Journal of Theoretical and Applied Information Technology
30th June 2020. Vol.98. No 12

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A REVIEW OF TRAINING DATA SELECTION IN SOFTWARE DEFECT PREDICTION

[1,2] BENYAMIN LANGGU SINAGA, [2]SABRINA AHMAD, [2]ZURAIDA ABAL ABAS

[1] Department of Informatics Engineering, Universitas Atma Jaya Yogyakarta, Indonesia

[2] Center for Advanced Computing Technology, Faculty of Information and Communication Technology Universiti Teknikal Malaysia Melaka, Malaysia

E-mail: benyamin.sinaga@uajy.ac.id, sabrinaahmad@utem.edu.my, zuraidaa@utem.edu.my

ABSTRACT

The publicly available dataset poses a challenge in selecting the suitable data to train a defect prediction model to predict defect on other projects. Using a cross-project training dataset without a careful selection will degrade the defect prediction performance. Consequently, training data selection is an essential step to develop a defect prediction model. This paper aims to synthesize the state-of-the-art for training data selection methods published from 2009 to 2019. The existing approaches addressing the training data selection issue fall into three groups, which are nearest neighbour, cluster-based, and evolutionary method. According to the results in the literature, the cluster-based method tends to outperform the nearest neighbour method. On the other hand, the research on evolutionary techniques gives promising results but is still scarce. Therefore, the review concludes that there is still some open area for further investigation in training data selection. We also present research direction within this area.

Keywords: *Software Defect Prediction, Training Data Selection, Nearest-Neighbor, Cluster-based, Evolutionary-based*

1. INTRODUCTION

Software defect prediction (SDP) becomes a critical activity to increase software quality and to reduce software testing effort. SDP develops a defect prediction model (SDP model) that enables the prediction of a defect-prone module [1]. The SDP model can categorize a software component as a defect or non-defect. The SDP model helps software developers to distribute the limited resources to test and review the modules that most likely contains defect [2][3][4]. Therefore, the utilization of the SDP model would be of benefits for the software developers since the model helps the software developers to focus on inspecting or testing the high defect-prone modules judiciously.

Many studies have developed models for the prediction of a software defect. They trained the SDP model using the past defect data from the same project to predict the defect in the next version [5]. This method is called With-in Project Defect Prediction (WPDP). Nevertheless, the past defect dataset is not always available, since the company either start a newly initiated project [6] or does not retain the historical defect from earlier projects [7]. Such a situation causes creating software defect prediction become unfeasible since the process of training cannot be conducted when defect dataset is unavailable. This problem is solved by leveraging data from other organizations in which local data from one organization is transferred to other organizations to make the training dataset available. Using those historical datasets, an SDP model is built and utilized to make prediction on the target projects [6], [8]. Such a strategy is known as cross-project defect prediction (CPDP).

CPDP approach has been an attractive approach for the solution of unavailable historical data. However, it is also challenging since most SDP models are developed using machine learning algorithms, which work under the common assumption that the underlying distribution of training datasets is similar to that of the testing dataset. Using historical defect data from other projects introduces a critical issue since source datasets and target datasets have different data distribution [5], [8], [9]. CPDP model built using a

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

machine learning algorithm will suffer from unsatisfied predictive performance since the conventional machine learning algorithm performs well if it trains using training data posses a similar distribution with testing data. The solution to this problem is to lessen the divergence in distribution between the training and testing data.

The conventional approaches to addressing such an issue are data transformation [9]–[12], normalization [9]. Both data transformation and normalization approaches use all training instances to train the prediction model, which may potentially contain irrelevant and noisy data. Zhang et al. [13] found that choosing a suitable transformation for a specific pair of training and testing instances is open to question. Prior studies [9], [14] even show that the effect of transformations on the modeling performance varies on the same dataset. Therefore, training data selection has a potential benefit to overcome the drawback of the previous approaches.

Training data selection that attempts to select the most relevant training instance from the software repository has been a significant issue for CPDP [15]. On the one hand, some research revealed that using a cross-project training dataset without a careful selection degraded the defect prediction performance [5], [6], [8]. On the other hand, several studies also reported that the SDP model developed using suitable cross-project data has a satisfied predictive performance [6] [5][16]. The selection of relevant training data increases the prediction performance of the model, even though this performance still cannot compete with the performance of WPDP. Therefore, how to choose the relevant data gathered from other domains for training a defect prediction model becomes a challenge [16].

There are many studies conducted on software defect prediction. Several excellent review papers have been published in this area [17]–[23]. Catal and Diri [21] conducted a review focusing on the conceptual classification of a software metric, datasets, and method. Later on, Radjenovic et al. [22] presented a survey on software metrics and the effect of context on the metric selection and metric performance. Malhotra [17] surveyed machine learning tehcniques, software metrics, and datasets used used to build the SDP mode. Hall et al. [18] presented a review paper discussing the independent variable, the effect of context in prediction performance, and methods to develop a software defect prediction model. Hosseini et al. [23] focused the survey on CPDP that summarizes and synthesis the independent variable, modeling techniques, and approaches to building defect prediction. However, there is no survey focusing on the training data selection works in the CPDP area. Since training data has been applied in SDP and indicated promising results to enhance the effectiveness of defect prediction, knowledge of current training data selection methods is required. The purpose of this paper is to present a review of training data selection techniques for researchers and practitioners. This paper offers a brief description of the background and state-of-the-art research progress. It also provides strength and limitation of the proposed methods as well as the potential challenges on this training data selection area. It also provides an opportunity for researchers to develop this specific research area further.

This paper is structured as follows: Section 2 defines training data selection, section 3 overviews the studies on training data selection during the period of 2009 – 2019, section 4 presents a discussion on existing training data selection approaches, and part 5 draws on the conclusion and highlights research directions.

2. ISSUES IN TRAINING DATA SELECTION

2.1 Distribution Difference Problem in Cross-Project Defect Prediction (CPDP)

CPDP develops an SDP model utilizing the historical datasets from the source project to make a defect prediction in the target projects [8]. CPDP becomes a common approach as it addresses the shortcoming of the training data that is required to construct a software defect predictor [6], [9], [11], [15], [24]–[29]. However, it also poses a challenging issue. Directly use cross-project datasets to learn a prediction model produces a model having unsatisfied predictive performance. Zimmerman et al. [8] investigated 622 cross-project predictions and found that the prediction performance was unacceptable; only 3.38% of the predictions worked successfully. Further on, He et al. [5] found that the successful ratio of five cross-project predictors was inadequate, which is in the range of 0.32% to 4.7%. Later, Turhan et al. [6] also concluded that the software defect predictor developed using all available cross-company data would contribute to a high false alarm rate. This issue might result from the divergence in the distribution between the training and testing data.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[5], [8], [9]. Thus, the CPDP problem is how to deal with data distribution divergence among the sources and the targets project.

CPDP is a kind of transfer learning problem [23] that corresponds to transductive transfer learning [30], [31], in which source task is the same as the target (i.e., predict the defect-prone module) and the source domain differs from target domain (i.e., source project and target project) [11], [32], [33]. There are three instance-based transfer learning approaches to deal with the distribution divergence issue in the literature namely: data transformation [9], [10], [12], [34], reweighting the training data [11], [25], [28] and selecting a part of the training data or training data selection. Therefore, related to transfer learning, training data selection is a type of instance-based transfer learning. This review paper focuses on the last strategy because of its higher applicability.

2.2 Training Data Selection

Training data selection is a process that tries to select the most relevant training instance with regards to the target instance. More specifically, let $\mathbf{S_S}$ be a source project dataset containing m instances, expressed as $\mathbf{S_S}$ = {ss_1, ss_2, ss_3, …, ss_m}, $\mathbf{S_T}$ be the selected training dataset having n instances, expressed as $\mathbf{S_T}$ = {st_1, st_2, …, st_n}, and $\mathbf{T_T}$ be the target dataset containing o instances expressed $\mathbf{T_T}$ = {tt_1, tt2_, …, tt_o}. Training data selection aims at forming the training dataset ($\mathbf{S_T}$) from the source dataset ($\mathbf{S_S}$) that contains the most relevant source instances to target instances in the dataset ($\mathbf{T_T}$). The selected training instances are utilized to develop an SDP model, which is applied to predict the defect of unlabeled instances in the target dataset. Domain in which learning a defect prediction is conducted is called the target project, while the domain from which the relevant instance comes is called the source project. It is assumed that the target project has unlabeled instances, and the source projects contain many labeled instances. Training data selection attempts to select training instances from labeled source instances based on the (1). The similarity of the labeled source instances to the unlabeled target instances or (2). The similarity of data distribution of the source datasets to that of the target dataset.

An essential factor pertinent to the training data selection is the relevant source instances. Relevant source instances mean that those instances are suitable for training a defect prediction model. Correct identification of appropriate training data is important, as the use of irrelevant training instances can harmfully impact the accuracy of an SDP model. A criterion that is used to determine the relevant training data is the similarity of the source instances to the instance in the target project [6][16]. Besides the similarity factor, there are other criteria to determine whether source instances are suitable for a building defect prediction. Training data instances may be unsuitable for building defect prediction if: (1). They contain noise [35], [36]. (2). They have labels conflicting with that of testing instances [25], [28], or (3). They do not have the same defect patterns as that of the target dataset [5]. The selection for training data will remove some source instances, and the performance of the model trained using the selected training data does not decrease [37].

2.3 General Training Data Selection Process

Through a literature review of training data selection research, most training data selection approaches follow the general process in Figure 1. The first step is preparation of the source data instances collected from other projects ($\mathbf{S_S}$). The second step is to selection of the relevant training data instances from source data instances based on the similarity to the target data instances ($\mathbf{T_T}$). Later, a defect prediction model is trained using the selected training instances ($\mathbf{S_T}$) to predict the defect in the target data instances.

Through the literature review, strategy to select relevant training data mainly implement the nearest neighbor, cluster-based, and evolutionary-based methods.

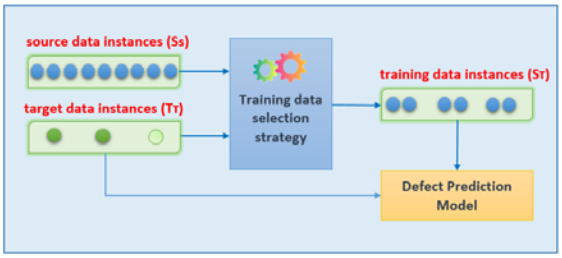

Figure 1: Training Data Selection Process (Taken from [16])

The description of each strategy is presented in Table 1, while Algorithm 1 – 3 depicts the general procedure for each strategy.

Algorithm 1 Nearest neighbor-based approach
input: *source instances ($\mathbf{S_S}$), target instances ($\mathbf{T_T}$)*
output: *selected training instances ($\mathbf{S_T}$)*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

for each training instance in target dataset do
calculate its distance to source instances
collect n nearest source instances based on distance
end
remove duplicate source instances
collect the remaining source instances as the training instances

return selected training instances

Algorithm 2 Cluster based approach
input: *source instances (S_S), target instances (T_T)*
output: *selected training instances (S_T)*

combine source instances and target instances into one group.
partition the group into clusters
retain the cluster having at least one target instances
collect the training instances in the retained subsets.

return selected training instances

Algorithm 3 Search-based approach
input: *source instances (S_S), target instances (T_T)*
output: *selected training instances (S_T)*

setup initial population collected randomly from source instances
while termination condition is not satisfied do
evaluate individual according to fitness function
add the selected individual to the pool of generation
create new generation using evolutionary operator
replace population

select the best generation as the selected training instance
return selected training instances

2.4 Training Data Source for Building Defect Prediction Model

The CPDP approach tries to leverage source data from one project to train a software defect predictor and then perform a defect prediction on the other software project. According to Herbold et al. [38], when building a software defect predictor, there are two approaches regarding the source of training data: defect prediction using only cross-project data (strict CPDP) and defect prediction using mixed data (mixed CPDP). For the first approach, the source of training data comes from either single or multiple cross-project data. For example, [6] proposed a relevancy filter to select the relevant data by employing cross-project data to train the software defect predictor model (Table 2 for the related studies).

Meanwhile, for the second approach, the source of training data is mixed from cross-project and unlabeled within-project. Several studies investigated the performance of the prediction model developed using mixed data [25], [28], [39]. Turhan et al. [39] utilized mixed data to develop a defect prediction model. They inferred that when limited historical data are available, construction of software defect predictor having performance similar to full within project predictor is feasible. Later on, Chen et al. [25] confirmed this finding. Using mixed data, they presented a Double Transfer Boosting (DTB) algorithm to deal with the distribution mismatch between cross-project and within-project.

Table 1: Overview of Training Data Selection Method

1.	**Nearest Neighbor Filter**	
	Description	Nearest Neighbor Filter training data selection is an approach to select relevant training instance by measure the similarity of an instance in the source datasets with instance in the target datasets using the notion of distance, such as Euclidean distance. This approach selects n source instances nearest to target instances.
	Related Studies	[6][15] [16], [40]–[43] [44][45]
	Strength	1. Most methods use KNN algorithm that has simple steps. 2. For project level granularity, it has low computational cost

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Limitation	1. It has polynomial run time and scalability problem with large datasets. 2. For project level granularity, the selected training data instances may include irrelevant source instance, which may result in false alarm rate.

2.	**Cluster-based filter**
Description	Cluster-based filter uses clustering method to find homogeneous clusters, where each cluster contains instances having similar characteristics [46], [47]. Based on this idea, a source instance is similar or relevant to the target instance if they are in the same cluster. Thus, the selected training instances are the source instances that are in the same cluster as the target instances.
Related Studies	[27][28], [48] [16][44]
Strength	1. Generally, cluster-based method performs better than the distance-based approaches. 2. Some cluster-based methods can remove noise [27], [28]
Limitation	1. The results is sensitive to the choice of the parameters of the clustering method, such the number of cluster, the radius [27], [28], initial centroid selection [46][44]. 2. Clustering-based approaches suffer from computational complexity 3. Performance depends on the quality of clustering.

3.	**Evolutionary-based filter**
Description	A evolutionary-based method uses the meta-heuristics procedure [49] to find the optimal training data instances. Hosseini and Turhan implemented the search-based procedure [50], [51] to select training data in software defect prediction research. They developed a search-based training data selection method using a genetic algorithm [52], [53]. Selected training data instancess are best chromosomes occured from the evolutionary process.
Related Studies	[52], [53]
Strength	1. Can deal with noisy, incomplete, imbalance, and inaccurate data. 2. This approach can use any classifier for the selection process.
Limitation	1. It has high computational time because of its iterative optimization process.

Subsequently, Yu et al. [28] also explored the benefits of the mixed data approach. They combined unlabeled and a limited amount of labeled within-company data as well as cross-company data to train defect predictor. The experiment showed that a mixed model for cross-company defect prediction could perform well, which is comparable to the performance of WPDP.

Table 2: Studies on Training Data Categorized According to the Source and Granularity of Training Data

		Source of training data	
		Cross project data	Mixed data
Granularity of training data selection	Source Project Level	[5], [16], [40]–[43]	[54]
	Instance Level	[6], [15], [27], [48]	[25], [28], [39]

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2.5 Training Data Selection Granularity

Besides the training data selection strategy, which is categorized into the distance-based, cluster-based, and search-based approach, another factor of interest is the training data selection granularity. Based on the previous studies, there are two levels of training data selection granularity, namely instance-level [6] and project (dataset) level [5], [16]. Table 3 presents training data selection studies categorized according to the selection strategy and granularity of training data.

At a project level, a dataset represents a release of a project (coarse-grained data). Figure 2 illustrates a dataset that contains 100 instances. A dataset can be represented as a vector of distributional characteristics (statistical characteristics), such as mean, median, min, max. For example, if a dataset in Figure 2 is represented using two distributional characteristics (i.e., min and median) then the datasets is formulated as V_dc = {min(F_1), median(F_1), min(F_2), median(F_2), …, min(F_m), median(F_20)}, where V_dc is vector of distributional characteristics.

Table 3: Studies on Training Data Categorized According to the Selection Strategy and Granularity of Training Data

		Selection Strategy		
		Nearest-neigbor	Cluster-based	Evolutionary-based
Granularity of training data selection	Instance Level	[6][55][2][45]	[27], [28], [48]	[52], [53]
	Project Level	[16][2]	[16][44]	NA
	Mixed	[2]		NA

Meanwhile, at an instance level, an instance refers to any record (a row) in a dataset, which may represent a file or a package (fine-grain data). An instance is expressed as a vector of feature values, i.e., I_i = {F_i1, F_i2, F_im}, where m denotes the number of features.

	F_1	F_2	F_3	F_4	…	F_m	
Instances	WMC	DIT	NOC	CBO		LOC	Defect
I_1	14	1	0	11		290	0
I_2	17	2	0	2		330	3
I_3	5	1	5	17		45	2
I_4	13	2	0	22		223	0
I_5	14	2	2	3		123	0
….							
I_100	12	1	0	4		400	1

Figure 2: An Example of a Dataset

3. PRACTICAL METHODS FOR TRAINING DATA SELECTION

Studies in CPDP has investigated several training data selection methods. These include works on relevancy filter that using distance measure between datasets [6], [15], [16], [41], [42], clustering approach [27], [28], [44], [48] and evolutionary approach [52], [53]. The following section will discuss the detail of each specific training data selection.

3.1. Nearest Neighbor Filter

The basic idea of this filter is the similarity of an instance in source datasets to its near instances in target datasets. The notion of instance has a different meaning with regards to the granularity level of the datasets. At an instance level, an instance is any record in a dataset (expressed as a vector of metric values). In contrast, at a project level, it represents a vector of data characteristics. The similarity of the datasets is measured using the notion of distance [56]. Selecting relevant training datasets using a nearest neighbor approach can be conducted based on the granularity level of the source dataset. For example, the authors in [6] [15] [57] [26] proposed an instance-level approach while studies in [5], [16], [40]–[43] introduced a project level filter to select the relevant source data.

Turhan et al. [6] present the idea of data filtering for the improvement of learning an SDP model. They introduced the Burak filter to choose the relevant training data using the K-Nearest Neighbor algorithm. This filter employs the testing instance as guidance for choosing the relevant training data instance (target-driven filter). It is a point-wise filter in which for each target instance, this filter chooses its ten nearest neighbor source data instances as the candidate for the training data. This filter, then, combines these selected instances (without duplication) to form a new training dataset, which is employed to develop an SDP model. Turhan et al. found that the developed model has performance close to that of an SDP model built using within-project data. They also claimed that identified information from cross-project data leads to improvements in detection rates. Burak Filter has relatively simple steps; however, it has a disadvantage in that each time the testing dataset instance changes, this filter must be repeated. This filter was also adopted in the work of [25], [45], [58] .

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

To improve Burak Filter, Peter et al. [15] proposed a source-driven filter, which enables the source instances to find their nearest testing instances. The core idea of this filter is a conjecture that a large defect dataset has more defect information. The training dataset is usually more massive than the testing dataset; thus, it might be more suitable to use training instance when identifying the relevant training data. This filter combines the K-Means [46] method with K-Nearest Neighbor. It first combines the source dataset and the target datasets and then partitions the combined dataset using the K-Means algorithm. The cluster has at least one target data instance is retained. Subsequently, for each source data instance in the selected group, the filter finds the closest target instance. Finally, for each target instance, the nearest source instances are chosen as the training data, using Euclidean distance. Based on the results, Peter filter outperforms the within-project and the Burak filter. This filter has a simple step; however, it has an exponential run time, and it does not scale with large datasets [59].

Similar to Burak Filter, testing-driven filter at the instance level, He et al. [57] introduced an improved method to select training data by considering not only the similarity between the training and the target instance but also the number of the defect of each training instance. They proposed a training data selection method, called TDSelector. When choosing the relevant training data instance, the TDSelector employs a scoring scheme that uses two rank scores of each training instance as the input. For each training instance, the first score calculated based on its similarity to the testing instances, and the second one according to the number of defects. The scoring scheme, subsequently, calculates the final score by considering the similarity ranking and defect ranking. Based on the final score, for each testing instance, the method will collect the top-k training instances. TDSelector then combines all set of the top-k training instances into final training datasets after removing the redundant instances. To validate the success of the proposed approach, they conducted an experiment using 15 open-source datasets from 14 different projects. This study concluded that information about the number of defects is valuable to build the defect prediction as this defect information could be used to predict defect proneness, which is proven by the improvement of the model performance in terms of G and AUC. TDSelector outperforms Peter Filter and TCA+. Although the result seems to promise, benchmarking of their proposed approach was conducted only to method Peter Filter and TCA+ [9]. Therefore, the generalizability of the TDSelector for other classification algorithms remains debatable.

Ryu et al. [26] proposed selective learning to address the problem of the distribution gap between training and test data. They introduced the Hybrid Instance Selection using Nearest-Neighbor (HISNN) to remove irrelevant training data instances. HISNN employed two-phase instance filtering, namely: test data instance selection and training data instance filtering, respectively. Training data selection is conducted in several steps, (1). identification and removal of an existing outlier in the source data, (2). identification of the source data that similar to test data using K Nearest Neighbor, and (3). combining the result of both previous steps as the selected training data set. HISSN performed classification of local knowledge and global knowledge, using K Nearest Neighbor naïve Bayes, respectively. The study validated the algorithm by using 13 selected public datasets and concluded that the HISSN algorithm provides a promising performance CPDP setting.

To address the unsatisfied results and high computational time of the prior study, Herbold [16] proposed a strategy using the KNN algorithm to select a relevant training dataset. Following the result of [5], Herbold employed distributional characteristics (i.e., standard deviation and mean) to detect similarity among datasets. Based on the between Euclidean distance between characteristic vectors, Herbold employs the Nearest Neighbor filter to choose source data projects closest to the target project. Herbold experimented with a cross-project prediction using 71 version datasets available from 38 different open source projects. The training data was predominantly imbalanced. This condition usually contributed to biased classifiers that were in favor of a non-defect-prone class. To deal with this issue, Herbold adjusted the weight of each training instance to make the overall weight of training and testing instances to be the same. He concluded that predictive performance improves significantly**,** based on success rate and recall. Also, project-level selection (based on distributional characteristics) rather than point-wise instance-level selection results in the prediction model of low computational time. However, the result is still unfavorable for actual use.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

He et al. [41] conducted a study to provide a guiding principle to choose the relevant training data available from other projects. Different from the prior studies, they suggested employing multilevel granularity in a single defect prediction model. They introduced a method to simplify training data by utilizing two levels of granularity, i.e., project level and instance level. At the project level, referred to as rTDS, training data simplification selects training data releases most similar to the testing data release based on the distributional characteristic of the dataset (median, mean, min, max, and standard deviation). While, at the instance level, referred to as iTDS, it tries to choose the most suitable training instances according to their distance to the testing instance. Both approaches use using the K Nearest Neighbor algorithm with Euclidean distance [6], [15], [16].

To examine the effect of multilevel granularity on prediction performance, He et al. [41] combine both techniques (i.e., rTDS and iTDS) into a two-step method, namely riTDS. There are two instance selection approaches in the second step of the riTDS. The first strategy uses the testing instance as guidance in selecting the training instances nearest to the testing instance (identical to Turhan filter). At the same time, the second approach employs training instance as the guidance on labeling the testing instance and collecting the training instance closest to the testing instance (similar to Peter Filter). They validated the proposed approach using 34 datasets and J48, LR, NB, RF, and SVM as the predictors. The result shows that using a multilevel granularity approach can provide better performance provided that it uses the appropriate filter.

In their study on predicting defect inducing changes, Fukushima et al. [60], later extended by [61], proposed a procedure for selection of the relevant training data using Spearman correlation. Spearman correlation is used to assess the similarity between training datasets and testing datasets. First, the procedure calculates the Spearman correlation between each metric (independent variable) with the class label of the training datasets (dependent variable). Second, it continues to select three metrics from source datasets (sr_1, sr_2, sr_3) having the highest Spearman correlation value. This procedure also chooses the same metrics from testing datasets (ts_1, ts_2, ts_3). Then, this method computes the pair-wise correlation between the selected metrics to define two correlation vector, such as the first vector has corr(sr_1, sr_2), corr(sr_1, sr_3), corr(sr_2, sr_3) as the elements, while the second vector contains corr(ts_1, ts_2), corr(ts_1, ts_3), corr(ts_2, ts_3). Last, the process calculates the distance between the two vectors using Euclidean distance. The training dataset having the lowest distance is chosen as the training dataset. This proposed method has a limitation in that it can only select one source project data. Xia et al. [33] identified that using multiple sources when constructing software defect prediction can prevent overfitting and improve performance.

Liu et al. [43] adapt the training data selection to address the performance problem of TCA+ [9]. They found that if TCA+ is trained using only one randomly selected source project data, its performance is unstable, known as dataset shift problem [62]. They also revealed that using all source projects may lead to unsatisfied performance (in terms of F measure). Differ with Herbold [16] that uses an unsupervised approach, Liu et al. introduce a supervised approach by proposing a source project estimator (SPE) to choose the suitable training data. SPE tries to select two source project datasets having a similar distribution to the target project by learning two regression models. As the independent variables, the regression models employ the metric from source project data. The final regression model is the model that scores the best according to F measure and PofB20. The proposed method has an advantage in which the regression models do not necessarily need to update continuously, which leads to an acceptable developing time and opportunity for practical use. This approach, however, uses only two data sources when predicting the defect. Also, if the target project is unlabeled, this method uses a hypothetical source project as the target project when learning the regression model, which may not represent the typical properties of a defect in the target project.

Addressing the same problem to Liu et al. [43], Wen et al. [42] investigated the training data selection method using data transformation. They proposed four strategies to select source project data according to the value of the mean, standard deviation, and median of the source and the target datasets. Logarithmic transformation transforms mean, standard deviation, and median of each dataset. The median value is also transformed using the Z score transformation. The closeness of each source project data to its target data is measured using Euclidean distance, taking as input the refined value of the mean, standard deviation, and median

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

of the source and target data. The source project data having minimum distance with the target data is selected as training data. The result shows that this method outperforms the training data selection method by Herbold in terms of F and accuracy. However, this proposed method has a limitation in that it can only select one source project data. Xia et al. [33] identified that using multiple sources when constructing software defect prediction can prevent overfitting and improve performance. Also, employing various source projects rather than a single project have the advantage of being able to provide more information when building an SDP model [59].

He et al. [40] studied the use of an SDP model trained using open source data and applied to predict the defect on the proprietary project. They propose a procedure adapting Hido's framework to measure the closeness of training and testing dataset by calculating the marginal distribution of training dataset and testing dataset**.** To measure the similarity, they sample k instances from the training and testing data randomly and combine both sample datasets into a single dataset. Then, they learn a logistic regression classifier on the combined dataset and calculate the model accuracy. The accuracy is used as the measurement of the distance between training and testing datasets. Based on an argument that characteristics of a project generating a dataset reflect the characteristics of its instances, He et al. [40] select the relevant training data at the data level [16] rather than at the instance level [6], [15]. Their procedure employs feature selection to address the distribution gap problem. This research experimented CPDP with using 34 version datasets available from 14 different open source projects and 34 version datasets available from 7 various proprietary projects. The results show that He et al.'s filter outperforms than Burak filter in terms of performance (better G-measure and PF than that of Burak filter) and runtime complexity. He et al. concluded that for CPDP to achieve the best performance, it does not need to use all features. This conclusion supports the finding of the previous studies [63], [64].

3.2. Cluster-based Filter

The cluster-based technique is a common approach in SDP research to overcoming the mismatch between source and target data distribution. In transfer learning research, Qiu et al. [65], for example, partition the source data into clusters and give different weights for instances in each subset to lessen the adverse impact of irrelevant training data. Chen et al. [66] cluster the source and target data to create Multi-view learning, and Zhang et al. [29], [67] divide the source data into partitions to investigate the context-aware rank transformation. While, in unsupervised learning, Zhang et al. [68], [69] utilized clustering to perform defect prediction using quad-tree and spectral clustering and Nam and Kim [70] used clustering to label the unlabeled datasets automatically. Moreover, the cluster-based approach is also used in the data filtering area, which is the emphasis of this paper.

The notion of cluster-based methods is to find groups of instances (the combination of training instances and testing instances) such that the characteristics of an instance are as similar as possible to that of others in the same cluster [46], [47]. Based on this idea, the data similarity in each cluster is used to identify the training data near to the testing data. Jureczko and Madeyski [71] experimented with the clustering software project and determined that the clusters of software entities are available. They predicted that the prediction model performs well on all projects located in the same cluster. The cluster-based approach has a general procedure consisting of several steps, as shown in Algorithm 3. Selecting relevant training datasets using this approach can be conducted according to the granularity level of the source dataset. The studies in [27] [28] [48] proposed an instance-level approach, while the authors in [16] [44] employed a project-level approach to select the relevant source data.

Kawata et al. [27] proposed to use density-based spatial clustering (DBSCAN) [72] for the selection of relevant training data. Adopting the general procedure, they used DBSCAN to create clusters of training data. By using the selected training data, they construct a defect prediction model using linear regression, random forest, naive Bayes, and K Nearest Neighbor as the predictor. They conducted an empirical study using 56 datasets available from different open source projects repository and concluded that using DBSCAN can improve the predictive performance of the model. Kawata et al. 's method outperforms Burak filter and Peter filter, with regards to G-measure and AUC. This method, in part, employed a similar procedure to Peter's, they partition cross-project and within-project simultaneously. However, DBSCAN has a more straightforward procedure because this method itself removes irrelevant data.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Using the general procedure above, Yu et al. [48] studied the use of the clustering method to build a high-performance prediction model. They employed agglomerative clustering for partitioning the group into clusters. Differ to Kawata et al. 's work that five predictors, they used support vector machine and naive Bayes as the predictors. The result showed that this approach provides a promising result as it successfully reduces the detrimental effect of irrelevant training data. Their method outperforms Burak, Peter, and Kawata filter.

Later, Yu et al. [28] also adopted the same approach as in [27][48]. However, to improve the previous studies [6], [15], [25], [27], [48] focusing on the distance between dataset when filtering training data, they consider class information (class label) when selecting the relevant training data. Yu et al. used semi-supervised density-based spatial clustering [73] to filter out irrelevant training data. They employed mixed data as source data. This method creates clusters of source data, and then for each cluster selected, it chooses source data instances having the same class label as that of testing data instance as the relevant training data. To validate the success of their proposed approach, Yu et al. experimented with 15 datasets available from different open source projects repository. As the classifier, they use linear regression, random forest, and Naive Bayes (NB). This study concludes that using a cluster-based method could enhance the performance of defect predictor. The results show that Yu et al. 's method outperforms the Burak filter and Peter filter.

To address the runtime and scalability problem, Herbold [16] proposed a characteristic-based filtering method, which uses distributional characteristics to detect similarity among datasets (i.e., source datasets and target datasets). Firstly, it computes the distributional characteristics of each dataset. This characteristic is represented as the vector of distributional characteristics (i.e., mean and standard deviation). Secondly, this approach merges the vector of the distributional characteristic of the source dataset and the target dataset into a single set. Subsequently, this method partitions the combined set into clusters of distributional characteristic vectors using the K-Means and chooses the source data in the same group as target data as training data. Herbold experimented with a cross-project prediction using 71 version datasets available from 38 different open source projects. The proposed method has proven to be very efficient and be able to deal with the scalability problem. However, similar to [5], [6], the result is still unfavorable for actual use.

Li et al. [44] introduced a two-level training data selection method to address the issue found in [16]. At the first level, they adopted Herbold's method to filter the training data instances according to the source and the target project similarity, measured using four data characteristics (i.e., min, max, mean, and standard deviation). After collecting the training data instances, at the second level, the method employs the K-means algorithm to create clusters of similar instances [15]. The final training data instances are instances that are collected from each cluster having at least one target instance. The experiment concludes that the predictive performance of the proposed method showing a favorable result. This filter also outperforms the Burak Filter, Peter Filter, and Herbold with regards to the average of AUC.

Each cluster-based method mentioned above develops a single defect prediction model using the instances collected from all clusters. Menzies et al. [74], [75] introduced the notion of the local filter. The core idea of this filter is to partitions the datasets into subsets of data whose similar properties, and then for each cluster, a defect prediction model is built accordingly. Menzies et al. [74] also introduced a global filter, which uses all available source data to construct an SDP model. Algorithm 4 presents a generic algorithm for local filters. Menzies et al. found that there may be an advantage of using such an approach when training a machine learning model. Later on, several studies such as [76], [77], [78], [79], [38], [58], have investigated the validity of using local model.

Algorithm 4 Local cluster based approach [44]

input: *source instances (**S_S**), target instances (**T_T**)*
output: *average predictive performance*

combine source instances and target instances into one group.
partition the group into clusters
retain the cluster having at least one target instances

for each *retained cluster* ***do***
  Collect all training instances.
  Collect all testing instances
  Build defect prediction model using training instances
  Predict defect on testing instances
end

return *average predictive performance*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The results of the studies were inconclusive. Bettenburg et al. [76], [77] confirmed the result of Menzies et al. They concluded that using local filter results in a better fit and predictive performance than the global filter. However, studies in [38], [58], [79] found rather different results. Herbold et al. [38] applied the local filter in the cross-project study and found that the performance of both models differs slightly according to F-measure and precision. Further studies explored the use of the local model in effort-aware [58] and just-in-time defect prediction [79]. They concluded that the local model underperforms the global model with regards to the classification performance. However, the local model is superior to the global model, with regards to effort-aware performance. The result also discovered that the data size affects the performance of the local model. The small size of the dataset worsens the performance of the local model. Bettenburg et al. also revealed that the critical factors affecting the successful use of the local model are the variance of the datasets, the choice of the clustering algorithm as well as the careful setting of the algorithm parameters (especially for the parametric algorithm).

3.3. Evolutionary-based Filter

Recent research on training data selection has adopted an evolutionary-based approach. Hosseini et al. [52], [53] investigated a search-based instance selection using a genetic algorithm filter to generate relevant training data. This approach aims to deal with the data quality issue, i.e., noisy data. They proposed a genetic algorithm to guide training data selection to take the dataset having the same characteristics as the testing dataset. Similar to Burak filter [6], this method employs the testing instance as guidance on choosing the relevant training data instance.

Training data selection starts when this method divides the testing dataset into several parts at random. It then inputs each piece into the Nearest Neighbor filter to choose the most suitable training instances. These instances will later be used as the validation set. Subsequently, the genetic algorithm is utilized to generate the best training dataset. To validate their proposed approach, Hosseini et al. experimented with 13 open-source datasets. The selected training data are employed to train defect models using Naive Bayes (NB) as the predictor. The empirical result confirmed that the proposed method could enhance the defect prediction performance, which is better than NN filter, naive CPDP, and WPDP. Similar results of using an evolutionary-based method also found in [80]. It is consistent with the recommendation of Malhotra [81] stated that having high accuracy and high AUC, the search-based technique is suitable for building the defect prediction model. Because of its iterative optimization process, however, this method has a high computational time.

4. DISCUSSION

This section provides a brief discussion about the comparative performance of the approaches reviewed in section 3. Most training data selection method in the literature falls into a nearest-neighbor category, which consists of instance-level and project level filtering. The instance-level approach generally uses the K-Nearest Neighbor algorithm. This algorithm is simple; however, it retains noisy instances [82] and does not scale with large data. Based on some reported results [44], [83], Peter Filter seems to outperform the other instance-level methods. It may because Peter Filter only selects the representative training data so that it enhances the suitability of selected training data.

Meanwhile, project-based filter differs from the instance-based filter in the way they measure the source and target data similarity. In the instance-based filter, such as Turhan filter or Peter filter, measuring similarity is performed on each pair of training data instances and target data instance. While, in the project-based filter, the similarity between two project datasets is calculated based on the pair-wise project, which represented the distributional characteristics of the datasets. Therefore, the project-level filtering approach has a lower computational time than the instance-level methods. It is also scalable to large datasets. However, using such coarse-level granularity may result in false alarm as the selected training data may include irrelevant training instances [41].

Most studies under review found that training data selection using a cluster-based method performs better than the distance-based approaches. For example, Li et al. [44] found that local filter outperforms Peter Filter, Burak Filter, and Herbold. The other studies, such as [27], [48] also provide similar results. Recent research by Bin et al. [83] proposed a retaining ratio, a measure that corresponds to the number of selected training data. They measured the retaining ratio for several filters and investigated the effect of the retaining ratio on the performance. They found that cluster-based

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

filters [27][76] have a higher training ratio than distance-based based filters [6], [15]. Based on the retaining ratio, cluster-based filter better than the pointwise based filter, which conforms to the above finding.

With regards to the performance, the question about the effect of the dataset granularity on the predictive performance seems to be inconclusive. The result of [44], for example, shows that Herbold's approach performs worse than Turhan Filter and Peter Filter when using NB as the predictor. However, when using SVM as the predictor, Herbold's method outperforms the two instance-based filter. This phenomenon may be explained using the conclusion of Menzies et al. [64] and Wanatabe et al. [84]. They argued that each dataset has different characteristics. Also, the behavior of a predictor is affected by the dataset characteristic. For example, C4.5 performs worse for imbalanced data, while K-Nearest Neighbor is sensitive to noise. Hall et al. [18] found that the predictive performance of model may be dependent on the predictor. Therefore, datasets characteristic rather than data granularity that affects the prediction performance.

5. CONCLUSION AND RESEARCH DIRECTION

This article presents the review of the current state of training data selection methods. Through a literature review, the existing methods are categorized into three groups of nearest-neighbor, cluster-based, and evolutionary-based approaches. For each category, training data selection can be conducted at instance-level and project-level. The strength and limitations of each group are summarized and presented in Table 1. Table 3 shows that most training data selection method in the literature falls into a nearest-neighbor category, which consists of instance-level and project level filtering. Most methods use the KNN algorithm that has simple steps; however, it has a polynomial run time and scalability problem with large datasets. Moreover, for project-level granularity, the selected training data instances may include irrelevant source instances, which may result in a false alarm rate. The cluster-based method performs better than the distance-based approaches. Nevertheless, it is sensitive to the choice of the parameters of the clustering method. The evolutionary-based method provides promising accuracy, but it much slower than the other methods.

In software defect prediction, training data selection is an important task, since training data selection can select the training instances similar to testing instances, which can deal with the distribution difference problem. The selection of relevant training data increases the prediction performance of the model. Therefore, further research on this issue needs to be performed. Improving the method of selecting training data is a promising way to decrease distributional differences between training and test data.

Several research challenges in this area need to be addressed. First, previous studies reported that using data characteristics is of benefit for training data selection. Several training data selection methods use distributional data characteristics to assess the datasets similarity among the source and target projects. This approach has proven to be able to deal with the run time and scalability problem. However, each study used a different combination of distributional data characteristics. For example, He et al. [5] used sixteen data characteristics, Herbold [16] used two data characteristics (mean, standard deviation), while Li et al. [44] used four data characteristics (max, min, mean, standard deviation). Most studies did not mention their reason for choosing such data characteristics. Those studies claimed that the predictive performance of their proposed model improved, even though the overall success rate was still incomparable to that within project defect prediction. Therefore, dealing with relevant training data is still an open issue. Further research needs to investigate the most suitable distributional characteristics for building a defect prediction model.

Secondly, There are many clustering algorithms in the literature [46], [47], therefore exploring other clustering algorithms for selecting the training data is an excellent opportunity for investigation, especially for the parametric algorithm. Besides, the comparison of prediction performance among existing cluster-based method is limited; therefore, which cluster-based techniques present the best result still in question. What is more, most of the studies working on this area use the non-parameter clustering algorithm. Bettenburg et al. [77] point out that the clustering algorithm, along with its parameter calibration, affects the effectiveness of a clustering model. For further explanation, consult Bettenburg et al. [77] and Tantithamthavorn et al. [85].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Finally, there is a limited study on the use of the evolutionary method, such as a search-based technique, to filter the training data. Results witnessed that the advanced approach provides promising predictive performance compared to the traditional way. Hossein et al. [52], [53] found that the genetic algorithm can boost the performance of a simple naive Bayes classifier. Therefore, more studies on this approach, combined with clustering and distributional characteristics, is still an open area for research. Malhotra et al. [86] also recommended further investigation reducing the the long computational time of this techniques.

ACKNOWLEDGEMENT

This research has been supported by Universitas Atma Jaya Yogyakarta Indonesia and Universiti Teknikal Malaysia Melaka.

REFERENCES:

[1] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 485–496, 2008.

[2] P. He *et al.*, "Simplification of Training Data for Cross-Project Defect Prediction," in *http://arxiv.org/*, 2014.

[3] Y. Zhou *et al.*, "How far we have progressed in the journey? An examination of cross-project defect prediction," *ACM Trans. Softw. Eng. Methodol.*, vol. 27, no. 1, pp. 1:1--1:51, Apr. 2018.

[4] H. Ji and S. Huang, "A New Framework Consisted of Data Preprocessing and Classifier Modelling for Software Defect Prediction," *Math. Probl. Eng.*, vol. 2018, 2018.

[5] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, "An investigation on the feasibility of cross-project defect prediction," *Autom. Softw. Eng.*, vol. 19, no. 2, pp. 167–199, Jun. 2012.

[6] B. Turhan, T. Menzies, B. Bener, A. B. Bener, and J. Di Stefano, "On the Relative Value of Cross-Company and Within-Company Data for Defect Prediction," *Empir. Softw. Eng.*, vol. 14, no. 5, pp. 540–578, Oct. 2009.

[7] B. A. Kitchenham, E. Mendes, and G. H. Travassos, "Cross versus within-company cost estimation studies: A systematic review," *IEEE Trans. Softw. Eng.*, vol. 33, no. 5, pp. 316–329, 2007.

[8] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-Project Defect Prediction," in *Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering on European software engineering conference and foundations of software engineering symposium - E*, 2009, pp. 91–100.

[9] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *Proceedings - International Conference on Software Engineering*, 2013, pp. 382–391.

[10] A. E. C. Cruz and K. Ochimizu, "Towards logistic regression models for predicting fault-prone code across software projects," *2009 3rd Int. Symp. Empir. Softw. Eng. Meas. ESEM 2009*, pp. 460–463, 2009.

[11] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," *Inf. Softw. Technol.*, vol. 54, no. 3, pp. 248–256, Mar. 2012.

[12] S. Watanabe, H. Kaiya, and K. Kaijiri, "Adapting a fault prediction model to allow inter language reuse," *Proc. - Int. Conf. Softw. Eng.*, pp. 19–24, 2008.

[13] F. Zhang, I. Keivanloo, and Y. Zou, "Data Transformation in Cross-project Defect Prediction," *Empir. Softw. Eng.*, vol. 22, no. 6, pp. 3186–3218, Dec. 2017.

[14] Y. Jiang, B. Cukic, and T. Menzies, "Can data transformation help in the detection of fault-prone modules?," in *DEFECTS'08: 2008 International Symposium on Software Testing and Analysis - Proceedings of the 2008 Workshop on Defects in Large Software Systems 2008, DEFECTS'08*, 2008, pp. 16–20.

[15] F. Peters, T. Menzies, and A. Marcus, "Better cross company defect prediction," in *IEEE International Working Conference on Mining Software Repositories*, 2013, pp. 409–418.

[16] S. Herbold, "Training data selection for cross-project defect prediction," in *Proceedings of the 9th International Conference on Predictive Models in Software Engineering - PROMISE '13*, 2013, pp. 1–10.

[17] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Appl. Soft Comput. J.*, vol. 27, pp. 504–518, Feb. 2015.

[18] T. Hall, S. Beecham, D. Bowes, D. Gray, and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

S. Counsell, "A Systematic Literature Review on Fault Prediction Performance in Software Engineering," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1276–1304, 2012.

[19] S. S. Rathore and S. Kumar, "A study on software fault prediction techniques," *Artif. Intell. Rev.*, pp. 1–73, 2017.

[20] C. Catal, "Expert Systems with Applications Software fault prediction : A literature review and current trends," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 4626–4636, 2011.

[21] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 7346–7354, 2009.

[22] D. Radjenović, M. Heričko, R. Torkar, A. Živkovič, and D. Radjenovic, "Software Fault Prediction Metrics : A Systematic Literature Review," *Inf. Softw. Technol.*, vol. 55, no. 8, pp. 1397–1418, Aug. 2013.

[23] S. Hosseini, B. Turhan, and D. Gunarathna, "A Systematic Literature Review and Meta-Analysis on Cross Project Defect Prediction," *IEEE Trans. Softw. Eng.*, vol. X, no. 2, pp. 111–147, 2019.

[24] B. Turhan, A. Tosun Misirli, and A. Bener, "Empirical evaluation of the effects of mixed project data on learning defect predictors," in *Information and Software Technology*, 2013, vol. 55, no. 6, pp. 1101–1118.

[25] L. Chen, B. Fang, Z. Shang, and Y. Tang, "Negative samples reduction in cross-company software defects prediction," *Inf. Softw. Technol.*, vol. 62, no. 1, pp. 67–77, Jun. 2015.

[26] D. Ryu, J. I. Jang, and J. Baik, "A Hybrid Instance Selection Using Nearest-Neighbor for Cross-Project Defect Prediction," *J. Comput. Sci. Technol.*, vol. 30, no. 5, pp. 969–980, Sep. 2015.

[27] K. Kawata, S. Amasaki, and T. Yokogawa, "Improving relevancy filter methods for cross-project defect prediction," in *Proceedings - 3rd International Conference on Applied Computing and Information Technology and 2nd International Conference on Computational Science and Intelligence, ACIT-CSI 2015*, 2016, pp. 2–7.

[28] X. Yu, M. Wu, Y. Jian, K. E. Bennin, M. Fu, and C. Ma, "Cross-company defect prediction via semi-supervised clustering-based data filtering and MSTrA-based transfer learning," *Soft Comput.*, vol. 22, no. 10, pp. 3461–3472, May 2018.

[29] F. Zhang, A. Mockus, I. Keivanloo, and Y. Zou, "Towards building a universal defect prediction model with rank transformed predictors," *Empir. Softw. Eng.*, vol. 21, no. 5, pp. 2107–2145, Oct. 2016.

[30] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[31] A. Arnold, R. Nallapati, and W. W. Cohen, "A comparative study of methods for transductive transfer learning," in *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2007, pp. 77–82.

[32] Y. Shi, Z. Lan, W. Liu, and W. Bi, "Extending semi-supervised learning methods for inductive transfer learning," *Proc. - IEEE Int. Conf. Data Mining, ICDM*, pp. 483–492, 2009.

[33] X. Xia, D. Lo, S. J. Pan, N. Nagappan, and X. Wang, "HYDRA: Massively Compositional Model for Cross-Project Defect Prediction," *IEEE Trans. Softw. Eng.*, vol. 42, no. 10, pp. 977–998, Oct. 2016.

[34] F. Zhang, A. Mockus, I. Keivanloo, and Y. Zou, "Towards building a universal defect prediction model with rank transformed predictors," *Empir. Softw. Eng.*, vol. 21, no. 5, pp. 2107–2145, 2016.

[35] S. Kim, H. Zhang, R. Wu, and L. Gong, "Dealing with noise in defect prediction," in *Proceedings of the 33rd International Conference on Software Engineering*, 2011, pp. 481–490.

[36] W. Tang and T. M. Khoshgoftaar, "Noise identification with the k-means algorithm," in *16th IEEE International Conference on Tools with Artificial Intelligence*, 2005, pp. 373–378.

[37] E. Kocaguneli, T. Menzies, J. Keung, D. Cok, and R. Madachy, "Active Learning and effort estimation: Finding the essential content of software effort estimation data," *IEEE Trans. Softw. Eng.*, vol. 39, no. 8, pp. 1040–1053, 2013.

[38] S. Herbold, A. Trautsch, and J. Grabowski, "Global vs. local models for cross-project defect prediction: A replication study," *Empir. Softw. Eng.*, vol. 22, no. 4, pp. 1866–1902, 2017.

[39] B. Turhan, A. Tosun Misirli, A. Bener, A. Tosun Mısırlı, and A. Bener, "Empirical evaluation of the effects of mixed project data on learning defect predictors," in *Information and Software Technology*, 2013, vol. 55, no. 6,

pp. 1101–1118.

[40] Z. He, F. Peters, T. Menzies, and Y. Yang, "Learning from open-source projects: An empirical study on defect prediction," in *International Symposium on Empirical Software Engineering and Measurement*, 2013, pp. 45–54.

[41] P. He, B. Li, D. Zhang, and Y. Ma, "Simplification of Training Data for Cross-Project Defect Prediction," 2014.

[42] W. Wen, B. Zhang, X. Gu, and X. Ju, "An Empirical Study on Combining Source Selection and Transfer Learning for Cross-Project Defect Prediction," in *IBF 2019 - 2019 IEEE 1st International Workshop on Intelligent Bug Fixing*, 2019, pp. 29–38.

[43] C. Liu, D. Yang, X. Xia, M. Yan, and X. Zhang, "A two-phase transfer learning model for cross-project defect prediction," *Inf. Softw. Technol.*, vol. 107, no. November 2018, pp. 125–136, Mar. 2018.

[44] Y. Li, Z. Huang, Y. Wang, and B. Fang, "Evaluating data filter on cross-project defect prediction: Comparison and improvements," *IEEE Access*, vol. 5, pp. 25646–25656, 2017.

[45] A. Bispo, R. Prudencio, and D. Veras, "Instance selection and class balancing techniques for cross project defect prediction," *Proc. - 2018 Brazilian Conf. Intell. Syst. BRACIS 2018*, pp. 552–557, 2018.

[46] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[47] D. Xu and Y. Tian, "A Comprehensive Survey of Clustering Algorithms," *Ann. Data Sci.*, vol. 2, no. 2, pp. 165–193, 2015.

[48] X. Yu, J. Liu, W. Peng, and X. Peng, "Improving Cross-Company Defect Prediction with Data Filtering," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 27, no. 09n10, pp. 1427–1438, Nov. 2017.

[49] R. Malhotra, M. Khanna, and R. R. Raje, "On the application of search-based techniques for software engineering predictive modeling : A systematic review and future directions," *Swarm Evol. Comput.*, vol. 32, no. August 2016, pp. 85–109, 2017.

[50] R. Malhotra, "Search based techniques for software fault prediction: Current trends and future directions," *7th Int. Work. Search-Based Softw. Testing, SBST 2014 - Proc.*, pp. 35–36, 2014.

[51] M. Harman and B. F. Jones, "Search-based software engineering," *Inf. Softw. Technol.*, vol. 43, no. 14, pp. 833–839, 2001.

[52] S. Hosseini, B. Turhan, and M. Mäntylä, "A benchmark study on the effectiveness of search-based data selection and feature selection for cross project defect prediction," *Inf. Softw. Technol.*, vol. 95, pp. 296–312, Mar. 2018.

[53] S. Hosseini, B. Turhan, and M. Mäntylä, "Search Based Training Data Selection For Cross Project Defect Prediction," in *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering - PROMISE 2016*, 2016, pp. 1–10.

[54] Aarti, G. Sikka, and R. Dhir, "An investigation on the effect of cross project data for prediction accuracy," *Int. J. Syst. Assur. Eng. Manag.*, vol. 8, no. 2, pp. 352–377, Jun. 2017.

[55] P. He, Y. He, L. Yu, and B. Li, "An Improved Method for Cross-Project Defect Prediction by Simplifying Training Data," *Math. Probl. Eng.*, vol. 2018, 2018.

[56] J. Soler, F. Tenc, L. Gaubert, and B. Cedric, "Data Clustering and Similarity," in *Twenty-Sixth International Florida Artificial Intelligence Research Society Conference*, 2013, pp. 492–495.

[57] P. He, Y. He, L. Yu, and B. Li, "An Improved Method for Cross-Project Defect Prediction by Simplifying Training Data," *Math. Probl. Eng.*, vol. 2018, 2018.

[58] X. Yang, H. Yu, G. Fan, K. Shi, and L. Chen, "Local versus Global Models for Just-In-Time Software Defect Prediction," *Sci. Program.*, vol. 2019, pp. 1–13, 2019.

[59] Z. Li, X. Y. Jing, X. Zhu, H. Zhang, B. Xu, and S. Ying, "On the Multiple Sources and Privacy Preservation Issues for Heterogeneous Defect Prediction," *IEEE Trans. Softw. Eng.*, vol. 45, no. 4, pp. 391–411, Apr. 2019.

[60] T. Fukushima, Y. Kamei, S. McIntosh, K. Yamashita, and N. Ubayashi, "An empirical study of just-in-time defect prediction using cross-project models," *dl.acm.org*, pp. 172–181, 2014.

[61] Y. Kamei, T. Fukushima, S. McIntosh, K. Yamashita, N. Ubayashi, and A. E. Hassan, "Studying just-in-time defect prediction using cross-project models," *Empir. Softw. Eng.*, vol. 21, no. 5, pp. 2072–2106, Oct. 2016.

[62] B. Turhan, "On the dataset shift problem in

software engineering prediction models," *Empir. Softw. Eng.*, vol. 17, no. 1–2, pp. 62–74, 2012.

[63] H. Wang, T. M. Khoshgoftaar, and N. Seliya, "How Many Software Metrics Should be Selected for Defect Prediction ?," *Proc. Twenty-Fourth Int. Florida Artif. Intell. Res. Soc. Conf.*, no. Mi, pp. 69–74, 2011.

[64] T. Menzies, J. Greenwald, and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 2–14, 2007.

[65] S. Qiu, L. Lu, and S. Jiang, "Multiple-Components Weights Model for Cross-Project Software Ddefect Prediction," *IET Softw.*, vol. 12, no. 4, pp. 345–355, Aug. 2018.

[66] J. Chen *et al.*, "Multiview Transfer Learning for Software Defect Prediction," *IEEE Access*, vol. 7, no. c, pp. 8901–8916, 2019.

[67] F. Zhang, A. Mockus, I. Keivanloo, and Y. Zou, "Towards building a universal defect prediction model," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 182–191.

[68] F. Zhang, Q. Zheng, Y. Zou, and A. E. Hassan, "Cross-project defect prediction using a connectivity-based unsupervised classifier," in *Proceedings - International Conference on Software Engineering*, 2016, vol. 14-22-May-, pp. 309–320.

[69] P. S. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree-based K-means clustering algorithm," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 6, pp. 1146–1150, 2012.

[70] J. Nam and S. Kim, "CLAMI: Defect prediction on unlabeled datasets," in *Proceedings - 2015 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015*, 2015, pp. 452–463.

[71] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering - PROMISE '10*, 2010, p. 1.

[72] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Kdd*, vol. 96, no. 34, pp. 226–231, 1996.

[73] L. Lelis and J. Sander, "Semi-supervised density-based clustering," in *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2009, pp. 842–847.

[74] T. Menzies, A. Butcher, A. Marcus, T. Zimmermann, and D. Cok, "Local vs. global models for effort estimation and defect prediction," in *2011 26th IEEE/ACM International Conference on Automated Software Engineering, ASE 2011, Proceedings*, 2011, pp. 343–351.

[75] T. Menzies *et al.*, "Local versus global lessons for defect prediction and effort estimation," *IEEE Trans. Softw. Eng.*, vol. 39, no. 6, pp. 822–834, 2013.

[76] N. Bettenburg, M. Nagappan, and A. E. Hassan, "Think locally, act globally: Improving defect and effort prediction models," in *IEEE International Working Conference on Mining Software Repositories*, 2012, pp. 60–69.

[77] N. Bettenburg, M. Nagappan, and A. E. Hassan, "Towards improving statistical modeling of software engineering data: think locally, act globally!," *Empir. Softw. Eng.*, vol. 20, no. 2, pp. 294–335, 2015.

[78] G. Scanniello, C. Gravino, A. Marcus, and T. Menzies, "Class level fault prediction using software clustering," in *2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013 - Proceedings*, 2013, pp. 640–645.

[79] M. El Mezouar, F. Zhang, and Y. Zou, "Local versus Global Models for Effort-aware Defect Prediction," in *Proceedings of the 26th Annual International Conference on Computer Science and Software Engineering*, 2016, pp. 178–187.

[80] W. Rhmann, "Cross project defect prediction using hybrid search based algorithms," *Int. J. Inf. Technol.*, Aug. 2018.

[81] R. Malhotra, "Search based techniques for software fault prediction: current trends and future directions," 2014, pp. 35–36.

[82] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. Kittler, "A review of instance selection methods," *Artif. Intell. Rev.*, vol. 34, no. 2, pp. 133–143, 2010.

[83] Y. Bin, K. Zhou, H. Lu, Y. Zhou, and B. Xu, "Training Data Selection for Cross-Project Defection Prediction: Which Approach Is Better?," in *International Symposium on Empirical Software Engineering and Measurement*, 2017, vol. 2017-Novem, pp. 354–363.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[84] S. Watanabe, H. Kaiya, and K. Kaijiri, "Adapting a fault prediction model to allow inter languagereuse," in *Proceedings of the 4th international workshop on Predictor models in software engineering - PROMISE '08*, 2008, p. 19.

[85] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "Automated parameter optimization of classification techniques for defect prediction models," in *ieeexplore.ieee.org*, 2016, pp. 321–332.

[86] R. Malhotra, M. Khanna, and R. R. Raje, "On the application of search-based techniques for software engineering predictive modeling: A systematic review and future directions," *Swarm Evol. Comput.*, vol. 32, no. October 2016, pp. 85–109, Feb. 2017.