同项目的名称，纵轴表示 FeCTrA 方法使用不同分类器后得到的 F1 值。为了便于区分，论文使用不同的颜色填充盒图以表示不同的分类器。

从图 6 中可以看出，在 Relink 数据集中的三个项目上，J48 和 SVM 分类器的预测性能较差，在 Apapche 和 Safe 项目上，使用 J48 作为分类器得到的 F1 都小于 0.5。而使用 LR、NB 和 RF 作为分类器得到的 F1 值相对较好，其中 NB 分类可以取的最好的 F1 值。例如，在 Apache 和 Safe 项目上，使用 NB 作为分类器得到的性能最高。从图 7 中也可以看出，除了 EQ 项目，使用不同的分类器在不同的项目上得到的性能都相对较好，且性能也比较稳定。例如，除了 LR 在 LC、ML 和 PDE 上表现波动较大，其他分类器在各个项目上表现都比较稳定。此外，NB 在 EQ, JDT，LC 和 ML 中表现最好，其次 RF 也能获得比较好的结果。

基于上述分析，不同类型的分类器对 FeCTrA 方法的性能会造成一定的影响，其中 NB 分类器整体性能表现更好。

4.5 有效性影响因素分析

这一节主要分析可能影响到论文实证研究结论有效性的影响因素。具体来说：（1）内部有效性主要涉及到可能影响到实验结果正确性的内部因素。最主要的有效性影响因素是实验代码的实现是否正确，为了减少重新实现各种基准方法过程中引入的人为因素的影响，我们使用了第三方提供的成熟框架，例如来自 Matlab 和 Weka 中的机器学习包。此外，我们采用了跨项目缺陷预测开源工具 CrossPare[56]提供的代码，该工具已经实现了当前跨项目缺陷预测领域的一些经典方法。（2）外部有效性主要涉及到实验研究得到的结论是否具有一般性。为了确保实证研究结论的一般性，我们选择了软件缺陷预测问题研究中经常使用的 Relink 数据集和 AEEEM 数据集，这两个数据集累计包含了 8 个具有一定代表性的开源项目，同时这些项目也覆盖了不同类型的应用领域，可以确保研究结论具有一定的代表性。（3）结论有效性主要涉及到使用的评测指标是否合理。论文重点考虑了 F1 指标，该指标是 Precision 和 Recall 指标的综合衡量，在软件缺陷预测领域被广泛使用[6, 7, 57, 58]，因此可以更好的评估模型的综合性能。

5 总结与展望

论文提出一种新颖的基于特征迁移和实例迁移的跨项目软件缺陷预测方法 FeCTrA。该方法主要包含特征迁移和实例迁移两个阶段。在特征迁移阶段，基于特征之间的关联性，将已有特征进行聚类分析，随后基于特征在源项目和目标项目之间的分布相似性，将每个簇中的特征从高到低进行排序，并选出指定数量的特征，从而可以有效地移除无关特征和冗余特征。在实例迁移阶段，使用 TrAdaboost 技术，依据目标项目中少量的已标注实例，从源项目中挑选出大量与目标项目分布相同的实例构建训练集，从而可以有效的缩小源项目和目标项目之间的分布差异。此外，本文基于 Relink 和 AEEEM 数据集对该方法展开了实证研究并验证了该方法的有效性。

论文仍存在很多值得探讨的下一步工作。首先, FeCTrA 方法在进行特征迁移的时候采用了聚类分析方式。在初始簇中心挑选不理想的情况下, 可能需要花费很长的时间才能达到簇中心的收敛。论文在该阶段挑选了在两个数据集中分布最相似的前几个特征作为初始簇中心。后续的研究需要分析考虑不同初始簇中心的选择对整个方法性能的影响。其次，论文在特征迁移阶段仅考虑了迁移特征的比例对方法性能的影响，下一步工作可以从特征的类别角度出发考虑特征迁移，即迁移何种类别的特征最有效。最后， 需要将论文方法应用到实际的软件测试过程中。部分研究工作[59, 60]表明，有超过 90%的开发人员愿意采用缺陷预测工具，但是将缺陷预测应用于实际项目仍然存在一定的挑战性。首先，大部分研究仅预测软件模块内部是否存在缺陷，而没有提供相应的预测依据和修复建议。其次大部分研究将程序模块的粒度设置为类/文件，因此即便能准确预测到软件模块内含有缺陷，仍然需要花费大量的时间去定位和修复这些缺陷。因此，在实际的项目应用中，如果想得到开发人员的积极反馈，需要进一步完善缺陷预测工具，给出预测结果及理由、缺陷位置和修复建议等。

References:

[1] Xiang Chen, Qing Gu, Wangshu Liu, Shulong Liu, and Chao Ni. Survey of static software defect prediction. Journal of Software, 2016, 27(1):1-25(in Chinese).

[2] Qing Wang, Shujian Wu, and Mingshu Li. Software defect prediction. Journal of Software, 2008, 19(7):1565-1580(in Chinese).

[3] Tracy Hall, Beecham, Sarah, Bowes, David, Gray, David, and Steve Counsell. A systematic literature review on fault prediction performance in software engineering. IEEE Transactions on Software Engineering, 2012,38(6):1276-1304.

[4] Seyedrebvar Hosseini, Burak Turhan, and Dimuthu Gunarathna. A systematic literature review and metaanalysis on cross project defect prediction. IEEE Transactions on Software Engineering, 2017, PP(99):1-1.

[5] Xiang Chen, Liping Wang, Qing Gu, Zan Wang, Chao Ni, Wangshu Liu, and Qiuping Wang. A survey on crss-project software defect prediction methods. Chinese Journal of Computers, 2018, 41(1):254-274.

[6] Xin Xia, David Lo, Sinno Jialin Pan, Nachiappan Nagappan, and Xinyu Wang. Hydra: Massively compositional model for cross-project defect prediction. IEEETransactionsonSoftwareEngineering,2016, 42(10):977-998.

[7] Chao Ni, Wang-Shu Liu, Xiang Chen, Qing Gu, Dao-Xu Chen, and Qi-Guo Huang. A cluster based feature selection method for cross-project software defect prediction. Journal of Computer Science and Technology(in english) , 2017, 32(6):1090-1107.

[8] ChaoNi, WangshuLiu, QingGu, XiangChen, andDaoxuChen. Fesch: A feature selection method using clusters of hybrid-data for cross-project defect prediction. In Proceedings of Computer Software and Applications Conference, 2017, pages 51-56.

[9] Seyedrebvar Hosseini, Burak Turhan, and Mika M˙ antyl˙ a. A benchmark study on the effectiveness of search based data selection and feature selection for cross project defect prediction.Information & Software Technology, 2018, 95:296-312.

[10] Rahul Krishna, Tim Menzies, and Wei Fu. Too much automation? the bellwether effect and its implications for transfer learning. In Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, 2016, pages 122-131.

[11] Zhiqiang Li, Xiaoyuan Jing, Xiaoke Zhu, and Hongyu Zhang. Heterogeneous defect prediction through multiple kernel learning and ensemble learning. In Proceedings of the IEEE International Conference on Software Maintenance and Evolution, 2017, pages 91-102.

[12] Jaechang Nam, Sinno Jialin Pan, and Sunghun Kim. Transfer defect learning, 2013, 8104:382-391.

[13] Fayola Peters, Tim Menzies, and Andrian Marcus. Better cross company defect prediction. In Proceedings of IEEE Working Conference on Mining Software Repositories, 2013, pages 409-418.

[14] Burak Turhan, Tim Menzies, Ays ¸e B Bener, and Justin Di Stefano. On the relative value of cross-company and within-company data for defect prediction. Empirical Software Engineering, 2009, 14(5):540-578.

[15] Thomas Zimmermann, Nachiappan Nagappan, Harald Gall, Emanuel Giger, and Brendan Murphy. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In Proceedings of the joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, 2009, pages 91-100.

[16] Y. U. Shusi and Shuigeng Zhou. Software engineering data mining: a survey. Journal of Frontiers of Computer Science and Technology, 2012, 6(31):1-31(in Chinese).

[17] Baljinder Ghotra, Shane Mcintosh, and Ahmed E. Hassan. Revisiting the impact of classification techniques on the performance of defect prediction models. In Proceedings of International Conference on Software Engineering, 2015, pages 789-800.

[18] FayolaPeters, TimMenzies, andLucasLayman. Lace2: better privacy-preserving data sharing for cross project defect prediction. In Proceedings of International Conference on Software Engineering, 2015, pages 801-811.

[19] Chakkrit Tantithamthavorn, Shane Mcintosh, Ahmed E. Hassan, Akinori Ihara, and Kenichi Matsumoto. The impact of mislabelling on the performance and interpretation of defect prediction models. In Proceedings of International Conference on Software Engineering, 2015, pages 812-823.

[20] Xiao-Yuan Jing, Fei Wu, Xiwei Dong, Fumin Qi, and Baowen Xu. Heterogeneous cross-company defect prediction by unified metric representation and cca-based transfer learning. In Proceedings of the Joint Meeting on Foundations of Software Engineering, 2015, pages 496-507.

[21] Mijung Kim, Jaechang Nam, Jaehyuk Yeon, Soonhwang Choi, and Sunghun Kim. Remi: defect prediction for efficient api testing. In Proceedings of the Joint Meeting on Foundations of Software Engineering, ESEC/FSE, 2015, pages 990-993.

[22] Jaechang Nam and Sunghun Kim. Clami: Defect prediction on unlabeled datasets (t). In Proceedings of International Conference on Automated Software Engineering, 2015, pages 452-463.

[23] Danijel Radjenović, Marjan Heričko, Richard Torkar, et al. Software fault prediction metrics: A systematic literature review. Information & Software Technology, 2013, 55(8):1397-1418.

[24] Tim Menzies, Jeremy Greenwald, and Art Frank. Data mining static code attributes to learn defect predictors. IEEE Transactions on Software Engineering, 2007, 33(1):2-13.

[25] Qinbao Song, Zihan Jia, M Shepperd, and Shi Ying. A general software defect-proneness prediction framework. IEEE Trans. Software Eng, 2011, 37(3):356-370.

[26] Amritanshu Agrawal and Tim Menzies. Is ”better data” better than ”better data miners”?: On the benefits of tuning smote for defect prediction. In Proceedings of the International Conference on Software Engineering, 2018, pages 1050-1061.

[27] Xiao Yu, Jin Liu, Zijiang Yang, Xiangyang Jia, Qi Ling, and Sizhe Ye. Learning from imbalanced data for predicting the number of software defects. In Proceedings of the International Symposium on Software Reliability Engineering, 2017, pages 78-89.

[28] Zhou Xu, Jin Liu, Zijiang Yang, Gege An, and Xiangyang Jia. The impact of feature selection on defect prediction performance: An empirical comparison. In Proceedings of the International Symposium on Software Reliability Engineering, 2016, pages 309-320.

[29] Takafumi Fukushima, Yasutaka Kamei, Shane McIntosh, Kazuhiro Yamashita, and Naoyasu Ubayashi. An empirical study of just-in-time defect prediction using cross-project models. In Proceedings of the 11th Working Conference on Mining Software Repositories ACM, 2014, pages 172-181.

[30] Ying Ma, Guangchun Luo, Xue Zeng, and Aiguo Chen. Transfer learning for cross-company software defect prediction. Information and Software Technology, 2012, 54(3):248-256.

[31] Song Wang, Taiyue Liu, and Lin Tan. Automatically learning semantic features for defect prediction. In Proceedings of the International Conference on Software Engineering, 2016, pages 297-308.

[32] Lin Chen, Bin Fang, Zhaowei Shang, and Yuanyan Tang. Negative samples reduction in cross-company software defects prediction. Information and Software Technology, 2015, 62:67 - 77.

[33] Peng He, Bing Li, and Yutao Ma. Towards cross-project defect prediction with imbalanced feature sets. Computer Science, 2014.

[34] Jaechang Nam and Sunghun Kim. Heterogeneous defect prediction. In Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, 2015, pages 508-519.

[35] Shi Zhong, Taghi M. Khoshgoftaar, and Naeem Seliya. Unsupervised learning for expert-based software quality estimation. In Proceedings of High Assurance Systems Engineering, 2004, pages 149-155.

[36] Feng Zhang, Quan Zheng, Ying Zou, and Ahmed E. Hassan. Cross-project defect prediction using a connectivity-based unsupervised classifier. In Proceedings of the International Conference on Software Engineering, 2016, pages 309-320.

[37] Yibiao Yang, Yuming Zhou, Jinping Liu, Yangyang Zhao, Hongmin Lu, Lei Xu, Baowen Xu, and Hareton Leung. Effort-aware just-in-time defect prediction: simple unsupervised models could be better than supervised models. In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2016, pages 157-168.

[38] Yuming Zhou, Yibiao Yang, Hongmin Lu, Lin Chen, Yanhui Li, Yangyang Zhao, Junyan Qian, and Baowen Xu. How far we have progressed in the journey? an examination of cross-project defect prediction. ACM Transactions on Software Engineering and Methodology, 2018, 27(1):1.

[39] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. IEEE Transactions on Knowledge & Data Engineering, 2010, 22(10):1345-1359.

[40] Fuzhen Zhuang, Ping Luo, Hui Xiong, Yuhong Xiong, Qing He, and Zhongzhi Shi. Cross-domain learning from multiple sources: A consensus regularization perspective. IEEE Transactions on Knowledge & Data Engineering, 2010, 22(12):1664-1678.

[41] Wenyuan Dai, Qiang Yang, Gui Rong Xue, and Yong Yu. Boosting for transfer learning. In Proceedings of Machine Learning, Proceedings of the Twenty-Fourth International Conference, 2007, pages 193-200.

[42] Wenyuan Dai, Gui Rong Xue, Qiang Yang, and Yong Yu. Transferring naive bayes classifiers for text classification. In Proceedings of National Conference on Artificial Intelligence, 2007, pages 540-545.

[43] Samarth Swarup and Sylvian R. Ray. Cross-domain knowledge transfer using structured representations. In Proceedings of National Conference on Artificial Intelligence, 2006, pages 506-511.

[44] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. Journal of Machine Learning Research, 2004, 5(12):1205-1224.

[45] Kenji Kira and Larry A Rendell. The feature selection problem: traditional methods and a new algorithm. In Proceedings of Tenth National Conference on Artificial Intelligence, 1992, pages 129-134.

[46] Marco D’Ambros, Michele Lanza, and Romain Robbes. Evaluating defect prediction approaches: a benchmark and an extensive comparison. Empirical Software Engineering, 2012, 17(4):531-577.

[47] Fayola Peters and Tim Menzies. Privacy and utility for defect prediction: Experiments with MORPH. In International Conference on Software Engineering, 2012, pages 189-199.

[48] Rongxin Wu, Hongyu Zhang, Sunghun Kim, and Shing Chi Cheung. Relink: recovering links between bugs and changes. In Proceedings of ACM Sigsoft Symposium and the European Conference on Foundations of Software Engineering, 2011, pages 15-25.

[49] M D'Ambros, M Lanza, and R Robbes. An extensive comparison of bug prediction approaches. Mining Software Repositories, 2010, pages 31-41.

[50] Jiawei Han and Micheline Kamber. Data mining: Concepts and techniques. Data Mining Concepts Models Methods & Algorithms Second Edition, 2011, 5(4):1 - 18.

[51] Frank Wilcoxon. Individual comparisons by ranking methods. Biometrics Bulletin, 1945, 1(6):80-83.

[52] Janez Ar. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research, 2006, 7(1):1-30.

[53] Shulong Liu, Xiang Chen, Wangshu Liu, Jiaqiang Chen, Qing Gu, and Daoxu Chen. Fecar: A feature selection framework for software defect prediction. In Proceedings of Computer Software and Applications Conference, 2014, pages 426-435.

[54] Kehan Gao, Taghi M Khoshgoftaar, Huanjing Wang, and Naeem Seliya. Choosing software metrics for defect prediction: an investigation on feature selection techniques. Software Practice & Experience,2011, 41(5):579-606.

[55] Sunghun Kim, Hongyu Zhang, Rongxin Wu, and Liang Gong. Dealing with noise in defect prediction. In Proceedings of International Conference on Software Engineering, 2011, pages 481-490.

[56] Steffen Herbold. Crosspare: A tool for benchmarking cross-project defect predictions. In Proceedings of International Conference on Automated Software Engineering Workshop, 2015, pages 90-96.

[57] Zhimin He, Fengdi Shu, Ye Yang, Mingshu Li, and Qing Wang. An investigation on the feasibility of cross project defect prediction. Autom. Softw. Eng., 2012, 19(2):167-199

[58] Foyzur Rahman, Daryl Posnett, and Premkumar Devanbu. Recalling the "imprecision" of cross-project defect prediction. In Proceedings of ACM SIGSOFT Symposium on the Foundations of Software Engineering, 2012, pages 1-11.

[59] Ahmed E. Hassan David Lo Jianwei Yin Xiaohu Yang Zhiyuan Wan, Xin Xia. Perceptions, expectations, and challenges in defect prediction. IEEE Transactions on Software Engineering, 2018.

[60] Chris Lewis, Zhongpeng Lin, Caitlin Sadowski, Xiaoyan Zhu, Rong Ou, and E James Whitehead Jr. Does bug prediction support human developers? findings from a google case study. In Proceedings of the 2013 International Conference on Software Engineering,2013, pages 372–381.

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

[2] 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, 2018,41（1）:254-274.