Journal of Computers Vol. 29 No. 6, 2018, pp. 52-65
doi:10.3966/199115992018122906005

BTHYDRA+: Towards a Comprehensive Model to Conduct Cross-project Defect Prediction

Ning Huang[1], Jinglong Fang[2], Dan Wei[2], Bin Chen[2], and Xingqi Wang[2*]

[1] School of Computer Science, Hangzhou Dianzi University, ZheJiang Province,
925958391@qq.com

[2] Key Laboratory of Complex Systems Modeling and Simulation Ministry of Education,
School of Computer Science, Hangzhou Dianzi University, ZheJiang Province,
{fjl, weiwd, chenbin, xqwang}@hdu.edu.cn

Received 13 July 2017; Revised 28 November 2017; Accepted 9 January 2018

Abstract. Cross-project defect prediction (CPDP) is a field of study where researchers need to build a universal model by using within-project data to predict defects on other projects. However, variations in the distribution of source and target projects have an influence on the performance of classifiers. To enable effective cross-project defect prediction, we propose a comprehensive model containing data preprocessing and classifier transferring to construct a better classification space and strengthen the performance of classifiers. In preprocessing step, one baseline is calculated for every dataset from its non-defective samples based on the distance to all other non-defective samples and the data is transformed by using rank function. Genetic algorithm and ensemble learning are selected as the way in transferring step to extract effective representation from source projects and boost the capability of weak classifiers. We use Naive Bayes, Support Vector Machine and Classification and Regression Trees as classifiers and apply this model on five open resource projects (one Apache and four Eclipse projects) and NASA MDP dataset. Selecting accuracy as fitness in genetic algorithm improves the performance of classification. The model we proposed yields similar results and obtains higher precision comparing the within-project models. Meanwhile, it obtains better performance than the state-of-the-art methods on cross-project defect prediction. These results show that our model provides an opportunity to training classifiers by using more samples from different projects.

Keywords: classifier transferring, cross-project defect prediction, data preprocessing

1 Introduction

Software defect prediction (SDP) is a crucial process in the software development. It helps developers to find some potential defects in the sample, and reduce the cost of the entire project. In the past, many researchers [1] worked on improving the performance of defect prediction algorithm on local dataset. In recent years, more attention has been focused on how to handle cross-company or cross-project defect prediction [2]. The difference between these two researches is that organizations predict defects by their own historical data in within-project defect prediction (WPDP), while cross-project defect prediction (CPDP) needs to apply a model built from within-project data on other projects or even new releases of the same project. Zimmermann et al. [3] use defect prediction models learnt from one project on another, but the success rate is only 3.4%. So within-project data must be carefully preprocessed before being utilized globally.

* Corresponding Author

Generally, cross-project defect prediction is a challenging task since the models that is trained on one or a set of projects might not generalize well to other projects. The difficulty for building CPDP models is to solve variations of data in training and target projects [4]. Zhang et al. [5] consider two ways to overcome this challenge: (1) only use similar data from projects for training and testing [6], (2) transform data from the training project and the target project to the same distribution [7]. Meanwhile, Xia et al. [8] put forward a HYbrid moDel Reconstruction Approach (HYDRA) based on genetic algorithm and ensemble learning. However, these two methods either only have the function of preprocessing, or only focus on the performance of the classifiers. Therefore, we combine the advantages of preprocessing method and classifier transferring. A better comprehensive model is proposed to unify the distribution of data in different projects and strength the capability of classifiers.

In this study, we first propose a baseline transformation to address the variations in the distribution of training and target projects. Each project dataset has two classes of defect (minority) and non-defect (majority). One baseline is looked for among non-defective samples for every dataset, and used to transform corresponding data respectively. These baselines are required to be closest to the non-defective samples in other datasets. In other words, they are some standards of normal module that are most likely to be recognized by other projects in its own project data. So different project data can be pulled back to a same level by preprocessing with using these standards. Then we improve HYDRA model on threshold generation and use it to predict those samples from target projects after transformation. Threshold is generated randomly in Xia's article. This method may cause the entire GA phase to crash, because almost *Comp(j)* are smaller than threshold. Generation delay helps to evaluate range of *Comp(j)* in advance, which not only avoids the calculation problem of F-measure, but also reduce the cost during searching threshold. Moreover, fitness selection is studied for find a way to promote the capability of classifiers on imbalance problem. Both Accuracy and F-measure are selected as Fitness to measure performance of parameters.

We apply our approach to five open source projects (Eclipse, Equinox, Mylyn, Lucene, and PDE), which is also the dataset used by Zhang et al. [5] in their research. In addition to Naive Bayes (NB), Support Vector Machine (SVM) and Classification and Regression Trees (CART) are added to evaluate the performance of our model on different classifiers. Besides Within model, we compare our model against seven cross-project prediction methods [3, 5, 9-13]. The results show that (1) in the contrast between cross-project and within-project defect prediction, our models yields comparable F-measure and have a few drawback on accuracy (lower than 1%-6%), but far from within-project model on recall (lower than 14%); (2) in the comparison of cross-project preprocessing methods, the accuracy and F-measure of our model are higher than others'; (3) utilizing accuracy as a fitness increases the accuracy of classification without decrease of F-measure. For examining the generalizability of our method, we apply it to NASA MDP dataset. The results provides a similar performance as we mentioned before. Additionally, SVM is considered as an efficient classifier in CPDP rather than NB or CART because it has good ability of non-linear classification and excellent robustness. In summary, the major contribution of our study is providing a comprehensive model combining preprocessing method and classifier transferring to address the problem existing in cross-project defect prediction. This thought enables us to build a universal model for different classifiers on a large set of projects.

The remainder of this paper is organized as follows. Section 2 summarizes the related work in the field of CPDP. Section 3 describes our approach and Section 4 presents the results and discussions of experiments. We conclude and provide insights for future work in Section 5.

2 Related Work

Software defect prediction is an effective way to detect defective samples or documents in software system. It builds a model by historical data from the same software or project to predict label of test data. However, it is rare that sufficient historical data is available for a new project, but there is plenty of data from other projects. Thus, more attention has been focused on how to effectively utilize the data from other projects. This approach provides a new research direction in defect prediction, which is called cross-project defect prediction (CPDP). The aim of CPDP is to build a model from other projects to successfully predict defects in their own projects. Usually, simply applying models from training projects on target projects does not lead to accurate prediction. In Fig. 1, some methods that can be used in CPDP are collected and divided into three categories: Original, Data Preprocessing and Classifier Transferring.

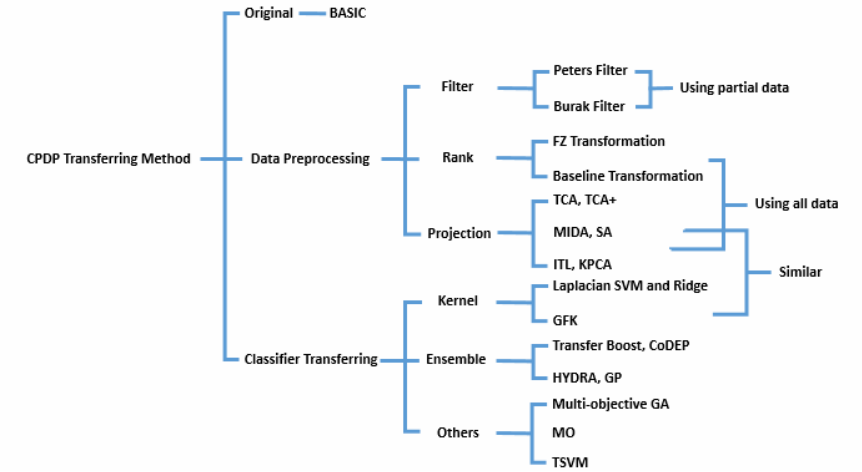


Fig. 1. Methods to solve cross-project defect prediction

Zimmermann et al. [3] run 622 cross-project predictions on 12 real-world applications. They do not using any transferring method, but only make one-to-one classification (BASIC)[1]. Decision trees are used to identify the factors (i.e., process, code data and domain) that do influence the success of cross-project prediction. They conclude that cross-project prediction is a serious problem, that is, simply using projects in the same domain does not work to build accurate prediction models. Process, data and domain need to be quantified, understood and evaluated before prediction models are built and used.

Peters et al. [11] consider that when local data is scarce, more information exists in other projects. Accordingly, they improve Burak filter (a state-of-the-art relevancy filter) [14] and propose Peters Filter to solve the problem existing cross-project predictions. This filter selects training data via the structure of other projects. To assess the performance of the Peters filter, they compare it with two other approaches for quality prediction. Within-company learning and cross-company learning with the Burak filter. The results show that: (1) within-company predictors are weak for small datasets; (2) the Peters filter + cross-company builds better predictors than both within-company and the Burak filter + cross-company; and (3) the Peters filter builds 64% more useful predictors than both within-company and the Burak filter + cross-company approaches.

However, Peters filter only uses similar data from projects for training and testing. It is improved when Feng Zhang et al. [5] propose their context-aware rank transformations. It is a preprocessing method that can transform all of data from training projects and target projects to the same distribution. They cluster projects based on the similarity of the distribution and derive the rank transformations using quantiles of predictors for a cluster. Then they fit the universal model on the transformed data of 1,398 open source projects hosted on SourceForge and GoogleCode. The results suggest that a universal defect pre-diction model may be an achievable goal.

Finding a better classification space is regarded as a main method to overcome challenge from difference between source project and target project. Domain adaptation solves this problem in a target domain by utilizing the training data in a different but related source domain. TCA [9], TCA+ [15], MIDA [16], SA [17], ITL [18] and KPCA [10] are used for searching space according some criteria and projecting the data. TCA is one of most common method proposed by Pan et al. [9]. It tries to learn some transfer components across domains in a Reproducing Kernel Hilbert Space (RKHS) using Maximum Mean Discrepancy (MMD). As a result, standard machine learning methods can be applied in the source domain for use in the target domain. They verify effectiveness and efficiency of TCA by experiments on two real-world applications: cross-domain indoor WiFi localization and cross-domain text classification.

Kernel function [19] in classifiers also perform similar role in searching space, but the difference to projection is that it can only combine with some certain classifier like SVM. Belkin et al. [12] propose a family of learning algorithms (i.e., LapSVM and LapRR) based on a new form of regularization. They focus on a semi-supervised framework that incorporates labeled and unlabeled data in a general-purpose learner and also use properties of Reproducing Kernel Hilbert Space (RKHS) to prove new Representer theorems that provide theoretical basis for the algorithms. The experiments suggest that these semi-

[1] We refer to Zimmermann et al.'s approach as BASIC in this section

supervised algorithms are able to use unlabeled data effectively, which corresponds to application scenarios of cross-project prediction.

Meanwhile, some ensemble learning methods are applied like Voting in CoDEP [20], GP [21] and Adaboost in TransferBoost [22]. HYbrid moDel Reconstruction Approach (HYDRA) with using Adaboost is proposed by Xia et al. [8] performs well on cross-project defect prediction. Experiments on 29 datasets from the PROMISE repository with using logistic regression as the underlying classification algorithm are conducted. Their approach obtain better F-measure than TCA+, Peters filter, GP, MO, and CODEP. In addition, HYDRA on average can discover 33 percent of all bugs if developers inspect the top 20 percent lines of code. Moreover, HYDRA also can improve the F-measure of Zero-R. Other method like MO [23] and TSVM [13] also can be applied on cross-project prediction.

Honestly, there are some drawbacks in these studies. The filter removes a large number of samples, which will lose a lot of useful information. This method is suitable for the project possessing many samples. And only in this way, the shortcoming of filter can be hidden. The theoretical basis of rank method suggests is not sufficient, and this method only has preprocessing function. As for projection, different algorithms have different criteria. It is difficult to select a universal criterion for all the projects. Meanwhile, the projection does not utilize the information of labels very well. Kernel method are similar with projection method to some extent, but they are often applied to a certain classifier. In addition, ensemble learning can largely improve the performance of the weak classifiers. But the time and space complexity are greatly increased, and parameters (e.g., number of iteration) need to be optimized in the process.

However, the comprehensive model we proposed can effectively avoid the above shortcomings. The comprehensive model takes data preprocessing and classifier transferring into account. It adopts the baseline transformation to make full use of all data information including labels, and then ensemble the weak classifiers with hybrid model reconstruction approach to strengthen generalization ability. Moreover, we improve the process in the parameter searching step, and reduce the time complexity of HYDRA. Besides, the interface of different classifiers is provided in the model, which solves the problem that only one classifier is used in kernel methods.

3 Approach

3.1 Overall Architecture

The overall framework of our model is divided into two steps: preprocessing step and classification step, which is shown in Fig. 2. In preprocessing step, we first unify metrics of source projects and target project to ensure that all samples are under the same feature space. Then baseline transformation is used to reduce the variation of the distribution of data from different projects by utilizing the most representative samples to rank projects. In the classification step HYDRA+ algorithm combining different classifiers is used for cross-project defect prediction. This algorithm contains GA phase and EL phase. GA phase is aimed at finding the samples that can facilitate the prediction from the source projects, while EL phase is helpful to enhance the performance of the weak classifiers.

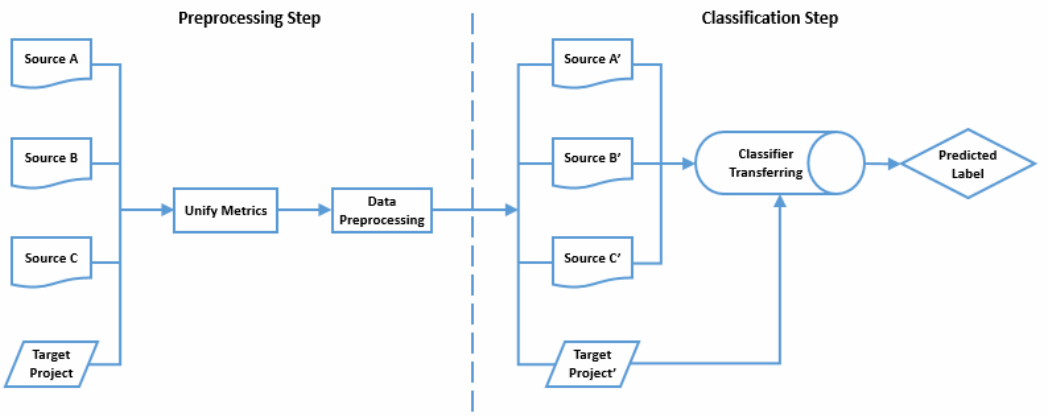


Fig. 2. Overall architecture of BTHYDRA+

The phase of unifying metrics is claimed briefly in section 4. In this section, we mainly discuss about baseline transformation and the improvements on HYDRA.

3.2 Baseline Transformation

The aim of transformation in cross-project defect prediction is to make it easier for classifiers to find a standard between defective and non-defective samples. If we find a baseline b in non-defective samples of a project P, and it can be recognized by other projects, then b will be the most universal sample in P for classification standard. By ranking all the samples in P according to b, the variations of the distribution among different projects can be solved. This process is simply described in Fig. 3.

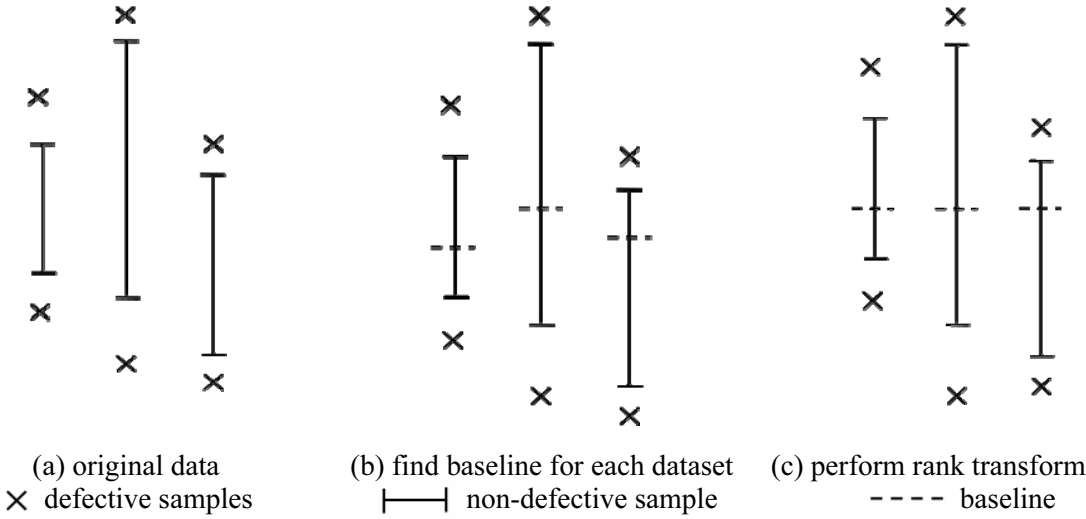


Fig. 3. Schematic diagram of baseline transformation

Fig. 3(a) is original data from three projects. In Fig. 3(b), three baselines are found among non-defective samples, and these three projects are transformed by them in Fig. 3(c). Defective samples are divided into the same level with the change of mutual relation among these projects, so classifiers can training model better with less cost.

We define $D=\{D_1, D_2, ..., D_L\}$ $k=1...L$ as L datasets, where D_k^+ represents defective samples in the kth dataset, while D_k^- represents non-defective samples in the kth dataset.

For each dataset, we can find a sample from its non-defective samples as baseline:

$$\begin{aligned} baseline_k &= min\sum_{l=1}^{L} \| D_{k_i}^- - D_l^- \|, \quad l=1...L \, and \, l \neq k, i=1...n_k \\ &= \min_{t \in D_k^-} \sum_{l=1}^{L} \| t - D_l^- \| \end{aligned} . \tag{1}$$

The sum of this sample to all non-defective samples in the other datasets is the smallest.

For each feature in datasets, it can be transformed by using rank function:

$$R(D_{(i,j)}) = B + Q(baseline_j) - Q(D_{(i,j)}) . \tag{2}$$

where the $Q(D_{(i,j)})$ is the tenth quantile of the i^{th} sample on the j^{th} feature. B is a base value, which ensures that the features after transformation are positive. Because tenth quantile is used, so we set B to 10. For example, there are 10 samples whose values of the j^{th} feature are: 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10 respectively. If the 7th sample is considered to be the baseline, the values of transformation is 4, 5, 6, 7, 8, 9, 10, 11, 12, 13.

3.3 HYbrid moDel Reconstruction Approach+

HYDRA [8] contains genetic algorithm (GA) phase and ensemble learning (EL) phase. In the GA phase, *(N+1)* classifiers are built for each source project S_i and training target data T_t. Genetic algorithm (GA) is used to search for the best composition of these classifiers. In the EL phase, multiple GA classifiers are built by running the GA phase multiple times, and these GA classifiers are composed according to Adaboost algorithm.

However, there are some faults in HYDRA. For example, F-measure cannot be calculated in $(N+1)^{th}$ classifier because randomly selecting samples in T_t, and β_k should be decreasing function of ε_k rather than increasing function. We propose generation delay of *threshold* in GA phase to solve the problem that almost *Comp(j)* are smaller than *threshold*.

Consider *N* source projects *{S_1; S_2; . . . ; S_N}*, and a training target data T_t. We build *(N+1)* classifiers from the source projects and the training target project. A GA classifier composes these *(N+1)* classifiers and assigns a label to an instance *j* as follows:

$$Label(j) = \begin{cases} 1 \ (i.e., defective), \ if \ Comp(j) \geq threshold \\ 0 \ (i.e., \ clean), \quad Otherwise, \end{cases}$$

where,

$$Comp(j) = \frac{\sum_{i=1}^{N+1} \alpha_i \times Score_i(j)}{LOC(j)} \quad . \tag{3}$$

Where $Score_i(j)$ is the likelihood score outputted by the i^{th} classifier for instance *j*, α_1 to α_{N+1} are the weights of the *(N+1)* classifiers. The *threshold* is the boundary used to decide whether an instance is defective or not, and *LOC(j)* is the number of lines of codes for instance *j*. Instance *j* will be classified as defective (i.e., y=1) if its composite score *Comp(j)* is larger than or equal to *threshold*; otherwise it is classified as clean. Note that α_1 to α_{N+1} and threshold are the parameters of a GA classifier. Thus, a GA classifier is denoted as ($\sum_{i=1}^{N+1} \alpha_i M_i$, *threshold*) where each M_i is a classifier, α_i is the weight of M_i, and *threshold* is the defect boundary. The searching space of all possible compositions corresponds to the various assignments of values to the weights $\{\alpha_1, \alpha_2, ..., \alpha_{N+1}\}$, and the defect boundary *threshold*. Each weight is a real number from zero to one.

Since the range of *comp (j)* is usually in [0, 1], almost *comp (j)* are greater than the *threshold* if the *threshold* and *alpha* are generated randomly at the same time. Meanwhile, the test samples are judged as defective modules, which will cause Fitness (F-measure) cannot be calculated and GA crash. We put forward that the *alpha* should be generated first, and then delay the generation of *threshold* after calculating the range of *comp (j)*. As formula (4) shown, it can effectively avoid the problem we mentioned before. Besides, the *threshold* is in the range of *comp (j)*, so F-measure is also easier to achieve the optimum solution, which largely save the cost in searching parameters.

$$threshold = rand() \cdot (\max C - \min C) + \min C \quad . \tag{4}$$

where,

$$\min C = \min(Comp(j)) \quad \max C = \max(Comp(j))$$

On the other hand, the selection of Fitness indicator in GA is also studied. Fig. 4 present the discrepancy on the weight in confusion matrix by selecting different indicator.

		True class	
		Positive	Negative
Predicted class	Positive	0.25	0.25
	Negative	0.25	0.25

(a) original

		True class	
		Positive	Negative
Predicted class	Positive	0.3	0.3
	Negative	0.3	0.1

(b) F-measure

		True class	
		Positive	Negative
Predicted class	Positive	0.4	0.1
	Negative	0.1	0.4

(c) accuracy

Fig. 4. Weight in confusion matrix by selecting different indicator

The concentration is divided equally in the original situation, while it on true negative class will be transferred when using F-measure as Fitness because precision and recall are emphasize the successful prediction on positive class. However, the amount of negative class is much more than positive class in test samples, so accuracy is selected as an alternative to F-measure in order to explore optimum performance of HYDRA+.

In HYDRA+, interfaces are provide for different classifiers, re-generation step is added to increase the adaptability of parameters to various projects and execution sequence is modified when $\varepsilon_k < \frac{1}{2}$ in EL phase.

4 Experiments

4.1 Preparation

Some datasets from the real project are usually used in software defect prediction experiment. Each dataset contains a number of software samples, which are characterized by static code metrics (e.g., Halstead [24-25], McCabe [26] of the method level) and they are converted into numerical samples. However, these datasets cannot be used directly when we make cross-project experiments, because cross-software defect prediction is a binary classification problem with the same task and same feature space in training and target projects, which is defined by transfer learning.

Different from the traditional learning model, the dataset needs to be preprocessed to ensure the same feature space in the cross-project perdition model. We simply claim the process of unifying metrics step in Fig. 5. This paper use five open source projects (i.e., one Apache project: Lucene, and four Eclipse projects: Eclipse, Equinox, Mylyn, and PDE) for the comparison on cross-project defect prediction. The feature has been integrate by Zhang et al. [5], so we only list basic information of these projects in Table 1.

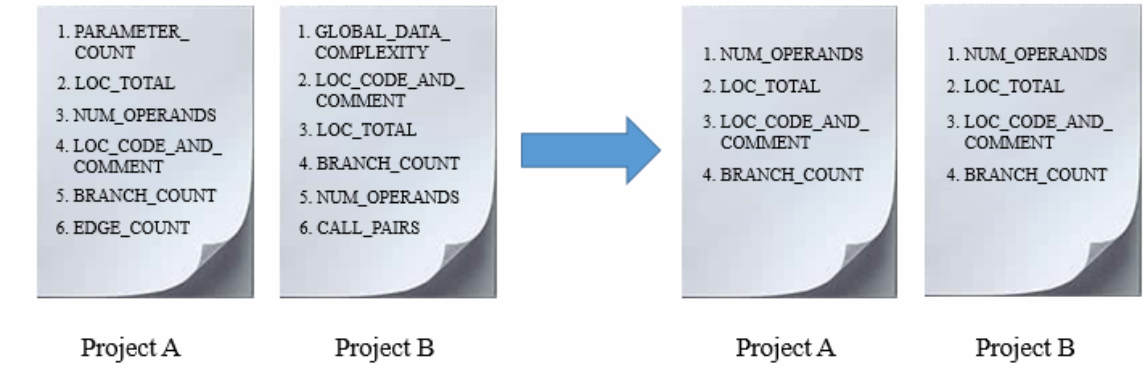


Fig. 5. Process of unifying metrics

Table 1. Open source projects description

	Eclipse	Equinox	Mylyn	Lucene	Pde
number of features	18	18	18	18	18
number of samples	997	324	1862	691	1497
number of defective samples	206	129	245	64	209
percentage of defective samples	20.66%	39.81%	13.16%	9.26%	13.96%

For examining the generalizability of methods, we apply them to NASA MDP dataset, which is also used by many researchers as general data to compare their algorithms, such as B. Turhan [27] and L. Guo [28] used Naive Bayes and Random Forest on it to predict defects respectively. This dataset contains more projects and samples, and it is listed in Table 2.

Table 2. NASA MDP dataset description

	CM1	JM1	KC1	KC3	MW1	MC1
number of features	37	21	21	39	37	38
number of samples	344	9593	2096	200	364	9277
number of defective samples	42	1759	325	36	27	68
percentage of defective samples	12.21%	18.34%	15.51%	18.00%	7.42%	0.73%
	MC2	PC1	PC2	PC3	PC4	PC5
number of features	39	37	36	37	37	38
number of samples	127	759	1585	1125	1399	1711
number of defective samples	44	60	16	140	178	471
percentage of defective samples	34.65%	7.91%	1.01%	12.44%	12.72%	27.53%

As the Table 2 shows, the feature spaces of these datasets are not exactly identical, so 20 common features are selected in NASA MDP dataset. After filtering some features, experiments are carried out for studying following three questions: (1) The performance of fitness selection and baseline transformation; (2) The differences between cross-project model and within-project model; 3) The comparison between cross-project transformation methods.

In cross-project defect prediction, leave-one-out method is used. It is a method that selecting one project as test set and the remaining is training set. The validity of this method is proved in Xia's article [8]. Meanwhile, in within-project defect prediction, ten-fold validation method [29] is used to evaluate the performance of classifiers. NB [30], SVM [31] and CART [32] are selected as classifiers in our experiment. Accuracy, precision, recall and F-Measure, which are calculated by confusion matrix, are selected as the performance index.

4.2 Evaluation of Fitness Selection and Baseline Transformation

The experiment is carried out through two groups. Both F-measure and accuracy is selected as fitness in each group, while the difference between this two groups is whether the model contains baseline transformation. Because of the use of generation delay about *threshold* in HYDRA+, the parameters like *PopSize*, *MaxGen* and *K* are set to 10, 10 and 5 respectively, which is much smaller than HYDRA's.

The results on five open source projects and NASA MDP dataset are shown in Table 3 and Table 4.

Table 3. Comparison of four classification model on five open source projects

	SVM				NB				CART			
	acc	Pre	rec	F	acc	pre	rec	F	acc	pre	rec	F
F_HYDRA+	71.49	65.15	31.11	41.27	68.80	67.92	30.13	39.93	64.92	61.73	25.77	35.12
A_HYDRA+	74.06	62.83	32.59	41.28	71.87	52.62	28.28	31.43	68.05	56.55	29.94	35.53
F_BTHYDRA+	74.65	23.99	40.96	18.24	78.91	38.08	36.30	36.03	71.61	49.23	31.69	34.79
A_BTHYDRA+	77.87	47.08	30.09	36.41	79.08	45.91	23.11	29.42	72.41	53.06	31.72	37.66

Table 4. Comparison of four classification model on NASA MDP dataset

	SVM				NB				CART			
	Acc	pre	rec	F	acc	pre	rec	F	acc	pre	rec	F
F_HYDRA+	72.10	58.07	22.14	30.81	59.69	76.04	19.30	28.67	68.29	55.86	19.02	24.70
A_HYDRA+	74.64	57.03	21.46	29.41	63.49	73.55	19.11	28.44	69.12	54.11	18.15	23.97
F_BTHYDRA+	78.63	30.02	25.89	25.89	63.56	74.50	17.60	25.82	72.07	52.94	18.60	24.11
A_BTHYDRA+	79.09	34.75	18.63	21.39	73.19	53.54	22.87	32.12	72.28	51.18	22.17	26.91

From the comparison between these four models in Table 3 and Table 4, we find that: (1) Baseline transformation can improve the accuracy of classification because variation of distribution from different

projects is reduced by using this preprocessing method. It makes the classifiers collect more effective training samples in another feature space. (2) Utilizing accuracy as a fitness can increase the accuracy of classification without decrease of F-measure. It might because there are some random values in the HYDRA, which interfere with the results of classifiers. So the specific application scenarios should be referred to select indicators like cost-sensitive function [33] as fitness in GA phase. (3) The classifiers obtain similar performance on five open source datasets and NASA MDP dataset, which indicates that our models can be applied to predict defects on the data with different size. 4) The precision is generally higher than the recall because the software defect prediction is an imbalance binary classification problem with many none-defective (negative) samples, but a few defective (positive) samples in the training set. This phenomenon makes the classifier more difficult to capture defective representation. Thus, it is easier to judge positive samples as negative samples compared to the opposite situation.

4.3 Comparison with the Within-project Model

We apply the within-project model on Eclipse, Equinox, Mylyn, Lucene, Pde, and NASA MDP dataset, respectively. And it is compared with our cross-project model, the results are listed in Table 5 and Table 6.

Table 5. Within-project prediction on five open source projects

	SVM				NB				CART			
	Acc	pre	rec	F	acc	pre	rec	F	acc	pre	rec	F
Eclipse	79.09	0.00	NaN		82.42	31.53	68.42	43.17	80.81	47.50	52.86	50.04
Equinox	66.25	97.47	53.90	69.42	73.75	44.48	81.87	57.64	72.81	66.81	65.25	66.02
Mylyn	86.88	2.39	54.42	4.57	83.82	32.42	36.64	34.40	82.10	28.53	30.33	29.40
Lucene	90.43	0.00	NaN		87.10	38.77	33.43	35.90	88.41	28.80	34.33	31.33
Pde	85.91	0.00	NaN		82.89	27.82	35.17	31.07	81.07	30.50	31.86	31.16
Avg	81.71	19.97	54.16	36.99	82.00	35.01	51.10	40.44	81.04	40.43	42.93	41.59
F_BTHYDRA+	74.65	23.99	40.96	18.24	78.91	38.08	36.30	36.03	71.61	49.23	31.69	34.79
A_BTHYDRA+	77.87	47.08	30.09	36.41	79.08	45.91	23.11	29.42	72.41	53.06	31.72	37.66

Table 6. Within-project prediction on NASA MDP dataset

	SVM				NB				CART			
	acc	pre	rec	F	acc	pre	rec	F	acc	pre	rec	F
CM1	83.82	0.00	NaN		85.00	31.50	31.35	31.42	82.94	23.86	31.90	27.30
JM1	81.99	6.08	64.03	11.11	81.38	19.57	46.73	27.59	76.40	33.03	34.89	33.94
KC1	84.11	3.55	47.35	6.61	82.39	34.80	38.00	36.33	82.49	39.42	45.50	42.24
KC3	79.50	0.00	NaN		76.00	29.45	42.96	34.95	78.50	37.76	38.33	38.05
MC1	99.57	44.79	71.17	54.98	94.64	28.77	4.33	7.52	99.29	37.38	60.74	46.28
MC2	68.33	0.00	0.00		75.83	42.00	56.67	48.24	64.17	56.00	50.88	53.32
MW1	90.38	0.00	NaN		76.54	50.74	23.35	31.98	88.08	41.17	50.00	45.16
PC1	92.40	0.00	0.00		90.53	33.53	38.75	35.95	89.07	32.78	45.17	37.99
PC2	99.18	0.00	NaN		95.44	5.83	2.26	3.26	98.29	0.00	0.00	
PC3	89.29	9.92	100.00	18.05	23.39	92.01	13.51	23.56	83.30	35.73	30.30	32.79
PC4	86.04	1.32	16.67	2.45	89.21	24.88	64.23	35.87	87.91	53.21	53.49	53.35
PC5	72.75	5.81	74.33	10.77	73.39	20.08	57.01	29.70	71.64	44.85	49.79	47.19
Avg	85.61	5.96	46.69	17.33	78.65	34.43	34.93	28.86	83.51	36.27	40.92	41.60
F_BTHYDRA+	78.63	30.02	25.89	25.89	63.56	74.50	17.60	25.82	72.07	52.94	18.60	24.11
A_BTHYDRA+	79.09	34.75	18.63	21.39	73.19	53.54	22.87	32.12	72.28	51.18	22.17	26.91

Note. Precision is 0 because TP is 0 and recall is NaN because FN is 0, which cause F-measure cannot be calculated.

Table 5 and Table 6 present that the within-project defect prediction has an advantage over other indicators except precision. According to the formula of the index, we conclude that this may be due to the decrease in TN and the increase in FN, which is also reported by Turhan et al. [34]. Moreover, there is little difference between the accuracy of our model and within-project prediction. In addition, the F-measure is promoted when using our model combining SVM and NB on NASA MDP dataset.

4.4 Improvements and Comparison Among Cross-project Models

Our models are compared with prior cross-project defect prediction approaches including: BASIC [3], FZ [5], TCA [9], KPCA [10], Pfilter [11], LapSVM [12], and TSVM [13], which ensures at least one algorithm is selected in each class of transferring method. For LapSVM[2] and TSVM[3], we use the source code provided by the authors.

We re-implement BASIC, FZ, TCA, KPCA and Pfilter through MATLAB. Notice that there is no context factors like programming language in NASA MDP dataset when we using FZ. Thus, the features with large variance is selected, and segmentation step is implemented through distance to the mean of that features. *RFB* is used as a kernel function in TCA because *RBF* is better than Linear in the Sinno's experiment on WiFi data, and we set *grammar* to 10 in RBF function. Similarly we also use *RBF* as kernel function for KPCA. In addition, the data is post-processed by PCA in TCA and KPCA, which may potentially decompose the information. The number of dimension after PCA is set to 9 and 10 in the NASA MDP dataset and the five open source projects respectively. It is worth noting that MATLAB cannot save more than 10000×10000 matrix when calculating RBF kernel, so in the experiment on NASA MDP dataset, 10,000 samples are randomly selected as training set when using TCA and KPCA. The results of comparison are listed in Table 7 and Table 8.

Table 7. Comparison of different cross-project classification models on five open source projects

	SVM				NB				CART			
	acc	pre	rec	F	acc	pre	rec	F	acc	pre	rec	F
Within	81.71	19.97	54.16	36.99	82.00	35.01	51.10	40.44	81.04	40.43	42.93	41.59
Basic	70.33	20.33	60.35	25.34	61.74	56.05	37.95	41.34	61.06	35.12	30.97	29.38
FZ	79.08	12.03	38.79	18.37	55.19	63.57	35.01	45.15	65.31	31.64	29.37	30.46
F_HYDRA+	71.49	65.15	31.11	41.27	68.80	67.92	30.13	39.93	64.92	61.73	25.77	35.12
A_HYDRA+	74.06	62.83	32.59	41.28	71.87	52.62	28.28	31.43	68.05	56.55	29.94	35.53
F_BTHYDRA+	74.65	23.99	40.96	18.24	78.91	38.08	36.30	36.03	71.61	49.23	31.69	34.79
A_BTHYDRA+	77.87	47.08	30.09	36.41	79.08	45.91	23.11	29.42	72.41	53.06	31.72	37.66
TCA	77.33	20.33	60.35	25.34	70.61	25.65	30.64	26.78	71.37	35.47	30.85	29.29
KPCA	73.69	15.79	66.85	24.60	76.81	18.04	54.78	27.42	75.79	28.55	32.72	28.81
Pfliter*					19.37	100.00	19.37	31.18	19.05	100.00	19.05	30.42
LapSVM	65.92	73.33	17.16	28.18								
TSVM	75.78	42.99	23.13	30.07								

Note. Pfliter select all training samples with the same label on five open source projects, so SVM cannot be trained.

Table 8. Comparison of different cross-project classification models on NASA MDP dataset

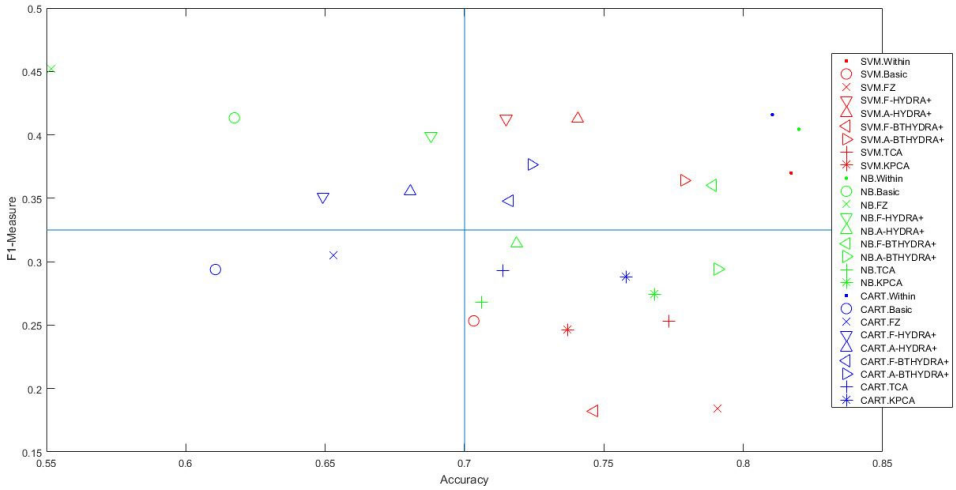
	SVM				NB				CART			
	acc	pre	Rec	F	acc	pre	rec	F	acc	pre	rec	F
Within	85.36	5.99	63.16	20.13	78.93	40.83	31.40	30.99	84.24	33.37	39.80	39.37
Basic	75.66	3.94	42.10	8.69	61.67	74.49	22.01	31.44	64.93	27.59	22.29	20.79
FZ	82.43	1.64	34.40	3.12	61.40	46.24	18.66	26.59	67.69	18.05	21.27	19.53
F_HYDRA+	72.10	58.07	22.14	30.81	59.69	76.04	19.30	28.67	68.29	55.86	19.02	24.70
A_HYDRA+	74.64	57.03	21.46	29.41	63.49	73.55	19.11	28.44	69.12	54.11	18.15	23.97
F_BTHYDRA+	78.63	30.02	25.89	25.89	63.56	74.50	17.60	25.82	72.07	52.94	18.60	24.11
A_BTHYDRA+	79.09	34.75	18.63	21.39	73.19	53.54	22.87	32.12	72.28	51.18	22.17	26.91
TCA	78.33	20.33	60.35	25.34	67.76	34.98	23.72	14.31	68.92	25.98	15.71	17.72
KPCA	74.69	13.52	53.90	22.46	62.37	28.72	42.79	34.10	65.02	21.39	24.08	18.93
Pfliter	63.35	46.08	20.39	23.39	79.34	80.82	18.53	30.66	68.06	44.68	22.04	25.29
LapSVM	62.73	74.74	15.27	26.38								
TSVM	71.21	39.23	26.43	31.59								

Table 7 and Table 8 prove that our model is generally dominant on three classifiers comparing with other methods. This may be attributed to following two points: (1) there is a preprocessing method that can find a better feature space for the classifiers. Baseline transformation draws on the FZ's point of view

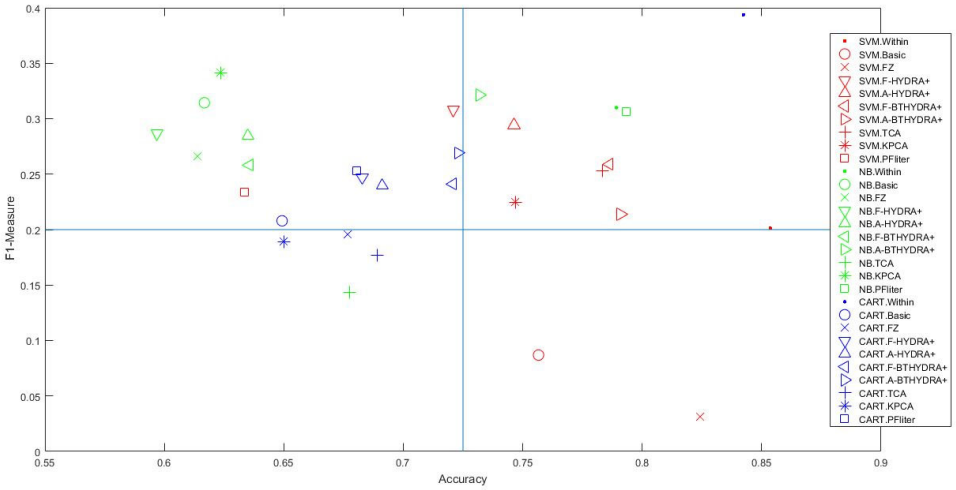
[2] https://github.com/tknandu/LapTwinSVM/tree/master/Primal_LapSVM/lapsvmp_v02
[3] http://svmlight.joachims.org/ and https://github.com/sods/svml

with using all of data. Oppositely, Pfliter only selects those training samples that are closest to the target project. In addition, compared to TCA and KPCA, BT play a better role in promoting because the label information is used. (2) When using HYDRA+ as transferring method on classifier, not only genetic algorithms is used to find from the target project in the classification of effective features, but also different weights on each GA classification model are given by AdaBoost based on the predicting error. Meanwhile, there are several classifiers like SVM in each GA classification model. The advantage on the amount of classifiers makes results better, which is similar to the function of deeper network in deep learning, even if HYDRA+ does not possess advanced transferring methods like LapSVM and TSVM.

Accuracy was often regarded as the main indicator to measure the performance of classifiers in the past, but Wang et al. [35] hold that there are some deviations when using accuracy to measure an imbalance problem. This is because an overall accuracy of 99% can be attained by a binary classifier that classifies all data samples as majority class [32]. Fortunately, F-measure is a trade-off between precision and recall, which is helpful to calibrate results from evaluation. It follows the setting used in many defect finding studies [31]. Fig. 6 are used in order to clearly observe the trend of different methods on these two indicators. Different colors are used to distinguish different classifiers, and the triangle with different orientations represents the models our proposed. It should be noted that LapSVM and TSVM are not drawn because they can only be applied to SVM, and we omit Pfliter on the five open source projects because there are some bugs when it filters data.



(a) Performance of different transferring methods on five open source projects



(b) Performance of different transferring methods on NASA MDP dataset

Fig. 6. Different transferring methods on accuracy and F-measure

Blue line is used to divide the area into four parts. The upper right part represents better performance with high accuracy and high F-measure. In Fig. 6, we find that: (1) Compared to NB and CART, SVM is more suitable to be a classifier for cross-software defect prediction, since SVM has good ability of non-linear classification and excellent robustness. (2) Our models are generally better than others and their results are closest to the Within's, especially on five open source projects. (3) The models we proposed are suited to different classifiers. (4) The performance of NB and CART on open source projects and NASA MDP datasets are different. The reason causes this phenomenon is that there are large number of samples in training step in NASA MDP dataset with using leave-one-out cross validation in the experiment, which lead to the situation of underfitting and overfitting when complex problems need to be solved by NB and CART (e.g., NB + Pfilter perform well on NASA MDP dataset because Pfilter prevent NB from underfitting by largely reducing the samples). In summary, robust classifiers like SVM should be chosen in cross-project defect prediction, especially in the case of using many samples from source projects.

5 Conclusion

In this study, we attempt to use a comprehensive model containing data preprocessing and classifier transferring to solve variations of data in cross-project defect prediction. We first propose a baseline transformation method to make distribution of different projects more similar. We then use the genetic algorithm and ensemble learning in HYDRA, and put forward generation delay of parameters to reduce cost during searching threshold. By comparing with the original HYDRA, we find that baseline transformation is conducive to promote performance of classifier. Moreover, we apply within-project models on five open source project and NASA MDP dataset, and further find that our cross-project model has higher precision than within-project models. Because of the underperformance on accuracy, we improve it by using accuracy as Fitness in genetic algorithm. We also evaluate the performance of our models with a series of state-of-the-art approaches. The results show that our models generally better than others on different classifiers.

However, there still exist several unsolved problems including experimental method and difference between research and reality. For instance, leave-one-out method is considered to be an effective way to detect the reliability of cross-project software defect prediction. But the correlation among datasets may have an influence on the final results (e.g., when PC1 is selected as test set, PC2-5 in training sets can improve the performance of the prediction). In the future study, we plan to remove the relevant data set and then experiment. Meanwhile, although the experiment on NASA MDP dataset is added to demonstrate the generality of our method, it is unclear if the universal model also performs well on real software development projects. We attempt to test our method on projects provided by developers.

Acknowledgments

This research work was supported by ZheJiang Provincial Natural Science Foundation of China under Grant Nos. LQ16F020006, LY17F020023 and LQ17F020003, and the Defense Industrial Technology Development Program of China under Grant Nos. JCKY2016415C005 and JSZL2016415B002, ZheJiang Provincial Science and Technology Department Foundation of China under Grant No.2015C33003.

References

[1] T. Hall, S. Beecham, D. Bowes, D. Gray, S. Counsell, A systematic literature review on fault prediction performance in software engineering, IEEE Transactions on Software Engineering 38(6)(2012) 1276-1304.

[2] F. Peters, T. Menzies, L. Gong, H. Zhang, Balancing privacy and utility in cross-company defect prediction, IEEE Transactions on Software Engineering 39(8)(2013) 1054-1068.

[3] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, B. Murphy, Cross-project defect prediction: a large scale experiment on

data vs. domain vs. process, in: Proc. the 12th European Software Engineering Conference and the 17th ACM SIG-SOFT Symposium on the Foundations of Software Engineering, (ESEC/FSE '09), 2013.

[4] J. Nam, S. J. Pan, S. Kim, Transfer defect learning, in: Proc. the 35th International Conference on Software Engineering (ICSE '13), 2013.

[5] F. Zhang, A. Mockus, I. Keivanloo, Y. Zou, Towards building a universal defect prediction model with rank transformed predictors, Empir Software 21(5)(2016) 2107-2145.

[6] T. Menzies, A. Butcher, A. Marcus, T. Zimmermann, D. Cok, Local vs. global models for effort estimation and de-fect prediction, in: Proc. the 2011 26th IEEE/ACM International Conference on Automated Software Engineering, 2011.

[7] Y. Ma, G. Luo, X. Zeng, A. Chen, Transfer learning for cross-company software defect prediction, Information and Software Technology 54(3)(2012) 248-256.

[8] X. Xia, D. Lo, S.-J. Pan, N. Nagappan, X. Wang, HYDRA: massively compositional model for cross-project defect prediction, IEEE Transactions on Software Engineering 42(10)(2016) 977-998.

[9] S.J. Pan, I.W. Tsang, J.T. Kwok, Q. Yang, Domain adaptation via transfer component analysis, IEEE Trans. Neural Networks 22(2)(2010) 199-210.

[10] B. Schölkopf, A. Smola, K. Müller, Kernel principal component analysis, in: Proc. the 7th International Conference on Artificial, 2005.

[11] F. Peters, T. Menzies, A. Marcus, Better cross company defect prediction, in: Proc. 10th Int. Workshop Mining Softw. Repositories, 2013.

[12] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7(2010) 2399-2434.

[13] J. Thorsten, Transductive inference for text classification using support vector machines, in: Proc. the Sixteenth International Conference on Machine Learning, 1999.

[14] B. Turhan, T. Menzies, A. Bener, J. Di Stefano, On the relative value of cross-company and within-company data for defect prediction, Empirical Software Engineering 14(2009) 540-578.

[15] J. Nam, S. J. Pan, S. Kim, Transfer defect learning, in: Proc. Int. Conf. Soft. Eng., 2013.

[16] K. Yan, L. Kou, D. Zhang, Domain adaptation via maximum independence of domain features. <http://arxiv.org/abs/1603.04535>, 2016.

[17] B. Fernando, A. Habrard, M. Sebban, T. Tuytelaars, Unsupervised visual domain adaptation using subspace alignment, in: Proc. 2013 IEEE International Conference on Computer Vision, 2013.

[18] Y. Shi, L. Angeles, F. Sha, Information-theoretical learning of discriminative clusters for unsupervised domain adaptation, in: Proc. the 29th International Conference on Machine Learning, 2012.

[19] B. Gong, Y. Shi, F. Sha, K. Grauman, Geodesic flow kernel for unsupervised domain adaptation, in: Proc. the 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[20] A. Panichella, R. Oliveto, A. De Lucia, Cross-project defect prediction models: L'Union fait la force, in: Proc. 2014 Softw. Evol. Week - IEEE Conf. Softw. Maintenance, Reengineering, Reverse Eng. CSMR-WCRE 2014 - Proc, 2014.

[21] Y. Liu, T. M. Khoshgoftaar, N. Seliya, Evolutionary optimization of software quality modeling with multiple repositories, IEEE Trans. Softw. Eng. 36(2010) 852-864.

[22] E. Eaton, M. des Jardins, Selective transfer between learning tasks using task-based boosting, in: Proc. 25th AAAI Conf. Artif. Intell, 2011.

[23] G. Canfora, A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, S. Panichella, Multi-objective cross-project defect prediction, in: Proc. IEEE 6th Int. Conf. Softw. Testing, Verification Validation, 2013.

[24] M.-H. Halstead, Elements of Software Science, Elsevier Science, New York, 1997.

[25] Y. Jiang, B. Cukic, Fault prediction using early lifecycle data, in: Proc. 8th IEEE International Symposium on Software Reliability, 2007.

[26] McCabe, T, A complexity measure, IEEE Transactions on Software Engineering SE-2(1976) 308-320.

[27] B. Turhan, A. Bener, Analysis of naive bayes assumptions on software fault data: an empirical study, Data & Knowledge Engineering 68(2)(2009) 278-290.

[28] L. Guo, Y. Ma, B. Cukic, H. Singh, Robust prediction of fault-proneness by Random Forests, in: Proc. 15th Int'l Symp. Software Reliability Eng, 2004.

[29] M. D'Ambros, M. Lanza, R, Robbes, Evaluating defect prediction approaches: a benchmark and an extensive comparison, Empir Software Eng 17(2012) 531-577.

[30] Y. Penga, G. Wanga, H. Wangb, User preferences based software defect detection algorithms selection using MCDM, Information Sciences 191(2012) 3-13.

[31] I.-H. Laradji, M. Alshayeb, L. Ghouti, Software defect prediction using ensemble learning on selected features, Information and Software Technology 58(2015) 388-402.

[32] S. Shivaji, S.-E. James, W. Jr., S.-R. Akella, Senior, S. Kim, Senior reducing features to improve code change-based bug prediction, IEEE Transactions On Software Engineering 39(4)(2012) 552-569.

[33] J. Zheng, Cost-sensitive boosting neural networks for software defect prediction, Expert Systems with Applications 37(2010) 4537-4543.

[34] B. Turhan, T. Menzies, A.B. Bener, D.-J. Stefano, On the relative value of cross-company and within-company data for defect prediction, Empirical Software Engineering 21(2016) 2107-2145.

[35] S. Wang, X. Yao, Using class imbalance learning for software defect prediction, IEEE Transactions On Reliability 62(2)(2013) 434-443.