Software Engineering and Applications 软件工程与应用, 2024, 13(3), 346-357
Published Online June 2024 in Hans. https://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2024.133035

Hans汉斯

基于PCA-Smote-XGBoost的软件缺陷预测研究

曾子安，李英梅

哈尔滨师范大学计算机科学与信息工程学院，黑龙江 哈尔滨

收稿日期：2024年4月24日；录用日期：2024年6月21日；发布日期：2024年6月30日

摘　要

随着软件系统的复杂性日益增加，软件缺陷预测成为了确保软件质量的重要手段。本研究提出了一种基于PCA-Smote-XGBoost的软件缺陷预测模型，旨在提高缺陷预测的准确性和效率。本文采用主成分分析(PCA)进行数据降维，保留95%的方差，以减少特征数量并提取关键信息；利用Smote过采样方法解决数据不平衡问题；结合XGBoost算法构建预测模型，并通过实验验证模型的有效性。在软件缺陷预测常用数据集的十一个项目中，实验结果表明，该模型在软件缺陷预测方面相较于其他八种基准模型，具有最高的准确率ACC和F1，能够有效地辅助软件开发团队识别潜在的缺陷风险。

关键词

软件缺陷预测，PCA，Smote，XGBoost

Research on Software Defect Prediction Based on PCA-Smote-XGBoost

Zi'an Zeng, Yingmei Li

College of Computer Science and Information Engineering, Harbin Normal University, Harbin Heilongjiang

Received: Apr. 24th, 2024; accepted: Jun. 21st, 2024; published: Jun. 30th, 2024

Abstract

With the increasing complexity of software systems, software defect prediction has become an important means to ensure software quality. This study proposes a software defect prediction model based on PCA-Smote-XGBoost, aiming to improve the accuracy and efficiency of defect prediction. This article uses Principal Component Analysis (PCA) for data dimensionality reduction, retaining 95% of the variance to reduce the number of features and extract key information; uses Smote oversampling method to solve the problem of data imbalance; builds a prediction model using the XGBoost algorithm and validates its effectiveness through experiments. Among the eleven common-

文章引用: 曾子安, 李英梅. 基于 PCA-Smote-XGBoost 的软件缺陷预测研究[J]. 软件工程与应用, 2024, 13(3): 346-357.
DOI: 10.12677/sea.2024.133035

ly used datasets for software defect prediction, experimental results show that the model has the highest accuracy ACC and F1 compared to the other eight benchmark models in software defect prediction, and can effectively assist software development teams in identifying potential defect risks.

Keywords

Software Defect Prediction, PCA, Smote, XGBoost

1. 引言

软件缺陷预测作为软件工程中的关键环节，对于提高软件质量和降低维护成本具有重要意义。随着机器学习技术的发展，基于数据驱动的方法为软件缺陷预测提供了新的视角和强大的工具。传统的软件测试方法依赖于人工经验和规则，这些方法在面对大规模、高复杂性的软件系统时显得力不从心。随着机器学习技术的进步，基于数据驱动的软件缺陷预测方法开始受到重视。这些方法通过分析历史数据，学习缺陷模式，从而在软件开发的早期阶段预测潜在的缺陷。然而，软件数据的高维性和复杂性给特征提取和模型构建带来了挑战，如何从众多特征中识别出与缺陷高度相关的信息，是提高预测模型性能的关键。

针对这一挑战，本文提出了一种基于 PCA-Smote-XGBoost 的软件缺陷预测模型。该模型结合了主成分分析(PCA)的降维能力、合成少数过采样(Smote)技术以及 XGBoost 算法的强大预测性能。PCA 作为一种经典的线性降维技术，能够从原始数据中提取出最重要的特征，从而简化模型并减少计算复杂度。Smote 作为一种过采样方法，能够平衡数据集中的类别分布，尤其适用于处理不平衡数据集，增强模型对少数类别的识别能力。XGBoost 算法则是一种基于梯度提升的决策树算法，它通过迭代优化来提高模型的预测准确性，尤其在处理结构化数据方面表现出色。

本研究的目标是构建一个高效、准确的软件缺陷预测模型，以便在软件开发过程中及时发现潜在的缺陷，从而提高软件的质量和可靠性。我们通过 PCA 进行数据降维，利用 Smote 过采样方法处理软件缺陷数据集中的类别不平衡问题，结合 XGBoost 算法构建预测模型。本文通过十一个项目下的八种基准模型实验对比出了 PCA-Smote-XGBoost 模型的优越性，证明了该模型在软件缺陷预测方面的有效性和实用性。

2. 软件缺陷预测定义及相关研究

软件缺陷是由于开发或维护过程存在的错误产生的，导致计算机无法正常运行的错误或功能性缺陷。传统的软件缺陷检测方法有手动测试、自动化分析、静态分析、代码审查，这些方法都局限于人力资源与时间资源，而基于机器学习和深度学习的缺陷预测具有高效率、低成本的优点。目前面临的挑战主要有以下六个方面：数据集的来源与处理、代码向量的表征方法、预训练模型的提高、深度学习模型的探索、细粒度预测技术、预训练模型的迁移。

软件缺陷预测框架如下图 1 所示。

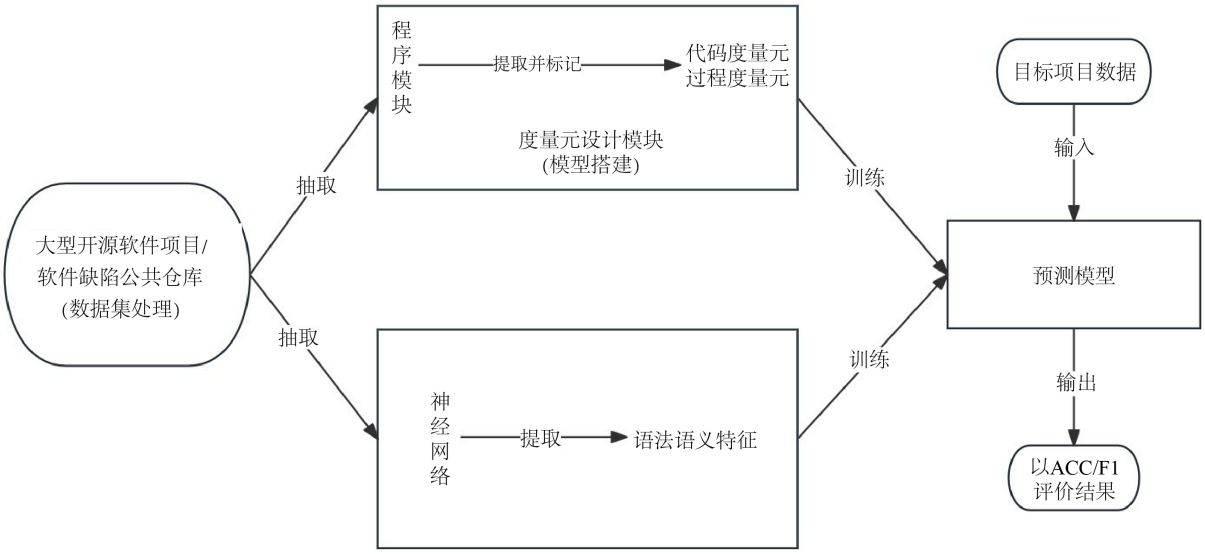

Figure 1. Framework of software defect prediction model
图 1. 软件缺陷预测模型框架

1) 数据集处理

从大型开源软件项目、软件缺陷公共仓库、开源社区收集缺陷数据。

2) 定义与软件缺陷高度相关的度量元，提取程序模型的软件度量特征或者使用神经网络提取源代码的语法语义特征(模型搭建)，利用机器学习或深度学习构建预测模型。

3) 使用性能评价指标对构建的预测模型进行评价。

2.1. 相关研究

2.1.1. 采样方法研究

软件缺陷数据集从包含软件项目开发周期的软件仓库里收集。然而，在挖掘软件历史仓库的过程中，数据集中包含一些降低预测模型性能的不必要的信息，如异常值、高维度、类不平衡、数据差异等。

通常，软件产品大致符合二八原则，80%的缺陷集中在 20%的程序模块中，缺陷数据分布不均匀，有缺陷的模块要远少于无缺陷模块，即类不平衡问题。类不平衡严重影响了分类模型的性能，无缺陷的实例支配着数据样本，分类器会偏向于无缺陷的实例。而在软件测试中，有缺陷的模块被误报为无缺陷的模块会给软件公司带来更大的损失。

纪晨辉等[1]提出了基于领域合成的过采样方法，该方法分为两个阶段：样本预处理和样本合成。对于每个少数类样本，计算其周围的邻居样本，并根据邻居样本中少数类和多数类的数量，将样本分类为安全类、边界类和噪声类。对于每个边界点，找到其 k 个最近邻点，并根据这些邻居样本中多数类和少数类的数量差异，确定需要生成的合成样本数量。计算每个特征的最大值和最小值，并找到中间范围，这个范围即为可以合成的特征值的范围。在这个范围内随机生成特征值，并将生成的合成样本添加到列表中。最后，将合成的样本与原始数据集合并，得到过采样后的数据集。该方法在 AEEEM 数据集和 NASA 数据集上的表现优于其他传统的过采样算法。饶珍丹等[2]在过采样预处理阶段，提出了 AJCC-Ram 的过采样方法，该方法基于改进的 ADASYN 自适应过采样和 CURE-SMOTE 过采样，分别在类边缘和类中心层面生成新样本。具体来说，该方法首先根据少数类样本的 K 近邻样本将其分为边缘类和中心类。对于边缘类样本，采用改进的 ADASYN 算法进行过采样，以增强边界处样本的多样性；对于中心类样本，采

用 CURE-SMOTE 算法进行过采样，以保证样本数据的原始分布属性。此外，为了解决过采样可能引起的噪声问题，还采用了 CLNI (最近列表噪声识别)方法进行噪声过滤和数据清理。Goyal [3]提出了一种新的基于邻域的欠采样 N-US，首先识别“缺陷”(buggy)实例的邻域，并找出属于“无缺陷”(clean)类的邻居。计算这些“无缺陷”邻居与“缺陷”实例的距离，并确定那些距离小于所有邻居距离中位数的邻居为“待删除”(Nominated-for-Elimination)数据点。统计这些“待删除”数据点在不同“缺陷”实例邻域中的出现次数，选择那些在多个邻域中重复出现的“待删除”数据点进行删除。删除这些“待删除”的“无缺陷”数据点后，得到一个更加平衡的欠采样训练数据集。张丽等[4]提出了一种改进的过采样方法，这种方法基于聚类分析，特别是针对少数类的样本进行处理，以提高过采样的质量和代表性。使用基于聚类分析的特征选择算法 FECAR (Feature Clustering and Feature Ranking)，通过最小描述长度原则(MDLP)对特征进行离散化，计算特征之间的关联性，以及特征与类标之间的信息增益，进而进行特征聚类，选择关键特征集。对少数类样本进行 K-means 聚类，然后根据聚类结果，对每个簇中的样本计算合成样本的数量。合成样本的总数等于多数类和少数类样本的差值。在合成新样本时，考虑了样本的关键特征权重和与簇心的距离权重，以此来确定每个样本的合成数量，并使用改进的 SMOTE 算法合成新样本。

2.1.2. 特征重要性研究

Rajbahadur 等[5]探讨了两类特征重要性方法：分类器特定(Classifier Specific, CS)方法和分类器不可知(Classifier Agnostic, CA)方法。这些方法用于从缺陷分类器中派生特征重要性排名。研究发现，即使是对于同一个数据集和分类器，不同的特征重要性方法可能会计算出不同的特征重要性排名。因此，除非这些方法之间存在强烈的一致性，否则这种可互换使用可能会导致结论不稳定。CS 方法利用给定分类器的内部信息来衡量每个特征对分类器预测的贡献程度。而 CA 方法将分类器视为“黑盒”，不使用任何特定于分类器的详细信息来计算特征的重要性。CA 方法的主要优势是它们可以用于任何分类器，无论是可解释的还是黑盒分类器。Gao 等[6]探讨了在软件缺陷预测中处理数据不平衡问题时可解释性的重要性，并提出了一种基于规则的可解释模型来直接处理不平衡数据。通过实证研究，论文发现重采样技术[7] (如过采样、欠采样和 SMOTE)会对基于规则的可解释模型产生显著影响，包括改变特征重要性和增加模型复杂度。此外，直接在原始不平衡数据[8]上构建有意义的可解释模型是不可行的，因为缺陷覆盖率低且性能差。软移除机制[9] (Soft-Removal for Covered Instances)：对于被覆盖的少数类实例，不是直接移除，而是通过权重递减的方式使其对后续规则的影响减小，但仍有可能被其他规则覆盖。改进的增益函数[10] (Improvement on FOIL Gain)：使用加权的少数类实例数量来计算候选项的增益，同时降低覆盖多数类实例的候选项的重要性。容忍较低[11]的覆盖率(Tolerance of Lower Coverage)：接受信心度高于最小覆盖率阈值的规则，以确保模型能够尽可能覆盖缺陷实例。

3. 实验数据及数据不平衡

3.1. 实验数据

本次实验使用的十一个项目为 AEEEM [12]下的 EQ，NASA 下的 mw1，SOFTLAB 下的 ar1 与 ar4，MORPH 下的 ant-1.4、jedit-4.0、jedit-4.3、log4j-1.0、camel-1.0、poi-2.5 与 velocity-1.4。

以 AEEEM 数据集下的 EQ 项目[13]为例，表 1 展示了 61 个特征中前 10 个特征的数据示例。对于项目中[14]的每个模块，每个特征都有一个特定的值。

项目的平均值[15]是通过将每个模块的赋值之和除以项目中的模块数量计算得出。该表展示了项目中缺陷模块[16]的百分比，数据表现出明显的不平衡性。

Table 1. Overview of AEEEM data program metrics
表 1. AEEEM 数据计划指标概览

	projects	EQ
	模型数量	1862
	缺陷模型	13.15%
1	ck_oo_number of private methods numeric	1.01
2	LDHH_lcom numeric	0.002
3	LDHH_fanIn numeric	0.007
4	number of non-trivial bugs found until: numeric	3.65
5	WCHU_number of public attributes numeric	0.037
6	WCHU_number of attributes numeric	0.469
7	CvsW Entropy numeric	0.027
8	LDHH_number of public methods numeric	0.001
9	WCHU_fanIn numeric	1.007
10	LDHH_number of private attributes numeric	0.002

3.2. 数据不平衡

这项工作面临数据高度不平衡[17]的问题，其中约 13%的软件模块存在缺陷，而近 87%的模块是干净的(表 1)。因此，如果不对数据进行深入挖掘[18]，我们无法简单使用软件缺陷数据集来评估我们的模型。为了解决这一问题，我们采用了一种称为“合成少数过采样技术”[19] (SMOTE, Synthetic Minority Oversampling Technique)的方法。如下图 2 所示。

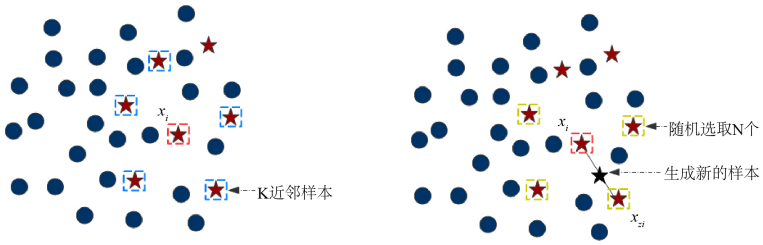

Figure 2. Schematic diagram of Smote oversampling
图 2. Smote 过采样原理图

该算法由 Chawla 提出，其原理是通过少数类样本与其 N 个近邻样本之间的线性插值[20]合成新的少数类样本。算法步骤如下：

对每个少数类样本，计算其与其他少数类样本的欧式距离，得到 N 个近邻。

从 N 个近邻中随机选择一个少数类样本，合成新的少数类样本。合成公式如式(1)所示：

$$x_{new} = x_i + rand(0,1) \times (x_{zi} - x_i) \tag{1}$$

其中，x_i 为第 i 个少数类样本，x_{zi} 为 x_i 的随机近邻样本。如上图 2 所示，左侧为计算 x_i 的 K 近邻样本，右侧为合成新样本。

4. XGBoost 的特征重要性分析及 PCA

4.1. 热力图

SHAP (SHapley Additive exPlanations)是一种基于博弈论中 Shapley 值的方法，用于解释机器学习模型的预测。它通过分配每个特征对预测结果的独特贡献，提供了模型决策的透明度。在软件缺陷预测中，SHAP 能帮助识别导致缺陷的关键因素，辅助开发人员进行有效的缺陷管理和质量控制。SHAP 的通用性使其适用于多种模型，增强了模型预测的可解释性。见图 3，以 AEEEM 数据集下的 EQ 项目为例，使用热力图进行分析。

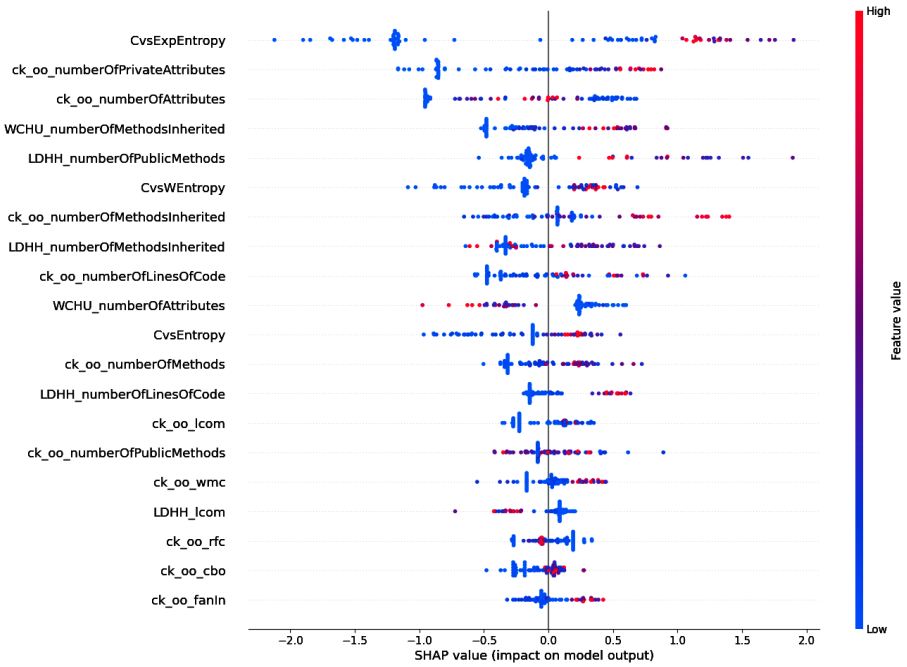

Figure 3. Characteristic heat map of EQ projects under the XGBoost model
图 3. XGBoost 模型下 EQ 项目的特征热力图

如上面的 summary_plot 图 3 所示，*y* 轴为项目特征名，*x* 轴中正数表示促进，负数表示抑制，颜色越红，表示值越大，作用就越强；颜色越蓝色，表示值越小，作用就越弱。可以总结为 *y* 轴表示每一个特征，*x* 轴表示对结果是促进还是抑制，不同的颜色表示作用效果。

4.2. XGBoost 可视化特征重要性

在机器学习中，特征选择对提升模型性能和解释性至关重要。本实验利用 XGBoost 算法对 AEEEM 公开数据集的 EQ 项目进行特征重要性分析和选择。XGBoost 是一种高效的梯度提升决策树算法，它通过迭代优化损失函数，适用于大规模和高维数据，且能输出特征重要性评分。如下图 4 所示。

我们采用基于特征重要性的过滤方法，通过 XGBoost 的“plot_importance”函数可视化特征重要性，并设置阈值筛选出关键特征。实验结果显示，通过选取重要性高的特征子集，模型在保持准确率的同时减少了特征数量，增强了模型的解释性并提升了性能。

选择的特征子集对模型预测的贡献显著，且模型准确率未因特征减少而下降，反而有所提升，这一发现证实了 XGBoost 在特征选择上的有效性。

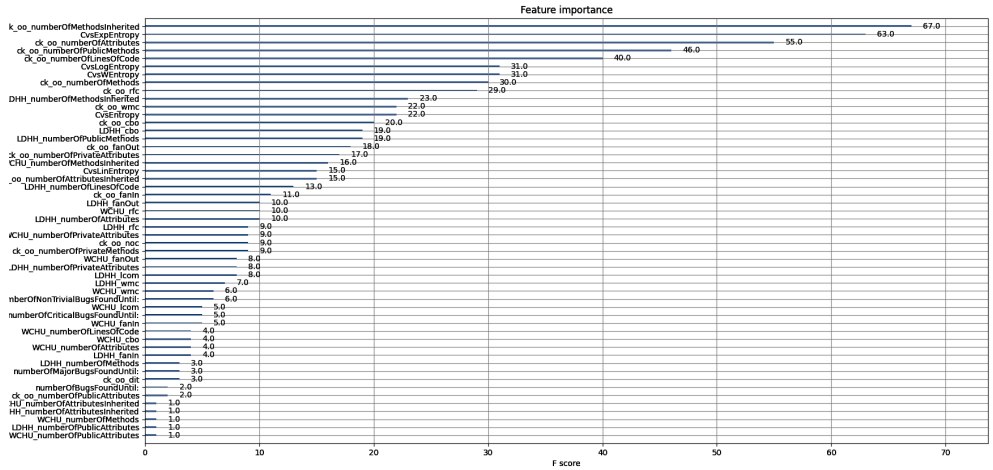

Figure 4. Feature importance score of EQ projects under the XGBoost model
图 4. XGBoost 模型下 EQ 项目的特征重要性得分

4.3. PCA 主成分分析

主成分分析(PCA)是多元统计分析中一种重要的降维技术，旨在通过减少数据集的维度来简化数据，同时保留最大的方差信息。该方法通过线性变换将原始数据投影到新的坐标系中，新坐标的基向量(主成分)捕捉数据中的主要变异。

PCA 的基本步骤包括：首先，构建原始数据矩阵 X；其次，对数据进行零均值化处理；然后，计算协方差矩阵以评估变量间的相关性；接着，求解协方差矩阵的特征值和特征向量；之后，根据特征值的大小对特征向量进行排序，并选择前 k 个特征向量组成投影矩阵 P；最后，通过 P 将原始数据映射到 k 维空间，得到降维后的数据。

方差描述一个数据的离散程度：

$$\operatorname{cov}(x)=\frac{1}{n}\sum_{i=1}^{n}\left(X_i-\overline{X}\right)\left(Y_i-\overline{Y}\right) \tag{2}$$

协方差描述两个数据的相关性，接近 1 就是正相关，接近−1 就是负相关，接近 0 就是不相关。

$$C\left(x_a,x_b\right)=\frac{1}{n}\sum_{i=1}^{n}\left(x_{ai}-\overline{x}_a\right)\left(x_{bi}-\overline{x}_b\right) \tag{3}$$

下面，我们以向量个数等于 3 为例，我们转变为矩阵的表示形式：

$$C\begin{pmatrix}\operatorname{cov}(x,x) & \operatorname{cov}(x,y) & \operatorname{cov}(x,z)\\ \operatorname{cov}(y,x) & \operatorname{cov}(y,y) & \operatorname{cov}(y,z)\\ \operatorname{cov}(z,x) & \operatorname{cov}(z,y) & \operatorname{cov}(z,z)\end{pmatrix} \tag{4}$$

从矩阵中会发现协方差矩阵其实就是由对角线的协方差与非对角线的协方差构成，所以协方差矩阵的核心理解：由向量自身的方差与向量之间的协方差构成。这个定义延伸到更多向量时同样受用。

在线性代数中给定一个方阵 A，它的特征向量 v 在 A 变换的作用下，得到的新向量仍然与原来的 v 保持在同一条直线上，且仅仅在尺度上变为原来的 λ 倍。则称 v 是 A 的一个特征向量，λ 是对应的特征值，即：

$$Av=\lambda v \tag{5}$$

这里的特征向量 v 也就是矩阵 A 的主成分，而 λ 则是其对应的权值也就是各主成分的方差大小。

至此，我们的目的就是寻找主成分，所以问题就转变成了寻找包含主成分最多的特征向量 v，但是这个 v 并不好直观的度量，由此我们可以转变为寻找与之对应的权值 λ，λ 越大，则所对应的特征向量包含的主成分也就越多，所以问题接着转变为了寻找最大值的特征值 λ。

通过计算向量矩阵得到其协方差矩阵，然后特征分解协方差矩阵得到包含主成分的特征向量与对应的特征值，最后通过特征值的大小排序后依次筛选出需要用于降维的特征向量个数。

5. PCA-Smote-XGBoost 模型实验步骤

5.1. 基准模型-Smote 实验步骤

1) 准备数据，对目标变量进行编码，标出有缺陷值。
2) 划分训练集与测试集，测试集不动，对训练集进行 Smote 过采样合成样本。
3) 将处理后的数据集导入到基准模型-Smote 模型中进行训练。
4) 保存基准模型-Smote 模型，对测试集进行预测，并计算 ACC (精确度)、F1。

5.2. PCA-Smote-XGBoost 实验流程

1) 归一化和标准化处理数据集。
2) 划分训练集与测试集，测试集不动，对训练集进行 Smote 过采样合成样本。
3) PCA 降维处理后，以 AEEEM 下的 EQ 项目为例，特征从 61 个降到 23 个，保留 95%的方差。
4) 将处理后的数据集导入到 PCA-Smote-XGBoost 模型中进行训练。
5) 保存 PCA-Smote-XGBoost 模型，对测试集进行预测，并计算 ACC (精确度)、F1。

6. 实验及结果分析

6.1. 评价指标

我们还使用精确度 Accuracy (ACC)和 F1 指标来评估所考虑模型的有效性。F1 指标是精确度和召回率之间的调和平均值。

TP (True Positive)：正确的视为异常样本。

FP (False Positive)：错误的视为异常样本。

FN (False Negative)：错误的视为正常样本。

TN (True Negative)：正确的作为正常样本。

注：P：异常样本；N：正常样本；T：正确的；F：错误的。

Accuracy：正确分类的样本数占总样本数的比例。

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}} \tag{6}$$

Precision：真实有缺陷占预测为有缺陷样本的比例。

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7}$$

Recall：正确视为有缺陷的样本占所有异常样本的比例。

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8}$$

F1：综合了 Precision 和 Recall 两指标。

$$\text{F-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

6.2. XGBoost-Smote 模型结果及分析

如表 2、表 3 所示，标粗的为最优值，以 ACC 精确度和 F1 为指标，在 NASA 数据集下的 MW1、KC1、PC1、PC5 项目；MORPH 数据集下的 camel-1.2、ivy-1.4、poi-1.5、redktor 项目进行对比实验，使用了 Smote (合成少数类过采样技术)、ADASYN (自适应合成抽样)、Smote_Tomek (去除那些与多数类样本最近的少数类样本，进一步提升分类效果)、Smote_ENN (移除一些噪声和边界不清晰的样本的欠采样技术)、ACTUAL (不使用采样方法)。XGBoost-Smote 模型明显优于其他四种情况。

Table 2. Comparison of four oversampling methods under XGBoost (ACC value of NASA dataset)
表 2. XGBoost 下四种过采样方法对比(NASA 数据集 ACC 值)

项目	Smote	ADASYN	Smote_Tomek	Smote_ENN	ACTUAL
MW1	**0.862**	0.843	0.843	0.784	0.823
KC1	**0.834**	0.812	0.793	0.725	**0.834**
PC1	**0.936**	0.914	0.914	0.783	0.918
PC5	**0.732**	0.718	0.702	0.663	0.718

Table 3. Comparison of four oversampling methods under XGBoost (F1 value of MORPH dataset)
表 3. XGBoost 下四种过采样方法对比(MORPH 数据集 F1 值)

项目	Smote	ADASYN	Smote_Tomek	Smote_ENN	ACTUAL
camel-1.2	**0.676**	0.672	0.669	0.553	0.645
ivy-1.4	**0.759**	**0.759**	**0.759**	0.670	**0.759**
poi-1.5	**0.730**	0.711	0.689	0.685	**0.730**
redktor	**0.788**	0.765	**0.788**	0.657	0.727

6.3. PCA-Smote-XGBoost 模型结果及分析

以 AEEEM 数据集下的 EQ 项目为例，为了提高模型的训练速度，使用 PCA 算法对特征进行降维，如下图 5 所示为 PCA 降维处理后，把 EQ 的特征降到 23 个，并保留 95%以上的信息。

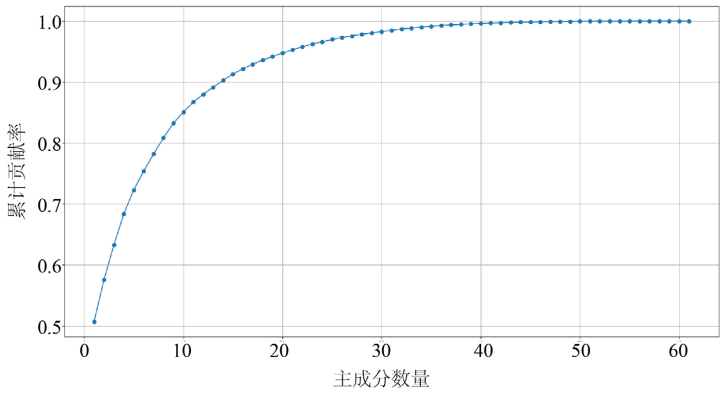

Figure 5. Relationship between the number of principal components and the cumulative contribution rate
图 5. 主成分数量和累计贡献率的关系

本研究同时以 ACC 和 F1 作为评价指标，在软件缺陷数据集的十一个项目中，与八种基准模型(AdaBoost 自适应增强这里简称为 AB、CatBoost Categorical Boosting 这里简称为 CB、Decision Tree 决策树这里简称为 DT、Gradient Boosting 梯度提升这里简称为 GB、K-Nearest Neighbor 近邻这里简称为 KNN、RandomForest 随机森林这里简称为 RF、SVM 支持向量机、XGBoost、Ridge 岭回归)进行对比实验，如下表 4 所示。

Table 4. Comparison of eight benchmark models under Smote (ACC/F1 value)
表 4. Smote 下八种基准模型对比(ACC/F1 值)

模型/项目	velocity-1.4	ant-1.4	jedit-4.0	jedit-4.3	log4j-1.0	poi-2.5	EQ	MW1	AR1	AR4	camel-1.0	平均值
GB	0.82/0.82	0.58/0.57	0.67/0.68	0.91/0.92	0.62/0.66	0.77/0.78	0.69/0.69	0.80/0.79	0.88/0.86	0.81/0.80	0.86/0.88	0.76/0.76
RF	0.85/0.84	0.66/0.65	0.70/0.70	0.92/**0.93**	0.66/0.69	0.71/0.72	0.67/0.67	0.78/0.76	0.84/0.84	0.77/0.73	**0.92**/**0.91**	0.77/0.76
CB	**0.87**/**0.87**	0.69/0.67	0.70/0.70	0.91/0.92	0.70/0.72	0.75/0.75	0.67/0.67	0.78/0.75	0.84/0.84	0.81/0.80	**0.92**/**0.91**	0.78/0.78
DT	0.85/0.83	0.63/0.61	0.72/0.72	**0.93**/**0.93**	**0.77**/**0.79**	0.81/0.81	0.69/0.69	0.74/0.74	0.92/0.92	0.77/0.73	0.85/0.88	0.78/0.78
AB	0.85/0.84	0.66/0.65	0.67/0.67	0.92/**0.93**	0.66/0.69	0.71/0.72	0.63/0.63	0.76/0.74	0.80/0.81	**0.86**/0.83	**0.92**/**0.91**	0.76/0.76
RD	0.70/0.70	0.69/0.71	0.59/0.62	0.72/0.81	**0.77**/**0.79**	0.72/0.73	0.66/0.65	**0.82**/0.78	0.84/0.84	**0.86**/**0.85**	0.79/0.84	0.74/0.75
KNN	0.62/0.63	0.58/0.59	0.66/0.67	0.80/0.81	0.59/0.62	0.64/0.66	0.66/0.65	0.60/0.63	0.76/0.80	0.68/0.69	0.79/0.85	0.67/0.69
SVM	0.67/0.58	0.66/0.66	0.74/0.72	0.84/0.89	0.70/0.72	0.50/0.52	0.64/0.64	0.78/0.77	0.84/0.84	0.72/0.70	0.72/0.80	0.71/0.71
PCA-XGB	**0.87**/**0.87**	**0.72**/**0.72**	**0.75**/**0.74**	**0.93**/**0.93**	**0.77**/**0.79**	**0.83**/**0.83**	**0.70**/**0.70**	**0.82**/**0.81**	**0.96**/**0.95**	**0.86**/**0.85**	**0.92**/**0.91**	**0.83**/**0.82**

为了更加直观地分析实验结果，本文由表 4 得到了如图 6、图 7 所示的折线图，从表 4、图 6、图 7 可以看出：PCA-Smote-XGBoost 模型在十一个软件缺陷数据集的项目中，相较于其他八种基准模型，在 ACC 和 F1 两个指标上，同时取得了全部第一，实验结果保留两位小数，在某些项目中出现并列第一的情况，同时 PCA-Smote-XGBoost 模型在十一个项目上的 ACC/F1 平均值为 0.83/0.82，比其他八种基准模型的平均值分别提高了 7%/6%、6%/6%、5%/4%、5%/4%、7%/6%、9%/7%、16%/13%、12%/11%，无论从局部还是整体来看，PCA-Smote-XGBoost 模型都优于其他八种基准模型。

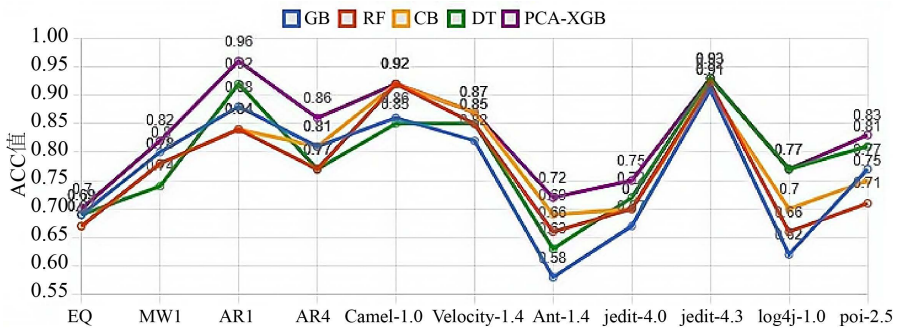

Figure 6. ACC values on eleven projects for the first four benchmark models
图 6. 前四种基准模型在十一个项目上的 ACC 值

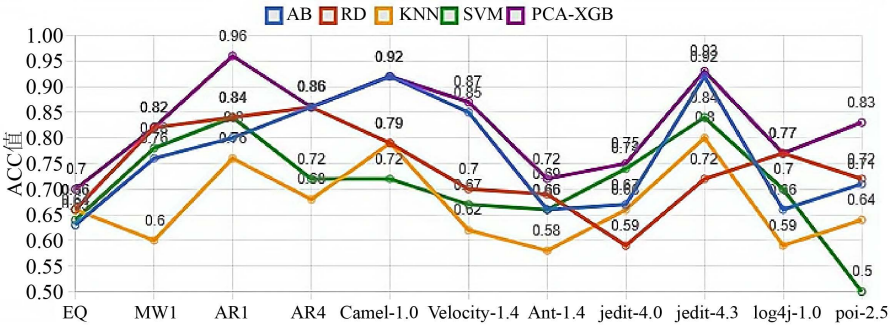

Figure 7. ACC values on eleven projects for the last four benchmark models
图 7. 后四种基准模型在十一个项目上的 ACC 值

6.4. 结束语

针对软件缺陷预测中成本高效率低和数据集的类不平衡问题，本文从软件缺陷数据集的特征入手，首先对 XGBoost 下的特征重要性进行了分析，比较了四种采样方法，得出了 XGBoost-Smote 的优越性，然后在基准模型-Smote 的基础上进行对比实验，给 XGBoost-Smote 加入 PCA，降低维度提高性能指标，提出了 PCA + Smote + XGBoost 模型，在软件缺陷数据集的十一个项目上验证了该模型的有效性。

参考文献

[1] 纪晨辉, 李英梅. 一种邻域合成的软件缺陷预测过采样方法[J]. 软件工程与应用, 2023, 12(6): 930-939. https://doi.org/10.12677/sea.2023.126091

[2] 饶珍丹, 李英梅, 董昊, 等. 多层次过采样集成的不平衡数据缺陷预测模型[J]. 小型微型计算机系统, 2023, 44(4): 888-896. https://doi.org/10.20009/j.cnki.21-1106/TP.2021-0634

[3] Goyal, S. (2021) Handling Class-Imbalance with KNN (Neighbourhood) Under-Sampling for Software Defect Prediction. *Artificial Intelligence Review*, **55**, 2023-2064. https://doi.org/10.1007/s10462-021-10044-w

[4] 张丽, 沈雅婷, 朱园园. 基于改进 SMOTE 的软件缺陷预测[J]. 计算机工程与设计, 2023, 44(10): 2965-2972. https://doi.org/10.16208/j.issn1000-7024.2023.10.012

[5] Rajbahadur, G.K., Wang, S.W., Oliva, G.A., Kamei, Y. and Hassan, A.E. (2022) The Impact of Feature Importance Methods on the Interpretation of Defect Classifiers. *IEEE Transactions on Software Engineering*, **48**, 2245-2261.

[6] Gao, Y.X., Zhu, Y. and Zhao, Y. (2022) Dealing with Imbalanced Data for Interpretable Defect Prediction. *Information and Software Technology*, **151**, Article ID: 107016.

[7] 宫丽娜, 姜淑娟, 姜丽. 软件缺陷预测技术研究进展[J]. 软件学报, 2019, 30(10): 3090-3114. https://doi.org/10.13328/j.cnki.jos.005790

[8] 李莉, 任振康, 石可欣. 代价敏感的 Boosting 软件缺陷预测方法[J]. 计算机工程, 2022, 48(3): 175-180. https://doi.org/10.19678/j.issn.1000-3428.0061316

[9] Shepperd, M., Song, Q., Sun, Z. and Mair, C. (2013) Data Quality: Some Comments on the NASA Software Defect Datasets. *IEEE Transactions on Software Engineering*, **39**, 1208-1215. https://doi.org/10.1109/tse.2013.11

[10] 刘旭同, 郭肇强, 刘释然, 等. 软件缺陷预测模型间的比较实验: 问题、进展与挑战[J]. 软件学报, 2023, 34(2): 582-624. https://doi.org/10.13328/j.cnki.jos.006714

[11] Wang, S., Liu, T. and Tan, L. (2016) Automatically Learning Semantic Features for Defect Prediction. *Proceedings of the* 38*th International Conference on Software Engineering*, Austin, 14-22 May 2016, 297-308. https://doi.org/10.1145/2884781.2884804

[12] Xuan, X., Lo, D., Xia, X. and Tian, Y. (2015) Evaluating Defect Prediction Approaches Using a Massive Set of Metrics: An Empirical Study. *Proceedings of the* 30*th Annual ACM Symposium on Applied Computing*, Salamanca, 13-17 April 2015, 1644-1647. https://doi.org/10.1145/2695664.2695959

[13] Menzies, T., Greenwald, J. and Frank, A. (2007) Data Mining Static Code Attributes to Learn Defect Predictors. *IEEE Transactions on Software Engineering*, **33**, 2-13. https://doi.org/10.1109/tse.2007.256941

[14] Jing, X., Ying, S., Zhang, Z., Wu, S. and Liu, J. (2014) Dictionary Learning Based Software Defect Prediction. *Proceed-*

ings of the 36*th International Conference on Software Engineering*, Hyderabad, 31 May-7 June 2014, 414-423. https://doi.org/10.1145/2568225.2568320

[15] Fukushima, T., Kamei, Y., McIntosh, S., Yamashita, K. and Ubayashi, N. (2014) An Empirical Study of Just-in-Time Defect Prediction Using Cross-Project Models. *Proceedings of the* 11*th Working Conference on Mining Software Repositories*, Hyderabad, 31 May-1 June 2014, 172-181. https://doi.org/10.1145/2597073.2597075

[16] Turhan, B., Menzies, T., Bener, A.B. and Di Stefano, J. (2009) On the Relative Value of Cross-Company and Within-Company Data for Defect Prediction. *Empirical Software Engineering*, **14**, 540-578. https://doi.org/10.1007/s10664-008-9103-7

[17] Tantithamthavorn, C. and Hassan, A.E. (2018) An Experience Report on Defect Modelling in Practice. *Proceedings of the* 40*th International Conference on Software Engineering*: *Software Engineering in Practice*, Gothenburg, 27 May-3 June 2018, 286-295. https://doi.org/10.1145/3183519.3183547

[18] Tantithamthavorn, C., McIntosh, S., Hassan, A.E., Ihara, A. and Matsumoto, K. (2015) The Impact of Mislabelling on the Performance and Interpretation of Defect Prediction Models. 2015 *IEEE/ACM* 37*th IEEE International Conference on Software Engineering*, Florence, 16-24 May 2015, 812-823. https://doi.org/10.1109/icse.2015.93

[19] Koziarski, M., Krawczyk, B. and Woźniak, M. (2019) Radial-Based Oversampling for Noisy Imbalanced Data Classification. *Neurocomputing*, **343**, 19-33. https://doi.org/10.1016/j.neucom.2018.04.089

[20] Zimmermann, T. and Nagappan, N. (2008) Predicting Defects Using Network Analysis on Dependency Graphs. *Proceedings of the* 13*th International Conference on Software Engineering*, Leipzig, 10-18 May 2008, 531-540. https://doi.org/10.1145/1368088.1368161