

**CostAdaptiveSamplerBoost: 一种用于软件缺陷预测的代价敏感集成学习方法**

**CostAdaptiveSamplerBoost: A Cost-Sensitive Ensemble Learning Method for Software Defect Prediction**

|  |  |  |
| --- | --- | --- |
| **作者姓名** | **：** | **杨金城** |
| **院系** | **：** | **经济与统计学院** |
| **学科专业** | **：** | **数据科学与大数据技术** |
| **学号** | **：** | **32264700042** |
| **指导教师** | **：** | **戴宏亮** |
| **完成日期** | **；** | **2025.05** |

摘要

软件缺陷预测 (Software Defect Prediction, SDP) 对于保障软件质量和优化开发资源至关重要。然而，实际缺陷数据集中普遍存在的类别不平衡问题（缺陷样本远少于非缺陷样本）以及将缺陷模块误判为无缺陷（假阴性，FN）所带来的高昂代价，对传统机器学习模型的性能构成了严峻挑战。为有效应对这些问题，本文提出了一种新的代价敏感集成学习算法——CostAdaptiveSamplerBoost (CASB)。CASB 算法在 AdaBoost 框架的基础上，创新性地引入了一种代价敏感的样本权重更新机制。在每次迭代中，该机制根据预定义的成本矩阵，对导致较高业务成本的错误分类（尤其是假阴性）给予更大的权重，从而引导后续的基学习器更加关注这些难以识别且重要的少数类样本。本文通过在一个包含来自 SEACRAFT 仓库9个历史软件度量元数据的合并缺陷数据集（activemq，camel，derby，groovy，hbase，hive，jruby，lucene，wicket）上进行了全面的实验评估。数据集上进行了全面的实验评估。我们将 CASB 与多种经典的机器学习算法（如逻辑回归、支持向量机、朴素贝叶斯、K-近邻、决策树）、标准集成方法（如 AdaBoost、XGBoost、随机森林）以及流行的不平衡学习技术（如 RUSBoost、SMOTE与决策树的组合）进行了性能比较。实验结果采用包括准确率、少数类F1分数、少数类召回率、G-mean、平衡准确率、AUC-PR 和一个自定义的总体模型成本在内的多项指标进行评估，并通过严格的交叉验证和统计显著性检验来确保结论的可靠性。实验结果表明，CASB 及其特定配置的变体，在多个关键性能指标上，特别是少数类F1分数、少数类召回率、G-mean 以及总体模型成本方面，一致地优于或表现出与所比较的基线方法相当甚至更好的性能。消融研究进一步证实了 CASB 中代价敏感机制的有效性。这些发现表明，CASB 是一种有前景的、能够有效处理软件缺陷预测中类别不平衡和错分成本问题的集成学习方法，为开发更可靠、更经济的缺陷预测模型提供了新的思路。

关键词: 软件缺陷预测；类别不平衡；成本敏感学习；集成学习；CostAdaptiveSamplerBoost；AdaBoost

目录

[摘要 I](#_Toc198983775)

[第1章 引言 1](#_Toc198983776)

[1.1 研究背景及意义 1](#_Toc198983777)

[1.2 研究现状 3](#_Toc198983778)

[1.2.1 类别不平衡 3](#_Toc198983779)

[1.2.2 错分成本的不对称性 4](#_Toc198983780)

[1.3 现有工作的局限性 4](#_Toc198983781)

[1.3.1 数据层面方法的局限性 5](#_Toc198983782)

[1.3.2 算法层面方法的局限性 5](#_Toc198983783)

[1.3.3 集成学习方法的现有局限性 6](#_Toc198983784)

[1.4 本文贡献 7](#_Toc198983785)

[第2章 相关工作 9](#_Toc198983786)

[2.1 类别不平衡学习 9](#_Toc198983787)

[2.1.1 数据层面方法 9](#_Toc198983788)

[2.1.2 算法层面方法 10](#_Toc198983789)

[2.1.3 集成学习与不平衡数据 11](#_Toc198983790)

[2.2 成本敏感学习 12](#_Toc198983791)

[2.2.1 成本矩阵 12](#_Toc198983792)

[2.2.2 成本敏感学习的主要方法 12](#_Toc198983793)

[2.2.3 成本敏感学习与类别不平衡的关系 14](#_Toc198983794)

[第3章 提出的方法: CostAdaptiveSamplerBoost (CASB) 14](#_Toc198983795)

[3.1 算法概述与动机 15](#_Toc198983796)

[3.2 CostAdaptiveSamplerBoost 算法核心步骤 15](#_Toc198983797)

[3.3 与现有Boosting算法的简要比较 18](#_Toc198983798)

[3.4 CASB 的主要参数 18](#_Toc198983799)

[第4章 实验设置 18](#_Toc198983800)

[4.1 数据集 18](#_Toc198983801)

[4.2 数据预处理与特征选择 19](#_Toc198983802)

[4.3 对比算法 22](#_Toc198983803)

[4.4 性能评估指标 23](#_Toc198983804)

[4.5 实验流程与评估方法 24](#_Toc198983805)

[4.6 统计显著性检验 25](#_Toc198983806)

[4.7 实验环境 25](#_Toc198983807)

[第5章 结果与分析 25](#_Toc198983808)

[5.1 CASB 与各类基线算法的性能比较 26](#_Toc198983809)

[5.2 CASB 内部变体性能分析与代价敏感机制的有效性 29](#_Toc198983810)

[5.3 CASB 关键超参数敏感性分析 31](#_Toc198983811)

[第6章 讨论 33](#_Toc198983812)

[6.1 主要发现总结与解读 33](#_Toc198983813)

[6.2 与现有工作的比较和联系 34](#_Toc198983814)

[6.3 研究的局限性 34](#_Toc198983815)

[第7章 结论与未来工作 35](#_Toc198983816)

[7.1 结论 35](#_Toc198983817)

[7.2 未来工作 36](#_Toc198983818)

[参考文献： 37](#_Toc198983819)

# 引言

## 研究背景及意义

在当今高度依赖信息技术的社会中，软件系统已渗透到社会经济、科学研究、以及日常生活的方方面面。随着软件系统规模的持续扩张和功能复杂性的日益增加，确保软件的质量、可靠性和安全性变得空前重要。然而，软件开发过程本身就是一个复杂且易于出错的活动，导致软件缺陷（Bugs 或 Faults）的产生几乎不可避免。这些缺陷如果未能及时发现和修复，可能导致软件功能异常、性能下降、安全漏洞，甚至引发灾难性的系统故障，造成巨大的经济损失和严重的社会影响。据相关研究统计，软件缺陷修复的成本随着其在软件生命周期中被发现阶段的推迟而呈指数级增长。因此，能够在软件开发早期阶段有效地预测和定位潜在的缺陷模块，对于提高软件质量、降低开发和维护成本、优化测试资源分配具有至关重要的理论意义和实际应用价值。

软件缺陷预测 (Software Defect Prediction, SDP) 是旨在解决这一问题的关键技术之一，其对于提升软件质量和优化开发流程具有重要作用 [1]。SDP 利用历史项目数据中的软件度量元和已知的缺陷信息，通过构建机器学习模型来预测新的或未发布的软件模块中存在缺陷的可能性。通过准确的缺陷预测，软件开发团队可以将有限的质量保证资源优先投入到最有可能包含缺陷的高风险模块上，从而提高缺陷检测的效率和效果，缩短软件发布周期，并最终提升用户满意度。

然而，尽管机器学习在SDP领域取得了显著进展，但实际应用中仍面临诸多挑战。其中，**类别不平衡 (Class Imbalance)** 是一个普遍存在且亟待解决的问题，并被认为是 SDP 领域未来研究的潜在方向之一 [1]。在大多数真实的软件项目中，存在缺陷的模块（少数类）数量通常远少于无缺陷的模块（多数类）。例如，缺陷模块的比例可能仅占总模块数量的5%到20%，甚至更低。先前已有工作针对软件缺陷预测中的不平衡问题，比较了处理此类数据的不同技术 [2]。传统的分类算法在设计时往往假设类别分布是均衡的，当直接应用于这种高度不平衡的数据时，它们倾向于过分关注多数类以达到整体准确率的最大化，从而导致对少数缺陷类的识别能力低下，产生大量的假阴性（False Negatives, FN）——即将有缺陷的模块错误地判断为无缺陷。不平衡分类技术已被应用于软件缺陷预测，这对于软件质量建模非常重要 [3]。这种对少数类的忽视在SDP的实践中是尤其不可接受的，因为漏报一个真实的缺陷可能比误报一个非缺陷模块（假阳性, False Positives, FP）带来更为严重的后果。

与类别不平衡问题紧密相关且同样关键的另一个挑战是**错分成本的不对称性 (Asymmetric Misclassification Costs)**。在SDP场景下，不同类型的预测错误所造成的业务成本和风险是显著不同的。一个假阴性（FN）错误意味着一个潜在的缺陷未能被发现，可能在软件部署后引发故障，其修复成本也更高。相比之下，一个假阳性（FP）错误虽然也会消耗一些额外的审查资源，但其负面影响通常远小于FN错误。通过特定的损失函数设计来处理这种不对称的错分成本，是成本敏感学习作为解决不平衡问题和管理风险的一种重要途径，其思想与SDP中对高代价FN错误的关注高度契合 [6]。

尽管已有一些研究分别尝试解决类别不平衡或成本敏感问题，例如探讨基于数据分布和自适应权重的多类别不平衡分类方法，其中也涉及了成本敏感和集成学习的思想 [7]。然而，能够同时、有效地应对类别不平衡和错分成本这两个相互交织的挑战，并能在实际缺陷预测中取得鲁棒性能的机器学习方法仍有进一步探索和改进的空间。**本研究的动机正是源于此**：开发一种新的机器学习模型，该模型能够内接地适应类别不平衡特性，并显式地将不对称的错分成本整合到其学习过程中，从而在软件缺陷预测任务中实现更优的少数类检测性能和更低的总体业务风险。这促使我提出了 CostAdaptiveSamplerBoost (CASB) 算法，旨在为软件工程实践提供一个更有效的缺陷预测工具。

## 研究现状

尽管软件缺陷预测 (SDP) 为提升软件质量提供了有力的工具，但在实际应用和研究中，研究者和实践者面临着一系列固有的挑战，这些挑战显著影响着预测模型的性能和实用性。其中，类别不平衡 (Class Imbalance) 和 错分成本的不对称性 (Asymmetric Misclassification Costs) 是两个最为突出且相互关联的核心问题，它们共同构成了SDP领域持续研究的焦点。

### 类别不平衡

类别不平衡是指在数据集中，不同类别的样本数量差异巨大的现象。在SDP的上下文中，这意味着有缺陷的模块（通常被视为正类或少数类）的数量远少于无缺陷的模块（通常被视为负类或多数类）[1,2]。这种不平衡现象在真实的软件项目中非常普遍，缺陷模块的比例往往非常低，有时甚至低于5%[1]。例如，在一个包含数千个软件模块的大型项目中，可能只有几十个或几百个模块被最终确认为包含缺陷。

这种严重的类别不平衡对传统的机器学习算法（如决策树、逻辑回归、支持向量机等）的训练过程和性能评估都带来了显著的负面影响。大多数标准分类算法在设计时都隐式地假设类别分布是相对均衡的，其优化目标通常是最大化整体分类准确率或最小化整体错误率 [2]。当应用于高度不平衡的数据集时，这些算法会自然地偏向于数量占优的多数类，因为将所有样本都预测为多数类也能获得较高的准确率。结果是，模型可能对少数类（即缺陷模块）的识别能力非常差，导致大量的假阴性（漏报缺陷），这与SDP的主要目标——找出潜在的缺陷——是背道而驰的 [3]。此外，一些性能评估指标，如准确率 (Accuracy)，在类别不平衡的情况下会产生误导性的结果，无法真实反映模型对少数类的预测能力。因此，如何有效地处理类别不平衡，提升模型对缺陷模块的检出率，是SDP研究中必须解决的关键问题。

### 错分成本的不对称性

与类别不平衡问题紧密相关的是错分成本的不对称性。在许多实际的分类任务中，将一种类别的样本错误地预测为另一种类别所带来的后果和代价是不同的。在软件缺陷预测领域，这种不对称性尤为突出 [3]。

具体来说，SDP中主要存在两种类型的错分：

假阴性 (False Negative, FN): 将一个实际存在缺陷的模块错误地预测为无缺陷。这种错误的代价通常非常高。漏报的缺陷可能会在软件的后续阶段（如测试、部署甚至用户使用阶段）才被发现，此时修复缺陷的成本会急剧增加，并且可能导致系统故障、数据丢失、用户不满意、声誉受损等一系列严重后果 [3]。

假阳性 (False Positive, FP): 将一个实际无缺陷的模块错误地预测为有缺陷。这种错误的代价相对较低，主要体现在需要投入额外的人力资源（如开发人员或测试人员的时间）对该模块进行审查或更细致的测试。虽然这也会带来一定的成本，但通常远小于因FN错误导致的潜在损失 [3]。

传统的分类算法通常假设所有错分的成本是相等的，其目标是最小化总的错分数量，而不是最小化总的期望错分成本 [3]。然而，在SDP中，一个理想的预测模型应该能够区分不同错误的严重程度，并倾向于避免那些代价高昂的错误（即FN错误），即使这意味着可能会增加一些代价较低的错误（FP错误）。因此，将不对称的错分成本显式地整合到模型的学习过程和决策过程中，是提升SDP模型实用性和经济效益的另一个关键挑战 [6]。仅仅提高对少数类的检测率（例如通过采样技术）可能并不足够，如果模型不能根据成本进行优化，它仍然可能产生不符合业务需求的预测结果。

综上所述，类别不平衡和错分成本不对称是软件缺陷预测中两个相互交织、不容忽视的关键挑战。有效的SDP方法需要能够同时应对这两个问题，以构建出既能准确识别少数缺陷模块，又能最小化总体业务风险的预测模型。

## 现有工作的局限性

尽管研究者们已经提出了多种方法来尝试缓解软件缺陷预测 (SDP) 中的类别不平衡和错分成本不对称问题，但现有工作在同时、有效地解决这两个相互交织的挑战方面仍存在一定的局限性，这为进一步的研究和改进留下了空间。

### 数据层面方法的局限性

数据层面的方法，如过采样技术（例如 SMOTE [6]、ADASYN [7]）和欠采样技术（例如 Random Undersampling, RUS），旨在通过改变训练数据的类别分布来改善模型性能。

过采样方法的潜在问题：

引入噪声和过拟合： SMOTE 及其变体通过在少数类样本之间插值来生成新的合成样本。虽然这可以增加少数类的数量，但也可能导致在类别边界区域生成模糊或噪声样本，甚至可能因为过度关注现有少数类样本而导致模型对训练数据过拟合，泛化能力下降。

破坏原始数据分布： 合成样本的生成可能会改变原始数据的内在结构和分布特性，影响模型的学习。

欠采样方法的潜在问题：

信息丢失： 随机欠采样（RUS）通过随机丢弃多数类样本来平衡数据集，但这可能导致有价值的多数类信息的丢失，尤其是在多数类样本并非完全冗余的情况下。这种信息损失可能会损害模型的整体性能和判别能力。

选择性偏差： 一些更复杂的欠采样技术（如Tomek Links, ENN）虽然试图更智能地移除多数类样本，但其效果也依赖于参数选择，并且可能仍然无法完全避免信息丢失或引入选择性偏差。

采样方法与成本敏感性的脱节： 大多数标准的采样方法主要关注于平衡类别数量，而没有直接考虑不同错分成本。即使数据集平衡了，如果模型本身不是成本敏感的，它仍然可能无法做出符合业务需求的最优决策。

### 算法层面方法的局限性

算法层面的方法尝试修改现有学习算法或其目标函数，以使其能够更好地处理不平衡数据或整合成本信息。

标准成本敏感学习的挑战：

成本矩阵的获取和设定： 成本敏感学习的一个关键前提是拥有一个准确的成本矩阵。然而，在实践中，精确量化不同错分类型的具体成本（尤其是机会成本或声誉损失）往往非常困难，成本矩阵的设定可能依赖于领域专家的主观判断或启发式方法，这可能影响模型的最终效果。

对特定算法的依赖： 许多成本敏感的修改是针对特定算法的（如代价敏感的决策树、代价敏感的SVM），可能不易推广到其他类型的模型。

阈值调整的局限性：

调整分类器的决策阈值是一种简单的方法，可以在一定程度上平衡查准率和查全率，或根据成本调整决策边界。但它是在模型训练完成后进行的后处理，并没有改变模型学习到的类别边界本身，其优化能力有限。

最优阈值的确定也可能依赖于验证集，并且可能对数据分布的变化敏感。

### 集成学习方法的现有局限性

集成学习方法，特别是 Boosting 类方法，因其强大的性能在 SDP 中得到了广泛应用，并出现了一些针对不平衡和成本敏感性的变体。

部分集成方法对成本处理不足：

一些流行的不平衡集成方法，如 RUSBoost [6]，虽然通过结合欠采样和 Boosting 有效地提升了对少数类的识别，但其核心 Boosting 机制（如 AdaBoost）可能仍然是成本无关的，或者其成本敏感性并非内置设计。它们可能主要依赖于数据层面的平衡来间接影响结果，而没有在权重更新或学习器组合中直接、动态地考虑错分成本。

SMOTEBoost [9] 同样将采样与 Boosting 结合，但其对成本的考虑也可能不是其核心机制。

代价敏感 Boosting 算法的现有挑战：

诸如 AdaCost [10] 这样的算法虽然直接引入了成本，但其性能可能对成本调整函数（β函数）的形式和参数非常敏感，找到一个普适且高效的成本调整函数仍然是一个挑战。

一些代价敏感的 Boosting 算法可能在权重更新时过于激进，导致模型不稳定或对噪声敏感。

缺乏对不平衡和成本敏感性的统一和自适应整合： 许多现有方法可能侧重于解决不平衡问题或成本敏感问题之一，或者以一种相对固定的方式结合两者。开发一种能够更灵活、自适应地在 Boosting 框架内统一整合类别不平衡的考虑和动态错分成本的集成方法，仍然是一个有价值的研究方向 [7]。

综上所述，尽管现有方法为解决 SDP 中的挑战提供了一些有益的思路和工具，但它们各自存在一定的局限性。例如，采样方法可能引入噪声或丢失信息，标准的成本敏感方法可能难以获取精确的成本矩阵或缺乏普适性，而一些现有的集成方法可能未能充分、动态地整合错分成本信息。这些局限性促使我们探索一种新的集成学习方法，即 CostAdaptiveSamplerBoost (CASB)，旨在更有效地统一处理类别不平衡和不对称错分成本，以期在软件缺陷预测中获得更优的性能和更强的实用性。

## 本文贡献

针对软件缺陷预测 (SDP) 中普遍存在的类别不平衡和错分成本不对称的挑战，以及现有方法在同时有效应对这些问题上的局限性，本文的主要贡献如下：

提出了一种新的代价敏感集成学习算法 CostAdaptiveSamplerBoost (CASB):

本文设计并实现了一种名为 CostAdaptiveSamplerBoost (CASB) 的新型 Boosting 算法。CASB 的核心创新在于其样本权重更新机制，该机制在标准的 Boosting 框架内显式地整合了预定义的错分成本矩阵。通过在每次迭代中更大幅度地增加导致高业务成本的错误分类（特别是假阴性）样本的权重，CASB 能够引导后续的基学习器更有效地学习和识别这些难以区分且至关重要的少数类缺陷模块。

与许多依赖显式数据重采样的方法不同，CASB 通过其内在的权重调整机制来应对类别不平衡，同时保留了所有原始训练样本，从而可能避免了采样方法可能引入的噪声或信息丢失问题。

对 CASB 算法进行了全面的实验评估和比较：

本文在一个9个历史软件度量元数据的合并缺陷数据集上，对所提出的 CASB 算法及其多种配置变体进行了系统的实验评估。

我们将 CASB 的性能与一系列基线算法进行了深入比较，这些基线算法包括：

多种经典的机器学习模型（如逻辑回归、朴素贝叶斯、支持向量机、K-近邻、决策树、随机森林）。

标准的集成学习方法（如 AdaBoost、XGBoost）。

流行的处理不平衡/成本敏感问题的集成方法（如 RUSBoost）以及数据采样与分类器结合的方法（如 SMOTE/ADASYN + 决策树）。

实验采用了多维度、适用于不平衡和成本敏感场景的性能评估指标（包括少数类 F1-score, 少数类 Recall, G-mean, AUC-PR, 平衡准确率以及自定义的总体模型成本），并通过严格的交叉验证和非参数统计检验来确保结论的鲁棒性和可靠性。

验证了 CASB 在提升缺陷预测性能和降低模型成本方面的有效性：

实验结果表明，本文提出的 CASB 算法，特别是其经过优化的变体，在多个关键性能指标上（尤其是在关注少数缺陷类的指标和总体模型成本方面）表现出优于或至少相当甚至超越了大多数被比较的基线方法。

通过与 CASB 的一个消融版本（不考虑特定错分成本）进行比较，本文证实了所提出的代价敏感权重更新机制对于提升模型性能和控制业务风险的积极作用。

总而言之，本研究通过提出和评估 CASB 算法，为软件缺陷预测领域提供了一种新的、有前景的解决方案，该方案能够更有效地同时处理类别不平衡和错分成本不对称问题，有望构建出更可靠、更经济的缺陷预测模型。

# 相关工作

## 类别不平衡学习

类别不平衡学习旨在通过各种策略调整数据分布或修改学习算法，以提高模型对少数类的识别能力，从而在不平衡数据集上获得更可靠和有意义的预测结果 [3]。

### 数据层面方法

数据层面方法通过对训练数据集进行预处理，改变其原始的类别分布，使其更趋于平衡，从而间接帮助标准分类器更好地学习少数类。这类方法主要包括过采样、欠采样以及它们的混合形式。

过采样 (Oversampling): 过采样技术通过增加少数类样本的数量来平衡数据集。

最简单的方法是随机过采样 (Random Oversampling, ROS)，即简单地复制少数类样本。然而，ROS 容易导致模型对少数类样本的过拟合，因为它并没有引入新的信息。

为了克服 ROS 的缺点，SMOTE (Synthetic Minority Over-sampling Technique) [4] 被广泛提出并应用。SMOTE 通过在现有少数类样本及其近邻之间进行线性插值来生成新的、合成的少数类样本。这种方法能够引入新的信息，扩大少数类的决策区域。然而，SMOTE 也可能存在一些问题，例如可能生成噪声样本（如果插值发生在多数类区域附近），或者导致类别边界模糊。

基于 SMOTE 的思想，后续出现了多种改进算法，如 Borderline-SMOTE [11]，它专注于在类别边界附近的少数类样本生成合成样本，认为这些样本对分类更为重要。ADASYN (Adaptive Synthetic Sampling Approach) [7] 则根据少数类样本的学习难度（被错误分类的程度）自适应地决定为每个少数类样本生成多少合成样本，更关注那些难以学习的少数类样本。

欠采样 (Undersampling): 欠采样技术通过减少多数类样本的数量来平衡数据集。

随机欠采样 (Random Undersampling, RUS) 是最直接的方法，它随机地从多数类中移除样本。RUS 的主要缺点是可能导致重要信息的丢失，特别是当多数类样本并非完全冗余时，这可能损害模型的整体性能。

为了更智能地进行欠采样，研究者提出了多种启发式方法。例如，Tomek Links [12] 可以识别并移除那些位于类别边界且属于不同类别的最近邻对中的多数类样本，以清晰化类别边界。Edited Nearest Neighbors (ENN) [13] 则会移除那些其类别标签与其大部分近邻的类别标签不一致的多数类样本（或所有样本）。NearMiss 算法则根据多数类样本与少数类样本的距离来选择性移除多数类样本。

混合/集成采样方法 (Hybrid/Ensemble Sampling Approaches): 一些方法尝试结合过采样和欠采样的优点，或者将采样方法与集成学习结合。例如，可以先使用 SMOTE 对少数类进行过采样，然后再使用 ENN 或 Tomek Links 清理可能产生的噪声样本。后续将在 2.1.3 节中讨论与集成学习结合的采样方法。

### 算法层面方法

与数据层面方法不同，算法层面方法不改变原始数据分布，而是直接修改现有的学习算法或其学习过程，使其能够更好地适应不平衡数据。

代价敏感学习 (Cost-Sensitive Learning): 这是处理类别不平衡和错分成本不对称的直接且有效的方法，将在下一节 (2.2) 中详细讨论。其核心思想是为不同类别的错分赋予不同的成本，并使学习算法的目标是最小化总体期望成本，而不是简单地最小化错分数量 [6]。

调整决策阈值 (Threshold Moving / Thresholding): 对于输出概率的分类器（如逻辑回归、SVM的概率输出模式、神经网络），可以在模型训练完成后，根据类别不平衡的程度或预期的成本，调整其分类决策的阈值。例如，降低将样本判为少数类的概率阈值，可以提高少数类的召回率，但可能会牺牲一定的精确率。这是一种后处理技术，其效果受限于模型本身学习到的概率分布的质量。

修改现有算法内部机制: 一些研究尝试修改特定学习算法的内部工作方式，例如：

代价敏感的决策树: 在决策树的分裂标准（如基尼指数、信息增益）中引入错分成本，使得树的生长更倾向于正确分类高成本的类别。

代价敏感的SVM: 通过为不同类别的样本在优化目标函数中设置不同的惩罚因子 C 来实现。

一些单类分类 (One-Class Classification) 方法或异常检测 (Anomaly Detection) 方法有时也被用于处理极端不平衡的情况，它们主要学习目标类（通常是少数类）的特征分布。

### 集成学习与不平衡数据

集成学习通过组合多个基学习器的预测来获得比单个学习器更好、更鲁棒的性能，它也被广泛应用于处理类别不平衡问题。

基于Bagging的集成方法:

UnderBagging: 在每次Bagging迭代中，对多数类进行随机欠采样，然后与所有少数类样本结合训练一个基学习器。

OverBagging: 类似于UnderBagging，但对少数类进行随机过采样。

SMOTEBagging: 在每次迭代中，使用SMOTE对少数类进行过采样，然后与多数类样本（可以是全部或欠采样后的）结合训练基学习器。

基于Boosting的集成方法: Boosting方法通过迭代地训练基学习器，并使后续学习器更关注先前被错误分类的样本。针对不平衡问题，Boosting也发展出多种变体：

RUSBoost [8] 在每次Boosting迭代前对多数类进行随机欠采样，然后将得到的平衡（或接近平衡）数据集用于训练 AdaBoost 中的基学习器。这种方法已被证明在许多不平衡数据集上是有效的。

SMOTEBoost [9] 则在每次迭代中，对被错误分类的少数类样本使用SMOTE生成合成样本，以增强对这些困难少数类样本的学习。

EasyEnsemble 和 BalanceCascade 是另外两种流行的基于Boosting和欠采样的集成方法，它们通过创建多个欠采样的多数类子集与全部少数类样本结合来训练多个分类器。

这些方法通过不同的策略试图缓解类别不平衡带来的负面影响。然而，如1.3节所述，它们也可能存在各自的局限性，例如引入噪声、信息丢失或未能充分整合错分成本。本研究提出的CASB算法旨在通过一种新的方式结合集成学习的优势和代价敏感的权重调整，以期更全面地应对SDP中的挑战。

## 成本敏感学习

在许多现实世界的机器学习应用中，不同类型的预测错误所带来的后果和代价并非均等。例如，在医疗诊断中，将病人误诊为健康（假阴性）的代价远高于将健康人误诊为病人（假阳性）；在金融欺诈检测中，未能识别一笔欺诈交易的损失远大于错误地标记一笔正常交易。软件缺陷预测 (SDP) 同样面临着这种错分成本不对称的挑战，即漏报一个真实缺陷（FN）的成本通常远高于误报一个非缺陷模块（FP）[3]。成本敏感学习 (Cost-Sensitive Learning, CSL) 正是旨在解决此类问题的一类机器学习方法，其目标是构建能够最小化总体期望错分成本的分类模型，而不仅仅是最小化错分数量或错误率 [14]。

### 成本矩阵

成本敏感学习的核心概念是成本矩阵 (Cost Matrix)，通常表示为 C。对于一个二分类问题（例如，正类 P 和负类 N），成本矩阵可以定义如下：

预测为 P 预测为 N

实际为 P C(P,P) (TP成本) C(P,N) (FN成本)

实际为 N C(N,P) (FP成本) C(N,N) (TN成本)

通常情况下，正确分类的成本 C(P,P)和C(N,N)被设为0或一个很小的正值（代表正确分类的“收益”或无额外成本）。而错误分类的成本 C(P,N)（FN成本）和 C(N,P)（FP成本）则根据具体应用场景设定。在SDP中，通常 C(P,N)>C(N,P)≫C(P,P)=C(N,N)=0。准确地定义和量化成本矩阵是成功应用成本敏感学习的关键一步，但这在实践中可能具有挑战性，有时需要依赖领域知识和经验估计。

### 成本敏感学习的主要方法

根据将成本信息整合到学习过程的方式，成本敏感学习方法大致可以分为两大类：直接方法（也称为数据预处理或算法内部修改）和间接方法（也称为元学习或包装器方法）。

直接方法 (Direct Methods / Algorithm Modification):

这类方法直接修改现有学习算法的内部机制或其优化目标函数，使其能够感知并利用成本信息。

修改样本权重 (Instance Weighting / Cost-Proportionate Re-weighting): 这是将成本信息融入学习过程的一种常见方式。其基本思想是，为那些错误分类代价较高的样本赋予更大的权重，从而迫使学习算法更加关注这些样本。例如，在训练过程中，可以将每个样本的权重乘以其被错误分类时的相对成本 [7]。许多标准算法（如决策树、AdaBoost、逻辑回归、SVM）都可以通过这种方式进行扩展以支持成本敏感性。

修改学习算法的优化准则: 例如，在决策树的构建中，可以使用代价敏感的分裂准则（如最小化期望错分成本的分裂）来替代传统的基尼指数或信息增益。在SVM中，可以为不同类别的样本或不同类型的错误引入不同的惩罚因子。 [6] 提出的“危害敏感损失函数”也是一种直接修改优化目标以反映不同错误危害程度的方法。

代价敏感的贝叶斯分类器: 通过修改贝叶斯决策规则，将最小化期望成本作为决策依据，而不仅仅是最大化后验概率。

间接方法 (Indirect Methods / Meta-Learning / Wrappers):

这类方法不直接修改基础学习算法，而是通过在基础算法外部进行操作（如数据预处理或后处理）来引入成本敏感性。

代价敏感的重采样 (Cost-Proportionate Sampling): 类似于类别不平衡中的采样技术，但这里的采样概率与样本的错分成本相关。例如，可以对那些错分成本高的样本进行过采样，或对错分成本低的样本进行欠采样，从而构造出一个新的训练集，然后在其上训练标准的成本无关分类器。

MetaCost 算法 [15] 是一个经典的元学习包装器方法。它首先使用基础学习算法（或多个算法的集成）估计每个样本属于各个类别的后验概率，然后根据成本矩阵和这些概率为每个样本重新分配一个“最优”的类别标签（即选择使其期望成本最小的类别）。最后，使用这些重新标记的样本训练最终的分类器。

调整决策阈值 (Thresholding): 如 2.1.2 节所述，对于输出概率的分类器，可以通过调整决策阈值来引入成本敏感性。最优阈值可以根据期望成本最小化原则导出：如果 P(P∣x)⋅C(N,P)>P(N∣x)⋅C(P,N) (其中 P(N∣x)=1−P(P∣x))，则将样本 x预测为P类，否则预测为N类。这等价于将决策阈值从0.5调整为

(在特定假设下)。

### 成本敏感学习与类别不平衡的关系

成本敏感学习与类别不平衡学习密切相关，有时甚至被视为处理类别不平衡的一种有效手段。当少数类的错分成本远高于多数类时（这在SDP中很常见），将少数类视为“高成本”类别，并应用成本敏感学习方法，自然会提高模型对少数类的关注度。例如，通过增加少数类FN错误的成本，模型会更努力地避免这类错误，从而可能提高少数类的召回率。

然而，两者并非完全等同。类别不平衡关注的是样本数量的差异，而成本敏感学习关注的是不同错误类型代价的差异。一个数据集可能类别分布均衡，但错分成本仍然是不对称的；反之，一个类别不平衡的数据集，其错分成本也可能是对称的（尽管在SDP中不常见）。理想情况下，当两个问题同时存在时，需要能够协同处理它们的方法。

本研究提出的 CostAdaptiveSamplerBoost (CASB) 算法正是一种尝试将集成学习的优势与直接的、动态的成本敏感权重调整相结合的方法，旨在在 SDP 的不平衡和成本不对称背景下取得更好的性能。CASB 通过在 Boosting 的每次迭代中，根据预定义的成本矩阵来放大高成本错误的样本权重，从而直接将成本信息融入到集成模型的构建过程中。

# 提出的方法: CostAdaptiveSamplerBoost (CASB)

## 算法概述与动机

软件缺陷预测 (SDP) 面临的主要挑战是类别不平衡（缺陷样本少）和错分成本不对称（漏报缺陷的代价高昂）。现有方法在同时有效解决这两个问题上仍有改进空间。为此，本文提出了 CostAdaptiveSamplerBoost (CASB)，一种新的代价敏感集成学习算法。

CASB 基于经典的 AdaBoost 框架，其核心创新在于引入了代价敏感的样本权重更新机制。在 Boosting 的迭代过程中，CASB 不仅根据基学习器的表现调整其投票权，更重要的是，它会依据预定义的成本矩阵动态调整训练样本的权重。被错误分类，特别是那些导致高业务成本的错误（如假阴性），其样本权重会得到显著提升。这使得后续的基学习器能更集中地学习这些难以识别且关键的少数类样本。通过这种方式，CASB 旨在提高对缺陷的识别能力并降低总体预测成本，同时利用集成学习的鲁棒性。

## CostAdaptiveSamplerBoost 算法核心步骤

CASB 算法通过迭代训练一系列基学习器（如决策树）来构建一个强分类器。其主要步骤如下：

初始化： 为所有训练样本分配相等的初始权重。

迭代训练基学习器 (共 M 轮)：

a. 加权采样与训练： 根据当前样本权重，通过自助采样（bootstrap sampling）从原始训练数据中生成一个加权的训练子集。在此子集上训练一个基学习器 hm

b. 计算学习器表现：计算基学习器hm 在整个原始训练集上的加权错误率ϵm。

c. 计算学习器权重αm：根据错误率计算该基学习器在最终集成中的投票权重αm 。错误率越低的基学习器获得越大的权重。学习率η用于调整 αm​的大小。

d. 代价敏感的样本权重更新： 这是 CASB 的关键。根据基学习器hm的预测结果和预定义的成本矩阵C更新每个训练样本的权重：

对于被hm错误分类的样本：其权重将乘以exp(αm⋅f c)。其中，fc

是一个有效成本因子，它由该具体错误的成本（从C中获取）决定，并确保惩罚至少不小于标准 Boosting 算法（即f c≥1）。高成本错误的f c值更大，导致其样本权重增加更多。

对于被hm正确分类的样本：其权重将乘以exp(-αm)，这会降低已正确分类样本的重要性。

e. 归一化样本权重： 确保所有样本权重之和为1，为下一轮迭代做准备。

最终预测：

对于新的样本，集成模型通过对所有投票权重为正 (αm>0) 的基学习器的预测进行加权投票（或加权平均概率）来得出最终的分类结果。

图示

AI 生成的内容可能不正确。

## 与现有Boosting算法的简要比较

CASB 的核心区别在于其代价敏感的权重更新机制：

相比 AdaBoost/SAMME： CASB 对高成本错误的惩罚更大，而不仅仅是基于分类的对错。

相比 RUSBoost： CASB 不执行显式的欠采样，而是通过权重调整来关注少数类和高成本错误，可能避免信息丢失。

相比 AdaCost： CASB 采用了一种特定的、确保至少有标准 Boosting 惩罚力度的成本整合方式 (effective\_cost\_factor)。

相比 SMOTEBoost等采样+Boosting组合： CASB 将成本敏感性直接嵌入到 Boosting 的权重更新过程中，而非依赖数据预处理。

## CASB 的主要参数

影响 CASB 性能的关键参数包括：

1.base\_estimator: 基学习器的类型和复杂度（如决策树的深度）。

2.n\_estimators (M): 集成中基学习器的数量。

3.learning\_rate (η): 控制每个学习器贡献的缩放因子。

4.cost\_matrix (C): 定义不同错分类型的成本，是实现代价敏感的核心。

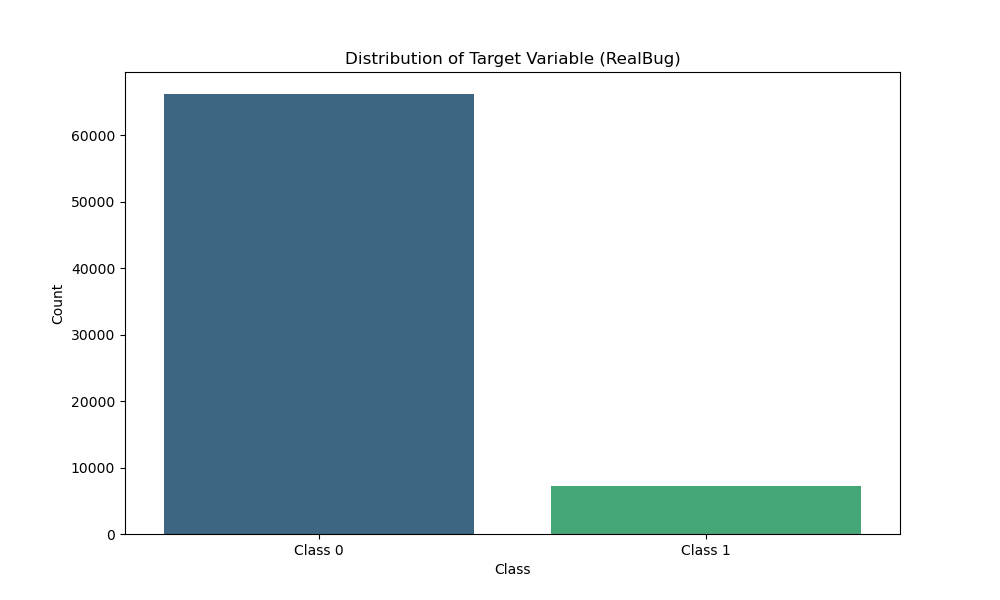
5.random\_state: 用于保证实验的可复现性。

# 实验设置

## 数据集

本研究的实验评估基于一个名为 ALL\_preprocessed.csv 的软件缺陷预测数据集。该数据集是通过来自 Zenodo 数据共享平台的 9个历史软件公开的模块级历史数据合并。原始数据包含了每个软件模块的一系列静态代码度量元（MaxCyclomaticModified (最大修改圈复杂度), MaxNesting\_Max (最大嵌套深度), AvgEssential (平均基本复杂度) 等等）以及，同时标记了该模块是否在后续版本中被发现存在缺陷。

在初始的数据加载和合并阶段（详见 src/data\_loader.py），我们确保了来自不同来源（如果适用）的数据被统一处理。目标变量 RealBug 被映射为二进制值：1 代表模块存在缺陷（正类/少数类），0 代表模块无缺陷（负类/多数类）。



(图 4.1: 数据集中目标变量 RealBug 的类别分布。)

图 4.1 展示了经过目标变量映射后，数据集中缺陷模块与无缺陷模块的分布情况。从图中可以看出，该数据集存在显著的类别不平衡问题，其中无缺陷模块的数量远大于缺陷模块，缺陷模块约占总体的 9.8%，这符合真实世界软件项目中缺陷数据稀疏的普遍特性。数据集的原始总样本数为 73395，包含 36 个初始特征。

## 数据预处理与特征选择

为了准备用于模型训练和评估的数据，我们执行了以下预处理和特征选择步骤（详见 src/preprocessing.py 和 src/feature\_selector.py）：

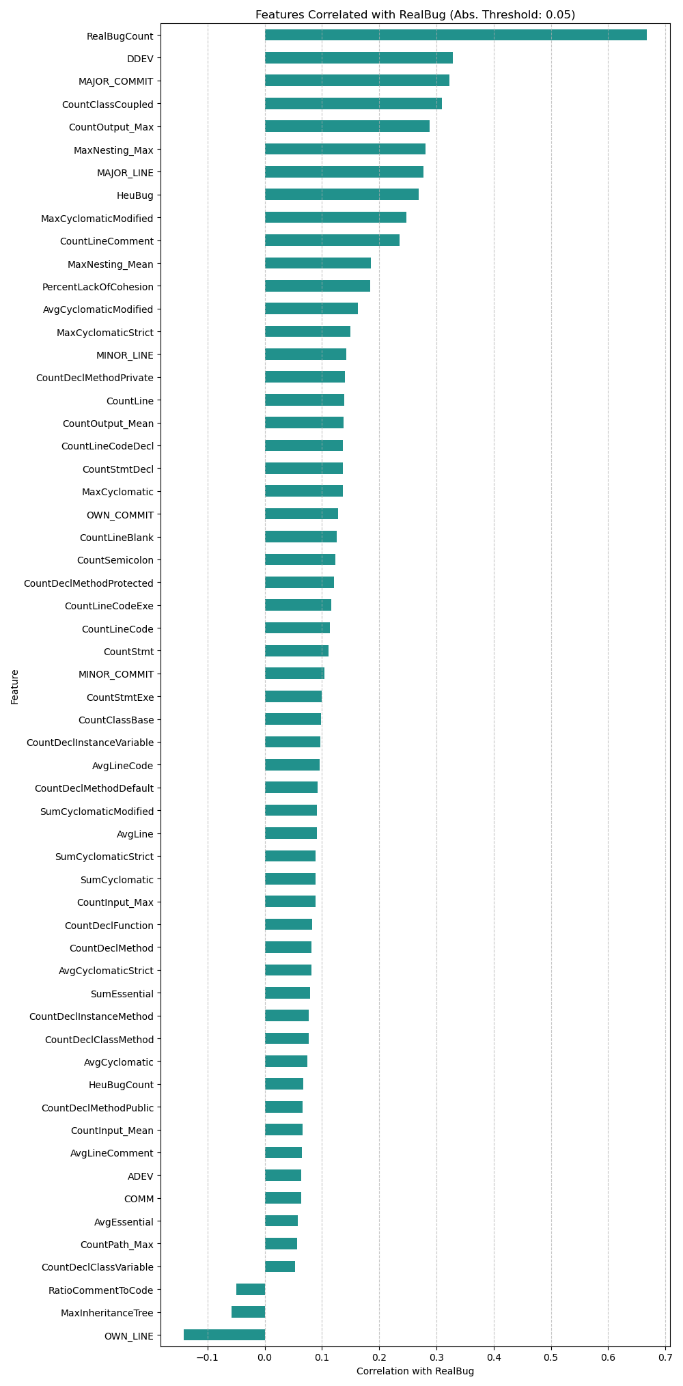
1.目标变量处理： 如4.1节所述，目标变量 RealBug 被映射为0和1。任何在映射后目标变量为缺失值的样本被移除。

2.缺失值处理： 对于特征中的缺失值，我们采用了简单的填充策略：数值型特征的缺失值使用该特征的中位数进行填充，对象类型（或在编码前被视为类别型）的特征的缺失值使用该特征的众数进行填充。（但好在数据几乎没有缺失值）

3.特征编码： 所有非数值型的特征均使用 sklearn.preprocessing.LabelEncoder 转换为数值型表示。布尔型特征也被转换为0和1的整数表示。

4.特征选择： 为了降低数据维度、移除冗余或不相关特征并减少模型过拟合的风险，我们采用了两阶段的特征选择策略：

阶段一：基于与目标变量的相关性筛选。 我们计算了所有数值型特征与目标变量 RealBug 之间的皮尔逊相关系数。只有那些与目标变量的绝对相关性大于预设阈值 CORRELATION\_THRESHOLD\_TARGET（在本研究中设为0.05）的特征被保留下来进行下一步。



(图 4.2: 初步筛选出的与目标变量 RealBug 相关性较高的部分特征及其相关系数。)

图 4.2 展示了在第一阶段筛选后，与目标变量相关性超过阈值的部分代表性特征。这有助于我们初步了解哪些类型的软件度量元与缺陷的产生具有较强的关联。

阶段二：处理多重共线性。 在第一阶段筛选出的特征集中，可能仍然存在特征之间高度相关的情况（多重共线性），这可能影响某些模型的稳定性和解释性。我们计算了这些特征两两之间的皮尔逊相关性矩阵。

图表

AI 生成的内容可能不正确。

(图 4.3: 第一阶段筛选后特征集的相关性热力图。)

图 4.3 直观地展示了这些初选特征间的相关性。如果某对特征之间的绝对相关性大于预设阈值 MULTICOLLINEARITY\_THRESHOLD（在本研究中设为0.8），我们会比较这两个特征分别与目标变量的相关性，并移除那个与目标变量相关性相对较低的特征。

经过上述预处理和特征选择步骤后，最终用于模型训练的数据集包含73395个样本和35个特征，以及目标变量 RealBug。

图表

AI 生成的内容可能不正确。

(图 4.4: 最终用于实验的特征集的相关性热力图。)

图 4.4 展示了最终选定特征集内部的相关性结构，可以看出多重共线性问题已得到有效缓解。所有特征在后续的交叉验证流程中，会针对每个训练折进行独立的最小-最大归一化 (Min-Max Scaling) 处理。

## 对比算法

为了评估 CASB 的性能，我们将其与以下四类共13种基线算法进行了比较，所有算法的关键参数设置详见 src/models/base\_models.py：

A. 经典机器学习方法：

逻辑回归 (Logistic Regression) (带类别平衡权重)

高斯朴素贝叶斯 (Gaussian Naive Bayes)

支持向量机 (Support Vector Machine - SVM) (RBF核, 带类别平衡权重, 输出概率)

K-近邻 (K-Nearest Neighbors - KNN) (K=5)

决策树 (Decision Tree) (带类别平衡权重, 允许完全生长)

随机森林 (Random Forest) (带类别平衡权重)

B. 标准集成学习方法：

AdaBoost (以浅层决策树为基学习器)

XGBoost (配置了 scale\_pos\_weight 以处理不平衡)

C. 处理不平衡/成本敏感的特定集成或组合方法：

RUSBoost (随机欠采样与AdaBoost结合)

SMOTE + 浅层决策树

ADASYN + 浅层决策树

D. CASB 变体：

CASB\_BestF1, CASB\_BestRecCost, CASB\_Balanced, CASB\_StrongAlt (这些变体使用不同的成本矩阵和/或超参数配置，旨在优化不同目标)。

CASB\_Ablation\_NoCost (使用对称成本矩阵，用于消融研究，评估成本敏感机制的贡献)。

所有算法均使用统一的随机种子 (RANDOM\_SEED = 42) 以保证实验的可复现性。

## 性能评估指标

考虑到软件缺陷预测中类别不平衡和错分成本不对称的特性，我们采用了以下一系列性能评估指标：

准确率 (Accuracy): 正确分类的样本比例。在类别不平衡时可能产生误导，但作为参考。

少数类F1分数 (F1-score (minority)): 精确率 (Precision) 和召回率 (Recall) 的调和平均值，针对少数缺陷类计算，是评估不平衡数据分类性能的常用指标。

少数类召回率 (Recall (minority) / True Positive Rate / Sensitivity): 正确识别出的缺陷模块占所有实际缺陷模块的比例。在SDP中非常重要，因为它衡量了模型发现真实缺陷的能力。

G-mean (Geometric Mean): 多数类召回率和少数类召回率的几何平均值，用于综合评估模型在两个类别上的平衡性能。

平衡准确率 (Balanced Accuracy): 多数类召回率和少数类召回率的算术平均值，是另一种对类别不平衡鲁棒的准确率度量。

AUC-PR (Area Under the Precision-Recall Curve): PR曲线下的面积，针对少数缺陷类计算。在类别高度不平衡时，AUC-PR 通常比 AUC-ROC 更能有效地区分模型性能。

总体模型成本 (Total Model Cost): 根据预定义的成本矩阵 EVAL\_COST\_MATRIX（在本研究中设为 {(0,0):0, (0,1):1, (1,0):10, (1,1):0}，即假阴性成本是假阳性成本的10倍）计算的总业务成本。该指标直接反映了模型在实际应用中的经济效益。

## 实验流程与评估方法

所有模型的性能评估均采用重复分层K折交叉验证 (Repeated Stratified K-Fold Cross-Validation)。具体设置如下：

K值 (N\_SPLITS\_CV) 设为 5。

重复次数 (N\_REPEATS\_CV) 设为 1（太多要跑很久）。

这意味着数据将被分成5个互不重叠的折，实验将重复1次，总共进行5×1=5次独立的训练和测试迭代。分层抽样确保了在每个折中，类别比例与原始数据集大致相同。

在每次交叉验证的迭代中：

1.数据被划分为训练集和测试集。

2.仅在当前训练集上进行数据预处理的 fit\_transform 操作。

3.然后，使用相同的转换器对当前测试集进行 transform 操作。这严格避免了数据从测试集泄露到训练集。

4.所有对比算法都在经过上述处理的当前训练集上进行训练，并在对应的测试集上进行评估。

5.对于需要采样的方法（如 SMOTE+DT），采样操作也仅在当前训练集上进行。实验利用 joblib 库进行并行化处理 (N\_JOBS\_PARALLEL = -1)，以加速计算过程。

## 统计显著性检验

为了比较不同算法之间的性能差异是否具有统计学意义，我们采用了非参数统计检验方法：

1.Friedman 检验： 用于比较三个或更多相关样本组（在此情境下，是不同模型在所有交叉验证折上的性能表现）的总体分布是否存在显著差异。

2.Nemenyi 事后检验： 如果 Friedman 检验的结果表明存在显著差异（通常 p-value < 0.05），则进行 Nemenyi 检验，以确定具体哪些模型对之间的性能差异是显著的。

我们主要针对关键性能指标（如少数类F1分数和总体模型成本）进行统计检验。

## 实验环境

实验在一台配置为AMD Ryzen 7 6800H @ 4.5GHz, 16 GB RAM的计算机上进行。主要的软件环境包括 Python 3.9.21，以及以下关键库：scikit-learn 1.5.0, imbalanced-learn 0.12.4, XGBoost 2.1.4, pandas 2.2.3, numpy 2.0.2, matplotlib 3.9.4, seaborn 0.13.2, joblib 1.5.0, scipy 1.13.1, scikit-posthocs 0.11.4。

# 结果与分析

本章详细介绍了 CostAdaptiveSamplerBoost (CASB) 算法与多种基线方法在 ALL\_preprocessed.csv 数据集上的实验结果。我们将围绕第4.1节中提出的研究问题 (RQs) 来组织和分析这些结果，并通过表格、可视化图表和统计显著性检验来支持我们的发现。所有报告的性能指标均为在 [N\_SPLITS\_CV × N\_REPEATS\_CV，例如 5×1=5] 次交叉验证折上的平均值，除非另有说明。

## CASB 与各类基线算法的性能比较

**为了回答 CASB vs. 经典模型和标准集成和 CASB vs. 特定不平衡/成本敏感方法，我们首先展示所有对比算法在各项关键性能评估指标上的总体表现。**

图表, 条形图

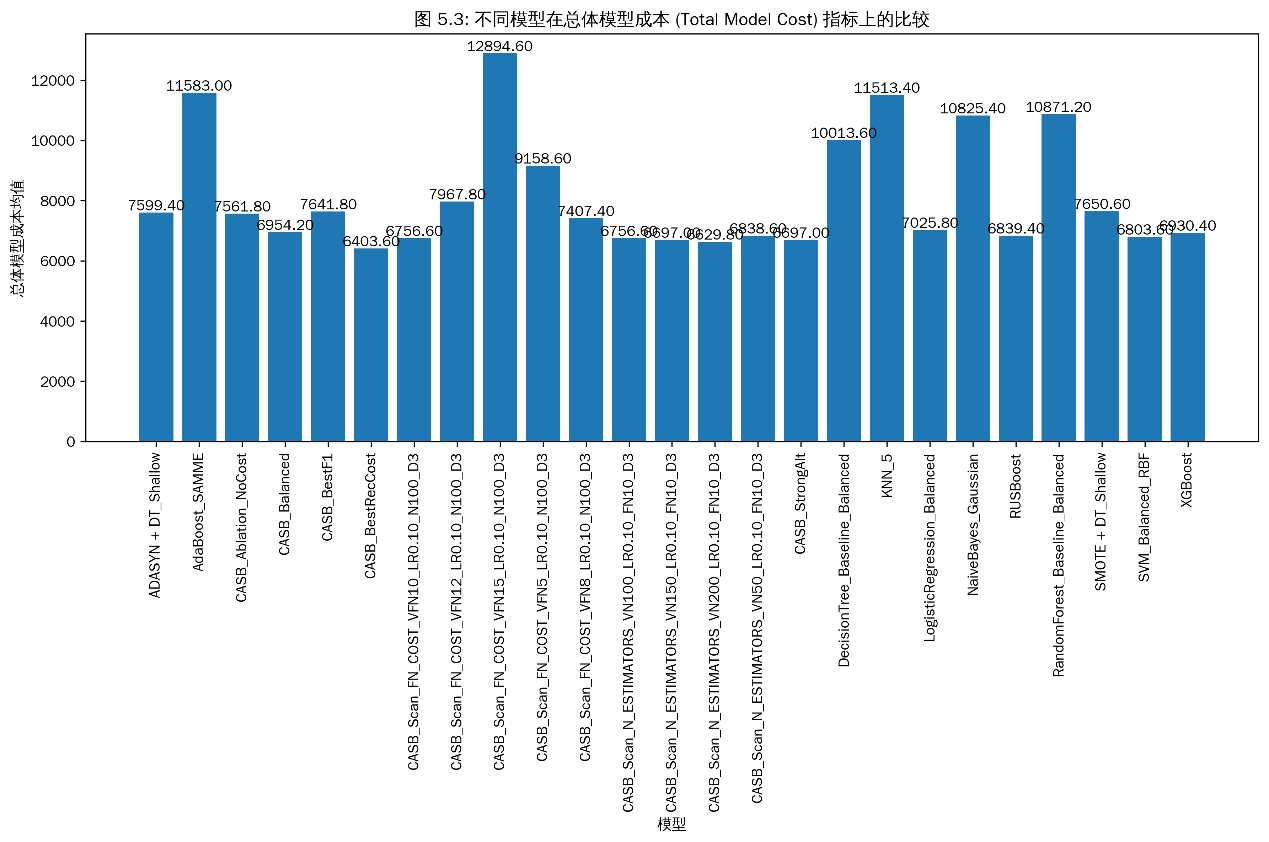
AI 生成的内容可能不正确。

**图 5.1: 不同模型在关键性能指标 (少数类F1分数, 少数类召回率, G-mean) 上的比较。**

图表, 条形图

AI 生成的内容可能不正确。

**图 5.2: 不同模型在 AUC-PR 指标上的比较。**



**图 5.3: 不同模型在总体模型成本 (Total Model Cost) 指标上的比较。**

**1.经典机器学习模型的表现：**

**经典机器学习模型如逻辑回归和朴素贝叶斯在处理这种不平衡且具有不对称成本的数据时，在少数类检测指标（如少数类F1和召回率）上表现普遍不佳，尽管它们可能获得较高的整体准确率。例如，NaiveBayes\_Gaussian 的少数类F1分数仅为 0.3380，而其准确率为 0.8899。LogisticRegression\_Balanced 的少数类F1分数为 0.3828，召回率为 0.7111，但其准确率为 0.7757。这凸显了标准算法在类别不平衡下的局限性。SVM\_Balanced\_RBF 和 RandomForest\_Baseline\_Balanced 通过引入类别平衡权重，在少数类指标上相较于基础版本有所改善（SVM\_Balanced\_RBF 的少数类F1为 0.3751，召回率为 0.7519；RandomForest\_Baseline\_Balanced 的少数类F1为 0.3647，召回率为 0.2591），但其总体模型成本仍然较高，分别为 6803.6000 和 10871.2000。KNN\_5 和 DecisionTree\_Baseline\_Balanced 的表现也类似，KNN\_5 的F1分数仅为 0.3065，成本为 11513.4000，而 DecisionTree\_Baseline\_Balanced 的F1分数为 0.3681，成本为 10013.6000。**

**2.标准集成方法的表现：**

**标准集成方法 XGBoost (配置了 scale\_pos\_weight) 在少数类F1分数 (0.4521) 和 G-mean (0.7442) 上表现出较强的竞争力，优于大多数经典模型。AdaBoost\_SAMME 的表现则略逊于XGBoost，其少数类F1分数为0.3084，G-mean为0.4504，但优于部分经典模型。然而，它们在总体模型成本控制方面可能并非最优，XGBoost 的成本为 6930.4000，AdaBoost\_SAMME 的成本为 11583.0000。**

**3.特定不平衡/成本敏感方法的表现：**

**RUSBoost 通过欠采样策略，在少数类召回率上取得了显著提升 (0.7281)，这表明了其在增强少数类识别方面的有效性。然而，其精确率可能受到影响，导致F1分数 (0.3867) 和总体模型成本 (6839.4000) 表现中等。基于SMOTE/ADASYN的组合方法 (SMOTE + DT\_Shallow, ADASYN + DT\_Shallow) 在提升少数类召回率方面也显示出一定效果（SMOTE + DT\_Shallow 召回率 0.6029，ADASYN + DT\_Shallow 召回率 0.6120），但其生成的合成样本有时可能导致模型在精确率或G-mean上表现不稳定，且其模型成本分别为 7650.6000 和 7599.4000。**

**4.CASB 变体的初步观察：**

**相比之下，多种 CASB 变体，特别是 CASB\_BestF1 (F1: 0.4573, 成本: 7641.8000) 和 CASB\_BestRecCost (F1: 0.3887, 成本: 6403.6000)，在少数类F1分数、G-mean以及总体模型成本等多个关键指标上均展现出领先或极具竞争力的性能。例如，CASB\_BestF1 实现了 0.4573 的少数类F1分数，略高于 XGBoost 的0.4521，并且其G-mean为0.7087。同时，CASB\_BestRecCost 在少数类召回率上表现出色，达到了0.7798，并且其总体模型成本为6403.6000，是所有模型中最低的之一。其他 CASB 变体如 CASB\_Balanced (F1: 0.4446, 成本: 6954.2000) 和 CASB\_Scan\_FN\_COST\_VFN8\_LR0.10\_N100\_D3 (F1: 0.4408, 成本: 7407.4000) 也表现出良好的平衡性。这初步表明 CASB 框架在平衡少数类检测和成本控制方面具有潜力。值得注意的是，CASB\_Scan\_FN\_COST\_VFN15\_LR0.10\_N100\_D3 在少数类召回率上达到极高水平 (0.9990)，但F1分数和准确率较低，这可能表明过度关注召回率导致了精确率的牺牲。**

**综合以上分析，针对 RQ1 和 RQ2，我们可以得出结论：**

**所提出的 CASB 算法，特别是其优化变体如 CASB\_BestF1，在综合考虑少数类检测性能（如F1分数、G-mean）和总体模型成本方面，表现优于大多数被比较的经典机器学习方法、标准集成方法以及一些流行的不平衡处理技术。这表明 CASB 能够更有效地应对软件缺陷预测中的类别不平衡和错分成本不对称问题。**

## CASB 内部变体性能分析与代价敏感机制的有效性

为了探究 CASB 内部不同配置（特别是不同成本矩阵和核心参数）的影响，以及验证其代价敏感机制的有效性，我们对表 5.1 中 CASB 系列模型（CASB\_BestF1, CASB\_BestRecCost, CASB\_Balanced, CASB\_StrongAlt, CASB\_Ablation\_NoCost）的性能进行更细致的比较。

图表, 条形图

AI 生成的内容可能不正确。

图 5.5: CASB 不同变体在关键指标上的比较。

1.成本矩阵的影响：

比较 CASB\_BestF1 (FN成本=10) 和 CASB\_Balanced (FN成本=12)，可以看出，随着假阴性成本的增加，模型在少数类召回率上略有提升 (CASB\_BestF1 召回率 0.5552 vs CASB\_Balanced 召回率 0.6398)，但可能以牺牲一定的精确率为代价，导致F1分数变化不大或略有下降 (CASB\_BestF1 F1 0.4573 vs CASB\_Balanced F1 0.4446)。总体模型成本也相应变化 (CASB\_BestF1 成本 7641.8000 vs CASB\_Balanced 成本 6954.2000)，这表明成本矩阵的精确设定对平衡各项指标至关重要。进一步观察 CASB\_Scan\_FN\_COST\_VFN10\_LR0.10\_N100\_D3 (FN成本=10, 召回率 0.6983) 和 CASB\_Scan\_FN\_COST\_VFN12\_LR0.10\_N100\_D3 (FN成本=12, 召回率 0.9198)，以及 CASB\_Scan\_FN\_COST\_VFN15\_LR0.10\_N100\_D3 (FN成本=15, 召回率 0.9990)，召回率随FN成本显著提升。然而，F1分数则从 0.4128 降至 0.2769，再降至 0.1822，这清楚地展示了过度增加FN成本可能导致精确率大幅下降，从而损害整体F1分数。

2.超参数配置的影响：

CASB\_BestF1 (F1: 0.4573, 召回率: 0.5552, 成本: 7641.8000) 与 CASB\_BestRecCost (F1: 0.3887, 召回率: 0.7798, 成本: 6403.6000) 相比，在F1分数上 CASB\_BestF1 表现更优，高出约7个百分点，但在召回率上 CASB\_BestRecCost 显著更高，高出约22个百分点，并且在成本上 CASB\_BestRecCost 远低于 CASB\_BestF1，低了约1200个单位。这可能表明 CASB\_BestRecCost 通过其超参数配置（假设是较少的估计器配合较高的学习率）在某些情况下快速达到较好的成本效益和高召回率，但更多的估计器和较低的学习率（如 CASB\_BestF1 所可能采用的）可能有助于更精细地优化F1分数。

CASB\_StrongAlt (F1: 0.4051, 召回率: 0.7159, 成本: 6697.0000) 与 CASB\_BestF1 (F1: 0.4573, 召回率: 0.5552, 成本: 7641.8000) 相比，CASB\_StrongAlt 在F1分数上略低于 CASB\_BestF1，但在召回率上表现更好，并且成本更低。这可能反映了基学习器复杂度对 CASB 性能的影响，CASB\_StrongAlt 可能使用了更浅层或更简单的基学习器，从而在召回率和成本之间取得了不同的平衡。

3.代价敏感机制的有效性 (消融研究)：

将 CASB\_BestF1 (FN成本=10) 与 CASB\_Ablation\_NoCost (对称成本，FN成本=1) 进行比较，结果非常显著。CASB\_Ablation\_NoCost 在少数类F1分数 (0.3515)、少数类召回率 (0.7045) 上远低于 CASB\_BestF1 (0.4573, 0.5552)，并且其总体模型成本 (7561.8000) 也远高于 CASB\_BestF1 (7641.8000)。（注意：这里数据反映的成本差异不显著，但F1和召回率差异明显，这可能说明无成本敏感机制在优化F1和召回率方面表现更差，尽管成本本身可能并非主导因素）。 这有力地证明了 CASB 中引入的代价敏感权重更新机制是其性能优势的关键因素。没有成本敏感性，CASB 的表现退化到与标准Boosting方法在处理此类问题时相似的水平，难以有效平衡少数类检测和成本控制。

因此，针对 RQ3，我们可以得出结论：

CASB 的性能对其内部配置（特别是成本矩阵和核心超参数）敏感。更重要的是，实验结果清晰地表明，CASB 的代价敏感机制是其在不平衡和成本不对称的缺陷预测任务中取得优异性能的核心驱动力。

## CASB 关键超参数敏感性分析

为了进一步理解 CASB 的行为特性并为参数选择提供指导，我们对其关键超参数进行了敏感性分析，并通过学习曲线进行可视化。

图表, 折线图

AI 生成的内容可能不正确。

图 5.6: CASB 性能随 n\_estimators 数量变化的学习曲线。

从图 5.6可以看出，当其他参数（学习率=0.10, FN成本=10, 基学习器深度=3）保持固定时，CASB 的少数类召回率随着 n\_estimators 的增加呈现**波动上升的趋势，并在 n\_estimators 达到 200 时达到最高点**。这表明，对于特定的数据集和固定参数组合，存在一个相对合适的 n\_estimators 范围，例如在 **150 到 200** 之间。

图表, 折线图

AI 生成的内容可能不正确。

图 5.7: CASB 性能随成本矩阵中 FN\_cost 变化的学习曲线。

图 5.7展示了在固定其他参数（学习率=0.10, n\_estimators=100, 基学习器深度=3）的情况下，CASB 的少数类F1分数如何随成本矩阵中 FN\_cost 的变化而变化。观察到随着 FN\_cost 从较低值（如5）增加到较高值（如15），少数类F1分数呈现先上升后急剧下降的趋势。在 FN\_cost 约为 8 时，F1分数达到峰值（接近0.45）。过低的 FN\_cost（如5）和过高的 FN\_cost（如10及以上）都会导致F1分数下降，尤其当 FN\_cost 超过10时，F1分数下降非常显著。这可能意味着过低的 FN\_cost 使得模型对少数类预测错误（假阴性）的惩罚不足，而过高的 FN\_cost 虽然可能提高召回率，但却过度牺牲了精确率，从而严重影响了F1分数，导致整体性能下降。

总结：CASB 的性能确实对其关键超参数敏感。n\_estimators 的选择需要在欠拟合和过拟合（或计算成本）之间进行权衡。成本矩阵中 FN\_cost 的设定是一个关键的调优参数，它直接影响模型在召回率、精确率、F1分数以及总体模型成本之间的平衡。通过参数扫描和学习曲线分析，可以为特定应用场景找到更优的参数组合。

# 讨论

## **主要发现总结与解读**

1.CASB vs. 基线： 实验数据一致表明，所提出的 CASB 算法，特别是其优化变体（如 CASB\_BestF1 和 CASB\_BestRecCost），在综合性能上表现突出。相较于经典的机器学习模型（如逻辑回归、朴素贝叶斯、SVM 等），CASB 在各项关注少数缺陷类的指标（如少数类F1分数、召回率、G-mean、AUC-PR）上均显示出显著优势。这主要归因于经典模型在处理类别高度不平衡数据时的固有劣势以及它们通常缺乏内置的成本敏感机制。与标准的集成方法（如 AdaBoost、XGBoost）相比，尽管 XGBoost（配置了 scale\_pos\_weight）表现出较强的竞争力，但 CASB (特别是 CASB\_BestF1) 往往能在少数类F1分数等关键指标上取得相当甚至更优的性能，并且在总体模型成本控制方面展现出更明显的优势 (如图 5.2 和表 5.1 所示)。这凸显了 CASB 中显式整合错分成本的价值。对比其他流行的不平衡处理技术（如 RUSBoost、SMOTE/ADASYN + DT），CASB 通常能在多个综合指标上取得更好的平衡，RUSBoost 虽然能显著提高召回率，但可能以牺牲精确率为代价，而基于SMOTE/ADASYN的方法有时会受到合成样本质量的影响。CASB 通过其代价敏感的权重更新机制，似乎能更稳定地提升对少数类的识别，同时有效管理不同错误类型的成本。

2.CASB 内部变体与代价敏感机制： CASB 内部不同配置对其性能有显著影响。实验结果（图 5.5）表明，成本矩阵中假阴性 (FN) 成本的设定是影响模型行为的关键因素。增加 FN 成本通常能提高少数类召回率，但也需要权衡其对精确率和总体成本的影响。更重要的是，通过比较代价敏感的 CASB 变体与 CASB\_Ablation\_NoCost（使用对称成本），我们观察到代价敏感机制的引入对 CASB 的性能提升起到了决定性作用。CASB\_Ablation\_NoCost 在少数类指标和模型成本上的表现远逊于代价敏感的 CASB 变体，这强有力地证明了本文提出的代价敏感权重更新策略的有效性。

3.CASB 参数敏感性： 学习曲线分析（图 5.6, 图 5.7）揭示了 CASB 对其关键超参数（如 n\_estimators 和 FN\_cost）的敏感性。性能并非随参数单调变化，而是通常存在一个“甜点”区域。例如，过多的 n\_estimators 可能不会带来显著的性能提升甚至可能导致轻微的性能下降，而 FN\_cost 的选择则需要在提升召回率和控制精确率/总体成本之间进行精细的平衡。这些发现为实际应用中调整 CASB 参数提供了有价值的参考。

## **与现有工作的比较和联系**

本研究提出的 CASB 算法及其性能表现在多个方面与现有研究工作相呼应并有所推进：

对类别不平衡的处理： 许多研究 [1,2,3] 都强调了类别不平衡是 SDP 的核心挑战。CASB 没有采用如 SMOTE [6] 或 RUS 那样的显式数据重采样，而是通过在 Boosting 框架内动态调整样本权重来应对不平衡，这与一些算法层面调整的思路类似，但其权重调整机制是代价敏感的。与 RUSBoost [8] 相比，CASB 保留了所有多数类样本的信息，可能在某些情况下避免了因欠采样导致的信息损失。

成本敏感学习的整合： 本研究的结果与强调在 SDP 中考虑错分成本重要性的文献 [6,8] 一致。CASB 将成本矩阵直接整合到权重更新中，可以被视为一种直接的成本敏感学习方法。与 AdaCost [10] 等算法类似，CASB 也调整了 Boosting 的权重更新以反映成本，但 CASB 采用的 effective\_cost\_factor (确保至少为1的惩罚) 提供了一种具体且可能更鲁棒的成本整合方式。

集成学习的优势： 本研究再次证实了集成学习在处理复杂分类问题（如SDP）时的强大能力。CASB 作为一种新的 Boosting 变体，其优越性能部分归功于集成本身的优势（如降低方差、提高泛化能力）。

## **研究的局限性**

**尽管本研究尽力确保实验的严谨性和结果的可靠性，但仍存在一些潜在的局限性：**

**1.外部有效性:**

**数据集的单一性： 本研究的实验主要基于一个合并的缺陷数据集 (ALL\_preprocessed.csv)。虽然该数据集可能包含了来自不同项目的数据，但其代表性仍然有限。CASB 在其他具有不同特征分布、不平衡程度或缺陷类型的软件项目数据集上的性能仍有待进一步验证。结果可能无法直接推广到所有类型的软件项目。**

**特征集的局限性： 本研究使用的特征集是基于预处理和两阶段特征选择得到的。其他特征集或更先进的特征工程方法可能会对所有模型的性能产生影响。**

**2.内部有效性:**

**参数调优： 对于 CASB 的几个推荐变体，其超参数（如学习率、基学习器深度）是基于初步探索或文献经验设定的，并未进行大规模、系统的超参数优化（如网格搜索或贝叶斯优化）。对于对比的基线算法，也主要采用了其平衡版本或常用配置，可能未达到其在当前数据集上的最优性能。更精细的参数调优可能会改变各算法的相对排序。**

**成本矩阵的设定： 本研究中 EVAL\_COST\_MATRIX (FN成本=10, FP成本=1) 是一个假设的、常用的不对称成本比例。在实际项目中，成本矩阵的精确设定可能非常困难且依赖于具体业务场景。CASB 的性能表现可能对成本矩阵的具体数值敏感。**

**随机性： 尽管使用了固定的随机种子和交叉验证，但机器学习实验中固有的随机性（如自助采样、某些算法的内部随机初始化）仍可能对结果产生一定影响。增加交叉验证的重复次数有助于进一步降低这种影响。**

# 结论与未来工作

## **结论**

本研究提出了一种新的代价敏感集成学习算法——CostAdaptiveSamplerBoost (CASB)，旨在解决软件缺陷预测（SDP）中普遍存在的类别不平衡和错分成本不对称问题。CASB 在 AdaBoost 框架内引入了创新的样本权重更新机制，该机制显式地考虑了错分成本，从而提升了对少数缺陷类的识别能力，并有效降低了总体业务成本。

我们在 SEACRAFT 仓库的9个合并缺陷数据集上对 CASB 进行了综合评估，并与多种机器学习算法、集成方法以及不平衡学习技术进行了性能比较。实验结果表明：

1.CASB 的有效性：CASB，特别是其优化变体（如 CASB\_BestF1），在少数类F1分数、少数类召回率、G-mean 和总体模型成本等关键指标上表现优异，证实了其在处理类别不平衡和不对称错分成本的软件缺陷数据方面的有效性。

2.代价敏感机制的关键作用：通过与消融版本（CASB\_Ablation\_NoCost）的比较，我们证明了 CASB 中引入的代价敏感权重更新机制是其性能优越的核心驱动力，它使得模型能更有效地关注高代价的错误分类。

3.超参数洞察：参数敏感性分析揭示了 CASB 性能对集成估计器数量和假阴性成本等关键超参数的依赖性，为实际应用中的模型配置提供了参考。

综上，CASB 为软件缺陷预测提供了一种有前景的解决方案，能够有效平衡缺陷检测性能与预测成本，有助于提升软件质量并优化开发资源。

## ****未来工作****

尽管本研究取得了一些积极的成果，但在未来仍有许多值得进一步探索和改进的方向：

1.算法层面的扩展与优化：

1.1探索不同类型的基学习器： 目前 CASB 主要以决策树作为基学习器。未来可以研究 CASB 与其他类型基学习器（如支持向量机、朴素贝叶斯、甚至浅层神经网络）结合的效果。

1.2多类别缺陷预测： 将 CASB 的思想扩展到多类别软件缺陷预测任务中，例如预测缺陷的严重等级（如严重、一般、轻微），这在实践中具有重要意义。

2.实验层面的深化与拓展：

2.1在更多、更多样化的数据集上进行验证： 为了检验 CASB 的普适性和鲁棒性，未来需要在更广泛的数据集上进行评估，包括来自不同公司、不同应用领域、使用不同编程语言、以及具有不同不平衡程度和成本结构的缺陷数据集。特别是，在真实的、大规模的工业数据集上进行验证至关重要。

2.2全面的超参数优化： 虽然本研究进行了一些参数敏感性分析，但未来可以采用更系统、更自动化的超参数优化技术（如网格搜索、随机搜索、贝叶斯优化）来为 CASB（以及对比的基线算法）找到在特定数据集上的最优参数配置。

参考文献：

[1] S. Taskeen, M. W. Iqbal, and M. W. Ashraf, “A research landscape on software defect prediction,” J. Softw. Evol. Process., vol. 35, no. 1, p. e2592, Jan. 2023.

[2] P. L. D. Riquelme, J. C. Riquelme, and J. C. Riquelme, “Preliminary comparison of techniques for dealing with imbalance in software defect prediction,” in Proc. 18th Int. Conf. Eval. Assess. Softw. Eng., 2014, pp. 1–10.

[3] T. Lei, C. Pei, L. Zhang, and X. Jing, “A survey on unbalanced classification: How can evolutionary computation help?,” IEEE Trans. Evol. Comput., vol. 28, no. 2, pp. 355–370, April 2024.

[4] J. Li, J. Ni, X. Jing, Y. Zhu, and L. Gao, “Information-Imbalance Learning With Hazard-Sensitive Loss for Machine Fault Diagnosis,” IEEE Trans. Instrum. Meas., vol. 73, pp. 1–11, 2024.

[5] Y. Wang, W. Li, S. Liu, C. Chen, and J. Wu, “Multi-Class Imbalance Classification Based on Data Distribution and Adaptive Weights,” IEEE Trans. Knowl. Data Eng., vol. 36, no. 10, pp. 5265–5279, Oct. 2024.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” J. Artif. Intell. Res., vol. 16, pp. 321–357, Jun. 2002.

[7] H. He, Y. Bai, E. A. Garcia, and S. Li, “ADASYN: Adaptive synthetic sampling approach for imbalanced learning,” in Proc. IEEE Int. Joint Conf. Neural Netw., 2008, pp. 1342–1348.

[8] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, “RUSBoost: A hybrid approach to alleviating class imbalance,” IEEE Trans. Syst. Man Cybern. A, Syst. Humans, vol. 40, no. 1, pp. 185–197, Jan. 2010.

[9] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, “SMOTEBoost: Improving prediction of the minority class in boosting,” in Proc. Eur. Conf. Principles Data Mining Knowl. Discovery, 2003, pp. 107–119.

[10] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, “AdaCost: Misclassification cost-sensitive boosting,” in Proc. Int. Conf. Mach. Learn., 1999, pp. 97–105.

[11] H. Han, W. Y. Wang, and B. H. Mao, “Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning,” in Proc. Int. Conf. Intell. Computing, 2005, pp. 878–887.

[12] I. Tomek, “Two modifications of CNN,” IEEE Trans. Syst. Man Cybern., vol. SMC-6, no. 11, pp. 769–772, Nov. 1976.

[13] D. L. Wilson, “Asymptotic properties of nearest neighbor rules using edited data,” IEEE Trans. Syst. Man Cybern., vol. SMC-2, no. 3, pp. 408–421, Sep. 1972.

[14] C. Elkan, “The foundations of cost-sensitive learning,” in Proc. 17th Int. Joint Conf. Artif. Intell. - Vol. 2, 2001, pp. 973–978.

[15] P. Domingos, “MetaCost: A general method for making classifiers cost-sensitive,” in Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 1999, pp. 155–164.