Using Bandit Algorithms for Project Selection in Cross-Project Defect Prediction

Takuya Asano
Kindai University
Higashi-osaka, Japan

Masateru Tsunoda
Kindai University
Higashi-osaka, Japan
tsunoda@info.kindai.ac.jp

Koji Toda
Fukuoka Institute of Technology
Fukuoka, Japan
toda@fit.ac.jp

Amjed Tahir
Massey University
Palmerston North, New Zealand
a.tahir@massey.ac.nz

Kwabena Ebo Bennin
Wageningen University & Research
Wageningen, Netherlands
kwabena.bennin@wur.nl

Keitaro Nakasai
National Institute of Technology, Kagoshima College
Kirishima, Japan
nakasai@kagoshima-ct.ac.jp

Akito Monden
Okayama University
Okayama, Japan
monden@okayama-u.ac.jp

Kenichi Matsumoto
Nara Institute of Science and Technology
Ikoma, Japan
matumoto@is.naist.jp

***Abstract*—Background: defect prediction model is built using historical data from previous versions/releases of the same project. However, such historical data may not exist in case of newly developed projects. Alternatively, one can train a model using data obtained from external projects. This approach is known as cross-project defect prediction (CPDP). In CPDP, it is still difficult to utilize external projects' data or decide which particular project to use to train a model. Aim: to address this issue, we apply bandit algorithm (BA) to CPDP in order to select the most suitable training project from a set of projects. Method: BA-based prediction iteratively reselects the project after each module is tested, considering the accuracy of the predictions. As baselines, we used simple CPDP methods such as training a model with randomly selected project. All models were built using logistic regression. Results: We experimented our approach on two datasets (NASA and DAMB, with a total of 12 projects). The BA-based defect prediction models resulted in, on average, a higher accuracy (AUC and F1 score) than the baselines. Conclusion: in this preliminarily study, we demonstrate the feasibility of using BA in the context of CPDP. Our initial assessment shows that the use BA for predicting defects in CPDP is promising and may outperform existing approaches.**

Keywords—fault prediction, CPFP, multi-armed bandit, external validity, online optimization, risk-based testing

I. INTRODUCTION

Defect prediction plays an important role in software quality control. The goal is to find defects as early in the development as possible. To achieve this, a defect prediction model is built using data collected on the previous version of the prediction target software. However, if the software is newly developed, or data was not collected on the previous versions, then there exist no training data to build a model. A feasible solution is to use data collected from other software projects (obtained internally or externally). This is referred to as cross-project defect prediction (CPDP). CPDP have been attracting increased attention in recent years [4], with many studies attempting to study the feasibility of using historical data obtained from publicly available datasets [6][8]. However, the characteristics of software projects can be quite diverse. Evidently, CPDP models trained on arbitrarily selected projects that are different from the target project does not perform so well [11]. To increase the accuracy of CPDP models, several studies [8][12] have proposed various solutions such as improving data filtering techniques, but the prediction performance of these models compared are still low to the performance of within-project defect prediction (WPDP) models [4].

In the paper, we propose the use of Bandit Algorithm (BA) to CPDP to help with selecting projects used as training data. BA was previously applied to WPDP models for selecting the best prediction model from multiple ones built by different prediction methods such as logistic regression and decision tree [3]. In [3], the best prediction method was selected from a number of candidate methods using BA, while in this study we select the best training project data from a number of candidate projects.

II. RELATED WORK

The majority of the CPDP approaches are data-driven which mainly focuses on how to handle training data [4]. One of the most common approaches in CPDP is to randomly select a project as training data from a set of different data sources. To enhance the accuracy of the prediction, previous studies [6][8][11][12] focused on finding projects that are similar to the target projects. The impact of these approaches is different among studies. For instance, the early work of Turhan et al. [11] attempted to identify projects with the most similar data instances to the test dataset using the nearest neighbor algorithm. However, the prediction performance was not significant in most cases. Kamei et al. [6] attempted to identify similar projects based on the distribution of different metrics, but this resulted in a low accuracy prediction. In contrast, Zimmermann et al. [12] estimated the similarity between training and target projects based on a number of characteristics such as the application domain of the projects. Kuramoto et al. [8] focused on the relationships between metrics such as lines of code (LOC) and complexity, and selected a project based on the similarity of the relationships between training and target projects. In addition to the simple approach, other CPDP studies have proposed more advanced data-driven approaches such as filtering data points of projects [11].

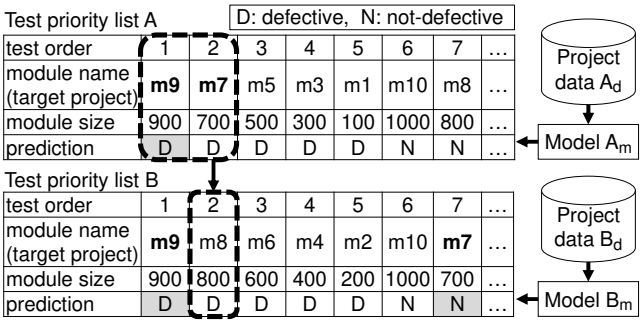

Fig. 1. Test priority (training project) selection based on BA.

Ensemble methods are modeling techniques which have been widely used in CPDP studies [4]. In defect prediction studies, ensemble methods often treat multiple prediction models built by different methods such as logistic regression and decision tree, and combine their prediction by a method such as voting. In CPDP studies [6][8], instead of voting for different prediction methods, this is typically applied to project selection. That is, the methods combine prediction of models trained by different projects. The ensemble methods have shown to result in high performance (i.e., in terms of AUC) [4]. For example, in CPDP context, the ensemble method based on voting showed higher accuracy compared to data-driven approach explained above [6]. Although voting is the simplest ensemble method, some studies have employed more advanced methods. For instance, in [8], random forest was used as ensemble method, and the accuracy was higher than the data-driven approach.

III. Bandit Algorithm Based Defect Prediction

A. Overview

Test priority based on defect prediction: In test prioritization, different modules can be tested in a sequential order. Same as study [11], we assume that the priority of each test will depend on the defectiveness of each model, so that defective modules are prioritized and tested before non-defective ones. Note that coding of all modules should be completed before the prediction, because the values of explanatory variables of the model such as LOC are unknown before the completion.

Fig. 1 illustrates an example of the test priority lists based on two different training projects. In the lists, modules (m1, m2, …) are sorted based on prediction outcomes and module size (i.e., defective *and* larger modules (in terms of LOC) are tested first). This is because maintaining larger modules may requires more time compared to smaller modules [9]. However, size can have a rather an indirect (mediating or moderating) effect between different metrics and defectiveness [10].

Based on the simple data-driven approach explained in Section II, the prediction model is built using training date from other projects. However, from a set of projects, it is not entirely clear which project will provide the most efficient results (i.e., with the highest accuracy).

Application of BA to CPDP: BA can be explained using an example from slot machines. Assume that a player has 100 coins to bet on two slot machines, and the chance of winning (i.e., expected reward) is different between the machines. To maximize the player's reward, BA does not recommend to select only one machine and bet all 100 coins on it, but recommends that the player bet a coin on each slot machine step by step, to seek the best ones..

As shown in Fig. 1, tested modules are shown in bold, and the gray-shaded cells show the prediction is same as the test results. (1) The priority list A is selected, and based on the list, modules m9 and m7 are tested (being the defective and largest modules). (2) Only the prediction of m9 corresponds with the actual test result on the list A. In contrast, for the list B, the prediction of m9 and m7 corresponds with the actual results. Consequently, the list B is more effective than A at this point. (3) The priority list is switched from A to B, and module m8 (being the defective, largest, and non-tested modules) is tested based on the list B.

In Fig. 1, list A is made by model A_m, which is built using project data A_d. List B is made in the same way (with model B_m and project B_d). That is, selecting the list by BA is equivalent to selecting the project data. Note that model retraining is not needed after the selection as the project data did not change during testing.

B. Reward

We refer to test priority lists based on prediction models as arms. Algorithm 1 illustrates the mechanism for our proposed BA-based defect prediction. In the algorithm, the variable *n_modules* denotes the number of test target modules, the variable *n_arms* is the number of arms.

Line 2 and 3 select the arm whose expected reward is the highest. We use Thompson Sampling (TS) to calculate rewards. TS samples a value from the beta distribution for each arm - an arm is then selected when the value is the maximum. TS has been shown to achieve higher accuracy compared to other BA algorithms [2].

The beta distribution is the distribution of the probability of "success" on a Bernoulli trial, and the distribution has parameters α and β. Assuming that "success" occurs m times, and "failure" occurs n times on the trial, the estimated probability of the occurrence of "success" will follow the beta distribution where α is $m + 1$ and β is $n + 1$. Using TS, the expected reward r is calculated as:

$$r = \text{random.beta}\,(m + 1, n + 1) \qquad (1)$$

where random.beta samples a value from beta distribution with $\alpha = m + 1$ and $\beta = n + 1$, m is the number of True Positives (TP), and n is the number of False Negatives (FN). As a result, the expected reward of an arm increases with the increase in the number of TP and the decrease in the number of FNs. Line 3 in Algorithm 1 denotes equation (1) of each arm.

When the prediction outcome is a False Positive (FP) or True Negative (TN), the value of n and m will not be changed. For example, in Fig. 1, m is increased after testing module m9, because the prediction is a TP for both arms. However, m and n are not increased after testing m7, because the prediction is FP

Algorithm 1: BA-based defect prediction using TS

```
1: for t in range(n_modules):
2:     for i in range(n_arms)])
3:         exp_rew[i] = random.beta(m[i] + 1, n[i] + 1)
4:     eval = test_and_eval(argmax(exp_rew))
5:     for i in range(n_arms):
6:         if eval[i] == TP then m[i] += 1
7:         if eval[i] == FN then n[i] += 1
```

on arm A and TN on arm B. This is because overlooking of defects usually occurs during software testing, and therefore a TN and FP is not always correct (we explain this in more detail in Section IV).

In line 4 of Algorithm 1, *test_and_eval* tests the module, and sets a list of *eval* of all arms based on a comparison of the test result and the prediction. *argmax(exp_rew)* returns the index of the arm whose expected reward is the highest. The method takes the index, and select the priority list of the arm as shown in Fig. 1. In lines 5 to 7, we set parameter *m* and *n* to calculate the expected reward *r* based on the evaluation of each arm. Line 1 means that the above procedure is conducted for each test target module.

C. Applicability of BA to defect predciotn

Although BA can be applied to various usage of defect predictions, the assumptions of BA fit well to the characteristics of integration test. That is, as explained in section A, each module is tested sequentially. Integration testing usually runs longer and will require more time to perform compared to unit testing (where modules are tested in isolation).

Generally, it is difficult to detect all defects during testing. Therefore, the comparison of prediction and test results (i.e., reward of BA) would not be absolutely accurate. However, most defects are found during integration test. A previous study [5] reported that about 83% of defects found after unit test were detected on integration test.

IV. Experiment

Dataset: For our experiment, we used projects obtained from two datasets, i.e., NASA and DAMB. Both datasets are often used in CPDP studies [4][8][11]. The NASA dataset contains a set of metrics and defects data collected (as part of the NASA Metrics Data Program) from a number of NASA projects, where DAMB was collected from open source projects. In CPDP studies, it is not unusual that datasets are filtered based on the programming language used (e.g., [11]). For consistency, we selected projects which were written in C from the NASA dataset. All the projects in DAMB are written in Java.

We included the following NASA projects: CM1, MC2, MW1, PC1, PC2, PC3, and PC4. On average, 1541 modules were included, and the rate of defective modules was 11.3%. These project contain data from 40 different metrics which are complexity metrics (e.g., McCabe Cyclomatic complexity and Halsted) and size-oriented metrics (e.g., lines of code and the number of operands). DAMB datasets were collected from five OSS projects by D'Ambros et al. [1] (namely: Eclipse JDT Core, Eclipse PDE UI, Eclipse Equinox Framework, Eclipse Mylyn and Apache Lucene). The projects have data obtained from 17 metrics including the CK metrics and size-oriented metrics such as lines of code and number of methods. On average, 1074 modules were included with modules defectiveness rate of 19.4%.

When evaluating our BA-based defect prediction, we applied it repeatedly 20 times on each project, and calculated the average of the evaluation criteria acquired from the 20 repetition. This is because we are using TS which is a probabilistic approach (see Section III.C), and hence we tried to suppress the randomness. Note that we did not apply cross validation methods such as k-fold cross validation as the training and test datasets are both different.

CPDP methods: To predict defective modules, we used logistic regression, as it is one of the most widely used methods in CPDP [4]. As a feature selection method, we applied correlation based feature selection, which is shown to be effective when used together with logistic regression [7]. For the NASA dataset, we built six prediction models m_1, m_2 … and m_6 using six different projects (e.g., MC2, MW1, PC1, PC2, PC3, and PC4) with different projects used as target and test datasets (i.e., CM1). Then, we used the prediction results as the arms of BA. Similarly, for the DAMB dataset, we built four prediction models using four projects as target and used one as the test dataset. This approach is alternated, with each project being used as a test dataset at least once.

From the data-driven approaches explained in section II, we adopted the simplest one as one of the baselines for this study. One of the baselines is the *simple data approach*. It selects a training dataset which resulted in the highest AUC, similar to the methods of previous studies [6][8][12]. As an example, for the CM1 as a testing entry, we found that the highest AUC measure can be achieved when PC4 is used as the training project, thus PC4 is selected.

Note that previous studies [6][8][12] used projects' similarity (i.e., between test target and external projects) to select a suitable training project. Here, instead of similarity, we used AUC. Using AUC is an advantageous to simple data approaches, because the similarity based selection did not always result in selecting the best training project.

The other baseline is the *simple ensemble*. It uses the same models as BA, and merged the predictions using a majority voting [13]. For example, if four out of six models (m_1 … m_6, as explained above) predict a module to be defective, the merged prediction becomes defective in return. Additionally, as a variant of the simple data approach, we compared our method with the prediction accuracy of a model when a training dataset is *randomly selected*. For instance, a model is randomly selected from model m_1 … m_6 to predict defects of CM1.

Test results accumulate in the middle of testing, and it can be regarded as a candidate of training data. For example, when there are 100 test target modules, and 50 modules have been tested, there are 50 test results which can be used for model training. Although it could be potentially treated as an arm in BA, we did not consider such data in this experiment (which is also similar to previous studies [6][8][12]).

Evaluation criteria: We used AUC and F1 score to evaluate the performance of each prediction model. Both are frequently used as a performance indicator in CPDP studies [4]. Prediction values are real numbers in most models. When we applied BA, the prediction values were converted into binary values (i.e., defective or not-defective) based on a cutoff value. We set the cutoff value as the closest point to top left corner of the ROC curve on a training project. Similarly, we also converted the prediction values derived by the baseline methods into binary values, when calculating AUC. Also, the converted prediction values were used to calculate F1 score.

Simulation of found defects during testing: To evaluate BA, we also consider missing (overlooking) defects during software testing. Typically, found defects after the software release are such missed defects. The study [5] reported that about 17% of defects are missed during integration test. That is, the miss could occur when the test result is non-defective (and defects might be found during system testing and after the software release) about 17% possibility. Therefore, and similar to [3], to simulate the miss, we randomly changed the evaluation of the evaluation as a TN (i.e., when prediction is defective) or a FP (i.e., when prediction is non-defective) at 20% probability when the modules are defective [a)].

V. Results and Discussion

A. Prediction Accuracy

Table I presents the AUC values of each CPDP method for each project and the average across the seven NASA projects. Similarly, Table II shows F1 score for the NASA datasets, and Table III and IV shows AUC and F1 score for the DAMB datasets. In these tables, the AUC value of shaded cells was lower than AUC of the BA (i.e., the prediction accuracy underperformed BA). We statistically compared criteria of BA with the three baseline methods (ensemble, data approach, and random project selection) using Wilcoxon signed rank test. The p-values of the test are shown in the tables.

The most effective training dataset was different for most test datasets. For instance, for the NASA dataset, PC3 was the most effective training dataset (in terms of AUC) for MW1, PC1, and PC4. When we used DAMB datasets, Mylyn was the most effective training dataset on JDT Core, PDE UI, and Lucene. Therefore, using a single project as a training dataset for every test dataset does not appear to be the most effective approach.

Regarding the average AUC values shown in Table I, the average AUC with the use of BA was the highest, followed by the data approach, ensemble method and random. The ranking of approaches (BA, data, ensemble, random) was the same in other the following Tables II, III and IV (for F1 score and AUC) as well. Next, we focus on the number of test datasets where a criteria of BA was larger than a baseline (i.e., shaded cells excluding 'avg.' column). The number of shaded cells was larger than half of the datasets on each method (row). For example, in Table III, when we compared BA with the simple data approach, although the difference of average AUC was very small (i.e. 0.001), AUC of BA was higher than the simple data approach on three to five projects.

Comparing the performance differences between BA and the baselines, most of the performance values were statistically significant at 0.05, except for AUC of the data approach (Table III). That is, BA attained higher prediction accuracy than the baselines in most cases. It was reported that CPDP models showed that, the median of AUC ranges between 0.65 to 0.75, and the median of F1 score ranges between 0.3 to 0.5 [4]. Thus, the AUC and F1 score results in our experiments are comparable to past studies.

a) We performed BA-based prediction and the simulation using code written in Python and Microsoft Excel VBA. The prediction models were built on R using the FSelector and pROC packages. For the replication, the scripts are published on https://zenodo.org/record/5057695#.YN5s4UxUu_i.

TABLE I. AUC of NASA datasets

method	CM1	MC2	MW1	PC1	PC2	PC3	PC4	avg.	p-value
ensemble	0.70	0.63	0.73	0.68	0.74	0.71	0.61	0.68	0.00
data	0.70	0.65	0.70	0.74	0.75	0.73	0.69	0.71	0.00
random	0.67	0.58	0.64	0.62	0.65	0.64	0.59	0.63	0.00
minimum	0.55	0.56	0.54	0.53	0.53	0.56	0.56	0.55	0.00
BA	0.74	0.67	0.78	0.69	0.78	0.72	0.72	0.73	-

TABLE II. F1 score of NASA datasets

method	CM1	MC2	MW1	PC1	PC2	PC3	PC4	avg.	p-value
ensemble	0.29	0.50	0.33	0.27	0.07	0.36	0.30	0.30	0.00
data	0.32	0.54	0.29	0.27	0.02	0.33	0.36	0.30	0.00
random	0.26	0.44	0.23	0.20	0.03	0.27	0.26	0.24	0.00
minimum	0.19	0.52	0.15	0.14	0.01	0.20	0.24	0.21	0.00
BA	0.31	0.60	0.32	0.27	0.04	0.37	0.39	0.33	-

TABLE III. AUC of DAMB datasets

method	JDT Core	PDE UI	Equinox	Lucene	Mylyn	avg.	p-value
ensemble	0.69	0.65	0.67	0.63	0.64	0.66	0.00
data	0.71	0.67	0.68	0.63	0.66	0.67	0.11
random	0.67	0.62	0.67	0.61	0.63	0.64	0.00
minimum	0.57	0.57	0.69	0.58	0.61	0.60	0.00
BA	0.64	0.67	0.71	0.67	0.67	0.67	-

TABLE IV. F1 score of DAMB datasets

method	JDT Core	PDE UI	Equinox	Lucene	Mylyn	avg.	p-value
ensemble	0.48	0.36	0.57	0.28	0.36	0.41	0.00
data	0.49	0.35	0.59	0.25	0.39	0.41	0.00
random	0.46	0.33	0.59	0.24	0.34	0.39	0.00
minimum	0.38	0.27	0.64	0.20	0.29	0.36	0.00
BA	0.57	0.41	0.63	0.28	0.42	0.46	-

These initial results suggest that BA works well in CPDP context as prediction accuracy was higher than the baseline methods. Note that these results do not suggest that BA-based defect prediction is superior to any other CPDP methods. Such a comparison of BA model with other methods is left for future work. The case this study makes is, if we use other CPDP methods as arms for BA, it would be expected to attain high prediction accuracy of BA prediction. The effect of BA might be impacted by the characteristics of the selected dataset, which is an issue that has been reported for other CPDP [4].

B. Prediction Tendency of BA

Comparison to a variant of ensemble: The prediction accuracy was slightly higher than the data approach. Intuitively, BA seeks the highest accuracy arm, and the prediction of the arm is the same as that of the data approach. Therefore, the highest accuracy of BA seems to be the same as of the data approach. However, the BA method does not select the best arm statically, but dynamically reselects an arm after the each module is tested. For instance, in Fig 1, although arm B predicts m1, m2 and m4 as defective, BA predicts m1, m3, m2, and m4 as defective. This indicates that BA-based defect prediction tends to predict more modules as "defective" than the baselines. To analyze the influence of the this tendency, we made a variant of ensemble method which aggressively predicts modules as *defective*. That is, if one or more models (e.g., model m_1 … m_6 explained in section IV) are predicted as *defective*, the method predicts all of them as *defective*. We call the method as *minimum voting*.

The prediction accuracy of the minimum voting is shown on the second row from the bottom of Table I to IV. AUC and F1 score values were the worst among the methods for both NASA and DAMB datasets. That is, minimum voting cannot improve prediction accuracy. Although BA tends to predict modules as *defective*, BA also tends to smartly select higher accuracy arms based on TS. This might positively impact prediction accuracy of BA.

Precision, recall, and TNR: Next, we analyzed the influence of the tendency to prediction accuracy in detail. To perform that, we focused on precision, recall, and TNR (true negative rate) of CPDP methods used in Section V.A.

Table V shows the average precision, recall and TNR values of the different methods in both used datasets. When we focus on minimum voting, although recall was higher than BA, precision was lower, and TNR was the lowest. This would be because minimum voting predicted too many modules as *defective*. In contrast, ensemble method lowers recall but improves TNR.

Compared with the baseline methods, precision of BA was almost the same on both datasets, and recall was greatly improved. This is the reason why F1 score of BA became higher than other methods. In contrast, TNR of BA was lower than other methods. Hence, AUC of BA became slightly better than the baselines. The results suggest that BA can improve recall, produces a consistent precision, and lowers TNR a bit.

TABLE V. AVERAGE OF PRECISION, RECALL, AND TNR OF PREDICTION MEDHODS

method	NASA			DAMB		
	precision	**recall**	**TNR**	**precision**	**recall**	**TNR**
ensemble	0.23	0.58	0.79	0.36	0.53	0.79
data	0.22	0.69	0.72	0.35	0.63	0.71
random	0.18	0.61	0.64	0.33	0.59	0.69
minimum	0.12	0.97	0.13	0.26	0.79	0.42
BA	0.21	0.88	0.57	0.36	0.73	0.61

VI. CONCLUSION

In this paper, we applied BA-based prediction to CPDP to evaluate the effect of BA in data selecting for CPDP. BA have been commonly used to help with a wider range of optimization problems. In our case, we use BA to build several candidates of prediction models based on several external projects and select the most suitable project from several candidates. Those different candidates are called the arms in BA.

In this primarily experiment using two datasets (i.e., NASA and DAMB) we built several logistic regression models using different set of projects as training. Using these models, we set arms of BA and the baselines based on simple data-driven approach. As a result, the average of AUC and F1 score of BA were higher than the baselines across the different datasets, and the performance differences were statistically significant in most cases. Additionally, compared with the baseline methods, precision of BA was almost same, and recall was greatly improved.

Experimental results suggest that BA works well in CPDP context. Although it does not imply BA-based defect prediction is superior to any other CPDP methods, BA-based method could have a good potentials when applied to CPDP. We focus mainly here on the selection of training data. However, these results are preliminarily in nature and further examination of the use of BA with a wider set of projects is needed. In the future we plan to compare this proposed BA based defect prediction to other CPDP methods, using the different prediction methods as arms in BA.

ACKNOWLEDGMENT

This research is partially supported by the Japan Society for the Promotion of Science [Grants-in-Aid for Scientific Research (C) and (S) (No.21K11840 and No. 20H05706).

REFERENCES

[1] M. D'Ambros, M. Lanza and R. Robbes, "An extensive comparison of bug prediction approaches," Proc. of Working Conference on Mining Software Repositories (MSR), pp.31-41, 2010.

[2] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," Proc. of International Conference on Neural Information Processing Systems (NIPS), pp.2249-2257, 2011.

[3] T. Hayakawa, M. Tsunoda, K. Toda, K. Nakasai, A. Tahir, K. Bennin, A. Monden, and K. Matsumoto, "A Novel Approach to Address External Validity Issues in Fault Prediction Using Bandit Algorithms," IEICE Trans. on Information and Systems, vol. E104.D, no. 2, pp.327-331, 2021.

[4] S. Hosseini, B. Turhan and D. Gunarathna, "A Systematic Literature Review and Meta-Analysis on Cross Project Defect Prediction," IEEE Transactions on Software Engineering, vol.45, no.2, pp.111-147, 2019.

[5] Information-technology Promotion Agency (IPA), Japan, The 2018-2019 White Paper on Software Development Projects, IPA, 2018 (in Japanese).

[6] Y. Kamei, T. Fukushima, S. Mcintosh, K. Yamashita, N. Ubayashi, and A. Hassan, "Studying just-in-time defect prediction using cross-project model,". Empirical Software Eng., vol.21, no.5, pp.2072-2106, 2016.

[7] M. Kondo, C. Bezemer, Y. Kamei, A. Hassan, and O. Mizuno, "The impact of feature reduction techniques on defect prediction models," Empirical Software Engineering, vol.24, no.4, pp.1925–1963, 2019.

[8] T. Kuramoto, Y. Kamei, A. Monden, and K. Matsumoto, "Fault-prone Module Prediction Across Software Development Projects - Lessons Learned from 18 Projects -," IEICE Transactions on Information and Systems, vol.J95-D, No.3, pp.425-436, 2012 (in Japanese).

[9] D. Sjøberg, B. Anda and A. Mockus, "Questioning software maintenance metrics: A comparative case study," Proc. International Symposium on Empirical Software Engineering and Measurement (ESEM), pp.107-110, 2012.

[10] A. Tahir, K. Bennin, X. Xiao, and S. MacDonell, "Does class size matter? An in-depth assessment of the effect of class size in software defect prediction," Empirical Software Engineering, vol.21, article no.106, 2021.

[11] B. Turhan, T. Menzies, A. Bener, and J. Stefano, "On the relative value of cross-company and within-company data for defect prediction," Empirical Software Engineering, vol.14, no.5, pp.540-578, 2009.

[12] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process," Proc. Joint Meeting of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering (ESEC/FSE), pp.91-100, 2009.

[13] Z. Zhou, Ensemble Methods: Foundations and Algorithms, Chapman and Hall/CRC, 2012.