ORESP：基于有序回归的软件缺陷严重程度预测方法 *

贾焱鑫[1]，陈 翔[1,2†]，葛 骅[1]，杨 光[1]，林 浩[1]

(1. 南通大学 信息科学技术学院，江苏 南通 226019; 2. 南京大学 计算机软件新技术国家重点实验室，南京 210023)

摘 要：为提高软件缺陷严重程度的预测性能，通过充分考虑软件缺陷严重程度标签间的次序性，提出一种基于有序回归的软件缺陷严重程度预测方法 ORESP，该方法首先使用基于 Spearman 的特征选择方法来识别并移除数据集内的冗余特征，随后使用基于比例优势模型的神经网络来构建预测模型。通过与五种经典分类方法的比较，所提的 ORESP 方法在四种不同类型的度量下均可取得更高的预测性能，其中基于平均 0-1 误差(MZE)评测指标，预测模型性能最大可提升 10.3%；基于平均绝对误差(MAE)评测指标，预测模型性能最大可提升 12.3%。除此之外，发现使用基于 Spearman 的特征选择方法可以有效提升 ORESP 方法的预测性能。

关键词：软件质量保障；缺陷严重程度预测；有序回归；特征选择；分类

中图分类号：TP311.5 doi: 10.19734/j.issn.1001-3695.2020.07.0249

Oresp: software defect severity prediction based on ordinal regression

Jia Yanxin[1], Chen Xiang[1,2†], Ge Hua[1], Yang Guang[1], Lin Hao[1]

(1. *School of Information Science & Technology*, *Nantong University*, *Nantong* 226019, *China*; 2. *State Key Laboratory for Novel Software Technology*, *Nanjing University*, *Nanjing* 210023, *China*)

Abstract: To improve the prediction performance of the severity of software defects, this paper proposes a defect severity prediction method ORESP based on ordinal regression by considering the order between different labels. This method uses the spearman-based feature selection method to identify and remove redundant features in the data set and then uses a neural network based on the proportional odds model to build prediction models. By comparing with the five classical classification methods, the proposed method ORESP can achieve better prediction performance under four different categories of module metrics. In terms of MZE (average zero-one error) measure, the performance of the prediction models can be improved by 10.3% at most; in terms of MAE (mean absolute error) , the performance of the prediction models can be improved by 12.3% at most. In addition, using the feature selection method based on Spearman can effectively improve the prediction performance of ORESP.

Key words: software quality assurance; defect severity prediction; ordinal regression; feature selection; classification

0 引言

随着软件规模和复杂度的日益提高，软件质量保障变得越来越重要，软件缺陷预测[1]是提高软件质量保障的一种方法，已成为软件仓库挖掘领域的一个重要研究课题。软件缺陷预测根据软件模块构建软件缺陷预测模型，通过预测模型可识别项目内潜在的缺陷模块，此时软件开发人员可对模型预测得到的缺陷模块进行相应的代码审查或软件测试。软件缺陷预测模型一方面可以减轻人工审查大量无缺陷模块的工作，又可以协助软件质量保障部门合理地分配有限的测试资源，是提高软件质量保障的一种有效手段。

现在一般使用缺陷跟踪系统来管理大规模软件项目并保障项目的开发质量。Bugzilla 和 JIRA 是目前主流的开源缺陷跟踪系统，其中 Bugzilla 系统会将一个缺陷模块的严重程度从高到低划分为 Blocker、Critical、Major、Normal、Minor、Trivial 和 Enhancement 七个级别，JIRA 系统会将一个缺陷模块的严重程度从高到低划分为 Blocker、Critical、Major、Minor 和 Trivial 五个级别，目前大部分研究人员[1]根据上述的严重程度将软件模块划分为有缺陷模块和无缺陷模块两种情况，此时软件缺陷预测问题建模为二分类问题。为了更合理的分配软件项目的测试资源，即优先分配测试资源给含有高危缺陷的程度模块。本文根据缺陷跟踪系统对缺陷模块严重程度的划分等级，将模块划分为三类：高严重程度模块(高危模块)，低严重程度模块(低危模块)和无缺陷模块。当软件项目的模块被划分为三类时，本文可发现模块间的分类具有一定的次序性，同时不同类型的误分类错误代价也不完全相同。例如将高危模块误分类为无缺陷模块的代价要显著高于将低危模块误分类为无缺陷模块的代价。

有序回归(ordinal regression)[2]是一种分类方法，该方法与传统分类方法的区别在于有序回归可预测带有次序性的标签。例如：当学生对老师进行综合质量评价时，学生满意度通常包括{差，一般，好，非常好，特别好}这五种等级，可看出标签“一般”比“差”要好，而“特别好”又比“一般”和“差”都要好，这是有序回归标签之间的次序性，不难看出，在缺陷严重程度预测问题中，模块的分类标签也具有这种次序性。在传统分类方法中，只会将不同的标签进行简单的数值转换，并不考虑标签之间的次序性。有序回归方法可以通过设定每种标签之间的阈值来规范这种次序性，相比传

收稿日期：2020-07-22；**修回日期：**2020-09-28 **基金项目：**国家自然科学基金资助项目(61702041)；南京大学计算机软件新技术国家重点实验室开放课题(KFKT2019B14)

作者简介：贾焱鑫(1996-)，女，山西文水人，硕士研究生，主要研究方向为软件质量保障；陈翔(1980-)，男(通信作者)，江苏南通人，副教授，硕导，博士，主要研究方向为软件质量保障(xchencs@ntu.edu.cn)；葛骅(1998-)，男，江苏常州人，本科生，主要研究方向为软件质量保障；杨光(1997-)，男，江苏海安人，硕士研究生，主要研究方向为软件质量保障；林浩(2000-)，男，江苏南京人，本科生，主要研究方向为软件质量保障.

统分类方法，本文首次将缺陷严重程度预测问题建模为有序回归问题，旨在进一步提升软件缺陷预测模型结果的准确性。

随后论文提出了一种基于有序回归的软件缺陷严重程度预测方法 ORESP，该方法首先使用基于 Spearman 的特征选择方法来尝试移除数据集内的冗余特征，然后利用基于比例优势模型(proportional odds model)的神经网络[2]对软件项目中模块的严重程度进行预测，该模型将比例优势模型[3]引入到神经网络的输出层，一方面可以保留神经网络模型多分类的优势，另一方面因为比例优势模型中累计和的思想，可通过阈值的限定考虑标签间的次序性。最后使用平均 0-1 错误(MZE)和平均绝对误差(MAE)对模型预测性能进行评估。该方法首先考虑了冗余特征在预测过程中的负面影响，又考虑了软件模块标签间的次序性。

论文基于代码复杂度和社交网络分析两个角度，考虑四种不同类型的模块度量元，同时选取了五种经典的多分类方法，包括逻辑回归(LR)、决策树(DT)、随机森林(RF)、K 最近邻(KNN)、梯度增强树(GB)进行对比。这些方法在构建模型时，均未考虑到模块标签间的次序性。基于四个大规模实际开源项目(累计 23 个版本)的实验结果表明：基于 MZE 和 MAE 的评测指标，论文所提的 ORESP 方法在性能上要优于五种机器学习领域中的经典多分类方法，其中基于平均 0-1 误差(MZE)，预测模型性能最大可提升 10.3%；基于平均绝对误差(MAE)，预测模型性能最大可提升 12.3%。除此之外，使用基于 Spearman 的特征选择方法有助于提升 ORESP 方法的预测性能。

1　研究背景和相关工作

1.1　相关工作

大多数研究工作将软件缺陷预测问题建模为二分类问题(即有缺陷模块和无缺陷模块)[1]。依据软件缺陷严重程度将软件缺陷预测问题建模为多分类问题研究工作有 You 等人[4]使用多逻辑回归与随机梯度下降的方法来预测软件缺陷模块的严重程度。Nguyen 等人[5]根据软件缺陷的数目对软件模块进行排序，以此确定软件模块的严重程度。Yang 等人[6]引入学习排序的方法进行软件缺陷严重程度的预测，并使用不同的指标进行了优化。Yadav 等人[7]使用模糊逻辑提取规则来预测软件缺陷的严重程度，而最终软件模块的严重程度依赖于模块中缺陷的数目。与上述研究工作不同，据本文所知，论文首次将软件缺陷的严重程度预测建模为有序回归问题并提出相应的解决方案。

1.2　程序模块的度量类型

在软件缺陷预测中，一般先从软件历史仓库中抽取程序模块，其程序模块粒度根据实际需要可以细分为文件、包、类或函数等，随后通过分析程序模块的复杂度和软件开发过程来设计度量元(metrics)，完成对抽取出的程序模块的度量。

论文主要考虑文献[8]中使用的四类度量元。其中：基于代码的度量元(CMs)重点关注模块代码规模和内在复杂度。后三类度量元均基于社交网络分析，具体来说，基于局部网络的度量元(ENs)重点关注模块直接调用或被调用情况。基于全局网络的度量元(GNs)重点关注模块所有调用或被调用情况。基于网络的度量元(NMs)综合考虑了基于局部网络的度量元和(ENs)基于全局网络的度量元(GNs)。论文中，本文考虑的 CMs 包含 23 个度量元，ENs 包含 36 个度量元，GNs 包含 17 个度量元，NMs 则包含 53(=36+17)个度量元。

1.3　模块类型的标注

目前大多数软件项目使用缺陷跟踪系统(例如Bugzilla和JIRA 等)来进行软件缺陷的维护与管理。其中，Bugzilla 将缺陷模块的严重程度从高到低划分为 Blocker、Critical、Major、Normal、Minor、Trivial 和 Enhancement 七个级别；而 JIRA 则将缺陷模块的严重程度从高到低划分为 Blocker、Critical、Major、Minor 和 Trivial 五个级别。

由于上述两个缺陷跟踪系统对缺陷模块严重程度的划分不同，故本文人为对其进行分类标注：高危模块，低危模块，无缺陷模块。针对 Bugzilla 系统，本文将严重程度为 Blocker、Critical 或 Major 的缺陷模块视为高危模块，严重程度为 Enhancement 和未出现在缺陷跟踪系统中的模块视为无缺陷模块，其余严重程度的模块均视为低危模块；针对 JIRA 系统，本文将严重程度为 Blocker、Critical 的缺陷视为高危模块，其余均视为低危模块，未出现在缺陷跟踪系统中的模块视为无缺陷模块。

2　论文所提的 ORESP 方法

论文采用有序回归方法[2]对软件缺陷严重程度进行预测。有序回归问题是由输入特征向量 X 及标记 y 构成，其中 $X_i \in \mathbb{R}^K$，$X_i = \{x_1, x_2, ..., x_k\}$ 是一个 k 维向量，具体来说 k 指的是不同类型度量元的个数；$y \in N^* = \{C_1, C_2, ..., C_Q\}$ 是正整数，其中 Q 表示标签的数目，即模块类别数目，因此具有 N 个样本的数据集可表示为 $D = \{(X_i, y_i), i = 1, 2, ..., N\}$。

该方法首先使用基于 Spearman 的特征选择方法对数据集进行特征选择，得到满足 Spearman 相关系数和方差膨胀因子(Variance Inflation Factor VIF)[9]条件的特征，然后使用基于比例优势模型的神经网络对软件模块进行不同严重程度的预测，构建有序回归模型，最后利用 MZE 和 MAE 指标来评估模型的预测性能。方法的整体框架如图 1 所示。

如在跨版本场景(见 3.2 节)下，选择当前版本(如 Apache Ant 1.5 版本)基于代码的度量元(CMs)作为测试集，那么前一版本(Apache Ant 1.4 版本)基于代码的度量元(CMs)将被作为训练集。首先训练集经过基于 Spearman 的特征选择，得到满足条件的 CMs 特征及数据，并输入基于比例优势模型的神经网络，构建相应的缺陷严重程度预测模型。然后，从测试集中挑选训练集所选特征的数据输入所构建的模型，实现对测试集模型类型的预测，其中模块类型指高严重程度模块(高危模块)，低严重程度模块(低危模块)和无缺陷模块。

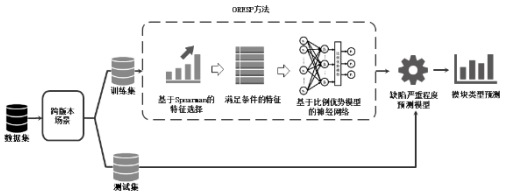


图 1　软件缺陷严重程度预测方法 ORESP 的整体框架

Fig. 1　Framework of software defect severity method ORESP

2.1　基于 Spearman 的特征选择方法

软件缺陷预测模型的性能很大程度上依赖于用于度量程序模块的度量元。已有研究工作[10]发现软件缺陷预测数据集中的一些度量元间存在相关性，同时冗余度量元在软件缺陷预测模型性能上具有负面作用。

为解决上述问题，ORESP 方法采用基于 Spearman 的特征选择方法来尝试识别并移除冗余度量元。其中 Spearman 相关系数是衡量两个度量元依赖性的非参数指标，当系数为-1 或+1，表明两个度量元单调相关；方差膨胀因子(Variance Inflation Factor VIF)用于衡量某个度量元(即自变量)与其他多个度量元(即其他变量)间的线性相关关系，VIF 取值越大，则这些度量元间存在更明显的共线性问题。

论文使用的基于 Spearman 的特征选择方法的伪代码如算法 1 所示。

算法 1　基于 Spearman 的特征选择方法

输入：*M* is a set of metrics,*sp.t* is a threshold for Spearman correlation,*vif.t* is a threshold for VIF.
输出：*M'* is a set of non-correlated metrics.
a) M' = M.
b) S = {(m_i,m_j)|abs(Spearman(*M*,*M*))>*sp.t*}.
c) for s(m_i,m_j) in *S* do;
avg.m_i = mean (s(m_i, others));
avg.m_j = mean (s(m_j, others));
remove.m = s (max(avg.m_i,avg.m_j), others);
M' = *M'* - remove.m;
end.
d) repeat;
V = VIF(*M'*);
v = {(m_k)|*V*(m_k)>*vif.t*};
M' = *M'* – v;
until v = ∅.
e) return *M'*.

算法 1 首先计算度量元的 Spearman 相关系数矩阵，并选择大于预设阈值的度量元对；随后分别计算度量元与其余度量元的 Spearman 相关系数均值，选择均值较大的度量元，并移除该度量元及与其相关系数较大的度量元；重复上一步，直到 Spearman 相关系数中不存在大于等于预设阈值的度量元；最后去除集合中方差膨胀因子大于预设阈值的度量元，完成特征选择。算法 1 中 *S* 和 *V* 分别为满足条件的相关系数集合与方差膨胀因子的集合，others 表示除该变量之外的其余变量。论文根据实际性能将阈值设置[11]如下：*sp.t*=0.7,*vif.t*=5。

2.2　基于比例优势模型的神经网络

传统的神经网络模型可以对实例进行多分类预测，但是忽略了输出标签之间的次序性。而基于比例优势模型的神经网络[2]将比例优势模型[3]加至神经网络的输出层，一方面可以保留神经网络模型多分类的优势，另一方面因为比例优势模型中累计和的思想，可通过阈值的限定考虑标签间的次序性。

该方法是一个单隐层的神经网络模型，其输入层和隐藏层与传统神经网络模型相同，不同之处在于：输出层中添加阈值 T_i 进行限制并求出小于等于某个类别的概率之和，即 $\sum i=P(y\leqslant i)$，其中 $P(y\leqslant i)=P(y=1)+P(y=2)+\ldots+P(y=i)$，然后利用比例优势模型的思想计算属于每个类别的概率，即 $P(y=i)=\sum i-\sum(i-1)$，图 2 为基于比例优势模型的神经网络结构图，其中 X_i 表示输入第 i 个特征；β_i 表示隐藏层第 i 个神经元的输入；$\sum i$ 表示属于 i 类之前所有类的概率；$\sum$表示所有概率的总和(即为 1)；P_i 表示属于第 i 类的概率。

在基于比例优势模型的神经网络结构中，输入层为 k 个度量元；隐藏层和输出层均使用 sigmoid 激活函数；损失函数采用交叉熵损失函数，为防止过拟合，增加 L2 正则化；使用 *iRprop*+[12]算法来弹性改变神经网络学习率以更新神经网络权重。对于超参取值，论文采用网格搜索的方法来确定超参最优取值，论文考虑的超参包括：隐藏层神经元节点数，*iRprop*+算法的迭代次数和 L2 正则化系数。

3　实证研究

本节对论文所提的 ORESP 方法的有效性展开实证研究，下面是实证研究中需要回答的两个实验问题。

RQ1　论文所提的 ORESP 方法的预测性能是否要优于传统的多分类预测方法？

RQ2　使用基于 Spearman 的特征选择方法是否能够提升 ORESP 方法的性能？

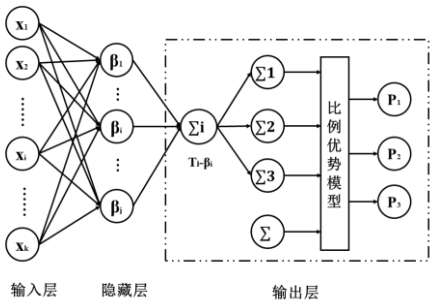


图 2　基于比例优势模型的神经网络结构

Fig. 2　Structure diagram of neural network based on proportional odds model

3.1　评测数据集

论文使用了四个经典的开源软件项目：Apache Ant、Eclipse、Firefox 和 Apache HBase，其中 Apache Ant 是可以支持自动化软件开发过程的工具；Eclipse 是一个跨平台的集成开发环境；Firefox 是开源的网页浏览器；Apache HBase 是一个分布式数据库。这些开源软件在所处领域、项目规模大小和编程语言等方面各不相同，因此可以确保实验结论具有一定的代表性。其具体的版本号、模块数和构成、以及缺陷模块所占比例如表 1 所示。

表 1　四个软件项目数据集的统计特征

Tab. 1　Statistical characteristics of four software project data sets

项目名称	项目版本	模块数	低危模块数	高危模块数	缺陷模块比例
Apache Ant	1.4	314	83	41	0.28
	1.5	338	272	102	0.53
	1.6	403	440	80	0.56
Eclipse	1.7	865	172	36	0.19
	2.0	4947	902	165	0.18
	2.1	6689	892	197	0.14
	3.0	7610	1684	463	0.22
	3.1	8912	2055	321	0.21
	3.2	10304	2329	303	0.20
Firefox	4.0	9504	809	232	0.10
	6.0	9631	451	67	0.05
	8.0	9510	426	76	0.05
	10.0	9840	1053	63	0.10
	12.0	10450	660	77	0.07
	14.0	10526	1123	118	0.11
Apache HBase	0.20	291	131	91	0.43
	0.89	311	89	88	0.36
	0.90	72	31	481	0.88
	0.92	522	146	108	0.33
	0.94	465	304	160	0.50
	0.95	792	465	216	0.46
	0.96	1395	260	45	0.18
	0.98	1414	269	208	0.25

3.2　实验场景

在实际的软件开发项目中，一般使用历史版本的数据来预测下一版本。因此论文基于跨版本场景来评估模型的预测性能。即，使用前一版本的数据集来训练模型，并用当前版本的数据集对所构建的模型进行评估。例如：基于 Apache Ant 1.4 版本的数据集来构建训练模型，随后使用 Apache Ant 1.5 版本的数据集对构建的模型进行性能评估。

3.3　评测指标

论文考虑了两种评估模型的评测指标[2]，即平均 0-1 误差(MZE)和平均绝对误差(MAE)。

MZE 指标可以衡量模型的错误比率，其变化范围为[0,1]，数值越小表明模型的预测性能越好，其与模型的全局表现相关，但并没有考虑标签之间的次序性，其计算公式为

$$MZE=\frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i\neq y_i\right) \tag{1}$$

其中：y_i 为真实标记，$\hat{y}_i$ 为模型的预测标记，N 为样本数目。

MAE 指标可以计算预测标签与真实标签绝对误差的平均值，其变化范围为[0,$Q-1$]，数值越小，则表明模型的预测性能越好。其计算公式如(2)所示。

$$MAE=\frac{1}{N}\sum_{i=1}^{N}\left|r(y_i)-r\left(\hat{y}_i\right)\right| \tag{2}$$

其中:$r(\cdot)$表示排序的次序，Q表示等级数。

3.4　结果分析与讨论

论文基于 R 语言和 Matlab 软件实现了 ORESP 方法，为了方便其他研究人员重现本文的研究工作，本文将代码进行了共享(https://github.com/JiaYanx/ORESP)。

1)针对 RQ1 的结果分析

在该研究工作中，本文首次将软件缺陷严重程度预测问题建模为有序回归问题，所以近两年暂无相关改进方法可以进行比较。为了突出论文所提方法 ORESP 的优越性，本文首先选择了五种机器学习领域中的经典多分类方法作为基准方法，分别为逻辑回归(LR)、决策树(DT)、随机森林(RF)、K 最近邻(KNN)、梯度增强树(GB)。这些分类方法基于 scikit-learn 框架实现，参数取值均使用默认取值。

为了比较更公平，本文在基准方法上使用了基于 Spearman 的特征选择方法。在实验时，分别考虑不同类型的度量元(即 CMs、ENs、GNs、NMs)，关于不同类型的度量元，已在本文 1.2 节中详细阐述。步骤为：先使用基于 Spearman 的特征选择方法对数据集中度量元进行选择，以移除冗余度量元，然后使用前一版本的数据集构建软件缺陷预测模型，最后利用当前版本的数据集对构建的模型进行评估。

论文所提的 ORESP 方法与基准方法的比较结果如表 2(基于 MZE 指标)和表 3(基于 MAE 指标)所示，其中对不同类型度量下最好的结果进行了加粗。

基于 MZE 指标的结果如表 2 所示，在基准方法中，逻辑回归(LR)方法的预测性能较好，但论文所提的 ORESP 与逻辑回归方法相比，其性能可以进一步提升，MZE 值分别提升了 0.009，0.057，0.005 和 0.003；在不同类型度量下，ORESP 最大提升性能分别为 0.038，0.103，0.092，0.070。

表 2　基于 MZE 评测指标的性能比较结果

Tab. 2　Performance comparison results in terms of MZE

方法	CMs	ENs	GNs	NMs
LR	0.321	0.310	0.316	0.312
DT	0.350	0.356	0.403	0.379
RF	0.337	0.322	0.325	0.327
KNN	0.335	0.321	0.338	0.329
GB	0.323	0.311	0.325	0.327
ORESP	**0.312**	**0.253**	**0.311**	**0.309**

基于 MAE 指标的结果如表 3 所示，在基准方法中，基于 CMs 和 ENs 度量下，梯度增强树(GB)方法的 MAE 值最小，基于 GNs 和 NMs 度量元，逻辑回归(LR)方法的 MAE 值最小，分别是 0.425，0.415，0.437 和 0.430，但论文所提的 ORESP 方法与基准方法相比，其性能可以得到进一步提升，其 MAE 值分别为 0.392，0.383，0.401 和 0.402；不同类型度量下，ORESP 最大提升性能分别为 0.064，0.083，0.123 和 0.093。

综上所述，论文所提的 ORESP 方法的预测性能要优于五种经典的基准方法。

表 3 基于 MAE 评测指标的性能比较结果

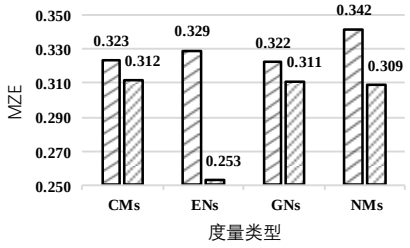
Tab. 3　Performance comparison results in terms of MAE

方法	CMs	ENs	GNs	NMs
LR	0.433	0.428	0.437	0.430
DT	0.456	0.466	0.524	0.495
RF	0.440	0.429	0.443	0.445
KNN	0.442	0.429	0.451	0.438
GB	0.425	0.415	0.440	0.445
ORESP	**0.392**	**0.383**	**0.401**	**0.402**

2)针对 RQ2 的结果分析

针对 RQ2，论文重点分析 ORESP 方法内的特征选择阶段对最终预测结果是否存在影响？即基于 Spearman 的特征选择方法是否能够提升 ORESP 方法的性能。

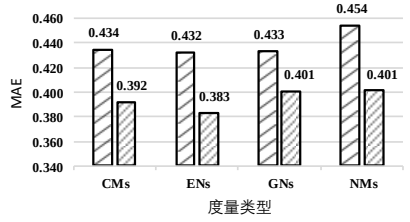
论文在相同的实验设置下，将使用了基于 Spearman 的特征选择方法的 ORESP 方法与未进行特征选择的 ORESP 方法进行了对比，具体结果如图 3(基于 MZE 指标)和图 4(基于 MAZ 指标)所示。



□未进行特征选择　□特征选择

图 3　基于 MZE 指标的比较结果

Fig. 3　Comparison Results in terms of MZE



□未进行特征选择　□特征选择

图 4 基于 MAE 指标的比较结果

Fig. 4　Comparison Results in terms of MAE

若基于 MZE 评测指标，在 CMs、ENs、GNs 和 NMs 这四种不同类型的度量下，使用特征选择的 ORESP 方法的性能可以分别提升 0.012，0.076，0.011 和 0.033，其中在 ENs 度量元上提升幅度最大；若基于 MAE 评测指标，在 CMs、ENs、GNs 和 NMs 这四种不同类型的度量下，使用特征选择的 ORESP 方法的性能可以分别提升 0.042，0.049，0.032 和 0.053。综上所述，使用基于 Spearman 的特征选择方法可以提升 ORESP 方法的预测性能。

4　结束语

软件缺陷预测[1,13-15]是当前软件仓库挖掘领域内的一个研究热点，通过提前预测出可疑缺陷模块，可以优化预测资源分配从而提升被测软件质量。论文通过考虑标签之间的次序性，首次将软件缺陷严重程度预测问题建模为有序回归问题，并提出 ORESP 方法，基于 4 种不同类型的度量元，对该方法的有效性进行了验证。除此之外，还深入分析了 ORESP 方法中基于 Spearman 的特征选择方法对模型预测能力的影响。

该方法仍然存在一些后续研究工作，具体来说：a)本文将尝试搜集更多的数据集，对论文结论的一般性进行验证。b)尝试考虑更为有效的特征选择方法，通过移除数据集中的

冗余特征和无关特征，来进一步提升 ORESP 方法的性能。

参考文献：

[1] 陈翔, 顾庆, 刘望舒, 等. 静态软件缺陷预测方法研究. 软件学报, 2016, 27 (1): 1-25. (Chen X, Gu Q, Liu WS, *et al.* Research on static software defect prediction methods. Journal of Software, 2016, 27 (1): 1-25.)

[2] Gutierrez P A, Perezortiz M, Sanchezmonedero J, *et al.* Ordinal regression methods: survey and experimental study [J]. IEEE Transactions on Knowledge and Data Engineering, 2016, 28 (1): 127-146.

[3] McCullagh, P. : Regression models for ordinal data (with discussion) . Journal of the Royal Statistical Society 42 (2) , 109–142 (1980) .

[4] You Guoan, Wang Feng, Ma Yutao. An empirical study of ranking-oriented cross-project software defect prediction [J]. International Journal of Software Engineering and Knowledge Engineering, 2016: 1511-1538.

[5] Nguyen TT, An TQ, Hai VT, *et al.* Similarity-based and rank-based defect prediction. In: Proc. of the Advanced Technologies for Communications (ATC 2014) . Hanoi, 2014. 321–325.

[6] Yang XX, Tang K, Yao X. A Learning-to-rank approach to software defect prediction [J]. IEEE Transactions on Reliability, 2015, 64 (1): 234-246.

[7] Yadav HB, Yadav DK. A fuzzy logic based approach for phase-wise software defects prediction using software metrics [J]. Information & Software Technology, 2015, 63 (63): 44-57.

[8] Chen L, Ma WWY, Zhou YM, *et al.* Empirical analysis of network measures for predicting high severity software faults [J]. Science in China Series F: Information Sciences, 2016, 59 (12) .

[9] Jiarpakdee J, Tantithamthavorn C, Hassan A E, *et al.* The Impact of correlated metrics on defect models. [J]. arXiv: Software Engineering, 2018.

[10] Ni C, Chen X, Wu FF, *et al.* An empirical study on pareto based multi-objective feature selection for software defect prediction. Journal of Systems and Software. 2019, 152: 215-238.

[11] Jiarpakdee J, Tantithamthavorn C, Treude C, *et al.* AutoSpearman: automatically mitigating correlated software metrics for interpreting defect models [C]. International Conference on Software Maintenance, 2018: 92-103.

[12] Duan XD, Wang YG, Pedrycz W, *et al.* AFSNN: A Classification Algorithm Using Axiomatic Fuzzy Sets and Neural Networks [J]. IEEE Transactions on Fuzzy Systems, 2018.

[13] Yuan ZD, Chen X, Cui ZQ, *et al.* ALTRA: Cross-project Software Defect Prediction via Active Learning and TrAdaBoost. IEEE Access. 2020, 8: 30037-30049.

[14] Chen DY, Chen X, Li H, *et al.* DeepCPDP: Deep Learning based Cross-Project Defect Prediction. IEEE ACCESS. 2019, 7: 184832-184848.

[15] 陈翔, 赵英全, 顾庆, 等. 基于文件粒度的多目标软件缺陷预测方法实证研究. 软件学报, 2019, 30 (12): 3694-3713. (Chen X, Zhao YQ, Gu Q, *et al.* An empirical study on multi-objective software defect prediction method based on document granularity. Journal of Software, 2019, 30 (12): 3694-3713.)