FeSCH: A Feature Selection Method using Clusters of Hybrid-data for Cross-Project Defect Prediction

Chao Ni[†], Wangshu Liu[†], Qing Gu[†*], Xiang Chen[†‡], Daoxu Chen[†]
[†] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[‡] School of Computer Science and Technology, Nantong University, Nantong, China
Email: jacknichao920209@gmail.com

***Abstract*—Cross project defect prediction (CPDP) is a challenging task since the predictor built on the source projects can hardly generalize well to the target project. Previous studies have shown that both feature mapping and feature selection can alleviate the differences between the source and target projects. In this paper, we propose a novel method FeSCH (Feature Selection using Clusters of Hybrid-data). In particular it includes two phases. The first is the feature clustering phase, which uses a density-based clustering method DPC to group highly co-related features into clusters. The second is the feature selection phase, which selects beneficial features from each cluster. We design three ranking strategies to choose appropriate features. During the empirical studies, we design experiments based on real-world software projects, and evaluate the prediction performance of FeSCH by analyzing the influence of ranking strategies. The experimental results show that FeSCH can outperform three baseline methods (i.e., WPDP, ALL, and TCA+) in most cases, and its performance is independent of the used classifiers.**

***Keywords*-Software Defect Prediction, Cross-project Defect Prediction, Feature Clustering, Feature Selection**

I. Introduction

Software defect prediction (SDP) is to predict fault-proneness of software modules so that test resources can be allocated effectively. SDP requires software metrics (i.e., features) to measure the code complexities or analyze the development processes. The defect prediction data are firstly gathered by mining software historical repositories through version control systems and bug tracking systems. Then a specific classifier is trained on the data, and used later to predict the potential defective software modules. By using SDP, developers can allocate the limited testing resources in a cost-effective manner.

previous SDP studies have focused on within-project defect prediction (WPDP), which builds the prediction model and predicts defects within the same project. Such handling suffers the problem that the target project may not have enough labeled modules for training. An ideal solution is to use data from other projects (called the source projects) to make prediction on the target project, which leads to the cross-project defect prediction (CPDP). CPDP is a challenging task since the predictor built on the source projects can hardly generalize well to the target project. Researchers have applied transfer learning to handle the differences in feature distributions between the source and target projects. Previous studies [1], [2] have shown that both feature mapping and feature selection can alleviate the differences between the source and target projects.

[*] Correspondence author. Email: guq@nju.edu.cn

In this paper, we propose a novel feature selection method FeSCH to make use of the source projects for fault prediction on the target project. This method includes two phases. In the first phase, it uses a density-based clustering method DPC [3] to divide the original feature set into multiple clusters. In the second phase, we propose three different ranking strategies to select appropriate features from each cluster. These ranking s-trategies are LDF (Local Density of Features), SFD (Similarity of Feature Distributions), and FCR (Feature-Class Relevance). Unlike feature mapping, we think these strategies can provide guidelines to choose useful software measures under CPDP.

To verify the effectiveness of the proposed FeSCH, we design experiments on AEEEM dataset, compare FeSCH to other methods, and analyze its design options. Final results show that FeSCH can outperform three baseline methods (i.e., WPDP, ALL, and TCA+ [1]) in most cases and its performance is independent of the used classifiers.

The main contributions of this paper can be summarized as follows: (1) We propose a novel feature selection method FeSCH for CPDP. We use the density-based clustering method to group the features into clusters. Then We design three different ranking strategies to select useful features from each cluster. (2) We conduct empirical studies based on real-world software projects to demonstrate the potentials of FeSCH and provide guidelines for using FeSCH.

II. Background and Related Work

In cross-project defect prediction, the characteristics of the target and source projects are usually different. Early researchers have conducted empirical studies under CPDP and their results were not optimistic [4], [5]. Then researchers proposed new approaches to improve the defect prediction performance under CPDP. These approaches are based on instance selection and weight setting [6]–[8], transfer learning [1], [9], multi-objective optimization [10], or ensemble learning [11]. Some researchers consider the case where the source and target projects have different feature sets. Nam and Kim [2] proposed a heterogeneous defect prediction method. Jing et al. [12] solved this problem by defining unified feature

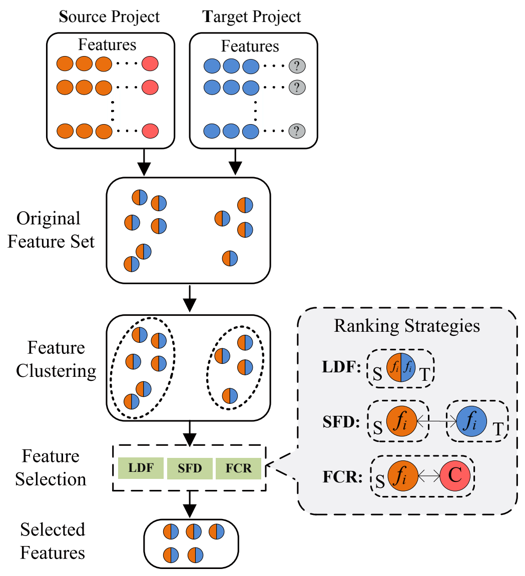

Fig. 1. The Framework of Our Method FeSCH

space and applying CCA (Canonical Correlation Analysis)-based transfer learning. Other researchers considered using unsupervised methods. The idea is that defective software modules usually have excessive feature values or distinctive value patterns. Nam and Kim [13] proposed CLAMI. Zhang et al. [14] used a connectivity-based unsupervised classifiers.

Feature selection is also widely used in software defect prediction. Gao et al. [15] applied feature selection for defect prediction on a large legacy software system in telecommunications. Shivaji et al. [16] applied feature selection in change-based defect prediction. Our previous work [17], [18] proposed a cluster based feature selection framework and a two-phase data preprocessing approach to improve the performance of defect predictors under WPDP.

In this paper, we assume the source and target projects have similar feature sets for CPDP, and then propose a novel feature selection method. Our method is extended from our previous work [17]. These extensions include new clustering method and new feature selection strategies.

III. The Method FeSCH

The framework of FeSCH (Feature Selection using Clusters of Hybrid-data) is shown in Figure 1. This framework includes two phases: the feature clustering phase and the feature selection phase. After FeSCH, a classifier is trained on the cross-project data using selected features as the defect predictor. In this paper, we choose the Logistic Regression as the classifier, which is commonly used [1], [2] and its performance can be guaranteed.

A. Phase 1: the Feature Clustering Phase

In this phase, we use DPC (Density Peaks Clustering) [3] to cluster features using data from both the source and the target software projects. In DPC, the cluster center has two characteristics: great density and large distance. The great density requires that a cluster center should be surrounded by those with smaller densities. The large distance requires that the center should have a large distance to those with greater densities.

In the context of CPDP, DPC is used to divide the features into non-predetermined number of clusters on all the available software data (i.e., both the source project and the target project). Three steps are involved in clustering features using DPC.

In the first step, for each feature f_i, its local density ρ_i and local distance δ_i are calculated. Let d_{ij} be the Euclidean distance of the features f_i and f_j, computed as follows:

$$d_{ij} = \sqrt{\sum_{k=1}^{m}(x_{i_k} - x_{j_k})^2} \tag{1}$$

Where m is the number of instances from the dataset, and x is the feature value in an instance. The local density ρ_i is the number of features within a pre-determined distance:

$$\rho_i = \sum_j \chi(d_{ij} - d_c) \tag{2}$$

Where $\chi(x) = 1$ if $x < 0$ and $\chi(x) = 0$ otherwise, d_c is the cutoff distance which ensures each feature has 2% of the total features as its locals.

The local distance δ_i is the Euclidean distance from the feature f_i to the nearest feature with greater local density:

$$\delta_i = \min_{j:\rho_j > \rho_i}(d_{ij}) \tag{3}$$

For the features with the greatest local density, their local distances are the greatest possible Euclidean distances:

$$\delta_i = \max_j(d_{ij}), \text{ if } f_i \text{ has the maximum } \rho_i \tag{4}$$

In the second step, we determine centers of the feature clusters using the calculated local densities and local distances. Firstly, for the feature f_i, we calculate the centrality measure γ_i as $\gamma_i = z(\rho_i) \times z(\delta_i)$. Where $z(\cdot)$ is the z-score normalization function. Secondly, we sort the features in descending order of γ_i, denoted as $\{\gamma\}_i^n$, where n is the total number of features. Based on $\{\gamma\}_i^n$, which conforms to the power law distribution, we can use its singular inflection point as the cut point γ_c. The features with centrality measures greater than γ_c are used as the cluster centers.

In the last step, suppose there are k centers determined, the remaining features are each assigned to the nearest center which has greater local density. By using DPC, the features are clustered in a single round and k needs not to be pre-specified.

B. Phase 2: the Feature Selection Phase

In phase 2, three ranking strategies are designed to rank the features in each cluster. Since clusters have non-similar size, and number of clusters is not pre-determined, the number of selected features is proportional to the cluster size (e.g., more features will be selected from larger clusters). Such handing may introduce redundancy, but it keeps the number of selected features. The three strategies are defined as follows.

1) LDF (Local Density of Features): The features are selected using the local density measure. The local density means the number of neighbors around a feature, which may reflect the degree of popularity or representativeness of the feature.

2) SFD (Similarity of Feature Distributions): The features are selected based on the level of distribution similarity between the source and the target project data. We use MIC (Maximal Information Coefficient) [19] to measure the difference of two distributions. Given two vectors X and Y, which corresponding to the same feature from the source project and the target project respectively, $MIC(X, Y)$ can be computed as follows:

$$MIC(X,Y) = \max_{g_x \times g_y < n^{0.6}} \frac{\max_{G \in (g_x, g_y)} I(X(G), Y(G))}{log\ min(g_x, g_y)} \tag{5}$$

Where G belongs to the grids generated by multiple $g_x \times g_y$ combinations, g_x and g_y are numbers of the equal partitions of vectors X and Y respectively, and n is the search range of G [19]. For each grid G, $X(G)$ and $Y(G)$ are vectors of the mean values of the partitions. We set the maximum possible $I(X(G), Y(G))$ as the MIC value. $I(\cdot)$ is the function of mutual information computed as follows:

$$I(X,Y) = H(X) + H(Y) - H(X,Y) \tag{6}$$

Where $H(\cdot)$ is the entropy function, which is computed as follows:

$$H(X) = -\sum_{x \in X} p(x) log_2 p(x) \tag{7}$$

3) FCR (Feature-Class Relevance): This strategy has been used in our previous work FECAR [17]. The main assumption is that the features strongly related to the class labels are worth to be selected. In the context of CPDP, since target data have no class labels, we use IG (information gain) to measure the feature-to-label relevance based on the source data. IG can be computed as follows:

$$IG(X|Y) = H(X) + \sum_{y \in Y} p(y) \sum_{x \in X} p(x|y) log_2 p(x|y) \tag{8}$$

Where X is the feature vector from the source data, Y is the class label vector, x and y are the unit values, and $p(\cdot)$ is the probability function. $H(\cdot)$ is the entropy function computed by Formula (7).

Comparing the three ranking strategies, LDF combines the source and target data as a whole, and selects the representative features. SFD makes a comparison between the source and target data, and selects the features which have similar distributions between these two datasets. FCR only considers the source data, and selects features which have high relevance to the class label. In our empirical studies, we will further compare these ranking strategies. Although multiple features may be selected from one cluster, the latter two strategies select features at a different perspective, and hence reduce the possible redundancy.

IV. Experimental Setup

A. Research Questions

In our empirical study, we want to investigate the following two research questions.

RQ1: Whether our FeSCH can further improve the performance of CPDP when compared with existing classical baseline methods?

RQ2: How do the design options (i.e., ranking strategy, feature selection ratio, and the classifier) of FeSCH affect the performance of CPDP?

B. Datasets

In the experiments, we use a classical datasets AEEEM for CPDP. Table I shows the details of this dataset. It was collected by D'Ambros et al. [20]. They considered 61 metrics: 5 previous-defect metrics, 5 entropy metrics, 17 source code metrics, 17 entropy-of-source-code metrics, and 17 churn-of-source-code metrics.

TABLE I
The Characteristics of the AEEEM Dataset

Projects	#Features	#Instances	#Faulty Instances (Percent)
EQ	61	324	129 (39.8%)
JDT	61	997	206 (20.7%)
LC	61	691	64 (9.3%)
ML	61	1,862	245 (13.2%)
PDE	61	1,497	209 (14.0%)

C. The Evaluation Measures

According to the ground truths and the predicted class labels, we can compute the number of true positives (*TP*), false positives (*FP*), true negatives (*TN*), and false negatives (*FN*) respectively. Then, we can calculate the precision measure p and recall measure r as follows:

$$p = \frac{TP}{TP + FP},\ r = \frac{TP}{TP + FN} \tag{9}$$

There always exists a tradeoff between p measure and r measure. To handle the dilemma, the $F1$ measure is defined as follows:

$$F1 = \frac{2 \times p \times r}{p + r} \tag{10}$$

To check the significance of performance comparison, we conduct the Wilcoxon signed-rank test [21], which is a non-parametric statistical hypothesis test, on the prediction results. For all the tests, the null hypotheses is that there is no difference between the trained predictors, and the significance level α is set as 0.05.

D. Experimental Design

To evaluate the performance of FeSCH in CPDP, we use three baseline methods: WPDP (within-project defect prediction without feature selection), ALL (All features under CPDP), and TCA+ [1]. For WPDP, we train and test the defect predictor on the target project, and use 2-fold cross validation (CV). We do not use 10-fold CV since, first, 2-fold CV is a common setting in CPDP [1] [2], second, 90% labeled instances available in the target project makes an unfair comparison under CPDP. To overcome possible bias in the random selection processes, we repeat the random CV split 100 times and report the average prediction results. For ALL, we train the defect predictor using all features from the source project, and use the trained predictor to predict defects on the target project. For TCA+, we directly use the results reported by Nam et al. [1].

We use Logistic Regression as the defect predictor, commonly used in previous research [1], [2]. During experiments, we use the implementation provided by LibLinear [22] and consider the option (i.e., -S 0 -B -1) used in TCA+ [1]. To calculate the relevance formulas, we use the MINE (Maximal Information-based Nonparametric Exploration) toolkits with default setting.

TABLE II
COMPARISON OF FESCH AND THE OTHER BASELINE METHODS ON AEEEM, THE LOGISTIC REGRESSION IS USED AS THE CLASSIFIER

Source ⇒ Target	**WPDP**	**ALL**	**TCA+**	**FeSCH**
JDT ⇒ EQ	0.58	0.30	0.60	**0.66**
LC ⇒ EQ	0.58	0.51	**0.62**	0.50
ML ⇒ EQ	0.58	0.24	**0.56**	0.46
PDE ⇒ EQ	0.58	0.43	**0.60**	0.55
EQ ⇒ JDT	0.53	0.39	**0.54**	0.32
LC ⇒ JDT	0.53	0.49	0.56	**0.62**
ML ⇒ JDT	0.53	0.42	0.43	**0.62**
PDE ⇒ JDT	0.53	0.47	0.48	**0.61**
EQ ⇒ LC	0.31	0.26	0.27	**0.45**
JDT ⇒ LC	0.31	0.26	0.31	**0.53**
ML ⇒ LC	0.31	0.10	0.25	**0.49**
PDE ⇒ LC	0.31	0.33	0.33	**0.52**
EQ ⇒ ML	0.26	0.19	0.23	**0.48**
JDT ⇒ ML	0.26	0.27	0.36	**0.49**
LC ⇒ ML	0.26	0.20	0.29	**0.43**
PDE ⇒ ML	0.26	0.28	0.29	**0.42**
EQ ⇒ PDE	0.33	0.36	0.33	**0.48**
JDT ⇒ PDE	0.33	0.28	0.38	**0.51**
LC ⇒ PDE	0.33	0.31	0.37	**0.49**
ML ⇒ PDE	0.33	0.27	0.37	**0.47**
Average	0.40	0.32	0.41	**0.51**

V. RESULTS ANALYSIS

A. Result Analysis for RQ1

The results of FeSCH on AEEEM dataset is shown in Table II. the first column presents specific cases of CPDP. For example, in Table II, the case "JDT ⇒ EQ" means the project "JDT" is used as the source project, and the project "EQ" is used as the target project. The rest columns list the results of three baseline methods and our proposed FeSCH. The last row provides the average performance. In FeSCH, the SFD ranking strategy is used, and the feature selection ratio is set as 30%. For each row, the best result is underlined, and the best result of CPDP methods (i.e., ALL, TCA+, and FeSCH) is set in bold.

From the last row of this table, we can find that FeSCH can achieve the best performance among the four methods. Considering each case of CPDP, compared with WPDP, FeSCH performs better in most cases. For example, in the case "LC ⇒ JDT", FeSCH achieves the best performance of 0.62, greater than 0.53 of WPDP. The evidences strongly suggest that FeSCH makes an effective use of the source projects. Compared with ALL, FeSCH performs better in nearly all the cases. The results prove that feature selection is essential for cross-project software defect prediction, conforming to the findings of Nam et al. [1]. The possible reasons are that irrelevant and redundant features will cause serious sabotage for software defect prediction, especially under CPDP. Compared with TCA+, FeSCH still performs better in majority of cases. The results suggest that feature selection is promising under CPDP, which may decrease the extra costs required for software measures. However, FeSCH still performs relatively poor in several cases, such as "EQ" is set as the target project. The reason maybe this dataset contains less than 350 instances and may not provide enough information for feature selection. In these cases, the feature space projection used in TCA+ is more helpful.

Table III summarizes the results of "Win/Draw/Loss" between FeSCH and the other three baseline methods. In Table III, each row refers to one project taken as the target project, and one of the rest projects served as the source project. From Table III, FeSCH has at least 16/20 wins on AEEEM, which demonstrates its effectiveness. To check whether the performance differences among WPDP, ALL, TCA+, and FeSCH are significant, we conduct the Wilconxon signed-rank test. the *p*-values of "FeSCH *vs.* WPDP", "FeSCH *vs.* ALL", and "FeSCH *vs.* TCA+" is 1.80e-03, 7.00e-05, and 2.56e-03, which indicates that FeSCH is statistically better than the baseline methods.

TABLE III
WIN/DRAW/LOSS OF FESCH COMPARED TO THE OTHER BASELINES

Target	Against(Win/Draw/Loss)		
	WPDP	ALL	TCA+
EQ	1/0/3	3/0/1	1/0/3
JDT	3/0/1	3/0/1	3/0/1
LC	4/0/0	4/0/0	4/0/0
ML	4/0/0	4/0/0	4/0/0
PDE	4/0/0	4/0/0	4/0/0
Total	**16/0/4**	**18/0/2**	**16/0/4**

In summary, FeSCH can outperform the three baseline methods in most CPDP cases. However, when the target project does not have enough data (e.g., less than 350 instances), the performance of FeSCH is decreased.

B. Result Analysis for RQ2

For RQ2, we study how the design options of FeSCH affect its performance for CPDP and hence provide a guideline for using FeSCH. Here we focus on three options: the ranking strategy, the feature selection ratio, and the trained classifier.

1) Effects of the Ranking Strategies: In FeSCH, three ranking strategies, LDF, SFD, and FCR are designed for selecting representative features from each cluster. Table IV shows the results of FeSCH when using different ranking strategies.

TABLE IV
COMPARISON AMONG THE THREE RANKING STRATEGIES ON AEEEM

Source ⇒ Target	WPDP	LDF	SFD	FCR
JDT ⇒ EQ	0.58	0.41	**0.66**	0.52
LC ⇒ EQ	**0.58**	0.38	0.50	0.55
ML ⇒ EQ	**0.58**	0.37	0.46	0.53
PDE ⇒ EQ	**0.58**	0.41	0.55	0.49
EQ ⇒ JDT	**0.53**	0.22	0.32	0.33
LC ⇒ JDT	0.53	**0.67**	0.62	**0.67**
ML ⇒ JDT	0.53	**0.63**	0.62	0.61
PDE ⇒ JDT	0.53	0.60	**0.61**	0.59
EQ ⇒ LC	0.31	0.34	0.45	**0.46**
JDT ⇒ LC	0.31	0.51	**0.53**	**0.53**
ML ⇒ LC	0.31	0.47	**0.49**	0.48
PDE ⇒ LC	0.31	0.47	**0.52**	0.45
EQ ⇒ ML	0.26	0.33	**0.48**	0.17
JDT ⇒ ML	0.26	0.39	**0.49**	**0.49**
LC ⇒ ML	0.26	**0.64**	0.43	0.45
PDE ⇒ ML	0.26	0.75	0.42	**0.80**
EQ ⇒ PDE	0.33	0.12	**0.48**	0.41
JDT ⇒ PDE	0.33	**0.51**	**0.51**	**0.51**
LC ⇒ PDE	0.33	0.52	0.49	**0.53**
ML ⇒ PDE	0.33	**0.50**	0.47	0.43
Avg	0.40	0.46	**0.51**	0.50
Win/Draw/Loss	—	14/0/6	**16/0/4**	14/0/6

In this table, the first column presents specific cases of CPDP. The second column presents the results of WPDP, which is used as the baseline. The last three columns show the results of FeSCH using each of the three ranking strategies. The last two rows of the this table show the average results and "Win/Draw/Loss" against WPDP. For each row, the best result is emphasized in bold.

Among the ranking strategies, SFD achieves the best average performance (i.e., 0.51 on AEEEM), while LDF performs worst (i.e., 0.46 on AEEEM). No ranking strategy can win the others in all the CPDP cases. For example, LDF still performs the best in 5 of the 20 cases on AEEEM. On the other hand, SFD only performs the best in 9 cases on AEEEM. LDF performs the worst when the project "EQ" is set as the target project, which suggests that LDF is sensible to the difference between the source and target projects. Considering the definitions of these ranking strategies, SFD selects features based on similarity of feature distributions between the target and source projects, FCR considers on the strong relevance between features and class labels on the source projects, while LDF calculates the Euclidean distances between features taking the source and target projects as a whole. The results suggest that both distribution similarity and class relevancy are suitable for selecting features in CPDP, while directly combining the data of the source and target projects may not work well.

In summary, all the three ranking strategies are effective and useful for FeSCH. The strategy SFD, which is to increase the similarity of feature distributions between the source and target projects, performs better in our experiments. The strategy LDF, although performs worst, is till promising in our experiments.

2) Effects of the Feature Selection Ratio: To study the effects of the feature selection ratio on FeSCH, we conduct experiments by varying the feature selection ratio from 10% to 100% step by 10%. The used ranking strategy is SFD. Figure 2 shows the results. In this figure, the x-axis indicates the feature selection ratio, and y-axis indicates the corresponding $F1$ measure. The trend lines represent the mean $F1$ measures on different target projects, where the variances are caused by using different source projects. From this figure, selecting 30% of the original features can achieve good-enough performance. When "LC", "PDE", or "ML" is set as the target project respectively, selecting 20% to 40% features is preferred. When "JDT" is set as the target project, the optimal selection ratio is about 20%. On the other hand, when "EQ" is set as the target project, the result is exceptional: the optimal ratio has a long range from 20% to 90%.

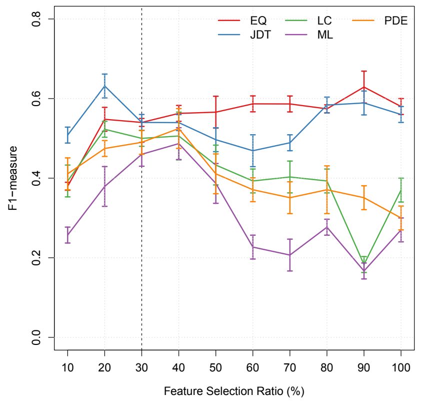

Fig. 2. Average Performance of FeSCH by varying the Feature Selection Ratio

In summary, For FeSCH, the feature selection ratio can be set within 20% to 40%, and 30% can be a preferred option, at least on AEEEM. Other CPDP cases are still required to provide sound evidence.

3) Effects on Different Classifiers: To check the performance of FeSCH when training different classifiers, we make a comparison among three commonly used classifiers: LR (Logistic Regression), NB (Naive Bayes), and RF (Random Forest). The ranking strategy used is SFD, and the feature selection ratio is 30%. Figure 3 shows the average performances on different target projects.

In Figure 3, the x-axis indicates the target projects, and y-axis indicates the corresponding $F1$ measures. Each bar represents the mean $F1$ measures on a specific target project by one of the three classifiers. From this figure, the $F1$

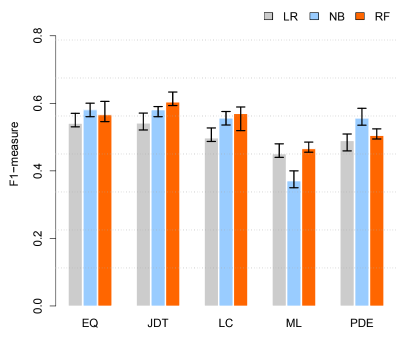

Fig. 3. Comparison among the classifiers after FeSCH

measures vary a lot among the 5 target projects, while on a single target project, little difference can be found among the three classifiers. For example, NB performs the best on “EQ” and “PDE”, while RF performs the best on “JDT”, “LC” and “ML”. LR looks inferior, but the decrease in $F1$ measure is less than 0.07. The above analysis suggests that the performance of FeSCH maybe independent of the classifiers used.

In summary, by applying FeSCH, the classifiers may not have discriminative effects on the prediction performance under CPDP. Researchers should pay attention to build high quality source data for training useful defect predictors for the target software.

VI. Conclusion and Future Work

In this paper, we propose a feature selection method FeSCH to minimize the difference in feature distributions between the source project and the target project data under CPDP. In our future work, we plan to design other novel feature selection strategies to improve the performance of FeSCH. We also plan to involve more heterogeneous projects and use other sophisticated classifiers (such as Support Vector Machine and neural networks) to verify the generality of our experimental results.

Acknowledgment

This work is supported in part by National Natural Science Foundation of China (Grant Nos. 61373012, 91218302, 61321491, 61202006), Collaborative Innovation Center of Novel Software Technology and Industrialization, National Basic Research Program of China (Grant No. 2009CB320705), and Open Project of State Key Laboratory for Novel Software Technology at Nanjing University (Grant No. KFKT2016B18).

References

[1] J. Nam, S. J. Pan, and S. Kim, “Transfer defect learning,” in *Proceedings of the International Conference on Software Engineering*, 2013, pp. 382–391.

[2] J. Nam and S. Kim, “Heterogeneous defect prediction,” in *Proceedings of the ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, 2015, pp. 508–519.

[3] A. Rodriguez and A. Laio, “Clustering by fast search and find of density peaks,” *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[4] 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,” in *Proceedings of the ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, 2009, pp. 91–100.

[5] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, “An investigation on the feasibility of cross-project defect prediction,” *Automated Software Engineering*, vol. 19, no. 2, pp. 167–199, 2012.

[6] B. Turhan, T. Menzies, A. B. Bener, and J. D. 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.

[7] F. Peters, T. Menzies, and A. Marcus, “Better cross company defect prediction,” in *Proceedings of the Working Conference on Mining Software Repositories*, 2013, pp. 409–418.

[8] Y. Ma, G. Luo, X. Zeng, and A. Chen, “Transfer learning for cross-company software defect prediction,” *Information and Software Technology*, vol. 54, no. 3, pp. 248–256, 2012.

[9] L. Chen, B. Fang, Z. Shang, and Y. Tang, “Negative samples reduction in cross-company software defects prediction,” *Information and Software Technology*, vol. 62, pp. 67 – 77, 2015.

[10] G. Canfora, A. De Lucia, M. D. Penta, R. Oliveto, A. Panichella, and S. Panichella, “Multi-objective cross-project defect prediction,” in *Proceedings of the International Conference on Software Testing, Verification and Validation*, 2013, pp. 252–261.

[11] X. Xia, D. Lo, S. J. Pan, N. Nagappan, and X. Wang, “Hydra: Massively compositional model for cross-project defect prediction,” *IEEE Transactions on Software Engineering*, pp. 977–998, 2016.

[12] X. Jing, F. Wu, X. Dong, F. Qi, and B. Xu, “Heterogeneous cross-company defect prediction by unified metric representation and cca-based transfer learning,” in *Proceedings of the ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, 2015, pp. 496–507.

[13] J. Nam and S. Kim, “Clami: Defect prediction on unlabeled datasets,” in *Proceedings of the International Conference on Automated Software Engineering*, 2015, pp. 452–463.

[14] F. Zhang, Q. Zheng, Y. Zou, and A. E. Hassan, “Cross-project defect prediction using a connectivity-based unsupervised classifier,” in *Proceedings of the International Conference on Software Engineering*, 2016, pp. 309–320.

[15] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, “Choosing software metrics for defect prediction: an investigation on feature selection techniques,” *Software: Practice and Experience*, vol. 41, no. 5, pp. 579–606, 2011.

[16] S. Shivaji, E. J. Whitehead, Jr., R. Akella, and S. Kim, “Reducing features to improve code change-based bug prediction,” *IEEE Transactions on Software Engineering*, vol. 39, no. 4, pp. 552–569, Apr. 2013.

[17] S. Liu, X. Chen, W. Liu, J. Chen, Q. Gu, and D. Chen, “Fecar: A feature selection framework for software defect prediction,” in *Proceedings of the Annual Computer Software and Applications Conference*, 2014, pp. 426–435.

[18] W. Liu, S. Liu, Q. Gu, J. Chen, X. Chen, and D. Chen, “Empirical studies of a two-stage data preprocessing approach for software fault prediction,” *IEEE Transactions on Reliability*, vol. 65, no. 1, pp. 38–53, 2016.

[19] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. Mcvean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti, “Detecting novel associations in large data sets,” *Science*, vol. 334, no. 6062, pp. 1518–1524, 2011.

[20] M. D’Ambros, M. Lanza, and R. Robbes, “An extensive comparison of bug prediction approaches,” in *Proceedings of the International Working Conference on Mining Software Repositories*, 2010, pp. 31–41.

[21] F. Wilcoxon, “Individual comparisons by ranking methods,” *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[22] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “Liblinear: A library for large linear classification,” *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.