Isolation Forest Filter to Simplify Training Data for Cross-Project Defect Prediction

Can Cui, Bin Liu, Shihai Wang
School of Reliability and Systems Engineering,
Beihang University
Beijing, China
cuican1414@buaa.edu.cn, liubin@buaa.edu.cn, wangshihai@buaa.edu.cn

Abstract—**Cross-project defect prediction (CPDP) is an active research area. When the historical data is limited or a new project is to develop, establishing CPDP models is very useful, which assists software testers to judge the defect-prone entities or software managers to focus on the "important" parts by allocating the manpower, budget, time. However, the dissimilarity of data distributions between the source projects and the target project decreases the performance of CPDP models. How to simplify or the cross-project training data is an important problem. To solve this issue, an isolation forest (iForest) filter is proposed. We use 15 versions of different java projects from open PROMISE Data Repository and five typical predictors (*naïve bayes* (NB), *decision tree* (DT), *logistic regression*(LR), *k-nearest neighbor*(*k*-NN) and *random forest*(RF) to build 1050 (15*14*5) software defect prediction models (SDPM). Meanwhile, we compare our models with Burak Filter models and Peter Filter models. From the results of performance measures, called AUC, balance, G-measure, G-mean, F1-measure, we can know that our iForest filter is feasible and even better than other two. Therefore, using iForest filter can make cross-project training data simple and build efficient SDPM.**

Keywords- Cross-project defect prediction (CPDP); Isolation Forest Filter; data mining; transfer learning ;training data simplification

I. Introduction

Software quality (SQ) has always been a concern in software engineering (SE). Software defect prediction (SDP) can predict many defects in developing software and plays an important role in SQ [1-3]. Therefore, SDP becomes a hot topic in more than a decade. If the historical data is sufficient, establishing a within-project defect prediction (WPDP) model is useful to predict the defect-prone [2-3]. While in the lack of historical data scenario, for example, a company is too small to collect data or the target project is new without historical releases, WPDP does not work without sufficient data from the same project. Therefore, Cross-project defect prediction (CPDP) is proposed. CPDP means that using data from other projects (training data) to build software defect prediction models (or software defect predictors) to predict the defect-prone of software entities for a target project [1, 4]. This approach uses "*transfer learning*" idea to address the SE problems [5]. However, there exists some challenges to conduct CPDP models. Using data from other projects can collect enough data, nevertheless, generally, the different data distribution about the training data and test data caused great trouble on SDPM. Zimmermann et al. [6] explored the CPDP for different software projects on data, domain and process. They advised researchers on data similarity for CPDP. Whereafter, many researchers' proposed different data filters at instance level and project level [7-10]. Burak et al. proposed a NN filter (also called Burak filter by other researchers) via *k-nearest neighbor* (*k*-NN) to make the training data distribution more similar to test data distribution [8]. Peter filter was make by *k-means* and *k-NN* [7]. He et al. [9] proposed a projects-filter not an instances-filter by focusing on the data distribution characteristics via *k*-NN. Then they combined their projects-filter with Burak filter and Peter filter to simplify the similar projects data as the final filtered training data. Kawata et al. [10] considered the noisy instances (aka. samples) in cross-projects by using *density-based spatial clustering of applications with noise* (DBSCAN) filter approach. The procedure is similar with Peter filter. They filtered some noisy instances to gain acceptable performance.

These data filter approaches to make SDPM perform better. However, they all assume that the nearer distance or denser density stands for the more similar of two instances or projects [7-11]. These approaches are strong affected by different magnitude data values. Although data normalization or standardization are usually used before filter methods, how to find high quality data is still a valuable exploration. Unlike these filters above, we propose an isolation forest (iForest) filter, which filters noisy data without considering distance or density. We follow Burak that using test data to guide data filter and delete the abnormal points (outliers) by estimating training data. To validate our approach, we use 15 releases of 15 different projects, which is developed by an object-oriented language called Java, from open PROMISE Data Repository. We conduct O2O CPDP by five classical classifiers, which are *naïve bayes* (NB), *decision tree* (DT), *logistic regression* (LR), *k-nearest neighbor* (*k*-NN) and *random forest* (RF). Totally, 1050 (15*14*5) SDPM are developed. Meanwhile, we

This work was supported by Foundation of No.61400020404.

compare our filter with Burak Filter models and Peter Filter. From the results of performance measures, called AUC, balance, G-mean, G-measure, F1-measure, we can know that our iForest filter is feasible and even better than other two. Therefore, using iForest filter can make cross-project training data simple and build efficient SDPM.

The rest organization is as follows: Section II introduces the background and related works. Section III explains the iForest filter approach proposed in the paper. Experiment is set up and results are discussed in Section VI. Section V makes a conclusion and recommends some future work.

II. Background And Related Works

WPDP usually use historical data from the target project or the released versions of the same project to build classification or regression models via machine learning approaches [12-14]. However, WPDP do not work when lacking of data. Therefore, some researchers tried to use a "*transfer learning*" idea to collect data from other project, which is different with the target one on domain, development environment, development team, company [5,15].

From the perspective of metrics sets, there are two categories in CPDP: heterogeneous defect prediction (HDP) and homogeneous defect prediction [16]. If the source projects and the target project are heterogeneous, there are homogeneous metrics sets, which means that they have different features (aka. attributes or metrics). Before performing software defect prediction models SDPM, it is important and necessary to convert the feature dimensions at the same size. Kim et al. proposed an HDP approach to solve the homogeneous metrics sets [16]. For homogeneous defect prediction (common metrics, CM), there are common metrics between the source projects and the target project. Directly using cross-project data can build SDPM. Zimmermann et al. [6] build 622 one-to-one (O2O) CPDP models, which are one project is as a source project and one is as a target project. However, the performance of the models usually unacceptable, which is the acceptable rate is 3.4%. They proposed 40 similarity metrics, including domain metrics, process metrics and code metrics. Meanwhile, they point out that it is a promising challenge to focus on the similarity between the source and target projects. In our paper, we focus on the O2O *heterogeneous* defect prediction, i.e., O2O CPDP_CM.

NN filter is proposed to seek for similar instances for CPDP_CM by Burak et al. [8]. They use *k*-NN algorithm to select *k* nearest neighbor about for every test instance. They used Euclidean distance to calculate the similarity between the training instances and the test instances. They assume that the nearer two instances, the more similar between them. Therefore, finding similar training instances for all test instances is useful. In their paper, they set *k*=10. They do not consider the duplicate of select training instances. They argue that their approach can filter more similar instances, gain good results and improve the performance of SDPM than without using NN filter. However, from the opposite perspective, Peter considered the rich cross training data includes more information than limited test data [7]. Thus, they combined training data and test data. Then, they use *k-means*, which is a cluster algorithm by using Euclidean distance to divide the combined data into *k* clusters. Then, they deleted the clusters which do not contain test instances. For retain clusters, they using *k-NN* (*k*=1) to find the nearest test instance for every training instance in the same cluster. Finally, they find the nearest training instance for every filtered test instance by *k-NN* (*k*=1). If the training instance is selected, the next nearest one is chosen. To compare with NN filter, they set a cluster contains 10 instances of *k-means*. Menzies et al. Herbord [17] used two algorithms called *EM cluster* and *k-NN* as two strategies to filter similar projects by using data characteristics (mean values and standard deviation values of data). He then combined his approach with NN filter and Peter filter. Bettenburg, Menzies, Herbord and their participators[18-19] discussed local models and global models for CPDP. They found that a local model is minor better with global models. And local models are suitable for specific subsets. Bin et al.[20] compared 9 data filters to give advice for software workers which one to select. They mainly supported making local models is useful for specific subsets and advised that using the global model for CPDP.

Though there are some filters to select similar instances or delete noisy data, their methods all depend on distance or density, which is sensitive to the feature values. Data normalization or standardization can improve models' performance to some extent but change the original features values of projects. To solve this issue, we using an unsupervised clustering algorithm, iForest, which is usually used in anomaly detection to discard noisy data (outliers) and select similar data (inliers).

III. Methodology

In this section, we mainly introduce the iForest filter approach and the O2O CPDP_CM approach.

A. *iForest filter*

iForest is an unsupervised machine learning approach[21]. It uses a fixed size of sub-instances to conduct a small number trees. This method uses path length to judge anomalies and normal instances and doesn't depend on distance or density measure. More details is shown in [21].

We are inspired by Burak filter and simplify data of a source project based on the target project data. To the best of our knowledge, it is the first time a special algorithm for anomaly detection approach is used in SDP. The pseudocode to filter data set by Isolation Forest filter is shown in Fig. 1. We support Burak and his partners that the target data is the direction. Therefore, we assume that all the feature values from the target project are normal points (inliers). From Fig. 1, we can see that using the target project data set (all the feature values) as training data to fit an iForest model, and then applying a source project data set (all the feature values with label values) as test data to the model. By the model, all the instances in source project were predicted. If the prediction value is larger than a cut-off value, the corresponding instance is labeled as an inlier, otherwise, the opposite. All the instances detected to inliers are as candidate training data for O2O CPDP_CM. Note that before establishing the iForest filter, shuffle the order of the target data set is needed, which avoids the influence of the randomness from iForest approach.

```
Algorithm 1 Isolation Forest Filter to Simplify Training Data for CPDP
Input: Source data set $Trf = \{Trf_1, Trf_2, \cdots, Trf_p\}$;
    $Trl = \{Trl_1, Trl_2, \cdots, Trl_p\}$;
Target data set without labels $Tef = \{Tef_1, Tef_2, \cdots, Te_q\}$;
Output: Filtered data set for CPDP;
Approach:
1: Let filtered training data $FTrf = \varnothing$, $FTrl = \varnothing$;
2: Shuffle the order of a target data set $Tef_i$ in $Tef$;
3: Using $Tef_i$ to establish an iForest model $M$;
4: Apply $M$ by input an source data set $Trf_j$ in $Trf$, and gain the predict results $P=\{P_1, P_2, \cdots, P_m\}$, where $m$ is the instance number of the training data;
5: for $P_k$ in $P$:
6:     if $P_k > P_0$:  // $P_0$ is a cut-off value to judge outliers or inliers
7:     $FTrf \leftarrow Trf_j(Tr_j.\text{index}(P_k))$  // the select training instance with feature values corresponding to $P_k$
8:     $FTrl = Trl_j(Tr_j.\text{index}(P_k))$  // the select training instance with label values corresponding to $P_k$
9:     end if
10: end for
11: return $FTrf$, $FTrl$; //filtered feature and label values.
```

Figure 1. Pseudocode of filter data set by Isolation Forest Filter

B. O2O CPDP_CM

After filtering useful data and deleting noisy data, O2O CPDP_CM are conducted. In the scenario, the source data set includes feature values and label values and the target data set only with feature values are known. The procedure pseudocode of building models is indicated in Fig. 2.

From Fig. 2, we can know that using target data set as training data to fit an iForest filter model to gain cleaned source data set. Then, the cleaned source data set is used as training data to fit a classification model. Please note that using one target project data set and one source data set to build one classification model. Meanwhile, the source data set and target data set have common metrics, i.e., the same feature number and the same feature name.

IV. Case Study

A. Data Sets for Expenriment

To prove the effectiveness of the method, we use 15 data sets from 15 different projects of PROMISE Data Repository. These projects all developed by an object-oriented language named Java. There are three reasons we select these projects: a) the data sets is open, therefore, it is easy for other researchers to gain these data to reproduce our work; b) the project is the earlier version or with less instances, which is suitable for CCDP scenario; c) many researchers use these data sets to certify their SDP approaches.

```
Algorithm 2 One-to-One CPDP with Common Metrics
Input: Source data set: $Tr = \{Tr_1, Tr_2, \cdots, Tr_p\}$;
       Target data set: $Te = \{Te_1, Te_2, \cdots, Te_q\}$;
       Classification approaches $clf$;
Output: Performance measures of CPDP:
       $PM=\{PM_1, PM_2, \cdots, PM_{p\times q}\}$
Approach:
 1: Let filtered training data $PM = \varnothing$;
 2: for $Te_i$ in $Te$:
 3:     for $Tr_j$ in $Tr$:
 4:         $FTrf_j$, $FTrl_j$ = iForest($Tef_i$) // filter the source data
 5:         $clf.fit(FTrf_j, FTrl_j)$  // fit a classification model
 6:         $pre\_Tel_i = clf.predict(Tef_i)$//predict the target labels
 7:         $PM_{ij}=evaluation(Tel_i, pre\_Tel_i)$  //evaluation results
 8:         $PM \leftarrow PM_{ij}$ //gain performance meauses
 9:     end for
10: end for
11: return $PM$.
```

Figure 2. Pseudocode of One-to-One CPDP with Common Metrics

There are 20 metrics in every data set. The project name and version, instance number and defect rate of every project are list in the first three columns in Tab. I. Before applying these data sets, we do some preprocessing: a) we convert the numeric labels to binary labels. The label value, which is larger than 0, is labeled as 1 meaning defects exist in the modules. Otherwise, the label labeled 0; b) if there are duplicate instances in a data set, we just keep one; c) if there are ambiguous instances with same feature values but different label values, we delete all of them. The reasons are as follows: a) we make classification models in the paper; b) if the duplicate instances are noisy, they may decrease the performance of CCDP; c) the ambiguous samples influence the judgement for iForest filter model or classification model, if these instances play a decisive role. The left instance numbers and defect rates after the preprocessing procedure is list in the last two columns of Tab. I.

B. Expenriments Design

We use 5 classical and different classifiers. LR is based on statistical learning [13]. NB is based on probability theory. DT belongs to decision tree classifier. *k*-NN is a lazy learner and RF is a popular ensemble methods.

The experimental environment is an Intel(R) Core (TM) i7-6700U CPU @ 3.40 GHz 3.41 GHz 16.0 GB RAM desktop running Windows 10 (64 bit) and simulation software Python (Version 3.6). Default values of all the classifiers from sklearn package are used. To avoid the randomness of RF, DT, iForest, k-means, we run our code 30 times. In total, 15*14*5*30 model are built.

To valid our approach, we set three research questions and 5 group experiments:

TABLE I. DETAILS OF 15 DATA SETS BEFORE AND AFTER PREPROCESS

Project	Instances Number[b]	Defect Rate[b]	Instances Number[c]	Defect Rate[c]
ant-1.3[a]	125	16.00%	125	16.00%
arc-1	234	11.54%	213	11.74%
camel-1.0	339	3.83%	327	3.98%
e-learning-1	64	7.81%	57	8.77%
jedit-3.2	272	33.09%	268	33.58%
log4j-1.0	135	25.19%	135	25.19%
lucene-2.0	195	46.67%	191	47.12%
poi-1.5	237	59.49%	203	60.10%
prop-6	660	10.00%	377	8.49%
redaktor-1	176	15.34%	169	14.79%
synapse-1.0	157	10.19%	153	10.46%
system-1	65	13.85%	63	14.29%
tomcat-6.0	858	8.97%	796	9.67%
xalan-2.4	723	15.21%	694	15.85%
xerces-init	162	47.53%	146	44.52%

a. Before symbol "-", *ant* stands for the project name, after symbol "-", *1.3* stands for the project version. The meaning of the first column is similar with "ant-1.3".
b. Number and defect rate of original instances of a project.
c. Number and defect rate of preprocessed instances of a project.

RQ1: Can iForest approach be used for simplifying training data for CPDP? Experiment 1: Build O2O CCDP_CM by the LR classifier with iForest filter as Group1. Build O2O CCDP_CM by LR classifier without any filter as Group2.

RQ2: If the answer of RQ1 is "Yes", which classifier performs best on iForest filter? Experiment 2: Build O2O CCDP_CM by the LR, DT, k-NN, NB and RF classifiers with iForest filter as Group3.

RQ3: If the answer of RQ1 is "Yes", Can iForest approach as a filter select more similar and useful training data than representative NN filter and Peter filter? Experiment 3: Retain the results of Group 3. Conduct O2O CCDP_CM by 5 classifiers with NN filter and Peter filter marked as Group4 and Group5, respectively.

To indicate the results from the models to answer the RQ1 and RQ2, we focus on mean values of AUC, G-Measure, F-Measure, Balance. Therefore, 15*14*5* 4 results were gained.

AUC, has been used in many models is not affected by class imbalance problem The AUC value is from 0 to 1, when it is larger than 0.5 means the model is valid. The values G-Measure, G-Mean, F-Measure and Balance are range from 0 to 1. The larger of the value, the better of the classification model.

C. *Results And Discussion*

To answer RQ1, experiment 1 was performed. We select the mean AUC, G-Measure, G-mean, F1-Measure and Balance values on all target projects. The results are shown in Fig.3.

Except AUC values, G-Measure, G-mean, F1-Measure and Balance LR classifier with the iForest filter are all larger than without any filter. From the results, we get the conclusion that iForest filter can simplify training data to some extent. The effect to the performance will be discussed in next part.

To answer RQ2, Experiment 2 was carried out. The AUC performance values of classifiers on every target projects are depicted in Fig.4.

From Fig. 4, it is noted that RF performs best, while NB performs worst of the five classifiers. At first sight, our result is contrary to the conclusion from [20]. They think AUC is well of large data sets for RF and of small data sets for NB. Actually, we used data sets from PROMISE Data Repository. The largest samples size of the selected data sets is 796 from project tomcat-6.0. All the data sets in the paper belong to "small" data sets. Therefore, in our paper, the best suitable classifier is RF about AUC.

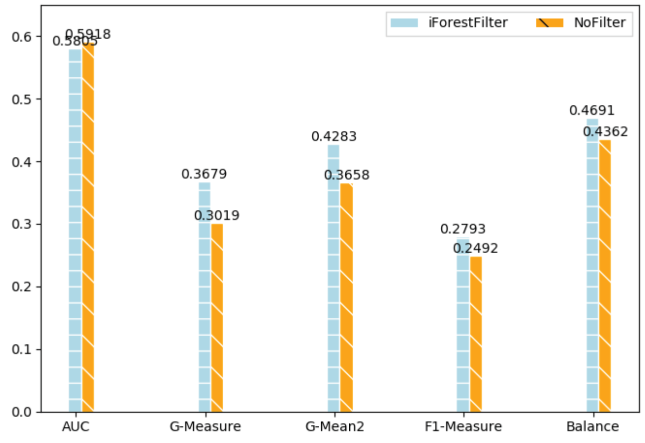

Figure 3. Mean Measure Values On All Target Projects via LR Classifier

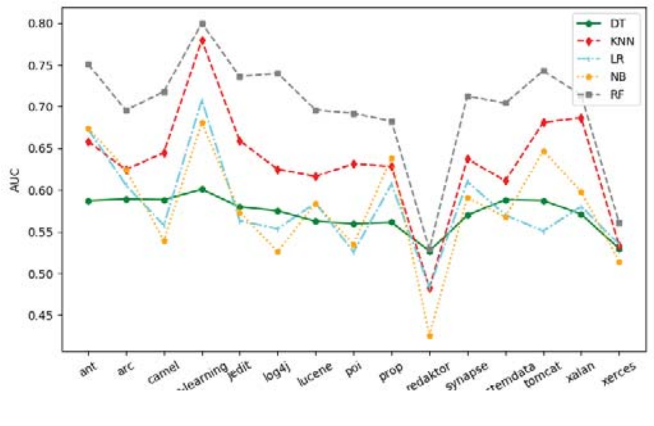

Figure 4. Mean AUCs of Different Classifiers on Different target projects

From Fig. 4, DT classifier is robust about AUC on every target project. However, the values of AUC is a bit low. All the classifiers perform best on e-learning-1 project, while they all perform worst on redactor-1 project, whose AUC value below to 0.5. This means the filter training data can help improve the performance of CCDP on e-learning-1 data sets. The possible reasons are: a) for e-learning-1 data set, though the instances number is the least, the feature values to relatively fit an suitable iForest filter model to filter the noisy data and retain more useful data from the source projects; b) for redactor-1 data set, the feature values span too much, so when it fit an iForest model, it may predict useful values to abnormal, i.e., redactor data sets may delete useful information to decrease the performance of the classification models.

From Fig. 3 and 4, we suspect that the worst AUC performance on redactor-1 data sets may influence the average AUC of all projects. Therefore, LR with the iForest filter to simplify training data performances worse than LR without any filter to use all original training data.

To answer RQ3, we carried out Experiment 3. The average values of all the projects list in Tab. II. The standard deviation values of the performance measures is shown in Tab. III.

TABLE II. MEAN VALUES OF PERFROMANCE MEASURES AND THE WIN\TIE\LOSS RESULTS FROM FIVE CLASSIFIERS (BOLD VALUES MEANS THE BEST VALUES OF THE PERFORMANCE IN THE CLASSIFIER)

Measures	Filters	DT	KNN	LR	NB	RF
AUC	iForest	**0.5718**	**0.6332**	**0.5805**	0.5813	**0.6982**
	Burak	0.5683	0.5758	0.5660	**0.6375**	0.6930
	Peter	0.5709	0.5634	0.4761	0.4981	0.6464
G-Measure	iForest	**0.4908**	0.4063	**0.4691**	0.5273	0.4497
	Burak	0.4856	**0.4741**	0.4485	**0.5494**	**0.4582**
	Peter	0.4863	0.4050	0.4507	0.3763	0.4296
G-Mean	iForest	0.2796	0.2424	**0.2793**	0.2983	**0.2750**
	Burak	0.2711	0.2621	0.2508	**0.3211**	0.2677
	Peter	**0.2896**	**0.2690**	0.2528	0.2420	0.2496
F1-Measure	iForest	**0.4717**	0.2791	**0.4283**	**0.5090**	0.3706
	Burak	0.4632	**0.4435**	0.3880	0.5377	**0.3861**
	Peter	0.4448	0.2713	0.3949	0.2256	0.3412
Balance	iForest	**0.4180**	0.2347	**0.3679**	0.4768	0.3151
	Burak	0.4075	**0.3830**	0.3266	**0.4951**	**0.3297**
	Peter	0.3991	0.2395	0.3484	0.1924	0.2878
w\t\l	iF-B[a]	4\0\1	1\0\5	5\0\0	1\0\4	2\0\3
	iF-P[b]	4\0\1	3\0\2	5\0\0	5\0\0	5\0\0

a. Before symbol "-", iF stands for iForest filter approach, after symbol "-", B stands for Burak filter. This means the iForest filter compares with the Burak filter.
b. The iF-P means the iForest filter compares with the Peter filter.

From Tab. II and III, we can verify the phenomenon from Fig. 4: a) DT classifier with the iForest filter is a robust classifier for O2O CCDP_CM. b) NB classifier with our filter performs worst in all performance measures. c) For AUC values, 4 of 5 classifiers with our approach performs best, and the best value got by the RF classifier with our filter.

Of course, there are some other results we can get in Tab. II and III. From the win\loss\tie, we can see that LR with iForest filter and DT with iForest filter are the top 2 in 5 classifiers. While k-NN with iForest filter is worse than NN fitler and Peter filter. The last two filters both use k-NN to select instances and the selected instances is more suitable for the k-NN classification model than the selected by iForest filter to a great extent. For NB and RF, the comparisons are different. For Peter filter, iForest filter performances always better that it. However, for Burak filter, iForest filter mainly performance worse than it. Totally, the iForest filter can get a better results than Peter filter on five classifiers and performance better than Burak filter on LR and DT classifiers.

TABLE III. STANDARD DEVIATION VALUES OF PERFORMANCE MEASURES FROM FIVE CLASSIFIERS (BOLD VALUES MEAN THE STANDARD DEVIATION VALUES OF THE PERFORMANCE IN THE CLASSIFIER ARE LOWER THAN 0.05.)

Measures	Filters	DT	KNN	LR	NB	RF
AUC	iForest	**0.0214**	0.0663	0.0561	0.0678	0.0691
	Burak	**0.0236**	**0.0324**	0.0631	0.0672	0.0682
	Peter	**0.0323**	0.0530	0.0653	0.0508	0.0637
G-Measure	iForest	**0.0281**	**0.0404**	**0.0327**	0.0570	**0.0422**
	Burak	**0.0309**	**0.0335**	**0.0357**	0.0562	**0.0368**
	Peter	**0.0401**	0.0575	**0.0476**	**0.0349**	**0.0453**
G-Mean	iForest	0.0745	0.0619	0.0625	0.0983	0.0641
	Burak	0.0683	0.0676	0.0560	0.0794	0.0563
	Peter	0.0800	0.1024	0.0911	0.1163	0.0597
F1-Measure	iForest	**0.0433**	0.0807	0.0511	0.0708	0.0804
	Burak	**0.0484**	0.0517	0.0637	0.0629	0.0649
	Peter	0.0584	0.0999	0.0779	0.0658	0.0755
Balance	iForest	**0.0483**	0.0743	0.0561	0.0755	0.0755
	Burak	0.0533	0.0567	0.0640	0.0773	0.0633
	Peter	0.0643	0.1034	0.0824	0.0639	0.0790

V. CONCLUSION AND FUTURE WORK

The purpose of us is to simplify the training data for CCDP to improve the model performance. Considering the common filter approaches for O2O CCDP depend on distance and density, and the anomaly detection approach called iForest can find noisy data, which is different from the distance or density approaches. We conduct 1050 (15*14*5) O2O CCDP models by iForest filter combined with six classifiers. The data sets are from PROMISE Data Repository. Meanwhile, 2100 O2O CCDP models by Burak filter and Peter filter combined with six classifiers, respectively. From the results and discussion above, we can conclude that our iForest filter is feasible and can improve the performances CCDP of some classifiers, especially of LR, NB and RF. From the results, we make two proposals: a) building DT, NB and RF models, iForest filter and Burak filter are recommended; b) conducting LR models, iForest filter should be chosen.

Though there are some possible reasons to explain the poor performance of iForest. Finding the factors which influence the iForest filter is a key issue. In the paper, we did not consider the high dimensionality of the features and data imbalance. Exploring these situations on iForest filter is the future work and direction.

ACKNOWLEDGMENT

We are very thankful for the data sets providers to make us do some research on software defect prediction. Meanwhile, we thank the reviewers for their valuable comments.

REFERENCES

[1]. Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, "An investigationon the feasibility of cross-project defect prediction," Automated Software Engineering, vol. 19, pp. 167–199, 2012.

[2]. T. Menzies, J. Greenwald, and A. Frank. "Data mining staticcode attributes to learn defect predictors," IEEE Transactionson Software Engineering, vol. 33, no. 1, pp. 2–13, 2007.

[3]. C. Catal, and B. Diri, "A systematic review of software fault prediction studies," Expert Systems with Applications, vol. 36 no. 4, pp. 7346-7354, 2009.

[4]. G. Canfora, A. D. Lucia, M. D. Penta, R. Oliveto, A. Panichella, and S. Panichella, "Multi-objective cross-project defect prediction," In Proceedings of the 6th IEEE International Conference on Software Testing, Verification and Validation (ICST), 2013.

[5]. Y. Ma, G. Luo, X. Zeng, and A. Chen. "Transfer learning for cross company software defect prediction," Information and SoftwareTechnology, vol. 54, no. 3, pp. 248–256, 2012.

[6]. 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 the 7th joint meeting of the European Software Engineering Conference (ESEC) and the ACM SIGSOFT Symposium on The Foundations of Software Engineering(FSE), ACM, pp. 91-100, 2009.

[7]. F. Peters, T. Menzies, and A. Marcus, "Better cross company defect prediction," Mining Software Repositories (MSR), 10th IEEE Working Conference on IEEE, pp. 409-418, 2013.

[8]. 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.

[9]. H. Peng, B. Li,D. Zhang, and Y. Ma, "Simplification of Training Data for Cross-Project Defect Prediction," Computer Science, 2014.

[10]. K. Kawata, S. Amasaki, and T. Yokogawa. "Improving Relevancy Filter Methods for Cross-Project Defect Prediction," Software Engineering & Advanced Applications, IEEE, 2015.

[11]. M. Jureczko, and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," In Proceedings of the 6th International Conference onPredictive Models in Software Engineering (PROMISE), ACM, 2010.

[12]. T. Menzies, A. Butcher, A. Marcus, T. Zimmermann, and D. Cok. "Local vs. global models for effort estimation anddefect prediction," In Proceedings of the 26th IEEE/ACMInternational Conference on Automated Automated Software Engineering (ASE), pp. 343–351, 2011.

[13]. A. E. Camargo Cruz, and K. Ochimizu, "Towards logistic regression models for predicting fault-prone code across software project,". In Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 460–463. IEEE Computer Society, 2009.

[14]. N. Nagappan, T. Ball, and A. Zeller. "Mining metrics to predict component failures," In Proceedings of the ACM/IEEE28th International Conference on Software engineering, 2006.

[15]. L.C. Briand, W.L. Melo, and J. Wüst, "Assessing the applicability of fault proneness models across object-oriented software projects," IEEE Transactoins on Software Engineering, vol. 28, no. 7, pp.706-720, 2002.

[16]. J. Nam, W. Fu, S. Kim, T. Menzies, and L. Tan, "Heterogeneous defect prediction," IEEE Transactions on Software Engineering,vol. 1-1,pp.99, 2015.

[17]. S. Herbold, "Training data selection for cross-project defect prediction," International Conference on Predictive Models in Software Engineering, 2013.

[18]. S. Herbold, A. Trautsch, and J. Grabowski, "Global vs. local models forcross-project defect prediction," Empirical Software Engineering, 2016.

[19]. N. Bettenburg, M. Nagappan, and A. Hassan. "Think locally, act globally: Improving defect and effort prediction models," In Proceedings of the 9th IEEE Working Conference onMining Software Repositories (MSR), pp. 60–69, 2012.

[20]. C. Catal, and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," Information Sciences, vol. 179, no.8, pp. 1040-1058, 2009.

[21]. F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation-Based Anomaly Detection," ACM Transactions on Knowledge Discovery from Data, vol. 6, no. 1, pp. 1-39, 2012.