

# Supplementary Material for: How Far We Have Progressed in the Journey? An Examination of Cross-Project Defect Prediction

YUMING ZHOU, YIBIAO YANG, HONGMIN LU, LIN CHEN, YANHUI LI, and  
YANGYANG ZHAO, Nanjing University  
JUNYAN QIAN, Guilin University of Electronic Technology  
BAOWEN XU, Nanjing University

## A A COMPARISON OF FIVE SUPERVISED CPDP MODELS WITH MANUALDOWN

As mentioned in Section 3.3, of the 42 supervised CPDP studies, 4 studies [5, 26, 28, 47] reported only the median or mean prediction performance on the target projects, while one study [119] was applied to too few target projects. For each of these five studies, we are unable to apply the analysis procedure shown in Figure 3 to examine the statistical significance or the magnitude of the difference between the proposed supervised CPDP model and the simple module size model. Table 14 provides a performance comparison of these studies with the ManualDown(50%) model. As can be seen, overall, the ManualDown(50%) model achieves a comparable or even better prediction performance in most cases.

Table 14. The Comparison of Five CPDP Studies with the Size Model in the Classification Scenario

CPDP study	The CPDP model	N	Indicator	Model performance		Is CPDP model Better?
				CPDP	ManualDown	
[47]	LR-DBSCAN	21	Median F <sub>1</sub>	0.462	0.538	No
			Median G <sub>1</sub>	0.559	0.703	No
			Median AUC	0.662	0.801	No
[26]	DT-FILTER	34	Median F <sub>1</sub>	0.588	0.512	Yes
[5]	LR-DS	44	Median F <sub>1</sub>	0.503	0.512	No
			Median AUC	0.660	0.736	No
[28]	NB-riTDS-2(3)	34	Mean F <sub>1</sub>	0.475	0.490	No
			Mean G <sub>1</sub>	0.613	0.630	No
[119]	LR	4	Mean F <sub>1</sub>	0.4168	0.4173	No
			Median F <sub>1</sub>	0.438	0.373	Yes

## B THE 86 DATASETS USED IN SECTION 5

Table 15 describes the 86 datasets used in Section 5. The first and second columns, respectively, list the group name and project name. For each project, the third to the fifth columns, respectively, list the number of modules, the number of metrics, and the percentage of defective modules. The last column lists the size metric used to build the simple module size models used in Section 5.

Table 15. Details on 86 Datasets

Group	Project	#Modules	#Metrics	%Defective	Module size metric
AEEEM	JDT core	997	31	20.66%	ck_oo_numberOfLinesOfCode (Number of lines of code)
	Equinox	324	31	39.81%	
	Lucene	691	31	9.26%	
	Mylyn	1862	31	13.16%	
	PDE	1497	31	13.96%	
JURECZKO	ant 1.3	125	20	16.00%	loc (Number of lines of code)
	ant 1.4	178	20	22.47%	
	ant 1.5	293	20	10.92%	
	ant 1.6	351	20	26.21%	
	ant 1.7	745	20	22.28%	
	arc	234	20	11.54%	
	berek	43	20	37.21%	
	camel 1.0	339	20	3.83%	
	camel 1.2	608	20	35.53%	
	camel 1.4	872	20	16.63%	
	camel 1.6	965	20	19.48%	
	ckjm	10	20	50.00%	
	e-learning	64	20	7.81%	
	forrest 0.7	29	20	17.24%	
	ivy 1.1	111	20	56.76%	
	ivy 1.4	241	20	6.64%	
	ivy 2.0	352	20	11.36%	
	jedit 3.2	272	20	33.09%	
	jedit 4.0	306	20	24.51%	
	jedit 4.1	312	20	25.32%	
	jedit 4.2	367	20	13.08%	
	jedit 4.3	492	20	2.24%	
	kalkulator	27	20	22.22%	
	log4j 1.0	135	20	25.19%	
	log4j 1.1	109	20	33.94%	
	log4j 1.2	205	20	92.20%	
	lucene 2.0	195	20	46.67%	
	lucene 2.2	247	20	58.30%	
	lucene 2.4	340	20	59.71%	
	nieruchomosci	27	20	37.04%	
	pbeans1	26	20	76.92%	
	pbeans2	51	20	19.61%	
	pdftranslator	33	20	45.45%	
	poi 1.5	237	20	59.49%	
	poi 2.0	314	20	11.78%	
JURECZKO	poi 2.5	385	20	64.41%	loc (Number of lines of code)
	poi 3.0	442	20	63.57%	
	redaktor	176	20	15.34%	
	serapion	45	20	20.00%	

(Continued)

Table 15. Continued

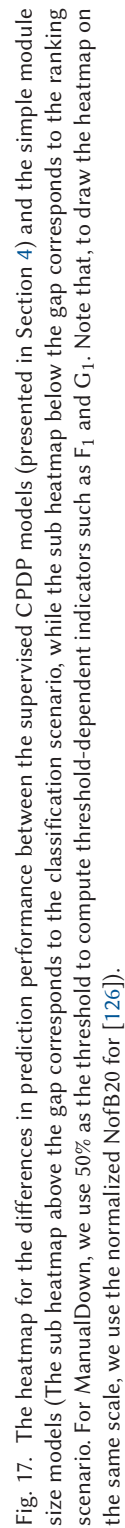
Group	Project	#Modules	#Metrics	%Defective	Module size metric
	skarbonka	45	20	20.00%	
	sklebagd	20	20	60.00%	
	synapse 1.0	157	20	10.19%	
	synapse 1.1	222	20	27.03%	
	synapse 1.2	256	20	33.59%	
	systemdata	65	20	13.85%	
	szybkafucha	25	20	50.00%	
	termoproject	42	20	30.95%	
	tomcat	858	20	8.97%	
	velocity 1.4	196	20	75.00%	
	velocity 1.5	214	20	66.36%	
	velocity 1.6	229	20	34.06%	
	workflow	39	20	51.28%	
	wspomaganiepi	18	20	66.67%	
	xalan 2.4	723	20	15.21%	
	xalan 2.5	803	20	48.19%	
	xalan 2.6	885	20	46.44%	
	xalan 2.7	909	20	98.79%	
	xerces 1.2	440	20	16.14%	
	xerces 1.3	453	20	15.23%	
	xerces 1.4	588	20	74.32%	
	xerces init	162	20	47.53%	
	zuzel	29	20	44.83%	
MDP	CM1	344	37	12.21%	LOC_EXECUTABLE (The number of lines of executable code for a module)
	JM1	9593	21	18.34%	
	KC1	2096	21	15.51%	
	KC3	200	39	18.00%	
	MC1	9277	38	0.73%	
	MC2	127	39	34.65%	
	MW1	264	37	10.23%	
	PC1	759	37	8.04%	
	PC2	1585	36	1.01%	
	PC3	1125	37	12.44%	
	PC4	1399	37	12.72%	
	PC5	17001	38	2.96%	
NETGENE	httpclient	361	465	56.79%	SumnumChangedFiles (The sum of changed files)
	jackrabbit	542	465	41.51%	
	lucene	1671	465	20.71%	
	rhino	253	465	43.08%	
RELINK	Apache2.0	194	60	50.52%	CountLineCode (Number of lines of code)
	Safe	56	41	39.29%	
	zxing1.6	399	51	29.57%	

### C THE DIFFERENCES IN PREDICTION PERFORMANCE OF THE EXISTING SUPERVISED CPDP MODELS AND THE SIMPLE MODULE SIZE MODELS

Figure 17 uses a heatmap to show the differences in prediction performance between the existing supervised CPDP models investigated in Section 4 and the simple module size model. Each row represents the combination of a supervised CPDP model and a performance indicator, while each column represents a target release. For example, the name of the first row is “[122]Zhang\_F<sub>1</sub>,” which represents the performance indicator F<sub>1</sub> of the supervised CPDP model proposed by Zhang et al. in Reference [122]. Each cell in the heatmap is filled with **gray**, **cyan**, or **red**. The gray color represents that (1) the performance difference is zero or (2) the performance difference is unavailable, as the supervised CPDP model is not applied to the corresponding target release in the original CPDP study. The cyan color represents a positive performance difference, while the red color represents a negative performance difference. In contrast to the other performance indicators, a low ED [52] or NECM [48, 59] means a better performance. Before drawing the heatmap, we multiplied the ED(NECM) difference by  $-1$ . Consequently, in Figure 17, the cyan color indicates that the CPDP model has a superior performance, while the red color indicates that the CPDP model has an inferior performance. Note that, many CPDP studies report a number of cross-project combinations with the same target releases. For example, for the target release Ant1.6, Wang et al. [109] reported two combinations: “Camel 1.4→Ant 1.6” and “Poi 3.0→Ant 1.6” but did not tell us how to automatically determine the source project for Ant 1.6. In this case, we use their average performance difference with ManualDown(50%) to draw the heatmap. From Figure 16, we can see that there are a large number of “gray” cells, indicating that most CPDP studies only use a small number of target releases to evaluate the proposed CPDP models. Furthermore, compared with ManualDown, most CPDP models exhibit either a better or a worse performance, depending on the target releases involved.

Figure 18 uses a heatmap to show the differences in AUC between 30 models investigated in Section 5 and ManualDown. Those 30 models include 24 supervised CPDP models and 4 baseline models presented in Herbold et al.’s benchmark study [35], Zhang et al.’s SC model [124], and Nam et al.’s CLA model [78]. As can be seen, for most of the 86 target releases, most benchmarked CPDP models are inferior to ManualDown. This is especially true when we look at the right half of the heatmap. Overall, it appears that the 24 supervised CPDP models analyzed in Herbold et al.’s benchmark study do not show a performance superior to ManualDown on most of the investigated 86 target releases. In addition, we can observe that both SC and CLA, the two state-of-the-art unsupervised models, underperform the simple module size model ManualDown in most cases.

Figure 19 uses a heatmap to show the difference in recall between CamargoCruz09-NB (the best model reported in Herbold et al.’s study [35, 36]) and ManualUp. We can observe that CamargoCruz09-NB underperforms ManualUp on most of the 86 datasets.



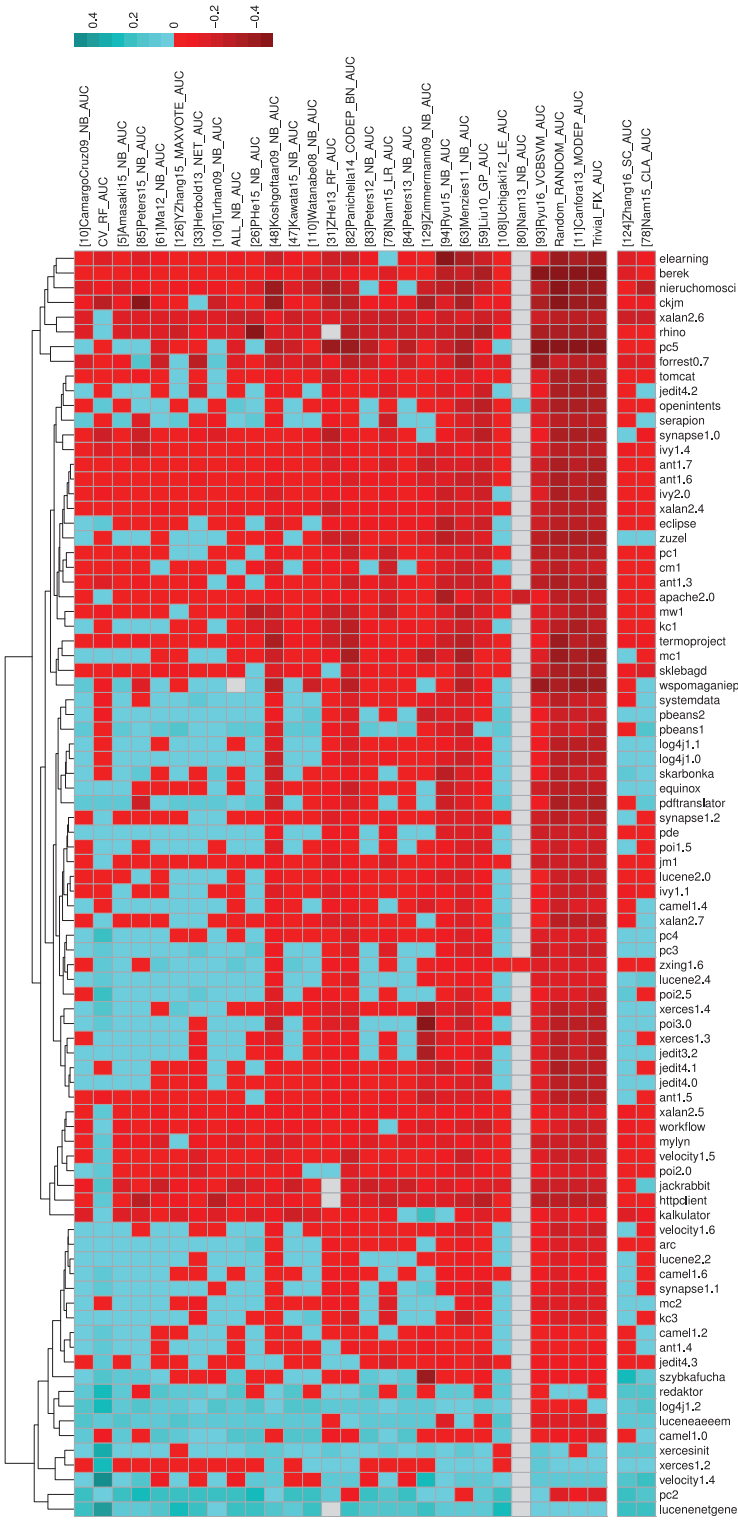


Fig. 18. The heatmap for the differences in AUC between 30 models (presented in Section 5) and the simple module size model (The sub heatmap above the gap corresponds to 24 supervised CPDP models and 4 baseline models presented in Herbold et al.'s benchmark study [35], while the sub heatmap below the gap corresponds to the two state-of-the-art unsupervised defect prediction models).

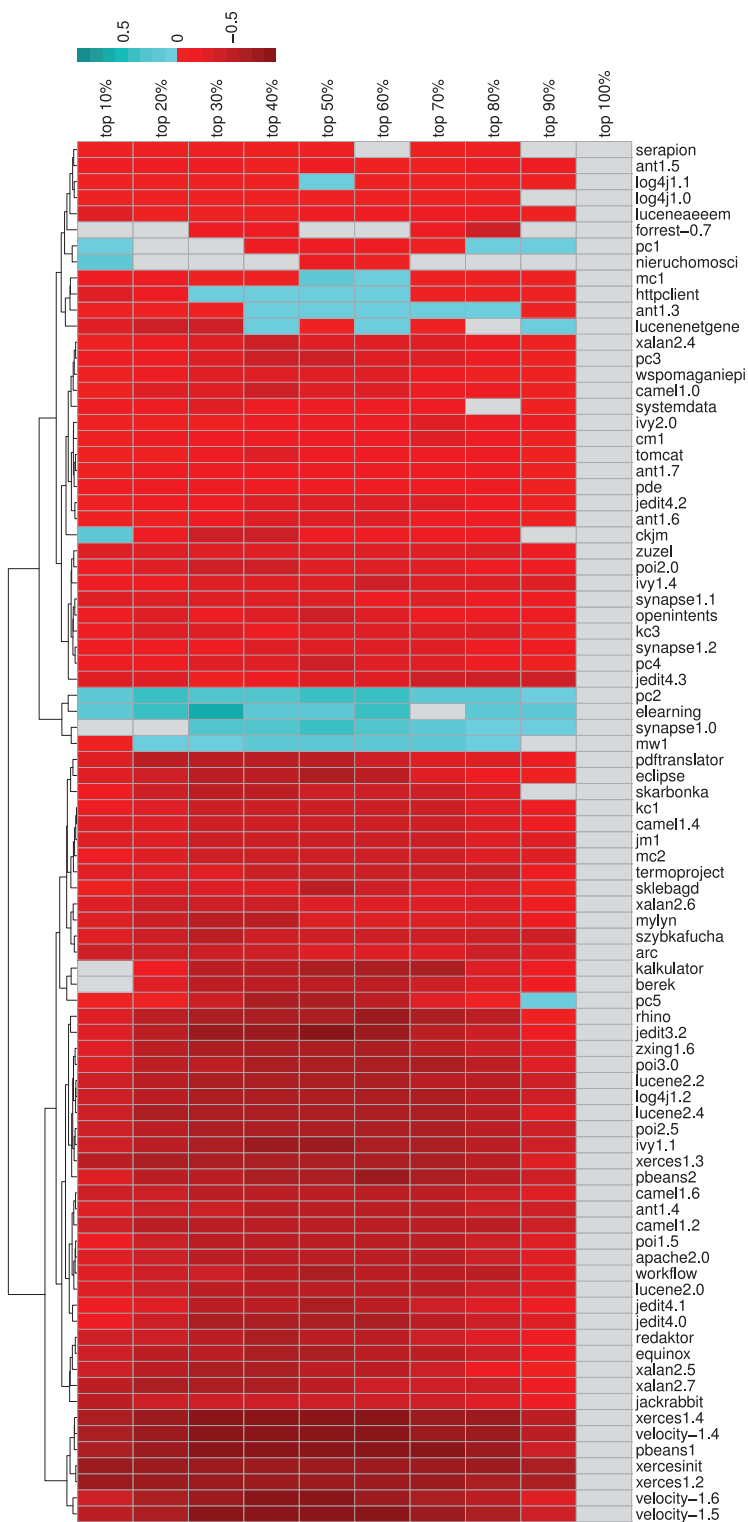


Fig. 19. The heatmap for the difference in recall between CamargoCruz09-NB and ManualUp.

## D INFLUENCE OF MODULE GRANULARITY ON PERFORMANCE COMPARISON

In Section 5.1, when comparing simple module size models with the supervised CPDP models reported in Herbold et al.'s benchmark study [35], we did not distinguish the module granularity. Indeed, of the 86 datasets, 12 NASA datasets are at the method level and the other 74 datasets are at file level. To examine the influence of module granularity, we re-run the above analyses with 12 method-level datasets and 74 file-level datasets separately.

*Influence on classification performance comparison.* Figure 20 and Figure 21, respectively, use boxplots to report the distributions of the improvement in AUC of Herbold et al.'s benchmarked models over ManualDown for the 12 method-level datasets and the 74 file-level datasets. In each figure, a blue box-plot indicates “significantly better,” a red box-plot indicates “significantly worse,”

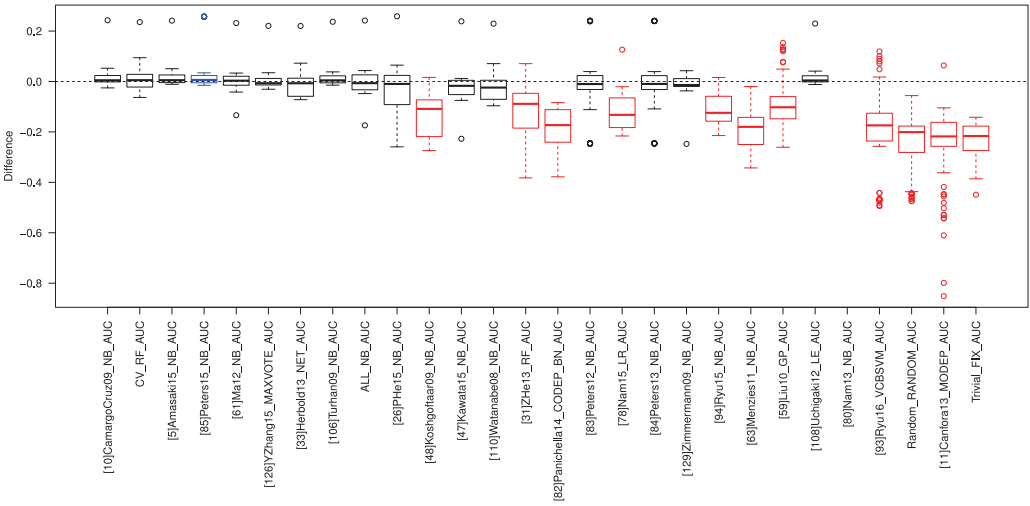


Fig. 20. The improvement in AUC on the 12 method-level datasets.

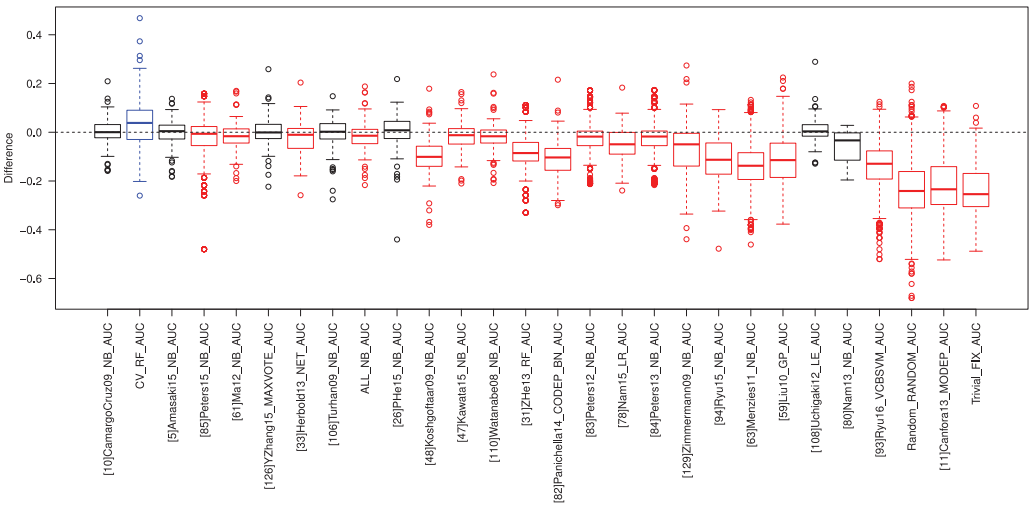


Fig. 21. The improvement in AUC on the 74 file-level datasets.



and a black box-plot indicates “no significant difference” (according to the Benjamini-Hochberg-corrected p-value returned by the Wilcoxon signed-rank test). From Figure 20, we can see that only Peters\_15\_NB is significantly better than ManualDown (BH-corrected p-value  $< 0.001$ , Cliff’s delta = 0.236). From Figure 21, we can see that only CV\_RF is significantly better than ManualDown (BH-corrected p-value = 0.017, Cliff’s delta = 0.154). Therefore, the overall conclusion drawn from Figure 20 is similar to that from Figure 21: almost all the benchmarked supervised CPDP models do not outperform ManualDown (the results from  $F_1$ ,  $G_1$ , and MCC are similar). In other words, varying module granularity does not change our conclusion that the simple module size model is very competitive with existing CPDP models in the classification scenario.

*Influence on ranking performance comparison.* Herbold et al. [35] reported that, overall, according to  $F_1$ ,  $G_1$ , MCC, and AUC, CamargoCruz09-NB performed the best among all the investigated models. Figure 22 and Figure 23, respectively, use boxplots to report the distributions of the improvement in Recall of CamargoCruz09-NB over ManualUp for the 12 method-level and the 74 file-level datasets. The observation from Figure 22 is similar to that from Figure 23: CamargoCruz09-NB does not outperform ManualUp for all the investigated top  $x\%$ . Therefore, varying module granularity does not change our conclusion that the simple module size model is very competitive with existing CPDP models in the ranking scenario.

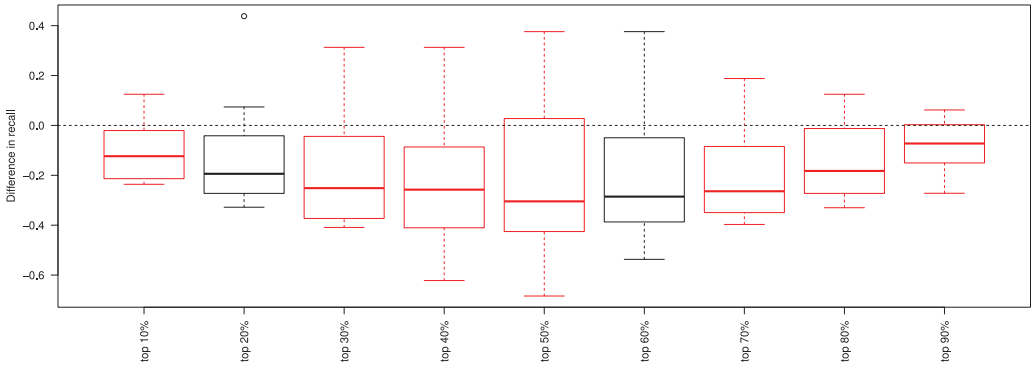


Fig. 22. The improvement in Recall on the 12 method-level datasets.

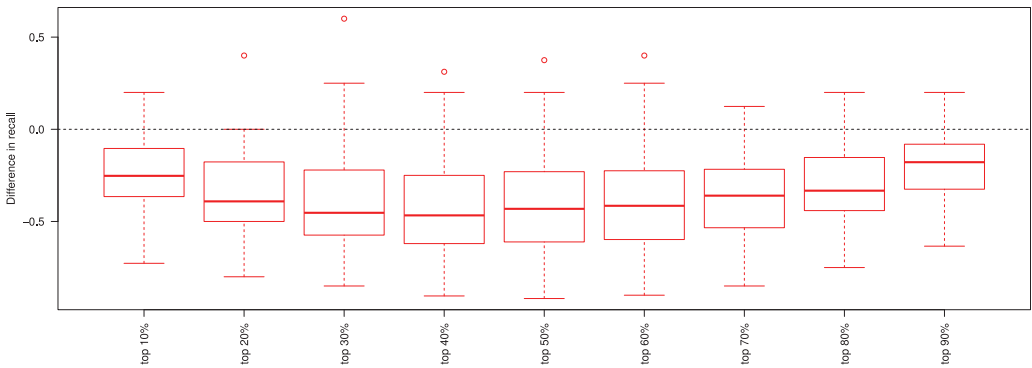


Fig. 23. The improvement in Recall on the 74 file-level datasets.

## E INFLUENCE OF PROJECT SIZE AND DEFECTIVE PERCENTAGE ON PERFORMANCE COMPARISON

In Section 5.1, when comparing simple module size models with the supervised CPDP models reported in Herbold et al.'s benchmark study [35], we did not filter out those datasets that contained very few modules and those datasets in which most modules were defective. An anonymous reviewer pointed out such datasets should be filtered out of the studied sets to prevent them from skewing the experimental results. The reason is that, in practice, there is no need to apply a defect prediction model to a very small project. Furthermore, in a real development environment, it is common that most modules in a project are not defective.

To examine the influence of project size and defective module percentage, we applied the following filters to the 86 datasets used in Herbold et al.'s benchmark study: (1) excluding the datasets whose modules were less than 200 and (2) excluding the datasets whose defective module percentages were larger than 30%. Consequently, we obtained 36 datasets. Figure 24 uses boxplots to report the distributions of the improvement in AUC of Herbold et al.'s benchmarked models over the 36 datasets. We can see that no benchmarked supervised CPDP model outperforms ManualDown (the results from  $F_1$ ,  $G_1$ , and MCC are similar). Figure 25 uses boxplots to report the distributions of

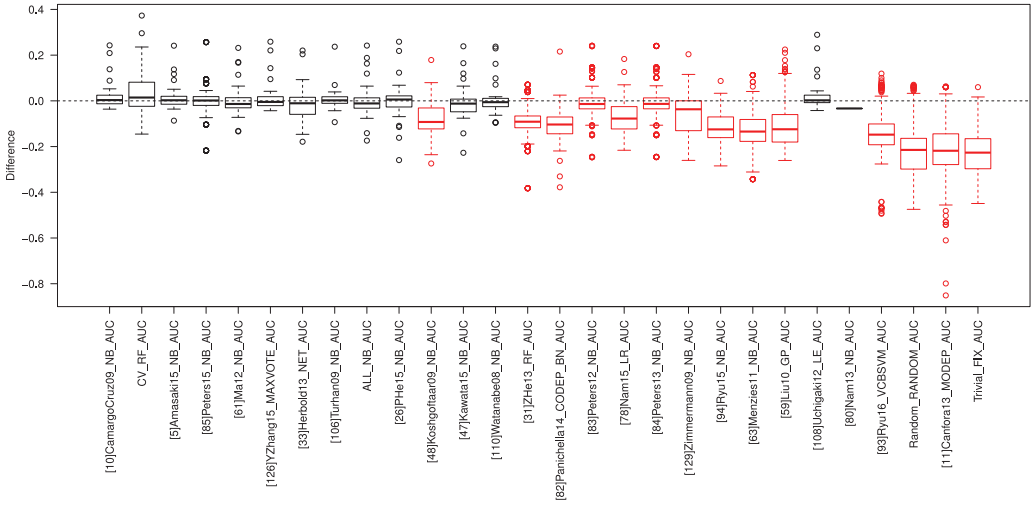


Fig. 24. The improvement in AUC on the 36 selected datasets.

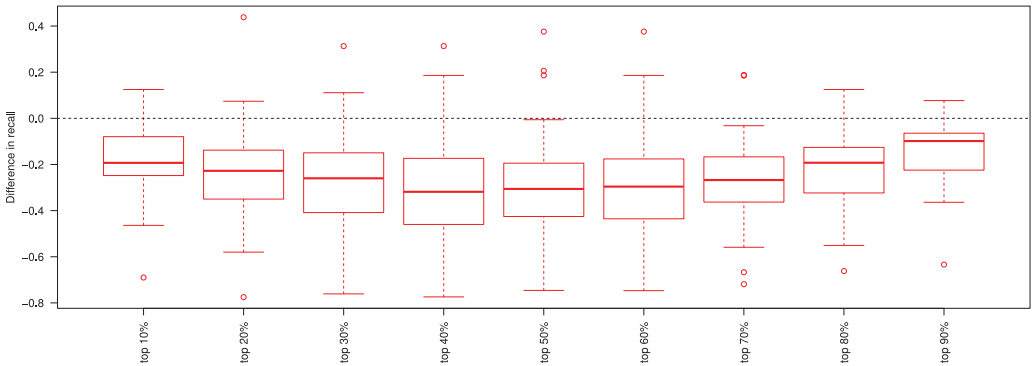


Fig. 25. The improvement in Recall on the selected 36 datasets.

the improvement in Recall of CamargoCruz09-NB over ManualUp for the 36 datasets. We can see that CamargoCruz09-NB does not outperform ManualUp for the entire investigated top  $x\%$ . The above experimental results show that our overall conclusions remain valid without change. This demonstrates that simple module size models have a robust performance over a range of different project size and defective module percentages.

## F THE SHORT DESCRIPTIONS OF THE SUPERVISED CPDP MODELS INVESTIGATED IN SECTION 4

- Zhang et al.'s MT+ [122]: It is an enhanced version of Multiple Transformations (MT) by automatically selecting the most appropriate training project for each target project based on the parameter of the Box-Cox transformation. For a given target project, MT is a model that integrates three LR (Logistic Regression) models respectively built with the log-transformed, rank-transformed, and Box-Cox transformed data.
- Stuckman et al.'s Metric+CFA [101]: It is a Random Forest (RF) model built with software metrics after applying a dimensionality reduction technique called CFA (Confirmatory Factor Analysis).
- Aarti et al.'s ANN [1]: It is an Artificial Neural Network (ANN) model built with code metrics.
- Herbold et al.'s EM [37]: It is a Support Vector Machine (SVM) model built with the data in a cluster generated by the Expectation-Maximization (EM) algorithm.
- Jing et al.'s SSTCA-ISDA [42]: It is a RF model built with the SSTCA transformed data and the class imbalance learning technique ISDA (Improved Subclass Discriminant Analysis).
- Krishna et al.'s BellWether [52]: It is a RF model built with the exemplary project that produces the best predictions on all the others. The rationale behind this is the bellwether effect, i.e., "when a community of programmers work on a set of projects, then within that community there exists one exemplary project, called the bellwether, which can define quality predictors for the other projects" [52].
- Xia et al.'s HYDRA [116]: It is a two-layer hierarchical composition of a massive number of LR models. For a given target project, HYDRA uses all the  $N$  source project data and a small number (e.g., 5%) of labeled target project data (called training target data) as the training data. At the bottom layer, HYDRA uses GA (genetic algorithm) to combine  $N + 1$  LR models (they build a model for each source project data merged with the training target data and another model for the training target data alone) to obtain a GA model. At the upper layer, HYDRA combines a number of GA classifiers generated by the principle of AdaBoost to obtain a compositional model.
- Zhang et al.'s Universal [123]: It is a Naïve Bayes (NB) model built with the context-aware rank transformed data of 1385 open-source projects hosted on SourceForge and GoogleCode.
- Wang et al.'s DBN [109]: It is an Alternating Decision Tree (ADTree) model built with the semantic features automatically learned from source code by a deep learning technique called Deep Belief Network (DBN).
- Ryu et al.'s VCB-SVM [93]: It is a model built using a technique called Value-Cognitive Boosting with Support Vector Machine (VCB-SVM) that takes into class imbalance learning and transfer learning.
- Ryu et al.'s MONBNN [92]: It is a model built using a technique called Multi-Objective Naïve Bayes with the Nearest-Neighbor filter (MONBNN) that takes into account the class imbalance context.

- Hosseini et al.'s GIS [39]: It is a NB model built with the source project data selected by a search method called Genetic Instance Selection (GIS).
- You et al.'s ROCPPD [118]: It is a MLR (Multiple Linear Regression) model built with a technique called Ranking-Oriented Cross-Project Defect Prediction (ROCPDP).
- Cheng et al.'s CCT-SVM [17]: It is a model built with a technique called Cost-sensitive Correlation Transfer Support Vector Machine (CCT-SVM) designed for dealing with the class imbalance problem under heterogeneous cross-project defect prediction.
- Kaur et al.'s POP [45]: It is a Bayes Network (BN) model built with the source project data selected by a method called Patterns by Ordered Projections (POP).
- Catal et al.'s Threshold [14]: The metric thresholds derived from source projects are used to predict defect-prone modules in a target project.
- He et al.'s TDSelector [29]: It is a LR model built with the source project data selected by a technique called TDSelector that takes into account both instance similarity and the number of defects each training instance has.
- Ryu et al.'s HISNN [94]: It is a NB model built with the source project data selected by Hybrid Instance Selection Using Nearest-Neighbor (HISNN) that deals with the class imbalance problem.
- Canfora et al.'s MODEP(LR) [12]: It is a LR model whose coefficients are optimized in terms of both cost and performance by a genetic algorithm (i.e., find as many defects for as little costs possible).
- Peters et al.'s LACE2 [85]: It is a K-Nearest Neighbor (KNN) model built with the privatized source project data by a multi-party privacy-preserving data-sharing algorithm called LACE2 (Large-scale Assurance of Confidentiality Environment (LACE)).
- Jing et al.'s CCA+ [41]: It is a KNN model built with the source project data (represented in a unified metric format) transformed by a technique called Canonical Correlation Analysis (CCA).
- Nam et al.'s HDP [79]: It is a LR model with the Kolmogorov-Smirnov Test-based matching for the heterogeneous metric sets between source and target projects.
- Singh et al.'s NB [99]: It is a NB model built with design metrics.
- Zhang et al.'s Max [126]: It is a compositional model that uses the maximum voting method to combine six underlying models, including LR, BN, Radial Basis Function Network (RBFN), Multi-Layer Perceptron (MLP), ADTree, and Decision Table (DT).
- Cao et al.'s TCANN [13]: It is an ANN model built with Transfer Component Analysis Neural Network (TCANN) that takes into account the noise data, class imbalance, and transfer learning.
- Panichella et al.'s CODEP [82]: It is a model that combines six base models, including LR, BN, RBF, MLR, ADTree, and DT. CODEP(BN): the combination technique is BN; CODEP(LR): the combination is LR.
- Mizuno et al.'s Text [66]: It is a defect-prone filtering model (inspired by the spam mail filtering) that predicts defects using tokens from source code.
- He et al.'s IFS(tca) [27]: It is a LR model built with the source project data (represented in a distribution-characteristic base format) transformed by a technique called Transfer Component Analysis (TCA).
- Peters et al.'s LACE(m40) [84]: It is a NB model built with the privatized source project data by a single-party privacy-preserving data-sharing algorithm called LACE (an instance pruner CLIFF + a data mutator MORPH). Here, m40 denotes 40% of the original source project data are retained after CLIFF is applied.

- Peters et al.'s Peter-filter [86]: It is a RF model built with the source project data selected by a training-instance-guided filter.
- Turhan et al.'s Mixed model [107]: It is a NB model built with the mixed data, i.e., the data from source projects and from other versions (not the target release) in the target project.
- Nam et al.'s TCA+ [80]: It is a LR model built with the source project data transformed by a combination of standardization and TCA.
- Ma et al.'s TNB [61]: It is a NB model built with the weighted source project data, where the weights are based on the similarity between source and target project data.
- Uchigaki et al.'s Ensemble [108]: It is a weighted average of the outputs of a collection of univariate LR models.
- He et al.'s DT [32]: It is a DT model built with the selected source project data by a brute-force search.
- Liu et al.'s GP(V-V) [59]: It is a compositional model of a collection of genetic-programming-(GP) based models built with a Validation-and-Voting (V-V) strategy. When building the GP-based models, parts of the training data are used for internal validation to avoid overfitting. A majority vote strategy is used combine the predictions of a collection of GP-based models.
- Khoshgoftaar et al.'s MLMD [48]: It is a compositional model that uses a majority vote strategy to combine the predictions of multiple learners induced on multiple training datasets.