Appendices For "Performance evaluation in defect prediction: what underlying pitfalls have surfaced and what should we do next?"

XUTONG LIU, SHIRAN LIU, ZHAOQIANG GUO, PENG ZHANG, YIBIAO YANG
HONGMIN LU, YANHUI LI, LIN CHEN, and YUMING ZHOU

Appendix A. Systematic review of performance indicator usage in recent SDP studies

Table 1: The performance indicators used in recent studies that are published within the latest six years (2017~2022) and aim to propose a new defect prediction model

	SSTCA+ISDA [3]	HDP-KS [46]	Bellwether [15]	MSMDA [4]	DBN [5]	FENCES [6]	DESCo [9]	top-core [21]	KSETE [7]	EASC [16]	TDTSR [8]
Indicator	2017	2018	2019	2019	2020	2020	2020	2021	2021	2022	2022
precision					x	x					x
recall	x				x	x			x	x	x
PF	x								x		x
F1					x	x	x			x	x
F4	x										
G1			x	x					x		
MCC									x		
AUC	x	x		x		x	x		x	x	x
recall@20%LOC					x				x	x	
IFA									x	x	
Popt								x		x	
PII@20%LOC										x	
PII@1000LOC										x	
PII@2000LOC										x	
recall@1000LOC										x	
recall@2000LOC										x	

Table 1 summarizes the performance indicators used in the 11 representative defect prediction studies shown in Table 1. In this table, a cell with "x" means that a study (indicated by the column) uses the corresponding indicator (indicated by the row) for model performance evaluation. As can be seen, more than 15 indicators are involved in these 11 defect prediction studies. In particular, there is no single common performance indicator used in all the 11 studies. Considering that most performance indicators do not coincide with each other, i.e., one model may have a higher performance under indicator A but may have a lower performance under indicator B, the diverse usage of indicators and lack of reporting detailed prediction results make across-study comparison unfeasible.

Appendix B. Detailed descriptions of SDP models evaluated in our work

- ***Bellwether.*** Given *N* projects from a community, there is an exemplary project (called bellwether) whose data yields the best predictions on all others. According to [1], we should use the bellwether project data to train a model for predicting defects in new projects in the community. In practice, the defect prediction on a target project proceeds as follows. First, take the target project as the holdout and select the best-performing project from the remaining projects. The selection is conducted in a round-robin style, and G1 is the default performance indicator used to select the "bellwether". Second, build a defect prediction model with the bellwether project data (the default modeling technique is random forest). Third, apply the resulting model to predict defects for the modules in the target project. In our study, we use the replication package shared by Krishna et al. [1] to build the Bellwether model.

- ***EASC***. According to [2], EASC indeed consists of two models: *EASC_E* and *EASC_NE*. The former is suggested as the baseline model under the effort-aware performance evaluation, while the latter is suggested as the baseline model under the non-effort-aware performance evaluation. For a target project, both *EASC_E* and *EASC_NE* first rank the predicted defective modules and non-defective modules separately and then concatenate them to obtain a final ranking. The difference is that *EASC_E* ranks the modules in descending order by *score*/LOC, while *EASC_NE* ranks the modules in descending order by *score**LOC. Here, LOC denotes the module size as measured by lines of code, while *score* is the defect-proneness predicted by a classier. In default, the used classifier is Naïve Bayes. In our study, we strictly follow the process described in Ni et al.'s paper [2] to build the *EASC_E* and *EASC_NE* models.
- ***SC.*** Zhang et al. [3] used a connectivity-based classifier, spectral clustering (SC) [4], for defect prediction. SC is performed on a graph consisting of nodes and edges. Each node represents a module, and each edge represents the connection between modules, and the weight is measured by the similarity of metric values between two modules. To construct an unsupervised defect prediction model, SC first minimizes the normalized cut of the graph. In this way, SC can achieve a low similarity across the two subgraphs and gain high similarity within each subgraph. Then, SC labels each of the two subgraphs. Based on the insight that for most metrics, the defective modules generally have larger values than the clean modules, the subgraph that has a larger average sum of metric values is labeled as "defective", and the other subgraph is labeled as "clean". In our study, we use the replication package shared by Zhang et al. [3] to build the SC model.
- ***CLA.*** The key idea of CLA is to label an unlabeled dataset by using the magnitude of metric values [5]. Specifically, CLA believes that the defective modules generally have larger values than the clean modules for most metrics. CLA includes two steps: **C**lustering instances and **LA**beling instances in clusters. First, count for each module how many of their metrics exceed the median value of a metric in the target project (referred as K). Then, group the modules with the same K value into the same cluster. Third, label the top half clusters as defective and the bottom half clusters as clean. In our study, we use the replication package shared by Nam et al. [5] to build the CLA model.
- ***FCM.*** Fuzzy C-Means (FCM) is a widely used fuzzy clustering algorithm [6] and was found to be one of the best unsupervised techniques in [7]. FCM is a clustering algorithm in which each data point belongs to each cluster center on the basis of the distance between the center and itself to a certain degree. To get prediction results of *n* modules in the target project using FCM, first, apply the FCM algorithm with clustering as 2 to get a degree score matrix with *n* rows and 2 columns. Each column indicates the degree that a module belongs to a cluster. Then, to label each module, the module is considered to belong to the cluster in which its degree is bigger than in another cluster. At last, the cluster that contains more instances is considered as "clean", and another cluster is considered as "defective". In our study, we strictly follow the process described in [7] to build the FCM model.
- ***ManualDown and ManualUp.*** ManualDown and ManualUp [8-10] are two module size-based unsupervised baseline models recommended in prior studies [11]. Given a target project, ManualDown and ManualUp respectively predict the defect proneness of a module *m* as *score*(*m*) = SLOC(m) and *score*(*m*) = 1/SLOC(*m*). The intuition behind ManualDown is that larger modules may have more defects and therefore should be inspected/tested first. In contrast, the intuition behind ManualUp is that smaller modules may have larger defect density and therefore should be inspected/tested first.
- ***CamargoCruz09-NB, Amasaki15-NB, and Peters15-NB***. Herbold et al. [12] conducted a comparative analysis of 114 CPDP models on 86 defect data sets. They reported that the three best-performing models were CamargoCruz09-NB, Amasaki15-NB, and Peters15-NB. CamargoCruz09-NB standardizes the target and training data using the logarithm and the target data as a reference point. Amasaki15-NB combines attribute selection and relevancy filtering: first, the data is log-transformed; then, the metrics that are closest to any other metric are moved out from the whole dataset; finally, the instances that are the closest to any other instance are filtered out. Peters15-NB leveraging LACE2 as an extension of CLIFF and MORPH for privatization. All of those three models use Naïve Bayes as the classifier. In our study, we use the replication package shared by Herbold et al. [12,13] to build these three models.
- ***Top-core***. For a project, Qu et al. [14] applied the "coreness" of a module in the class dependency network to improve the order of classes in the suspicious class list produced by effort-aware bug prediction models. First, compute the "coreness" of each module in the target project. Second, predict the defect-proneness score $p(m)$ using a prediction algorithm. Third, define the predicted risk of a module *m* as $R_{\text{topcore}}(m) = p(m) / E(m) \times coreness(m)$, where $E(m)$ denotes the code inspection effort E(m) measured by SLOC. Fourth, rank the modules in descending order by the predicted risk. Indeed, top-core rescales the predicted scores by the "coreness" and module size. The intuitions behind top-core are: (1) a module with a larger "coreness" is more likely to be defective and hence should be ranked at the front; and (2) a module with a larger size tends to have a lower predicted defect density and hence should be ranked at the bottom. In our study, we use the replication package shared by Qu et al. [14] to build the top-core model.
- ***MSMDA***. Li et al. [15] proposed a multi-source selection based manifold discriminant alignment (MSMDA) approach to utilizing

multiple sources of training data effectively. In nature, MSMDA is an incremental optimization process by adding a data source at each iteration to generate the final training data. For a target project, first, use each source project with a limit percentage (10% as default) of modules (named training target data) in the target project to train the prediction model and test on the remaining modules of the target project (named test target data). Second, add the source project that performs best in terms of the goal performance indicator into the data cache and record its performance as the threshold. Third, combine a source project with the modules in data cache and training target data to train a model and test on the test target data. If the performance is better than the threshold, then add this source project into the data cache and refresh the threshold. The process is complete when all the source projects have been traversed. Finally, use the source projects in the data cache as the training data. In our study, we use the replication package shared by Li et al. [4] to build the MSMDA model.

- ***KSETE***. Tong et al. [16] proposed a kernel spectral embedding transfer ensemble (KSETE) model for HDP. In nature, this model combines kernel spectral embedding, transfer learning, and ensemble learning to find the latent common feature space for the source and target datasets. First, use SMOTE on the source dataset to alleviate the class-imbalance problem and normalize the source dataset and the target dataset with Z-score. Second, randomly sample the same number of modules from the source and target datasets. Third, perform a kernel spectral embedding to get the projected source and target datasets. The goal is to find a latent common kernel feature subspace that the subspace not only maximizes the similarity between the projected source and target datasets but also preserves the intrinsic characteristic of both the source and target datasets. The process is repeated to find a series of pairs of projected source and target datasets. Finally, build a classifier on each projected source dataset and combine multiple classifiers to predict the labels of the projected target dataset using transfer ensemble learning. In our study, we use the KSETE replication package shared by Tong et al. [16] to build the KSETE model.

Appendix C. Evaluating MSMDA and top-core under MATTER

To investigate the progress in defect prediction achieved by new defect prediction models, we apply MATTER to evaluate two new defect prediction models, MSMDA [15] and top-core [14]. The evaluation of MSMDA and top-core are conducted on their original published data sets separately for the following reasons:

(1) **Data availability problem for top-core.** In top-core, for a module, the predicted defectiveness depends on its “coreness” that is computed from a graph extracted from the project’s source code. In their original study, the authors of top-core provided such information for 8 test projects. However, for the other test projects used in our study, such information is unavailable. At the first glance, it seems that we could extract such information for each test project by ourselves. However, many data sets do not provide the corresponding source code and hence we have to download their source code from their project websites. During this process, we find that there are two problems. First, since many test projects do not provide the exact version number, we are unable to find the corresponding correct source code. Second, for those test projects that provide the corresponding version number, we find that many modules in the test data sets are missing in the source code we download due to unknown reasons. This means we cannot compare top-core with other defect prediction models under the same modules in a test project. Therefore, we only use the same 8 test projects (shared by the authors of top-core) to compare ONE and top-core (see Table 2).

(2) **Computation complexity problem for MSMDA.** MSMDA is a multi-source selection based manifold discriminant alignment model. We use a 64GB RAM Windows machine to run single-source MDA, which is a simplified version of MSMDA and hence has a lower computation complexity. However, MDA is still a high memory usage program, which is mainly caused by its matrix computations and the “out-of-memory failure” occurs when larger datasets such as JURECZKO are used. This indicates that MSMDA does not scale to large data sets due to the inherent high computation complexity. Therefore, we report their comparison on the same test sets (see Table 3) as used in the paper that MSMDA is originally proposed and evaluated.

Detailed experiment settings of top-core vs. ONE. In the original top-core study, their evaluation of top-core is conducted on (1) cross-validation data split strategy and (2) forward-release data split strategy. The former strategy randomly split the data set into training data and test data, which is considered to be unrealistic for a real-world scenario [17], for example, modules in a project that are developed earlier may be divided into test data while modules that are developed later may be divided into training data, then the prediction model will be built on the future knowledge that should not be known. The second strategy cannot be replicated by us due to the cross-version data sets with “coreness” are not available (i.e., the data availability problem stated before). Therefore, the comparison between top-core and ONE is conducted under the all-to-one data split strategy on projects in Table 2. For example, when Camel in Table 2 is the test project, all other seven projects are used to be the training data as a whole.

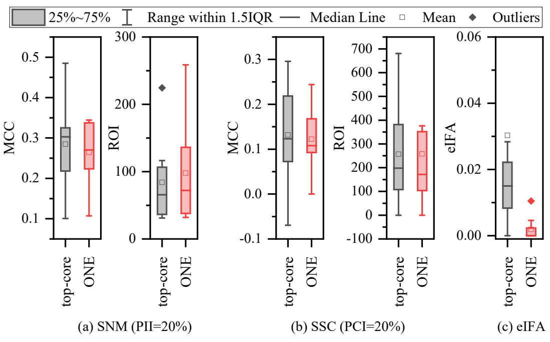
Detailed experiment settings of MSMDA vs. ONE. MSMDA is a multi-source selection based model designed specifically for HDP (Heterogeneous Defect Prediction) scenario. In particular, in its original study, MSMDA uses all the projects from external datasets plus 10% modules in the test sets as the training data, and the remaining 90% modules in the test sets as the test data. For example, when EQ from the AEEEM dataset in Table 3 is the target project, then all other 23 projects outside the AEEEM dataset are used to be the training data as a whole, meanwhile, 10% of modules in EQ are plus to the training data and the remaining 90% of modules will be the test data. Since our goal is to evaluate the effectiveness of MSMDA, we decide to evaluate MSMDA under its original goal scenario by following its data split strategy to compare MSMDA with ONE on projects in Table 3. Note that ONE is also conducted on the same remaining 90% of modules to make a fair comparison.

Table 2: Details of data sets in the original top-core studies whose "coreness" of modules are publicly available

Study	Dataset	Project	#instance	%defective	#metrics
Top-core	tera-PROMISE	Camel	965	19%	20
		Ivy	352	11%	20
		Log4j	109	34%	20
		Poi	442	64%	20
		Synapse	256	34%	20
		Tomcat	858	9%	20
		Velocity	229	34%	20
		Xalan	885	46%	20

Table 3: Details of data sets used in the original MSMDA studies

Study	Dataset	Project	#instance	%defective	#metrics
MSMDA	NASA	CM1	327	12.84%	37
		MW1	253	10.67%	37
		PC1	705	8.65%	37
		PC3	1077	12.44%	37
		PC4	1458	12.21%	37
	SOFTLAB	AR1	121	7.44%	29
		AR3	63	12.70%	29
		AR4	107	18.69%	29
		AR5	36	22.22%	29
		AR6	101	14.85%	29
	ReLink	Apache	194	50.52%	26
		Safe	56	39.29%	26
		ZXing	399	29.57%	26
	AEEEM	EQ	324	39.81%	61
		JDT	997	20.66%	61
		LC	691	9.26%	61
		ML	1862	13.16%	61
		PDE	1497	13.96%	61
	PROMISE	ant1.3	125	16.00%	20
		arc	234	11.54%	20
		camel1.0	339	3.83%	20
		poi1.5	237	59.49%	20
		redaktor	176	15.34%	20
		skarbonka	45	20.00%	20
		tomcat	858	8.97%	20
		velocity1.4	196	75.00%	20
		xalan2.4	723	15.21%	20
		xerces1.2	440	16.14%	20



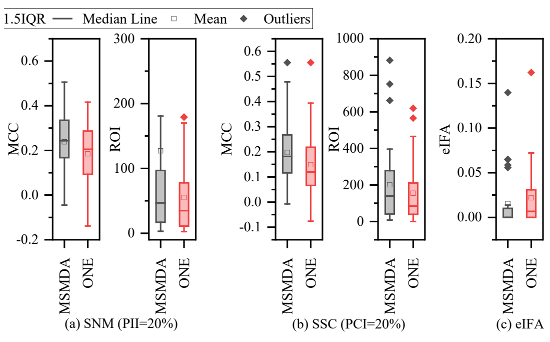


Figure 1: The performance distributions of top-core and ONE on the same target sets with the original study of top-core in terms of MCC, ROI, and eIFA under SNM and SSC

Figure 2: The performance distributions of MSMDA and ONE on the same target sets with the original study of top-core in terms of MCC, ROI, and eIFA under SNM and SSC

Fig 1. reports the performance distributions of top-core vs. ONE. From Fig. 1, we can observe that:

- MCC: top-core exhibits a similar distribution compared with ONE (p-value = 0.641 under SNM, p-value = 0.742 under SSC);
- ROI: top-core exhibits a similar distribution compared with ONE (p-value = 0.055 under SNM, p-value = 0.945 under SSC);
- eIFA: top-core is significantly worse than ONE (p-value = 0.022) and the effect size is large (cliff's delta = 0.813).

By the above results, we can conclude that top-core underperforms ONE.

Fig.2 reports the performance distributions of MSMDA vs. ONE. From Fig. 2, we can observe that:

- MCC: Compared with ONE, MSMDA performs slightly better under SNM (p-value = 0.021, cliff's delta = 0.212, a small effect size) and similarly under SSC (p-value = 0.057);
- ROI: Under SNM, no significant difference is observed (p-value = 0.264). MSMDA is significantly better (p-value = 0.036) than ONE but the effect size is trivial (cliff's delta = 0.102) under SSC;
- eIFA: there is no significant difference between MSMDA and ONE (p-value = 0.305).

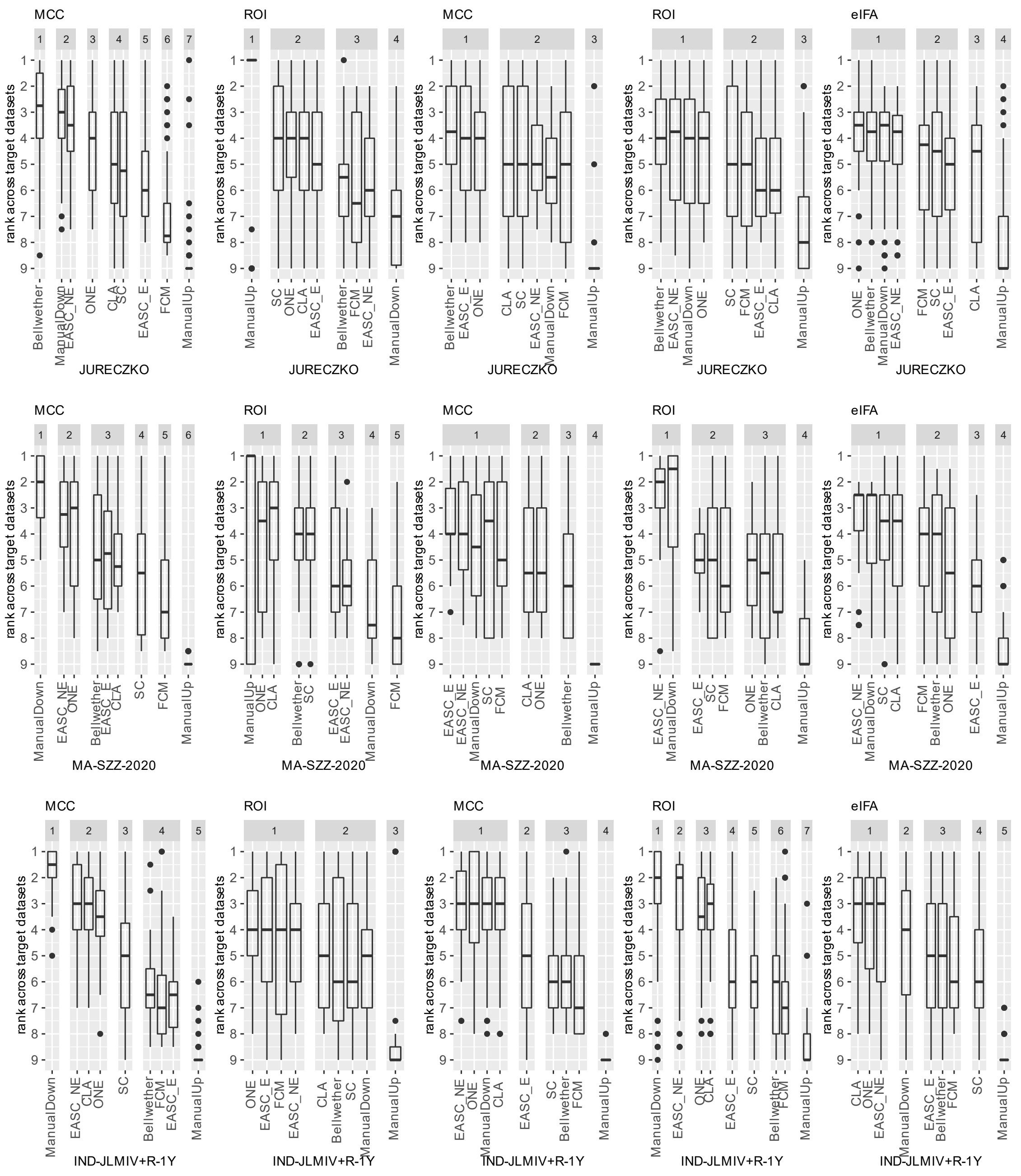
By the above results, we can conclude that, from the viewpoint of practical application, MSMDA does not lead to an important progress in prediction performance.

Appendix D. Comparisons between ONE and other baseline models on individual datasets.

To evaluate whether ONE is "strong" enough as a baseline model for defect prediction, we compare its prediction performance with other defect prediction models that are recommended to be baseline models or have the potential to be baseline models (Bellwether [1], EASC_E [2], EASC_NE [2], ManualDown [8-10], ManualUp [8-10], SC [3], CLA [5], and FCM [7]). In this section, we report the comparison results on five individual datasets used in our experiment, AEEEM [18], JURECZKO [19], MA-SZZ-2020 [20], IND-JLMIV+R [21], and ReLink [22], respectively. Table 4 reports the mean and standard deviation values and Table 5 reports the median values in terms of MCC, ROI, and eIFA of each baseline model on five datasets, respectively. The best mean/median performance value of each indicator is marked in **bold** while the worst mean/median performance value is marked in underlined. In Fig. 3, the multiple models are grouped by the non-parametric Scott-Knott ESD test [23], and models in the same group are negligible in their performance ranking distributions. Note that we do not conduct the Scott-Knott ESD test on AEEEM and ReLink since they only contain five and three samples (i.e., test projects), respectively, and the statistical test on such a small sample is usually considered to be unreliable.

According to Table 4, Table 5, and Fig. 3, we have the following observations:

(1) According to Fig. 3, (1) on JURECZKO, ONE performs at the top-level under SSC and at the upper-middle level under SSC, meanwhile its eIFA is the best among models; (2) on MA-SZZ-2020, ONE performs top-level ROI under SNM, while performs middle-level on other indicators; (3) on IND-JLMIV+R, ONE performs top-level MCC under SSC, ROI under SNM, and eIFA, while the ROI under SSC and MCC under SNM are at the upper-middle-level.



(a) under SNM (PII=20%) (b) under SSC (PCI=20%) (c) eIFA

Figure 3: The distributions of rankings of multiple models' comparisons in terms of performance indicator and the non-parametric Scott-Knott ESD test result (the smaller the group number on the top of the plot, the better the performance rankings)

(2) ONE achieves better prediction performance than ManualDown and ManualUp. First, ManualDown performs consistently badly in terms of ROI under SNM, since it requires a lot of code inspection effort to inspect the top largest modules when models are allowed to inspect the same number of modules. Second, for a total of 25 times multiple models' comparisons, ManualUp performs the worst median values for 21/25 comparisons and worst mean values for 20/25 comparisons, it is particularly not good at evaluations under SSC and the eIFA. As can be seen from Fig 3, it is at the worst to the middle level in terms of ROI under SNM.

(3) As can be seen from Table 4, EASC_NE and Bellwether are the best supervised models among the multiple models' comparisons on multiple datasets. However, supervised models such as EASC_NE, EASC_E or Bellwether cannot provide a stable performance for target data when the training data changes. As the training data varies with the data split strategy or the data resampling algorithm, the prediction performance of a supervised model on the same target project may vary in a range, on the contrary, the performance of ONE on the same target will keep the same. Considering those supervised models are not often

significantly better than ONE according to the model grouping of the Scott-Knott ESD test, we still recommend ONE, rather than those compared supervised models to be the baseline model in MATTER.

Table 4. The mean(standard deviation) MCC, ROI, and eIFA values of each baseline model on five datasets

Dataset	Model	SNM (PII=20%)		SSC (PCI=20%)		eIFA
		MCC	ROI	MCC	ROI	
AEEEM	Bellwether	0.276(0.085)	133.0(63.9)	0.184(0.059)	541.2(301.6)	**0.000(0.000)**
	EASC-E	0.237(0.079)	157.0(103.3)	0.169(0.024)	420.4(219.5)	0.001(0.002)
	EASC-NE	**0.311(0.085)**	126.3(61.5)	0.126(0.065)	607.0(424.2)	0.012(0.018)
	SC	0.141(0.122)	188.1(89.3)	0.124(0.119)	310.1(230.3)	0.006(0.006)
	CLA	0.241(0.060)	127.3(43.1)	0.152(0.077)	428.3(296.2)	0.001(0.003)
	FCM	0.017(0.092)	126.0(62.1)	**0.192(0.098)**	464.1(251.4)	0.001(0.002)
	ManualDown	0.279(0.109)	113.1(60.1)	0.125(0.066)	**609.9(439.6)**	0.012(0.018)
	ManualUp	-0.149(0.111)	**1652.7(1018.4)**	-0.287(0.110)	96.9(33.6)	0.025(0.027)
	ONE	0.252(0.094)	141.5(72.9)	0.144(0.082)	455.9(267.9)	0.003(0.006)
JURECZKO	Bellwether	0.261(0.184)	53.4(72.3)	**0.144(0.120)**	**181.1(192.4)**	**0.011(0.035)**
	EASC-E	0.165(0.146)	78.6(160.8)	0.123(0.137)	140.5(160.6)	0.018(0.051)
	EASC-NE	0.263(0.161)	50.0(61.1)	0.115(0.119)	167.9(212.0)	0.021(0.047)
	SC	0.170(0.199)	96.5(224.1)	0.105(0.154)	159.1(179.4)	0.024(0.061)
	CLA	0.204(0.173)	66.8(100.9)	0.101(0.136)	145.1(174.9)	0.024(0.048)
	FCM	0.019(0.215)	61.6(94.2)	0.071(0.178)	149.7(198.9)	0.022(0.055)
	ManualDown	**0.277(0.170)**	45.4(51.0)	0.101(0.115)	160.7(206.3)	0.021(0.046)
	ManualUp	-0.191(0.140)	**2633.2(6377.9)**	-0.293(0.164)	78.6(137.6)	0.067(0.096)
	ONE	0.248(0.174)	56.4(65.5)	0.129(0.119)	158.5(196.2)	0.013(0.041)
IND-JLMIV+R	Bellwether	0.070(0.118)	30.3(22.9)	0.047(0.107)	55.0(62.7)	0.040(0.055)
	EASC-E	0.072(0.130)	49.9(145.4)	0.087(0.139)	85.7(139.6)	0.036(0.047)
	EASC-NE	0.245(0.122)	31.1(21.4)	0.197(0.129)	141.5(158.3)	0.027(0.051)
	SC	0.128(0.130)	29.6(21.6)	0.048(0.089)	53.9(59.4)	0.045(0.046)
	CLA	0.253(0.121)	29.7(21.2)	0.177(0.119)	144.1(152.5)	0.019(0.032)
	FCM	0.049(0.145)	44.4(86.3)	0.025(0.116)	45.7(58.9)	0.048(0.061)
	ManualDown	**0.307(0.127)**	28.5(19.1)	0.193(0.121)	**167.3(161.2)**	0.033(0.053)
	ManualUp	-0.147(0.048)	**3636.1(18902.9)**	-0.280(0.099)	14.2(18.5)	0.203(0.102)
	ONE	0.237(0.110)	32.4(23.7)	**0.202(0.113)**	148.9(167.8)	**0.017(0.027)**
MA-SZZ-2020	Bellwether	0.184(0.111)	82.1(55.3)	0.120(0.101)	169.8(98.7)	0.012(0.020)
	EASC-E	0.192(0.088)	84.8(59.5)	0.152(0.052)	210.5(171.1)	0.017(0.011)
	EASC-NE	0.238(0.077)	76.4(55.8)	**0.166(0.083)**	**294.1(190.3)**	0.008(0.020)
	SC	0.136(0.100)	88.0(62.3)	0.119(0.090)	204.0(195.3)	0.011(0.022)
	CLA	0.183(0.076)	89.2(65.7)	0.132(0.082)	174.4(138.5)	0.007(0.010)
	FCM	0.047(0.156)	55.9(54.3)	0.127(0.096)	184.6(162.3)	0.016(0.050)
	ManualDown	**0.263(0.090)**	66.0(48.4)	0.129(0.093)	257.5(223.8)	0.017(0.047)
	ManualUp	-0.159(0.062)	**278.3(356.6)**	-0.277(0.090)	60.9(68.4)	0.095(0.103)
	ONE	0.232(0.091)	78.7(60.7)	0.111(0.074)	182.7(174.6)	0.029(0.049)
ReLink	Bellwether	0.139(0.212)	66.7(41.6)	0.107(0.152)	115.8(87.0)	**0.000(0.000)**
	EASC-E	0.192(0.329)	122.9(86.1)	**0.212(0.184)**	116.1(74.7)	0.002(0.002)
	EASC-NE	**0.361(0.224)**	41.8(22.6)	0.161(0.068)	**157.2(99.0)**	**0.000(0.000)**
	SC	0.053(0.078)	65.4(38.0)	0.022(0.045)	88.8(60.5)	0.007(0.006)
	CLA	0.283(0.132)	46.2(27.7)	0.134(0.145)	116.5(67.3)	0.019(0.034)
	FCM	0.059(0.374)	208.8(306.3)	-0.013(0.366)	82.4(33.2)	0.007(0.012)
	ManualDown	0.338(0.253)	37.6(18.1)	0.150(0.079)	145.1(82.1)	**0.000(0.000)**
	ManualUp	-0.238(0.095)	**2047.0(1796.2)**	-0.372(0.179)	59.5(42.9)	0.032(0.027)
	ONE	0.272(0.163)	48.9(26.7)	0.170(0.111)	134.3(80.9)	0.006(0.006)

Table 5. The median MCC, ROI, and eIFA values of each baseline model on five datasets

Dataset	Model	SNM (PII=20%)		SSC (PCI=20%)		eIFA
		MCC	ROI	MCC	ROI	
AEEEM	Bellwether	0.223	150.8	0.151	575.8	**0.00**
	EASC-E	0.242	171.4	0.170	317	0.00
	EASC-NE	**0.262**	152.2	0.154	**592.6**	0.00
	SC	0.167	196.2	0.170	250.3	0.00
	CLA	0.252	146.5	0.177	276.6	0.00
	FCM	-0.039	153.9	**0.243**	298.8	0.00
	ManualDown	0.257	149.3	0.154	561.4	0.00
	ManualUp	-0.124	**1963.6**	-0.262	101.1	0.01
	ONE	0.218	189.5	0.156	526.4	0.00
JURECZKO	Bellwether	0.249	29.8	**0.145**	**112.8**	**0.00**
	EASC-E	0.165	34.5	0.138	96.4	0.00
	EASC-NE	0.263	29.6	0.089	87.2	0.00
	SC	0.146	36.5	0.095	104.5	0.00
	CLA	0.194	32.6	0.098	89.3	0.00
	FCM	0.002	24.7	0.090	73	0.00
	ManualDown	**0.271**	28.6	0.086	84.1	0.00
	ManualUp	-0.179	**340.3**	-0.291	26.8	0.03
	ONE	0.228	32.7	0.124	104.6	0.00
IND-JLMIV+R	Bellwether	0.053	25.6	0.032	33.2	0.02
	EASC-E	0.082	**29.5**	0.089	45.8	0.02
	EASC-NE	0.246	25.6	0.193	95.7	0.00
	SC	0.123	22.9	0.049	44.1	0.03
	CLA	0.248	24.6	0.182	97.9	**0.00**
	FCM	0.031	24.1	0.019	25.5	0.02
	ManualDown	**0.266**	24.7	**0.199**	**117.9**	0.01
	ManualUp	-0.142	0	-0.273	9.9	0.20
	ONE	0.206	26.2	0.196	94.6	0.00
MA-SZZ-2020	Bellwether	0.203	71.8	0.118	156.4	0.00
	EASC-E	0.183	69.1	**0.160**	173.3	0.01
	EASC-NE	0.259	61.8	0.146	**296.4**	**0.00**
	SC	0.132	80.4	0.137	137.2	0.00
	CLA	0.169	72.3	0.124	153.3	0.00
	FCM	0.097	44.7	0.137	204.4	0.00
	ManualDown	**0.271**	62.2	0.139	240.4	0.00
	ManualUp	-0.139	**109.5**	-0.285	33.6	0.07
	ONE	0.251	68.5	0.122	155.6	0.00
ReLink	Bellwether	0.160	69.3	0.103	89.9	**0.00**
	EASC-E	0.188	141.5	**0.197**	102.7	0.00
	EASC-NE	0.281	48.5	0.126	161.7	0.00
	SC	0.064	76.6	0.040	110.9	0.01
	CLA	0.243	51	0.087	104.5	0.00
	FCM	0.064	49.1	0.003	71.6	0.00
	ManualDown	0.281	47.7	0.117	**161.7**	0.00
	ManualUp	-0.214	**1024.9**	-0.387	71.6	0.02
	ONE	**0.281**	62.9	0.132	129.3	0.00

Based on the above observations, under both SNM and SSC, ONE has a competitive prediction performance in multiple models' comparisons on most of the five defect datasets. Although ManualDown and ManualUp are also simple in implementation and stable in prediction results for a given target data, ONE performs better and is preferred to be a global reference point for across-study comparison. For the other compared models, (1) the performance of supervised models depends on the choice of the training data and therefore they are inappropriate to be a global reference point for across-study model performance comparison; (2) although SC, CLA, and FCM are unsupervised (i.e., they are free of the influence of the training data), their prediction performance may vary with the feature set used for model building. Therefore, they are also inappropriate to be a global reference point for across-study comparison. In summary, the comparison results on five individual datasets support ONE to be the appropriate baseline model in evaluating defect prediction models.

Appendix E. MCC performance distributions of EASE_NE under PII=20% on the test set

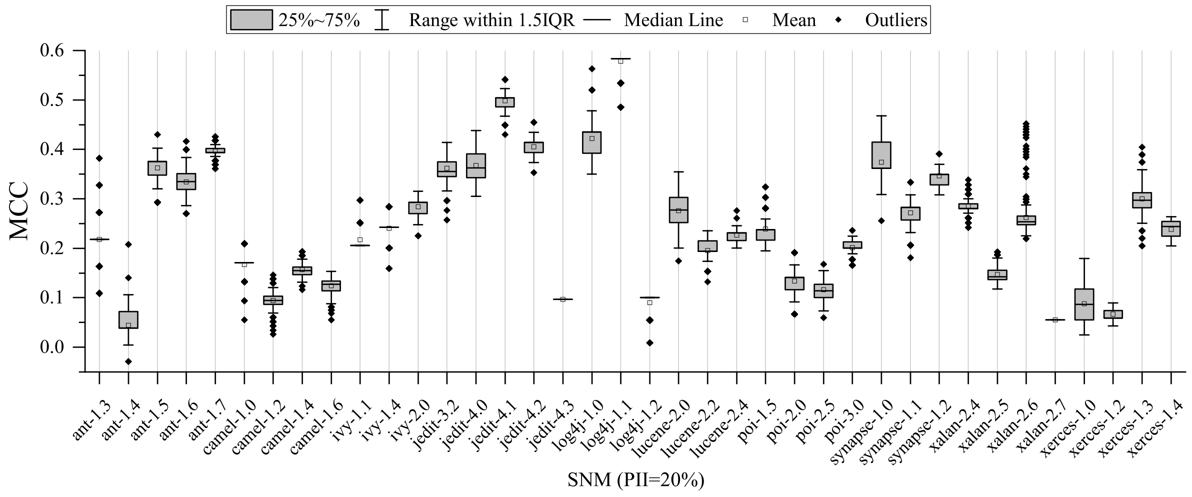


Figure. 4. MCC performance distributions of EASE_NE under PII=20% on the test set (i.e., the x-axis) when randomly sampling half sets from all available training sets 500 times in the JURECZKO dataset as the training data

Fig. 4 shows the MCC performance distributions of EASE_NE under PII=20% on the test set when randomly sampling half sets from all available training sets 500 times in the JURECZKO dataset as the training data. As can be seen, the prediction performance of EASC_NE on the same test project varies a lot with the training data used. This pitfall makes it inappropriate to be a global reference point for across-study model performance comparison.

Appendix F. Evaluating state-of-the-art models under MATTER on individual datasets.

In this section, we report the effectiveness of four state-of-the-art defect prediction models (CamargoCruz09-NB [12], Amasaki15-NB [12], Peters15-NB [12], and KSETE [16]) under MATTER on five individual datasets, AEEEM [18], JURECZKO [19], MA-SZZ-2020 [20], IND-JLMIV+R [21], and ReLink [22], respectively. Table 6 reports the mean and standard deviation values and Table 7 report the median values in terms of MCC, ROI, and eIFA of each state-of-the-art defect prediction model and the baseline ONE. The best mean/median performance value of each indicator is marked in **bold** while the worst mean/median performance value is marked in underlined. In Fig. 4, the multiple models are grouped by the non-parametric Scott-Knott ESD test [23], and models in the same group are negligible in their performance ranking distributions. Note that we do not conduct the Scott-Knott ESD test on AEEEM and ReLink for the same reason in Appendix C.

According to Table 6, Table 7, and Fig. 5, we have the following observations:

(1) In terms of eIFA, on all datasets (JURECZKO, MA-SZZ-2020, and IND-JLMIV+R) in Fig. 5, ONE achieves top-level according to the Scott-Knott ESD test, meanwhile, in Table 7, ONE performs the best median eIFA on three from five datasets. We can conclude that those four evaluated state-of-the-art defect prediction models are not able to cost less SQA-effort to find the first defective module than the baseline ONE.

(2) According to the mean and standard deviation values in Table 6, ONE wins for 8/25 comparisons, which is the best among compared models, and the second best is CamargoCruz09-NB who wins for 6/25 comparisons. According to the median values in Table 7, the best is ONE who wins for 10/25 comparisons and Peters15-NB is the second best who wins for 9/25 comparisons. Meanwhile, KSETE achieves 12/25 worst mean and 9/25 worst median values among multiple models' comparisons.

(3) According to the Scott-Knott ESD test result, we find that on the JURECZKO dataset, Peters15-NB performs the best among multiple models under both SNM and SSC; on the IND-JLMIV+R dataset, ONE performs the best among multiple models under

both SNM and SSC; on the MA-SZZ-2020 dataset, CamargoCruz09-NB and Peters15-NB performs best under SSC while ONE performs best under SNM. Overall, among the comparisons on multiple datasets, we found Peters15-NB and CamargoCruz09-NB perform relatively well in comparison with those four representative defect prediction models. However, none of them are significantly better than ONE under MATTER on more than one dataset.

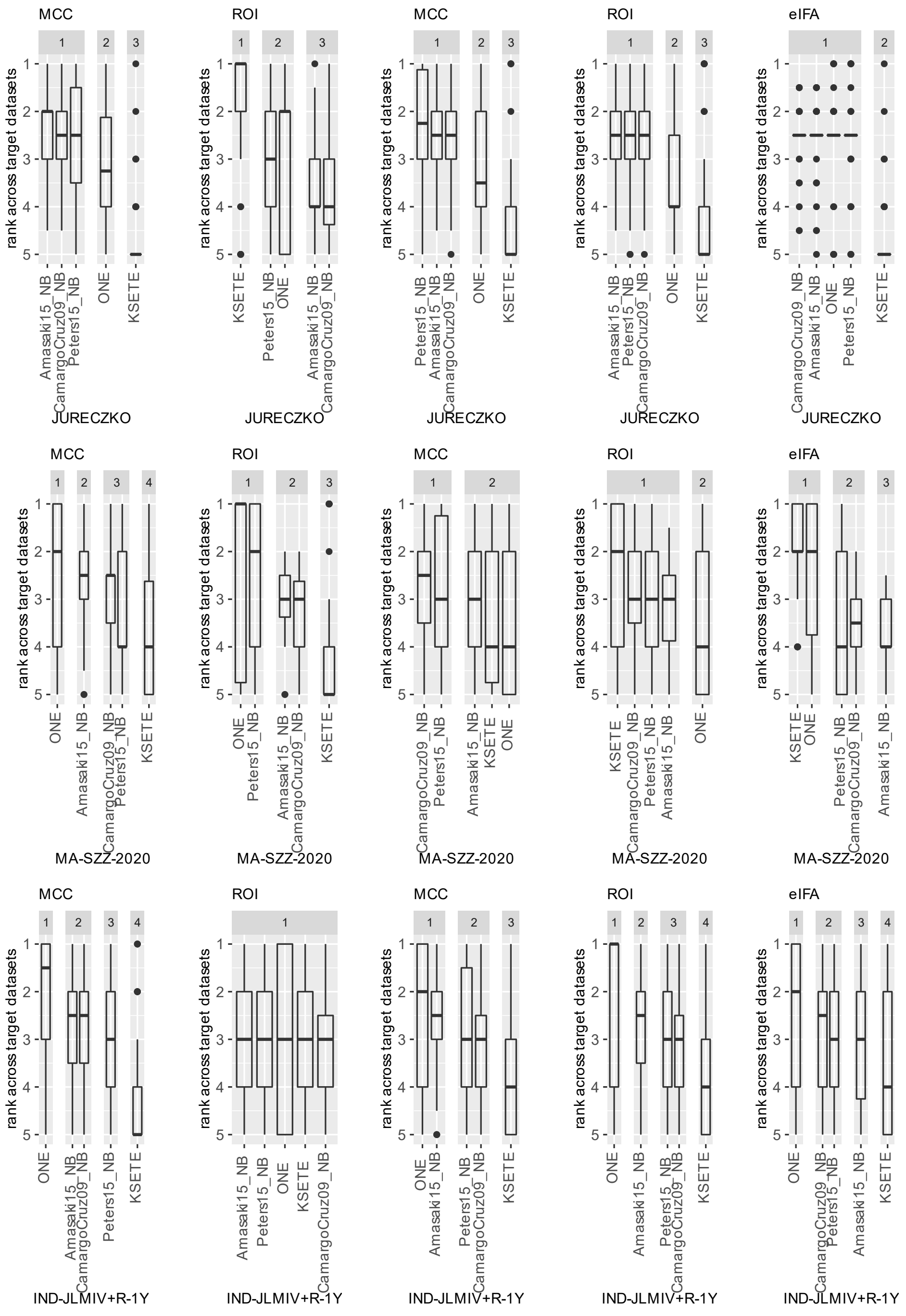
Conclusion. *In terms of defect datasets, ONE performs (1) top-level on IND-JLMIV+R, (2) top-level on MA-SZZ-2020 under SNM and bottom-level under SSC, (3) middle-level on JURECZKO, (3) and win the most time in terms of median and mean values. Overall, these models do not consistently outperform ONE on different datasets. This means that, if the practical prediction effectiveness is the goal, the real progress in defect prediction is not being achieved as has been reported in the literature.*

Table 6. The mean(standard deviation) MCC, ROI, and eIFA values of each state-of-the-art defect prediction model and ONE on five datasets

Dataset	Model	SNM (PII=20%)		SSC (PCI=20%)		eIFA
		MCC	ROI	MCC	ROI	
AEEEM	KSETE	0.284(0.086)	124.5(52.1)	0.181(0.049)	513.1(200.5)	**0.002(0.001)**
	CamargoCruz09-NB	**0.312(0.082)**	128.1(53.2)	**0.196(0.057)**	**664.1(322.9)**	0.012(0.021)
	Amasaki15-NB	0.308(0.092)	127.1(57.7)	0.194(0.069)	612.1(249.3)	0.009(0.011)
	Peters15-NB	0.303(0.077)	126.2(61.6)	0.172(0.036)	585.1(308.5)	0.015(0.018)
	ONE	0.252(0.094)	**141.5(72.9)**	0.144(0.082)	455.9(267.9)	0.003(0.006)
JURECZKO	KSETE	0.089(0.082)	**143.0(299.8)**	0.035(0.068)	137.5(166.4)	0.023(0.041)
	CamargoCruz09-NB	0.264(0.154)	57.5(84.3)	0.144(0.110)	195.0(203.9)	**0.008(0.022)**
	Amasaki15-NB	**0.267(0.154)**	57.4(83.7)	0.142(0.105)	**195.7(204.6)**	0.009(0.023)
	Peters15-NB	0.255(0.176)	60.5(85.7)	**0.148(0.150)**	189.3(196.8)	0.010(0.030)
	ONE	0.248(0.174)	56.4(65.5)	0.129(0.119)	158.5(196.2)	0.013(0.041)
IND-JLMIV+R	KSETE	0.150(0.093)	**34.5(24.7)**	0.117(0.079)	97.2(116.2)	0.024(0.022)
	CamargoCruz09-NB	0.191(0.119)	33.6(22.7)	0.159(0.130)	128.9(167.6)	0.028(0.046)
	Amasaki15-NB	0.195(0.114)	34.1(23.7)	0.164(0.120)	131.6(172.0)	0.030(0.044)
	Peters15-NB	0.157(0.136)	34.1(25.0)	0.153(0.163)	129.6(172.1)	0.033(0.059)
	ONE	**0.237(0.110)**	32.4(23.7)	**0.202(0.113)**	**148.9(167.8)**	**0.017(0.027)**
MA-SZZ-2020	KSETE	0.199(0.053)	71.7(60.0)	0.129(0.044)	232.1(178.0)	**0.012(0.014)**
	CamargoCruz09-NB	0.218(0.063)	79.3(55.3)	**0.167(0.084)**	**255.9(206.5)**	0.027(0.020)
	Amasaki15-NB	0.223(0.061)	79.4(55.0)	0.160(0.082)	245.5(194.0)	0.029(0.016)
	Peters15-NB	0.223(0.090)	**84.0(54.8)**	0.159(0.083)	226.5(174.2)	0.027(0.020)
	ONE	**0.232(0.091)**	78.7(60.7)	0.111(0.074)	182.7(174.6)	0.029(0.049)
ReLink	KSETE	0.152(0.065)	38.0(21.7)	0.130(0.096)	119.9(57.1)	**0.004(0.005)**
	CamargoCruz09-NB	0.299(0.198)	39.1(19.3)	0.050(0.170)	66.2(63.8)	0.056(0.053)
	Amasaki15-NB	**0.308(0.187)**	40.0(20.3)	0.093(0.096)	104.7(61.2)	0.048(0.024)
	ONE	0.272(0.163)	**48.9(26.7)**	**0.170(0.111)**	134.3(80.9)	0.006(0.006)

Table 7. The median MCC, ROI, and eIFA values of each state-of-the-art defect prediction model and ONE on five datasets

Dataset	Model	SNM (PII=20%)		SSC (PCI=20%)		eIFA
		MCC	ROI	MCC	ROI	
AEEEM	KSETE	**0.290**	146.3	0.167	570.4	0.002
	CamargoCruz09-NB	0.277	154.3	0.180	**693.1**	**0.000**
	Amasaki15-NB	0.266	157.6	0.160	637.6	0.003
	Peters15-NB	0.281	152.5	**0.183**	613.5	0.009
	ONE	0.218	**189.5**	0.156	526.4	**0.000**
JURECZKO	KSETE	0.092	43.0	0.032	90.9	0.014
	CamargoCruz09-NB	0.243	30.5	0.157	121.8	**0.000**
	Amasaki15-NB	0.250	32.0	0.155	125.7	**0.000**
	Peters15-NB	**0.263**	33.3	**0.163**	**125.7**	**0.000**
	ONE	0.228	32.7	0.124	104.6	**0.000**
IND-JLMIV+R	KSETE	0.145	26.9	0.122	64.5	0.016
	CamargoCruz09-NB	0.179	27.1	0.162	76.9	0.010
	Amasaki15-NB	0.194	**28.5**	0.162	76.0	0.011
	Peters15-NB	0.162	28.1	0.156	67.6	0.011
	ONE	**0.206**	26.2	**0.196**	**94.6**	**0.008**
MA-SZZ-2020	KSETE	0.208	56.1	0.128	197.7	0.007
	CamargoCruz09-NB	0.223	64.1	0.154	197.9	0.024
	Amasaki15-NB	0.231	63.7	0.139	**201.4**	0.025
	Peters15-NB	0.232	71.0	**0.158**	195.0	0.022
	ONE	**0.251**	68.5	0.122	155.6	**0.006**
ReLink	KSETE	0.177	50.5	0.092	137.4	0.002
	CamargoCruz09-NB	0.229	46.0	0.081	42.0	0.027
	Amasaki15-NB	0.229	46.2	0.054	107.8	0.050
	Peters15-NB	0.203	44.2	**0.132**	**138.6**	**0.000**
	ONE	**0.281**	62.9	0.132	129.3	0.007



(a) under SNM (PII=20%) (b) under SSC (PCI=20%) (c) eIFA

Figure 5: The distributions of rankings of multiple models' comparisons in terms of performance indicator and the non-parametric Scott-Knott ESD test result (the smaller the group number on the top of the plot, the better the performance rankings)

REFERENCES

[1] Rahul Krishna and Tim Menzies. 2019. Bellwethers: A Baseline Method for Transfer Learning. IEEE Transactions on Software Engineering 45, 11 (2019), 1081-1105. https://doi.org/10.1109/TSE.2018.2821670

[2] Chao Ni, Xin Xia, David Lo, Xiang Chen and Qing Gu. 2022. Revisiting Supervised and Unsupervised Methods for Effort-Aware Cross-Project Defect Prediction. IEEE Transactions on Software Engineering 48, 3 (2022), 786-802. https://doi.org/10.1109/TSE.2020.3001739

[3] Feng Zhang, Quan Zheng, Ying Zou and Ahmed E. Hassan. 2016. Cross-Project Defect Prediction Using a Connectivity-Based Unsupervised Classifier. In 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). Austin, TX, USA, 309-320. https://doi.org/10.1145/2884781.2884839

[4] Ulrike von Luxburg. 2007. A tutorial on spectral clustering. Statistics and Computing 17, 4 (2007), 395-416. https://doi.org/10.1007/s11222-007-9033-z

[5] Jaechang Nam and Sunghun Kim. 2015. CLAMI: Defect Prediction on Unlabeled Datasets (T). In 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). Lincoln, NE, USA, 452-463. https://doi.org/10.1109/ASE.2015.56

[6] James C. Bezdek, Robert Ehrlich and William Full. 1984. FCM: The fuzzy c-means clustering algorithm. Computers & Geosciences 10, 2 (1984), 191-203.

https://doi.org/10.1016/0098-3004(84)90020-7

[7] Ning Li, Martin J. Shepperd and Yuchen Guo. 2020. A systematic review of unsupervised learning techniques for software defect prediction. Information and Software Technology 122 (2020), 106287. https://doi.org/https://doi.org/10.1016/j.infsof.2020.106287

[8] Akif Günes Koru, Khaled El Emam, Dongsong Zhang, Hongfang Liu and Divya Mathew. 2008. Theory of relative defect proneness. Empirical Software Engineering 13, 5 (2008), 473. https://doi.org/10.1007/s10664-008-9080-x

[9] Akif Günes Koru, Hongfang Liu, Dongsong Zhang and Khaled El Emam. 2010. Testing the theory of relative defect proneness for closed-source software. Empirical Software Engineering 15, 6 (2010), 577-598. https://doi.org/10.1007/s10664-010-9132-x

[10] Akif Günes Koru, Dongsong Zhang, Khaled El Emam and Hongfang Liu. 2009. An Investigation into the Functional Form of the Size-Defect Relationship for Software Modules. IEEE Transactions on Software Engineering 35, 2 (2009), 293-304. https://doi.org/10.1109/TSE.2008.90

[11] Yuming Zhou, Yibiao Yang, Hongmin Lu, Lin Chen, Yanhui Li, Yangyang Zhao, Junyan Qian and Baowen Xu. 2018. How Far We Have Progressed in the Journey? An Examination of Cross-Project Defect Prediction. ACM Trans. Softw. Eng. Methodol. 27, 1 (2018), Article 1. https://doi.org/10.1145/3183339

[12] Steffen Herbold, Alexander Trautsch and Jens Grabowski. 2018. A Comparative Study to Benchmark Cross-Project Defect Prediction Approaches. IEEE Transactions on Software Engineering 44, 9 (2018), 811-833. https://doi.org/10.1109/TSE.2017.2724538

[13] Steffen Herbold, Alexander Trautsch and Jens Grabowski. 2019. Correction of "A Comparative Study to Benchmark Cross-Project Defect Prediction Approaches". IEEE Transactions on Software Engineering 45, 6 (2019), 632-636. https://doi.org/10.1109/TSE.2018.2790413

[14] Yu Qu, Qinghua Zheng, Jianlei Chi, Yangxu Jin, Ancheng He, Di Cui, Hengshan Zhang and Ting Liu. 2021. Using K-core Decomposition on Class Dependency Networks to Improve Bug Prediction Model's Practical Performance. IEEE Transactions on Software Engineering 47, 2 (2021), 348-366. https://doi.org/10.1109/TSE.2019.2892959

[15] Z. Li, X. Y. Jing, X. Zhu, H. Zhang, B. Xu and S. Ying. 2019. On the Multiple Sources and Privacy Preservation Issues for Heterogeneous Defect Prediction. IEEE Transactions on Software Engineering 45, 4 (2019), 391-411. https://doi.org/10.1109/TSE.2017.2780222

[16] Haonan Tong, Bin Liu and Shihai Wang. 2021. Kernel Spectral Embedding Transfer Ensemble for Heterogeneous Defect Prediction. IEEE Transactions on Software Engineering 47, 9 (2021), 1886-1906. https://doi.org/10.1109/TSE.2019.2939303

[17] Duksan Ryu, Jong-In Jang and Jongmoon Baik. 2015. A Hybrid Instance Selection Using Nearest-Neighbor for Cross-Project Defect Prediction. Journal of Computer Science and Technology 30, 5 (2015), 969-980. https://doi.org/10.1007/s11390-015-1575-5

[18] Marco D'Ambros, Michele Lanza and Romain Robbes. 2010. An extensive comparison of bug prediction approaches. In 7th International Working Conference on Mining Software Repositories, MSR 2010 (Co-located with ICSE). Cape Town, South Africa, 31-41. https://doi.org/10.1109/MSR.2010.5463279

[19] Marian Jureczko and Lech Madeyski. 2010. Towards identifying software project clusters with regard to defect prediction. In Proceedings of the 6th International Conference on Predictive Models in Software Engineering. Association for Computing Machinery, Timişoara, Romania, Article 9. https://doi.org/10.1145/1868328.1868342

[20] Shiran Liu, Zhaoqiang Guo, Yanhui Li, Chuanqi Wang, Lin Chen, Zhongbin Sun, Yuming Zhou and Baowen Xu. 2022. Inconsistent defect labels: essence, causes, and influence. IEEE Transactions on Software Engineering (2022) https://doi.org/10.1109/TSE.2022.3156787

[21] Steffen Herbold, Alexander Trautsch, Fabian Trautsch and Benjamin Ledel. 2022. Problems with SZZ and Features: An empirical study of the state of practice of defect prediction data collection. Empirical Software Engineering 27, 2 (2022), 1-49. https://doi.org/10.1007/s10664-021-10092-4

[22] Rongxin Wu, Hongyu Zhang, Sunghun Kim and Shing-Chi Cheung. 2011. ReLink: recovering links between bugs and changes. In Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering. Association for Computing Machinery, Szeged, Hungary, 15–25. https://doi.org/10.1145/2025113.2025120

[23] Chakkrit Tantithamthavorn. 2017. Scottknottesd: The scott-knott effect size difference (esd) test. R package version 2 (2017)