Software Engineering and Applications 软件工程与应用, 2023, 12(6), 975-988
Published Online December 2023 in Hans. https://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2023.126096

Hans汉斯

集成特征选择的代价敏感Boosting软件缺陷预测方法

唐鹤龙，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：2023年11月7日；录用日期：2023年12月21日；发布日期：2023年12月29日

摘　要

软件中潜在的缺陷会产生严重的后果，通过使用软件缺陷预测技术可以及时地检测到模块中的缺陷。然而，由于软件缺陷数据集中的类不平衡和高维度特征问题，会导致模型的预测性能下降，因此提出了一种集成特征选择的代价敏感Boosting软件缺陷预测方法(Cost-Sensitive Boosting for Feature Selection, CSBFS)。CSBFS首先采用了一种代价敏感的特征选择算法，该算法先计算每个特征对预测结果的贡献值，并根据不同错误类别的代价对贡献值进行调整，选择具有正向贡献的特征作为特征子集，解决了高维度特征的问题；接下来，将这个特征选择算法嵌入进Boosting算法中，在Boosting的每一轮迭代中，为每个基学习器选择合适的特征子集，从而增加了基学习器之间的多样性；此外，通过调整错误类别的权重，为第一类错分样本赋予更高的权重，以缓解类别不平衡问题，进一步提高了预测效果。在20个公开数据集上进行实验，以F-measure、Recall、AUC、G-mean等作为评价指标，实验结果验证了CSBFS方法的有效性。

关键词

软件缺陷预测，代价敏感，特征选择，集成学习

Cost Sensitive Boosting Software Defect Prediction Method for Integrated Feature Selection

Helong Tang, Yingmei Li

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

Received: Nov. 7th, 2023; accepted: Dec. 21st, 2023; published: Dec. 29th, 2023

文章引用: 唐鹤龙, 李英梅. 集成特征选择的代价敏感 Boosting 软件缺陷预测方法[J]. 软件工程与应用, 2023, 12(6): 975-988. DOI: 10.12677/sea.2023.126096

Abstract

Potential defects in software can have serious consequences and can be detected in a timely manner by using software defect prediction techniques. However, the problem of class imbalance and high dimensional features in the software defect dataset can lead to a degradation of the model's prediction performance, so a cost-sensitive Boosting for Feature Selection (CSBFS) method for software defect prediction with integrated feature selection is proposed. CSBFS method first employs a cost-sensitive feature selection algorithm. This algorithm first calculates the contribution value of each feature to the prediction result, adjusts the contribution value according to the cost of different error categories, and selects features with positive contribution as a feature subset, which solves the problem of high-dimensional features. Next, this feature selection algorithm is embedded into the Boosting algorithm, and a suitable feature subset is selected for each base learner in each iteration of Boosting, thus increasing the diversity among base learners. In addition, the prediction effect is further improved by adjusting the weights of the wrong categories and assigning higher weights to the first misclassified samples to alleviate the category imbalance problem. Experiments are conducted on 20 public datasets with F-measure, Recall, AUC, G-mean, etc. as evaluation indexes, and the experimental results validate the effectiveness of the CSBFS method.

Keywords

Software Defect Prediction, Cost-Sensitive, Feature Selection, Ensemble Learning

1. 引言

对于开发团队来说，提供安全可靠的软件是一个基本的要求。然而，软件中潜在的缺陷对软件的质量构成了严重的威胁，在软件开发的各个阶段中，由于人为因素、沟通问题、技术挑战等不可避免地引入各种缺陷。为此需要在软件发布前进行测试，找出软件中存在的问题并修复。在实际的开发过程中，对软件进行完备的测试需要投入大量的人力，而项目留给测试的资源往往是有限的，因此需要对那些最可能存在缺陷的模块进行测试。

软件缺陷预测(Software Defect Prediction, SDP)利用历史数据训练机器学习、数据挖掘或统计模型，并建立预测模型来预测未来引入的实例(如文件、模块或类)是有缺陷还是无缺陷[1]。准确的预测结果可以指导软件测试人员更有效地分配测试资源。软件缺陷预测的流程如图 1 所示，研究人员收集与软件开发相关的数据，包括代码库、版本控制历史、缺陷报告、代码度量等，从收集的数据中提取有助于预测缺陷的特征。接着将数据集划分为训练集和测试集，选择合适的机器学习、数据挖掘或统计模型作为学习器训练预测模型并使用测试集评估模型的性能。

大多数软件在实际运行中都是正常的，因此采集到的软件缺陷数据集中缺陷类别的样本数量远远少于非缺陷类别[2]，导致模型在预测时对数量较多的类别(非缺陷类)更敏感，而对数量较少的类别(缺陷类)效果较差。而实际情况与数据集的分布相反，对缺陷类的预测精确度要求比非缺陷类更高。软件缺陷预测的预测结果有错误与正确两种，预测错误会对项目进展产生不利影响。其中错误的预测又分为两种，

分别是将有缺陷的模块预测为无缺陷的模块和将无缺陷的模块预测为有缺陷的模块，第一类错误会将有缺陷的模块留在软件中，对软件的质量产生重大的不利影响，而第二类错误会增加测试的模块数量，增加测试成本。与第一类错误相比，第二类错误的代价成本明显更低，因此需要着重增加预测模型对缺陷类的预测效果。最近的相关研究表明，将代价敏感引入缺陷预测模型中可以有效的解决类不平衡引起的少数类预测效果不佳的问题[3]。

软件缺陷数据集的采集往往有不同的来源，如缺陷跟踪系统、版本控制系统、代码审查工具等，每个来源都可能提供不同或相同类型的信息，导致高维度和冗余特征的问题。高维度和冗余特征会使得数据分析和模型训练变得困难，模型容易变得过于复杂，过度拟合训练数据，从而在未见数据上的泛化能力下降[4]。常见的解决方法是进行特征选择或者使用降维技术(如主成分分析、因子分析等)，将高维度数据映射到低维度空间，保留主要信息的同时降低维度。

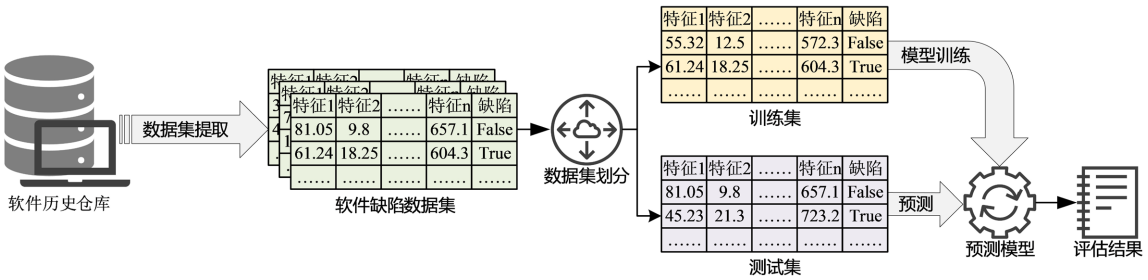

Figure 1. Software defect prediction process
图 1. 软件缺陷预测流程

当前研究人员用于软件缺陷预测的方法主要包括深度学习(Deep Learning, DL)、随机森林(Random Forest, RF)、支持向量机(Support Vector Machines, SVM)、逻辑回归(Logistic Regression, LR)、朴素贝叶斯(Naive Bayes, NB)、集成方法(Ensemble Methods, EL)等[5]。然而这些方法在面对上述提到的类不平衡、误分类代价不同、高维度和冗余特征等问题时不能得到良好的分类结果。

李等人[6]提出了 CSBst (Cost Sensitive Boosting)算法，该方法使用决策树桩(decision stump)作为集成模型的基分类器，在模型训练过程中增加产生第一类错误样本的权重，并在分类器最终集成的时候采用阈值移动的方法提高缺陷类的预测精度。其在集成阶段将阈值偏向于缺陷类的做法虽然会提高缺陷类的召回率，但由于非缺陷类占多数的数据分布，会使得整体的准确率大大降低。李勇等人[7]提出 C-AdaBoost 模型的集成算法，使用逐步回归前向算法选择最优特征子集，并在模型训练过程中使用 C 函数寻找每个基分类器的最优学习率，该模型优化了集成函数并在一定程度上降低了特征的维度，从而实现了更好的分类效果。Guo 等人[8]提出了 RSMOTE 算法，用于从高维采样空间生成少数类样本，以处理软件缺陷预测中的不平衡分布问题，通过合成少数类的方法可以从数据上解决不平衡问题，但合成的样本无法捕捉到原始数据集中的所有细微特征和分布，且与真实世界的数据可能不匹配。因此可能产生过拟合和降低预测精度的问题。Lu 等人[9]提出 AEUB (Adaptive Ensemble Undersampling-Boost)算法用于处理不平衡学习问题，将欠采样集成(Ensemble of Undersampling, US)技术、Real Adaboost、代价敏感的权重修改和自适应边界决策策略相结合，构建了一个混合算法。该方法将多种优化方法应用于预测模型上，可以有效的提升模型的预测性能，但同时也提高了模型的复杂度。饶珍丹等人[10]提出了一种多层次过采样集成的不平衡数据缺陷预测模型(AJCC-Ram + XGBoost, XG-AJC)，用于处理软件缺陷预测中不平衡数据的分类问题。该模型采用了多层次的过采样集成策略，针对类边缘和类中心分别采用不同的过采样算法来生成新的样本，并对生成的样本进行了噪声过滤处理。与传统的采样算法相比，这种方法在预测性能上表现

更出色。

针对上述问题，本文提出了一种集成特征选择的代价敏感 Boosting 软件缺陷预测方法(Cost-Sensitive Boosting for Feature Selection, CSBFS)，在特征选择与集成学习两个阶段同时使用代价敏感方法，解决不平衡数据集中少数类预测性能差的问题。提出一种新的特征选择方法，根据样本特征的边际贡献与代价函数得到特征的贡献值，并选择其中产生正面贡献的特征作为最终的特征集合。将特征选择融入到集成学习的过程之中，在每个基分类器训练前为其选择最合适的特征，以提高其预测性能，避免产生过拟合的同时提高各基分类器的多样性，以提升加权集合阶段的性能。

2. 相关工作

2.1. 代价敏感方法

代价敏感方法是一种在机器学习中用于处理不同类别分类错误代价不同的技术[11]。在许多实际应用中，不同类别的错误可能会导致不同的后果，因此传统的分类算法可能无法很好地满足实际需求。代价敏感方法通过考虑不同类别的代价，帮助机器学习模型更好地适应这种情况。在一些算法中，如逻辑回归(LR)和支持向量机(SVM)，可以通过调整样本的权重来引入代价敏感性。错误分类的样本被赋予更高的权重，从而在模型训练中更加关注这些错误。如图 2 所示，LR 和 SVM 应用代价敏感方法后在多个数据集上召回率均有显著的提升。

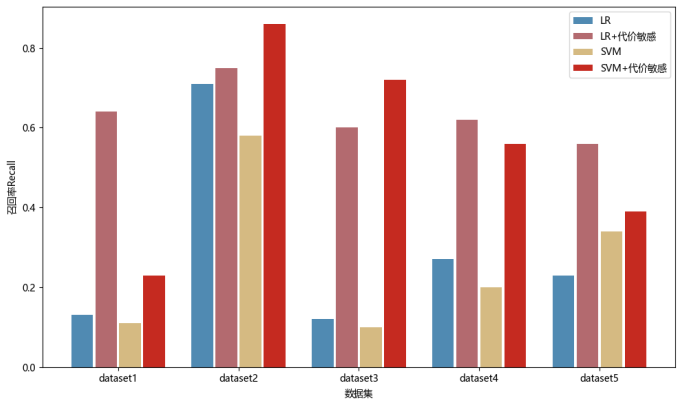

Figure 2. Comparison before and after using cost-sensitive methods
图 2. 使用代价敏感方法前后对比

代价矩阵是代价敏感分方法的核心[12]，在二分类问题中它是一个二维矩阵，其中的元素表示在不同真实类别和预测类别之间的分类代价。如表 1 所示，行表示预测类，列表示实际类。为表方便用“–”表示负类别，“+”表示正类别，$C_{-,-}(x)$表示真反例(True Negatives, TN)的代价，$C_{+,+}(x)$表示真正例(True Positive, TP)的代价，$C_{-,+}(x)$表示假正例(False Positives, FP)的错分代价，$C_{+,-}(x)$表示假反例(False Negatives, FN)的错分代价。代价矩阵反映了在错误分类时可能产生的不同后果。

Table 1. Cost matrix
表 1. 代价矩阵

	Predictedtarget	
Actualtarget	–	+
–	$C_{-,-}(x)$	$C_{-,+}(x)$
+	$C_{+,-}(x)$	$C_{+,+}(x)$

2.2. 特征选择方法

特征选择是指从原始特征集中选择一部分最具有代表性和信息量的特征，用于模型训练[13]。特征选择的目的是在减少特征维度的同时，保留对问题最有帮助的信息，以提高模型性能、降低过拟合风险、加快训练速度。软件缺陷预测中常用的特征选择方法主要有两类：过滤式(Filter)和包裹式(Wrapper)。如图 3 所示，过滤式特征选择是在模型训练之前对特征进行筛选，它通常将特征之间的统计量或相关性作为评价准则来评估特征的重要性。过滤式方法通常不涉及模型的训练，因此计算效率较高[14]。包裹式特征选择则是在特征选择过程中使用机器学习模型，通过不断训练模型来评估不同特征的重要性。包裹式方法的优点是可以考虑特征与目标之间的影响，更能准确地评估特征的重要性[15]。而由于需要多次训练模型，它的计算成本通常比过滤式高。

2.3. Boosting 集成学习

集成学习是一种机器学习方法，它将众多的弱学习器通过一定方法组合成一个强学习器，从而提高模型预测的准确性[16]。Boosting 方法是集成学习中的一种，通过迭代地训练模型，并增加错分样本权重，从而使模型更关注于错误分类的样本，进而提高整体性能。如图 4 所示，首先初始化每个样本的权重都为$1/M$，其中M是样本总数。接着使用当前的样本权重和样本训练弱学习器，计算学习器的错误率，并增加分类错误的样本在下一轮迭代中的权重。重复上一步直至迭代训练完所有的学习器，将各个弱学习器按其权重进行加权组合，形成一个强学习器。

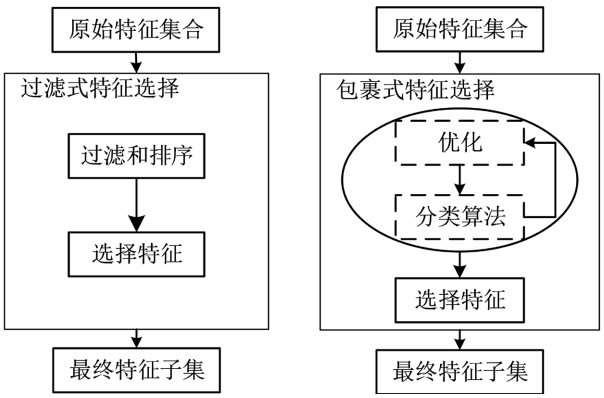

Figure 3. Workflow for filter and wrapper feature selection
图 3. 过滤式和包裹式特征选择的工作流程

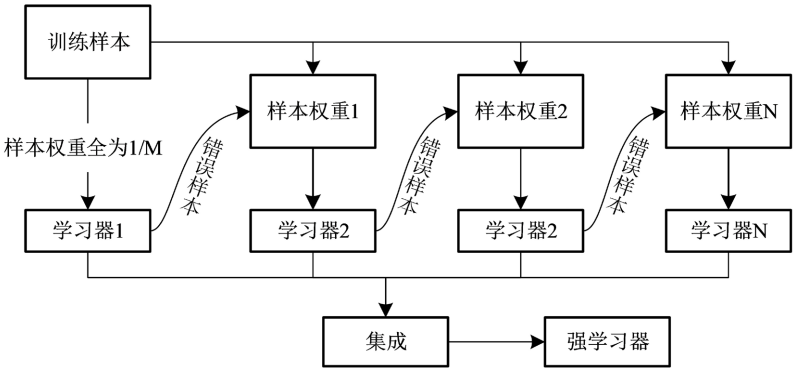

Figure 4. Boosting algorithm process
图 4. Boosting 算法流程

3. CSBFS 方法

集成特征选择的代价敏感 Boosting 软件缺陷预测方法主要由两部分构成：代价敏感的特征选择算法和代价敏感的 Boosting 算法。由于软件工程实践中对不同错分的代价不同，因此需要在特征选择与集成两个阶段分别使用代价敏感方法，最后将两个算法组合成为 CSBFS 方法。

3.1. 代价敏感的特征选择算法

为了有效地识别软件缺陷数据集中的冗余和不相关特征，并且保留与缺陷类别相关度高的特征，本文提出了一种代价敏感的特征选择算法。该方法基于合作博弈论，通过计算各样本特征的 Shaplely 值[17]确定特征的边际贡献。并在此基础上加入代价敏感思想，对产生第一类、第二类错误的样本特征根据代价矩阵降低相应的贡献值。最后将各特征的贡献值相加取平均作为平均贡献值，并选择其中对预测结果起正向作用的特征作为最终的特征子集。

1) 计算特征贡献值

使用 Shaplely 值来度量特征对预测结果的贡献度，对于每个特征，将其在不同特征顺序下的平均边际贡献相加，得到该特征的 Shapley 值：

$$\phi_i = \sum S \subseteq x \setminus \{x_i\} \frac{|S|!(n-|S|-1)!}{n!} \left(val_x\left(S \cup \{x_i\}\right) - val_x\left(S\right)\right) \tag{1}$$

其中 ϕ_i 是第 i 个特征的 Shapley 值，S 是除 x_i 外的所有特征子集，x 是计算贡献值的样本实例 $\{x_1, x_2, \ldots, x_n\}$，$n$ 为特征总数。$\frac{|S|!(n-|S|-1)!}{n!}$ 是特征子集 S 的权重。$val_x(S)$ 是在特征子集 S 上的模型预测，可以用如下公式计算：

$$val_x(S) = \int \hat{f}(x) d\mathbb{P}_{x \notin S} - E_X\left(\hat{f}(x)\right) \tag{2}$$

其中 $\hat{f}(x)$ 是 x 的预测，$E_X\left(\hat{f}(x)\right)$ 是预测平均值，$\int \hat{f}(x) d\mathbb{P}_{x \notin S}$ 是对不属于特征子集 S 中的特征进行多次积分。

由(1)式求得的 Shapley 值 ϕ_i 是一个任意实数，可以是正数、负数或零。正数表示该特征对于模型预测有正向贡献，负数表示负向贡献，而零表示该特征对预测没有影响。由于所求的 Shapley 值没有大小限制，过大或过小都会影响分类性能，因此通过式(3)将其值映射到 $[-1,1]$ 之间，得到归一化的 Shapley 值 φ_i，同时便于接下来的代价敏感实现。

$$\varphi_i = \frac{e^{\phi_i} - e^{-\phi_i}}{e^{\phi_i} + e^{-\phi_i}} \tag{3}$$

2) 引入代价敏感方法

从表 1 中可知 $C_{+,-}(x)$ 表示第一类分类错误的代价，$C_{-,+}(x)$ 表示第二类分类错误的代价，通常将第二类错误代价设为 1，而第一类错误的代价大于 1，具体的值视实际情况而定。代价敏感方法的计算可以如下公式实现：

$$cost_x = \begin{cases} 1 & y_x = h(x) \\ e^{-C_{+,-}(x)} & y_x = 1 \; and \; h(x) = 0 \\ e^{-C_{-,+}(x)} & y_x = 0 \; and \; h(x) = 1 \end{cases} \tag{4}$$

$$\varphi_i^{'} = \varphi_i \times cost_x \tag{5}$$

其中 y_x 代表实际值，$h(x)$ 代表模型预测值，$cost_x$ 代表样本 x 的代价调整函数。对于分类正确的样本，$y_i = h(x_i)$，其特征的 φ_i 值保持不变。对于第一类错分样本，$y_i = 1\ and\ h(x_i) = 0$，e 的指数项小于零，其值由代价矩阵控制，减少相应特征的 φ_i 值。对于第二类错分样本，$y_i = 0\ and\ h(x_i) = 1$，e 的指数项同样小于零，但由于第二类错分的代价小于第一类，因此对应特征的 φ_i 值减少量少于第一类。综合考虑上述的三种情况，最终得到代价敏感的归一化 Shapley 值 φ_i。

3) 进行特征选择

在选择特征之前，需要考虑特征在所有样本中的贡献情况，特征的平均贡献计算方法如下，其中 m 为样本的总数量，所得的 ω_i 为第 i 个特征在当前数据集下对模型的贡献程度。

$$\omega_i = \frac{1}{m}\sum_{j=1}^{m}\varphi_{i,j}^{'} \tag{6}$$

特征选择的目的是选择对预测贡献最大的特征，因此根据(6)式求得的特征贡献度选择最终的特征。特征选择的方法如下，选择对预测模型产生正向贡献的特征加入到最终特征子集 $\mathcal{R}$ 中，最终的 $\mathcal{R}$ 即为所求特征集合。

$$\mathcal{R} = \left\{\omega_i \middle| \omega_i \in \{\omega_1, \omega_2, \cdots, \omega_n\}, \omega_i > 0\right\} \tag{7}$$

3.2. 代价敏感的 Boosting 算法

为了使预测模型更关注于第一类分类错误问题，将代价敏感方法引入 Boosting 算法中。通过调整算法训练中数据的分布权重，使得下一个学习器更关注被我们提高权重的样本。对于第一个学习器，初始化每个样本的权重都相同，即 $D_1(x,y) = \frac{1}{M}$。计算学习器的错误率 $\gamma_n = \sum_m D_n(x_m, y_m) y_m f_n(x_m)$，并根据错误率计算学习器的权重 $\beta_n = \frac{1}{2} ln\left(\frac{1+\gamma_n}{1-\gamma_n}\right)$。

前面的方法与 Adaboost [18]方法相同，不同之处在接下来的数据权重更新阶段，根据代价矩阵更新样本权重，权重更新方法入下所示。

$$D_{n+1}(x,y) = \frac{D_n(x,y)\, exp\left(-cost_x \beta_n y_m f_n(x_m)\right)}{Z_n} \tag{8}$$

$D_{n+1}(x,y)$ 为更新完的数据权重，通过指数内部的 $-cost_x \beta_n y_m f_n(x_m)$ 代价敏感地调整数据权重，指数中的 $cost_x$ 参数从式(4)取得，由代价矩阵将样本数据划分为三个重要等级，按重要程度分别是第一类错误样本、第二类错误样本和分类正确样本，从而在下一轮的训练中按照这三个重要等级训练学习器。迭代训练完所有的学习器后按照式(9)将所有弱学习器集成为一个强分类器，集成时按照错误率加权弱学习器并对加权结果使用符号函数 $sign()$ 得到最终的预测结果。

$$G(x) = sign\left(\sum_{n=1}^{N} \beta_n f_n(x)\right) \tag{9}$$

3.3. CSBFS 方法实现

CSBFS 由 2.1 中的特征选择算法和 2.2 中的集成算法共同组成，将代价敏感的特征选择算法融入到集成算法之中。通过在每个弱学习器前添加特征选择算法，选择最适合当前弱学习器的特征子集以提高其预测性能，CSBFS 方法的流程如图 5 所示。由于各个弱学习器的关注点不同，其重要特征也各不相同，因此特征选择可以提高各弱学习器的多样性，最终提高集成模型的预测性能。CSBFS 方法将代价敏感思想同时应用在特征选择与集成两个阶段，以减少第一类分类错误的产生，提高预测性能。

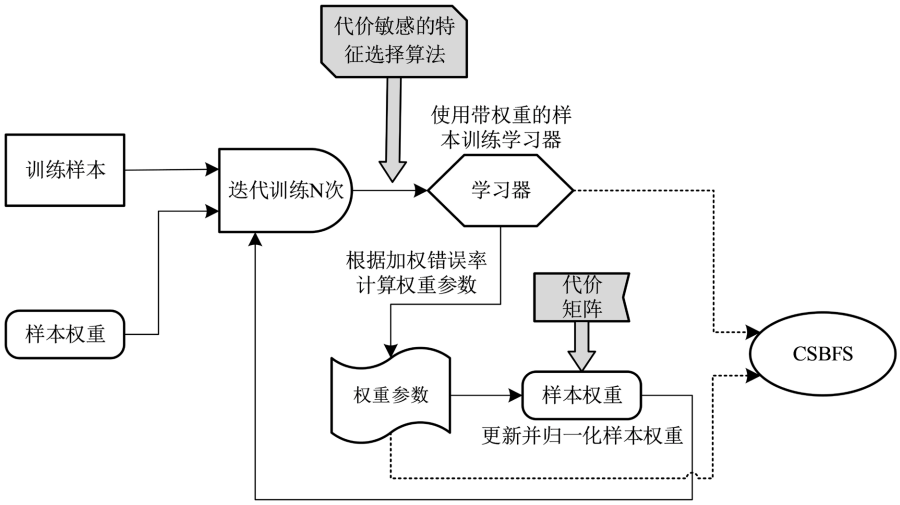

Figure 5. CSBFS algorithm process
图 5. CSBFS 方法流程

4. 实验研究

4.1. 数据集简介

本文使用 Shepperd 等人[19]对 NASAMDP 缺陷数据集清洗后的 CleanNASA 数据集、D'Ambros 等人[20]收集的 AEEEM 数据集和 Wu 等人[21]收集的 ReLink 数据集共三个数据集进行试验。其中 CleanNASA 中包含 CM1、JM1、KC1、KC3、MC1、MC2、MW1、PC1、PC2、PC3、PC4、PC5 共 12 个数据集，缺陷类占比最低为 2.31%，最高为 27.57%，类不平衡现象较为突出。AEEEM 中包含 EQ、JDT、LC、ML、PDE 共 5 个数据集，特征数量为 21 个，缺陷类占比最低为 8.26%，最高为 39.81%。ReLink 中包含 Apache、Safe、Zxing3 个数据集，特征数量为 26 个，缺陷类占比最低为 29.57%，最高为 50.52%，类别分布比较平衡，数据集的详细信息如表 2 所示。

4.2. 实验设置

本文使用表 2 中 3 个来源的共 20 个数据集进行实验，采用 5 折交叉法，进行 10 次实验取平均值。选取 RandomForest、DecisionTree、AdaBoost 、CAdaboost 和 CSBst 方法作为对比实验，对各实验的结果进行统计。并选取不同的代价因子 $C_{+,-}(x)$ 进行实验，以探究代价矩阵对预测结果的影响以及代价因子取何值时预测效果最佳。

Table 2. Dataset information
表 2. 数据集信息

数据集	粒度	特征数	模块总数	缺陷模块	缺陷占比(%)
CM1	函数	37	327	42	12.84
JM1	函数	21	7782	1672	21.49
KC1	函数	21	1183	314	26.54
KC3	函数	39	194	36	18.56
MC1	函数	38	1988	46	2.31
MC2	函数	39	125	44	35.20
MW1	函数	37	253	27	10.67
PC1	函数	37	705	61	8.65

Continued

PC2	函数	36	745	16	2.15
PC3	函数	37	1077	134	12.44
PC4	函数	37	1287	177	13.75
PC5	函数	38	1711	471	27.53
EQ	类	61	324	129	39.81
JDT	类	61	997	206	20.66
LC	类	61	691	64	9.26
ML	类	61	1862	245	13.16
PDE	类	61	1497	209	13.96
Apache	文件	26	194	98	50.52
Safe	文件	26	56	22	39.29
Zxing	文件	26	399	118	29.57

由于软件缺陷预测是一个二分类问题，因此可以用表 3 所示的混淆矩阵表示缺陷预测结果，其中 TN (True negative)表示无缺陷且预测正确的样本数量，TP (True positive)表示有缺陷且预测正确的样本数量，FP (False positive)表示无缺陷但预测为有缺陷的样本数量，FN (False negative)表示有缺陷但预测为无缺陷的样本数量。根据混淆矩阵可以计算出其它评价指标如精确率(Precision)、召回率(Recall)、误报率(FPR)、F1 分数(F-measure)、G 均值(G-mean)。其中 Precision 表示模型预测为缺陷类的样本中有多少是真正有缺陷的，计算公式为：

$$\text{Precision}=\frac{TP}{TP+FP}$$

Recall，也叫做 TPR，表示模型预测正确的缺陷类样本占所有实际有缺陷样本的比例，计算公式为：

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

FPR 表示无缺陷样本被错误地预测为有缺陷样本所占的比例，算公式为：

$$FPR=\frac{FP}{FP+TN}$$

F-measure 综合考虑了精确率和召回率，取值范围是 0 到 1，得分越接近 1，表示模型在精确率和召回率之间取得了良好的平衡，模型性能越好，计算公式为：

$$F1=2\times\frac{Precison\times Recall}{Precison+Recall}$$

G-mean 是一种综合评估指标，同时考虑对缺陷类和无缺陷类的预测能力，取值范围是 0 到 1，G-mean 值越接近 1，表示模型在缺陷类和无缺陷类上的预测能力都很好。对于不平衡数据集，G-mean 值越高表明模型有效地识别了两个类别，计算公式为：

$$G-mean=\sqrt{\text{Precision}\times\text{Recall}}$$

AUC 表示模型在不同阈值设置下 TPR 与 FPR 之间的面积。取值范围：0.5 到 1。当 AUC 等于 0.5 时，表示模型性能等同于随机猜测。AUC 越接近 1，表示模型对正类别和负类别的分类能力越好。

本实验使用 Python 语言，开发环境为 PyCharm2023，使用的操作系统为 Windows10，CPU 为 AMD Ryzen 5 5600 6-Core Processor 3.50 GHz，内存为 16GB。

Table 3. Confusion matrix
表 3. 混淆矩阵

	Predicted target	
Actual target	-	+
-	TN(True negative)	FP(False positive)
+	FN(False negative)	TP(True positive)

4.3. 结果分析

4.3.1. 实验效果对比

本文对 20 个数据集使用不同预测模型进行实验，分别记录下它们的 F1、Recall、AUC、G-mean 等评价指标。如表 4 所示，为 6 种预测模型在 20 个数据集上预测结果的 F1 值，实验结果表明，本文提出的 CSBFS 方法在绝大多数的数据集上的预测效果都比其他 5 个方法具有更高的 F1 值，只在 2 个数据集上小幅度地落后于具有最高 F1 值的分类算法，在 4 个数据集上的 F1 值与最高的预测算法持平。在 PC2 数据集上，由于该数据集的缺陷占比低至 2.15%，所有预测算法的性能都比较低，随机森林算法的 F1 值更是低至 0，这表明其对于缺陷类别的预测效果为零，而 CSBFS 算法相较于其他算法均有一定程度的领先。在 KC3、MW1、PC1 数据集中，CSBFS 算法相对于其他算法的提高程度最高。

Table 4. F1 of 6 algorithms on 20 datasets
表 4. 6 种算法在 20 个数据集上的 F1 值

	RF	DT	AdaBoost	CadaBoost	CSBst	CSBFS
CM1	0.14	0.25	0.27	0.29	0.3	**0.36**
JM1	0.28	0.33	0.33	0.35	0.34	**0.41**
KC1	0.41	0.39	0.39	0.39	**0.43**	**0.43**
KC3	0.14	0.32	0.40	0.35	0.35	**0.45**
MC1	0.25	0.36	0.37	0.40	0.18	**0.41**
MC2	0.47	0.50	0.50	0.49	0.5	**0.52**
MW1	0.23	0.24	0.27	0.24	0.3	**0.32**
PC1	0.31	0.34	0.33	0.36	0.36	**0.41**
PC2	0.00	0.06	0.06	0.05	**0.1**	**0.1**
PC3	0.19	0.32	0.31	0.22	0.36	**0.37**
PC4	0.55	0.53	0.52	0.49	0.53	**0.58**
PC5	0.49	0.48	0.49	0.49	**0.53**	**0.53**
EQ	**0.74**	0.64	0.64	0.66	0.69	0.69
JDT	0.59	0.55	0.54	0.54	0.6	**0.64**
LC	0.35	0.35	0.34	**0.38**	0.31	**0.38**
ML	0.32	0.34	0.35	0.33	0.44	**0.45**
PDE	0.31	0.30	0.30	0.34	0.38	**0.39**
Apache	**0.74**	0.66	0.68	0.62	0.72	0.71
Safe	0.61	0.59	0.62	0.55	0.63	**0.66**
Zxing	0.42	0.41	0.44	0.30	0.49	**0.53**

图 6、7、8 分别展示了 6 种算法在 20 数据集上的 Recall、AUC 和 G-mean。从图 6 中可以看出 CSBFS 在绝大多数数据集上的 Recall 都优于其他 5 个算法，这表明 CSBFS 算法相比其他算法对少数类拥有更高的预测性能，对缺陷类样本更加敏感。图 7 展示的柱状图表明 CSBFS 算法在多数数据集上比其他算法的 AUC 指标更高，说明其在不同概率阈值下都能够有效地区分有缺陷和无缺陷的样本，CSBFS 算法的整体

预测性能比其他 5 个算法更高。从图 8 中可以看出 CSBFS 算法在 20 个数据集上都拥有较高的 G-mean 值，相较于其他 5 个算法，除了在 Apache 和 Safe 数据集上排在第二，在其他的数据集上都持平或一定程度的领先于其他 5 个算法。这表明 CSBFS 算法在缺陷类和无缺陷类之间的平衡较好，在类别不平衡问题中的性能较高。

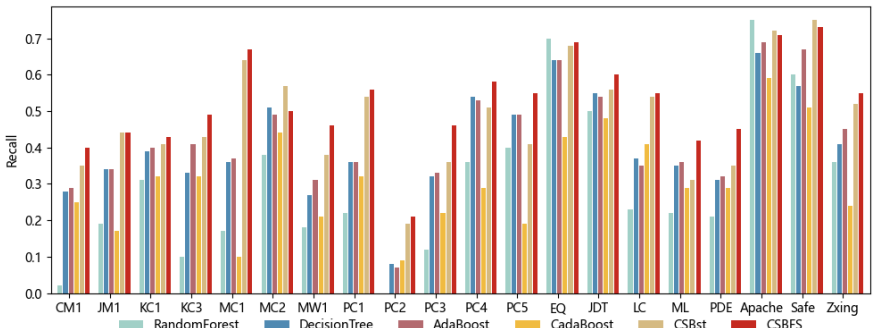

Figure 6. Recall of 6 algorithms on 20 datasets
图 6. 6 种算法在 20 个数据集上的 Recall

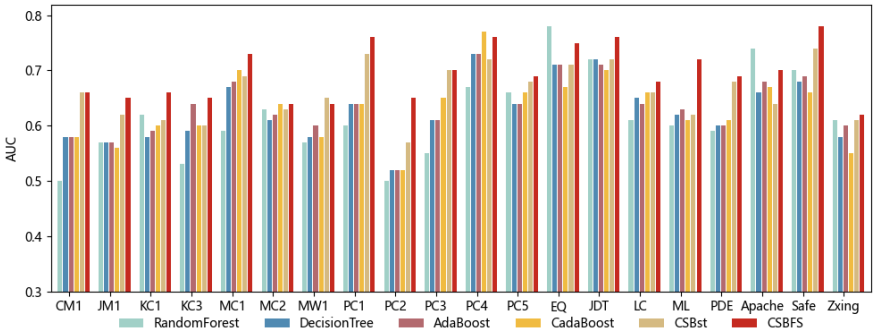

Figure 7. AUC of 6 algorithms on 20 datasets
图 7. 6 种算法在 20 个数据集上的 AUC

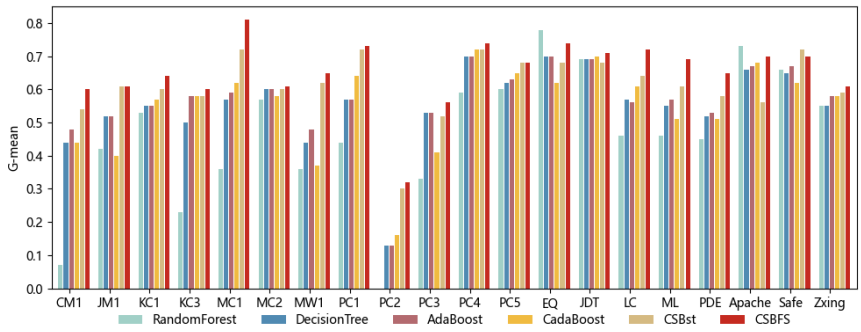

Figure 8. G-mean of 6 algorithms on 20 datasets
图 8. 6 种算法在 20 个数据集上的 G-mean

由上述可知除了个别数据集，CSBFS 算法在大多数数据集上的各项评价指标都领先于其他算法。经观察可知这些数据集分别是 MC2、EQ、Apache、Safe、Zxing，这些数据集的缺陷占比分别是 35.2%、39.81%、50.52%、39.29%、29.57%，都是比较平衡的数据。由此可知 CSBFS 算法在平衡数据上的预测效果与其他算法区别不大，而在不平衡数据上相比其他算法有着较大的领先，因此 CSBFS 算法更适合于不平衡数据的预测。

4.3.2. 缺陷占比的影响

为了探究 CSBFS 算法在不同缺陷占比下的预测性能，将数据集的缺陷占比及它们的评价指标绘制成图 9 所示的折线图，其中横坐标为缺陷占比，纵坐标为各指标的值。由图可知随着缺陷占比的不断提升，预测模型的性能也随之提升。在缺陷占比较低时预测模型性能提升速度较快，缺陷占比在 20%之后性能提升变得比较平稳，由此可知 CSBFS 算法在预测缺陷占比高于 20%的数据时都会有较好的预测结果。

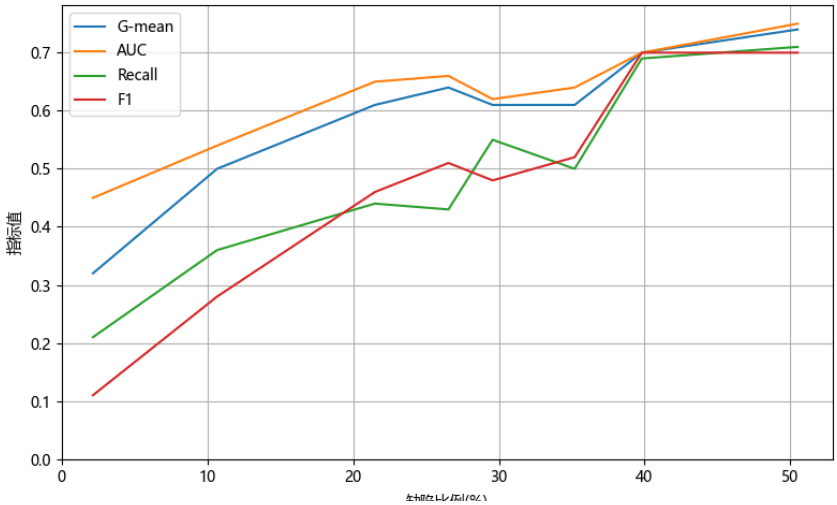

Figure 9. Performance of CSBFS algorithm under different defect proportions
图 9. 不同缺陷占比下 CSBFS 算法的性能

4.3.3. 代价因子的选择

代价敏感方法中代价因子的选择是一个关键点，合适的代价因子可以提高模型的预测性能，为了找出最合适的代价因子，我们使用平衡值[22] (bal)评价指标作为判断是否取得最佳代价因子的依据。bal 综合了 FPR 和 TPR 两个指标，同时考虑 FPR 和 TPR 的影响，是一个综合评价指标，bal 的值越高表明模型预测效果越好，其计算公式为：

$$bal = 1 - \sqrt{\frac{(0 - FPR)^2 + (1 - TPR)^2}{2}}$$

为了探究最佳代价因子的取值，我们取不同的代价因子进行实验，并统计它们的 TPR、FPR 及 bal 值。代价因子取值从 1 开始每次增加 0.1 直到 4 进行实验，每次实验进行 10 次取平均值记录结果，将所有结果标记在图中连线得到代价曲线。如图 10 为 PC3 数据集的代价曲线，由图像可知随着代价因子的取值不断增加，TPR 和 FPR 的值也随着增加，而 bal 的值先是增加，到达最高值 0.75 后开始下降，bal 最大时代价因子的取值为 2.6，因此 PC3 数据集的最佳代价因子为 2.6。从图中可以看出代价因子取值大于 2.9 之后 TPR 趋于平和而 FPR 仍然持续上涨，最终导致预测效果开始下降。

为了探究最佳代价因子的取值规律，对所有数据集按上文中的方法求得最佳代价因子，并将数据集按缺陷占比从小到大排列得到图 11。图 11 展示的为最佳代价因子与缺陷占比的关系，从图中可知随着缺陷占比的增加，最佳代价因子的取值逐渐下降，最佳代价因子与缺陷占比呈负相关。缺陷占比越低即数据集越不平衡时，所取的最佳代价因子就越高，但代价因子的取值也不会一直上涨，从图中来看最高的代价因子在 3.5 左右。随着缺陷占比的提高，即数据集越来越平衡，最佳代价因子的取值也就越来越接近 1。

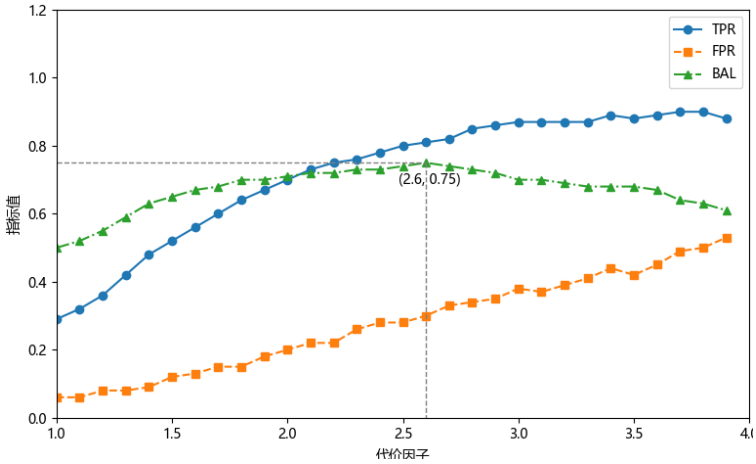

Figure 10. The impact of different cost factors on prediction results
图 10. 不同代价因子对预测结果的影响

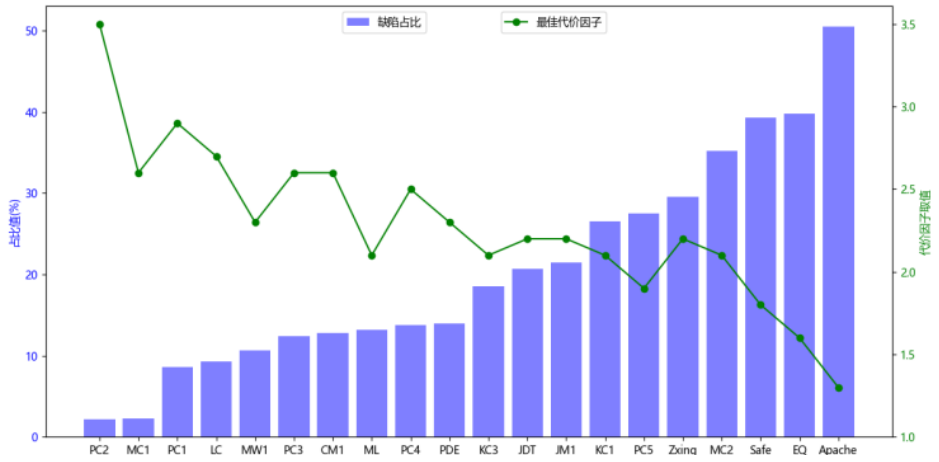

Figure 11. The relationship between the value data sets of the optimal cost factor
图 11. 最佳代价因子的取值数据集的关系

5. 结束语

本文针对软件缺陷预测领域广泛存在的类不平衡及高维度问题，提出了一种集成特征选择的代价敏感 Boosting 软件缺陷预测方法。

首先，为解决高维度问题，提出一种代价敏感的特征选择方法，通过计算各样本特征的边际贡献值来选择特征。同时为了在选择特征子集时更多地保留与缺陷类样本相关的特征，在计算特征贡献度的过程中引入代价敏感思想，根据代价矩阵更改不同特征的贡献值，从而减少错误的产生。

其次，将代价敏感的特征选择方法嵌入进集成学习中，为每个基学习器添加特征选择方法，有效提高了基学习器的预测性能，避免了基学习器之间的同质化，增加了基学习器预测的多样性。同时在集成学习中引入代价敏感思想，在样本迭代时根据样本的错分情况及代价矩阵调节样本权重，可以让第一类错误样本在下一轮训练中得到更多的关注。

此外，由于软件缺陷数据集中不可不免地存在一些噪声数据，而 Boosting 算法在迭代过程中可能会将噪声数据作为分类错误的数据，从而过度关注噪声数据，影响最终的预测性能。因此在未来的研究中，我们会着重探究在 Boosting 方法中噪声可容忍的处理方法及剔除噪声数据的方法。

基金项目

黑龙江省高等教育教学改革项目(SJGY20210457)；2023 年哈尔滨师范大学研究生创新项目(HSDSSCX2023-6)。

参考文献

[1] Li, Z., Jing, X.Y. and Zhu, X. (2018) Progress on Approaches to Software Defect Prediction. *IET Software*, **12**, 161-175. https://doi.org/10.1049/iet-sen.2017.0148

[2] Chen, L., Wang, C. and Song, S. (2022) Software Defect Prediction Based on Nested-Stacking and Heterogeneous Feature Selection. *Complex & Intelligent Systems*, **8**, 3333-3348. https://doi.org/10.1007/s40747-022-00676-y

[3] Eivazpour, Z. and Keyvanpour, M.R. (2021) CSSG: A Cost-Sensitive Stacked Generalization Approach for Software Defect Prediction. *Software Testing, Verification and Reliability*, **31**, e1761. https://doi.org/10.1002/stvr.1761

[4] 陈翔, 沈宇翔, 孟少卿, 等. 基于多目标优化的软件缺陷预测特征选择方法[J]. 计算机科学与探索, 2018, 12(9): 1420-1433.

[5] 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

[6] 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

[7] 李勇, 陈思萱, 贾海, 等. 基于 C-AdaBoost 模型的乳腺癌预测研究[J]. 计算机工程与科学, 2020, 42(8): 1414-1422.

[8] Guo, S., Dong, J., Li, H., *et al*. (2021) Software Defect Prediction with Imbalanced Distribution by Radius-Synthetic Minority Over-Sampling Technique. *Journal of Software*: *Evolution and Process*, **33**, e2362. https://doi.org/10.1002/smr.2362

[9] Lu, W., Li, Z. and Chu, J. (2017) Adaptive Ensemble Undersampling-Boost: A Novel Learning Framework for Imbalanced Data. *Journal of Systems and Software*, **132**, 272-282. https://doi.org/10.1016/j.jss.2017.07.006

[10] 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

[11] 万建武, 杨明. 代价敏感学习方法综述[J]. 软件学报, 2020, 31(1): 113-136. https://doi.org/10.13328/j.cnki.jos.005871

[12] Viaene, S. and Dedene, G. (2005) Cost-Sensitive Learning and Decision Making Revisited. *European Journal of Operational Research*, **166**, 212-220. https://doi.org/10.1016/j.ejor.2004.03.031

[13] 李郅琴, 杜建强, 聂斌, 等. 特征选择方法综述[J]. 计算机工程与应用, 2019, 55(24): 10-19.

[14] Xu, X.L., *et al*. (2021) RFC: A Feature Selection Algorithm for Software Defect Prediction. *Journal of Systems Engineering and Electronics*, **32**, 389-398. https://doi.org/10.23919/JSEE.2021.000032

[15] 张靖. 面向高维小样本数据的分类特征选择算法研究[D]: [博士学位论文]. 合肥: 合肥工业大学, 2014.

[16] Nahar, N., Ara, F., Neloy, M.A.I., *et al*. (2019) A Comparative Analysis of the Ensemble Method for Liver Disease Prediction. 2019 2*nd International Conference on Innovation in Engineering and Technology* (*ICIET*), Dhaka, 23-24 December 2019, 1-6. https://doi.org/10.1109/ICIET48527.2019.9290507

[17] Lundberg, S.M. and Lee, S.I. (2017) A Unified Approach to Interpreting Model Predictions. *Proceedings of the* 31*st International Conference on Neural Information Processing Systems*, Long Beach, 4-9 December 2017, 4768-4777.

[18] Freund, Y. and Schapire, R.E. (1997) A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, **55**, 119-139. https://doi.org/10.1006/jcss.1997.1504

[19] Shepperd, M., Song, Q., Sun, Z., *et al*. (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

[20] D'Ambros, M., Lanza, M. and Robbes, R. (2010) An Extensive Comparison of Bug Prediction Approaches. 2010 7*th IEEE Working Conference on Mining Software Repositories* (*MSR* 2010), Cape Town, 2-3 May 2010, 31-41. https://doi.org/10.1109/MSR.2010.5463279

[21] Wu, R., Zhang, H., Kim, S., *et al*. (2011) Relink: Recovering Links between Bugs and Changes. *Proceedings of the* 19*th ACM SIGSOFT Symposium and the* 13*th European Conference on Foundations of Software Engineering*, Szeged, 5-9 September 2011, 15-25. https://doi.org/10.1145/2025113.2025120

[22] 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