Innovations in Systems and Software Engineering (2022) 18:263–281
https://doi.org/10.1007/s11334-020-00380-5

S.I. : ACITSEP

Cross project defect prediction: a comprehensive survey with its SWOT analysis

Yogita Khatri[1] · **Sandeep Kumar Singh**[2]

Received: 15 September 2020 / Accepted: 23 November 2020 / Published online: 3 January 2021
© The Author(s), under exclusive licence to Springer-Verlag London Ltd. part of Springer Nature 2021

Abstract
Software fault prediction (SFP) refers to the process of identifying (or predicting) faulty modules based on its characteristics/software metrics. SFP can be done either using the same project data in both the training and testing phase i.e. within project defect prediction or using a different one, as done in cross-project defect prediction (CPDP). Previous works show that contemporary research in this field is progressing towards CPDP. To present the current state of progress and the future prospects of CPDP, this article presents a comprehensive survey of CPDP considering the latest work along with its SWOT analysis. This survey is targeted to present the novice researchers, academicians, and practitioners with the alphas and omegas of this contemporary challenging field. We have also carried a qualitative and quantitative evaluation of CPDP w.r.t some of the targeted research questions. A total of 34 significant primary CPDP studies published from 2008 to 2019 were selected. Both qualitative and quantitative data are extracted from each study. The collected data is then consolidated and analyzed to present a comprehensive report showing the current state of the art, along with the answers to the targeted research questions and finally the CPDP SWOT analysis. We observed that there exists a big scope for performance improvement in CPDP. Integration of feature engineering, exploration with different process metrics, hyperparameter tuning, class imbalance handling in CPDP setting are some of the ways identified for bringing enhancement in CPDP performance. Apart from this, we would like to conclude that there is a strong need to investigate Precision over the Recall and model's validity in terms of effort/cost-effectiveness.

Keywords Cross-project defect prediction · SWOT analysis · Software metrics · Transfer learning

1 Introduction

Software fault/defect prediction is an evergreen topic in the realm of software engineering. Its capability to identify/predict faulty modules before their release can help the software quality assurance personnel to utilize limited resources optimally. More efforts can be done on faulty modules as compared to non-faulty modules to achieve high confidence in the quality and reliability of the software.

Software fault prediction is divided into two branches, based on the project data used in the training and the testing phase: within project defect prediction (WPDP) and cross-project defect prediction. (CPDP). **Within project defect prediction** deals with using the same project data in both the training and testing phase whereas **cross-project defect prediction** involves using different project data in the training and testing phase. More formally, we can say, the model trained on project X can be used for testing project Y. The success of a prediction model majorly depends on the availability of suitable data. However, when an ample amount of suitable data is not available, the prediction model is at risk, since most of the machine learning models are built on the ground truth available in the training dataset. Thus, when enough historical data is not available, then generally, cross-project defect prediction comes into play.

To the best of our knowledge, the very first attempt to investigate the feasibility of using a prediction model built using the data of one project, for the testing of another project, was made by Briand et al. [1]. However, the results were discouraging. Zimmermann et al. [2] were also amongst the

✉ Yogita Khatri
yogitakhatri@jssaten.ac.in

[1] Department of Information Technology, JSS Academy of Technical Education, Noida, India

[2] Department of Computer Science Engineering and Information Technology, Jaypee Institute of Information Technology, Noida, India

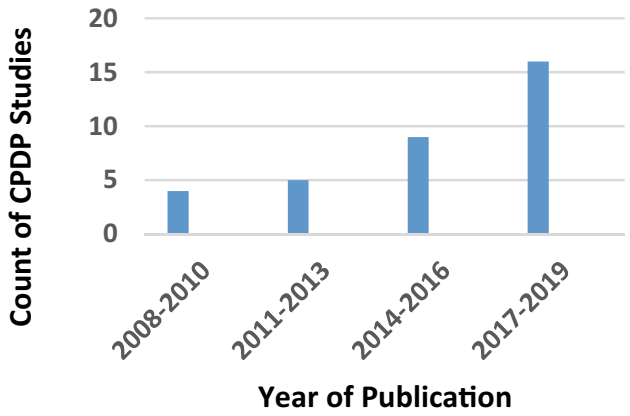

Fig. 1 Distribution of selected CPDP studies published from 2008 to 2019

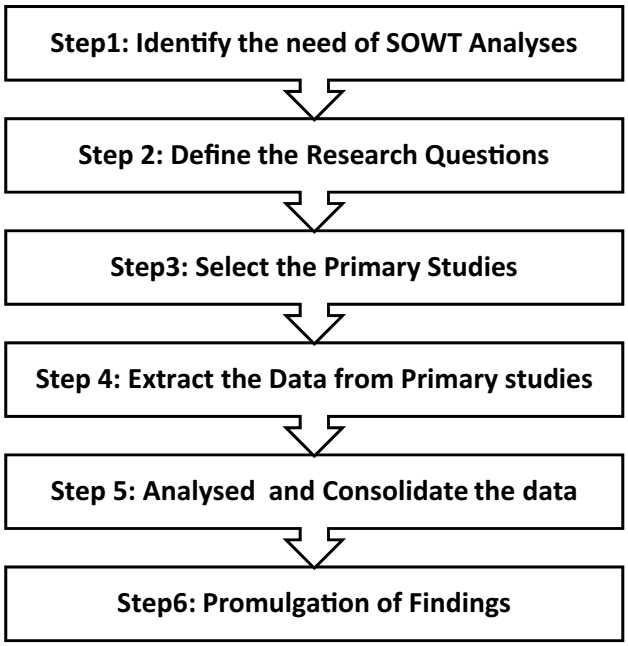

Fig. 2 Methodology adopted

few, who performed an initial study on CPDP. They carried a total of 622 cross-project experiments using logistic regression classifier, out of which only 3.4% of the total experiments were successful. The success of the CPDP model lies in reducing the distribution difference between the source and the target data, which was ignored and became the main reason for failure in the initial attempts. Thereafter, the research community has witnessed the growth in the researchers' interest especially concerning CPDP, as can be seen in Fig. 1.

Thus, plenty of research has been published, targeting different approaches to CPDP. To the best of our knowledge, nobody has yet attempted to carry the SWOT analysis of CPDP, to clearly project its current state of progress. Therefore, in this article, we have carried the SWOT analysis of cross-project defect prediction as our prime objective. This review is basically intended to help, particularly the novice researchers, academicians, and practitioners with the alphas and omegas of this contemporary challenging field. This article also presents a qualitative and quantitative evaluation of cross-project defect prediction, aligned with the research questions targeted. We identified, critiqued, evaluated, and summarized the existing research on cross-project defect prediction to find out its strengths, weaknesses, opportunities, and threats.

Further, we extracted the data to answer some specific research questions. Figure 2 illustrates, how the entire work is carried out and presented in this article. It starts with first, identifying the purpose and targeted audience of this SWOT analysis, which has already been explained above, followed by defining some specific research questions to keep the work focused. The research questions are presented in Table 1.

Several digital databases like ACM Digital Library, IEEE eXplore, Scopus, Springer, and Google Scholar were then explored to carefully select the primary studies as per the inclusion/exclusion guidelines mentioned in Table 2. We focused majorly on homogeneous CPDP approaches and a total of 34 significant primary studies were selected, out of which 30 are homogeneous CPDP studies and 4 are heterogeneous CPDP studies. Figure 3 shows the distribution of studies w.r.t different journals and conferences, highlighting the most significant software defect prediction journals and conferences. The quantitative data (consisting of datasets used, kinds of metrics used, modeling technique applied, evaluation measures used and statistical test applied) and qualitative data, both are extracted from each study. The collected data is then consolidated and analyzed to present SWOT analysis and the answers to the proposed research questions.

The rest of the paper is structured as follows: Sect. 2 provides a comprehensive report on cross-project defect prediction whereas Sect. 3 presents the discussion in line with the research questions undertaken. Section 4 presents the SWOT analysis of CPDP. Section 5 highlights the potential threats in the study and finally, Sect. 6 concludes the paper.

2 Comprehensive report on CPDP

Different researchers proposed different CPDP approaches. Figure 4 shows the two broad categories namely homogeneous CPDP and heterogeneous CPDP, in which the selected studies are grouped. Homogeneous CPDP involves the use of the same metrics set across the source dataset and target dataset, whereas, in heterogeneous

Table 1 Research questions

Research question ID	Research questions	Motivation
RQ1	What kind of datasets have been mostly used in CPDP?	To identify the datasets in CPDP
RQ2	What kind of modeling techniques have been mostly used in CPDP?	To identify modeling techniques in CPDP
RQ3	What kind of metrics have been mostly used in CPDP?	To identify metrics in CPDP
RQ4	What kind of Evaluation Parameters have been mostly used in CPDP?	To identify Evaluation Parameters in CPDP
RQ5	What kind of Statistical tests have been performed in CPDP?	To identify the Statistical Test in CPDP

Table 2 Inclusion/exclusion criteria

Inclusion criteria
Studies having some significant empirical investigation
Studies published in English only
Studies focusing on fault prediction using software metrics
Exclusion criteria
Studies dealing with literature reviews and systematic reviews
Non-empirical studies
Studies discussing defect datasets but not focusing on defect prediction as the target variable

CPDP, the source and the target datasets have dissimilar metrics set. For example, the datasets *JM1* and *KC1* from NASA MDP [3] contain 21common features each and *the ar4* dataset from a Turkish company named SOFTLAB contains 29 features. Both groups share 17 features in common. A CPDP model built with taking *JM1* as the source data and the *KC1* as the target data would then be classified as a homogeneous CPDP since each has the same metric set, however a CPDP model built using *JM1* as the source data and *ar4* as the target data or vice versa would belong to heterogeneous CPDP as both contain, unlike metric set. But, if selected 17 common metrics from each, then it will come under homogeneous CPDP. In this survey, we majorly focussed on homogeneous CPDP approaches, however, we have included some significant heterogeneous CPDP approaches in our survey to present a holistic view of CPDP.

Different CPDP studies evaluated their proposed model using a diverse set of evaluation parameters. Broadly, they can be grouped under two categories: non-effort based evaluation parameters and effort aware evaluation parameters. Non-effort based evaluation parameters are the standard machine learning evaluation parameters that do not take into account the effort/cost involved in identifying the faulty modules. Precision, Recall (PD), F-measure, G-measure, G-mean, PF (probability of false alarm), AUC (Area under ROC curve), Balance, Accuracy, MCC (Mathew's correlation coefficient) all are non-effort based measures used in the existing CPDP studies. Effort aware measures evaluate the model in terms of effort/cost incurred vs. benefit expected. They actually measure the model's true performance in a practical scenario. PofB20 (percentage of fault identified in 20% of lines), Effort aware Recall, Effort aware Precision, Effort

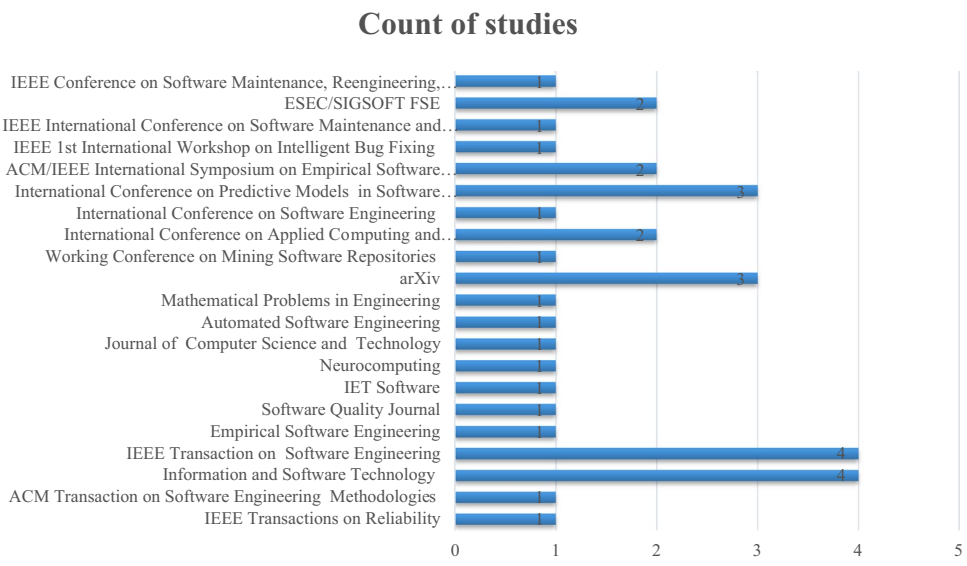

Fig. 3 Distribution of selected studies in Journals/Conferences

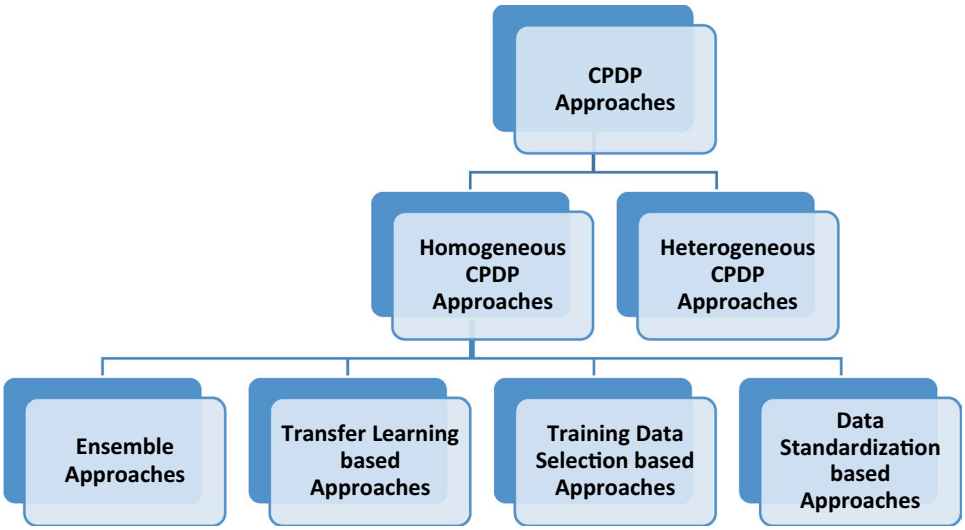

Fig. 4 Different approaches under CPDP

aware F-measure, IFA (initial false alarm) are some of the effort aware evaluation measures used in the existing CPDP studies.

2.1 Homogeneous CPDP approaches

Homogeneous CPDP approaches are further divided under four heads as data standardization-based approaches, training data selection-based approaches, transfer learning-based approaches, and ensemble approaches.

2.1.1 Data standardization based approaches

In cross-project defect prediction, we are extracting knowledge from the source project and applying that learned knowledge to predict labels for the target project. The tasks in both source and target domains are the same, however, the data distribution between the source and the target project may not be the same. For example, consider two projects *ant and camel 1.4 datasets* from the PROMISE repository [4]. Both have 20 features in common. However, the feature space is the same, but the distribution of each feature in both the dataset may differ. For instance, feature LCOM (lack of cohesion in methods) has a different distribution in both the datasets as can be seen in Table 3. Here we have taken five parameters to characterize a distribution i.e. mean, standard deviation, median, min, and max value.

Since machine learning algorithms perform well when both the testing and the target data are from a similar distribution. Therefore, the main crux lies in CPDP is, to make the data distribution similar across the source and the target projects. Initial attempts to make the distribution of the features homogeneous across both the source and the target datasets were made by data standardization-based techniques.

Table 3 Distributional characteristics of feature LCOM in ant and camel datasets

Distributional characteristics	LCOM	
	Ant	Camel 1.4
Mean	89.15	73.42
Std	349.94	429.91
Median	6	4
Min	0	0
max	6692	9792

To this end, Watanabe et al. [5] applied the concept of data standardization where each value of a metric in the target data is normalized by multiplying it by the mean of that metric of the source dataset and then dividing it by the mean of that metric of the target dataset as shown above. Here, M_T represents a metric from the target project and M_S represents the same metric from the source project. They evaluated their proposed work using two open-source projects named *Sakura Editor* and *jEdit*. They investigated two cross-project experiments, one with *jEdit* as a source project and *Sakura Editor* as a target project and the second one, with *jEdit* as a target project and *Sakura Editor* as a source project. The results obtained were promising and better than the traditional CPDP.

$$\text{Normalized value of metric } M_T = \left(\text{value of the Metric } M_T * \text{mean}\left(M_S\right)\right)/\text{mean}\left(M_T\right)$$

Similarly, Cruz and Ochimizu [6] applied the concept of log transformation to standardize the data across the source and the target projects. Log Transformation helps in reducing the skewness in the data and reduces the variability (or spread) as well. It tries to convert the given distribution

to more like a normal distribution, thus making the data distribution across the projects more similar. They made three univariate logistic regression models using the design and complexity metrics of one source project and tested it for two target projects. They observed encouraging results and thus highlighted the potential of cross-project defect prediction. Both studies highlighted the potential of data standardization-based techniques for CPDP, but Precision/Recall was low.

2.1.2 Training data selection based approaches

Training data selection based approaches involve filtering out those instances from the training set that resembles the most with the target dataset.

One of the initial attempts to use this instance filtering method in CPDP task was made by Turhan et al. [7]. They used the concept of K-nearest neighbor filter. For every target sample, they selected the top 10 nearest neighboring samples from the source project. Duplicate samples if any, were removed thereafter, leading to the construction of the final training dataset. They experimented on 10 datasets from NASA [3] and SOFTLAB and observed improvement in the probability of defect detection (PD) and PF as compared to directly using the cross-company (CC) data without any instance filtering for CPDP, but the probability of false alarm was still high as compared to WPDP.

To improve Turhan et al.'s results [7] further, Peter et al. [8] proposed a peter filter for training data selection. First, they combined the source data samples and target data samples in one set, followed by k means clustering and then rejecting the clusters not containing any target sample. In each of the remaining clusters, for each of the source sample, it then selected the nearest target sample. Those selected target samples built the popular set. For every target sample in the popular set, its greatest fan (closest in Euclidean distance) from the source samples was selected as its candidate for the final training data set. The empirical study on 56 datasets from the PROMISE repository [4] concluded the superiority of peter filter (with RF as the classifier) over NN filter [7] in terms of G-measure, however, the result was worse than NN filter [7] in terms of AUC and F1 score.

With a little change in Peter's filtering approach, Kawata et al. [9] proposed a relevancy based filter approach for training data selection. In this approach, source data samples and target data samples were combined as a whole, followed by applying the DBSCAN clustering algorithm and then eliminating clusters, not having any target sample. Source data samples from the remaining clusters form the final training set. The empirical study on 56 datasets from the PROMISE repository [4], carried with four different classifiers namely Random Forest (RF), Naïve Bayes (NB), Logistic Regression (LR), and K Nearest Neighbor (KNN) individually, concluded that their proposed approach outperformed NN filter [7] and peter filter [8] in terms of AUC and G-measure with LR and KNN as the learners. But, the results were discouraging in terms of F1 score.

With a change in the clustering technique in Kawata et al.'s work [9], Yu et al. [10] proposed a data filtering approach based on agglomerative clustering. The experimental results on 15 open-source datasets, revealed the efficiency of the proposed approach in filtering out noisy source data samples. They obtained improvement in the prediction performance on most of the datasets in terms of G-measure over NN-filter [7] and Kawata-filter [9] taking Naïve Bayes classifier as the learner, but the results were not good in terms of F-measure. Further, the selection of Naïve Bayes as the learner shows the author's bias as Kawata-filter [9] approach gave the best result with logistic regression and KNN, which must be taken into account while making a comparison with the proposed approach.

Using the same PROMISE datasets [4], Herbold [11] also proposed two different training dataset selection methods. The first method employed the EM clustering algorithm to create clusters in a source dataset merged with the target dataset. Clusters containing the target data samples constituted the final training data (excluding the target data samples from the selected cluster). The second method employed the use of the K nearest neighbor algorithm to find k nearest source projects for a particular target project. For both the above methods, they represented each dataset by its characteristic vector consisting of mean and the standard deviation as the two metrics to represent the marginal distribution of a dataset. He carried his empirical evaluation on 44 open-source datasets and observed an improvement of 9% in success rate (with SVM + RBF kernel) over traditional CPDP which utilizes all the available cross-company data for training without any treatment. But still, the success rate of the proposed CPDP approach was far less than the success rate of WPDP. However, from the two methods proposed, K nearest neighbor performed the best with the neighborhood size equal to 50% to 70% of the available candidate data. They suggested that taking a neighborhood size smaller than 50% would cause the model to underperform and taking a neighborhood size above 70% would not improve the model's performance further, rather will increase the model's complexity.

With a thought to perform the comparative analyses of various relevancy based filter approaches, to arrive at a conclusion about which one is the best, Bin et al. [12] replicated nine relevancy based filters namely global filter [13] (use all source datasets as training data, without any filtering), local filter [14], neighbor filter [13], NN filter [7], Peter filter [8], Kawatta filter [9], He filter [15], HeBurak filter [15], HePeter filter [15] on 33 datasets from PROMISE repository [4] and compared their performance using AUC and effort aware

measures. Based on their results, they concluded that the higher the instance retaining ratio, the better is the prediction performance, thus the global filter is always at the first level among all and there is no need to filter out the training data.

However, the findings are in contradiction to the study [13], which concluded the superiority of the neighbor filter over the global filter. The reason for this discrepancy could be the use of different clustering and classification techniques in both the studies (WHICH & WHERE in [13], and MCLUST & RF in [12]). Furthermore, the evaluation parameters were also different between the two studies. As per our observation, we believe that filter-based technique such as NN filter [7] is relatively better than global filter [13] particularly in terms of PF with acceptable PD, as it is externally validated in many other studies [16–18] also. Taking all cross-company data could lead to high PD, but at the same time also increases PF value due to the presence of negative instances [18].

Different from the above studies where training data selection was based on some filtering technique, Hosseini et al. [19] explored the application of a search-based technique for training data selection for effective CPDP. They came out with a novel approach named genetic instance selection (GIS), where, the genetic algorithm was used to identify the optimal training datasets, whose fitness was measured in terms of F-measure and G-mean on validation set produced by applying NN filter [7] method. They carried their experimentation on 13 datasets from the PROMISE repository [4]. Their result supported the superiority of their proposed approach GIS over traditional CPDP, NN filter [7], and WPDP in terms of F-measure and G-mean in a statistical sense.

Different from the above approaches, where training data selection was solely based on the similarity between the source and the target data, He et al. [20] proposed an improved strategy for training data selection based on two factors: similarity between the source and the target data and the defect count. They explored three methods namely Euclidean Distance, Cosine Similarity, and Manhattan Distance to measure the similarity along with 5 different methods namely Linear, Logistic, Log, Square-root, and Inverse Cotangent to normalize defect count, making a total of 15 different combinations. The combination of Euclidean distance as the similarity method and linear normalization for defect count worked the best. In comparison with the two baseline approaches peter filter [8] and TCA + [21], they observed an improvement in average AUC value on 14 datasets from the PROMISE repository [4] and AEEEM [22], with a medium and small effect size respectively.

So far we have witnessed some significant training data selection based CPDP approaches and analyzed that the performance of CPDP is still under question. Despite filtering out the relevant data, the performance is not up to the mark and didn't meet the benchmark postulated by Zimmermann et al. [2]. One of the reasons could be the absence of feature selection, since all the above approaches applied filters to select suitable samples from the given dataset, assuming all features are equally important and correlated with faults. There may be some features, which are redundant or not related to faults at all, leading to the poor performance of a CPDP model. This seems out to be a clear gap in the existing CPDP literature and must be focused upon.

Furthermore, the above techniques, specifically [7, 11] focused more on Recall than Precision, but if seen in the light of practical aspect, Precision carries more weight than Recall as high PF can frustrate the software quality personnel, leading to a significant drop in their confidence over the model's ability. Therefore, to bring this into practice, there is a need to focus more on improving Precision than Recall.

2.1.3 Transfer learning based approaches

Transfer learning involves applying the gained knowledge from a particular project to make inferences for another project from a related or unrelated domain. With a vision of using the concept of transfer learning as another way to reduce the data divergence in cross-company data, Ma et al. [17] tried to exploit a transfer learning method based on the Naïve Bayes algorithm called Transfer Naïve Bayes (TNB), to build a faster and highly efficient cross-company prediction model. They used the data gravitation formula to estimate the similarity between the samples of the source and the target datasets. They assigned a weight to each sample based on the similarity with the target sample and then finally combined this information with the Bayes logic to build their proposed approach. They used seven datasets from NASA [3] as source data to train the model and three SOFTLAB datasets as target data to test the model. They compared their proposed approach with NN filter [7] and a baseline model (CC) built using all cross-company data using Recall, PF, AUC, and F-measure as the evaluation parameters. Results showed higher AUC, F-measure, and lower PF values than NN-filter and CC methods significantly on all the target datasets. It also proved to be computationally less expensive than NN- filter method and comparable with the CC method, but they did not compare it with WPDP.

Thus, to make CPDP comparable to WPDP, Nam et al. [21] explored the use of transfer learning technique TCA (Transfer Component Analysis [23]) for CPDP. TCA maps the source and target datasets into a latent space, having the least distance in the distribution of the two, without any change in its geometrical structure and data variance. In addition to applying TCA, they also inspected the use of normalization before applying TCA (as normalization affects TCA's performance), which they called it as TCA + .

They proposed four different types of normalization and an automatic selection strategy to choose from the same, based on the similarity between the source and the target datasets. RELINK [24] and AEEEM [22] datasets were used for the experimental work and the result was measured in terms of F-measure. They claimed an average F-measure score of 0.61 and 0.41 on RELINK [24] and AEEEM [22] datasets using TCA+, which were far better than the result obtained (0.49 and 0.32 for RELINK [24] and AEEEM [22] respectively) using traditional CPDP approach. The results are also comparable to within project prediction performance. WPDP gave an average F-measure value of 0.53 and 0.42 on RELINK [24] and AEEEM [22] datasets respectively. However, the results may not stand true for other datasets.

Although TCA+proved its capability for CPDP, but it is not stable. Its prediction performance largely varies with different source projects. To address this shortcoming of TCA+, Liu et al. [25] proposed a two-phase transfer learning model (TPTL) for CPDP. A software project estimator (SPE) was established during the first phase to select suitable source projects from the candidate set using the two regression models. In the second phase, they exploited TCA+to construct two prediction models using the two projects selected in the previous phase. The final prediction is made, consolidating the individual prediction results of the above two models. They used 42 datasets from 42 releases of 14 different open source java projects from the PROMISE repository [4] and the results were computed in terms of F1 score and PofB20. In comparison with the other state-of-the-art CPDP models including TCA+[21], TDS [11], LT [6], and Dycom [26], the proposed TPTL model observed an average improvement of 4.98%, 36.12%, 27.13%, and 11.08% in terms of F1 score respectively overall 42 datasets and observed an average improvement of 91.60%, 71.01%, 11.27%, and 65.73% respectively in terms of PofB20 overall 42 datasets. Thus, TPTL not only resolves the issue of TCA+'s instability but also reduces the developer's effort to find the defective classes. Besides these improvements, the proposed model suffers from a limitation. Its model building time is relatively higher than the compared approaches which makes it infeasible particularly when the size of the source and target datasets are large.

Motivated by Liu et al.'s work [25], Wen et al. [27] also proposed an alternative to resolve the instability issue of TCA+, wherein they combined source selection along with TCA+. They proposed four different source selection techniques namely mean_log, median_log, std_log, and median_zscore, which also encompasses feature selection into it. They experimented with six different types of feature selection techniques namely ReliefF, GainRatio, Correlation, OneR, InfoGain, and Symmetrical Uncertainty. Their empirical study on 42 versions of 14 open source projects demonstrated that the median_zscore based TCA+technique with Relief Factor as the feature selection outperformed the TCA+[21], TDS [11], LT [6], and Dycom [26] in terms of F-measure. It also outperformed these approaches in terms of accuracy, when the number of features is greater.

Both studies proposed their approaches to resolve the instability issue of TCA+and shown improvement in terms of F-measure against TCA+[21], TDS [11], LT [6], and Dycom [26]. However, Liu et al.'s [25] approach outperformed Wen et al.'s [27] approach in terms of F-measure with an average score of 0.481 as compared to an average score of 0.457 for Wen et al.'s approach [27].

Observation So far we have seen that F-measure has been used majorly as the performance indicator for CPDP, with almost no attention to high false alarm. A model with a high false alarm could hamper the developer's confidence in the system and as a result, they might show their reluctance in using it. So keeping practical applicability in mind, Precision and PF should also be considered for the model's evaluation. Negative samples in the source data are the main reason for high PF value.

Targeting this issue in mind, Chen et al. [18] proposed a two levels based approach named Double Transfer Boosting (DTB) to improve the performance of CPDP, by reducing the negative samples in the cross-company data. The first level implemented the data gravitation method [28] to reshape the entire CC data to fit into WC data. The second Level utilized a limited amount of WC data and applied TrAdaBoost [29] and traditional Adaboost to CC data and WC data respectively. It then recalculated the weights for each instance based on the training error rate, with more weight being assigned to misclassified instances to improve the classifier performance. Compared with the NN filter [7], NN+WC, and NB models, the DTB model observed much lesser false alarms and produced statistically significantly better balanced performance in terms of G-measure and MCC with at least medium effect. On comparing with TNB [17] model, the proposed model obtained significantly better results on most experimental datasets, achieving a better overall G-measure and comparable MCC. Their proposed model also performed significantly better than WPDP, trained on a limited amount of WC data, and was comparable to WPDP with abundant samples. However, the major shortcoming of this approach is that it requires the availability of some amount of labeled target data and hence cannot be applied in a situation when labeled target data is not available.

Unlike other cross-project prediction studies, where researchers basically focused on either applying different data filters or using different transfer learning methods to reduce the marginal data distribution difference between the source and the target project, altogether ignoring the conditional distribution difference, Xu et al. [30] proposed a new transfer learning approach called BDA (Balanced Domain Adaptation). It utilized a balanced mix of two

different distributions (marginal and conditional) to bridge the gap between the source and the target project. Marginal distribution refers to the distribution of independent variables i.e. the features, whereas conditional distribution refers to the distribution of dependent variable i.e. the label, given the independent variables. They used maximum mean discrepancy (MMD) [23], a nonparametric measure to calculate the distribution difference between the source and the target data. It takes both the marginal distribution difference and the conditional distribution difference into account. Marginal distribution difference shows the deviation in the means of source and target data in Reproducing Kernel Hilbert Space, whereas conditional distribution difference is calculated summing the deviation in the means of the source and the target data in Reproducing Kernel Hilbert Space, over each distinct class label.

When calculating the difference between the two domain's data distribution, both marginal and conditional distribution have their own importance. When two projects are unlike, marginal distribution carries more weight as compared to conditional distribution, however, in the reverse case conditional distribution dominates over marginal distribution [31]. Therefore, the degree of importance for the two distributions depends on the source and the target project pair. Their proposed approach iteratively adjusts the weight of the two distributions to effectively reduce the distribution difference between the source and the target project pair. They experimented with four defect datasets namely AEEEM [22], RELINK [24], SOFTLAB, and NASA [3], consisting of 18 project data altogether. They evaluated the performance of their proposed approach using six measures namely F-measure, G-mean, Balance, AUC, Effort aware Recall, Effort aware F-measure against six filter-based approaches, and six transfer learning based approaches. They observed an average improvement of 23.8%, 12.5%, 11.5%, 4.7%, 34.2%, and 33.7% over four datasets in terms of F-measure, G-mean, Balance, AUC, Effort aware Recall, Effort aware F-measure respectively.

Observation: So far we have seen various transfer learning based CPDP approaches, majorly focusing on improving the CPDP prediction performance. However, one significant cause for poor CPDP performance i.e. the class imbalance, has not been addressed in any of the above techniques except study [18]. Class imbalance is one of the major problems in machine learning tasks. Whenever we have an uneven distribution of defective and non-defective samples in the dataset, the machine learning model generally, dominates the majority class and will give the wrong output for the minority class. In the context of software defect prediction, minority class (defective class) is our main concern, therefore it really becomes necessary to handle this class imbalance problem before the construction of a CPDP model.

Addressing this issue, Ryu et al. [32] proposed a transfer cost-sensitive boosting (TCSBoost) approach (which is essentially an amalgamation of knowledge transfer and class imbalance learning), to investigate its effectiveness over transfer learning and class imbalance learning alone in a CPDP setting where some labeled target data is available. They applied SMOTE (Synthetic Minority Oversampling technique) and Tomek Link under sampling technique to handle class imbalance problem and training instances were assigned weight based on the concept of the data gravitation method [28] and class imbalance. Based on their empirical evaluation of 15 datasets from the PROMISE repository, their proposed approach TCSBoost outperformed Naïve Bayes, Naïve Bayes + SMOTE, AdaCost [33], Transfer Boost [34], and NN filter [7] in terms of Recall, G-mean, and Balance. However, the probability of false alarm was challenging.

As pointed earlier also that researchers have given more importance to Recall over Precision, which actually can lead to high PF value and could hamper the developer's confidence in the ability of the CPDP model. Their proposed approach also worked on the same ground and achieved a better result in terms of Recall but at the cost of high PF. Therefore, there seems a strong need to investigate the importance of Precision over Recall to see its impact on the developer's effort.

Like Chen et al.'s work [18], this work also suffers from a limitation that their proposed approach cannot be applied in the case of the unavailability of some labeled target data.

Addressing the same issue of class imbalance, Tong et al. [35] proposed an approach, wherein a new minority oversampling technique based on transfer learning (TOMO) was introduced to handle class imbalance, followed by a modified transfer Naïve Bayes approach. In their proposed approach, samples were assigned weights based on feature similarity. Based on their empirical work on 11 public datasets, they demonstrated that their proposed approach observed an average improvement of at least 27.8% and 71.5% in terms of G-measure and MCC in comparison to state-of-the-art CPDP approaches.

All the above-mentioned approaches assumed that all the available source data are well labeled. However, in a situation where only a limited amount of labeled source data is available due to expensive human labeling, semi-supervised techniques come into play.

To this end, Wu et al. [36] attempted to provide an effective and consolidated solution to both within project semi-supervised fault prediction (WPSFP) and cross-project semi-supervised fault prediction (CPSFP) problems. They came out with a cost-sensitive kernelized semi-supervised dictionary learning (CKSDL) method for the same. They incorporated two changes in the semi-supervised dictionary learning method [37], firstly by doing kernel space mapping and

secondly by introducing a penalty to account for misclassification cost. Sixteen projects from 4 public defect datasets including NASA MDP [3], AEEEM [22] RELINK [24], and MORPH have been experimented upon and the performance of several state-of-the-art competing models was measured in terms of Precision, Recall, F-measure, and AUC. Based on the obtained average prediction results of five semi-supervised defect prediction approaches FTcF-MDS [38, 39], LDS [40], RusTri [41], ASDP [42], and NSGLP [43], a CPDP approach SCC [44], and a dictionary-learning-based fault prediction approach CDDL [45] on four datasets, the proposed approach outperformed on all four performance parameters, both in WPSFP and CPSFP scenarios.

Thus, we have seen a plethora of diverse transfer learning CPDP approaches and different results have been obtained from different studies since the evaluation parameters and the datasets were not the same. It is really difficult to obtain a true consensus about who wins over whom. Therefore, there is a need to make a comparative analysis among different transfer learning-based approaches by replicating them using the common datasets and common evaluation measures to know the true state of CPDP.

2.1.4 Ensemble approaches for CPDP

Ensemble approaches usually involve constructing several weak learners/classifiers (a classifier which is far away from the true classification and performs poorly irrespective of training data distribution) and making decisions by combining the decisions of all weak classifiers, intending to bring improvement in the performance compared to an individual learner.

To this end, a very unique approach was proposed by Panichella et al. [46] called CODEP (combined defect predictor), where nothing had been done regarding training data selection and reducing the data distribution difference. They applied six different classifiers namely Logistic Regression, Radial Basis Function Network, Bayes Network, Multilayer Perceptron, Decision Table, and Alternating Decision Tree in the first phase, and the result of each of them was supplied as input to another classifier (logistic regression and Bayes Network taken individually) in sequence. Since individual classifiers used in the first phase captured different defect probabilities for the same target data and identified a different set of samples as defect prone, so to get a better prediction performance, they combined these complementary classifiers in the second phase. The combined approach outperformed the standalone classifiers not only in terms of AUC but also in terms of cost-effectiveness. CODEP with Bayes network observed an average improvement of 10% to 41% and CODEP with logistic regression observed an average improvement of 6% to 37% in terms of average AUC value.

As we have already seen in Sects. 2.1.1, 2.1.2, and 2.1.3, that the two things are very important for CPDP task, firstly, how to curtail the data distribution difference across the cross-company data and secondly, which source projects should be selected from an available candidate set for training. Some studies [7–10] have selected all source datasets for training, giving equal weights to all, leading to high PF. It is therefore important to analyze the auxiliary power of different source projects for defect prediction to boost the effect of samples that resembles the target project most and at the same time to negate the effect of samples that resembles the least.

Motivated by this fact, Xia et al. [47] came out with a hybrid model reconstruction approach called HYDRA for CPDP, which consisted of two phases. The first phase led to the construction of a total of n + 1 base classifiers, each trained on one of the n source data with one classifier trained only on limited available labeled target data. It then followed by building a GA classifier, giving the best weights to multiple classifiers that maximize the fitness score (F1-Score) on the training target data. The second phase iterated the first phase k times leading to the construction of k GA classifiers, followed by assigning weights to each of these GA classifiers based on their training error rate on training target data, finally leading to the construction of a composite classifier at the end. They carried their empirical evaluation on 29 projects from the PROMISE repository [4] and compared their approach with 7 state-of-the-art approaches [2, 8, 21, 34, 46, 48, 49] and observed an average improvement of 40.21%, 26.22%, 34.99%, 47.43%, 28.61%, 30.14% and 39.49% in terms of F1 score respectively. It also outperformed WPDP (with 5 percent labeled data) in terms of PofB20 and F1-score by 62.40% and 19.46% respectively.

Similar to Xia et al.'s [47] work, Qiu et al. [50] proposed a novel multiple components weights (KMM-MCW) learning model with the same idea in mind that different software components have different auxiliary power for fault detection. They clustered the source project using spectral clustering and used the Kernel Mean Matching (KMM) algorithm to curtail the data distribution difference between the source components and the target project. Finally, they build a more precise ensemble classifier for the target project. On comparison with the seven state-of-the-art techniques namely DTB [18], TrAdaboost [29],TNB [17], TCA + [21], NN-Filter [7], KMM and Baseline, an average improvement of 11.2%, 11.5%, 4.1%, 35.8%, 25.5%, 10.7% and 26.17% was observed in accuracy overall 15 datasets from PROMISE repository [4] and an average improvement of 4.4%, 3.2%, 7.1%, 17.8%, 14.8%, 6.9% and 13.9% was observed in F-measure over all datasets.

To further improve the CPDP performance taking F-measure as the indicator, Chen et al. [51] proposed a new transfer learning framework called collective

transfer learning defect prediction (CTDP). It consisted of two phases, wherein, first phase leverage TCA + to expand the source project dataset using all four normalization methods described in the study [21], followed by building a base classifier for each of the expanded source datasets. The second phase utilizes a particle swarm optimization algorithm to assign adaptive weights to each of these base classifiers to make an ensemble classifier. They carried their empirical study on 28 open source projects from five different datasets and observed an improvement of 20%, 100%, 16.28%, 35.14%, and 47.06% over WPDP, baseline CPDP, TCA + [21], HYDRA [47], and CODEP [46] respectively in terms of F-measure.

So far we have seen a plethora of diverse CPDP studies, but due to different experimental setups, involving different datasets and different evaluation parameters, it is difficult to analyze which approach comprehensively performed the best.

To target this issue, Herbold et al. [16] investigated with 24 existing CPDP approaches using 85 software products from five datasets namely [52], NASA MDP [3], AEEEM [22], RELINK [24] and NETGENE [53] using four evaluation measures AUC, F-measure, G-measure and MCC together. They reported Cruz [6] at the top in the ranking list, followed by Turhan et al.'s approach [7], Menzies et al.'s approach [13], and Watanabe et al.'s approach [5]. They suggested these approaches as the benchmark for CPDP under a strict CPDP setting. Since only non-effort based evaluation measures AUC, F-measure, G-measure, and MCC have been used in the study [16] for comparing various CPDP approaches, therefore to obtain a holistic view of the current state of the CPDP, Herbold [54] revisited the same study [16] considering effort and cost metrics into consideration this time. They implemented 26 CPDP approaches and 3 baselines on two datasets namely [52] and AEEEM [22]. They observed the trivial baseline approach called FIX (which predicts everything as defective) on the top of the ranking list. The two CPDP approaches which performed the best in terms of effort and cost measures out of all 26 CPDP approaches were the approaches proposed by Liu et al. [48] and Canfora et al. [49]. However, no correlation was observed between the rankings obtained by the study [16] and study [54].

This insight uncovers the fact that an approach which seems good in terms of technical (or standard) machine learning non-effort based measures, may not stand equally good in terms of effort/cost-based measures. Therefore, there is an urgent need of bringing effort-based measures into consideration in addition to traditional non effort based measures to evaluate the CPDP model's effectiveness holistically.

On the same datasets, Zhou et al. [55] also carried a comparative study to inspect the true state of CPDP but arrived at a different conclusion. They proposed two approaches called Manual Down and Manual Up and compared them with the existing CPDP approaches using a large number of diverse evaluation parameters.

Both Manual Up and Manual Down do not require any training data and are simply based on just one feature i.e. the module size in terms of LOC. Manual Down is based on the assumption that the larger the module size, more will be the chances for fault proneness, however, Manual Up is based on the rationale that smaller modules are more fault-prone. Manual Up simply marks the top 50% of the modules sorted in ascending order of LOC as defective, and the rest 50% of the modules as non-defective, whereas Manual Down marks the top 50% of the modules sorted in descending order of LOC as faulty and rest 50% of the modules as non-faulty.

Their empirical investigation concluded the superiority of their simple module size based approaches over existing CPDP approaches. They raised a point saying that, if such a simple module size approach can outperform the existing state-of-the-art CPDP approaches, then the current state of the CPDP is not up to par and thus demands further improvement in its performance both technically and economically. Furthermore, they proposed Manual Down as the baseline, if measuring the performance using non-effort-based measures, and Manual UP as the baseline, if measuring the performance using effort-based measures. However, one limitation of their work is that they have not replicated the existing approaches and have taken the results as it is from the respective research papers. But it is always better to replicate the work with which to make the comparison, to ensure its correctness.

2.2 Heterogeneous CPDP approaches

Till now, we have seen homogeneous CPDP, where the source and the target datasets, both have the same metric set and consequently can be used directly without any feature matching (or feature selection). However, feature selection is an important step in any machine learning task since not all given features are the true representative of the dependent variable. There may be some redundant or irrelevant features, which if not removed, can drop the model's performance significantly. Therefore, it is always advisable to select all relevant features before the model construction. However, when the source and the target datasets have heterogeneous metric sets, there arises a question that "will the CPDP still be feasible?" The answer is, "Yes". But, a significant amount of feature's pre-processing work is required. Many researchers are attempting to validate the effectiveness of heterogeneous CPDP. Few studies [7, 17] have investigated heterogeneous CPDP by selecting the common metrics from the source and the target datasets. However, the results were not comparable to WPDP. Jing et al. [56] proposed unified metric

representation (UMR) for both the training and the testing data, which consisted of three types of metrics sets namely common metric set between the source and the target data, followed by metric set particular to the source project and then the metric set particular to the target project. While constructing the UMR of the source data, target specific metrics are set to zero whereas source-specific metrics are set to zero for constructing the UMR of the target dataset. Canonical correlation analysis (CCA) [57] transfer learning method was then applied on UMR of source and the target data to reduce the data distribution difference between them. Their experimental work on 14 datasets out shown the efficacy of their approach in comparison to state-of-the-art CPDP methods which is also comparable to WPDP.

Different from the above approach, where no feature selection was done, Nam et al. [58] proposed a heterogeneous CPDP approach wherein feature selection was done first, followed by a metrics matching mechanism, where the metrics from the target dataset having a similar distribution with the source metrics were identified. Three different metrics matching analyzers namely percentile-based analyzer, Spearman's correlation-based analyzer, and Kolmogorov–Smirnov test-based analyzer were proposed. Their empirical investigation on 28 datasets demonstrated the promising result of their proposed approach (with Kolmogorov–Smirnov test-based analyzer at threshold cut-off of 0.05 along with Chi square feature selection) in terms of AUC in comparison to WPDP and state-of-the-art heterogeneous CPDP approaches. However, their proposed approach assumed only the linear correlation between the source and the target metric sets, altogether ignoring the possibility of a non-linear correlation between them.

To overcome this linearly inseparable problem of heterogeneous CPDP, together with the class imbalance problem, Li et al. [59] proposed an approach, wherein first the source and the target data are mapped into high dimensional kernel space, so that the defective and non-defective data could be easily separable. They designed a kernel correlation alignment method with nine Gaussian based kernel functions and a linear kernel function, followed by integrating multiple kernel classifiers with ensemble based learning to mitigate the effect of class imbalance. An extensive empirical investigation on 30 public datasets from five different groups manifested the efficacy of their proposed approach over WPDP, state-of-the-art homogeneous CPDP and heterogeneous CPDP approaches in terms of AUC.

Further ahead, Li et al. [60] presented another approach, which used the transfer kernel canonical correlation analysis method to reduce the data distribution difference in non-linear feature space and also incorporated cost-sensitive learning to counteract class imbalance. Their detailed experimental work on 28 datasets from five different groups revealed the promising results of their approach over state-of-the-art homogeneous CPDP and heterogeneous CPDP approaches in terms of Recall, PF, F-measure, AUC, and G-mean.

Thus, it is now evident from the existing literature that the cross-project defect prediction has the potential to be used in a situation where no historical data is available as demonstrated by many of the CPDP studies that the results are comparable to within project defect prediction and can be utilized when no or limited amount of labeled target data is available. Existing CPDP approaches showed promising results in terms of non- effort based performance measures like Recall, Precision, F-measure, PF, G-measure, G-mean, Accuracy, Balance, MCC, and AUC, but none of them was able to comply with the benchmark postulated by Zimmermann et al. [2]. All existing approaches failed to achieve 0.75 Recall, Precision, and accuracy simultaneously for all datasets under test [16]. Apart from this, very few of them [12, 19, 30, 46, 47, 54, 55] have investigated their model performance in terms of cost/effort measures like Effort aware Recall, Effort aware Precision, Effort aware F-measure, PofB20, IFA(Initial False Alarm), etc. Before putting CPDP into practice, we need to have a cost and benefit analysis, therefore, it really becomes necessary to confirm the model's validity in terms of business perspective, which is missing in the above work. However, Herbold [54] attempted to throw light on the same issue and also provide the significance of effort-based measures in a practical sense.

3 Discussion

A comprehensive picture of cross-project defect prediction has been presented in Sect. 2 and now, we summarize the literature concerning every research question targeted in the beginning.

RQ1: What kind of datasets have been used the most in CPDP?

Answer: Both the open-source and the closed source datasets have been explored in CPDP, but most studies have selected open-source datasets for their experimental work. Table 4 lists the kind of datasets used in each of the primary studies. Only 6% of the selected studies have worked

Table 4 Open/closed dataset usage in CPDP

Type of datasets used	Primary studies
Open source datasets	[5, 6, 13, 15, 20, 21, 25, 47, 61]
Closed source/academic/proprietary datasets	[7, 17]
Mixed datasets	[2, 8–12, 16, 18, 19, 27, 30, 32, 35, 36, 46, 50, 51, 54–56, 58–60]

on closed source datasets, 26% of the primary studies have used only open-source dataset, whereas 68% of the primary studies chose mixed datasets to work upon. However, the studies that used mixed datasets, majorly consist of open-source datasets with approximately 2 to 3 closed/academic/proprietary datasets. Mostly PROMISE defect datasets [4] have been used the most across all studies. The researcher's bias in favoring/selecting the open-source datasets seems obvious since defect data collection involves a huge investment in terms of money and time. But seeing its practical applicability in a real scenario, it demands more tests/investigations specifically on commercial/private datasets.

RQ2: What kind of modeling techniques have been used the most in CPDP?

Answer: Table 5 presents the model(s) used in each of the selected CPDP studies. We considered the model(s) which gave the best result in a particular study. However, for a comparative study, all models used are considered. Logistic Regression is the most used modeling technique followed by Naïve Bayes due to their best performance in most of the selected CPDP studies. However, other models have not been used much. The reason for their underperformance could be many folds, such as some models like NN are quite sensitive to hyper parameter tuning, whereas other models such as SVM are sensitive to features distribution. A decision tree could lead to model overfitting, whereas KNN could be time-consuming.

Although LR and NB worked well on most of the datasets used in these studies, but still, we cannot generalize the above statement as the model's performance is associated with many other factors also such as the kinds of software metrics used, the type of performance indicators used, the quality of data used and the hyperparameters setting applied.

Table 5 Models used in CPDP

Models used	Primary studies
NB	[7, 10, 15–19, 32, 35, 54]
LR	[2, 6, 9, 16, 20, 21, 25, 27, 30, 46, 47, 50, 54, 58–60]
RF	[8, 12, 16, 54]
SVM	[11, 16, 54]
NN	[56]
Decision tree	[5, 16, 51, 54]
Decision table	[61]
KNN	[9]
Bayes network	[46]
RBF Net	[16, 54]
WHICH	[13, 54]

Nonetheless, LR and NB are the two most dominant models in the CPDP setting.

RQ3: What kind of metrics have been used the most in CPDP?

Answer: Both product and process metrics have been used in cross-project defect prediction. But product metrics (mainly object-oriented metrics, LOC and Complexity metrics) have been used most dominantly as it is explored in all the selected CPDP studies, whereas process metrics have been used in only 38% of the selected studies along with product metrics. Since existing studies [62–65] have already witnessed the performance of process metrics in WPDP as successful fault predictors, so it can also be applied well in cross-project defect prediction. There is a strong need for more investigation on the application of process metrics in CPDP.

RQ4: What kind of Evaluation Parameters have been used the most in CPDP?

Answer: A diverse range of evaluation measures (Precision, Recall, F-measure, G-measure, G-mean, PF, AUC, MCC, Balance, PofB20, accuracy, effort aware measures) have been used across the selected CPDP studies and a great deal of inconsistency has been observed among all, in terms of their usage. However, F1 has been observed as the most used evaluation parameter, followed by the AUC score, as can be seen in Fig. 5. But, the selection of some evaluation measures, most importantly the use of accuracy, seems a little biased/not appropriate in some studies [27, 50]. In the context of software fault prediction, accuracy may not be the right choice to capture the model performance. For example, consider a dataset comprising of 95% of non-defective samples and 5% of defective samples. A high value of accuracy such as 95%, will fail to judge the model's capability if all 5% of defective samples will be misclassified. Since our objective is to identify defective samples, a model having 95% accuracy is just a total wastage of efforts and cost as it failed to detect any of the defective samples. Therefore, accuracy is not a good indicator to capture the performance of a fault prediction model. Instead, the selection of performance indicators should be based on the objective taken in hand and not on the popularity. For example, the studies [12, 19, 30, 46, 47, 54, 55] selected effort-based performance indicators along with non-effort based measures to capture their proposed model's performance which were perfectly aligned with the objective taken in their proposed work. For instance, Herbold et al. [16] carried an empirical investigation to inspect the true state of the CPDP by replicating and evaluating 24 existing CPDP approaches using non-effort based measures only. Further in the study [54], he replicated the same experiment to know who wins over whom, this time in terms of effort-based measures. The selection of evaluation parameters in both studies was completely aligned with the objective undertaken. Apart from this, one

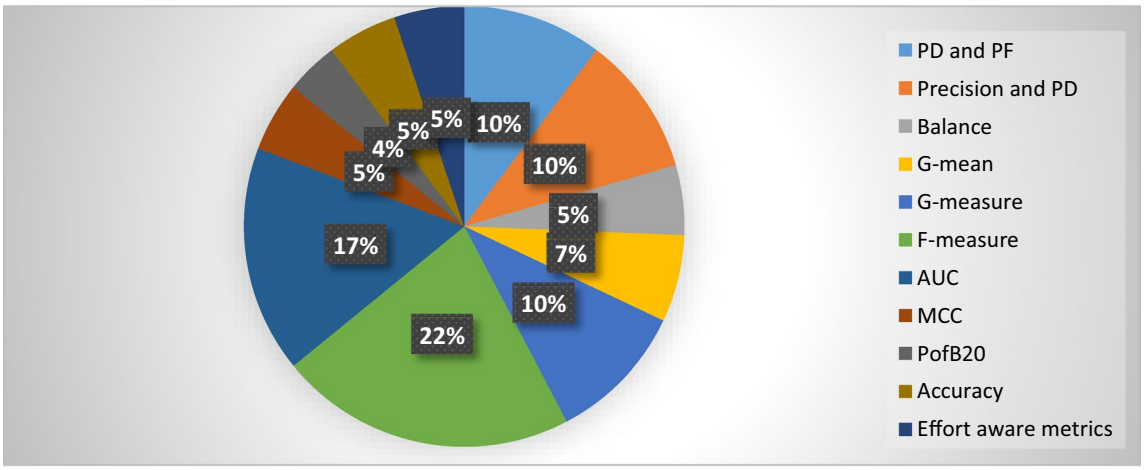

Fig. 5 Proportion of evaluation metrics used in selected CPDP studies

major inference that we dug out of this is that most studies have given more weight to Recall than Precision and as a consequence has got a high value of PF. But if we look from a practical aspect, having a high value of false alarm could shake the developer's belief in the capability/utility of the model and as a result, they might switch back to their conventional (manual inspection) way of doing software reviews.

RQ5: What kind of Statistical Test have been used the most in CPDP?

Answer: Different studies opted for a different statistical test to validate their model in comparison to other competitive approaches. Figure 6 showcases the diverse set of statistical tests used in all selected studies. It can be clearly seen that the Wilcoxon Signed Rank test has been used mostly, along with Cliff delta effect size (Cliff delta and Cohen's d are applied post a statistical test, just to measure the magnitude by which a particular technique outperforms the other), followed by Wilcoxon Rank Sum test (Mann–Whitney U test).

Wilcoxon Signed-Rank test is a non-parametric test that is generally used to compare two dependent (related matched pair) samples to check whether they both have come from the same population or not. On the contrary, the Wilcoxon Rank Sum test is used to compare two independent samples. Friedman's test is also a non-parametric test, applied when the performance of several different techniques need to be compared across multiple test attempts. The suitability of a

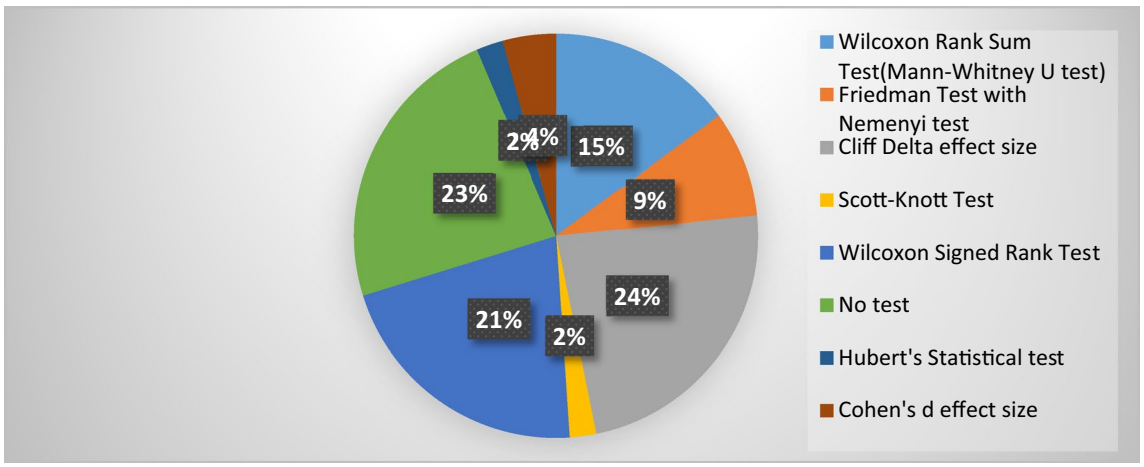

Fig. 6 Proportion of different statistical test used in selected studies

particular statistical test in a particular study is of great concern. Different statistical tests are based on different assumptions. Their inappropriate use could mislead the results.

4 SWOT analysis of CPDP

Now, we present SWOT analysis of cross-project defect prediction, highlighting its strengths/power, weaknesses/vulnerabilities, opportunities/favorable chances, and threats/risks to give a deep insight into this subarea, thus enabling academicians, researchers, and practitioners to enrich and enhance their knowledge to build a more proficient and productive approach.

4.1 Strengths

Several factors contribute to the success of cross-project defect prediction. Few significant contributors are as follows:

- Support from the corpus of cross-company data

Many companies such as NASA and SOFTLAB have made their defect datasets available for public consumption. Apart from this, many researchers also have contributed their defect datasets (Jureczko and Madeyski [52], AEEEM [22], and RELINK [24]) in public repositories. As mentioned before, what we need, is the availability of cross-company data for training. If selected carefully, can achieve result comparable to within project defect prediction. Many studies [18, 21, 51] have reported that the result obtained, are comparable to within project defect prediction. And, thus supports the validity of using a CPDP model in the absence of availability of costly defect data, as a significant step for increasing the developer's productivity by improving the software quality.

- Cost reduction in data collection

Collecting defect data for each software module and analyzing the corresponding class label, is a time consuming and costly activity. Several studies [18, 32, 47, 50] demonstrated that by utilizing a very few amount of within project data (10% to 20%), together with cross-company data, an efficient CPDP model can be constructed. This finding eliminated the unnecessary/worthless need of collecting all defect data and harmonize the balance between the cost incurred vs. benefits realized.

4.2 Weaknesses

Several vulnerable factors responsible for bringing the CPDP's progress graph down are as follows:

- Poor selection of cross company data

The initial attempt [2] to check the effectiveness of a CPDP model was a big failure and the main reason was no selection of cross-company data. Furthermore, the target and the cross-company data, both had different data distribution. According to study [7, 32], when data is used directly without any instance filtering, could lead to low Precision and a high PF value. On the other hand, when used carefully, could increase the Precision with a decrease in the PF value as reported in the study [18]. Thus, the success of the CPDP task lies in the careful selection of training data in a way that the training data distribution should match with the target data distribution. A plethora of different approaches as discussed above, have been proposed by different researchers, highlighting the importance of source selection. But still, there is a scope of improvement as the victory of CPDP is vulnerable to source selection.

- High computational cost

As compared to WPDP, CPDP experiments incur more computational cost as the data does not straight away fit for the training phase. For example, the study [7] used a KNN filter to select 10 nearest samples from the CC data for every sample in the target project to construct the final training dataset. The bigger the target project, the more will be the cost of computing the final training data. Similarly, the studies [21, 25, 27, 51] which used TCA to reduce the data distribution difference are computationally very inefficient. So, a significant amount of work (for reducing the data distribution difference between the source data and target data) needs to be done in the pre-processing of data before the learning phase.

4.3 Opportunities

Some of the gaps that we have analyzed and which can be turned into possible chances for improvement are as follows:

- Incorporation of feature engineering into CPDP

Existing studies [22, 66–73] have already shown the importance of feature selection/feature extraction in context to WPDP. But only 14.7% of CPDP studies [19, 27, 58, 60, 61] have been reported, addressing this issue in context to CPDP. Seeing the success rate of feature engineering in the

WPDP setting, it can also be explored in the CPDP setting as an important means to improve the prediction performance of a CPDP model.

- Scope for process metrics as successful fault predictors

All the selected CPDP studies have used product metrics as the predictors in building their CPDP model, however, only 38% of the selected studies have also incorporated process metrics and previous fault metric along with the product metrics in their experimental settings. Seeing the excellent performance of process based metrics in WPDP experiments [62–65] one can also explore the validity of different process metrics in the CPDP setting. Some process metrics such as the module's age and code churn, number of developers working on a module, have been used in only 29.4% of CPDP studies [16, 21, 30, 36, 54–56, 58–60], as they are the part of AEEEM dataset [22], however, the impact of some process metrics like number of files modified, number of subsystems modified, number of directories modified, developer's experience has not been investigated into CPDP setting till now. Therefore, further exploration with these unseen process metrics into the CPDP setting is needed.

- Scope for regression task

Majorly, two kinds of the task are being targeted in context to software fault prediction namely classification and regression. Classification refers to the task of classifying/predicting the software modules as faulty or non-faulty based on some of its measured traits/properties (software metrics), whereas regression refers to the task of estimating/predicting the fault count in each software module based on its measured features. Based on the estimated fault count, modules can be prioritized and thus a ranking can be obtained, specifying the order of module's testing. Most of the CPDP experiments which have been carried out recently are catering to the classification task. However, very limited work has been reported for regression. As per our analysis, 97% of the selected studies deal with classification, whereas only 3% of the selected studies cater to regression task**s.** Generally, software developers are available with an effort equal to 20% of total LOC (lines of code). Existing work [74] suggests that 80% of the fault lies in the 20% files, therefore identifying those 20% of files becomes very important. Ranking of modules based on the estimated fault count helps in identifying that 20% of files that can account for 80% of the total faults and thus helps in reducing the developer's effort in identifying the faultiest modules. Since very limited work has been done concerning this, there exists a big scope for performance improvement, particularly in regression models. Thus, this gap can be exploited as an opportunity to come up with some innovative ideas to target particularly regression in CPDP.

- Room for improvement in prediction's performance

Although a plethora of diverse approaches have been proposed for CPDP, but most of them have missed out on taking class imbalance, feature preprocessing, hyperparameter tuning, and effort-based evaluation measures into consideration. Only 20.58% of the selected CPDP studies [12, 19, 30, 46, 47, 54, 55] have validated their proposed models using effort-based measures, which are very important for their practical applicability. Similarly, class imbalance, hyperparameter tuning, and multicollinearity are the three significant issues, which could lead to poor prediction performance, but have not been given due importance in cross-project defect prediction. So this insight brings out a hope/a way to amalgamate different class imbalance and features preprocessing techniques with existing CPDP approaches and their evaluation on effort-based measures, with the positivity of observing improvement in the prediction performance.

4.4 Threats/risks

There are some factors which can lead to a risky situation, where the findings reported, could be misleading. Some of the significant threats are as follows:

- Inconsistency in the use of performance evaluation parameters

Different studies have used different combinations of evaluation parameters to validate their respective approaches. As a result, it becomes cumbersome to compare and analyze their relative performances. Figure 5 is representing the proportion of different evaluation parameters used in the selected studies. The most commonly used metric is F-measure (the harmonic mean of Recall and Precision), followed by AUC, PD & PF, and PD & Precision. The least used metrics are MCC, accuracy, PofB20, and effort aware measures. Different performance indicators lead to a different intuitive interpretation of the underlying model. Actually, the selection of evaluation metrics should depend upon the objective of the study undertaken. If the focus is more on correctly identifying faulty instances, then the metrics like Precision, Recall, PF, F-measure, G-mean, and G-measure may produce a fruitful result. At the same time, one can also choose between Precision and Recall. If both are equally important, F_β with $\beta = 1$ will be the right choice. However, if Recall weighs more than Precision, F_β with $\beta = 2$ can be used. If Precision is more important than Recall, F_β with $\beta = 0.5$ will be the right choice. When the correct classification of both the faulty and non-faulty classes are equally important, then

metrics like accuracy, Matthew's coefficient of correlation (MCC) could be used. AUC is a threshold independent metric and is less affected by class imbalance, therefore can be used to compare the relative performances of various classifiers. When inspecting the performance of a ranking based model, then the metrics like PofB20%, FPA (fault percentile average), AUCEC (area under cost-effectiveness curve) may suffice. On the other hand, when inspecting the practical applicability of the model, effort-based metrics including, PofB20%, FPA, AUCEC, Effort aware Recall, Effort aware Precision, Effort aware F-measure, and IFA comes into play. Thus, the selection of correct evaluation parameters is important for the success of a CPDP model. Erroneous selection of performance measurement metrics could pose danger to the validity of a CPDP model.

- Inconsistency in the use of statistical test techniques

A great deal of disparity has been observed in the usage of statistical testing techniques across different studies, as can be seen in Fig. 6. Selecting an appropriate hypothesis testing technique is important for the model's validation. For example, when applying the Friedman-Nemenyi test to compare several approaches, its ranking based results sometimes fail to discriminate between a good and a bad approach. Therefore, one must carefully look into the weaknesses possessed by a particular test, before using it.

Further, Cliff delta effect size calculation or Cohen's d effect size calculation, which are generally carried post another statistical test to measure the magnitude by which a particular model is better than the other and is always advisable, is attempted by only 28% of the selected studies. Further, to our big surprise, 23% of the selected studies have not used any statistical test, which is not a good practice. One must validate their findings statistically to gain confidence in their respective approach. Several factors decide the applicability of a particular type of tests, such as the size of test data, the type of sample's distribution (Gaussian/Non-Gaussian), sample's dependence/independence, their underlying assumptions, and many more. Therefore, the right selection of a statistical technique is important for model validation.

- Threat to external validity

All the CPDP approaches that we came across, worked on some specific datasets. To the best of our knowledge, there is no effective universal CPDP approach that can fit in all datasets. This is a common threat reported in all CPDP studies considered in the survey.

5 Threats to validity

We have selected 34 primary significant studies in this survey to uncover the major strengths, weaknesses, opportunities, and threats about CPDP. Since our major focus is on homogeneous CPDP, therefore we selected a total of 30 homogeneous CPDP studies and 4 heterogeneous CPDP studies to present the current state of CPDP holistically. Although we have carefully selected all CPDP studies using our well-defined inclusion and exclusion criteria, but it may be possible that some significant studies have been missed out. To the best of our knowledge, we believe that selected studies were sufficient to carry the SWOT analysis of CPDP and to present the current state of CPDP as the majority of our selected studies are from recent times as can be seen in Fig. 1.

6 Conclusion

Cross Project Fault Prediction (CPDP) is gaining much attention from various researchers across the globe. To synthesize and analyze the current state of the art and also to present what milestones need to be covered, we performed a comprehensive survey on CPDP along with its SWOT analyses.

The comprehensive report presented in this work is based on 34 primary different approaches used for CPDP. From these studies, qualitative and quantitative data were extracted and analyzed to present the SWOT analysis as well as to frame answers for the targeted research questions. SWOT analysis of cross-project defect prediction highlighted the grey areas to further improve the CPDP performance. It has been found that none of the existing approaches have been able to comply with the benchmark postulated by Zimmermann et al. [2]. It has been also found that all the existing approaches failed to achieve 0.75 recall, precision, and accuracy simultaneously for all datasets under test. Hence we can conclude that there exists a big scope for performance improvement in the field of CPDP. Incorporation of feature engineering, exploration with process metrics (such as number of files modified, number of subsystems modified, number of directories modified, developer experience, etc.), class imbalance handling, and hyperparameter tuning can be investigated in CPDP setting to bring improvement in its performance. Further ahead, our analysis also reported that LR and NB have been heavily used models in CPDP. The product metrics (object-oriented metrics, complexity metrics, and LOC) were the most used fault predictors in CPDP. Open source datasets have been used extensively in most of the works on CPDP. F-measure and AUC evaluation measures have been found to be widely used in CPDP. Wilcoxon

Signed-Rank test has been observed to be the heavily used statistical test among all in CPDP studies.

Finally, to conclude analytically, we would say that the performance of a CPDP model is linked with the way it is built. The kind of metrics used, the types of modeling techniques applied, the diverse approaches explored, the quality of data used, and the kinds of the statistical test applied, all have a significant impact on the model's capability. Existing approaches are more or less Recall oriented and failed in achieving good Precision and low probability of false alarm. A high false alarm could hamper the developer's confidence in the system. Therefore, there is a need to investigate the importance of precision over recall to see its impact on the developer's productivity. Moreover, 82.35% of the selected CPDP studies focused on traditional machine learning performance evaluation measures, with little or no emphasis on cost/effort aware performance measures. Before putting CPDP into practice, we need to have a cost and benefit analysis, therefore, it really becomes necessary to confirm the model's validity in terms of cost-effectiveness, which was missing in 82.35% of the selected studies. Furthermore, existing work focused more on instance selection with no or very little emphasis on feature selection. There can be some features that are not related to fault and can degrade the performance of a CPDP model. Only 14.7% of the selected CPDP studies targeted feature selection in their work. Therefore, future studies must consider these points and should take into account feature engineering and effort aware performance measures along with non-effort based performance measures as well, to depict the true performance of the proposed model.

We hope that the findings reported in this survey will be fruitful for carrying future research and will assist in turning its weaknesses into possible strengths.

References

1. Briand L, Melo W, Wust J (2002) Assessing the applicability of fault-proneness models across object-oriented software projects. IEEE Trans Softw Eng 28:706–720. https://doi.org/10.1016/j.chemosphere.2013.11.049
2. 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: ESEC-FSE'09—proceedings of the Joint 12th European software engineering conference and 17th ACM SIGSOFT symposium on the foundations of software engineering. pp 91–100
3. Shepperd M, Song Q, Sun Z, Mair C (2013) Data quality: some comments on the NASA software defect datasets. IEEE Trans Softw Eng 39:1208–1215. https://doi.org/10.1109/TSE.2013.11
4. Menzies T, Caglayan B, Kognoli E, Carl J, Peters F, Turhan B (2007) The promise repository of empirical software engineering data
5. Watanabe S, Kaiya H, Kaijiri K (2008) Adapting a fault prediction model to allow inter language reuse. In: Proceedings—international conference on software engineering. pp 19–24
6. Cruz AEC, Ochimizu K (2009) Towards logistic regression models for predicting fault-prone code across software projects. In: 2009 3rd international symposium on empirical software engineering and measurement, ESEM 2009. pp 460–463
7. 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 14:540–578. https://doi.org/10.1007/s10664-008-9103-7
8. Peters F, Menzies T, Marcus A (2013) Better cross company defect prediction. In: IEEE international working conference on mining software repositories. pp 409–418
9. Kawata K, Amasaki S, Yokogawa T (2015) 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. pp 2–7
10. Yu X, Zhang J, Zhou P, Liu J (2017) A data filtering method based on agglomerative clustering. In: Proceedings of the international conference on software engineering and knowledge engineering, SEKE. Knowledge Systems Institute Graduate School, pp 392–397
11. Herbold S (2013) Training data selection for cross-project defect prediction. In: ACM international conference proceeding series. Association for Computing Machinery, pp 1–10
12. Bin Y, Zhou K, Lu H, Zhou Y, Xu B (2017) Training data selection for cross-project defection prediction: which approach is better? In: International symposium on empirical software engineering and measurement. IEEE Computer Society, pp 354–363
13. Menzies T, Butcher A, Cok D, Marcus A, Layman L, Shull F, Turhan B, Zimmermann T (2013) Local versus global lessons for defect prediction and effort estimation. IEEE Trans Softw Eng 39:822–834. https://doi.org/10.1109/TSE.2012.83
14. Bettenburg N, Nagappan M, Hassan AE (2012) Think locally, act globally: improving defect and effort prediction models. In: IEEE international working conference on mining software repositories. pp 60–69
15. He P, Li B, Zhang D, Ma Y (2014) Simplification of training data for cross-project defect prediction. Comput Sci Software Eng 2:17
16. Herbold S, Trautsch A, Grabowski J (2018) A comparative study to benchmark cross-project defect prediction approaches. IEEE Trans Softw Eng 44:811–833. https://doi.org/10.1109/TSE.2017.2724538
17. Ma Y, Luo G, Zeng X, Chen A (2012) Transfer learning for cross-company software defect prediction. Inf Softw Technol 54:248–256. https://doi.org/10.1016/j.infsof.2011.09.007
18. Chen L, Fang B, Shang Z, Tang Y (2015) Negative samples reduction in cross-company software defects prediction. Inf Softw Technol 62:67–77. https://doi.org/10.1016/j.infsof.2015.01.014
19. Hosseini S, Turhan B, Mantyl M (2016) Search based training data selection for cross project defect prediction. In: ACM international conference proceeding series. Association for Computing Machinery, New York, New York, USA, pp 1–10
20. He P, He Y, Yu L, Li B (2018) An improved method for cross-project defect prediction by simplifying training data. Math Probl Eng 2018:1–18. https://doi.org/10.1155/2018/2650415
21. Nam J, Jialin Pan S, Kim S (2013) Transfer defect learning. In: 35th International conference on software engineering (ICSE). pp 382–391
22. D'Ambros M, Lanza M, Robbes R (2012) Evaluating defect prediction approaches: a benchmark and an extensive comparison. In: Empirical software engineering. pp 531–577

23. Pan SJ, Tsang IW, Kwok JT, Yang Q (2011) Domain adaptation via transfer component analysis. IEEE Trans Neural Networks 22:199–210. https://doi.org/10.1109/TNN.2010.2091281
24. Wu R, Zhang H, Kim S, Cheung SC (2011) ReLink: Recovering links between bugs and changes. ESEC/FSE 2011:15–25
25. Liu C, Yang D, Xia X, Yan M, Zhang X (2019) A two-phase transfer learning model for cross-project defect prediction. Inf Softw Technol 107:125–136. https://doi.org/10.1016/j.infsof.2018.11.005
26. Minku L, Sarro F, Mendes E, Ferrucci F (2015) How to make best use of cross-company data for web effort estimation? In: International symposium on empirical software engineering and measurement. IEEE Computer Society, pp 172–181
27. Wen W, Zhang B, Gu X, Ju X (2019) 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. Institute of Electrical and Electronics Engineers Inc., pp 29–38
28. Peng L, Yang B, Chen Y, Abraham A (2009) Data gravitation based classification. Inf Sci (NY) 179:809–819. https://doi.org/10.1016/j.ins.2008.11.007
29. Dai W, Yang Q, Xue GR, Yu Y (2007) Boosting for transfer learning. In: ACM international conference proceeding series. ACM Press, New York, New York, USA, pp 193–200
30. Xu Z, Pang S, Zhang T, Luo XP, Liu J, Tang YT, Yu X, Xue L (2019) Cross project defect prediction via balanced distribution adaptation based transfer learning. J Comput Sci Technol 34:1039–1062. https://doi.org/10.1007/s11390-019-1959-z
31. Wang J, Chen Y, Hao S, Feng W, Shen Z (2018) Balanced distribution adaptation for transfer learning. In: Proceedings of IEEE international conference data mining, ICDM 2017-November. pp 1129–1134
32. Ryu D, Jang JI, Baik J (2017) A transfer cost-sensitive boosting approach for cross-project defect prediction. Softw Qual J 25:235–272. https://doi.org/10.1007/s11219-015-9287-1
33. Fan W, Stolfo SJ, Zhang J, Chan PK (1999) AdaCost: misclassification cost-sensitive boosting. In: Sixteenth international conference on machine learning (ICML'99)
34. Eaton E, Desjardins M (2011) Selective transfer between learning tasks using task-based boosting. In: Twenty-fifth AAAI conference on artificial intelligence, AAAI 2011. pp 337–342
35. Tong H, Liu B, Wang S, Li Q (2019) Transfer-learning oriented class imbalance learning for cross-project defect prediction. In: Science & Technology on Reliability & Environmental Engineering Laboratory, Beihang University, Beijing, China
36. Wu F, Jing XY, Sun Y, Sun J, Huang L, Cui F, Sun Y (2018) Cross-project and within-project semisupervised software defect prediction: a unified approach. IEEE Trans Reliab 67:581–597. https://doi.org/10.1109/TR.2018.2804922
37. Liu X, Song M, Tao D, Zhou X, Chen C, Bu J (2014) Semi-supervised coupled dictionary learning for person re-identification. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition. IEEE Computer Society, pp 3550–3557
38. Lu H, Cukic B, Culp M (2012) Software defect prediction using semi-supervised learning with dimension reduction. In: 2012 27th IEEE/ACM international conference on automated software engineering, ASE 2012—Proceedings. pp 314–317
39. Lu H, Cukic B, Culp M (2014) A semi-supervised approach to software defect prediction. In: Proceedings—international computer software and applications conference. IEEE Computer Society, pp 416–425
40. Catal C (2014) A comparison of semi-supervised classification approaches for software defect prediction. J Intell Syst 23:75–82. https://doi.org/10.1515/jisys-2013-0030
41. Ma Y, Pan W, Zhu S, Yin H, Luo J (2014) An improved semi-supervised learning method for software defect prediction. J Intell Fuzzy Syst 27:2473–2480. https://doi.org/10.3233/IFS-141220
42. Thung F, Le XBD, Lo D (2015) Active semi-supervised defect categorization. In: IEEE international conference on program comprehension. IEEE Computer Society, pp 60–70
43. Zhang ZW, Jing XY, Wang TJ (2017) Label propagation based semi-supervised learning for software defect prediction. Autom Softw Eng 24:47–69. https://doi.org/10.1007/s10515-016-0194-x
44. Zhang F, Zheng Q, Zou Y, Hassan AE (2016) Cross-project defect prediction using a connectivity-based unsupervised classifier. In: Proceedings—international conference on software engineering. IEEE Computer Society, pp 309–320
45. Jing XY, Ying S, Zhang ZW, Wu SS, Liu J (2014) Dictionary learning based software defect prediction. In: Proceedings—international conference on software engineering. IEEE Computer Society, New York, New York, USA, pp 414–423
46. Panichella A, Oliveto R, De Lucia A (2014) Cross-project defect prediction models: L'Union fait la force. In: 2014 Software evolution week—IEEE conference on software maintenance, reengineering, and reverse engineering, CSMR-WCRE 2014—proceedings. IEEE Computer Society, pp 164–173
47. Xia X, Lo D, Pan SJ, Nagappan N, Wang X (2016) HYDRA: Massively compositional model for cross-project defect prediction. IEEE Trans Softw Eng 42:977–998. https://doi.org/10.1109/TSE.2016.2543218
48. Liu Y, Khoshgoftaar TM, Seliya N (2010) Evolutionary optimization of software quality modeling with multiple repositories. IEEE Trans Softw Eng 36:852–864. https://doi.org/10.1109/TSE.2010.51
49. Canfora G, De Lucia A, Di Penta M, Oliveto R, Panichella A, Panichella S (2013) Multi-objective cross-project defect prediction. In: Proceedings—IEEE 6th international conference on software testing, verification and validation, ICST 2013. IEEE, pp 252–261
50. Qiu S, Lu L, Jiang S (2018) Multiple-components weights model for cross-project software defect prediction. IET Softw 12:345–355. https://doi.org/10.1049/iet-sen.2017.0111
51. Chen J, Hu K, Yang Y, Liu Y, Xuan Q (2019) Collective transfer learning for defect prediction. Neurocomputing. https://doi.org/10.1016/j.neucom.2018.12.091
52. Jureczko M, Madeyski L (2010) Towards identifying software project clusters with regard to defect prediction. In: ACM international conference proceeding series. ACM Press, New York, New York, USA, p 1
53. Herzig K, Just S, Rau A, Zeller A (2013) Predicting defects using change genealogies. In: 2013 IEEE 24th international symposium on software reliability engineering, ISSRE 2013. pp 118–127
54. Herbold S (2018) Benchmarking cross-project defect prediction approaches with costs metrics. In: University of Goettingen, Insititute of Computer Science, Göttingen, Germany
55. Zhou Y, Yang Y, Lu H, Chen L, Li Y, Zhao Y, Qian J, Xu B (2018) How far we have progressed in the journey? An examination of cross-project defect prediction. ACM Trans Softw Eng Methodol 27:1–51. https://doi.org/10.1145/3183339
56. Jing X, Wu F, Dong X, Qi F, Xu B (2015) Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning. In: 2015 10th Joint meeting of the european software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, ESEC/FSE 2015—proceedings. Association for Computing Machinery, Inc, New York, New York, USA, pp 496–507
57. Hardoon DR, Szedmak S, Shawe-Taylor J (2004) Canonical correlation analysis: an overview with application to learning methods. Neural Comput 16:2639–2664

58. Nam J, Fu W, Kim S, Menzies T, Tan L (2018) Heterogeneous defect prediction. IEEE Trans Softw Eng 44:874–896. https://doi.org/10.1109/TSE.2017.2720603
59. Li Z, Jing XY, Zhu X, Zhang H (2017) Heterogeneous defect prediction through multiple kernel learning and ensemble learning. In: Proceedings—2017 IEEE international conference on software maintenance and evolution, ICSME 2017. Institute of Electrical and Electronics Engineers Inc., pp 91–102
60. Li Z, Jing XY, Wu F, Zhu X, Xu B, Ying S (2018) Cost-sensitive transfer kernel canonical correlation analysis for heterogeneous defect prediction. Autom Softw Eng 25:201–245. https://doi.org/10.1007/s10515-017-0220-7
61. He P, Li B, Liu X, Chen J, Ma Y (2015) An empirical study on software defect prediction with a simplified metric set. Inf Softw Technol 59:170–190. https://doi.org/10.1016/j.infsof.2014.11.006
62. Moser R, Pedrycz W, Succi G (2008) A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In: Proceedings—international conference on software engineering. pp 181–190
63. Nagappan N, Ball T (2005) Use of relative code churn measures to predict system defect density. In: Proceedings—27th international conference on software engineering, ICSE05. Association for Computing Machinery, New York, New York, USA, pp 284–292
64. Hassan AE, Holt RC (2005) The top ten list: dynamic fault prediction. In: IEEE international conference on software maintenance, ICSM. pp 263–272
65. Hassan AE (2009) Predicting faults using the complexity of code changes. In: Proceedings—international conference on software engineering. pp 78–88
66. Singh P, Pal NR, Verma S, Vyas OP (2017) Fuzzy rule-based approach for software fault prediction. IEEE Trans Syst Man Cybern Syst 47:826–837. https://doi.org/10.1109/TSMC.2016.2521840
67. Abaei G, Selamat A (2014) A survey on software fault detection based on different prediction approaches. Vietnam J Comput Sci 1:79–95. https://doi.org/10.1007/s40595-013-0008-z
68. Gao K, Khoshgoftaar TM, Wang H, Seliya N (2011) Choosing software metrics for defect prediction: an investigation on feature selection techniques. Softw Pract Exp 41:579–606. https://doi.org/10.1002/spe.1043
69. Shivaji S, James Whitehead E, Akella R, Kim S (2013) Reducing features to improve code change-based bug prediction. IEEE Trans Softw Eng 39:552–569. https://doi.org/10.1109/TSE.2012.43
70. Liu W, Liu S, Gu Q, Chen J, Chen X, Chen D (2016) Empirical studies of a two-stage data preprocessing approach for software fault prediction. IEEE Trans Reliab 65:38–53. https://doi.org/10.1109/TR.2015.2461676
71. Khoshgoftaar TM, Gao K, Napolitano A, Wald R (2014) A comparative study of iterative and non-iterative feature selection techniques for software defect prediction. Inf Syst Front 16:801–822. https://doi.org/10.1007/s10796-013-9430-0
72. Xu Z, Xuan J, Liu J, Cui X (2016) MICHAC: Defect prediction via feature selection based on Maximal Information Coefficient with Hierarchical Agglomerative Clustering. In: 2016 IEEE 23rd international conference on software analysis, evolution, and reengineering, SANER 2016. Institute of Electrical and Electronics Engineers Inc., pp 370–381
73. Turabieh H, Mafarja M, Li X (2019) Iterated feature selection algorithms with layered recurrent neural network for software fault prediction. Expert Syst Appl 122:27–42. https://doi.org/10.1016/j.eswa.2018.12.033
74. Ostrand TJ, Weyuker EJ, Bell RM (2005) Predicting the location and number of faults in large software systems. IEEE Trans Softw Eng 31:340–355. https://doi.org/10.1109/TSE.2005.49

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.