知识系统化：面向持续集成的机器学习

Ali Kazemi Arani计算机科学学院 澳大利亚阿德莱德大学澳大利亚阿德莱德
ali.kaze
miarani@adelaide.edu.au

Mansooreh Zahedi计算与信息系统学院 墨尔本大学澳大利亚墨尔本
mansooreh.zahedi@unimelb.edu.au

Triet Huynh Minh Le Muhammad Ali Babar计算机科学学院 计算机科学学院 阿德莱德大学 阿德莱德大学阿德莱德，澳大利亚 阿德莱德，澳大利亚 triet.h.le@adelaide.edu.au
ali.babar@adelaide.edu.au

arXiv:2304.02829v1 [cs.SE] 6 Apr 2023

摘要—持续集成（CI）已成为一种成熟的软件开发实践，可在软件开发过程中自动、持续地集成代码更改。文献中报告的用于自动化持续集成阶段的机器学习（ML）方法日益增多。对用于持续集成阶段的机器学习方法进行知识系统化（SoK）正当其时，也具有重要意义。本文从不同方面系统化梳理了机器学习在持续集成中的应用。我们的系统性分析还指出了现有基于机器学习的解决方案的不足之处，改进这些不足有助于推动该领域的发展。

索引词—**持续集成、机器学习、系统性文献综述**

I. 引言

近年来，软件开发行业明显转向采用持续集成（持续集成）实践。持续集成流程是一种软件开发方法，旨在通过持续将代码更改集成到共享代码库中，提高软件交付的速度和可靠性。持续集成流程的目标是在开发过程的早期发现并修复问题和缺陷，避免它们演变成更难解决且成本更高的重大问题 [1]。

持续集成和 DevOps 日益普及，以及这些流程所涉及的数据量和复杂性不断增加，促使研究人员提出基于机器学习（机器学习）的解决方案，以实现持续集成阶段的自动化 [1]并进一步推动 AIOps 的实现 [2]。机器学习方法可以通过分析开发数据、部署日志文件以及运行环境中的数据，并作出自动化决策来支持持续集成中的快速反馈循环，而这些正是机器学习方法所独有的优势 [3]。这些技术还可用于高效预测复杂任务的结果。例如，机器学习方法可以根据代码的先前版本预测不同类型的软件缺陷，而无需运行当前版本 [4]。此外，基于机器学习的解决方案可以提供准确的估算，促进对频繁变化的适应 [5]，并协助软件工程师及时作出决策 [3]。在持续集成环境中，基于机器学习的解决方案所凸显的优势将有助于减少人工干预并提高基于云环境的持续集成服务效率，而这些是基于云环境的基本要求 [6]。

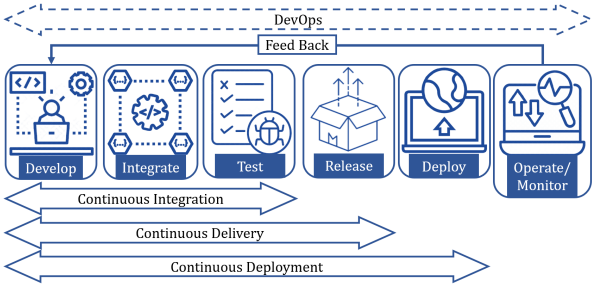


图 1. 持续软件工程的关键步骤和概念。**注释：**矩形：步骤。双向箭头：自动化阶段。虚线箭头：跨越多个步骤。

鉴于在持续集成中应用机器学习解决方案时所采用的技术多种多样，以及人们对这一领域日益增长的兴趣，有必要系统地识别通过机器学习方法实现持续集成任务自动化时所采用的先进实践。这些知识可以为实践者和研究人员提供有价值的参考，帮助他们了解现有机器学习技术的潜力，并作出明智的决策，将机器学习模型应用于真实的持续集成环境中。据我们所知，目前尚不存在针对持续集成先进机器学习实践的知识系统化（SoK）。

为弥合这一差距，我们回顾了过去十年间为持续集成阶段开发的现有机器学习解决方案。具体而言，我们分析了在同行评审场所发表的相关科学文章。我们首先确定了已通过机器学习实现自动化的持续集成阶段。此外，我们还深入探讨了这些最先进的机器学习解决方案所采用的关键机器学习实践，例如数据/特征工程、学习算法和评估流程。这些知识可为从业者提供指导，帮助他们采用或开发高性能机器学习解决方案，以大规模实现持续集成阶段的自动化。我们还讨论了如何改进当前机器学习解决方案在真实世界的持续集成环境中的应用。

II. 背景

本综述聚焦于持续集成（CI），持续集成是持续软件工程（CSE）的一部分，旨在根据客户体验以及软件运营和维护提供快速且频繁的反馈 [7]。如图1 所示，持续集成是持续软件工程的第一步，先于持续交付（CDE）和持续部署（CD）阶段。因此，提高持续集成的性能将产生积极影响

影响CSE 的整个流程。持续集成包括软件构建、测试和验证 [7]。该流程要求一天内进行多次集成，并需要频繁的自动化测试和及时反馈，以防止问题蔓延到交付阶段，或影响其他团队成员的开发流程 [1]。最后，开发人员通过持续集成中的验证阶段，以及通过维护（在运维中进行监控）已部署的软件，获得有关检测到的缺陷或性能问题的反馈 [8]。

大多数关于 ML 方法在持续集成中应用的现有系统综述/系统映射研究，主要聚焦于测试用例优先级排序（TCP）和测试用例选择（TCS）方法。Pan et al. [9] 调查了持续集成环境中所提出的基于机器学习的 TCP 和 TCS 解决方案所采用的特征集和评估指标。此外，在 [10],中，Lima et al. 仅关注持续集成中的测试用例优先级排序方法，并报告了该任务所采用的机器学习方法类型和评估方法。相比之下，我们的综述并未将搜索范围限制于持续集成中的某一特定解决方案，而是调查了持续集成环境中的所有基于机器学习的方法。这种方法有助于更全面地理解机器学习在持续集成中的应用，以及该领域最先进方法所采用的技术。本研究旨在为该领域的研究人员和实践者提供有价值的见解，并确定需要进一步研究的方向，以提高基于机器学习的解决方案在持续集成中的有效性。

III. 方法论

我们旨在遵循系统文献综述（SLR）指南 [11]，对基于机器学习的方法在持续集成中的应用进行知识系统化。我们对相关研究进行了严格且系统的检索，并采用各种标准来确保纳入综述的研究质量。为实现这一目标，我们研究了以下两个研究问题（RQs）。

- **RQ1**：基于机器学习的解决方案所涉及的持续集成阶段有哪些?
- **RQ2**：在各个持续集成阶段的最先进机器学习解决方案的开发中采用了哪些技术?

这些问题的答案将有助于突出机器学习已展现出潜力的持续集成阶段，以及利用机器学习实现持续集成阶段自动化的当前最佳实践。

A. 研究筛选

我们使用 Scopus[1] 作为查找相关研究的主要来源，因为它收录了软件工程和计算机科学领域的许多期刊和会议，这些期刊和会议与持续集成相关的研究 [12], [13]有关。

随后，我们设计了一个检索字符串，以从 Scopus 中检索相关研究。我们的检索字符串基于三个部分进行设计，包括 **A)** 机器学习、人工智能及其相关同义词，**B)**持续集成、DevOps 和 CSE 的同义词，以及 **C)** “软件”、“信息系统”、“信息技术”、“云”和“服务”的同义词工程”术语。这种方法确保我们的搜索全面且涵盖了该领域广泛的相关研究。通过在搜索字符串中结合这三个部分，我们得以检索到大多数关于基于机器学习的方法在持续集成和 DevOps 中应用的相关研究。

[1]https://www.scopus.com/

B. 纳入与排除标准

为识别并剔除不相关及低质量的研究，我们制定了以下纳入标准，并依据这些标准筛选论文。

1)页数超过四页的研究论文 2) 论文的全文以英文提供 3) 论文的关键主题是将基于机器学习的方法应用于持续集成

使用上述设置，我们在 Scopus索引系统 上运行了检索字符串，共检索到1,662项研究。在应用纳入标准并采用滚雪球法以降低遗漏研究的风险 [14], 后，我们获得了包含 44 项相关研究的最终列表。直观而言，最新的研究通常会提出最先进的解决方案。然而，部分最新研究可能并未包含最先进的机器学习解决方案。为确保与 RQ2 的相关性，我们全面审查了每篇论文的全文，以确认每篇论文是否提出了一种新颖的方法，并证明其方法优于相关文献中当前最先进的方法。这将帮助实践者和研究人员确定在利用其基于机器学习的方法实现持续集成任务自动化时所适用的情境。

C. 数据提取与数据综合

基于系统文献综述指南，我们创建了数据提取表，以系统地从选定研究中提取信息，从而回答研究问题 [11]。我们采用主题分析 [15]来综合和分析RQ1的数据，并采用统计分析来分析RQ2。该方法使我们能够从选定研究中提取和组织数据，识别模式和主题，并回答研究问题。

IV. 结果

A. RQ1：持续集成阶段与机器学习应用

通过审阅和分析选定论文，我们确定了五个由机器学习实现自动化的持续集成测试阶段。图2展示了基于机器学习的方法在这五个持续集成阶段中的应用。以下将根据这五个已确定阶段在持续集成流水线中的顺序，对其进行介绍。

1) 单元测试（*UT*）：UT 是持续集成流水线 [16] 中事件链的第一步。当开发者检入（提交）新编译或修改后的代码并构建代码时，该测试会在隔离环境中检查并验证开发的代码 [17]。为了确保主分支不含错误，开发者在此步骤中会将所有更改提交到开发者分支[18]。在选出的 44 项研究中，有两项提出了基于机器学习的解决方案，通过预测单元测试的结果而无需实际运行测试，以降低发布有缺陷软件的风险。Lee et al. [19],提出了一种使用机器学习预测单元测试结果的方法，并展示了其

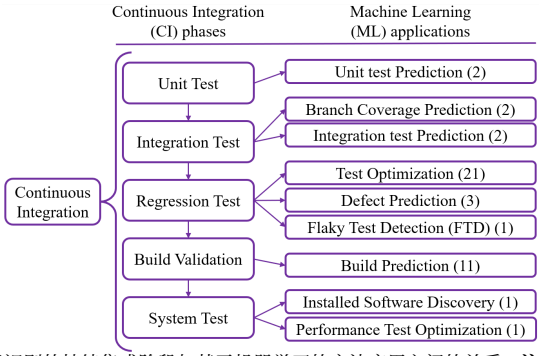


图 2. 已识别的持续集成阶段与基于机器学习的方法应用之间的关系。**注：** 括号内列出了已发表论文的数量。

估计静态检查器引发的警报状态的有效性。Vig等人 [20] 提出了一种通过使用机器学习预测单元测试结果来估计所需测试工作量的方法。他们比较了不同的机器学习方法和特征集，以评估其方法的有效性。

2）集成测试（*IT*）：集成测试是持续集成中的一个步骤，在新开发的模块通过单元测试验证后，将其与同一分支上的现有模块进行集成 [21]。集成测试的目的是验证系统或子系统的所有模块能否无缝协同工作。在此阶段，机器学习技术可用于预测测试用例的结果 [21], , 并识别测试用例的分支覆盖率，从而在不遗漏任何需要测试的更改的情况下，以更少的测试运行次数检测缺陷 [22]。

3）回归测试（*RT*）：在验证并测试新软件单元或经过修改的软件单元与其他软件单元的集成后，需要根据先前设计的测试用例对当前版本进行完整测试 [23]。全部 25 项回归测试研究的目标可分为以下几类：采用测试用例优先级排序（TCP）和测试用例选择（TCS）策略开展的“测试优化”研究；将已提交代码的测试套件结果预测为“缺陷预测”研究；以及“检测不稳定测试”研究。不稳定测试是指通过或失败结果不一致的测试，会使持续集成流水线变得不可靠 [24]。在测试用例优先级排序和测试用例选择研究领域，机器学习模型的目标是优先安排更有可能发现开发中软件缺陷的测试用例，或选择其中的一个子集 [25]。这一目标通过分析源代码中的先前更改以及先前测试运行的结果来实现，以识别表明更有可能发现缺陷的模式。

4）构建验证（*BV*）：根据前几个持续集成阶段的结果，开发人员可以较有把握地了解已开发软件单元的功能和性能。因此，这个版本的软件可以合并到主分支中。构建验证测试旨在确保集成代码的稳定性，以便在发布软件并将其交给系统测试团队之前进行验证 [26]。由于构建软件产品的计算成本很高 [27],，构建结果预测是所介绍的基于机器学习的解决方案在构建验证阶段的主要目标。构建预测旨在通过预测构建过程的结果，降低软件的资源使用量和构建时间。预测基于对以下数据的分析得出：项目内数据或跨项目数据 [28]。该持续集成阶段的 11 项研究中，有 9 项使用持续集成流水线的批处理数据来训练其模型。然而， [26] 和 [29] 的作者分别使用数据流和数据变化来训练其机器学习模型。

5) 系统测试 *(ST)*：在此步骤中，会测试所有与持续集成相关的方面和质量因素，包括不同软件单元的性能、功能和兼容性。此阶段可以由部分用户或内部系统测试团队执行 [30]。例如，会要求测试用户（在正式发布前接收所开发软件的用户）提供其使用完全集成软件产品的体验 [31]。此阶段的基于机器学习的解决方案旨在发现已安装软件，以确保被测系统的合规性、安全性和效率 [32], , 并优化测试套件，以检测所开发软件产品中的性能缺陷 [31]。

B. RQ2：用于自动化持续集成测试阶段的最先进基于机器学习的方法

根据 RQ1 的结果，我们确定了每项持续集成自动化任务中最先进的机器学习解决方案。所选论文列表、所收集数据的属性，以及用于训练和评估机器学习方法的采用的技术，均在表I 中进行了分析和展示。下文将介绍开发最先进的机器学习解决方案的不同阶段中获得的关键经验。

1) 数据源和数据准备：根据所审阅的研究，持续集成测试领域的大多数论文（44篇中的24篇）采用了开源项目。开源项目之所以广受欢迎，是因为能够获取不同类型的开源项目数据，包括源代码、测试代码及其元数据，还可以访问基于云的工具（如 Travis CI和 GitHub）中的数据。例如，Pan et al. [33],在242个开源项目上评估了其模型，Parryetal. [24] 则在26个开源 Python项目上评估了其模型。

此外，表 I 展示了用于确保机器学习模型输入数据质量的不同数据准备技术。机器学习的数据准备第一步是使用各种重采样算法来平衡数据。这些算法包括 Weka [21], 中采用 SMOTE 等过采样技术来增加少数类中的实例数量 [35], , 以及采用 TOMEK 等欠采样技术 [36]。此外，研究人员还可以通过使用 SMOGN技术，结合过采样和欠采样方法 [37]。另一方面，在持续集成语境下，目前可用的数据集需要通过移除无关数据 [19]或噪声 [32]进行清理。Yaraghiet al. [25],采用了一种新颖的数据准备方法，并创建了一个依赖图，其中节点表示源文件，每条边显示从源节点到目标节点的依赖关系，以确定测试用例的代码覆盖率。

2) 特征工程：由于数据集中各特征值的分布各不相同，对其进行归一化非常重要。

表 I 最先进解决方案采用的技术、数据及特征。

参考文献	持续集成阶段- 机器学习应用	数据源- 数据类型	Data 准备	特征 工程	学习 算法（多个）	评估方法 - 指标
Lee et al. [19]	UT - 测试预测	三星项目（工业）- 源代码	过滤 - 选择源代码行	标注 - word2vec	NN （监督式）	随机 K 折交叉验证 - F1、精确率、召回率和准确率
Grano 等人 [22]	IT - 分支覆盖率预测	Google 和 Apache（OS）- 源代码和代码 MD	未报告	缩放 - 归一化	RF、NN、SVM (监督式)	随机 K 折交叉验证 - 平均绝对误差（MAE）
Abdalkareem 等人 [21]	IT - 测试预测	GitHub（操作系统）- 提交元数据	平衡 - Weka	加权 - TF/IDF	决策树 (C4.5) (监督式)	随机 K 折交叉验证 - F1, 精确率, 召回率和曲线下面积
Yaraghi et al. [25]	RT - 测试优化	GitHub（操作系统）- 构建日志、测试代码和源代码 MD	构建数据 - 依赖图的源代码文件和测试用例	缩放和加权 - 归一化和 TF/IDF	RF、SVM、XGBoost （监督式）	随机 K 折交叉验证 - $APFD_C$
Pan 等人 [33]	RT - 缺陷预测	GitHub（操作系统）- 代码 MD 和测试 MD	过滤 - 基于最后两个结果测试用例的	未报告	决策树、RF、MLP、NB, SVM, LR (监督式)	排序的 K 折 - F1、曲线下面积、MCC 和 G-度量
Parry et al. [24]	RT - 不稳定测试检测	Apache（操作系统）- 测试 MD 和测试代码	平衡 - TOMEK、SMOTE, 编辑后的最近邻	缩放 - 标准化	RF （监督式）	分层 K 折 - F1、精确率、召回率, FP、FN 和 TP
Saidani 等人 [34]	BV - 构建预测	TravisTorrent (OS) - 源代码 MD 和测试 MD	平衡 - SMOTE	未报告	LSTM、RNN （监督式）	排序的 K 折 - F1、精确率、召回率、准确率和曲线下面积
Byrne 等人 [32]	ST - 已安装软件发现	模拟云环境 - 系统日志	过滤 - 降噪	标注 - Columbus	强化学习（Vowpal Wabbit） （监督式）	随机 K 折交叉验证 - F1、精确率和召回率
Porres 等人 [31]	ST - 性能测试 Op优化	模拟测试场景 - 测试 MD	未报告	缩放 - 归一化	DNN （监督式）	逐步评估 - 阳性预测值（阳性预测值）

注释： OS = 开源,MD = 元数据,NN = 神经网络,DT = 决策树, RF = 随机森林, DNN = 深度神经网络, RNN = 递归神经网络, LR = 线性回归, RL = 强化学习。

性和范围。可以通过使用统计方法来实现这一点，例如将值重新缩放到特定范围内，如 [0,1] 进行归一化 [22]，通过将特征值除以所有值的标准差进行标准化 [24],或应用对数变换等技术，使值达到相同的数量级，并减小极高值或极低值的影响 [38]。由于特征值的多样性以及数值特征的可用性，缩放在持续集成的语境中被广泛采用（44项研究中有17项）。此外，为了将基于文本的值转换为机器学习方法能够理解的输入，例如系统日志[32] 和源代码 [19]，，通常会使用标注和分词技术（44项研究中有9项）。在所识别的最先进研究中，采用了 Word2vec [19] 和 Columbus [32] 方法从文本中提取标签。TF-IDF（词频-逆文档频率）是一种用于以归一化方式衡量文本中词语重要性的技术 [21]。它可以通过提高低频出现词语的权重，帮助解决文本数据中的类别分布不均衡问题。

*3)*学习算法：如表 I 所示，当前所有最先进的研究均采用了监督学习方法。此外，需要注意的是，在 2020 年至本文撰写时发表的所有论文都采用了监督式机器学习算法。这一趋势表明，近年来监督式方法在研究人员中越来越受欢迎。监督式学习的普及可归因于模型可接受的准确率，以及持续集成环境中自动化工具所提供的带标签数据。

此外，表 I 显示，决策树和随机森林等基于树的算法已被广泛用作最先进解决方案中的分类方法。

项。基于树的算法也被用于从44项入选研究中进行的19项研究。基于树的算法之所以受欢迎，有三个原因。首先，持续集成环境会持续产生海量数据，机器学习模型需要频繁重新训练 [25]。由于训练基于树的算法所需的计算资源较少，因此适合在实际的持续集成环境中采用 [39]。其次，基于树的算法在对未见过的数据进行分类时，通常具有较高的性能 [26]。例如，为了在持续集成中自动执行测试用例优先级排序任务，Yaraghi et al. [25] 将其基于树的（XGBoost）模型与BERT [40],这一最先进的语言表示模型进行了比较。他们发现，尽管其模型的训练成本低于 BERT，但其方法取得了与BERT相对相近的结果。第三，基于树的算法具有良好的可解释性，因此人类能够理解它们 [41]。

根据表 I，除了基于树的算法外，神经网络（NN）算法也在六项研究中得到普遍应用，包括长短期记忆网络（LSTM）、多层感知机（MLP）、递归神经网络（RNN）和深度神经网络（DNN）。尽管神经网络算法的训练需要大量计算资源，但其广受欢迎的主要原因之一是能够自动从数据集中提取特征并做出准确预测 [31]。

*4)*评估方法：K 折法广泛用于评估机器学习模型。它包括将数据划分为 K 个相等的部分（折），然后在 K个评估轮次中的每一轮，使用数据中的K-1 个部分训练模型，并在剩余的未见过的部分上进行评估。根据所选研究，其中44项研究中有19 项使用了这一技术。一般而言，评估方法分为随机抽样和排序抽样。在随机抽样下，数据的每一行

表 II 常用的性能度量指标及其描述和公式。

度量	描述	公式
精确率	检测到的阳性结果百分比正确的实例	$\frac{TP}{TP+FP}$
召回率	正例的比例正确识别的那些	$\frac{TP}{TP+FN}$
F1-得分	召回率与精确率的调和平均值。	$\frac{2\times Precision\times Recall}{Precision+Recall}$
准确率	正确分类的实例百分比例	$\frac{TP+TN}{TP+FP+FN+TN}$

注： TP=真正例，TN=真负例，FP=假正例，FN=假负例。

具有被选作训练、测试和留出数据集的相同可能性，而在排序抽样中，测试集必须比训练数据集更新。在排序K 折法中，与训练数据集相比，未见测试数据折必须从较早的折中选取。

F1 分数、精确率和召回率是文献中常用的评估指标，分别有 16、20 和 23 项研究使用了这些指标。它们是该领域使用最频繁的指标。这些指标的描述和公式见表 II。此外，还使用了曲线下面积（“ROC 曲线下面积”），其中 ROC 图以真正例率（ TPR 或召回率）为纵轴，以假正例率（ $\frac{FP}{FP+TN}$ ）为横轴。

与有效性指标相比，效率（成本）评估指标受到的关注显著较少。具体而言，只有 Yaraghi et al. [25]报告了与成本相关的评估指标，即成本感知的故障检测平均百分比（$APFD_c$）。Malishevsky et al. [42] 提出了 $APFD_c$ 的另一种形式，将检测成本与故障严重程度结合起来，如公式（1）所示。在该指标中， n 表示测试套件 T 中的测试用例数量，其成本为 t_i ，而 f_i 表示故障的严重程度。TF_i 是第一个揭示该故障的测试用例 i。

$$APFD_c = \frac{\sum_{i=1}^{m}(f_i \times (\sum_{j=TF_i}^{n} t_j - \frac{1}{2}t_{TF_i}))}{\sum_{i=1}^{n} t_i \times \sum_{i=1}^{m} f_i} \quad (1)$$

V. 结论与未来机遇

在本研究中，我们梳理了由机器学习自动化的持续集成阶段，以及这一新兴领域中相应的最先进的机器学习解决方案。我们发现，来自多个开源项目的数据通常被用于开发机器学习解决方案。此外，研究还采用了过滤和平衡输入数据、缩放特征值等预处理技术，以提升模型性能。在所选研究中，基于树的算法较为常见，因为它们能够以较低的计算开销从大规模持续集成数据中提取模式。此外，我们还总结了所采用的各种评估方法和指标，以便于未来工作与既有研究之间进行比较。尽管当前机器学习解决方案取得了令人鼓舞的结果，但我们指出了可能阻碍这些解决方案在现实世界中得到应用的三个关键挑战，并提出了应对这些挑战的潜在方向。

现实性评估： 尽管随机评估方法存在已知局限，但该领域相当数量的研究仍采用了这些技术。具体而言，文献中有五项最先进的研究，以及接受评审的44项研究中的18项，采用了这类方法。随机评估方法存在局限，在现实环境中可能并不可靠，因为它们使用未来数据训练模型，而持续集成环境中的数据是按时间顺序生成的，这会导致结果缺乏稳健性和泛化能力。因此，机器学习模型必须仅使用当前数据进行训练，并在新的、未见过的数据上进行评估，以确保在部署到现实环境中时能够对模型性能进行现实性评估 [43]。

成本基准： 除了报告性能和有效性指标外，还需要报告成本指标。这些成本指标包括训练所需时间以及执行预测/分类所需时间。所有选定的先进研究中，只有 [25], 使用了基于成本的指标。然而，这些研究并未报告所提出模型的训练和预测成本/时间。因此，可以结合性能和成本指标进行全面的成本效益分析。这种分析能够更详细地揭示在工业环境中采用所提出模型时可能面临的权衡。

尚未充分探索的持续集成领域：根据所审阅的研究，机器学习方法在持续集成测试环境中的应用仅限于九个领域，如图2所述。Yang 等人开展的一项关于软件工程中预测模型的近期综述研究 [44], 强调了将基于机器学习的方法应用于各种软件工程测试领域的潜力。然而，经过进一步考察可以发现，在持续集成文献背景下，仍有若干尚未得到充分探索的软件工程测试领域。一些领域包括崩溃预测、错误行为检测、缺陷严重性分类、测试报告分类与评估、故障注入、缺陷报告管理任务（包括缺陷报告分配、缺陷报告预测与分类）、软件质量评估与可靠性、漏洞和恶意软件检测、代码异味检测、缺陷可追溯性检测方法，以及代码克隆检测。目前仍不清楚这些领域及其各自的机器学习解决方案在多大程度上能够适用于持续集成。因此，建议研究人员和从业者除了关注当前领域外，也应探索这些尚未涉及的领域，以最大限度地利用机器学习实现持续集成自动化。

致谢

我们感谢 DrZohaib Md.Jan 和 Roshan Namal Rajapakse 在本研究第一阶段收集和分析数据过程中所作的贡献。

参考文献

[1] M.Shahin、M. A.Babar和 L. Zhu，《持续集成、交付与部署：关于方法、工具、挑战和实践的系统综述》，*IEEEAccess*，第 5 卷，第 3909–3943 页，2017年。[2] A.Sen，《DevOps、DevSecOps、AIOps 范式与集成测试运营》，载于计算、通信与智慧世界的演进技术，第 211–221 页，Springer，2021 年。

[3] I. Figalist、A. Biesdorf、C. Brand、S. Feld 和 M. Kiermeier，“使用无监督机器学习支持 DevOps 反馈循环”，载于《2019 年 IEEE 智能系统与应用创新国际研讨会（INISTA）》，第 1–6 页，IEEE，2019 年。[4] M. Kawalerowicz 和 L. Madeyski，“持续构建结果预测：真实软件项目设置中的小样本实验”，载于《工业、工程及其他应用智能系统国际会议》，第 412–425 页，Springer，2021 年。[5] P. Pospieszny，“软件估算：迈向规范性分析”，载于《第 27 届软件度量国际研讨会暨第 12 届软件过程与产品度量国际会议论文集》，第 221–226 页，2017 年。[6] B. P. Rimal、A. Jukan、D. Katsaros 和 Y. Goeleven，“云计算系统的架构需求：企业云方法”，《网格计算期刊》，第 9 卷，第 3–26 页，2011 年。[7] B. Fitzgerald 和 K.-J. Stol，“持续软件工程：路线图与议程”，《系统与软件期刊》，第 123 卷，第 176–189 页，2017 年。[8] P. Debois 等，“DevOps：一场正在酝酿的软件革命”，《信息技术管理期刊》，第 24 卷，第 8 期，第 3–39 页，2011 年。[9] R. Pan、M. Bagherzadeh、T. A. Ghaleb 和 L. Briand，“使用机器学习进行测试用例选择与优先级排序：系统文献综述”，《实证软件工程》，第 27 卷，第 2 期，第 1–43 页，2022 年。[10] J. A. P. Lima 和 S. R. Vergilio，“持续集成环境中的测试用例优先级排序：系统映射研究”，《信息与软件技术》，第 121 卷，第 106268 页，2020 年。[11] B. A. Kitchenham 和 S. Charters，“软件工程中开展系统文献综述的指南：技术报告”，软件工程组，EBSE 技术报告，基尔大学与达勒姆大学计算机科学系，第 2 卷，2007 年。[12] B. Kitchenham、R. Pretorius、D. Budgen、O. P. Brereton、M. Turner、M. Niazi 和 S. Linkman，“软件工程中的系统文献综述——一项三级研究”，《信息与软件技术》，第 52 卷，第 8 期，第 792–805 页，2010 年。[13] M. Daneva、D. Damian、A. Marchetto 和 O. Pastor，“需求工程中的实证研究方法与研究：我们已经走了多远？”，《系统与软件期刊》，第 95 卷，9，2014 年，第 1 页。[14] C. Wohlin，“系统文献研究中的滚雪球法指南及其在软件工程中的复现”，载于《第 18 届软件工程评估与评价国际会议论文集》，第 1–10 页，2014 年。[15] V. Braun 和 V. Clarke，“在心理学中使用主题分析”，《心理学定性研究》，第 3 卷，第 2 期，第 77–101 页，2006 年。[16] S. Stolberg，“通过持续集成实现敏捷测试”，载于《2009 年敏捷会议》，第 369–374 页，IEEE，2009 年。[17] Y. Zhou、H. Leung、Q. Song、J. Zhao、H. Lu、L. Chen 和 B. Xu，“深入研究面向对象系统中的结构指标与单元测试性的关系”，《中国科学：信息科学》，第 55 卷，第 12 期，第 2800–2815 页，2012 年。[18] R. Martins、R. Abreu、M. Lopes 和 J. Nadkarni，“持续集成中的测试套件选择的监督式学习”，载于《2021 年 IEEE 软件测试、验证与确认国际会议研讨会（ICSTW）》，第 239–246 页，IEEE，2021 年。[19] S. Lee、S. Hong、J. Yi、T. Kim、C.-J. Kim 和 S. Yoo，“使用卷积神经网络对持续集成中的误报静态检查器告警进行分类”，载于《2019 年第 12 届 IEEE 软件测试、验证与确认会议（ICST）》，第 391–401 页，IEEE，2019 年。[20] V. Vig 和 A. Kaur，“使用机器学习算法估算传统发布模型和快速发布模型的测试工作量并进行预测”，《智能与模糊系统期刊》，第 35 卷，第 2 期，第 1657–1669 页，2018 年。[21] R. Abdalkareem、S. Mujahid 和 E. Shihab，“一种改进 CI 跳过提交检测的机器学习方法”，《IEEE 软件工程汇刊》，2020 年。[22] G. Grano、T. V. Titov、S. Panichella 和 H. C. Gall，“自动化测试中的分支覆盖率预测”，《软件：演化与过程》，第 31 卷，第 9 期，第 e2158 页，2019 年。[23] S. Ali、Y. Hafeez、S. Hussain 和 S. Yang，“面向敏捷软件开发与持续集成策略的增强型回归测试技术”，《软件质量期刊》，第 1–27 页，2019 年。[24] O. Parry、G. M. Kapfhammer、M. Hilton 和 P. McMinn，“评估用于机器学习检测有序与非有序——依赖型易失测试”，载于*2022* 年*IEEE*软件测试、验证与确认会议（*ICST*），第 93–104页，IEEE，2022 年。[25] A. S. Yaraghi、M. Bagherzadeh、N.Kahani和L. Briand，“持续集成环境中的可扩展且准确的测试用例优先级排序”，*IEEE* 软件工程汇刊，2022年。[26] J. Finlay、R. Pears 和 A.M. Connor，“使用源代码指标进行软件构建结果预测的数据流挖掘”，信息与软件技术，第56卷，第 2 期，第183–198页，2014 年。[27] Z.Xie 和M. Li，“通过半监督在线曲线下面积优化减少持续集成中的软件构建工作量”，载于*IJCAI*，第2875–2881 页，2018 年。[28] J. Xia、Y. Li和 C. Wang，“持续集成结果跨项目可预测性的实证研究”，载于*2017* 年第*14*届 *Web* 信息系统与应用会议（*WISA*），第 234–239 页，IEEE，2017年。[29] F.Hassan 和 X.Wang，“持续集成中用于避免停滞的变更感知构建预测模型”，载于*2017* 年 *ACM/IEEE* 软件工程与度量实证国际研讨会（*ESEM*），第 157–162 页，IEEE，2017 年。[30] J. Vemulapati、A. S. Khastgir 和C.Savalgi，“基于人工智能的大数据与云驱动应用性能基准测试及分析：深入考察”，载于*2019*年*ACM/SPEC*性能工程国际会议论文集，第103–109 页，2019 年。[31] I. Porres、T. Ahmad、H. Rexha、S. Lafond 和 D.Truscan，“使用判别器神经网络的自动探索式性能测试”，载于*2020* 年 *IEEE*软件测试、验证与确认国际会议研讨会（*ICSTW*），第 105–113 页，IEEE，2020年。[32] A. Byrne、E. Ates、A. Turk、V.Pchelin、S.S. Duri、S. Nadgowda、C.Isci 和A. Coskun，“Praxi：从实践中学习的云软件发现”，*IEEE* 云计算汇刊，2020年。[33] C. Pan 和 M. Pradel，“持续测试套件失败预测”，载于第*30* 届 *ACMSIGSOFT* 软件测试与分析国际研讨会论文集，第 553–565 页，2021 年。[34] I. Saidani、A. Ouni 和 M. W. Mkaouer，“使用深度学习改进持续集成构建失败预测”，自动化软件工程，第29 卷，第 1 期，第1–61页，2022 年。[35] N. V. Chawla、K. W. Bowyer、L. O.Hall和W. P. Kegelmeyer，“SMOTE：合成少数类过采样技术”，人工智能研究期刊，第 16 卷，第 321–357 页，2002 年。[36] S. Dalla Palma、D. Di Nucci、F. Palomba 和D.A.Tamburri，“使用产品和过程指标进行基础设施即代码的项目内缺陷预测”，*IEEE*软件工程汇刊，第 1 1 页，2021 – 。[37] A. Sharif、D. Marijan和 M.Liaaen，“Deeporder：用于持续集成测试的测试用例优先级排序的深度学习方法”，*arXiv* 预印本*arXiv:2110.07443*，2021 年。[38] G. Grano、T. V. Titov、S. Panichella 和 H. C. Gall，“它会有多高？使用机器学习模型预测自动化测试中的分支覆盖率”，载于*2018*年 *IEEE* 软件质量评估机器学习技术研讨会（*MaLTeSQuE*），第 19–24 页，IEEE，2018 年。[39] K. W. Al-Sabbagh、R.Hebig 和 M.Staron，“类噪声对持续测试用例选择的影响：基于工业数据的受控实验”，载于产品导向软件过程改进国际会议，第287–303 页，Springer，2020 年。[40] J. Devlin、M.-W. Chang、K.Lee和K. Toutanova，“BERT：用于语言理解的深度双向变换器预训练”，*arXiv* 预印本*arXiv:1810.04805*，2018 年。[41] I. H. Witten 和 E. Frank，“数据挖掘：实用的机器学习工具与技术及 Java 实现”，*ACMSigmod Record*，第 31卷，第 1 期，第 76–77页，2002 年。[42] A.G. Malishevsky、J. R. Ruthruff、G.Rothermel 和S.Elbaum，“成本感知的测试用例优先级排序”，技术报告，内布拉斯加大学林肯分校技术报告 TR-UNL-CSE-2006-0004，2006 年。[43] D. Elsner、F. Hauer、A.Pretschner 和S.Reimer，“对持续集成中回归测试优化可直接获取信息的实证评估”，载于第*30* 届 *ACMSIGSOFT* 软件测试与分析国际研讨会论文集，第491–504 页，2021 年。[44] Y. Yang、X. Xia、D. Lo、T. Bi、J.Grundy 和 X. Yang，“软件工程中的预测模型：挑战与机遇”，*ACM* 软件工程与方法学汇刊（*TOSEM*），第 31 卷，第 3 期，第 1–72 页，2022 年。