基于调优树的软件缺陷预测堆叠集成模型

丁浩杰 [1]
DING Haojie

摘 要 软件缺陷预测是一种利用机器学习技术自动识别软件中潜在缺陷的方法。与单一分类器相比，集成学习方法在软件缺陷预测的预测性能上表现出显著的优势。然而，以往的研究在软件缺陷预测中普遍采用集成模型，并使用其默认的超参数设置，这通常被视为次优选择。对此，提出了一种基于调优树的软件缺陷预测堆叠集成模型。对基于树的四种集成模型（随机森林、极度随机树、自适应提升和极致梯度提升）的超参数进行了网格搜索优化，随后将这些调优后的树类模型作为基分类器，通过堆叠的方式构建集成模型。实验结果表明，基于调优树的堆叠集成方法可以在 F_1 值和 AUC 值两个度量上提升模型的预测性能。

关键词 堆叠集成；随机森林；软件缺陷预测；超参数优化

doi：10.3969/j.issn.1672-9528.2024.07.003

0 引言

软件开发过程中，确保软件的高品质是至关重要的。然而，在实际操作中，受到时间和财务预算的限制，往往难以实现这一目标。有效识别并修正软件中的缺陷，尽管对于提升软件质量至关重要，但这一过程既复杂又成本高昂。为了解决这一问题，软件缺陷预测技术应运而生。该技术能够对软件系统中的缺陷模块进行定位和分类，使质量保障措施得以有的放矢，而不是对整个系统进行全面的审查。通过运用机器学习算法，对那些可能包含缺陷的软件组件进行辨识和归类，这一过程被称为软件缺陷预测。采用软件缺陷预测技术，可以在降低测试成本的同时，确保软件产品及时发布，进而全面提升软件产品的品质。

在软件缺陷检测领域，研究人员已经采纳了多种机器学习算法，如决策树和集成学习方法，以提升检测的准确性 [1]。这些算法普遍利用源代码的多种度量指标来构建预测模型，其中 McCabe、Halstead 以及静态代码度量是应用最为广泛的三种。研究显示，集成学习方法，特别是堆叠和提升技术，在软件缺陷预测方面相较于单一分类器有着更为卓越的性能 [2]。然而，过往的研究往往在构建集成模型时采用了默认的超参数设置，这通常并不是最优的选择。超参数的优化对于分类器的性能有着决定性的影响，已有研究通过超参数优化技术，实现了预测性能的显著提升。Osman 等人 [3] 基于 K 最近邻和支持向量机算法，在五个开源的 Java 系统上研究了优化机器学习模型的超参数是否能提高其预测能力，其结果表明模型对其超参数的敏感性各不相同，调整超参数可以提升 K 最近邻和支持向量机的预测性能，并且进行超参数优化阶段，大多数模型的超参数的默认值都会发生变化。Khan 等人 [4] 提出了一种融合机器学习分类器和人工免疫网络（artificial immune network，AIN）的软件缺陷预测模型，并采用超参数优化策略以提高缺陷预测的准确性。研究结果显示，通过超参数优化后的机器学习分类器，结合 AIN 及其相关应用，在软件缺陷预测方面的表现优于采用默认超参数的分类器。

在本研究中，深入分析了堆叠集成技术以及超参数优化对基于树的集成模型性能的影响。首先对随机森林、极度随机树、自适应提升和极致梯度提升等四种基于树的集成模型的超参数进行了网格搜索优化，随后将这些调优后的树类模型作为基分类器进行堆叠集成。实验结果表明，在 F_1 值和 AUC 值两个性能度量上，与调优过后的树类模型相比，基于调优树的软件缺陷预测堆叠集成模型的预测性能更好，证明了其在软件缺陷预测领域的有效性。

1 基础知识

本文提出的基于调优树的软件缺陷预测堆叠集成模型，基分类器为多种基于决策树的算法，比如随机森林、极度随

1. 山西科技学院 山西晋城 048000

[基金项目] 山西省教育厅高等学校科技创新项目（2022L621）；山西科技学院校内科研基金项目（XKY002）；山西科技学院校内科研基金项目（XKY024）

机树、自适应提升和极致梯度提升，在对其进行超参数优化之后，以堆叠泛化（stacking generalization）的方式对调优后的基分类器进行集成，本节将概述相关基础知识。

1.1 随机森林（random forest，RF）

随机森林是一种基于装袋（bagging）的机器学习算法，它由众多小的决策树组成。每棵树都是在原始数据集的一个随机子集上训练而成，以增强模型的泛化能力。在树的每个节点上，算法会随机选择一个特征子集来寻找最优的分割点，这样的随机性减少了过拟合的风险。在分类问题中，随机森林通过多数投票机制来确定最终的分类结果。

1.2 极度随机树（extra trees，ET）

极度随机树，与随机森林算法相似，但在随机性方面更进一步。ET 与 RF 的区别主要体现在两个方面：（1）每棵决策树都是使用整个数据集构建的，而不是 RF 中的随机子集；（2）ET 在树的每个节点上随机选择分割，而不是像 RF 那样选择最佳分割。这种额外的随机性使得 ET 在训练过程中更加快速，并且在某些情况下能够提高模型的泛化能力。

1.3 自适应提升（AdaBoost，Ada）

AdaBoost 依次训练一系列基础分类器，并通过调整每个实例的权重来聚焦于难以分类的样本，对于那些被错误分类的实例，其权重会得到增加。随后，算法会用一个新的基础学习器来拟合调整权重后的数据集。最终，AdaBoost 通过加权多数投票的方式整合所有基础分类器的输出，其中每个分类器的投票权重取决于其预测准确性（即更准确的分类器被赋予更大的权重）。

1.4 极致梯度提升（eXtreme Gradient Boosting，XGB）

极致梯度递升 [5] 进一步优化了传统梯度提升的方法。XGB 不直接使用梯度，而是利用损失函数的二阶导数来训练新的基础分类器。这种改进使得 XGB 在模型训练过程中更加精确、计算效率更高，因此在实践中被广泛认为比传统的梯度提升算法更加优秀和有效。

1.5 堆叠集成（Stacking）

Stacking[6] 是一种异构的集成学习技术，它通过一个被称为元分类器的高层模型来组合多个基本分类器的输出，以此构建出一个最终的预测模型。这些基本分类器可以采用不同的学习算法，并且通常使用整个训练数据集进行训练。在构建最终的预测模型时，基本分类器的输出将被用作元分类器的输入特征。这个过程有助于综合各个基本分类器的优势，从而提高整体的预测性能。为了防止过拟合，堆叠集成通常会采用分层交叉验证的方法。具体来说，训练数据集会被分成两个部分：一部分用于训练基本分类器，另一部分用于训练元分类器。这种方法可以确保元分类器能够在未见过的数据上学习到更加泛化的模式，从而提高模型的泛化能力。

堆叠集成算法的伪代码如下。

算法 1：Stacking 集成

输入：训练数据集 D，基本分类器集合 C，元分类器 M，折叠数 K。

输出：最终的堆叠集成模型。

（1）将数据集 D 划分为 K 个折叠 $D_1,D_2,\cdots,D_K$。

（2）对于每个基本分类器 $c \in C$:

A. 对于 i=1 到 K:

i. 使用 D_i 进行训练，其余 K-1 个折叠进行测试。

ii. 记录测试集的预测结果作为新的特征。

B. 使用所有折叠的训练数据集 D 训练基本分类器 c。

（3）构建新的训练集 D'，其中每个实例的特征由步骤 2 中记录的预测结果组成。

（4）使用 D' 训练元分类器 M。

（5）返回由基本分类器集合 C 和元分类器 M 组成的堆叠集成模型。

2 实验设置

2.1 数据集

本文实验所采用的数据集来源于在软件缺陷预测领域被广泛使用的 nasa 数据集。Tantithamthavorn 等人 [7] 的研究证明了每个变量的数据量（the number of events per variable，EPV）对缺陷预测模型存在着影响，EPV 值越高，预测模型越稳定。此外，Peduzzi 等人 [8] 指出，大约 10 个 EPV 对于机器学习模型的准确估计是必不可少的。因此，在 11 个被广泛使用的 nasa 数据集中，本文选取了其中 EPV 大于 10 的数据集。数据集具体信息如表 1 所示。

表 1 本文实验选用的数据集相关信息

数据集	变量数	缺陷样本数	非缺陷样本数	总样本数	EPV	缺陷率
KC1	21	314	869	1183	79.6	26.54%
PC5	38	471	1240	1711	15.0	27.53%
JM1	21	1672	6110	7782	12.4	21.49%

在实验前对数据集进行了预处理，即对数据转换，对数据集中的每个自变量的值都通过公式 x=log(1+x) 进行转换，其中 x 是变量的原始数值。

Kwabenad 等人 [9] 的研究表明，随机下采样抽样（RUS）在多个性能指标上被证明是更稳定的重采样方法，可以显著地提升类不平衡数据集的预测性能，因此在进行实验前也对数据集进行了随机下采样抽样处理。

2.2 超参数优化

超参数优化是指寻找和确定机器学习模型最佳超参数值的过程。超参数优化技术包括网格搜索、随机搜索、遗传算法和差分进化等。这些技术可以帮助人们在给定的数据集上找到性能最好的超参数组合，以提高模型的预测能力和泛化能力。

网格搜索[10]系统地遍历搜索空间中的所有可能的超参数值组合。这个搜索空间由一系列待探索的超参数及其各自的可能值构成。对于每一种超参数组合，网格搜索算法会构建并评估一个模型，然后返回表现最佳模型的超参数值作为最优解。

在本研究中，采用了网格搜索方法来寻找集成模型的最优超参数。首先，确定了如表 2 所示的每个基分类器的搜索空间（即超参数及其可能的值）。接着，使用调优数据集（原始数据集被划分为两个部分，其中 10% 用于超参数调优，剩余的 90% 用于训练和测试），使用网格搜索算法，找到每个模型超参数的最优值。对于每个集成模型，通过实施 5 次重复的分层 2 倍交叉验证来构建和评估多个模型，以此来探索所有可能的超参数组合。随后，根据 AUC（曲线下面积）分数来确定表现最佳模型的最优超参数值。最后，使用这些最优超参数值在训练 / 测试数据集上构建集成模型。

表 2 本文实验超参数调优的相关信息

算法	超参数名称	描述	优化值范围
随机森林（random forest (RF)） 极度随机树（Extra Trees (ET)）	n_estimators	森林中决策树的数量	[100, 50, 40, 30]
	max_depth	决策树最大深度	[1, 4, 8]
	min_samples_leaf	叶子结点含有的最少样本	RF=[1, 10, 5]
			ET=[1, 2, 4]
	criterion	节点的划分标准	[gini, entropy]
Adaboost 提升树（Ada）	n_estimators	弱学习器的最大迭代次数	[50, 100, 1000]
	learning_rate	每个弱学习器的权重缩减系数	[1, 0.1, 0.001, 0.01]
XGBoost（XGB）	n_estimators	梯度提升树的数量	[100, 50, 500, 1000]
	max_depth	弱学习器的最大树深度	[0.3, 0.1, 0.001, 0.01]
	learning_rate	每个弱学习器的权重缩减系数	[6, 3, 4, 5]

2.3 模型构建

在本实验中，数据集被划分为两组：第一组为 10% 的数据用于超参数调优，即调优数据集；第二组为 90% 的数据用于训练和测试集成模型，即训练 / 测试数据集。构建了四个基于树的模型，包括随机森林（RF）、极度随机树（ET）、自适应提升（Ada）、极致梯度提升（XGB），以及使用经过调整的基于树的模型作为基础模型，并选择逻辑回归作为元模型，构建了堆叠集成。在本研究中，用到的相关算法使用 Python 和 scikit-learn 实现。

在本研究中，采用了分层的 10 倍交叉验证方法，并重复进行了 10 次，以验证集成模型的性能。具体来说，每个数据集被分割成 k 个相等大小的折叠，其中每个折叠被用作一次模型评估的测试集，而剩余的 k-1 个折叠则用于训练模型。这种技术被重复执行了 10 次（即 k=10），确保每个折叠恰好被用作一次测试数据集。为了获得最终的性能估计，对这 10 次迭代的检测性能进行了平均。在分层交叉验证中，每个折叠中的实际比例大致等于原始数据集中的比例，确保了模型训练和测试的公平性。整个过程重复了 10 次，对每个被评估的模型进行了 100 次独立运行。最终，报告的模型性能是基于这 100 次独立运行的平均值。

2.4 度量

2.4.1 F_1 值

F_1 值是精确率和召回率的调和平均数。F_1 值计算公式为：

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (1)$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2)$$

$$\text{recall（TPR）} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3)$$

式中：真阳性（TP）是指被正确分类为缺陷的缺陷实例的数量，假阳性（FP）是指被错误分类为缺陷的非缺陷实例的数量，假阴性（FN）是指被错误分类为非缺陷的缺陷实例的数量。

2.4.2 AUC

曲线下面积（area under the curve，AUC）是指在接收者操作特征（receiver operating characteristic，ROC）曲线下面积的百分比。它通过比较真阳性率（true positive rate，TPR）和假阳性率（false positive rate，FPR）来评估模型的性能。AUC 度量计算的是 ROC 曲线下的面积，这个面积越大，表示模型的性能越好。AUC 的值介于 0 ～ 1 之间，其中 1 表示最佳性能，而 0 表示最差性能。

3 实验结果

3.1 超参数调优结果

表 3 为 4 个基于树的集成模型在 3 个不同的数据集上通过网格搜索算法得到的最优参数列表，后续的堆叠集成模型将使用下列参数构建基分类器。

表 3 各基准模型经过调优后的参数列表

数据集	RF 最优参数组合	ET 最优参数组合	Ada 最优参数组合	XGB 最优参数组合
	RF[n_estimators, max_depth, min_samples_leaf, criterion]	ET[n_estimators, max_depth, min_samples_leaf, criterion]	Ada[n_estimators, learning_rate]	XGB[n_estimators, max_depth, learning_rate]
KC1	[50,1,1, 'entropy']	[30,4,8, 'entropy']	[100,1]	[100,6,0.3]
PC5	[50,1,1, 'entropy']	[50,1,1, 'gini']	[50,0.01]	[1000,6,0.001]
JM1	[30,1,1, 'gini']	[30,4,4, 'gini']	[1000,0.001]	[500,3,0.01]

3.2 堆叠集成模型性能比较

表 4 展示了堆叠集成（Stacking_LR）与调优过后的基于树的模型在 F_1 得分、AUC 方面的预测性能比较。根据这两个评价指标的平均值得分，发现堆叠集成的预测性能非常优秀，并且在几乎所有数据集上表现一致。此外，XGB 也显示出竞争力强的预测性能，在 JM1 数据集上，其 F_1 值和 AUC 值与堆叠集成相当。另一方面，Ada 在所有数据集上的性能度量平均值最低的，代表它的表现最差。

表 4 调优后的基准模型与 stacking 集成的模型性能对比

数据集	度量	RF	ET	Ada	XGB	Stacking_LR
KC1	F_1	62.03%	63.11%	59.54%	60.92%	**63.13%**
	AUC	68.04%	68.12%	63.71%	65.16%	**68.25%**
PC5	F_1	68.35%	66.55%	67.42%	69.86%	**70.69%**
	AUC	73.23%	73.29%	72.32%	75.96%	**76.40%**
JM1	F_1	61.40%	62.31%	61.89%	**64.94%**	64.55%
	AUC	67.92%	69.16%	69.56%	**71.01%**	70.88%
F_1 平均值		63.93%	63.99%	62.95%	65.24%	**66.12%**
AUC 值平均值		69.73%	70.19%	68.53%	70.71%	**71.84%**

4 结语

在本实证研究中，探讨了使用经过调优的树的集成模型为基分类器，使用逻辑斯特回归为元分类器来构建堆叠集成，并将其应用于软件缺陷预测领域。研究结果表明，基于调优树的集成模型的堆叠集成在预测性能上优于其他基于微调树的集成模型。

作为未来的研究方向，可以考虑研究不同的异构集成方法，例如投票，并将其与所提出的集成模型的使用进行比较。此外，可以探索超参数优化对堆叠集成的影响，并使用和比较不同的元分类器。

参考文献：

[1]MALHOTRA R.A systematic review of machine learning techniques for software fault prediction[J].Applied soft computing, 2015,27:504-518.

[2]ALJAMAAN H, ALAZBA A.Software defect prediction using tree-based ensembles[C]//Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering.New York: ACM,2020:1-10.

[3]OSMAN H, GHAFARI M, NIERSTRASZ O. Hyperparameter optimization to improve bug prediction accuracy[C]// Proceedings of the 2017 IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE). Piscataway:IEEE,2017:33-38.

[4]KHAN F, KANWAL S, ALAMRI S, et al. Hyper-parameter optimization of classifiers, using an artificial immune network and its application to software bug prediction[J].IEEE access, 2020, 8: 20954-20964.

[5]CHEN T, GUESTRIN C.XGBoost: a scalable tree boosting system[C]//Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16).New York:ACM,2016:785-794.

[6]WOLPERT D H.Stacked generalization[J].Neural Netw, 1992, 5: 241-259.

[7]TANTITHAMTHAVORN C, MCINTOSH S, HASSAN A E, et al. An empirical comparison of model validation techniques for defect prediction models[J].IEEE transactions on software engineering, 2017, 43:1-18.

[8]PEDUZZI P, CONCATO J, KEMPER E, et al.A simulation study of the number of events per variable in logistic regression analysis[J].Journal of clinical epidemiology, 1996, 49: 1373-1379.

[9]KWABENA E B, JACKY W K, AKITO M.On the relative value of data resampling approaches for software defect prediction[J].Empirical software engineering, 2019,24:602-636.

[10]BERGSTRA J, BARDENET R, BENGIO Y, et al. Algorithms for hyper-parameter optimization[C]//Proceedings of the 24th International Conference on Neural Information Processing Systems.New York:Curran Associates Inc.,2011:2546-2554.

【作者简介】

丁浩杰（1997—），男，山西晋城人，硕士，助教，研究方向：软件缺陷预测、网络与信息安全等。

（投稿日期：2024-05-13）