Soft Computing (2018) 22:3461–3472
https://doi.org/10.1007/s00500-018-3093-1

METHODOLOGIES AND APPLICATION

Cross-company defect prediction via semi-supervised clustering-based data filtering and MSTrA-based transfer learning

Xiao Yu[1,2] · Man Wu[3] · Yiheng Jian[4] · Kwabena Ebo Bennin[5] · Mandi Fu[3] · Chuanxiang Ma[2,6]

Published online: 8 March 2018

Abstract
Cross-company defect prediction (CCDP) is a practical way that trains a prediction model by exploiting one or multiple projects of a source company and then applies the model to a target company. Unfortunately, larger irrelevant cross-company (CC) data usually make it difficult to build a prediction model with high performance. On the other hand, brute force leveraging of CC data poorly related to within-company data may decrease the prediction model performance. To address such issues, we aim to provide an effective solution for CCDP. First, we propose a novel semi-supervised clustering-based data filtering method (i.e., SSDBSCAN filter) to filter out irrelevant CC data. Second, based on the filtered CC data, we for the first time introduce multi-source TrAdaBoost algorithm, an effective transfer learning method, into CCDP to import knowledge not from one but from multiple sources to avoid negative transfer. Experiments on 15 public datasets indicate that: (1) our proposed SSDBSCAN filter achieves better overall performance than compared data filtering methods; (2) our proposed CCDP approach achieves the best overall performance among all tested CCDP approaches; and (3) our proposed CCDP approach performs significantly better than with-company defect prediction models.

Keywords Cross-company defect prediction · Transfer learning · SSDBSCAN · Multi-source TrAdaBoost

1 Introduction

Software defect prediction is one of the most important software quality assurance techniques. It aims to detect the defect proneness of new software modules via learning from defect data. Therefore, defect prediction is often used to help to reasonably allocate limited development and maintenance resources (Shepperd et al. 2014; Song et al. 2011; Malhotra 2015).

Support vector machine (Elish and Elish 2008; Gray et al. 2009; Yan et al. 2010), neural network (Arar and Ayan 2015; Vashisht et al. 2015; Erturk and Sezer 2016), extreme learning machine (Mesquita et al. 2016), decision tree (Wang et al. 2012; Seliya and Khoshgoftaar 2011), Naïve Bayes (Dhanajayan and Pillai 2016), dictionary learning (Jing et al. 2014) and ensemble learning (Laradji et al. 2015; Siers and Islam 2015; Sun et al. 2012) paved the way for classification-based methods in the field of defect prediction. These methods use software metrics to properly predict whether a module is defect-prone or not, but they are usually confined to within-company defect prediction (WCDP). WCDP works well if sufficient data are available to train a defect prediction model. However, it is difficult for a new company to perform WCDP if there are limited historical data. Cross-company defect prediction (CCDP) is a practical approach to solve the problem. It trains a prediction model by exploiting one or multiple

Communicated by V. Loia.

✉ Chuanxiang Ma
mcx838@hubu.edu.cn

Xiao Yu
1451185513@qq.com

1 School of Computer, Wuhan University, Wuhan, China

2 School of Computer Science and Information Engineering, Hubei University, Wuhan, China

3 Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, China

4 School of Information and Electronics, Beijing Institute of Technology, Beijing, China

5 Department of Computer Science, City University of Hong Kong, Hong Kong, China

6 Educational Informationalization Engineering Research Center of Hubei Province, Wuhan, China

projects of a source company and then applies the model to a target company project (Jing et al. 2015).

1.1 Motivation

Most existing CCDP approaches (Jing et al. 2015; Briand et al. 2002; Zimmermann et al. 2009; Turhan et al. 2009; Peters et al. 2013; Ryu et al. 2015; Kawata et al. 2016; Ma et al. 2012; Nam et al. 2013; Shukla et al. 2016) focus on using only cross-company (CC) data to build a proper prediction model. Unfortunately, larger irrelevant CC data usually make it difficult to build a prediction model with high performance (Chen et al. 2015). In fact, if there are limited amount of labeled WC data, the data are not enough to perform WCDP, but it may help a lot to improve the performance of CCDP. Another scenario is that companies may already have their defect prediction models in place, and making use of CC data may improve the performance of these models (Turhan et al. 2013).

The challenges of performing CCDP with limited amount of labeled WC data usually include:

(1) How to weaken the impact of irrelevant CC data to improve the performance of CCDP.

The ability to transfer knowledge from a source company to a target company depends on how they are related. The stronger the relationship, the more usable will be the CC data. The performance of CCDP is generally poor because of larger irrelevant CC data. The irrelevant data have bad effects on the prediction outcome (Yao and Doretto 2010). To solve the problem, one of the most efficient methods is filter technology. For example, Turhan et al. (2009) and Peters et al. (2013) proposed the NN filter and the Peters filter to select the CC instances which are mostly similar to WC data as the training dataset. However, since CC data are collected from different development environments or application fields, the labels of CC data may be in conflict with WC data even if they are close in distance, which tends to generate false prediction results (Chen et al. 2015). More specifically, if the defect-prone WC instance selects the negative defect-free CC instances, it may result in a low rate of recall. On the other hand, if the defect-free WC instance selects the negative defect-prone CC instances, it could lead to a high false alarm rate.

(2) How to avoid negative transfer when leveraging multiple CC data.

The effectiveness of the transfer is affected by the relationship between CC data and WC data. Rather than improving the performance, brute force leveraging of CC data poorly related to WC data may decrease the prediction model performance. Chen et al. (2015) developed the double transfer boosting (DTB) approach for CCDP. DTB approach merges all CC data as a source, relies on only the source, and therefore is intrinsically vulnerable to negative transfer.

1.2 Contribution

Considering the above challenges, in this paper, we aim to provide an effective solution for CCDP, and the contributions of our paper are summarized as follows.

(1) Before applying transfer learning methods for CCDP, we propose a novel data filtering method (i.e., SSDBSCAN filter) to filter out irrelevant CC data. The process of SSDBSCAN filter is based on semi-supervised density-based clustering (SSDBSCAN) algorithm (Lelis and Sander 2009). The SSDBSCAN filter combines limited amount of labeled WC data, unlabeled WC data and CC data, finds clusters by using SSDBSCAN algorithm and selects the CC instances that have the same class label as the WC instances in the same cluster. In this case, we use the class information (i.e., an instance is defect-prone or defect-free) of the limited amount of labeled WC data and CC data, which avoids WC instance to select some CC instances with different class label.
(2) Based on the filtered CC data, we for the first time introduce the multi-source TrAdaBoost (MSTrA) algorithm (Yao and Doretto 2010), an effective transfer learning method to perform CCDP. MSTrA trains and combines a set of weak prediction models to build a stronger ensemble defect prediction model by using not only CC data but also limited amount of labeled WC data. In each training round, MSTrA transfers knowledge not from one but from multiple CC data to avoid negative transfer and reduces the weights of irrelevant instances in CC data to weaken the impact of irrelevant CC data continuously.

We call the entire approach for CCDP as MSTrA+. We evaluate MSTrA+ on 15 public datasets selected from the PROMISE data repository (Boetticher et al. 2007) with three performance metrics. The experimental results demonstrate that the proposed approach outperforms several representative CCDP approaches.

1.3 Organization

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 describes the proposed MSTrA+ approach for CCDP. Sections 4 and 5 show the experiment setup and experiment results, respectively. Section 6 discusses the potential threats to validity. Finally, Sect. 7 addresses the conclusion and points out the future work.

2 Related work

In this section, we briefly review the existing cross-company and cross-project defect prediction approaches. These approaches can be categorized into two main types: defect prediction only using CC data (Arar and Ayan 2015; Vashisht et al. 2015; Erturk and Sezer 2016; Mesquita et al. 2016; Wang et al. 2012; Seliya and Khoshgoftaar 2011; Dhanajayan and Pillai 2016; Jing et al. 2014), defect prediction using not only CC data but also limited amount of labeled WC data (Laradji et al. 2015; Siers and Islam 2015).

2.1 Defect prediction using only CC data

In order to solve the problem of new companies that have too limited historical data for better WCDP performance, cross-project (CP) and cross-company defect prediction were proposed.

Briand et al. (2002) used logistic regression and MARS (multivariate adaptive regression splines) models to learn a defect predictor, which is also the earliest work on CCDP. Zimmermann et al. (2009) studied CCDP models on 12 real-world applications datasets. Their results indicate that CCDP is still a serious challenge because of the different distribution between WC data and CC data. In order to narrow the distribution gap, there are two mainstream ways.

The first one is to apply filter technology to find out the best suitable training data (e.g., Turhan et al. 2009; Peters et al. 2013; Ryu et al. 2015; Kawata et al. 2016). For example, Turhan et al. (2009) proposed a nearest neighbor (NN) filter to select the most similar k CC instances for every WC instance as the filtered CC data. Peters et al. (2013) introduced the Peters filter. The Peters filter lets the CC instances find their nearest WC instances, and the ones nearest to their WC instances are selected for the final filtered CC data. It is worthy of note that different from our proposed SSDBSCAN filter, all the data filtering methods of the above literatures do not use the class information. Since the labels of CC data may be in conflict with WC data even if they are close in distance, the performance of these data filtering methods has still been challenged. Our SSDBSCAN filter considers using the class information of limited amount of labeled WC data and CC data, so that these WC instances tend to avoid selecting some CC instances with different class label.

The second mainstream way is to design effective defect predictor based on transfer learning techniques (e.g., Ma et al. 2012; Nam et al. 2013; Shukla et al. 2016). For instance, Ma et al. (2012) proposed Transfer Naïve Bayes (TNB) approach, Nam et al. (2013) proposed a novel transfer defect learning approach, TCA+, by extending TCA (transfer component analysis). Another challenge in CCDP is that the set of metrics between the source company data and target company data is usually heterogeneous. Jing et al. (2015) proposed a unified metric representation (UMR) for the data of source and target companies and introduced canonical correlation analysis (CCA), an effective transfer learning method, into CCDP to make the data distributions of source and target companies similar. The approaches above focus on using only CC data to build predictors. Considering there are limited amount of labeled WC data, the data are not enough to perform WCDP, but it may help a lot to improve the performance of CCDP.

2.2 Defect prediction with limited amount of labeled WC data

Turhan et al. (2013) introduced a mixed model of within and cross-data for CCDP to investigate the merits of using mixed project data for binary defect prediction. Results show that when there is limited project history, mixed model for CCDP can achieve good performance which can be comparable to WCDP. It provided a new idea to CCDP that the use of a small amount of labeled WC data would be very valuable to improve the performance of CCDP.

Chen et al. (2015) introduced a novel approach named double transfer boosting (DTB) to narrow the gap of different distributions between CC data and WC data and to improve the performance of CCDP by reducing negative samples in CC data. However, it merges all CC data as one source and the result only relies on the single source so that it is prone to negative transfer, which is exactly what we will solve in this paper.

3 Methodology

In this section, we present our MSTrA+ approach for CCDP. MSTrA+ is built based on mixed training data consisting of CC data and limited amount of labeled WC data. Its main steps are as follows: (1) in order to narrow the distribution gap between CC data and WC data, MSTrA+ firstly uses the proposed SSDBSCAN filter to filter out irrelevant CC data and then uses data gravitation (Peng et al. 2009) for reweighting the whole distribution of the filtered CC data to fit WC data; (2) MSTrA+ mixes limited amount of labeled WC data with reweighted CC data to build the prediction model by using Multi-Source TrAdaBoost algorithm. The framework of the proposed MSTrA+ approach is shown in Fig. 1.

3.1 Data Preprocessing

3.1.1 SSDBSCAN filter

Previous work (Turhan et al. 2009) found that using raw CC data directly would increase false alarm rate due to larger

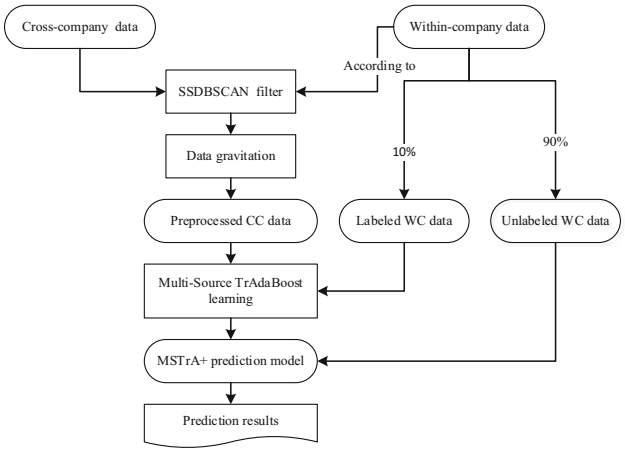

Fig. 1 The framework of the proposed MSTrA+ approach

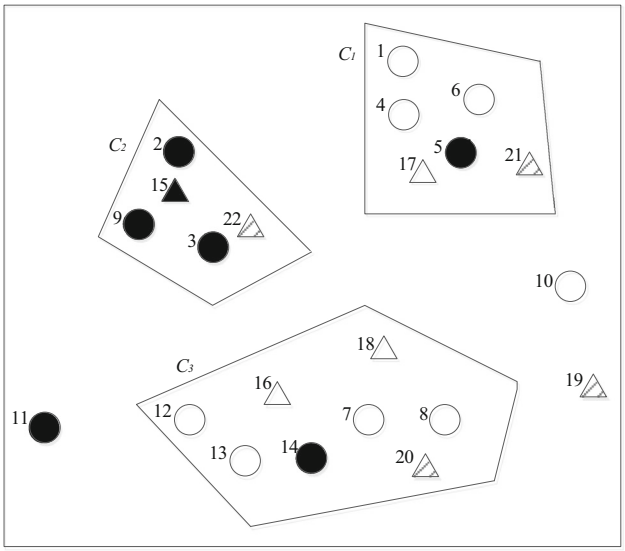

Fig. 2 Resulting clusters of a set of instances using the SSDBSCAN algorithm

irrelevant instances in CC data; thus, data preprocessing is necessary before building the prediction model. Therefore, the SSDBSCAN filter is proposed to filter out irrelevant CC data.

The process of SSDBSCAN filter is based on the SSDBSCAN algorithm which is proposed by Lelis and Sander (2009). SSDBSCAN uses the class information of the limited amount of labeled WC data to find clusters, so that each instance in the limited amount of labeled WC data is contained in a density-based cluster and all pairs of the instances with different labels belong to different clusters. Since instances in a cluster are similar to one another, yet dissimilar to instances in other clusters, those CC instances with the same class label as the WC instances in the same cluster have the similar defect distribution characteristics to these WC instances. Therefore, the SSDBSCAN filter assumes that the CC instances which have the same class label as the WC instances in the same cluster are the most valuable instances in CC data.

The details of SSDBSCAN filter are as follows.

(1) Combine limited amount of labeled WC data, unlabeled WC data, and CC data,
(2) Find clusters by using SSDBSCAN algorithm,
(3) Collect the CC instances that have the same class label as the WC instances in the same cluster.

Since each cluster consists of at least one labeled WC instance, we can filter out the CC instances that have the different class label from the WC instance in the same cluster even if they are close in distance, which tends to avoid false prediction results.

Figure 2 shows an illustrative example of the SSDBCAN filter. There are 22 instances generated by hand, where "●" represents the defect-prone CC instance, "○" represents the defect-free CC instance, "▲" represents the defect-prone CC instance, "△" represents the defect-free CC instance, and "⧋" represents the unlabeled WC instance. These instances are partitioned into three clusters by using the SSDBSCAN algorithm, namely, $C_1 = \{x_1, x_4, x_5, x_6, x_{17}, x_{21}\}$, $C_2 = \{x_2, x_3, x_9, x_{15}, x_{22}\}$, and $C_3 = \{x_7, x_8, x_{12}, x_{13}, x_{14}, x_{16}, x_{18}, x_{20}\}$. Take the cluster C_1 for example, since the CC instances x_1, x_4 and x_6 have the same class label as the labeled WC instance x_{17}, these CC instances are selected to form the final CC training data. Since the CC instance x_5 has the different class label from the labeled WC instance x_{17}, the CC instance is discarded. The CC instances in the clusters C_2 and C_3 are selected in the same manner. Therefore, the final CC training instances consist of x_1, x_2, x_3, x_4, x_6, x_7, x_8, x_9, x_{12} and x_{13}.

3.1.2 Data gravitation

Then, the entire distribution of filtered CC data is changed by applying the data gravitation method (Peng et al. 2009). Suppose that an instance x_i can be described by $x_i = (a_{i1}, a_{i2}, \ldots, a_{ik})$, where a_{ij} is the jth attribute value of the ith instance and k is the number of the attributes.

(1) We compute two vectors, $\text{Max} = \{\max_1, \max_2, \ldots, \max_k\}$ and $\text{Min} = \{\min_1, \min_2, \ldots, \min_k\}$ to represent the attribute value distribution of WC data, where $\max_i$ is the maximum value of the ith attribute, $\min_i$ is the minimal value of the ith attribute.

(2) For each instance x_i in CC data, the degree s_i of similarity to WC data is computed according to Eq. (1)

$$s_i = \sum_{i=1}^{k} h\left(a_{ij}\right) \tag{1}$$

where a_{ij} is the jth attribute value of the instance x_i, $h(a_{ij}) = 1$, if $\min_j \leqslant a_{ij} \leqslant \max_j$; otherwise, $h(a_{ij}) = 0$.

(3) The weight w_i of instance x_i in CC data can be calculated by Eq. (2) according to the formulation of data gravitation (Peng et al. 2009).

$$w_i = s_i/(k - s_i + 1)^2 \tag{2}$$

where k is the number of the attributes.

According to this formula, the weight w_i of instance x_i shows the similarity of x_i to WC data, and the greatest w_i will be assigned when $s_i = k$. Therefore, the entire distribution of the filtered CC data is reweighted to be close to WC data.

3.2 Multi-source TrAdaBoost learning

Let $\mathrm{D}^{\mathrm{SK}} = \{(x_1^{\mathrm{S1}}, c_1^{\mathrm{S1}}), \ldots, (x_n^{\mathrm{SK}}, c_n^{\mathrm{SK}})\}$ be the kth CC data, where n is the number of instances in the kth CC data, $c_i^{\mathrm{SK}} \in$ {true, false} is the class label of instance x_i^{SK}. Let $\mathrm{D}^{\mathrm{T}} = \{(x_1^{\mathrm{T}}, c_1^{\mathrm{T}}), \ldots, (x_m^{\mathrm{T}}, c_m^{\mathrm{T}})\}$ be limited amount of labeled WC data, where m is the number of instances in labeled WC data, c_i^{T} is the class label of instance x_i^{T}.

During SSDBSCAN filter and data gravitation, filtered CC data $\mathrm{D}^{\mathrm{S1}}, \ldots \mathrm{D}^{\mathrm{SN}}$ and labeled WC data D^{T} are assigned different weight according to Eq. (2).

In each training round, combine the kth CC data and the limited amount of labeled WC data to train a candidate weak prediction model. In our paper, we choose Naïve Bayes (Lewis 1998) as the base prediction model due to its effectiveness in defect prediction (Hall et al. 2012). The final weak prediction model $\boldsymbol{f_t}(x)$ in tth iteration is one of the candidate weak prediction models which has the minimal prediction error on labeled WC data. In other words, every weak prediction model is selected from CC data that appear to be the most closely related to WC data. The prediction error function is defined according to Eq. (3).

$$\varepsilon_t = \sum_{j=1}^{m} \frac{w_i^t \left| f_t\left(x_i\right) - c_i \right|}{\sum_{i=1}^{m} w_i^t} \tag{3}$$

$$\text{Set } \beta_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t} \tag{4}$$

In this way, we import knowledge not from one but from multiple sources, thus decreasing the risk for negative transfer. In each training round, the instances in WC data are given more importance if the instances are misclassified. They are believed to be the "most informative" for the next round, so the weight of the misclassified instances is increased according to Eq. (5).

$$w_i^{\mathrm{T}} = w_i^{\mathrm{T}}\ e^{\beta_t \left| f_t\left(x_i^{\mathrm{T}}\right) - c_i^{\mathrm{T}} \right|} \tag{5}$$

The instances in CC data are given less importance if the instances are misclassified. They are believed to be the most dissimilar to WC data, so the weight of the misclassified instances are decreased according to Eq. (6) in order to weaken their impacts in the next round through multiplying the Hedge (β) defined in Eq. (7).

$$w_i^{\mathrm{SK}} = w_i^{\mathrm{SK}}\ e^{-\beta_s \left| f_t\left(x_i^{\mathrm{SK}} - c_i^{\mathrm{SK}}\right) \right|} \tag{6}$$

$$\beta_{\mathrm{s}} = \frac{1}{2} \ln \left(1 + \sqrt{2 \ln \frac{n_s}{M}}\right) \tag{7}$$

Once the weights of all misclassified instances are updated, the weights of all instances are normalized so that their sum remains 1.

After several iterations, the instances in CC data that fit WC data will have larger training weights, while the instances in CC data that are dissimilar to WC data will have lower weights. The instances in CC data with larger training weights intend to build a better prediction model. The final prediction model $F(x)$ can be expressed as follows:

$$F(x) = \operatorname{sign}\left(\sum_t \beta_t \boldsymbol{f_t}\left(x\right)\right) \tag{8}$$

Algorithm 1 presents the pseudo-code of the MSTrA+ approach to perform CCDP.

Algorithm 1. MSTrA+ approach

Input: filtered CC data $\mathrm{D}^{\mathrm{S1}}, \ldots, \mathrm{D}^{\mathrm{SN}}$, limited amount of labeled WC data D^{T}, and the maximum number of iterations M
Output: a prediction model $F(x)$
1. Initialize a weight vector $(\mathbf{w}^{\mathrm{S1}}, \ldots, \mathbf{w}^{\mathrm{SN}}, \mathbf{w}^{\mathrm{T}})$ using Eq. (2)
2. **for** t = 1, …, M **do**
3. Empty the set of candidate weak prediction models
4. Normalize to 1 the weight vector $(\mathbf{w}^{\mathrm{S1}}, \ldots, \mathbf{w}^{\mathrm{SN}}, \mathbf{w}^{\mathrm{T}})$
5. **for** k = 1, …, N **do**
6. Train the candidate weak prediction model $\boldsymbol{f}_t^{\mathrm{K}}(\mathrm{x})$ over the combined data $\mathrm{D}^{\mathrm{SK}} \cup \mathrm{D}^{\mathrm{T}}$, using weight $(\mathrm{w}_i^{\mathrm{SK}}, \mathrm{w}^{\mathrm{T}})$
7. Compute the error of $\boldsymbol{f}_t^{\mathrm{K}}(\mathrm{x})$ on D^{T} using Eq. (3)
8. **end for**
9. Find the weak prediction model $\boldsymbol{f}_t(\mathrm{x})$ which has the minimal error
10. Update weights vector $(\mathbf{w}^{\mathrm{S1}}, \ldots, \mathbf{w}^{\mathrm{SN}}, \mathbf{w}^{\mathrm{T}})$ for the next round using Eq. (5) and Eq. (6)
11. **end for**
12. **return** $F(x) = \operatorname{sign}(\sum_t \beta_t \boldsymbol{f}_t\left(x\right))$

Table 1 Details of experiment datasets

No.	Name	#Instance	#Defects	%Defect	Description
1	*ant*	125	20	16	Open-source
2	*arc*	234	27	11.5	Academic
3	*camel*	339	13	3.8	Open-source
4	*elearn*	64	5	7.8	Academic
5	*jedit*	272	90	33.1	Open-source
6	*log4j*	135	34	25.2	Open-source
7	*lucene*	195	91	46.7	Open-source
8	*poi*	237	141	59.5	Open-source
9	*prop*	660	66	10	Proprietary
10	*redaktor*	176	27	15.3	Academic
11	*synapse*	157	16	10.2	Open-source
12	*system*	65	9	13.8	Open-source
13	*tomcat*	858	77	9	Open-source
14	*xalan*	723	110	15.2	Open-source
15	*xerces*	162	77	47.5	Open-source

4 Experiments setup

In this section, we describe the experimental setup in detail, including the experiment datasets, performance measures and research questions.

4.1 Dataset

In this experiment, we employ 15 available and commonly used datasets which can be obtained from PROMISE repository (Boetticher et al. 2007). The details about the datasets are shown in Table 1, where #Instance represents the number of instances, #Defects represents the total number of faults in the release, and %Defect represents the percentage of defect-prone instances. Each instance in the 15 datasets has the same 20 independent code attributes, including the lines of code, weighted methods per class, depth of inheritance tree. A more detailed description of the 20 independent code attributes is listed in Table 2.

4.2 Performance measures

In the experiment, we employ three commonly used performance measures including PD, PF and G-measure. They are defined in Table 3 and summarized as follows.

(1) Possibility of detection (PD) is defined as the ratio of the number of defect-prone modules that are correctly predicted to the total number of defect-prone modules.

Table 2 Code attributes of the datasets

No.	Attribute	Description
1	*wmc*	Weighted methods per class
2	*dit*	Depth of inheritance tree
3	*noc*	Number of children
4	*cbo*	Coupling between object classes
5	*rfc*	Response for a class
6	*lcom*	Lack of cohesion in methods
7	*ca*	Afferent couplings
8	*ce*	Efferent couplings
9	*npm*	Number of public methods
10	*lcom3*	Lack of cohesion in methods
11	*loc*	Lines of code
12	*dam*	Data access metric
13	*moa*	Measure of aggregation
14	*mfa*	Measure of functional abstraction
15	*cam*	Cohesion among methods of class
16	*ic*	Inheritance coupling
17	*cbm*	Coupling between methods
18	*amc*	Average method complexity
19	*max_cc*	Maximum McCabe's cyclomatic complexity
20	*avg_cc*	Average McCabe's cyclomatic complexity

Table 3 Performance measures

		Actual	
		Yes	No
Predicted	Yes	TP	FP
	No	FN	TN
PD	$\frac{TP}{TP+FN}$		
PF	$\frac{FP}{FP+TN}$		
G-measure	$\frac{2\times PD\times(1-PF)}{PD+(1-PF)}$		

(2) Possibility of false alarm (PF) is defined as the ratio of the number of defect-prone modules that are incorrectly predicted to the total number of defect-free modules.
(3) G-measure is a trade-off measure that balances the performance between PD and PF. A good prediction model should have high PD and low PF, thus leading to a high G-measure.

4.3 Research questions

To assess our proposed MSTrA+ approach, this paper explores the following questions.

RQ1: Is the SSDBSCAN filter in MSTrA+ effective, compared with other data filtering methods for CCDP?

MSTrA+ mainly consists of two stages, data filtering stage and transfer learning stage. Data filtering stage selects the

most valuable CC data while transfer learning stage is added to build the prediction model. To investigate the effectiveness of our proposed SSDBSCAN filter, we compare the SSDBSCAN filter against two state-of-the-art data filtering methods (NN filter Turhan et al. 2009 and Peters filter Peters et al. 2013). Methods in comparison are provided below.

(1) NN filter is based on the widely used K-nearest neighbors (KNN) algorithm (Fukunaga and Narendra 1975) to filter out irrelevant CC data. It can find out the most similar $K \times N$ instances from CC data while N is the number of instances in WC data and K is the parameter of the KNN algorithm. In our experiment, we follow the original work (Turhan et al. 2009) to set K as 10.
(2) Peters filter first clusters the CC instances with the WC instances using the k-means algorithm (Jain 2010). Next, clusters containing at least one WC instance are kept and others are rejected. For each CC instance, we find its nearest WC instance. Finally, each WC instance chooses its closest CC instance as a candidate for the filtered CC data while rejecting all the others. With approximately 4400 instances in CC data and WC data, we follow the original work (Peters et al. 2013) to set k as 440 for k-means, i.e., one cluster per $r = 10$ WC + CC instances. We use this value for r since the NN filter chooses 10-nearest neighbors for each WC instance (Turhan et al. 2009).

SSDBSCAN filter requires only one parameter, a minimum number of points *MinPts*, which defines the density level of clusters. In this paper, we use $MinPts = 10$, i.e., the neighborhood of a WC instance contains at least 10 CC instances (Lelis and Sander 2009). We use this value for *MinPts* since the NN filter chooses 10-nearest neighbors for each WC instance (Turhan et al. 2009). In the future work, we will explore other values of *MinPts*.

In order to compare the performance of these data filtering methods, we choose three representative classifiers as the basic prediction model, Naive Bayes (NB) (Lewis 1998), Random Forest (RF) (Breiman 2001), and Logistic Regression (LR) (Hosmer and Lemeshow 2000). The reason we choose these classifiers is that these classifiers fall into three different families of learning methods. NB is a probabilistic classifier; RF is a decision tree classifier; and LR is a linear model for classification.

In every experiment, one dataset is selected as WC data and the rest are regarded as CC data to conduct the experiment. The CC data are considered as basic training data which will be adjusted in every experiment. WC data will be randomly divided into two parts: 10% labeled WC data and 90% unlabeled WC data. The SSDBSCAN filter uses the class information of 10% labeled WC data, while the NN filter and the Peters filter do not use the class information. In order to be fair, the 90% unlabeled WC data are taken as test data for all data filtering methods. All the methods will be repeated 30 times in every experiment to avoid sample bias. Then, the mean values of performance for all the methods are calculated.

RQ2: Does our proposed MSTrA+ approach perform better than other approaches in CCDP experiments?

This question validates the important criterion of defect prediction: the performance improvement in terms of PD, PF and G-measure (as defined in Sect. 4.2). To answer this question, we compare our approach against four state-of-the-art approaches used in CCDP (NB Lewis 1998, TNB Ma et al. 2012, NN + WC Turhan et al. 2013 and DTB Chen et al. 2015). More details are provided below:

(1) Naïve Bayes (NB) is a probabilistic classifier based on Bayes theorem. It has been widely applied in prior work as a basic prediction model to investigate the performance of CCDP.
(2) TNB firstly reweights CC data by the data gravitation method, then builds a transfer Naïve Bayes classifier on reweighted CC data.
(3) NN + WC (nearest neighbor filter with WC data) mixes p% WC data with CC data which were processed by NN filter as training data. In our experiment, we choose p as 10. Then, Naïve Bayes classifier is chosen as the basic prediction model on the training data.
(4) DTB firstly uses the NN filter, SMOTE (Hosmer and Lemeshow 2000) and data gravitation to process CC data. Then, limited amount of labeled WC data and reweighted CC data are mixed to build prediction model using the transfer boosting algorithm (Dai et al. 2007).

In every experiment, one dataset is selected as WC data and the rest are regarded as CC data to conduct the experiment. The CC data are considered as basic training data which will be adjusted in every experiment. WC data will be randomly divided into two parts: 10% labeled WC data as training data mixed with CC data in our MSTrA+ approach, the DTB approach and the NN + WC approach, and the remainder is taken as test data for all approaches in order to be fair. All the approaches will be repeated 30 times in every experiment to avoid sample bias. Then, the mean values of performance for all approaches are calculated.

RQ3: How does the proposed MSTrA+ approach perform compared to WCDP?

Previous studies (Turhan et al. 2009, 2013) have suggested that it is difficult for the performance of CCDP to reach that of WCDP. We want to clarify how well WCDP models perform with limited amount of labeled WC data. In addition, since our MSTrA+ approach also employs limited amount of labeled WC data, a comparison between MSTrA+ and WCDP models can validate the effectiveness of our MSTrA+

Table 4 Comparison with NN filter and Peters filter in terms of PD, PF and G-measure

Model	Metric	SSDBSCAN filter	NN filter	Peters filter
NB	PD	**0.777**	0.775	0.767
	PF	0.482	0.505	**0.438**
	G-measure	**0.603**	0.580	0.594
	W/D/L		8/2/5	11/0/4
RF	PD	**0.853**	0.822	0.823
	PF	**0.596**	0.652	0.643
	G-measure	**0.494**	0.439	0.440
	W/D/L		14/0/1	12/1/2
LR	PD	0.816	0.809	**0.826**
	PF	**0.547**	0.585	0.569
	G-measure	**0.551**	0.509	0.520
	W/D/L		11/1/3	8/2/5

Bold indicates the better values

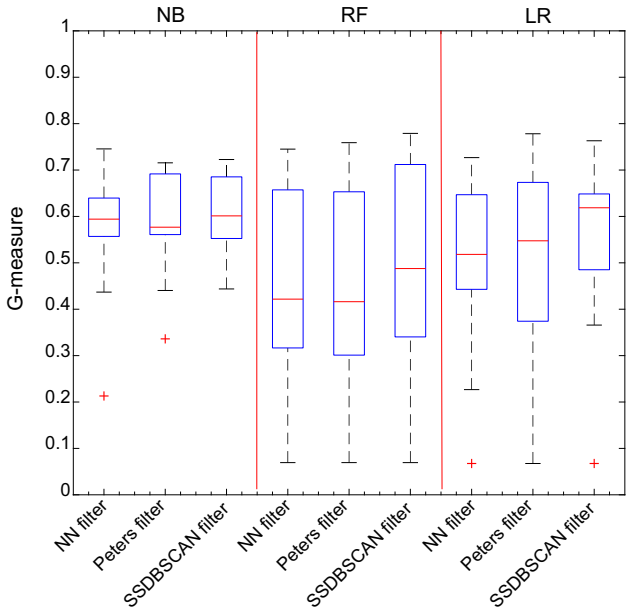

Fig. 3 Box plots for G-measure on 15 datasets with three classifiers

approach. If the MSTrA+ approach can outperform WCDP, a new company can exploit our proposed MSTrA+ approach to perform CCDP at the early stages of development activities.

To address this question, we employ three representative classifiers, Naïve Bayes (NB), Random Forests (RF) and Logistic Regression (LR) as the WCDP models. In every experiment, one dataset is selected as WC data and the rest are regarded as CC data to conduct the experiment. WC data will be randomly divided into training set and test set (10% and 90%, respectively). The WCDP models are trained by the training set, while our MSTrA+ model is trained by the training set mixed with CC data. In order to be fair, all models are tested on the WC test set. The process will be repeated 30 times in every experiment to avoid sample bias. Then, the mean values of performance for all models are calculated.

5 Experiment results

In this section, we present detailed experimental results to answer our three research questions mentioned above.

5.1 Results for RQ1

In this subsection, we compare our proposed SSDBSCAN filter with two state-of-the-art data filtering methods (NN filter Turhan et al. 2009 and Peters filter Peters et al. 2013). Table 4 records the average PD, PF and G-measure of all 15 datasets with three different data filtering methods on three classifiers, NB, RF and LR. The column W/D/L, short for Win/Draw/Loss, presents the number of datasets, on which SSDBSCAN filter performs better than, the same as, or worse than another method, in terms of G-measure.

Table 4 shows that on all three classifiers, SSDBSCAN filter performs better G-measure values (total average is 0.603, 0.494 and 0.551, respectively) than all the other methods. For NB classifier, SSDBSCAN filter achieves the best average PD and G-measure values, but fails in the best PF value. For RF classifier, SSDBSCAN filter can achieve the best values in terms of all the three metrics, comparing with all other methods. For LR classifier, SSDBSCAN filter can achieve the best PF and G-measure values, but fails in the best PD value.

The Win/Draw/Loss values shows that, on three classifiers, SSDBSCAN filter outperforms others on over half of projects in terms of G-measure. Figure 3 shows the box plots of G-measure values, with three data filtering methods for three classifiers on the 15 datasets. For NB classifier, the median value by SSDBSCAN filter is higher than that by NN filter and Peters filter, while the maximum value is a litter lower than that by Peters filter. For RF classifier, the median value by SSDBSCAN filter is much higher than that by all other methods. In addition, the maximum value by SSDBSCAN filter is much higher than that by other methods. For LR classifier, the median value by SSDBSCAN filter is much higher than that by NN filter and Peters filter, while the maximum value is a litter lower than that by Peters filter.

According to the experiment results in Table 4 and Fig. 3, we conclude that the proposed SSDBSCAN filter can yield better prediction results than the compared data filtering methods. Therefore, we employ the SSDBSCAN filter to select the most suitable CC data in our MSTrA+ approach.

Table 5 Comparison with NB, TNB, NN + WC and DTB in terms of PD and PF

No.	Dataset	MSTrA+		NB		TNB		NN + WC		DTB	
		PD	PF	PD	PF	PD	PF	PD	PF	PD	PF
1	ant	0.842	0.313	**1.000**	0.807	0.819	0.524	0.399	**0.275**	0.811	0.370
2	arc	0.502	**0.113**	0.681	0.580	0.745	0.413	**0.807**	0.648	0.605	0.272
3	camel	0.437	**0.073**	0.578	0.642	0.564	0.290	**0.784**	0.710	0.487	0.300
4	elearn	0.804	**0.226**	**1.000**	0.657	**1.000**	0.393	0.900	0.383	0.675	0.243
5	jedit	0.707	0.181	0.881	0.732	0.475	**0.168**	**0.932**	0.628	0.568	0.237
6	log4j	0.694	0.247	0.907	0.598	0.635	**0.134**	**0.936**	0.709	0.611	0.234
7	lucene	0.743	0.353	**0.769**	0.711	0.580	**0.221**	0.750	0.548	0.581	0.382
8	poi	0.551	**0.241**	0.893	0.750	0.416	0.228	**0.910**	0.684	0.632	0.415
9	prop-6	0.694	**0.236**	**0.868**	0.789	0.529	0.336	0.860	0.628	0.670	0.331
10	redactor	0.683	**0.323**	**1.000**	0.853	0.634	0.513	1.000	0.891	0.616	0.679
11	synapse	0.804	**0.394**	**0.958**	0.839	0.775	0.422	0.935	0.781	0.871	0.490
12	system	0.793	0.370	0.816	0.617	0.563	**0.260**	**0.817**	0.341	0.717	0.340
13	tomcat	0.543	**0.197**	**0.918**	0.637	0.914	0.592	0.690	0.359	0.712	0.396
14	xalan	0.732	**0.187**	**0.938**	0.693	0.604	0.356	0.961	0.685	0.654	0.400
15	xerces	**0.563**	0.384	0.437	0.684	0.319	**0.268**	0.437	0.631	0.370	0.274
	Average	0.673	**0.256**	**0.843**	0.706	0.638	0.341	0.808	0.593	0.639	0.358

Bold indicates the better values

5.2 Results for RQ2

In this subsection, we compare our proposed MSTrA+ approach with four state-of-the-art CCDP approaches [NB (Lewis 1998), TNB (Ma et al. 2012), NN + WC (Turhan et al. 2013) and DTB (Chen et al. 2015)]. Table 5 records the experimental results on 15 datasets regarding the PD and PF values, and Fig. 4 illustrates these results with scatter plots. Note that a CCDP approach has more points distributed at bottom right if it has higher PD value and lower PF value.

The figure shows that the NB approach often achieves the best PD (total average is 0.843) but the worst PF (total average is 0.706) therefore occupying the top right positions in Fig. 4. After filtering some irrelevant CC instances in CC data and mixing with 10% of WC data, the NN + WC approach achieved a 16% reduction over the NB approach in terms of PF on average. It is very obvious that the transfer learning approaches including TNB, DTB and MSTrA+ have lower PF value (total average is 0.341, 0.358 and 0.256, respectively) than the other two approaches, which leads to most points being distributed in the bottom right corner in Fig. 4. Compared with TNB and DTB, the MSTrA+ approach improves the average PD value at least by 0.035 (= 0.673–0.638). In addition, the MSTrA+ approach achieves the best PF value (total average is 0.256) among all tested CCDP approaches and yields more bottom right scatter points.

Table 6 shows the G-measure values of all tested approaches on 15 datasets. It is very clear that MSTrA+ improves the average G-measure value at least by 0.044 (= 0.669–0.625). The Win/Draw/Loss records also indicate that MSTrA+

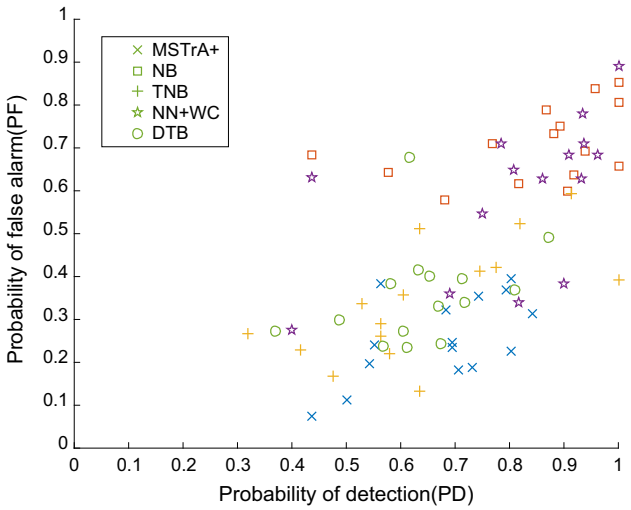

Fig. 4 Scatter plots of average PD and PF for five CCDP approaches on 15 datasets

wins over other approaches on most projects. Compared with NB, TNB, NN + WC and DTB, MSTrA+ wins on 15, 11, 13 and 12 datasets, respectively.

We also perform the Wilcoxon signed-rank test (Wilcoxon 1945) to analyze whether the performance values of MSTrA+ are statistically significant different with those of the compared approaches. The Wilcoxon signed-rank test is a non-parametic method of statistically testing the significance performance of multiple approaches. For the performance values of two approaches compared, the null hypothesis

Table 6 Comparison with NB, TNB, NN + WC and DTB in terms of G-measure

Dataset	MSTrA+	NB	TNB	NN + WC	DTB
ant	**0.752**	0.323	0.602	0.099	0.709
arc	0.567	0.519	0.655	0.488	**0.661**
camel	0.584	0.442	**0.629**	0.423	0.574
elearn	**0.801**	0.508	0.756	0.724	0.713
jedit	**0.725**	0.411	0.605	0.532	0.651
log4j	0.697	0.557	**0.733**	0.444	0.679
lucene	**0.694**	0.420	0.665	0.564	0.598
poi	**0.661**	0.391	0.540	0.469	0.607
prop	**0.684**	0.339	0.589	0.519	0.669
redaktor	**0.634**	0.256	0.550	0.196	0.422
synapse	**0.753**	0.275	0.662	0.355	0.643
system	0.632	0.466	0.640	**0.730**	0.687
tomcat	0.569	0.520	0.564	**0.660**	0.653
xalan	**0.675**	0.462	0.623	0.474	0.625
xerces	**0.613**	0.360	0.444	0.400	0.490
Average	**0.669**	0.416	0.617	0.472	0.625
W/D/L		15/0/0	11/0/4	13/0/2	12/0/3
p value		**0.001**	**0.025**	**0.003**	0.061
Hedges'g		**3.083**	0.703	**1.494**	0.583

Bold indicates the better values

states that there exists no significant difference between the two approaches. A p value less than 0.05 indicates that the null hypothesis is rejected. That is, the difference between the two approaches is identified as statistically significant. The significant test is implemented in IBM SPSS Statistics (Field 2001). In addition, we compute the effect size, Hedges'g (Kampenes et al. 2007), to quantify the amount of difference between two approaches. If the value of Hedges'g is greater than 1, this indicates that the performance of the prevision approach has a greater effect than that of the latter approach.

The Wilcoxon test shows that MSTrA+ performs statistically better than NB, TNB and NN + WC (p value = 0.001, 0.025 and 0.003, respectively) and is comparable to DTB (p value = 0.061). Moreover, the effect size of Hedges'g values for NB and NN + WC is greater than 1.0 (Hedges'g= 3.083 and 1.494). As for TNB and DTB, Hedges'g= 0.703 and 0.583, which also can be considered a medium-size effect (i.e., greater than 0.5, but less than 1.0).

According to the experiment results in Tables 5, 6 and Fig. 4, we can conclude that MSTrA+ has acceptable PD value and can obtain better PF value in most experiments, and it almost always achieves the higher G-measure. In other words, MSTrA+ has better overall performance than other approaches in CCDP experiments.

5.3 Results for Q3

In this subsection, we compare the performance results of MSTrA+ with three WCDP models, including Naïve Bayes (NB), Random Forests (RF) and Logistic Regression (LR). Table 7 shows the comparison results on 15 datasets in terms of G-measure.

Although these selected classifiers have been proved effective in WCDP, the experimental results seem to unsatisfactory. (The average G-measure values of these WCDP models are less than 0.444.) It is probably because 10% of WC data is not enough to perform WCDP. The MSTrA+ approach significantly outperforms NB, RF and LR, as it wins 14, 15 and 15 datasets, respectively. Furthermore, the results of Wilcoxon test show that MSTrA+ performs statistically better than these WCDP models (p values are all less than 0.05). Moreover, the effect size of Hedges'g values for NB and NN + WC is greater than 1.0 (Hedges'g = 1.736, 2.235 and 2.533, respectively).

According to the experiment results in Table 7, we conclude that our MSTrA+ approach performs significantly better than WCDP based on limited amount of labeled WC data. Therefore, a new company can exploit our proposed MSTrA+ approach to perform CCDP at the early stages of development activities when there is limited amount of labeled WC data.

6 Validity threats

In this section, we discuss several validity threats that may have an impact on the results of our studies.

External validity Threats to external validity occur when the results of our experiments cannot be generalized. As a preliminary result, we performed our experiments on 15 public datasets to explore the generality of our approach. Although these datasets have been widely used in many software defect prediction studies, we still cannot claim that our approach can be generalized to other datasets. Nevertheless, this work provides a detailed experimental description, including parameter settings. Therefore, other researchers can easily replicate our approach on new datasets.

Internal validity In our study, we repeat 30 times to avoid sample bias and calculate average results to verify the performance of all test approaches. We compared our SSDBSCAN filter with two state-of-the-art data filtering methods. To avoid the comparison bias, those methods were implemented in strict accordance with the authors instructions in related paper. In addition, we compared our MSTrA+ approach with four state-of-the-art CCDP approaches. Since the authors of the paper (Chen et al. 2015) give us the original implementation of DTB and their own implementation version of NB, NN + WC and TNB, we avoid the comparison bias.

Table 7 Comparison with WCDP models in terms of G-measure

Dataset	MSTrA+	NB	RF	LR
ant	**0.752**	0.332	0.328	0.393
arc	**0.567**	0.336	0.237	0.329
camel	**0.584**	0.239	0.052	0.191
elearn	**0.801**	0.177	0.187	0.167
jedit	**0.725**	0.646	0.635	0.579
log4j	**0.697**	0.551	0.496	0.463
lucene	**0.694**	0.603	0.597	0.506
poi	**0.661**	0.552	0.598	0.525
prop	**0.684**	0.507	0.176	0.396
redaktor	**0.634**	0.517	0.415	0.443
synapse	**0.753**	0.243	0.144	0.218
system	**0.632**	0.216	0.113	0.185
tomcat	0.569	**0.667**	0.268	0.443
xalan	**0.675**	0.586	0.279	0.455
xerces	**0.613**	0.491	0.560	0.549
Average	**0.669**	0.444	0.339	0.389
W/D/L		14/0/1	15/0/0	15/0/0
p value		**0.001**	**0.001**	**0.001**
Hedges'g		**1.736**	**2.235**	**2.533**

Bold indicates the better values

Construct validity In experiments, we mainly use PD, PF and G-measure to measure the effectiveness of the proposed approach. Nonetheless, other evaluation measures such as F-measure and AUC measure can also be considered.

7 Conclusion and future work

In this paper, we address the issues of how to weaken the impact of irrelevant CC data and how to avoid negative transfer when leveraging multiple CC data. We propose an effective solution for CCDP when there are limited amount of labeled WC data. Firstly, we propose a novel semi-supervised clustering-based data filtering method (i.e., SSDBSCAN filter) to filter out irrelevant CC data. Then, we for the first time introduce multi-source TrAdaBoost algorithm into CCDP, such that the risk of negative transfer is decreased by adopting knowledge from multiple CC data.

We conduct experiments on 15 publicly available datasets to evaluate the performance of the proposed approach. The experimental results indicate that the proposed approach can effectively weaken the impact of irrelevant data and avoid negative transfer to improve the performance of CCDP. The proposed MSTrA+ approach is an effective approach for CCDP.

In the future, we would like to validate the generalization ability of our approach on more software datasets, real-world datasets in particular. In addition, class imbalance is a natural characteristic of defect datasets and it will be interesting to apply some well-known data sampling approaches (Bennin et al. 2017a, b) to resolve the negative impact of class imbalance after filtering the data.

Acknowledgements This work is partly supported by the grants of National Natural Science Foundation of China (61070013, 61300042, U1135005, 71401128), the Fundamental Research Funds for the Central Universities (Nos. 2042014kf0272, 2014211020201) and Natural Science Foundation of HuBei (2011CDB072).

Compliance with ethical standards

Conflicts of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent This article does not contain any studies with human participants.

References

Arar Ömer Faruk, Ayan Kürşat (2015) Software defect prediction using cost-sensitive neural network. Appl Soft Comput 33:263–277

Bennin KE, Keung J, Phannachitta P, et al (2017) MAHAKIL: diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. IEEE Trans Softw Eng. https://doi.org/10.1109/TSE.2017.2731766

Bennin K, Keung J, Monden A, et al (2017) The significant effects of data sampling approaches on software defect prioritization and classification. In:11th International symposium on empirical software engineering and measurement, ESEM 2017

Boetticher G, Menzies T, Ostrand T (2007) PROMISE Repository of empirical software engineering data, West Virginia University, Department of Computer Science. http://promisedata.org/repository

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Briand LC, Melo WL, Wust J (2002) Assessing the applicability of fault-proneness models across object-oriented software projects. IEEE Trans Softw Eng 28(7):706–720

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357

Chen L, Fang B, Shang Z et al (2015) Negative samples reduction in cross-company software defects prediction. Inf Softw Technol 62:67–77

Dai W et al (2007) Boosting for transfer learning. In: 24th International conference on Machine learning, pp 193–200

Dhanajayan RCG, Pillai SA (2016) SLMBC: spiral life cycle model-based Bayesian classification technique for efficient software fault prediction and classification, Soft Computing, 1-13

Elish KO, Elish MO (2008) Predicting defect-prone software modules using support vector machines. Softw J Syst Softw 81(5):649–660

Erturk Ezgi, Sezer Ebru Akcapinar (2016) Iterative software fault prediction with a hybrid approach. Appl Soft Comput 49:1020–1033

Field AP (2001) Discovering statistics using SPSS for windows: advanced techniques for beginners, pp 551–552

Fukunaga K, Narendra PM (1975) A branch and bound algorithm for computing k-nearest neighbors. IEEE Trans Comput 100(7):750–753

Gray D, Bowes D, Davey N, et al (2009) Using the support vector machine as a classification method for software defect prediction with static code metrics. In: International conference on engineering applications of neural networks. Springer, Berlin, pp 223–234

Hall T, Beecham S, Bowes D, Gray D, Counsell S (2012) A systematic literature review on fault prediction performance in software engineering. IEEE Trans Softw Eng 38(6):1276–1304

Hosmer DW, Lemeshow S (2000) Introduction to the logistic regression model. Appl Logist Regres 1–30

Jain K (2010) Data clustering: 50 years beyond K-means. Pattern Recognit Lett 31(8):651–666

Jing X et al (2015) Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning. In Proceedings of the 10th joint meeting on foundations of software engineering, pp 496–507

Jing XY, Ying S, Zhang ZW, Wu SS, Liu J (2014) Dictionary learning based software defect prediction. In: Proceedings of the 36th International Conference on Software Engineering, pp 414–423

Kampenes V By et al (2007) A systematic review of effect size in software engineering experiments. Inf Softw Technol 49(11):1073–1086

Kawata K, Amasaki S, Yokogawa T (2016) Improving relevancy filter methods for cross-project defect prediction, applied computing & information technology, pp 1–12

Laradji IH, Alshayeb M, Ghouti L (2015) Software defect prediction using ensemble learning on selected features. Inf. Softw. Technol. 58:388–402

Lelis L, Sander J (2009) Semi-supervised density-based clustering. In: 9th IEEE international conference on data mining, pp 842–847

Lewis DD (1998) Naive (Bayes) at forty the independence assumption in information retrieval. In: European conference on machine learning, pp 4–15

Ma Y, Luo G, Zeng X, Chen A (2012) Transfer learning for cross-company software defect prediction. Inf Softw Technol 54(3):248–256

Malhotra Ruchika (2015) A systematic review of machine learning techniques for software fault prediction. Appl Soft Comput 27:504–518

Mesquita PP Diego et al (2016) Classification with reject option for software defect prediction. Appl Soft Comput 49:1085–1093

Nam J, Pan SJ, Kim S (2013) Transfer defect learning. In: Proceedings of the 2013 international conference on software engineering. IEEE Press, pp 382–391

Peng L, Yang B, Chen Y, Abraham A (2009) Data gravitation based classification. Inf Sci 179(6):809–819

Peters F, Menzies T, Marcus A (2013) Better cross company defect prediction. In: Proceedings of the 10th international workshop on mining software repositories, pp 409–418

Ryu D, Jang JI, Baik J (2015) A hybrid instance selection using nearest-neighbor for cross-project defect prediction. J Comput Sci Technol 30(5):969–980

Seliya N, Khoshgoftaar TM (2011) The use of decision trees for cost- sensitive classification: an empirical study in software quality prediction. Wiley Interdiscip Rev Data Min Knowl Discov 1(5):448–459

Shepperd M, Bowes D, Hall T (2014) Researcher bias The use of machine learning in software defect prediction. IEEE Trans Softw Eng 40(6):603–616

Shukla S, Radhakrishnan T, Muthukumaran K, et al (2016) Multi-objective cross-version defect prediction, Soft Computing 1-22

Siers MJ, Islam MZ (2015) Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem. Inf. Syst. 51:62–71

Song Q, Jia Z, Shepperd M et al (2011) A general software defect-proneness prediction framework. IEEE Trans Softw Eng 37(3):356–370

Sun Z, Song Q, Zhu X (2012) Using coding-based ensemble learning to improve software defect prediction. IEEE Trans Syst Man Cybern Part C (Appl Rev) 42(6):1806–1817

Turhan B, Menzies T, Bener AB, Di Stefano J (2009) On the relative value of cross-company and within-company data for defect prediction. Empir Softw Eng 14(5):540–578

Turhan B, Tosun Mısırlı A, Bener A (2013) Empirical evaluation of the effects of mixed project data on learning defect predictors. Inf Softw Technol 55(6):1101–1118

Vashisht V, Lal M, Sureshchandar GS et al (2015) A framework for software defect prediction using neural networks. J Softw Eng Appl 8(8):384

Wang J, Shen B, Chen Y (2012) Compressed C4. 5 models for software defect prediction. In: 12th international conference on quality software, pp 13–16

Wilcoxon Frank (1945) Individual comparisons by ranking methods. Biom Bull 1(6):80–83

Yan Z, Chen X, Guo P (2010) Software defect prediction using fuzzy support vector regression. In: International Symposium on Neural Networks. Springer, Berlin Heidelberg, pp 17–24

Yao Y, Doretto G (2010) Boosting for transfer learning with multiple sources. In: IEEE conference on computer vision and pattern recognition, pp 1855–1862

Zimmermann T, Nagappan N, Gall H, Giger E, Murphy B (2009) 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 and the ACM SIGSOFT symposium on The foundations of software engineering, pp 91–100