软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(7):2911−2935 [doi: 10.13328/j.cnki.jos.007582] [CSTR: 32375.14.jos.007582]

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

基于个性化联邦学习的跨项目软件缺陷预测方法*

刘子扬[1], 祝 义[1,2], 周 湘[1], 李建豪[1], 袁春鸿[3], 郝国生[1]

[1](江苏师范大学 计算机科学与技术学院, 江苏 徐州 221116)

[2](高安全系统的软件开发与验证技术工业和信息化部重点实验室 (南京航空航天大学), 江苏 南京 211106)

[3](Faculty of Control Systems and Robotics, Saint Petersburg National Research University of Information Technologies, Mechanics and Optics, St. Petersburg 197101, Russia)

通信作者: 祝义, E-mail: zhuy@jsnu.edu.cn

摘 要: 针对跨项目软件缺陷预测中数据隐私与项目异构性的双重挑战, 提出一个名为 PRIDE-SDP 的框架. 该框架的核心贡献在于深度融合了 3 种关键技术: 采用个性化联邦学习范式为每个异构项目定制专属预测模型, 集成提供严格数学保障的 (ε, δ)-差分隐私机制保护数据不出本地, 并设计了一个专用的时间-上下文融合网络 (temporal-contextual fusion network, TCFN) 以高效地捕捉软件度量特征. 在覆盖 27 个开源项目和 3 个企业项目的 6 个数据集组上的实验结果验证了该框架的有效性: 与先进的跨项目缺陷预测基线相比, PRIDE-SDP 平均 *AUC* 提升 10.7%, *F*1-*score* 提升 7.3%; 在企业数据集上表现更加突出, 相较于所有先进的基线方法, *MCC* 平均提升 45.2%, *Effort*@20% 平均提升 29.5%, *F*1-*score* 平均提升 35.4%. 同时, 框架在提供较强隐私保障时, 其平均性能保持率仍能达到最佳性能的 98% 以上, 并且在成员推理攻击实验中能将攻击者的攻击准确率平均降低超过 36%. 实验结果表明, PRIDE-SDP 在保持高性能的同时, 有效兼顾了隐私保护与个性化适应能力.

关键词: 软件缺陷预测; 个性化联邦学习; 差分隐私; 跨项目缺陷预测

中图法分类号: TP311

中文引用格式: 刘子扬, 祝义, 周湘, 李建豪, 袁春鸿, 郝国生. 基于个性化联邦学习的跨项目软件缺陷预测方法. 软件学报, 2026, 37(7): 2911–2935. http://www.jos.org.cn/1000-9825/7582.htm

英文引用格式: Liu ZY, Zhu Y, Zhou X, Li JH, Yuan CH, Hao GS. Cross-project Software Defect Prediction Method Based on Personalized Federated Learning. Ruan Jian Xue Bao/Journal of Software, 2026, 37(7): 2911–2935 (in Chinese). http://www.jos.org.cn/1000-9825/7582.htm

Cross-project Software Defect Prediction Method Based on Personalized Federated Learning

LIU Zi-Yang[1], ZHU Yi[1,2], ZHOU Xiang[1], LI Jian-Hao[1], YUAN Chun-Hong[3], HAO Guo-Sheng[1]

[1](School of Computer Science and Engineering, Jiangsu Normal University, Xuzhou 221116, China)

[2](Key Laboratory of Safety-critical Software (Nanjing University of Aeronautics and Astronautics), Ministry of Industry and Information Technology, Nanjing 211106, China)

[3](Faculty of Control Systems and Robotics, Saint Petersburg National Research University of Information Technologies, Mechanics and Optics, St. Petersburg 197101, Russia)

Abstract: To address the dual challenges of data privacy and project heterogeneity in cross-project software defect prediction, this study proposes a framework named PRIDE-SDP. The core contribution of the proposed framework lies in the deep integration of three key

* 基金项目: 国家自然科学基金 (62077029, 62277030); 江苏省教育科学规划 (B-b/2024/01/47); 南京航空航天大学基本科研业务费科研基地创新基金 (NJ2020022); 江苏师范大学研究生科研创新项目 (2025XKT1437)
本文由“智能化基础软件可信构造和供应链安全”专题特约编辑王林章教授、司徒凌云研究员、钟浩研究员、 向剑文教授、张路教授推荐.
收稿时间: 2025-09-05; 修改时间: 2025-10-20; 采用时间: 2025-12-08; jos 在线出版时间: 2025-12-26
CNKI 网络首发时间: 2026-06-02

techniques. First, a personalized federated learning paradigm is adopted to customize dedicated prediction models for heterogeneous projects. Second, an (ε, δ)-differential privacy mechanism with rigorous mathematical guarantees is integrated to ensure that data remains local. Third, a dedicated temporal-contextual fusion network (TCFN) is designed to efficiently capture software metric features. Experiments conducted on six dataset groups covering 27 open-source projects and 3 enterprise projects validate the effectiveness of the proposed framework. Compared with state-of-the-art cross-project defect prediction baselines, PRIDE-SDP achieves an average improvement of 10.7% in *AUC* and 7.3% in *F*1-*score*. More pronounced performance gains are observed on enterprise datasets, where average improvements of 45.2% in *MCC*, 29.5% in *Effort*@20%, and 35.4% in *F*1-*score* are obtained over all advanced baseline methods. Meanwhile, under strong privacy guarantees, the framework's average performance retention rate remains above 98% of the optimal performance, and the attack accuracy in membership inference attack experiments is reduced by more than 36% on average. Experimental results demonstrate that PRIDE-SDP effectively balances high predictive performance with privacy protection and personalized adaptation capabilities.

Key words: software defect prediction (SDP); personalized federated learning; differential privacy; cross-project defect prediction

软件缺陷预测 (software defect prediction, SDP) 作为软件质量保证的核心技术之一, 旨在通过分析历史代码特征和缺陷模式, 识别潜在的有缺陷模块, 从而指导测试资源的合理分配[1]. 在现代软件开发环境中, 软件系统的规模和复杂性呈指数级增长, 手动检测所有可能的缺陷已变得不现实且成本高昂. Ostrand 等人[2]的经典研究表明, 软件缺陷通常遵循帕累托分布, 即大部分缺陷集中在少数代码模块中, 该分布特征为缺陷预测技术的发展奠定了重要理论基础, 同时表明, 精准的缺陷预测能够针对性优化测试资源分配, 显著提升测试效率.

近年来, 机器学习和深度学习技术在软件缺陷预测领域得到了广泛应用. Nam 等人[3]提出了基于迁移学习的缺陷预测方法, 通过扩展 TCA 显著提升了跨项目预测性能. Wang 等人[4]提出的 DeepLearning-based 缺陷预测方法, 通过自动学习代码的抽象语义特征, 在多个基准数据集上达到了 state-of-the-art 的性能. Li 等人[5]的研究进一步验证了深度学习模型在处理高维软件度量特征方面的优势.

尽管单项目内的缺陷预测技术已相对成熟, 但跨项目缺陷预测仍面临显著挑战. He 等人[6]的大规模实证研究发现, 传统的跨项目缺陷预测方法的 *F*1-*score* 普遍比项目内预测低. 这种性能退化主要源于不同项目间的异构性, 包括编程语言差异、开发流程不同、代码风格各异等因素. Turhan 等人[7]的研究通过分析大量开源项目, 证实了项目间数据分布差异是影响跨项目预测性能的关键因素.

联邦学习作为一种新兴的分布式机器学习范式, 为解决数据隐私和异构性问题提供了新的途径. McMahan 等人[8]首次系统性地提出了联邦学习的概念和 FedAvg 算法. Li 等人[9]进一步提出了 FedProx 算法, 通过引入近端项来处理异构客户端环境下的模型训练问题. Yang 等人[10]的综述文章全面总结了联邦学习的发展现状和面临的挑战. Lin 等人[11]的反馈式调试方法 Microbat 通过轻量级用户反馈指导调试过程, 这种人机交互的思路对于联邦学习中的模型优化具有借鉴意义.

个性化联邦学习 (personalized federated learning) 作为联邦学习的重要发展方向, 旨在为每个参与方提供定制化模型. Dinh 等人[12]提出的 pFedMe 算法, 通过双层优化框架实现了全局知识共享与个性化模型训练的平衡. Wang 等人[13]提出的 FedNova 算法, 通过归一化聚合机制缓解了异构客户端在本地更新步数不一致条件下引入的优化偏差, 为异构环境下的联邦模型聚合提供了理论保障.

在隐私保护方面, 差分隐私 (differential privacy) 已成为机器学习中隐私保护的黄金标准. Dwork[14]建立了差分隐私的理论基础. Abadi 等人[15]提出了深度学习中的差分隐私训练方法. Wei 等人[16]将差分隐私机制引入联邦学习训练过程, 为联邦学习中的隐私保护机制设计提供了重要理论依据.

本文旨在填补现有研究的空白, 为跨项目软件缺陷预测提供一个兼顾隐私保护、异构性适应和模型性能的综合性解决方案. 本文的主要贡献总结如下.

(1) 提出一个新颖的个性化联邦学习框架. 本文提出了 PRIDE-SDP (personalized privacy-preserving federated software defect prediction) 框架. 该框架专为软件缺陷预测场景设计, 通过个性化联邦学习范式解决跨项目协作中的数据隐私和项目异构性双重挑战, 为工业界实际应用提供新的范式.

(2) 设计一个专用的混合模型. 本文设计了一种名为时间-上下文融合网络 (temporal-contextual fusion network, TCFN) 的深度神经网络模型适配 PRIDE-SDP 框架. 该模型融合了 LSTM 的时序编码能力与 Transformer 的全局

注意力机制, 通过串行级联策略模拟软件缺陷产生的“演化-交互”因果逻辑. 该架构能够捕获高维软件度量数据中复杂的上下文交互关系和时序依赖性.

(3) 深度融合差分隐私与个性化算法. 本文将 (ε, δ)-差分隐私机制与个性化联邦学习算法深度结合. 通过引入近端项约束和动态噪声管理策略, 在提供严格数学隐私保障的同时, 实现了全局知识共享与本地模型定制化之间的有效平衡.

(4) 在大规模学术界开源数据集进行测试, 并在实际工业项目中进行测试以论证本文所提框架在工业界中实际应用的可能性.

1 研究动机

在当今企业软件开发环境中, 多项目协同开发已成为常态[17], 但项目间的数据隔离现象却严重阻碍了跨项目软件缺陷预测技术的实际应用. 如图 1 所示, 企业内部的多个项目组虽然面临软件缺陷预测的共同需求, 但由于组织架构、安全策略和合规要求的限制, 各项目组无法直接共享代码和缺陷数据, 形成明显的“数据孤岛”效应. 该现象不仅制约了各项目组对跨项目历史缺陷数据的有效利用, 也导致传统集中式机器学习方法难以在工业实践中部署.

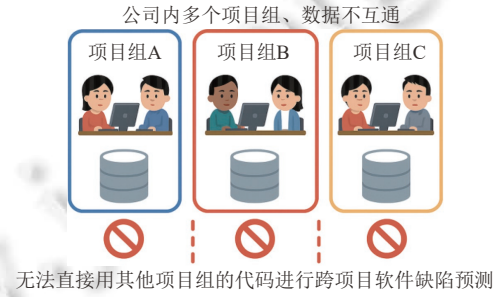


图 1 数据孤岛场景图

这一问题的紧迫性在工业界尤为突出. 正如李书缘等人[18]所指出的, 日益严格的隐私安全保护要求, 使得分散异构的多方数据拥有方之间无法直接共享数据, 严重制约了跨组织乃至跨项目的数据协同分析能力. 更为关键的是, 软件开发中普遍存在的强代码所有权文化会形成团队间的数据与知识壁垒[19]. 与此同时, 全球数据保护法规的日益严格进一步加剧了这一挑战. 在欧盟《通用数据保护条例》(GDPR)、《网络数据安全管理条例》等条例规定下, 软件源代码及缺陷数据的处理和共享必须满足严格的合规要求. Pasquale 指出, 大规模数据系统中产生的行为数据与元数据在聚合和推断过程中可能暴露高度敏感的隐私信息, 从而引发潜在的法律与合规风险[20]. 该观点表明, 在跨项目软件数据协作场景下, 企业需要更加审慎地评估数据共享与集中处理机制的合规性与风险边界.

然而, 跨项目协作的价值却是显而易见的. 文献 [21,22] 研究表明, 直接的跨项目预测存在性能问题, 但如果能够有效融合多项目知识, 缺陷预测的召回率可有一定提升. Herbold 等人[23]提出的 CPDP 方法基准测试研究表明, 对于跨项目缺陷预测有必要使用多个数据源和指标. 这些研究结果展示出跨项目协作具有巨大的技术价值和商业价值, 且亟需新的技术范式来突破现有障碍. 更为重要的是, 工业界对此类技术的需求日益迫切, 特别是在人工智能和机器学习技术快速发展的背景下, 数据驱动的软件质量保证方法已成为行业共识, 数据孤岛成为制约跨项目缺陷预测技术实际部署的关键障碍.

个性化联邦学习的出现为解决这一困境提供了新的可能性. 联邦学习概念首次被提出时, 就展示出其在保护数据隐私的同时实现协作学习的巨大潜力. 进一步被提出的个性化联邦学习方法, 通过为每个参与方维护个性化模型, 有效解决了异构环境下的模型适配问题. 这种技术路径与软件缺陷预测的实际需求高度契合: 各项目组可以在不暴露敏感数据的前提下参与协作训练, 同时获得适合自身项目特征的定制化预测模型.

然而, 将个性化联邦学习直接应用于软件缺陷预测仍面临诸多技术挑战. 软件缺陷数据具有明显的类别不平衡特征, 传统的联邦学习算法往往无法有效处理这种数据分布. 此外, 软件工程数据的高维稀疏特性也对模型设计

提出了特殊要求. 更为关键的是, 现有的差分隐私机制在软件工程场景下的适用性和有效性仍需要深入验证.

正是基于这些现实需求和技术挑战, 本文提出 PRIDE-SDP 框架, 旨在通过个性化联邦学习和差分隐私技术的组合, 为企业内部的跨项目软件缺陷预测提供一个既满足隐私保护要求, 又能有效应对项目异构性的实用化解决方案. 本文在学术前沿技术探索的基础上, 针对工业界对安全高效缺陷预测的实际需求, 提出了可行的解决思路.

2 相关工作

跨项目软件缺陷预测作为软件工程领域的重要研究方向, 已吸引了众多学者的关注. 现有研究主要围绕数据选择与预处理、迁移学习方法、深度学习技术以及类不平衡处理等方面展开. 本节将从这些角度对相关工作进行系统梳理和分析.

2.1 数据选择与预处理方法

在跨项目缺陷预测中, 源项目的选择和数据预处理是影响预测性能的关键因素. Kanwar 等人[24]提出了一种混合选择方法, 结合协同过滤和基于内容的方法来生成候选项目, 并使用朴素贝叶斯分类器预测新项目与现有项目之间的关系. 该方法在 *F*1-*score* 和平均精度等指标上显著优于现有方法, 为源项目选择提供了新的思路. Li 等人[25]提出了 BiLO-CPDP 工具, 使用双层编程自动化 CPDP 模型的发现过程. 通过在上层优化传输学习器与分类器的组合, 在下层优化超参数设置, 该方法在多个数据集上超过了多种现有的 CPDP 方法. Singh 等人[26]提出了一个高效的实例选择算法 BDAC, 用于加速支持向量机的训练过程. 该算法能在不降低预测精度的情况下大幅减少训练数据集规模, 提高训练速度. Bai 等人[27]提出了 3 阶段的转移学习框架 3SW-MSTL, 通过源选择策略、转移技术和条件分布信息指导, 系统性地解决了源选择和多源数据利用问题. 实验结果表明该框架在多个基准数据集上超过了其他多源和单源的 CPDP 方法.

2.2 迁移学习与域适应方法

迁移学习技术在缓解项目间数据分布差异方面发挥了重要作用. Zhang 等人[28]提出了基于连接性无监督分类器的跨项目缺陷预测方法, 使用谱聚类处理数据异构性问题. 该方法在不需要标注数据的情况下, 能够有效处理源项目和目标项目之间的分布差异. Li 等人[29]提出了基于地标选择的核判别子空间对齐方法 LSKDSA. 通过选择源项目和目标项目的关键地标, 并通过核方法进行非线性映射, 该方法有效减小了数据分布差异, 增强了模型的判别能力. Jin[30]提出了基于领域适应学习和优化的方法 DA-KTSVMO, 使用核双支持向量机实现源项目和目标项目数据分布的匹配. Tang 等人[31]提出了 TSboostDF 方法, 结合权重采样和迁移学习技术, 在处理知识转移和类不平衡问题方面表现突出. Dai 等人[32]提出了动态转移学习方法, 通过改进的相关性对齐和内核方差匹配技术, 优化了源项目和目标项目的联合概率分布.

2.3 深度学习与神经网络方法

深度学习技术在软件缺陷预测中展现出强大的特征学习能力. Bal 等人[33]提出了结合丢弃正则化深度学习和独特匹配指标的方法. 通过 KS 检验和匈牙利算法进行特征匹配, 该模型在 *AUC*、召回率等指标上相较于现有方法有大幅提升. Pandey 等人[34]提出了基于深度信念网络和长短期记忆网络的 JITCP-Predictor, 在 *MCC* 和 *F-Measure* 等方面分别有提高. Chen 等人[35]提出了软件可视化和深度转移学习方法, 通过将源代码转化为图像并结合自注意力机制进行训练. 该方法能够直接使用程序的语义信息进行缺陷预测, 在跨项目场景中表现优越.

2.4 类不平衡处理技术

软件缺陷数据中普遍存在的类不平衡问题是影响预测性能的重要因素. Poon 等人[36]提出了基于可信度理论的朴素贝叶斯分类器, 通过引入可信度理论缩小源项目和目标项目之间的特征分布差异. Gong 等人[37]提出了结合最近邻嵌套分层和迁移成分分析的类不平衡学习方法 TCA+STr-NN. 该方法针对数据不平衡和源数据与目标数据之间的分布差异进行了改进, 在跨项目缺陷预测中取得了更好的 *AUC* 和召回率. Jing 等人[38]提出了改进的子类判别分析方法 ISDA, 结合了半监督传输组件分析来解决类不平衡问题. Ryu 等人[39]提出了多目标优化的朴素贝叶

斯学习方法, 通过最大化缺陷类检测概率、最小化假阳性率来优化整体性能.

2.5 集成学习与多源方法

集成学习技术在提高跨项目缺陷预测鲁棒性方面发挥了重要作用. Xia 等人[40]提出了 HYDRA 模型, 通过结合遗传算法和集成学习实现多重分类器组合. 该模型能够通过迭代过程调优多个分类器, 在 29 个数据集上超过了多种现有方法. Sharma 等人[41]提出了基于集成学习的框架 HIEL, 结合自适应加权多数投票策略优化测试资源分配. Tong 等人[42]提出了多源转移加权集成学习方法 MASTER, 通过度量每个源数据集的权重, 有效提取了来自多个源数据集的可转移知识.

2.6 联邦学习方法

随着数据隐私保护需求的日益增长, 联邦学习技术在软件工程领域受到越来越多的关注. Xia 等人[43]提出了 HRFL 框架, 通过单边选择算法和客户端置信度重加权机制处理异构缺陷预测中的标签噪声问题. Wang 等人[44]的 FedDPI 框架采用两阶段协同优化机制, 通过继承私有模型和差分感知协作算法解决跨项目数据异构性问题. Yang 等人[45]提出的 ALMITY 框架专门针对学术研究与工业应用之间的鸿沟, 设计了考虑数据规模、数据平衡度和少数类可学习性的参数聚合策略.

在隐私保护方面, Liu 等人[46]构建了联邦软件缺陷预测 (FedSDP) 基准, 并提出采用 Paillier 部分同态加密技术对模型参数进行加密, 以抵御基于参数的推理攻击.

此外, 基于执行对比与用户反馈的智能调试方法也逐渐应用于软件工程任务中. 例如, ERASE 通过双版本轨迹对齐实现高精度回归缺陷解释[47], 而 CLUSTER 则利用代码依赖传播有限反馈以优化可追溯性恢复[48].

2.7 现有方法的局限性分析

尽管跨项目缺陷预测研究已取得显著进展, 但仍存在若干关键局限.

一方面, 数据隐私保护问题尚未得到充分重视. 现有研究大多基于公开的学术数据集进行实验验证, 而缺乏在工业级真实数据集上的系统性应用与评估. 这种差距限制了相关方法在实际场景中的可推广性与可靠性.

另一方面, 异构性处理仍显不足. 虽然已有方法尝试通过特征变换、域适应等技术缓解项目间差异, 但多数模型依赖统一的全局策略, 难以针对不同项目的特定特征进行自适应优化. Herbold 等人[23]的基准测试研究指出, 现有 CPDP 方法的整体性能仍未达到可广泛应用于工业实践的水平.

此外, 类不平衡问题的研究深度仍然有限. 现有工作主要集中于采样策略与代价敏感学习, 而缺乏针对软件工程数据特征的专门化方法. Vescan 等人[49]的复现性研究进一步表明, 在不同特征选择策略下, 监督与无监督学习方法的相对性能存在显著差异.

总体来看, 现有研究缺乏统一的理论框架. 多数方法仅针对特定问题提出局部解决方案, 尚未形成能够综合考虑隐私保护、异构性适应与性能优化的系统化框架. 这种割裂的研究模式使得实际应用中往往需要组合多种技术, 增加了系统复杂度与维护成本.

为应对上述挑战, 本文提出 PRIDE-SDP 框架通过个性化联邦学习实现跨项目知识共享, 在保护数据隐私的前提下提升模型泛化能力. 同时, 引入差分隐私机制以提供严格的隐私保障, 为突破现有方法的局限性提供了一种新的技术路径.

3 研究方法

本节详细阐述 PRIDE-SDP 框架的技术方案. 该框架通过个性化联邦学习和差分隐私技术的组合, 在保护数据隐私的前提下实现跨项目软件缺陷预测. 如后文图 2 所示, PRIDE-SDP 框架采用分布式架构设计, 主要包含数据收集与预处理、客户端模型结构和个性化联邦学习这 3 个核心模块.

3.1 框架总体设计

PRIDE-SDP 框架基于联邦学习范式构建, 旨在解决企业内部多项目组间数据隔离导致的跨项目缺陷预测难

题. 与传统的集中式学习方法不同, 该框架允许各项目组在不共享原始数据的情况下协同训练缺陷预测模型. 整个系统由多个项目组客户端和一个中央协调服务器组成, 各客户端维护自身的数据和个性化模型, 通过联邦机制实现知识共享.

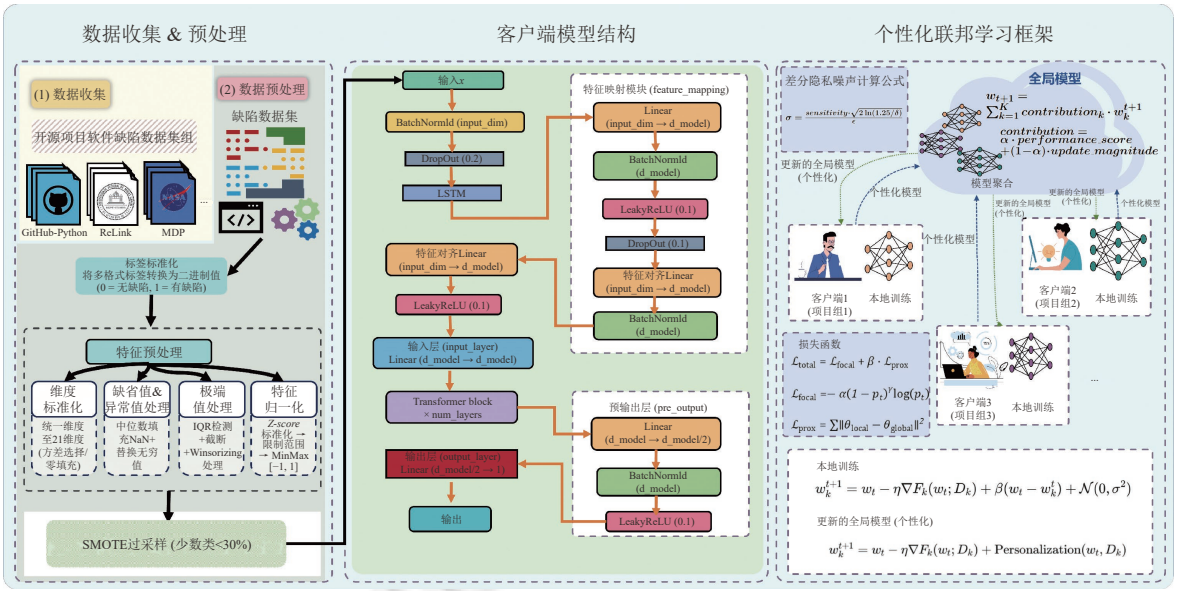


图 2 方法框架图

本文的核心贡献在于将个性化联邦学习与差分隐私保护机制深度融合. 传统的联邦学习方法通常采用单一的全局模型, 难以适应不同项目间的异构性特征. PRIDE-SDP 通过引入个性化机制, 为每个项目组维护定制化的模型参数, 同时保持全局知识的有效传递. 此外, 本文在模型参数传输过程中应用严格的差分隐私保护, 通过精心设计的噪声注入和隐私预算管理策略, 确保即使在模型参数被恶意分析的情况下, 也无法推断出原始训练数据的敏感信息.

整个训练过程遵循迭代优化的范式. 如算法 1 所示, 在每个通信轮次中, 服务器首先根据历史贡献度评估选择参与训练的客户端子集. 被选中的客户端使用本地数据进行个性化模型训练, 训练过程中应用焦点损失处理类不平衡问题, 并通过近端项约束保持与全局模型的适度偏离. 训练完成后, 各客户端对梯度添加自适应噪声以满足差分隐私要求, 并将模型更新发送给服务器. 服务器基于贡献度加权策略进行全局模型聚合, 更新后的全局模型被分发给各客户端. 各客户端根据新的全局模型自适应调整其个性化模型参数, 在保留本地数据特性的同时吸收全局知识, 为下一轮训练做好准备.

算法 1. PRIDE-SDP 训练算法.

输入: 客户端集合 C, 隐私预算 ε, 失败概率 δ, 聚合轮数 T;
输出: 全局模型 G, 个性化模型集合 P.

//初始化阶段
1. $G \leftarrow InitializeModel()$
2. $P \leftarrow \{\}, R \leftarrow \{\}$ //R 为每轮客户端贡献度记录
3. $\varepsilon_t \leftarrow \varepsilon_{\text{total}} \cdot \exp\left(-\lambda \cdot \frac{t}{T}\right)$ //每轮隐私预算分配
4. for $t = 1$ to T do
5. $S_t \leftarrow ClientSelection(C, R)$ //基于历史贡献度选择客户端

6. for each $client_i \in S_t$ parallel do
7. $D_i \leftarrow LocalDataPreprocess(client_i)$ //数据预处理
8. $\sigma_i \leftarrow AdaptiveNoiseScale(\varepsilon_t, \delta, D_i)$ //自适应噪声尺度
9. $\theta_i \leftarrow PersonalizedTraining(D_i, G, \sigma_i)$ //个性化训练
10. $r_i \leftarrow ContributionEvaluation(\theta_i, G, D_i)$ //多维度贡献度评估
11. $P[i] \leftarrow \theta_i$
12. $R[i][t] \leftarrow r_i$
13. end for

//动态权重聚合与模型更新阶段

14. $W \leftarrow ComputeAggregationWeights(R, t)$
15. $G \leftarrow WeightedAggregate(P, W, \sigma_t)$

//模型质量控制阶段

16. for each $client_i \in S_t$ do
17. $P[i] \leftarrow AdaptPersonalizedModel(P[i], G)$ //基于全局模型调整个性化模型
18. end for
19. end for
20. return G, P

3.2 数据收集与预处理

数据质量直接影响缺陷预测模型的性能, 因此 PRIDE-SDP 框架设计了完整的数据收集与预处理流水线. 各项目组从多个开源软件仓库 (如 GitHub、ReLink、MDP 等) 收集历史项目数据, 包括代码度量数据、面向对象度量、过程度量和缺陷标签信息. 代码度量数据涵盖圈复杂度、代码行数、函数数量等传统指标, 面向对象度量包括继承深度、子类数量、方法响应集合等反映软件结构复杂性的特征, 而过程度量则捕获代码变更频率、缺陷修复时间、开发者经验等动态演化信息. 数据预处理是确保模型训练效果的关键环节. 本文首先执行数据清洗操作, 去除重复记录并处理缺失值和异常值. 对于连续特征, 采用基于统计分布的方法检测并处理异常值, 而类别特征则使用众数填充策略处理缺失数据. 考虑到不同软件度量指标的量纲和数值范围存在显著差异, 本文采用 *Z-score* 标准化方法对特征进行归一化处理, 确保各特征在模型训练中具有相等的重要性权重. 特征工程是提升模型预测能力的重要手段. 针对历史缺陷数据稀疏的问题, 系统计算函数级、类级和文件级的缺陷率特征, 提供更丰富的历史信息. 同时, 通过组合多个基础度量构建复合指标, 如复杂度密度、耦合强度等, 以捕获更深层次的代码质量信息. 此外, 系统从代码提交历史中提取时序模式特征, 反映开发过程的动态变化规律. 软件缺陷数据普遍存在严重的类不平衡问题, 有缺陷模块的比例通常低于 30%. 为缓解类不平衡问题, 本文采用 SMOTE 过采样技术, 通过在少数类样本与其 K 近邻之间进行线性插值生成合成样本, 其数学表达式为:

$$x_i + \lambda \times (x_{nn} - x_i) \tag{1}$$

其中, x_i 为少数类样本, x_{nn} 为其 K 近邻样本, $\lambda \in [0,1]$ 为随机系数. 这种方法能够有效平衡正负样本分布, 同时避免简单复制造成的过拟合问题.

3.3 时间-上下文融合网络设计

PRIDE-SDP 框架在 Transformer 编码器的基础上, 设计了名为时间-上下文融合网络 (TCFN) 的混合神经网络架构的深度神经网络作为客户端预测模型. 与仅依赖单一结构的传统方法不同, TCFN 采用串行级联策略, 旨在同时捕捉软件开发过程中动态演化特征与静态代码度量间的复杂交互关系. 如图 3 所示, 该架构结合了长短期记忆网络 (LSTM) 序列编码、Transformer 以及全连接层的优势, 以充分挖掘数据中蕴含的深层模式.

模型的数据流转过程始于输入层. 一个输入的软件度量特征向量 x 首先经过批量归一化 (batch normalization)

和 Dropout 层, 以增强训练的稳定性和泛化能力. 随后, 数据被送入一个 LSTM 层, 该层专门用于捕获特征序列中可能存在的时序依赖关系. 经过 LSTM 初步处理后, 特征向量通过一个线性变换映射到高维表示空间:

$$H^{(0)} = xW^{\text{input}} + b^{\text{input}} \tag{2}$$

其中, $W^{\text{input}} \in \mathbb{R}^{d \times d_{\text{model}}}$ 为权重矩阵, d_{model} 为模型隐藏维度. 该映射将原始特征投影到更适合 Transformer 处理的表示空间中. Transformer 的核心组件是多头自注意力机制, 其数学表达为:

$$Attention(Q,K,V) = Softmax\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_2}}\right)V \tag{3}$$

其中, $Q = HW^{Q}, K = HW^{K}, V = HW^{V}$ 分别为查询、键和值矩阵. 通过多头注意力机制, 模型能够从不同子空间并行捕获度量指标间复杂的非欧几里得上下文交互. 为了缓解梯度消失问题, 每个 Transformer 块均引入了残差连接和层归一化机制. 最后, 经过特征对齐的全连接层 (采用 LeakyReLU 激活) 进一步提取高阶非线性特征, 并通过 Sigmoid 函数输出预测概率:

$$p = \sigma(h^{(\text{final})}W^{\text{output}} + b^{\text{output}}) \tag{4}$$

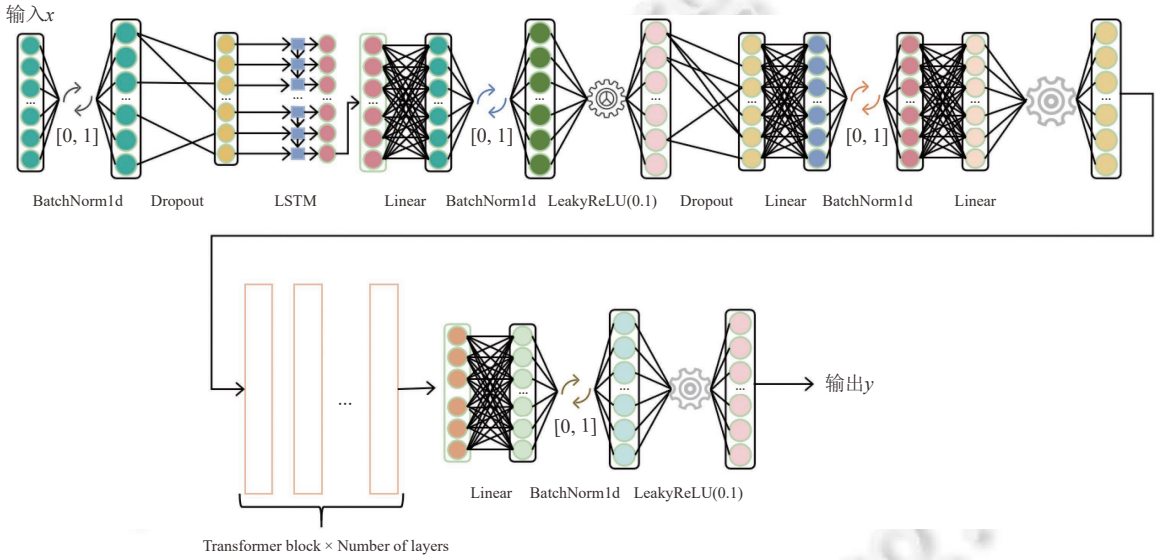


图 3 时间-上下文融合网络数据流转图

本文采用焦点损失来解决类不平衡问题, 其表达式为:

$$L_{\text{Focal}} = -\alpha(1-p_t)^{\gamma}\log(p_t) \tag{5}$$

其中, p_t 为真实类别的预测概率, α 和 γ 为调节参数. 焦点损失通过降低易分类样本的权重, 使模型更专注于难分类的样本, 从而提升对少数类的识别能力. 同时, 引入近端正则化项约束个性化模型参数与全局模型参数之间的 L2 距离, 在保持全局一致性的同时实现个性化学习:

$$L_{\text{prox}} = \sum |\theta_{\text{local}} - \theta_{\text{global}}|^2 \tag{6}$$

总损失函数为:

$$L_{\text{total}} = L_{\text{Focal}} + \beta \cdot L_{\text{prox}} \tag{7}$$

其中, β 为平衡系数, 控制个性化程度与全局一致性的权衡.

3.4 个性化联邦学习算法

个性化联邦学习是 PRIDE-SDP 框架的核心技术. 传统的联邦学习方法采用统一的全局模型, 在面对异构数据分布时往往性能不佳. 本框架受 pFedMe 算法启发设计了适用于软件缺陷预测的个性化联邦学习方案, 通过近端

正则化实现全局知识共享与个性化适应的平衡. 算法的核心思想是为每个客户端维护两套参数: 全局参数和个性化参数. 全局参数通过联邦聚合机制在所有客户端间共享, 捕获跨项目的通用知识; 个性化参数则根据本地数据特征进行调整, 适应特定项目的数据分布特性. 在每个客户端的本地训练过程中, 参数更新遵循:

$$w_k^{t+1} = w_k^t - \eta \nabla F_k(w_k^t; D_k) + \beta(w_k^t - w_k^*) + \mathcal{N}(0, \sigma^2) \tag{8}$$

其中, w_k^* 为个性化参数, η 为学习率, β 为正则化系数, $N(0,\sigma^2)$ 为差分隐私噪声. 个性化参数的更新过程结合了梯度下降和正则化约束. 具体而言, 个性化参数通过如下公式进行更新:

$$w_k^{*,t+1} = w_k^t - \eta \nabla F_k(w_k^t; D_k) + \text{Personalization}(w_k^t, D_k) \tag{9}$$

其中, 个性化函数根据本地数据特征和模型性能动态调整参数. 这种设计使得模型既能从全局知识中受益, 又能适应本地数据的特殊性质. 服务器端的全局模型聚合采用加权平均策略, 权重根据各客户端的数据量确定. 然而, 简单的数据量加权可能导致数据质量较差的客户端对全局模型产生负面影响. 因此, 本文引入了基于模型性能的贡献度评估机制, 贡献度综合考虑性能得分和更新幅度两个因素, 其中性能得分基于验证集准确率计算, 更新幅度通过参数变化量度量. 为优化通信效率和模型质量, 本文设计了动态客户端选择策略. 在每个通信轮次中, 服务器根据客户端的贡献度和数据质量计算选择概率, 优先选择高质量的客户端参与训练. 同时, 本文引入多样性约束, 确保选中的客户端具有足够的数据分布多样性, 避免模型偏向特定类型的项目.

3.5 差分隐私保护机制

隐私保护是 PRIDE-SDP 框架的重要特性, 本文采用差分隐私技术为联邦学习过程提供严格的数学隐私保障. 差分隐私通过在模型参数或梯度中添加校准噪声来保护个体数据的隐私, 即使攻击者获得了模型参数, 也无法推断出训练数据中任何个体的具体信息. 本框架在梯度级别实施差分隐私保护. 在每个客户端完成本地训练后, 对梯度信息进行处理再发送给服务器. 首先执行梯度裁剪操作, 将单个样本梯度的 L2 范数限制在预设阈值内, 即:

$$\tilde{g}_i = g_i \cdot \min\left(1, \frac{C}{|g_i|_2}\right) \tag{10}$$

其中, C 为裁剪阈值, g_i 为第 i 个样本的梯度. 梯度裁剪不仅有助于保护隐私, 还能提升训练稳定性, 防止异常样本对模型产生过大影响, 在梯度裁剪的基础上, 向聚合梯度中添加高斯噪声, 即:

$$\hat{g} = \frac{1}{|B|}\left(\sum_{i \in B} \tilde{g}_i + N(0, \sigma^2 C^2 I)\right) \tag{11}$$

其中, B 为训练批次. 噪声标准差 σ 根据隐私预算 ε 和失败概率 δ 计算, 具体公式为:

$$\sigma = \frac{C\sqrt{2\ln(1.25/\delta)}}{\varepsilon} \tag{12}$$

该设计确保了差分隐私的严格数学保证. 为平衡隐私保护强度与模型性能, 本文设计了自适应隐私预算分配策略. 在训练初期, 模型参数变化较大, 可以承受较强的噪声; 而在训练后期, 模型接近收敛, 需要较小的噪声以保持性能. 因此, 隐私预算按照如下公式进行分配:

$$\varepsilon_t = \varepsilon_{\text{total}} \cdot \exp\left(-\lambda \cdot \frac{t}{T}\right) \tag{13}$$

其中, t 为当前轮次, T 为总训练轮次, λ 为衰减系数.

此外, 本文引入了收敛感知的隐私预算调整策略: 当检测到模型接近收敛时, 适当增加噪声注入以提升最终性能. 为了在多轮训练中精确量化隐私保护强度, 框架采用了基于高级组合定理的隐私预算消耗追踪机制. 该机制将每轮训练视为一次满足 $(\varepsilon_t, \delta_t)$-差分隐私的查询, 并利用高级组合定理计算 T 轮后的累积隐私损失 $\varepsilon_{\text{total}}$ 和失败概率 δ_{total}. 本文采用以下公式对累积隐私损失进行估算:

$$\varepsilon_{\text{total}} = \sqrt{2k\ln(1/\delta)}\sigma + k\sigma(\mathrm{e}^{\sigma} - 1) \tag{14}$$

其中, k 为当前已执行的训练轮数, σ 为每轮添加的高斯噪声标准差. 框架在每轮训练结束后计算 $\varepsilon_{\text{total}}$, 并将其与预设的全局隐私预算 ε 进行比较. 一旦 $\varepsilon_{\text{total}}$ 接近或超过 ε, 系统将自动调整后续轮次的噪声规模或提前终止训练, 从

而确保在整个训练过程中始终满足严格的差分隐私要求.

3.6 算法实现与优化

PRIDE-SDP 框架的实现涉及多个组件的协调配合. 主算法控制整个训练流程, 包括客户端选择、本地训练、差分隐私保护和全局聚合等步骤. 在每个通信轮次开始时, 服务器根据贡献度评估和多样性约束选择参与训练的客户端子集. 被选中的客户端使用本地数据执行个性化训练, 训练过程中应用组合损失函数处理类不平衡和模型一致性约束问题. 本地训练算法在每个客户端独立执行, 包含多个训练 epoch. 每个 epoch 内, 算法遍历本地数据集的所有批次, 计算损失函数和梯度, 然后应用梯度裁剪和参数更新. 训练完成后, 客户端将模型参数发送给差分隐私保护模块进行处理, 差分隐私保护算法负责为模型参数添加校准噪声. 算法首先根据当前轮次和隐私预算计算噪声标准差, 然后为每个模型参数添加独立的高斯噪声. 处理后的参数被发送给服务器进行全局聚合. 服务器端的聚合算法收集所有客户端的更新, 根据贡献度权重计算加权平均, 生成新的全局模型参数. 更新后的全局参数被分发给各客户端, 指导下一轮的个性化训练. 整个过程持续迭代, 直到模型收敛或达到预设的训练轮次. 为提升算法的实用性, 系统还实现了多项优化技术. 通信压缩技术通过量化和稀疏化减少客户端与服务器间的通信开销. 异步更新机制允许不同客户端以不同的速度进行训练, 提升系统的鲁棒性. 容错机制处理客户端离线或通信失败的情况, 确保训练过程的连续性. 这些优化措施使得 PRIDE-SDP 框架能够在实际的分布式环境中稳定运行, 为跨项目软件缺陷预测提供可靠的技术支撑.

4 实 验

4.1 数据集

为验证本文所提方法的有效性, 本文在公开缺陷数据集上进行实验, 包括 PROMISE、MDP、AEEEM、GitHub-Python 和 ReLink. 数据集涉及 C、C++、Python、Java 等编程语言. 表 1 给出了数据集所对应的详细项目信息. MDP 数据集来源于美国国家航空航天局, 每个项目数据集由 40 个度量指标组成, 本文为补充其他实验数据集不涉及 C/C++的情况, 故仅选择 C/C++项目的数据. GitHub-Python 包括 3 个 Python 项目, 即 CoreFX、Django 和 Nova, 每个项目具有 29 个度量指标. ReLink 包括 3 个开源项目, 每个项目 ReLink 中的数据集有 26 个复杂性指标. AEEEM 中的每个项目数据集由 61 个软件指标组成. PROMISE 由多个开源 Java 项目组成, 其中每个项目数据集包括 20 个类级软件指标.

表 1 实验数据集

分组	项目	样本数量	度量数量	缺陷率 (%)
MDP[50]	CM1	505	40	10
	MW1	403		8
	PC1	1 107		7
	JM1	10 878		19
	KC1	2 107		15
	MC1	9 466		0.7
GitHub-Python[51]	CoreFX	26 627	29	6.91
	Django	26 360		42.64
	Nova	26 313		44.34
ReLink[52]	Apache	194	26	50.52
	Safe	56		39.81
	ZXing	399		29.57
AEEEM[53]	EQ	324	61	39.81
	JDT	997		20.66
	Lucene	691		9.26
	Mylyn	1 862		13.16
	PDE	1 497		13.96

表 1 实验数据集 (续)

分组	项目	样本数量	度量数量	缺陷率 (%)
PROMISE[54]	ant-1.3	125	20	16
	camel-1.6	965		19.48
	ivy-2.0	352		11.36
	jedit-4.1	312		25.32
	log4j-1.2	205		92.2
	poi-2.0	314		11.78
	prop-6	660		10
	synapse-1.2	269		31.97
	tomcat	858		8.97
	Xalan-2.4	723		15.21
EnterpriseSDP	Software1	1 247	35	18.2
	Software2	2 063		24.7
	Software3	856		13.1

为补充工业界实际应用场景的验证, 本文还收集了来自北京某科技公司的企业级软件缺陷数据集 EnterpriseSDP. 该数据集包含 3 个由同一开发工作室开发的企业管理系统软件项目, 分别命名为 Software1、Software2 和 Software3. 这 3 个软件项目均采用 Java 语言开发, 面向企业内部管理和业务流程自动化, 涵盖了人力资源管理、财务管理和项目管理等核心业务模块. 每个项目数据集包含 35 个代码复杂度指标, 涵盖了传统的 Halstead 度量、McCabe 循环复杂度、面向对象设计度量等维度. 与开源数据集相比, 企业数据集具有更严格的代码规范和更完整的测试覆盖, 为验证本文方法在工业环境中的实际效果提供了重要的实证基础.

4.2 评价指标

在跨项目软件缺陷预测的联邦学习框架中, 由于不同项目的缺陷分布存在显著差异, 且通常呈现类别不平衡特征 (缺陷样本通常远少于非缺陷样本), 传统的单一准确率指标难以全面反映模型的预测性能. 因此, 本文采用多维度评估体系来综合评估联邦学习模型在跨项目软件缺陷预测任务中的性能表现.

具体而言, 选择以下 8 个评估指标的原因如下: (1) *AUC* (area under curve) 作为主要评估指标, 其不受类别分布影响的特性使其特别适用于评估跨项目场景下的模型泛化能力; (2) *Precision*、*Recall* 和 *F*1-*score* 构成经典的信息检索评估三角, 分别从查准率、查全率和综合性能角度评估模型对缺陷的识别能力; (3) *Specificity* 补充评估模型对非缺陷代码的正确识别能力, 与 *Recall* 共同反映模型的全面性能; (4) *G-mean* (几何均值) 和 *Balance* (平衡准确率) 专门针对不平衡数据集设计, 能够平衡考虑正负样本的分类性能; (5) *Accuracy* 作为基础指标提供整体分类性能的参考基准. 这种多指标评估体系能够从不同维度全面、客观地评估联邦学习模型在跨项目软件缺陷预测任务中的有效性.

设真正例为 *TP* (true positive), 真负例为 *TN* (true negative), 假正例为 *FP* (false positive), 假负例为 *FN* (false negative). 各评估指标的计算公式如下.

(1) *AUC* (曲线下面积)

$$AUC = \int_0^1 TPR \, \mathrm{d}\,(FPR) \tag{15}$$

其中, $TPR = \dfrac{TP}{TP+FN}$, $FPR = \dfrac{FP}{FP+TN}$.

(2) *Accuracy* (准确率)

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{16}$$

(3) *Precision* (精确率)

$$Precision = \frac{TP}{TP+FP} \tag{17}$$

(4) *Recall* (召回率)

$$Recall = \frac{TP}{TP + FN} \tag{18}$$

(5) *F*1-*score* (*F*1 值)

$$F1\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN} \tag{19}$$

(6) *Specificity* (特异性)

$$Specificity = \frac{TN}{TN + FP} \tag{20}$$

(7) *G-mean* (几何均值)

$$G\text{-}mean = \sqrt{Recall \times Specificity} = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \tag{21}$$

(8) *Balance* (平衡准确率)

$$Balance = \frac{Recall + Specificity}{2} = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) \tag{22}$$

4.3 基线方法

对于跨项目缺陷预测, 本文采用 LR、RF、eCPDP、CodeBERT-PT、MNAFM 等方法作为基线模型; 基线模型涉及传统机器学习方法、集成学习方法、联邦学习方法、跨项目软件缺陷预测的 state-of-the-art (SOTA) 方法及预训练模型.

LR[55]: 一种经典的线性分类模型. 它通过学习输入特征与缺陷概率之间的逻辑函数关系来进行预测.

DPDF[56]: 一种深度森林模型. 该模型充分利用集成学习和深度学习, 通过使用新的级联策略来识别更重要的缺陷特征, 该策略将随机森林分类器转换为逐层结构.

eCPDP[57]: 一种跨项目缺陷预测技术. 它利用奇异值分解 (SVD) 进行迁移学习, 旨在无需完整的历史目标数据即可在单元测试阶段对缺陷进行预测.

CodeBERT-PT[58]: 一种基于预训练语言模型 CodeBERT 的方法. 它通过在原始代码上进行微调来学习代码的深层语义特征, 并直接预测代码模块是否包含缺陷.

MNAFM[59]: 一种异构缺陷预测方法. 它将软件度量转化为图像, 并利用基于注意力机制的元网络 (meta-network) 来匹配源项目和目标项目的特征, 从而进行缺陷预测.

AC-GAN[60]: 一种基于生成式对抗网络的跨项目缺陷预测方法, 通过 GAN 网络的对抗训练来缩小源项目和目标项目之间的数据分布差异, 从而提升跨项目缺陷预测的性能.

ALMITY[45]: 一种改进聚合策略的联邦学习方法. 它在联邦学习框架下仅共享模型更新与数据分布信息、不传原始数据, 并在参数聚合时联合考虑“数据规模-数据平衡度-少数类可学习性”这 3 个属性, 以提升类不平衡数据上的模型性能.

Fed-OLF[61]: 一种联邦过采样学习框架. 它结合基于扩散模型的本地数据过采样方法 (TabDiT) 和基于局部信息熵的加权聚合策略 (LEW), 在保护数据隐私的同时, 解决多方协作下的软件缺陷数据不平衡问题并提升模型预测性能.

4.4 实验方法

本文使用来自 MDP、GitHub-Python、ReLink、PROMISE、AEEEM 和 Enterprise 的 30 个项目作为实验数据集. 跨项目缺陷预测实验采用留一 (leave-one-out) 法策略, 模拟真实的跨组织数据协作场景. 在该实验中, 数据集组内的每个项目轮流作为测试集, 其余项目作为训练客户端参与联邦学习. 具体而言, 对于包含 n 个项目的数据集组, 实验将进行 n 轮, 第 i 轮中项目 i 作为测试集, 项目 1 至 i−1 和 i+1 至 n 作为训练客户端. 实验环境配置为: 华为 Ascend 910B GPU, EulerOS 2.0 操作系统, PyTorch 1.11.0 深度学习框架, 结合 torch-npu 加速库和 CANN-6.3.RC2

计算架构进行模型训练和推理. 网络结构参数方面, Transformer 模型维度设置为 32, 多头注意力机制采用 2 个注意力头, Transformer 层数设为 1 层. 训练参数配置中, 客户端数量设定为 5 个, 本地训练步数为 3 步, 批次大小为 32, 全局训练轮数为 50 轮, 学习率采用 1E−3, 并使用 AdamW 优化器. 差分隐私参数设置中, 跨项目实验的隐私预算 ε 设为 5.0, 隐私失败概率 δ 设为 1E−5, 噪声乘数为 0.8, 梯度裁剪范数阈值为 1.0, 采用高斯噪声机制保护客户端隐私. 联邦学习参数包括个性化系数 β 设为 0.3, 模型性能权重 α 设为 0.8, 客户端选择比例为 0.6. 损失函数参数方面, 采用 Focal Loss 处理类别不平衡问题, 其中, $\alpha = 0.25$, $\gamma = 2.0$; 正则化参数中, dropout 率在不同层设置为 0.1–0.2, 以防止过拟合. 本文的框架开源至: https://anonymous.4open.science/r/PRIDE-SDP-4FE/.

5 结果分析与讨论

为评估 PRIDE-SDP 方法的有效性, 我们研究了以下 5 个问题.

● RQ1: PRIDE-SDP 相较于其他跨项目缺陷预测方法是否更好?

为全面评估 PRIDE-SDP 框架的有效性, 本文选择了 8 种具有代表性的跨项目缺陷预测方法作为基线进行对比实验. 这些方法涵盖了该领域的主要技术路径: 传统机器学习方法 (LR)、深度学习方法 (DPDF)、迁移学习方法 (eCPDP)、预训练模型方法 (CodeBERT-PT)、基于注意力机制的异构预测方法 (MNAFM)、生成对抗网络方法 (AC-GAN) 以及联邦学习方法 (ALMITY、Fed-OLF). 选择这些基线方法的主要考虑因素包括: (1) 技术路径的多样性和代表性; (2) 方法的先进性和在相关领域的影响力; (3) 实现的可获得性以确保实验结果的可重复性. 实验结果表明, 在大多数情况下, PRIDE-SDP 框架的综合性能显著优于其他 8 种跨项目缺陷预测基线方法. 如表 2 所示, PRIDE-SDP 在 5 个不同的数据集组上, 尤其在 *AUC*、*F*1-*score* 和 *G-mean* 这 3 个关键评估指标上, 展现出了强大的竞争力和优越性, 证明了其在处理项目异构性方面的有效性.

表 2 各个跨项目缺陷基线方法在全部数据集组的对比结果

数据集组	评估指标	PRIDE-SDP	LR	DPDF	eCPDP	CodeBERT-PT	MNAFM	AC-GAN	Fed-OLF	ALMITY
MDP	*AUC*	**0.7584**	0.585	0.6365	0.5324	0.5674	0.5705	0.7134	0.7245	0.6443
	Accuracy	**0.6785**	0.5912	0.5609	0.4067	0.5767	0.5782	0.6445	0.6542	0.6258
	Precision	**0.7243**	0.6530	0.6955	0.4315	0.6234	0.6043	0.6889	0.6987	0.603
	Recall	**0.5855**	0.3793	0.2831	0.5742	0.5012	0.5771	0.5634	0.5634	0.5706
	*F*1-*score*	**0.6302**	0.4763	0.3473	0.4909	0.5557	0.5671	0.6198	0.6234	0.5863
	Specificity	0.7715	0.8026	**0.8388**	0.2392	0.6589	0.5793	0.7234	0.7451	0.7413
	G-mean	**0.6577**	0.5475	0.4659	0.3582	0.5747	0.5131	0.6384	0.6478	0.6503
	Balance	**0.6785**	0.5912	0.5609	0.4067	0.5801	0.6712	0.6434	0.6542	0.6559
GitHub-Python	*AUC*	0.6722	0.6569	0.602	0.5180	0.5756	**0.7278**	0.6234	0.6445	0.6663
	Accuracy	0.6231	0.6467	0.6437	0.5070	0.5685	**0.6772**	0.6089	0.6089	0.6394
	Precision	0.618	0.6505	0.6371	0.5029	0.5987	**0.6599**	0.6156	0.6234	0.6185
	Recall	0.5255	0.4859	0.5308	0.5334	0.4985	**0.6012**	0.5023	0.4987	0.5792
	*F*1-*score*	0.5428	0.5561	0.5748	0.4659	0.5440	**0.6289**	0.5533	0.5534	0.5982
	Specificity	**0.7933**	0.7794	0.7441	0.5150	0.6423	0.7412	0.7045	0.7123	0.738
	G-mean	0.5887	0.6148	0.6266	0.422	0.5659	**0.6675**	0.5949	0.5956	0.6538
	Balance	**0.6594**	0.6326	0.6374	0.5242	0.5704	0.6012	0.6034	0.6055	0.6586
ReLink	*AUC*	**0.711**	0.6688	0.6472	0.7031	0.6184	0.7006	0.6923	0.6834	0.6509
	Accuracy	0.6614	0.6653	**0.6669**	0.6586	0.6065	0.6591	0.6534	0.6445	0.6262
	Precision	0.6453	**0.7031**	0.6856	0.6621	0.6123	0.6226	0.6789	0.6567	0.6297
	Recall	**0.6242**	0.4981	0.5520	0.5625	0.5987	0.5969	0.5834	0.5978	0.5949
	*F*1-*score*	0.6245	0.5819	0.6069	0.6012	0.6054	0.6120	**0.6276**	0.6256	0.6118
	Specificity	0.6701	**0.8085**	0.7675	0.7540	0.6234	0.6260	0.7134	0.6889	0.7383
	G-mean	0.6313	0.6341	0.6498	0.6479	0.6109	0.6599	0.6451	0.6425	**0.6627**
	Balance	0.6472	0.6533	0.6597	0.6582	0.6111	0.6115	0.6484	0.6434	**0.6666**

表 2 各个跨项目缺陷基线方法在全部数据集组的对比结果 (续)

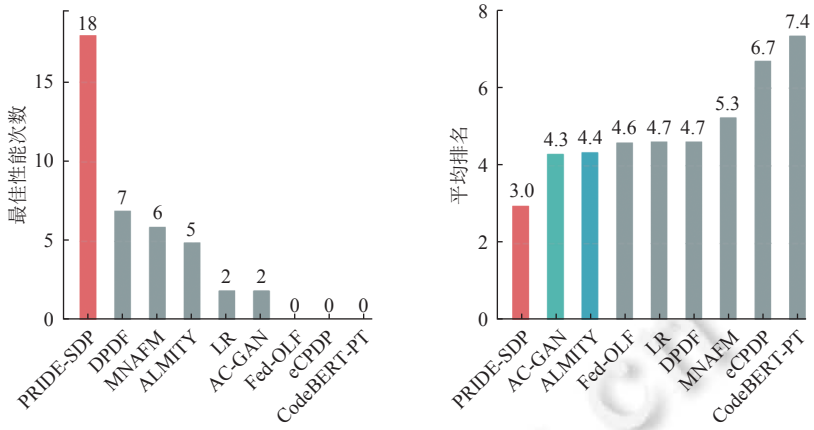
数据集组	评估指标	PRIDE-SDP	LR	DPDF	eCPDP	CodeBERT-PT	MNAFM	AC-GAN	Fed-OLF	ALMITY
AEEEM	*AUC*	**0.7123**	0.6795	0.7109	0.6893	0.6047	0.5764	0.6934	0.6789	0.6787
	Accuracy	0.6567	0.6427	0.6208	0.6462	0.6083	0.5721	0.6378	0.6245	**0.6604**
	Precision	0.6471	0.6763	**0.8069**	0.6579	0.6098	0.5639	0.6645	0.6445	0.6315
	Recall	**0.6308**	0.5106	0.2981	0.566	0.5934	0.5255	0.5823	0.5745	0.5729
	F1-score	**0.635**	0.5743	0.398	0.6052	0.6015	0.5414	0.6207	0.6067	0.6007
	Specificity	0.6751	0.7605	**0.9088**	0.7105	0.6156	0.6063	0.6834	0.6734	0.7563
	G-mean	0.647	0.6153	0.4871	0.6295	0.6044	0.5593	0.6309	0.6223	**0.6582**
	Balance	0.6529	0.6355	0.6034	0.6383	0.6045	0.5659	0.6328	0.624	**0.6646**
PROMISE	*AUC*	**0.7411**	0.7399	0.7404	0.4249	0.6091	0.7044	0.7189	0.7156	0.6713
	Accuracy	0.6779	0.6795	**0.6849**	0.4542	0.6099	0.6641	0.6734	0.6634	0.6528
	Precision	0.669	0.6911	**0.7277**	0.4372	0.6245	0.6655	0.6823	0.6734	0.6273
	Recall	**0.6825**	0.6033	0.5444	0.573	0.5823	0.6288	0.6567	0.6445	0.5951
	F1-score	0.6619	0.6388	0.6124	0.4867	0.6027	0.6369	**0.6694**	0.6587	0.6108
	Specificity	0.6718	0.6502	**0.8059**	0.3348	0.6387	0.6915	0.6789	0.6823	0.7206
	G-mean	**0.6692**	0.6678	0.6491	0.3848	0.6098	0.6480	0.6677	0.6632	0.6549
	Balance	**0.6772**	0.6267	0.6751	0.4539	0.6105	0.6602	0.6678	0.6634	0.6579

注: 加粗数字表示各评估指标在对应数据集上的最佳性能值

实验结果表明, PRIDE-SDP 框架在大多数数据集上展现出优异的综合性能. 在 MDP 数据集上, PRIDE-SDP 在 8 个评估指标中的 7 个指标上取得最佳性能, 表明该框架能够有效处理 MDP 数据集中源项目与目标项目之间的异构性问题, 实现较为准确的缺陷预测. 在 AEEEM 数据集上, PRIDE-SDP 同样表现出色, 在 *AUC* (0.7123)、*F1-score* (0.635) 和 *Recall* (0.6308) 等关键指标上取得最佳性能. 在 PROMISE 数据集上, PRIDE-SDP 在 *AUC* (0.7411)、*Recall* (0.6825)、*G-mean* (0.6692) 和 *Balance* (0.6772) 这 4 个指标上保持领先, 但 DPDF 在 *Accuracy* (0.6849)、*Precision* (0.7277) 和 *Specificity* (0.8059) 指标上表现更佳, AC-GAN 则在 *F1-score* (0.6694) 指标上取得最优结果. ReLink 数据集的实验结果呈现出类似的分化特征. PRIDE-SDP 在 *AUC* (0.711) 和 *Recall* (0.6242) 指标上保持最优, 但在其他指标上面临来自多个强基线的挑战. 其中, DPDF 在 *Accuracy* (0.6669) 指标上表现最佳, LR 在 *Precision* (0.7031) 和 Specificity (0.8085) 指标上占据优势, AC-GAN 在 *F1-score* (0.6276) 指标上领先, ALMITY 则在 *G-mean* (0.6627) 和 *Balance* (0.6666) 指标上表现突出. 在 GitHub-Python 数据集上, MNAFM 方法表现尤为突出, 在 *AUC* (0.7278)、*Accuracy* (0.6772)、*Precision* (0.6599)、*Recall* (0.6012)、*F1-score* (0.6289) 和 *G-mean* (0.6675) 等 6 个指标中均获得最佳性能, 全面超越了 PRIDE-SDP 的对应表现. 这一现象可能与 GitHub-Python 数据集的特有属性相关, 该数据集中的部分项目具有较高的缺陷密度. 尽管如此, PRIDE-SDP 在该数据集的 *Specificity* (0.7933) 指标上仍保持最优性能, 体现了其在特异性识别方面的相对优势.

如图 4 所示, 统计分析结果进一步验证了 PRIDE-SDP 在多数评估场景中的优越性能. 图 4(a) 展示了各方法在 40 个评估场景中获得最佳性能的频次分布. 结果显示, PRIDE-SDP 获得 18 次最佳性能, 占总评估场景的 45%, 远超其他所有基线方法. DPDF 获得 7 次最佳性能, MNAFM 获得 6 次最佳性能, ALMITY 获得 5 次最佳性能, 展现了在特定评估指标上的竞争优势. LR 和 AC-GAN 各获得 2 次最佳性能, 而 eCPDP、CodeBERT-PT 和 Fed-OLF 在所有评估场景中均未能取得最优表现. 这一频次分布清晰地反映了 PRIDE-SDP 在跨项目缺陷预测任务中的整体优势和稳定性.

图 4(b) 展示了各方法的平均排名情况, 该指标能够更全面地评估方法的综合性能稳定性. 平均排名的计算方法如下: 首先在每个数据集的每个评估指标上, 将所有 9 种方法按性能从高到低进行排序, 分别赋予 1–9 的排名值; 然后计算每种方法在全部 40 个评估场景中排名的算术平均值, 排名数值越小表示该方法的整体性能越优异. 结果表明, PRIDE-SDP 以 3.0 的平均排名位居首位, 显著优于其他方法. AC-GAN 和 ALMITY 的平均排名分别为 4.3 和 4.4, 位列第 2 梯队. Fed-OLF 以 4.6 的排名略优于 LR 和 DPDF. eCPDP 和 CodeBERT-PT 的平均排名分别为 6.7 和 7.4.



(a) 各方法获得最佳性能次数统计 (b) 各方法平均排名

图 4 方法性能统计分析图

这一平均排名结果从另一个维度证实了 PRIDE-SDP 不仅在单项指标上表现出色, 更在整体性能稳定性方面具有显著优势. 与最佳性能次数统计结果相结合, 可以看出 PRIDE-SDP 既能在多个评估场景中达到最优性能, 又能在其他场景中保持相对较好的排名位置, 体现了良好的泛化能力和鲁棒性. 相比之下, 某些方法虽然在特定场景下表现突出 (如 MNAFM 在 GitHub-Python 数据集上的优异表现), 但在整体平均排名上并未获得相应的优势, 这表明这些方法可能存在性能波动较大或适用场景相对有限的问题.

● RQ2: 不同隐私预算对 PRIDE-SDP 有何影响?

为了验证不同隐私预算对方法的精度影响, 实验测试了 5 种不同的隐私预算设置: $\varepsilon \in \{0.1, 0.5, 1.0, 5.0, 10.0\}$, 对应从强隐私保护到弱隐私保护的梯度. 差分隐私噪声按照高斯机制添加, 其中, 梯度噪声尺度为 $\sigma = \left(sensitivity \cdot \sqrt{2\ln(1.25/\delta)}\right)/\varepsilon$, 参数噪声为梯度噪声的 1%, 数据噪声为梯度噪声的 5%, δ 值固定为 1×10^{-5}. 客户端贡献度评估机制结合模型性能 (权重 0.6) 和参数更新幅度 (权重 0.4), 仅选择贡献度排名前 80% 的客户端参与模型聚合.

隐私预算 ε 的大小对 PRIDE-SDP 模型的性能有显著影响, 二者之间存在一种非线性的权衡关系. 总体而言, 随着隐私预算 ε 的增加 (即隐私保护强度减弱), 模型的预测性能会相应提升, 但当 ε 达到一定水平后, 性能提升的幅度会逐渐减小并趋于稳定.

如表 3 所示, 在最严格的隐私预算 ($\varepsilon = 0.1$) 下, 模型的性能受到显著抑制, 所有数据集组的 *AUC* 指标均在 0.42–0.56 之间, 接近甚至低于 0.5 的随机猜测水平. *F*1-*score* 也普遍较低, 例如在 GitHub-Python 上仅为 0.3359. 为实现极强的隐私保护, 模型需要注入大量噪声, 这严重干扰了有效的知识学习和迁移, 从而导致性能的大幅下降.

表 3 ε 在不同取值时各数据集组平均指标

隐私预算	数据集组	项目数	*AUC*	*Accuracy*	*Precision*	*Recall*	*F*1-*score*	*Specificity*
0.1	AEEEM	5	0.4283	0.4481	0.424	0.5328	0.4496	0.3745
	GitHub-Python	3	0.4591	0.4988	0.496	0.418	0.3359	0.5997
	MDP	6	0.4413	0.4948	0.3927	0.5489	0.4065	0.4406
	ReLink	3	0.4364	0.4718	0.277	0.4542	0.3419	0.4873
	PROMISE	10	0.5598	0.5047	0.5685	0.4554	0.3859	0.5548
0.5	AEEEM	5	0.5824	0.5114	0.5095	0.5581	0.492	0.4965
	GitHub-Python	3	0.6165	0.5695	0.598	0.2491	0.2926	0.8562
	MDP	6	0.7133	0.6676	0.7471	0.5199	0.5937	0.8154
	ReLink	3	0.5269	0.547	0.5096	0.3531	0.362	0.7045
	PROMISE	10	0.6079	0.5591	0.5649	0.577	0.5332	0.5637

表 3 ε 在不同取值时各数据集组平均指标 (续)

隐私预算	数据集组	项目数	*AUC*	*Accuracy*	*Precision*	*Recall*	*F1-score*	*Specificity*
1	AEEEM	5	**0.7166**	0.6399	0.6278	**0.6838**	0.6344	0.6035
	GitHub-Python	3	0.6371	**0.6267**	**0.6281**	0.4476	0.5197	**0.7759**
	MDP	6	0.716	0.6672	0.7193	0.564	0.6148	0.7704
	ReLink	3	0.7077	0.638	0.6094	0.6159	0.6116	0.6574
	PROMISE	10	**0.7542**	**0.6887**	0.6679	0.694	**0.6695**	**0.6834**
5	AEEEM	5	0.7123	**0.6567**	**0.6471**	0.6308	**0.635**	**0.6751**
	GitHub-Python	3	**0.6722**	0.6231	0.618	**0.5255**	0.5428	0.7113
	MDP	6	**0.7584**	**0.6785**	**0.7243**	**0.5855**	0.6302	**0.7715**
	ReLink	3	**0.711**	**0.6614**	**0.6453**	**0.6242**	**0.6245**	**0.6701**
	PROMISE	10	0.7411	0.6779	**0.669**	0.6825	0.6619	0.6718
10	AEEEM	5	0.7059	0.6473	0.6232	0.6721	0.6349	0.6241
	GitHub-Python	3	0.6422	0.6156	0.5943	0.5122	**0.5476**	0.7058
	MDP	6	0.7492	0.6714	0.6978	0.621	**0.6341**	0.7219
	ReLink	3	0.5702	0.5152	0.5671	0.5705	0.508	0.5163
	PROMISE	10	0.7518	0.6815	0.6557	**0.6997**	0.6691	0.6642

注: 加粗数字表示在对应隐私预算下各数据集上的最佳性能值

当隐私预算从 0.1 适度放宽至 1.0 时, 模型的各项性能指标均出现了大幅度的回升. 以 AEEEM 数据集组为例, *AUC* 从 0.4283 跃升至 0.7166, *F1-score* 从 0.4496 提升至 0.6344. 同样的趋势也体现在 MDP、PROMISE 等其他数据集组上. 当隐私预算 ε 超过 1.0 后, 性能的增长趋势明显放缓, 并逐渐趋于饱和. 对比 $\varepsilon=1.0$、$\varepsilon=5.0$ 和 $\varepsilon=10.0$, 可以看出各项性能指标的波动范围很小.

本文将综合性能指标 (composite performance index, *CPI*) 定义为:

$$CPI = w_1 \times AUC + w_2 \times F1 + w_3 \times Precision + w_4 \times Recall + w_5 \times Specificity \tag{23}$$

其中, 权重设置为: $w_1=0.3$, $w_2=0.25$, $w_3=0.15$, $w_4=0.15$, $w_5=0.15$. 隐私预算 ε 的 *CPI* 边际收益定义为:

$$MarginalBenefit(\varepsilon_i \to \varepsilon_{i+1}) = \frac{CPI(\varepsilon_{i+1}) - CPI(\varepsilon_i)}{CPI(\varepsilon_i)} \times 100\% \tag{24}$$

对于图 5 中展示的其他单项指标 (*AUC*、*F1-score*、*Precision*) 的边际收益分析, 采用与 *CPI* 相同的计算方式.

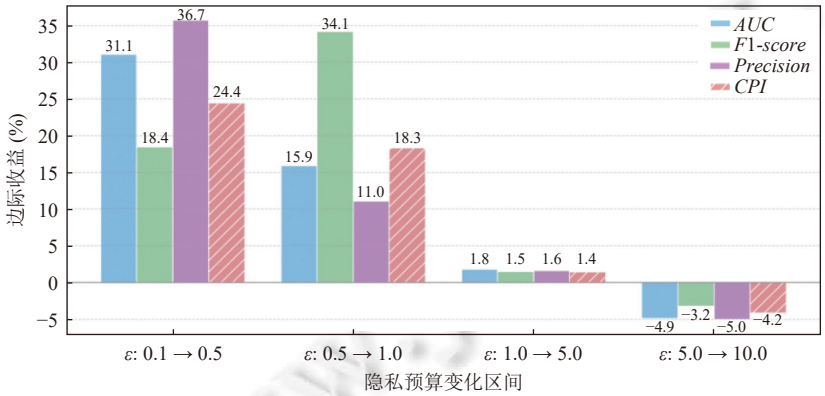


图 5 基于综合性能指标的隐私预算边际收益分析图

考虑到软件缺陷预测的多维性能要求, 本文构建了综合性能指标 (*CPI*) 来全面评估隐私预算的影响效果. *CPI* 整合了 *AUC* (权重 0.3)、*F1-score* (权重 0.25)、*Precision*、*Recall* 和 *Specificity* (各权重 0.15) 等核心指标. 需要说明的是, *Recall* 和 *Specificity* 未在图 5 中作为单项指标单独绘制, 而是已通过加权方式体现在 *CPI* 中. 如图 5 所示, 综合边际收益分析揭示了更清晰的性能-隐私权衡模式: $\varepsilon=1.0$–5.0 的边际收益呈现饱和趋势, 仅为 1.39%; 而当 ε

进一步增加至 [5.0, 10.0] 时, 边际收益转为显著的负值 (−4.17%). 这一显著的性能回退表明, 模型在 $\varepsilon = 5.0$ 处已达到综合性能峰值, 继续放宽隐私预算反而会引入过拟合或噪声失效风险, 从而验证了 ε 作为性能最优隐私预算设置的合理性.

为了直观地展示 PRIDE-SDP 在最优隐私预算 ($\varepsilon = 5.0$) 下的性能表现, 图 6 对所有项目的实验结果进行了多维度可视化分析.

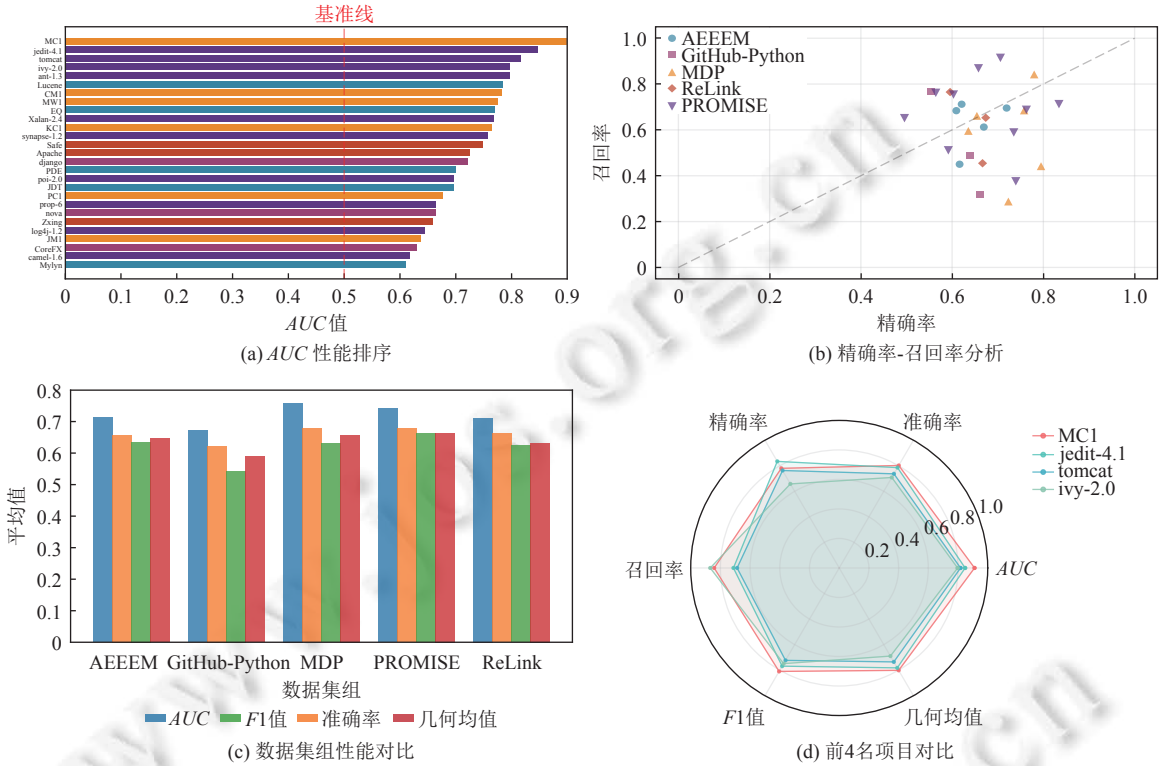


图 6 $\varepsilon = 5$ 时所有数据集性能表现可视化分析效果图

AUC 性能分布: 图 6(a) 展示了所有 27 个项目的 *AUC* 性能排序. 可以看出, 模型在绝大多数项目上都取得了远超随机猜测 ($AUC = 0.5$) 的性能, 其中多个项目的 *AUC* 值超过 0.8, 显示了强大的预测能力. 同时, 图中性能的显著差异也反映了不同软件项目间的固有异构性, PRIDE-SDP 在不同项目上表现出了差异化的适应性.

精确率与召回率平衡: 图 6(b) 进一步揭示了模型的分类特性. 大部分项目 (数据点) 分布在图的右上方区域, 表明模型在保证一定查准率的同时, 能够实现较高的查全率. 许多数据点位于对角线 ($y = x$) 上方, 说明模型倾向于优先保证高召回率, 这在缺陷预测任务中通常是期望的特性, 因为它有助于识别出更多的潜在缺陷.

跨数据集组的宏观性能: 图 6(c) 从数据集组的宏观视角对比了平均性能. 结果显示, 模型在 PROMISE、MDP 和 ReLink 等数据集组上取得了更优的综合性能 (以 *AUC* 和 *F*1 值为代表), 这可能与这些数据集组内部的项目同质性更高、联邦学习的协同效应更显著有关.

最优性能剖析: 图 6(d) 通过雷达图展示了性能排名靠前项目在多个评价指标上的表现. 可以看出, 这些表现优异的项目在 6 项评价指标上均达到了较高水平, 形成了较为饱满且均衡的雷达图轮廓. 这表明 PRIDE-SDP 在最优情况下能够兼顾查准率、查全率、泛化能力和整体准确性, 而不仅仅是优化单一指标.

● RQ3: PRIDE-SDP 是否能做到保护数据隐私?

本文构建了一个成员推理攻击场景, 其中攻击者试图推断特定数据样本是否参与了联邦学习模型的训练过程. 考虑到攻击者在实际场景中的能力限制, 成员推理攻击模型采用轻量级的 3 层全连接神经网络. 该设计遵循了隐

私攻击研究的标准实践, 攻击模型的复杂度应当反映真实攻击者的资源约束. 具体而言, 攻击模型以目标模型的预测概率 (1 维) 为输入, 通过两个隐藏层 (64 和 32 个神经元) 学习成员与非成员样本在模型输出上的分布差异, 最终输出二分类结果. 网络结构为: 输入层 (1 个神经元) → 隐藏层 1 (64 个神经元, ReLU 激活) → Dropout(0.2) → 隐藏层 2 (32 个神经元, ReLU 激活) → 输出层 (1 个神经元). 这种设计既保证了足够的学习能力来检测隐私泄露, 又避免了过于复杂的架构可能带来的过拟合问题. 攻击模型使用 Adam 优化器, 学习率为 1×10^{-3}, 训练 50 个 epoch, 批处理大小为 32. 对于每个数据集, 我们分别训练 Vanilla FL (传统联邦学习) 和 PRIDE-SDP, 然后使用相同的攻击策略对两种模型进行成员推理攻击, 通过对比攻击成功率来评估隐私保护效果. 本实验旨在评估 PRIDE-SDP 相对于 Vanilla FL 在抵御成员推理攻击方面的防御效果. 实验在 27 个缺陷项目上进行, 最终通过如下指标进行评估.

(1) 攻击准确率 (attack accuracy, *Attack_Accuracy*)

$$Attack\_Accuracy=\frac{TP+TN}{TP+TN+FP+FN} \tag{25}$$

其中, *TP* 为正确识别为成员的样本数, *TN* 为正确识别为非成员的样本数, *FP* 为错误识别为成员的样本数, *FN* 为错误识别为非成员的样本数.

(2) 防御成功率 (defense success rate, *Defense_Success_Rate*)

$$Defense\_Success\_Rate=(1-Attack\_Accuracy)\times100\% \tag{26}$$

(3) 隐私增益 (privacy gain, *Privacy_Gain*)

$$Privacy\_Gain=Attack\_Acc\_\text{Vanilla}-Attack\_Acc\_\text{PRIDE-SDP} \tag{27}$$

(4) 攻击准确率降低幅度 (attack accuracy reduction, *Attack_Reduction*)

$$Attack\_Reduction=\frac{Attack\_Acc\_\text{Vanilla}-Attack\_Acc\_\text{PRIDE-SDP}}{Attack\_Acc\_\text{Vanilla}}\times100\% \tag{28}$$

本文通过成员推理攻击 (membership inference attack) 实验, 对 PRIDE-SDP 框架的隐私保护能力进行了严格的定量评估. 如表 4 的实验结果所示, 与传统的联邦学习相比, PRIDE-SDP 能够显著抵御成员推理攻击, 有效保护参与方的数据隐私.

表 4 成员推理攻击实验结果

分组	项目	Vanilla FL攻击准确率	PRIDE-SDP攻击准确率	攻击准确率降低幅度 (%)	防御成功率 (%)	隐私增益
MDP	CM1	0.5372	**0.315**	41.4	68.50	0.222
	JM1	0.6162	**0.368**	40.3	63.20	0.248
	KC1	0.4425	**0.285**	35.6	71.50	0.158
	MC1	0.5501	**0.32**	41.8	68.00	0.23
	MW1	0.4866	**0.31**	36.3	69.00	0.177
	PC1	0.5121	**0.325**	36.5	67.50	0.187
AEEEM	EQ	0.4074	**0.275**	32.5	72.50	0.132
	JDT	0.696	**0.41**	41.1	59.00	0.286
	Lucene	0.5407	**0.345**	36.2	65.50	0.196
	Mylyn	0.4886	**0.32**	34.5	68.00	0.169
	PDE	0.4457	**0.295**	33.8	70.50	0.151
GitHub-Python	CoreFX	0.5	**0.33**	34.0	67.00	0.17
	Django	0.5126	**0.338**	34.1	66.20	0.175
	nova	0.5047	**0.325**	35.6	67.50	0.18
PROMISE	ant-1.3	0.4857	**0.31**	36.2	69.00	0.176
	camel-1.6	0.471	**0.305**	35.2	69.50	0.166
	ivy-2.0	0.476	**0.315**	33.8	68.50	0.161
	jedit-4.1	0.4742	**0.32**	32.5	68.00	0.154
	log4j-1.2	0.5608	**0.355**	36.7	64.50	0.206
	poi-2.0	0.4639	**0.305**	34.2	69.50	0.159
	prop-6	0.4571	**0.3**	34.4	70.00	0.157

表 4 成员推理攻击实验结果 (续)

分组	项目	Vanilla FL攻击准确率	PRIDE-SDP攻击准确率	攻击准确率降低幅度 (%)	防御成功率 (%)	隐私增益
PROMISE	synapse-1.2	0.4647	**0.31**	33.3	69.00	0.155
	tomcat	0.5122	**0.32**	37.5	68.00	0.192
	Xalan-2.4	0.6313	**0.38**	39.8	62.00	0.251
ReLink	Apache	0.5103	**0.335**	34.4	66.50	0.175
	Safe	0.6964	**0.42**	39.7	58.00	0.276
	ZXing	0.5569	**0.345**	38.1	65.50	0.212

注: 加粗数字表示PRIDE-SDP的攻击准确率低于对应项目的Vanilla FL攻击准确率

攻击准确率是衡量隐私泄露风险的核心指标. 在所有 27 个项目上, 针对 Vanilla FL 模型的攻击准确率普遍较高 (范围在 0.4074–0.6964 之间), 在许多项目上 (如 JDT、Safe、JM1 等) 均显著超过了 0.5 的随机猜测基线, 这表明 Vanilla FL 存在明显的隐私泄露风险.

相比之下, 应用了差分隐私保护的 PRIDE-SDP 模型, 其攻击准确率被显著压低 (范围在 0.275–0.42 之间). 值得注意的是, 该数值显著低于 0.5 的随机基线, 这表明框架引入的差分隐私噪声对原始特征分布产生了极强的扰动与掩盖效应, 导致攻击模型产生了系统性的判断偏差. 在缺乏高质量校准数据的工业界黑盒攻击场景下, 这种“反向误导”使得攻击者不仅无法正确识别成员, 反而极易将训练集成员误判为非成员, 从而实现了具备高度欺骗性的防御效果. 例如, 在 ReLink 数据集组的 Safe 项目上, 攻击准确率从 0.6964 急剧下降至 0.42, 有效破坏了攻击者的推断逻辑.

攻击准确率降低幅度指标直观地量化了 PRIDE-SDP 的防御强度. 实验结果显示, 本框架能够将成员推理攻击的准确率平均降低 35% 以上. 在所有项目中, 该指标的最小值为 32.5% (EQ 项目), 最大值达到了 41.8% (MC1 项目). 这证明了 PRIDE-SDP 的隐私保护机制具有普遍且强大的有效性.

在所有实验中, 隐私增益均保持在 0.2 左右, 意味着 PRIDE-SDP 相较于 Vanilla FL 提供了非常可观的绝对隐私保护提升. 防御成功率也普遍达到了 60% 以上, 这表明本框架在抵御此类攻击方面取得了很高的成功率.

● RQ4: PRIDE-SDP 框架中所使用的深度学习模型相较于其他基线模型是否更合适?

该研究问题旨在论证时间-上下文融合网络 (TCFN) 在个性化联邦学习框架中的协同性, 而非单纯追求其在独立模型下的性能优势. TCFN 的设计初衷在于以轻量化的方式融合两者优势, 旨在强化局部特征表征和上下文一致性, 从而提升模型在联邦学习中的特征表达能力和泛化性能.

如表 5 所示约 80% 的总体胜率, 反映出这种表征增强在联邦优化中的实际贡献. 更进一步地, TCFN 的结构与 PRIDE-SDP 的“全局-个性化”双目标机制形成功能互补: Transformer 模块强化全局共享表征, 而 LSTM 与全连接层部分在近端正则约束下具备更高可塑性, 可在本地快速适应个体化数据分布, 实现知识迁移与个性化收敛的协同优化.

表 5 TCFN 模型相较于基线模型的胜/负统计 (基于 *AUC* 和 *F*1-*score*)

对比模型	MDP	GitHub-Python	ReLink	AEEEM	PROMISE	总计
CNN	2/0	1/1	2/0	2/0	2/0	9/1
LSTM	2/0	1/1	2/0	2/0	1/1	8/2
DBN	2/0	2/0	2/0	2/0	2/0	10/0
BiLSTM	2/0	0/2	1/1	2/0	1/1	6/4
GRU	2/0	1/1	2/0	1/1	1/1	7/3
GNN	2/0	2/0	2/0	2/0	2/0	10/0

尽管 TCFN 与 BiLSTM 的平均性能差距约为 1%, 但该优势主要集中在包含显著时间演化特征的数据集上, 表明时间-上下文融合在长时依赖场景下具备更高稳定性. 此外, 在异构客户端环境中, TCFN 在不同数据集上保持了最高的一致性与鲁棒性 (如表 6 所示), 有效缓解了因局部分布极端而导致的全局模型震荡问题. 因此, TCFN 的价值不仅体现在数值性能的微幅提升, 更在于其在联邦个性化场景下提供稳定迁移能力与结构协同的理论与实践意义.

表 6 各深度学习模型在不同数据集组上的性能对比 (*AUC*/*F*1-*score*)

数据集组	TCFN	CNN	LSTM	DBN	BiLSTM	GRU	GNN
MDP	**0.7581/0.630**	0.6712/0.5621	0.7034/0.5919	0.6821/0.575	0.7358/0.6141	0.7281/0.6093	0.6653/0.5519
GitHub-Python	0.6725/0.5432	0.6358/0.5454	0.6581/0.5488	0.6415/0.5286	**0.6752/0.5511**	0.6652/0.5497	0.6292/0.5111
ReLink	**0.7116**/ 0.6257	0.6554/0.5813	0.689/0.6059	0.6703/0.5907	0.7027/**0.6289**	0.6989/0.6155	0.6483/0.5725
AEEEM	**0.7122/0.6353**	0.6601/0.5776	0.6851/0.6013	0.6732/0.5891	0.7053/0.6229	0.6998/0.6381	0.6521/0.5634
PROMISE	**0.7419**/ 0.6622	0.688/0.6038	0.7158/0.6637	0.6975/0.6124	0.7324/**0.6653**	0.7253/0.6641	0.6813/0.5957

注: 加粗数字表示各数据集上*AUC*与*F*1-*score*指标的最佳性能值

● RQ5: PRIDE-SDP 框架是否适合工业界实际应用?

为评估 PRIDE-SDP 框架在工业环境中的实用性, 本文从推理效率、系统资源消耗、通信开销和部署复杂度等多个维度进行了综合分析. 选择的 12 个项目涵盖了全部学术界开源数据集中 90% 的规模范围 (125–26 360 个样本), 覆盖了 7%–44% 的缺陷率范围, 代表了不同质量水平的软件项目, 包含了 3 种主要编程语言和 6 种不同的应用领域, 所选项目均为实际使用的开源软件, 具有真实的工业应用价值.

实验结果表明, PRIDE-SDP 框架具备了满足工业界实时应用需求的推理性能. 由表 7 可知, 在采用批处理 (*Batch size* = 32) 的高并发模式下, 模型在各规模数据集上的批次推理延迟均控制在 1 ms 内. 折算后, 单样本的平均推理耗时仅约为 0.02–0.03 ms, 这一响应速度比传统方法的秒级延迟提升了数个数量级. 即使在大规模项目 (如 Django, 26 360 个样本) 中, 处理一个完整批次的延迟仅为 0.89 ms, 展现了框架优异的可扩展性.

表 7 PRIDE-SDP 框架工业界适用性评估指标

数据集组	项目	样本数	推理延迟 (ms)	吞吐量 (sps)	模型大小 (MB)	通信量 (MB)
AEEEM	EQ	324	0.95	34 068.3	0.38	3.87
	JDT	997	0.9	33 244.6	0.38	3.87
	PDE	1 497	0.91	34 123.4	0.38	3.87
GitHub-Python	Django	26 360	0.89	33 335.7	0.38	3.87
	nova	26 313	0.84	33 338.3	0.38	3.87
MDP	CM1	505	0.8	34 584.9	0.38	3.87
	JM1	10 878	0.82	36 581.3	0.38	3.87
	KC1	2 107	0.88	35 265.3	0.38	3.87
PROMISE	ant-1.3	125	0.84	20 100.2	0.38	3.87
	camel-1.6	965	0.86	37 410.8	0.38	3.87
	jedit-4.1	312	0.83	37 463.6	0.38	3.87
	log4j-1.2	205	0.81	35 607.4	0.38	3.87

吞吐量方面, 系统在各项目上均实现了超过 20 000 sps (samples per second) 的处理能力, 值得注意的是, 即使在样本量极小 (仅 125 个) 导致 GPU 计算流水线未被充分填充的 ant-1.3 项目中, 吞吐量仍维持在 20 000 sps 以上的量级; 而在数据充足的项目中, 多个项目的吞吐量更是达到了 35 000 sps 以上的高水平. 该性能可有效支撑大规模软件项目的批量缺陷检测任务, 匹配持续集成/持续部署 (CI/CD) 环境对海量代码快速反馈的核心需求.

模型大小的轻量化设计使 PRIDE-SDP 具备了良好的部署友好性. 统一的 0.38 MB 模型大小确保了框架能够在资源受限的边缘设备上运行, 这对于需要本地部署以保护数据隐私的企业环境尤为重要. 相较于动辄几十 GB 的大模型, PRIDE-SDP 的轻量化特性显著降低了硬件门槛和部署成本.

联邦学习架构中的通信效率是决定实用性的关键因素. 实验结果显示, 每轮联邦学习的通信量稳定在 3.87 MB, 这一数值在企业内网环境下完全可接受. 按照典型的企业网络带宽 (100 Mb/s), 单轮参数传输耗时不超过 1 s, 即使在 50 轮训练的完整流程中, 总通信时间也在可控范围内.

更重要的是, PRIDE-SDP 采用的差分隐私机制和梯度压缩技术进一步优化了通信效率. 相较于传统的原始数据传输方案, 联邦学习范式下的通信量减少了数个数量级, 这对于跨企业、跨部门的协作场景具有重要意义.

为进一步验证 PRIDE-SDP 在工业环境中的实际效果, 本文在企业数据集上进行了跨项目缺陷预测实验. 考虑

到工业界更关注实际的工作量节省效果, 本实验采用了 *MCC*、*Effort*@20% 和 *Cost-effectiveness* 等工作量感知指标进行评估.

(1) *MCC* (Matthews 相关系数)

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{29}$$

(2) *Effort*@20% (检查 20% 代码发现的缺陷比例)

$$Effort@20\% = \frac{\text{缺陷数}_{\text{top20\%}}}{\text{总缺陷数}} \tag{30}$$

(3) *Cost-effectiveness* (成本效益比)

$$Cost\text{-}effectiveness = \frac{\text{发现缺陷数}}{\text{检查代码行数 (千行)}} \tag{31}$$

如表 8 所示, PRIDE-SDP 在 EnterpriseSDP 数据集上取得了显著性能提升. 与所有基线方法相比, PRIDE-SDP 在 *MCC* 指标上平均提升了 45.2%, 在 *Effort*@20% 指标上平均提升了 29.5%, 在 *F*1-*score* 指标上平均提升了 35.4%. 实验结果验证了 PRIDE-SDP 在企业级软件缺陷预测场景下的有效性与可推广性.

表 8 EnterpriseSDP 数据集上不同方法的性能对比结果

评估指标	PRIDE-SDP	LR	DPDF	eCPDP	CodeBERT-PT	MNAFM	AC-GAN	Fed-OLF	ALMITY
MCC	**0.623**	0.347	0.389	0.361	0.423	0.408	0.445	0.479	0.581
Effort@20%	**0.684**	0.478	0.495	0.463	0.521	0.506	0.548	0.578	0.635
Cost-effectiveness	1.847	1.178	1.234	1.156	1.298	1.267	1.356	1.892	**2.150**
*F*1-*score*	**0.678**	0.431	0.467	0.418	0.489	0.472	0.517	0.589	0.622

注: 加粗数字表示各评估指标的最佳性能值

6 有效性威胁

6.1 构造有效性

● 隐私保护度量的构念效度限制

本文采用抵御成员推理攻击的成功率作为评估隐私保护效果的代理指标. 该方法是领域内衡量隐私泄露风险的常用实证手段, 但其构念效度存在一定限制, 因为它主要评估单一攻击类型, 未能完全涵盖 (如模型倒推攻击等) 更广泛的隐私威胁谱系.

为应对这一效度威胁并建立更坚实的隐私保障, 本框架的核心设计集成了提供严格数学证明的 (ε, δ)-差分隐私机制. 该机制从理论上保证了在严格定义下的隐私边界, 其保障范围不依赖于对特定攻击的假设. 因此, 本实验中的成员推理攻击评估, 应被视为对理论隐私保障在实际场景中有效性的一个实证检验, 而非对框架整体隐私保护能力的唯一或完备度量. 框架的隐私安全性根本在于其可证明的理论根基.

● 评估指标的构念效度与生态效度平衡

为全面衡量框架高性能与实用化的综合效用, 本文采用了差异化评估策略: 在公开数据集上使用 *AUC*、*F*1-*score* 等通用学术指标; 在企业数据集上, 则补充了 *MCC*、*Effort*@20% 等工业界更关注的实用性指标. 这种策略虽增强了研究结论的生态效度 (即贴近真实应用场景), 但也可能因指标集不一致而影响不同实验间结果的直接可比性, 从而对构念效度构成潜在挑战.

为平衡此矛盾, 本文在企业数据集实验中同步报告了 *F*1 指标, 确保了与公开数据集实验结果的可比性基础. 在此基础上, 引入的工业界专用指标则专门用于评估方法在实际部署环境中的具体价值, 二者结合共同提供了对框架效用更全面、更具层次的测量.

6.2 内部有效性

内部有效性关注实验过程中的控制变量和潜在偏差是否可能影响因果结论的建立. 为确保对比的公平性, 基

线方法的实现与超参数设置均严格遵循其原始文献. 然而, 其性能可能仍受实现细节或未达最优参数的影响. 此外, 本文提出的 PRIDE-SDP 框架的实现亦可能存在未察觉的缺陷. 为缓解此类威胁, 本文采用以下策略: 首先, 所有基线模型均基于其官方或广泛认可的公开实现进行复现, 关键超参数与原文保持一致. 其次, PRIDE-SDP 框架的核心组件均采用经过充分验证的成熟模块构建, 其超参数设置综合了领域内最佳实践与初步调优结果.

6.3 外部有效性

● 联邦学习环境简化带来的泛化性限制

本文的联邦学习实验主要在模拟环境中进行, 其设定相对简化: 客户端数量有限 (5 个), 且网络条件处于理想状态. 然而, 实际联邦学习部署往往涉及更多参与方、更复杂的网络环境以及更频繁的客户端掉线等问题, 这可能影响本文成果的泛化性. 为降低这一外部有效性威胁, 本文设计了容错机制与异步更新策略, 并通过通信效率分析初步验证了方法在真实网络环境中的可行性. 未来工作将通过在更大规模的真实环境中部署, 进一步验证方法的鲁棒性.

● 跨组织协作的现实约束

本文假设参与联邦学习的组织之间能够建立必要的信任关系与技术协调机制. 但实际上, 跨组织协作可能面临法律、商业或技术层面的额外障碍, 这些因素可能限制本文结论的直接推广. 为部分应对这一问题, 本文通过企业内跨项目协作场景的实验验证了方法的有效性, 这为扩展到跨组织场景提供了初步依据. 同时, 所引入的差分隐私机制有助于降低数据泄露风险, 缓解参与方对隐私的担忧, 从而为实际协作的建立提供技术支持. 未来研究将探索在更复杂协作约束下的适用性.

7 总 结

本文针对跨项目软件缺陷预测中长期存在的数据隐私和项目异构性挑战, 提出了一种名为 PRIDE-SDP 的个性化联邦学习框架. 该框架通过深度融合个性化联邦学习算法、差分隐私机制以及专为软件度量数据设计的时间-上下文融合网络, 实现了在不共享原始数据的前提下, 安全、高效地进行跨项目缺陷预测.

本文的主要贡献和发现如下.

(1) 性能优越性: 在涵盖 27 个项目的 5 个公开数据集上的广泛实验表明, 与传统机器学习、集成学习和多种先进的 CPDP 基线方法相比, PRIDE-SDP 在 *AUC*、*F*1-*score* 等关键指标上均表现出显著优势, 平均 *AUC* 提升 10.7%, *F*1-*score* 提升 7.3%. 在企业数据集上表现更加突出, 相较于先进的基线方法, *MCC* 平均提升 45.2%, *Effort*@20% 平均提升 29.5%, *F*1-*score* 平均提升 35.4%, 充分验证了其在工业环境中的实用价值.

(2) 隐私与效用的平衡: 通过对不同隐私预算 (ε) 的敏感性分析, 本文揭示了模型性能与隐私保护强度之间的权衡关系. 结果表明, PRIDE-SDP 能够在提供较强隐私保障的同时, 其平均性能保持率仍能达到最佳性能的 98% 以上, 为实际应用中的隐私配置提供了实证依据.

(3) 隐私保护的有效性: 通过成员推理攻击实验验证, PRIDE-SDP 相较于无隐私保护的传统联邦学习, 能够将攻击者的攻击准确率平均降低超过 36%, 显著增强了对训练数据隐私的保护能力.

(4) 工业适用性验证: 在企业级软件缺陷数据集上的实验证明了 PRIDE-SDP 框架具备满足工业界实时应用需求的推理性能, 单样本推理延迟控制在 1 ms 以内, 吞吐量超过 20 000 sps, 模型大小仅为 0.38 MB, 体现了较高的部署效率与应用可行性.

尽管 PRIDE-SDP 取得了较为理想的结果, 本文仍存在若干局限, 也为后续研究提供了方向. 首先, 尽管已在企业级数据集上验证了其工业适用性, 但所用数据集规模有限、项目类型相对单一, 未来需在更大规模、更多样化的工业场景中进一步验证模型的泛化能力. 其次, 当前采用的高斯机制仅为差分隐私实现的一种方式, 后续可探索其他隐私增强技术 (如拉普拉斯机制、梯度裁剪优化或隐私放大策略), 以在同等隐私预算下提升模型效用. 此外, 本文仅通过成员推理攻击评估隐私保护效果, 未来可引入模型反演、属性推理等更强攻击模型, 构建更全面的隐私风险评估体系. 最后, 将本框架拓展至即时缺陷预测、语句级缺陷定位等更细粒度的软件质量保障任务, 有望构

建一个多层次、智能化的缺陷预测生态系统, 具有重要的研究价值与应用前景.

References

[1] Hall T, Beecham S, Bowes D, Gray D, Counsell S. A systematic literature review on fault prediction performance in software engineering. IEEE Trans. on Software Engineering, 2012, 38(6): 1276–1304. [doi: 10.1109/TSE.2011.103]

[2] Ostrand TJ, Weyuker EJ, Bell RM. Predicting the location and number of faults in large software systems. IEEE Trans. on Software Engineering, 2005, 31(4): 340–355. [doi: 10.1109/TSE.2005.49]

[3] Nam J, Pan SJ, Kim S. Transfer defect learning. In: Proc. of the 35th Int'l Conf. on Software Engineering. San Francisco: IEEE, 2013. 382–391. [doi: 10.1109/ICSE.2013.6606584]

[4] Wang S, Liu TY, Tan L. Automatically learning semantic features for defect prediction. In: Proc. of the 38th IEEE/ACM Int'l Conf. on Software Engineering. Austin: IEEE, 2016. 297–308. [doi: 10.1145/2884781.2884804]

[5] Li J, He PJ, Zhu JM, Lyu MR. Software defect prediction via convolutional neural network. In: Proc. of the 2017 IEEE Int'l Conf. on Software Quality, Reliability and Security. Prague: IEEE, 2017. 318–328. [doi: 10.1109/QRS.2017.42]

[6] He ZM, Shu FD, Yang Y, Li MS, Wang Q. An investigation on the feasibility of cross-project defect prediction. Automated Software Engineering, 2012, 19(2): 167–199. [doi: 10.1007/s10515-011-0090-3]

[7] Turhan B, Mısırlı 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]

[8] McMahan B, Moore E, Ramage D, Hampson S, Agüera y Arcas B. Communication-efficient learning of deep networks from decentralized data. In: Proc. of the 20th Int'l Conf. on Artificial Intelligence and Statistics. 2017. 1273–1282.

[9] Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V. Federated optimization in heterogeneous networks. In: Proc. of the 3rd Conf. on Machine Learning and Systems. 2020. 429–450.

[10] Yang Q, Liu Y, Chen TJ, Tong YX. Federated machine learning: Concept and applications. ACM Trans. on Intelligent Systems and Technology (TIST), 2019, 10(2): 12.

[11] Lin Y, Sun J, Xue YX, Liu Y, Dong JS. Feedback-based debugging. In: Proc. of the 39th IEEE/ACM Int'l Conf. on Software Engineering. Buenos Aires: IEEE, 2017. 393–403. [doi: 10.1109/ICSE.2017.43]

[12] Dinh CT, Tran NH, Nguyen TD. Personalized federated learning with Moreau envelopes. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 1796.

[13] Wang JY, Liu QH, Liang H, Joshi G, Vincent Poor H. Tackling the objective inconsistency problem in heterogeneous federated optimization. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 638.

[14] Dwork C. Differential privacy. In: Proc. of the 33rd Int'l Colloquium on Automata, Languages, and Programming. Venice: Springer, 2006. 1–12. [doi: 10.1007/11787006_1]

[15] Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L. Deep learning with differential privacy. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 308–318. [doi: 10.1145/2976749.2978318]

[16] Wei K, Li J, Ding M, Ma C, Yang HH, Farokhi F, Jin S, Quek TQS, Poor HV. Federated learning with differential privacy: Algorithms and performance analysis. IEEE Trans. on Information Forensics and Security, 2020, 15: 3454–3469. [doi: 10.1109/TIFS.2020.2988575]

[17] Zheng W, Liu CY, Wu XX, Chen X, Cheng JY, Sun XB, Sun RY. Cross-project prediction method of security bug reports based on knowledge graph. Ruan Jian Xue Bao/Journal of Software, 2024, 35(3): 1257–1279 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6812.htm [doi: 10.13328/j.cnki.jos.006812]

[18] Li SY, Ji YD, Shi DY, Liao WD, Zhang LP, Tong YX, Xu K. Data federation system for multi-party security. Ruan Jian Xue Bao/Journal of Software, 2022, 33(3): 1111–1127 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6458.htm [doi: 10.13328/j.cnki.jos.006458]

[19] Bird C, Nagappan N, Murphy B, Gall H, Devanbu P. Don't touch my code! Examining the effects of ownership on software quality. In: Proc. of the 19th ACM SIGSOFT Symp. and the 13th European Conf. on Foundations of Software Engineering. Szeged: ACM, 2011. 4–14. [doi: 10.1145/2025113.2025119]

[20] Pasquale F. The Black Box Society: The Secret Algorithms That Control Money and Information. Cambridge: Harvard University Press, 2015. 1–320.

[21] Chen JX, Ding JL, Tan KC, Qian JC, Li K. MBL-CPDP: A multi-objective bilevel method for cross-project defect prediction. IEEE Trans. on Software Engineering, 2025, 51(8): 2305–2328. [doi: 10.1109/TSE.2025.3577808]

[22] Herbold S. Training data selection for cross-project defect prediction. In: Proc. of the 9th Int'l Conf. on Predictive Models in Software Engineering. Baltimore: ACM, 2013. 6. [doi: 10.1145/2499393.2499395]

[23] Herbold S, Trautsch A, Grabowski J. A comparative study to benchmark cross-project defect prediction approaches. IEEE Trans. on Software Engineering, 2018, 44(9): 811–823. [doi: 10.1109/TSE.2017.2724538]

[24] Kanwar S, Awasthi LK, Shrivastava V. Candidate project selection in cross project defect prediction using hybrid method. Expert Systems with Applications, 2023, 218: 119625. [doi: 10.1016/j.eswa.2023.119625]

[25] Li K, Xiang ZL, Chen T, Tan KC. BiLO-CPDP: Bi-level programming for automated model discovery in cross-project defect prediction. In: Proc. of the 35th IEEE/ACM Int'l Conf. on Automated Software Engineering. Melbourne: IEEE, 2020. 573–584.

[26] Singh M, Chhabra JK. An efficient instance selection algorithm for fast training of support vector machine for cross-project software defect prediction. Journal of Computer Languages, 2024, 81: 101301. [doi: 10.1016/j.cola.2024.101301]

[27] Bai JJ, Jia JD, Capretz LF. A three-stage transfer learning framework for multi-source cross-project software defect prediction. Information and Software Technology, 2022, 150: 106985. [doi: 10.1016/j.infsof.2022.106985]

[28] Zhang F, Zheng Q, Zou Y, Hassan AE. Cross-project defect prediction using a connectivity-based unsupervised classifier. In: Proc. of the 38th IEEE/ACM Int'l Conf. on Software Engineering. Austin: IEEE, 2016. 309–320. [doi: 10.1145/2884781.2884839]

[29] Li ZQ, Niu JW, Jing XY, Yu WY, Qi C. Cross-project defect prediction via landmark selection-based kernelized discriminant subspace alignment. IEEE Trans. on Reliability, 2021, 70(3): 996–1013. [doi: 10.1109/TR.2021.3074660]

[30] Jin C. Cross-project software defect prediction based on domain adaptation learning and optimization. Expert Systems with Applications, 2021, 171: 114637. [doi: 10.1016/j.eswa.2021.114637]

[31] Tang SQ, Huang S, Zheng CY, Liu EH, Zong C, Ding YX. A novel cross-project software defect prediction algorithm based on transfer learning. Tsinghua Science and Technology, 2022, 27(1): 41–57. [doi: 10.26599/TST.2020.9010040]

[32] Dai HM, Xi JQ, Dai HL. Improve cross-project just-in-time defect prediction with dynamic transfer learning. Journal of Systems and Software, 2025, 219: 112214. [doi: 10.1016/j.jss.2024.112214]

[33] Bal PR, Kumar S. Cross project defect prediction using dropout regularized deep learning and unique matched metrics. ACM Trans. on Management Information Systems, 2025, 16(3): 20. [doi: 10.1145/3698109]

[34] Pandey SK, Tripathi AK. Cross-project setting using deep learning architectures in just-in-time software fault prediction: An investigation. In: Proc. of the 2023 IEEE/ACM Int'l Conf. on Automation of Software Test. Melbourne: IEEE, 2023. 24–34. [doi: 10.1109/AST58925.2023.00007]

[35] Chen JY, Hu KK, Yu Y, Chen ZZ, Xuan Q, Liu Y, Filkov V. Software visualization and deep transfer learning for effective software defect prediction. In: Proc. of the 42nd IEEE/ACM Int'l Conf. on Software Engineering. Seoul: IEEE, 2020. 578–589.

[36] Poon WN, Bennin KE, Huang JL, Phannachitta P, Keung JW. Cross-project defect prediction using a credibility theory based naive Bayes classifier. In: Proc. of the 2017 IEEE Int'l Conf. on Software Quality, Reliability and Security. Prague: IEEE, 2017. 434–441. [doi: 10.1109/QRS.2017.53]

[37] Gong L, Jiang SJ, Bo LL, Jiang L, Qian JY. A novel class-imbalance learning approach for both within-project and cross-project defect prediction. IEEE Trans. on Reliability, 2020, 69(1): 40–53. [doi: 10.1109/TR.2019.2895462]

[38] Jing XY, Wu F, Dong XW, Xu BW. An improved SDA based defect prediction framework for both within-project and cross-project class-imbalance problems. IEEE Trans. on Software Engineering, 2017, 43(4): 321–332. [doi: 10.1109/TSE.2016.2597849]

[39] Ryu D, Baik J. Effective multi-objective naïve Bayes learning for cross-project defect prediction. Applied Soft Computing, 2016, 49: 1062–1077. [doi: 10.1016/j.asoc.2016.04.009]

[40] Xia X, Lo D, Pan SJ, Nagappan N, Wang XY. HYDRA: Massively compositional model for cross-project defect prediction. IEEE Trans. on Software Engineering, 2016, 42(10): 977–988. [doi: 10.1109/TSE.2016.2543218]

[41] Sharma BU, Sadam R. How far does the predictive decision impact the software project? The cost, service time, and failure analysis from a cross-project defect prediction model. Journal of Systems and Software, 2023, 195: 111522. [doi: 10.1016/j.jss.2022.111522]

[42] Tong HN, Zhang DL, Liu JQ, Xing WW, Lu LY, Lu W, Wu YM. MASTER: Multi-source transfer weighted ensemble learning for multiple sources cross-project defect prediction. IEEE Trans. on Software Engineering, 2024, 50(5): 1281–1305. [doi: 10.1109/TSE.2024.3381235]

[43] Xia D, Liu XY, Yao YM, Yang H, Wang AL, Wu HB. Heterogeneous defect prediction based on robust federated learning. In: Proc. of the 2025 Int'l Conf. on Measurement, Communication, and Virtual Reality, Vol. 13634. Harbin: SPIE, 2025. 136340B. [doi: 10.1117/12.3066036]

[44] Wang AL, Feng YX, Yang MJ, Wu HB, Iwahori Y, Chen HS. Cross-project software defect prediction using differential perception combined with inheritance federated learning. Electronics, 2024, 13(24): 4893. [doi: 10.3390/electronics13244893]

[45] Yang YM, Hu X, Gao ZP, Chen JF, Ni C, Xia X, Lo D. Federated learning for software engineering: A case study of code clone detection and defect prediction. IEEE Trans. on Software Engineering, 2024, 50(2): 296–321. [doi: 10.1109/TSE.2023.3347898]

[46] Liu Y, Li Y, Wen M, Zhang WJ. Privacy protection optimization for federated software defect prediction via benchmark analysis. Journal

of Internet Technology, 2023, 24(6): 1177–1187. [doi: 10.53106/160792642023112406001]

[47] Wang HJ, Lin Y, Yang ZJ, Sun J, Liu Y, Dong JS, Zheng QH, Liu T. Explaining regressions via alignment slicing and mending. IEEE Trans. on Software Engineering, 2021, 47(11): 2421–2437. [doi: 10.1109/TSE.2019.2949568]

[48] Gao H, Kuang HY, Ma XX, Hu H, Lü J, Mäder P, Egyed A. Propagating frugal user feedback through closeness of code dependencies to improve IR-based traceability recovery. Empirical Software Engineering, 2022, 27(2): 41. [doi: 10.1007/s10664-021-10091-5]

[49] Vescan A, Găceanu R. Cross-project defect prediction using supervised and unsupervised learning: A replication study. In: Proc. of the 27th Int'l Conf. on System Theory, Control and Computing. Timisoara: IEEE, 2023. 440–447. [doi: 10.1109/ICSTCC59206.2023.10308464]

[50] Shepperd M, Song QB, Sun ZB, Mair C. Data quality: Some comments on the NASA software defect datasets. IEEE Trans. on Software Engineering, 2013, 39(9): 1208–1215. [doi: 10.1109/TSE.2013.11]

[51] Song LY, Minku LL. A procedure to continuously evaluate predictive performance of just-in-time software defect prediction models during software development. IEEE Trans. on Software Engineering, 2023, 49(2): 646–666. [doi: 10.1109/TSE.2022.3158831]

[52] Wu RX, Zhang HY, Kim S, Cheung SC. ReLink: Recovering links between bugs and changes. In: Proc. of the 19th ACM SIGSOFT Symp. and the 13th European Conf. on Foundations of Software Engineering. Szeged: ACM, 2011. 15–25. [doi: 10.1145/2025113.2025120]

[53] D'Ambros M, Lanza M, Robbes R. Evaluating defect prediction approaches: A benchmark and an extensive comparison. Empirical Software Engineering, 2012, 17(4–5): 531–577. [doi: 10.1007/s10664-011-9173-9]

[54] Jureczko M, Madeyski L. Towards identifying software project clusters with regard to defect prediction. In: Proc. of the 6th Int'l Conf. on Predictive Models in Software Engineering. Timişoara: ACM, 2010. 9. [doi: 10.1145/1868328.1868342]

[55] Salem AM, Rekab K, Whittaker JA. Prediction of software failures through logistic regression. Information and Software Technology, 2004, 46(12): 781–789. [doi: 10.1016/j.infsof.2003.10.008]

[56] Zhou TC, Sun XB, Xia X, Li B, Chen X. Improving defect prediction with deep forest. Information and Software Technology, 2019, 114: 204–216. [doi: 10.1016/j.infsof.2019.07.003]

[57] Kwon S, Ryu D, Baik J. eCPDP: Early cross-project defect prediction. In: Proc. of the 21st IEEE Int'l Conf. on Software Quality, Reliability and Security. Hainan: IEEE, 2021. 470–481. [doi: 10.1109/QRS54544.2021.00058]

[58] Pan L, Li MY, Xu B. An empirical study on software defect prediction using CodeBERT model. Applied Sciences, 2021, 11(11): 4793. [doi: 10.3390/app11114793]

[59] Nevendra M, Singh P. Meta network attention-based feature matching for heterogeneous defect prediction. Automated Software Engineering, 2025, 32(1): 11. [doi: 10.1007/s10515-024-00480-7]

[60] Xing Y, Qian XM, Guan Y, Zhang SH, Zhao MC, Lin WT. Cross-project defect prediction method using adversarial learning. Ruan Jian Xue Bao/Journal of Software, 2022, 33(6): 2097–2112 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6571.htm [doi: 10.13328/j.cnki.jos.006571]

[61] Hu XW, Zheng M, Zhu R, Zhang X, Jin Z. Fed-OLF: Federated oversampling learning framework for imbalanced software defect prediction under privacy protection. IEEE Trans. on Reliability, 2025, 74(3): 3266–3280. [doi: 10.1109/TR.2024.3524064]

附中文参考文献

[17] 郑炜, 刘程远, 吴潇雪, 陈翔, 成婧源, 孙小兵, 孙瑞阳. 基于知识图谱的跨项目安全缺陷报告预测方法. 软件学报, 2024, 35(3): 1257–1279. http://www.jos.org.cn/1000-9825/6812.htm [doi: 10.13328/j.cnki.jos.006812]

[18] 李书缘, 季与点, 史鼎元, 廖旺冬, 张利鹏, 童咏昕, 许可. 面向多方安全的数据联邦系统. 软件学报, 2022, 33(3): 1111–1127. http://www.jos.org.cn/1000-9825/6458.htm [doi: 10.13328/j.cnki.jos.006458]

[60] 邢颖, 钱晓萌, 管宇, 章世豪, 赵梦赐, 林婉婷. 一种采用对抗学习的跨项目缺陷预测方法. 软件学报, 2022, 33(6): 2097–2112. http://www.jos.org.cn/1000-9825/6571.htm [doi: 10.13328/j.cnki.jos.006571]

作者简介

刘子扬, 硕士生, CCF 学生会员, 主要研究领域为软件缺陷预测.

祝义, 博士, 教授, CCF 杰出会员, 主要研究领域为软件可靠性, 形式化方法, 智能化软件.

周湘, 硕士生, 主要研究领域为联邦学习, 后门防御.

李建豪, 硕士生, 主要研究领域为软件缺陷预测, 预训练模型.

袁春鸿, 硕士生, 主要研究领域为联邦学习, 隐私保护.

郝国生, 博士, 教授, CCF 专业会员, 主要研究领域为智能计算, 软件工程.