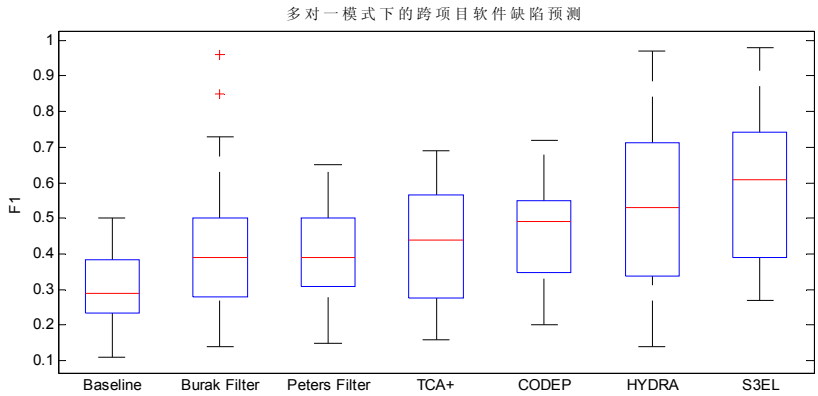

Fig.4 Box plot of many-to-one cross-project defect prediction

图 4 多对一跨项目缺陷预测结果箱线图

基于上述两组数据集的实验结果我们可以看出:与经典的跨项目缺陷预测方法相比,S^{3}EL 方法在多对一模式及一对一模式的预测场景中均能取得出平均最好的预测性能,因此初步验证了通过均值划分数据集的贝叶斯集成学习方法,能够显著提高朴素贝叶斯的预测能力.

3.4.2 针对 RQ2 的分析

为了验证 S^{3}EL 方法和同项目缺陷预测方法相比的性能优劣,本节在 Promise 数据集上分别用 5%和 90%的训练数据比例进行模型构建,并在剩余 95%和 10%的实例上进行缺陷预测.同项目预测模型同样使用朴素贝叶斯分类方法,随机抽样生成训练集,使用 *F*1 指标对预测性能进行评价.为了减小随机抽取训练集对结果的影响,方法均执行 3 次取平均 *F*1 值.表 7 给出了最终的实验结果.

Table 7 Compared result of *F*1 value between our approach and other within-project defect prediction

表 7 方法与同项目缺陷预测方法的 *F*1 值对比

目标项目	S^{3}EL	5%	90%
ant-1.3	0.39	0.21	0.33
ant-1.4	0.36	0.25	0.17
ant-1.5	0.41	0.34	0.5
ant-1.6	0.61	0.52	0.59
ant-1.7	0.52	0.39	0.65
log4j-1.0	0.51	0.32	0.67
log4j-1.1	0.65	0.49	0.86
log4j-1.2	0.94	0.96	0.64
lucene-2.0	0.67	0.49	0.67
lucene-2.2	0.74	0.64	0.71
lucene-2.4	0.74	0.73	0.45
poi-1.5	0.75	0.67	0.38
poi-2.0	0.31	0.16	0.3
poi-2.5	0.79	0.78	0.74
poi-3.0	0.81	0.81	0.42
redaktor	0.31	0.17	0.35
synapse-1.0	0.39	0.11	0.36
synapse-1.1	0.47	0.12	0.57
synapse-1.2	0.61	0.09	0.62
Tomcat	0.37	0.25	0.4
velocity-1.4	0.76	0.87	0.87
velocity-1.6	0.51	0.39	0.4
xalan-2.4	0.39	0.32	0.24
xalan-2.5	0.62	0.41	0.21
xalan-2.6	0.63	0.66	0.63
xalan-2.7	0.98	0.99	0.99
xerces-1.2	0.27	0.18	0.67
xerces-1.3	0.41	0.37	0.45
xerces-1.4	0.87	0.92	0.89
均值	0.58	0.47	0.55

结果表明:S^{3}EL 方法和 5%训练集的同项目缺陷预测相比,在指标均值上提升了 23%;和 90%训练集同项目缺陷预测相比,平均 *F*1 值相差不大,甚至略优,而半监督学习方法中仅用到 5%的样本做模型训练.可见,我们的方法跟与项目缺陷预测方法相比更具有应用价值.

3.4.3 实验结果的进一步分析

为了验证实验结论的可靠性,我们借助统计分析对实验结果进行方差检验.检验的假设是对比方法之间的平均预测性能 *F*1 没有显著差异,设置显著性水平为 0.05.我们分析一对一及多对一跨项目缺陷预测的 *F*1 指标,使用一种非参数的统计检验方法 Friedman 检验[47]来检验假设的可靠性.表 8 和表 9 分别显示了一对一模式及多对一模式下的缺陷预测实验的 Friedman 检验结果.*F* 值由组内平方和及组间平方和计算得到,*F* 值越大,表示拒绝原假设的可信度越高.*p* 值是 *F* 值对应的显著性水平,表 8 和表 9 中的结果 *p* 值都远小于显著性水平 0.05,所以原假设应该被拒绝,即一对一模式及多对一模式下的两组对比实验的结果均有显著差异.

Table 8 Friedman test for one-to-one cross-project defect prediction

表 8 一对一模式下,实验结果的 Friedman 检验

	平方和	自由度	均方	*F* 值	*p* 值
组间误差	69.4	5	13.88	20.72	0.000 9
组内误差	265.6	95	2.795 8		
总和	335	119			

Table 9 Friedman test for many-to-one cross-project defect prediction

表 9 多对一模式下,实验结果的 Friedman 检验

	平方和	自由度	均方	*F* 值	*p* 值
组间误差	257.966	6	42.994 3	55.59	0.000 0
组内误差	549.534	168	3.271		
总和	807.5	202			

Friedman 检验只能发现方法间的结果有显著差异,但是不能发现任意两种方法的具体差异性.因此,我们使用最小显著差异法 LSD(least-significant difference)来比较两种方法之间的差异.表 10 和表 11 显示了使用 LSD 方法[48]分别对一对一模式和多对一模式下的缺陷预测实验进行检验的结果.如果置信度下限和置信度上限之间没有包含 0(即同为正或同为负),则认为两种方法之间的差异是显著的.

Table 10 LSD test for one-to-one cross-project defect prediction

表 10 一对一模式下,实验结果的 LSD 检验

方法 *x*	方法 *y*	置信下限	平均误差(*y*–*x*)	置信上限
S^{3}EL	**基准方法**	**0.550 613 739**	**2.2**	**3.849 386 261**
S^{3}EL	Burak filter	–0.124 386 26	1.525	3.174 386 261
S^{3}EL	**Peters filter**	**0.025 613 739**	**1.675**	**3.324 386 261**
S^{3}EL	TCA+	–0.999 386 26	0.65	2.299 386 261
S^{3}EL	**CODEP**	**0.250 613 739**	**1.9**	**3.549 386 261**
基准方法	Burak filter	–2.324 386 26	–0.675	0.974 386 261
基准方法	Peters filter	–2.174 386 26	–0.525	1.124 386 261
基准方法	TCA+	–3.199 386 26	–1.55	0.099 386 261
基准方法	CODEP	–1.949 386 26	–0.3	1.349 386 261
Burak filter	Peters filter	–1.499 386 26	0.15	1.799 386 261
Burak filter	TCA+	–2.524 386 26	–0.875	0.774 386 261
Burak filter	CODEP	–1.274 386 26	0.375	2.024 386 261
Peters filter	TCA+	–2.674 386 26	–1.025	0.624 386 261
Peters filter	CODEP	–1.424 386 26	0.225	1.874 386 261
TCA+	CODEP	–0.399 386 26	1.25	2.899 386 261

Table 11 LSD test for many-to-one cross-project defect prediction

表 11 多对一模式下,实验结果的 LSD 检验

方法 x	方法 y	置信下限	平均误差(y–x)	置信上限
S^3EL	**基准方法**	**2.004 447 527**	**3.672 413 793**	**5.340 380 059**
S^3EL	**Burak filter**	**0.866 516 493**	**2.534 482 759**	**4.202 449 025**
S^3EL	**Peters filter**	**0.521 688 906**	**2.189 655 172**	**3.857 621 438**
S^3EL	**TCA+**	**1.263 068 217**	**2.931 034 483**	**4.599 000 749**
S^3EL	**CODEP**	**0.194 102 7**	**1.862 068 966**	**3.530 035 231**
S^3EL	HYDRA	–0.616 242 13	1.051 724 138	2.719 690 404
基准方法	Burak filter	–2.805 897 3	–1.137 931 03	0.530 035 231
基准方法	Peters filter	–3.150 724 89	–1.482 758 62	0.185 207 645
基准方法	TCA+	–2.409 345 58	–0.741 379 31	0.926 586 956
基准方法	**CODEP**	**–3.478 311 09**	**–1.810 344 83**	**–0.142 378 56**
基准方法	**HYDRA**	**–4.288 655 92**	**–2.620 689 66**	**–0.952 723 39**
Burak filter	Peters filter	–2.012 793 85	–0.344 827 59	1.323 138 68
Burak filter	TCA+	–1.271 414 54	0.396 551 724	2.064 517 99
Burak filter	CODEP	–2.340 380 06	–0.672 413 79	0.995 552 473
Burak filter	HYDRA	–3.150 724 89	–1.482 758 62	0.185 207 645
Peters filter	TCA+	–0.926 586 96	0.741 379 31	2.409 345 576
Peters filter	CODEP	–1.995 552 47	–0.327 586 21	1.340 380 059
Peters filter	HYDRA	–2.805 897 3	–1.137 931 03	0.530 035 231
TCA+	CODEP	–2.736 931 78	–1.068 965 52	0.599 000 749
TCA+	**HYDRA**	**–3.547 276 61**	**–1.879 310 34**	**–0.211 344 08**
CODEP	HYDRA	–2.478 311 09	–0.810 344 83	0.857 621 438

根据显著性检验结果可以看出:在一对一模式下的跨项目缺陷预测中,我们的方法显著优于基准方法、Peters 过滤法及 CODEP 方法,略优于 Burak 过滤法及 TCA+方法,而其他几种方法之间差异并不显著.在多对一模式下的跨项目缺陷预测中,我们的方法显著优于基准方法、Burak 过滤法、Peters 过滤法、TCA+、CODEP 方法,略优于 HYDRA 方法.HYDRA 方法显著优于基准方法及 TCA+方法.CODEP 显著优于基准方法.其余方法之间的差异并不显著.

3.5 有效性影响因素分析

影响实证研究内部有效性的因素主要有:(1) S^3EL 方法实现及对比经典方法的重现过程中存在的隐藏缺陷;(2) 遗传算法中存在的随机因素对结果可能会产生一定的影响;(3) 不同的验证集分布可能导致结果差异较大.针对以上问题,本文在这些方法的实现过程中经过反复测试,以避免编码过程中可能产生的缺陷.将重现的经典方法的执行结果与之前文献中得到的研究结果进行比较,并确保基本一致.除此之外,还通过独立执行多次,借助取均值的方式降低方法内的随机因素可能给结果带来的影响.

影响实证研究外部有效性因素主要有:(1) 实证研究得到的结论是否具有一般性;(2) 实验中用到的数据集是否存在一定的质量问题.针对以上问题,本文的跨项目缺陷预测实验使用了两个不同的公开数据集 AEEEM 及 Promise,每个数据集包含不同的开源项目及不同的缺陷分布,并且已被相关研究者们多次使用[2,42,49].此外,跨项目缺陷预测分别在不同的数据集上考虑了多对一模式和一对一模式,从而确保了实验结论的可信度.

影响结论有效性的主要因素是使用的评测指标是否合理.针对缺陷预测问题,缺陷类的关注度要高于非缺陷类,$F1$ 指标能够平衡缺陷类预测性能的查全率及查准率,是跨项目缺陷预测领域最常用的重要评价指标之一[2,49−50].最后,我们还借助 Friedman 检验和 LSD 方法对一对一模式及多对一模式下的跨项目缺陷预测结果进行了显著性检验,进一步验证了结论的有效性

4 总结和展望

本文提出了一种基于搜索的半监督集成跨项目缺陷预测方法.实验结果表明:S^3EL 方法在多个公开数据集上与多种典型的跨项目缺陷预测方法相比均能获得较好的预测性能,并且能够有效提高朴素贝叶斯的预测能力.同时,S^3EL 方法也能够很好地适应多对一模式及一对一模式下的跨项目缺陷预测问题,因此具有很好的应

用价值.

我们认为,该方法仍有一些后续工作值得扩展,具体来说:(1) 我们将尝试进行验证集的选取实验,通过特定方式选取最能代表目标项目数据分布的少量实例作为验证集,借助相关代码人员的标注结果,达到更优更稳定的预测效果;(2) 尝试以特定的比例对数据划分的结果进行抽样,以进一步缓解类不平衡问题;(3) 在 S^3EL 方法中进一步考虑特征选择方法[51,52],通过移除数据集中的冗余特征和无关特征来进一步提升跨项目缺陷预测的性能;(4) 考虑更多的数据集,对本文结论是否具有一般性进行验证.

References:

[1] Kim S, Whitehead EJ, Zhang Y. Classifying software changes: Clean or buggy? IEEE Trans. on Software Engineering, 2008,34(2): 181−196. [doi: 10.1109/TSE.2007.70773]

[2] Xia X, Lo D, Pan SJ, Nagappan N, Wang X. HYDRA: Massively compositional model for cross-project defect prediction. IEEE Trans. on Software Engineering, 2016,42(10):977−998. [doi: 10.1109/TSE.2016.2543218]

[3] Kim S, Zhang H, Wu R, Gong L. Dealing with noise in defect prediction. In: Proc. of the Int'l Conf. on Software Engineering. 2011. 481−490. [doi: 10.1145/1985793.1985859]

[4] Wang J, Shen B, Chen Y. Compressed C4.5 models for software defect prediction. In: Proc. of the Int'l Conf. on Quality Software. 2012. 13−16. [doi: 10.1109/QSIC.2012.19]

[5] Sun Z, Song Q, Zhu X. Using coding-based ensemble learning to improve software defect prediction. IEEE Trans. on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012,42(6):1806−1817. [doi: 10.1109/TSMCC.2012.2226152]

[6] Zhou MH, Guo CG. New thinking of software engineering based on big data. Communications of the CCF, 2014,10(3):37−42 (in Chinese).

[7] Canfora G, Lucia AD, Penta MD, Oliveto R, Panichella A, Panichella S. Multi-Objective cross-project defect prediction. In: Proc. of the Int'l Conf. on Software Testing, Verification and Validation. 2013. 252−261. [doi: 10.1109/ICST.2013.38]

[8] Briand LC, Melo WL, Wust J. Assessing the applicability of fault-proneness models across object-oriented software projects. IEEE Trans. on Software Engineering, 2002,28(7):706−720. [doi: 10.1109/TSE.2002.1019484]

[9] Cruz AEC, Ochimizu K. Towards logistic regression models for predicting fault-prone code across software projects. In: Proc. of the Int'l Symp. on Empirical Software Engineering and Measurement. 2009. 460−463. [doi: 10.1109/ESEM.2009.5316002]

[10] Nam J, Pan SJ, Kim S. Transfer defect learning. In: Proc. of the Int'l Conf. on Software Engineering. 2013. 382−391.

[11] Pan SJ, Yang Q. A survey on transfer learning. IEEE Trans. on Knowledge and Data Engineering, 2010,22(10):1345−1359. [doi: 10.1109/TKDE.2009.191]

[12] Zhuang FZ, Ping L, Qing HE, Shi ZZ. Survey on transfer learning research. Ruan Jian Xue Bao/Journal of Software, 2015,26(1): 26−39 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4631.htm [doi: 10.13328/j.cnki.jos.004631]

[13] Pelayo L, Dick S. Evaluating stratification alternatives to improve software defect prediction. IEEE Trans. on Reliability, 2012, 61(2):516−525. [doi: 10.1109/TR.2012.2183912]

[14] Chen X, Gu Q, Liu WS, Liu SL, Ni C. Software defect prediction. Ruan Jian Xue Bao/Journal of Software, 2016,27(1):1−25 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4923.htm [doi: 10.13328/j.cnki.jos.004923]

[15] Harman M, Mansouri SA, Zhang YY. Search-Based software engineering: Trends, techniques and applications. ACM Computing Surveys, 2012,45(1):1−61. [doi: 10.1145/2379776.2379787]

[16] Turhan B, Menzies T, Bener AB, Di Stefano J. On the relative value of cross-company and within-company data for defect prediction. Empirical Software Engineering, 2009,14(5):540−578. [doi: 10.1007/s10664-008-9103-7]

[17] Peters F, Menzies T, Marcus A. Better cross company defect prediction. In: Proc. of the IEEE Working Conf. on Mining Software Repositories. 2013. 409−418. [doi: 10.1109/MSR.2013.6624057]

[18] Panichella A, Oliveto R, Lucia AD. Cross-Project defect prediction models: L'Union fait la force. In: Proc. of the IEEE Conf. on Software Maintenance, Reengineering and Reverse Engineering. 2014. 164−173. [doi: 10.1109/CSMR-WCRE.2014.6747166]

[19] He Z, Shu F, Yang Y, Li M, Wang Q. An investigation on the feasibility of cross-project defect prediction. Automated Software Engineering, 2011,19(2):167−199. [doi: 10.1007/s10515-011-0090-3]

[20] Malhotra R, Raje R. An empirical comparison of machine learning techniques for software defect prediction. In: Proc. of the Int'l Conf. on Bioinspired Information and Communications Technologies. 2014. 320−327. [doi: 10.4108/icst.bict.2014.257871]

[21] Lessmann S, Baesens B, Mues C, Pietsch S. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. IEEE Trans. on Software Engineering, 2008,34(4):485−496. [doi: 10.1109/TSE.2008.35]

[22] Ghotra B, McIntosh S, Hassan AE. Revisiting the impact of classification techniques on the performance of defect prediction models. In: Proc. of the Int'l Conf. on Software Engineering. 2015. 789−800. [doi: 10.1109/ICSE.2015.91]

[23] Zhang Y, Lo D, Xia X, Sun J. An empirical study of classifier combination for cross-project defect prediction. In: Proc. of the IEEE Computer Software and Applications Conf. 2015. 264−269. [doi: 10.1109/COMPSAC.2015.58]

[24] Ryu D, Choi O, Baik J. Value-Cognitive boosting with a support vector machine for cross-project defect prediction. Empirical Software Engineering, 2014,21(1):43−71. [doi: 10.1007/s10664-014-9346-4]

[25] Ryu D, Jang J, Baik J. A hybrid instance selection using nearest-neighbor for cross-project defect prediction. Journal of Computer Science and Technology, 2015,30(5):969−980. [doi: 10.1007/s11390-015-1575-5]

[26] Turhan B, Misirli AT, Bener A. Empirical evaluation of the effects of mixed project data on learning defect predictors. Information and Software Technology, 2013,55(6):1101−1118. [doi: 10.1016/j.infsof.2012.10.003]

[27] Zhong S, Khoshgoftaar TM, Seliya N. Unsupervised learning for expert-based software quality estimation. In: Proc. of the IEEE Int'l Symp. on High Assurance Systems Engineering. 2004. 149−155. [doi: 10.1109/HASE.2004.1281739]

[28] Zhang F, Zheng Q, Zou Y, Hassan AE. Cross-Project defect prediction using a connectivity-based unsupervised classifier. In: Proc. of the Int'l Conf. on Software Engineering. 2016. 309−320. [doi: 10.1145/2884781.2884839]

[29] Nam J, Kim S. CLAMI: Defect prediction on unlabeled datasets. In: Proc. of the Int'l Conf. on Automated Software Engineering. 2015. 452−463. [doi: 10.1109/ASE.2015.56]

[30] Concas G, Marchesi M, Pinna S, Serra N. Power-Laws in a large object-oriented software system. IEEE Trans. on Software Engineering, 2007,33(10):687−708. [doi: 10.1109/TSE.2007.1019]

[31] Jiang Y, Cukic B, Menzies T. Can data transformation help in the detection of fault-prone modules? In: Proc. of the Workshop on Defects in Large Software Systems. 2008. 16−20. [doi: 10.1145/1390817.1390822]

[32] Menzies T, Greenwald J, Frank A. Data mining static code attributes to learn defect predictors. IEEE Trans. on Software Engineering, 2007,33(1):2−13. [doi: 10.1109/TSE.2007.256941]

[33] Song Q, Jia Z, Shepperd M, Ying S, Liu J. A general software defect-proneness prediction framework. IEEE Trans. on Software Engineering, 2011,37(3):356−370. [doi: 10.1109/TSE.2010.90]

[34] Zhang F, Mockus A, Keivanloo I, Zou Y. Towards building a universal defect prediction model. In: Proc. of the Working Conf. on Mining Software Repositories. 2014. 182−191. [doi: 10.1145/2597073.2597078]

[35] Rahman F, Devanbu P. How, and why, process metrics are better. In: Proc. of the Int'l Conf. on Software Engineering. 2013. 432−441. [doi: 10.1109/ICSE.2013.6606589]

[36] Bacchelli A, D'Ambros M, Lanza M. Are popular classes more defect prone? Lecture Notes in Computer Science, 2010,6013: 59−73. [doi: 10.1007/978-3-642-12029-9_5]

[37] Nagappan N, Ball T. Use of relative code churn measures to predict system defect density. In: Proc. of the Int'l Conf. on Software Engineering. 2005. 284−292. [doi: 10.1109/ICSE.2005.1553571]

[38] Hassan AE. Predicting faults using the complexity of code changes. In: Proc. of the Int'l Conf. on Software Engineering. 2009. 78−88. [doi: 10.1109/ICSE.2009.5070510]

[39] Moser R, Pedrycz W, Succi G. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In: Proc. of the Int'l Conf. on Software Engineering. 2008. 181−190. [doi: 10.1145/1368088.1368114]

[40] Herzig K, Just S, Rau A, Zeller A. Predicting defects using change genealogies. In: Proc. of the Int'l Symp. on Software Reliability Engineering. 2013. 118−127. [doi: 10.1109/ISSRE.2013.6698911]

[41] Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA data mining software: An update. ACM SIGKDD Explorations Newsletter, 2009,11(1):10−18. [doi: 10.1145/1656274.1656278]

[42] Ambros MD, Lanza M, Robbes R. An extensive comparison of bug prediction approaches. In: Proc. of the IEEE Working Conf. on Mining Software Repositories. 2010. 31−41. [doi: 10.1109/MSR.2010.5463279]

[43] Agarwal S. Data mining: Data mining concepts and techniques. In: Proc. of the Int'l Conf. on Machine Intelligence and Research Advancement. 2013. 203−207. [doi: 10.1109/ICMIRA.2013.45]

[44] Nguyen AT, Nguyen TT, Nguyen HA, Nguyen TN. Multi-Layered approach for recovering links between bug reports and fixes. In: Proc. of the ACM SIGSOFT Int'l Symp. on the Foundations of Software Engineering. 2012. 1−11. [doi: 10.1145/2393596.2393671]

[45] Tian Y, Lawall J, Lo D. Identifying Linux bug fixing patches. In: Proc. of the Int'l Conf. on Software Engineering. 2012. 386−396. [doi: 10.1109/ICSE.2012.6227176]

[46] Wu R, Zhang H, Kim S, Cheung SC. ReLink: Recovering links between bugs and changes. In: Proc. of the ACM SIGSOFT Symp. and the European Conf. on Foundations of Software Engineering. 2011. 15−25. [doi: 10.1145/2025113.2025120]

[47] Friedman M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association, 1937,32(200):675−701. [doi: 10.1080/01621459.1937.10503522]

[48] Wilcoxon F. Individual comparisons by ranking methods. Biometrics, 1945,1(6):80−83. [doi: 10.2307/3001968]

[49] Cao Q, Sun Q, Cao Q, Tan H. Software defect prediction via transfer learning based neural network. In: Proc. of the Int'l Conf. on Reliability Systems Engineering. 2015. 1−10. [doi: 10.1109/ICRSE.2015.7366475]

[50] Xu Z, Xuan J, Liu J, Cui X. MICHAC: Defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering. In: Proc. of the IEEE Int'l Conf. on Software Analysis, Evolution, and Reengineering. 2016. 370−381. [doi: 10.1109/SANER.2016.34]

[51] Liu SL, Chen X, Liu WS, Chen JQ, Gu Q, Chen DX. FECAR: A feature selection framework for software defect prediction. In: Proc. of the Annual Int'l Computers, Software and Applications Conf. 2014. 426−435. [doi: 10.1109/COMPSAC.2014.66]

[52] Liu WS, Liu SL, Gu Q, Chen JQ, Chen X, Chen DX. Empirical studies of a two-stage data preprocessing approach for software fault prediction. IEEE Trans. on Reliability, 2016,65(1):38−53. [doi: 10.1109/TR.2015.2461676]

附中文参考文献:

[6] 周明辉,郭长国.基于大数据的软件工程新思维.计算机学会通讯,2014,10(3):37−42.

[12] 庄福振,罗平,何清,等.迁移学习研究进展.软件学报,2015,26(1):26−39. http://www.jos.org.cn/1000-9825/4631.htm [doi: 10.13328/j.cnki.jos.004631]

[14] 陈翔,顾庆,刘望舒,刘树龙,倪超.静态软件缺陷预测方法研究.软件学报,2016,27(1):1−25. http://www.jos.org.cn/1000-9825/4923.htm [doi: 10.13328/j.cnki.jos.004923]

何吉元(1992－),女,江苏涟水人,硕士生,主要研究领域为软件测试,机器学习.

孟昭鹏(1962－),男,博士,教授,博士生导师,主要研究领域为机器学习,软件工程.

陈翔(1980－),男,博士,副教授,CCF 专业会员,主要研究领域为软件缺陷预测,软件缺陷定位,回归测试和组合测试.

王赞(1979－),男,博士,副教授,主要研究领域为软件工程,软件测试,机器学习,软件质量.

樊向宇(1992－),男,硕士生,CCF 学生会员,主要研究领域为软件测试,机器学习.