Heterogeneous Defect Prediction

Jaechang Nam and Sunghun Kim
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Hong Kong, China
{jcnam,hunkim}@cse.ust.hk

ABSTRACT

Software defect prediction is one of the most active research areas in software engineering. We can build a prediction model with defect data collected from a software project and predict defects in the same project, i.e. within-project defect prediction (WPDP). Researchers also proposed cross-project defect prediction (CPDP) to predict defects for new projects lacking in defect data by using prediction models built by other projects. In recent studies, CPDP is proved to be feasible. However, CPDP requires projects that have the same metric set, meaning the metric sets should be identical between projects. As a result, current techniques for CPDP are difficult to apply across projects with heterogeneous metric sets.

To address the limitation, we propose heterogeneous defect prediction (HDP) to predict defects across projects with heterogeneous metric sets. Our HDP approach conducts metric selection and metric matching to build a prediction model between projects with heterogeneous metric sets. Our empirical study on 28 subjects shows that about 68% of predictions using our approach outperform or are comparable to WPDP with statistical significance.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*software quality assurance*

General Terms

Algorithm, Experimentation

Keywords

Defect prediction, quality assurance, heterogeneous metrics

1. INTRODUCTION

Software defect prediction is one of the most active research areas in software engineering [8, 9, 24, 25, 26, 36, 37, 43, 47, 58, 59]. If software quality assurance teams can predict defects before releasing a software product, they can effectively allocate limited resources for quality control [36, 38, 43, 58]. For example, Ostrand et al. applied defect prediction in two large software systems of AT&T for effective and efficient testing activities [38].

Most defect prediction models are based on machine learning, therefore it is a must to collect defect datasets to train a prediction model [8, 36]. The defect datasets consist of various software metrics and labels. Commonly used software metrics for defect prediction are complexity metrics (such as lines of code, Halstead metrics, McCabe's cyclometic complexity, and CK metrics) and process metrics [2, 16, 32, 42]. Labels indicate whether the source code is buggy or clean for binary classification [24, 37].

Most proposed defect prediction models have been evaluated on within-project defect prediction (WPDP) settings [8, 24, 36]. In Figure 1a, each instance representing a source code file or function consists of software metric values and is labeled as buggy or clean. In the WPDP setting, a prediction model is trained using the labeled instances in Project A and predict unlabeled ('?') instances in the same project as buggy or clean.

However, it is difficult to build a prediction model for new software projects or projects with little historical information [59] since they do not have enough training instances. Various process metrics and label information can be extracted from the historical data of software repositories such as version control and issue tracking systems [42]. Thus, it is difficult to collect process metrics and instance labels in new projects or projects that have little historical data [9, 37, 59]. For example, without instances being labeled using past defect data it is not possible to build a prediction model.

To address this issue, researchers have proposed cross-project defect prediction (CPDP) [19, 29, 37, 43, 51, 59]. CPDP approaches predict defects even for new projects lacking in historical data by reusing prediction models built by other project datasets. As shown in Figure 1b, a prediction model is trained by labeled instances in Project A (source) and predicts defects in Project B (target).

However, most CPDP approaches have a serious limitation: CPDP is only feasible for projects which have exactly the same metric set as shown in Figure 1b. Finding other projects with exactly the same metric set can be challenging. Publicly available defect datasets that are widely used in defect prediction literature usually have heterogeneous metric sets [8, 35, 37]. For example, many NASA datasets in the

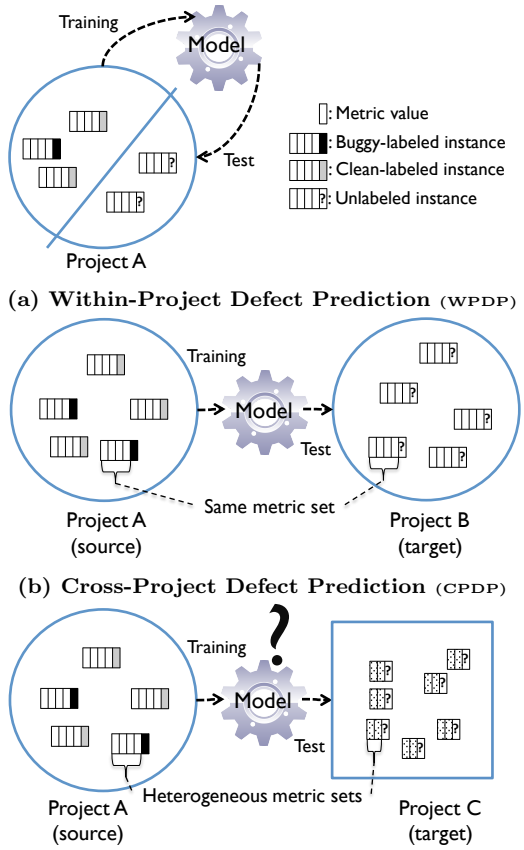

Figure 1: Various Defect Prediction Scenarios

PROMISE repository have 37 metrics but AEEEM datasets used by D'Ambroas et al. have 61 metrics [8, 35]. The only common metric between NASA and AEEEM datasets is *lines of code (LOC)*. CPDP between NASA and AEEEM datasets with all metric sets is not feasible since they have completely different metrics [51].

Some CPDP studies use only common metrics when source and target datasets have heterogeneous metric sets [29, 51]. For example, Turhan et al. use the only 17 common metrics between the NASA and SOFTLAB datasets that have heterogeneous metric sets [51]. However, finding other projects with multiple common metrics can be challenging. As mentioned, there is only one common metric between NASA and AEEEM. Also, only using common metrics may degrade the performance of CPDP models. That is because some informative metrics necessary for building a good prediction model may not be in the common metrics across datasets. For example, in the study of Turhan et al., the performance of CPDP (0.35) by their approach did not outperform that of WPDP (0.39) in terms of the average f-measure [51].

In this paper, we propose the heterogeneous defect prediction (HDP) approach to predict defects across projects even with heterogeneous metric sets. If the proposed approach is feasible as in Figure 1c, we could reuse any existing defect datasets to build a prediction model. For example, many PROMISE defect datasets even if they have heterogeneous metric sets [35] could be used as training datasets to predict defects in any project.

The key idea of our HDP approach is matching metrics that have similar distributions between source and target datasets. In addition, we also used metric selection to remove less informative metrics of a source dataset for a prediction model before metric matching.

Our empirical study shows that HDP models are feasible and their prediction performance is promising. About 68% of HDP predictions outperform or are comparable to WPDP predictions with statistical significance.

Our contributions are as follows:

- Propose the heterogeneous defect prediction models.
- Conduct an extensive and large-scale empirical study to evaluate the heterogeneous defect prediction models.

2. BACKGROUND AND RELATED WORK

The CPDP approaches have been studied by many researchers of late [29, 37, 43, 51, 59]. Since the performance of CPDP is usually very poor [59], researchers have proposed various techniques to improve CPDP [29, 37, 51, 54].

Watanabe et al. proposed the metric compensation approach for CPDP [54]. The metric compensation transforms a target dataset similar to a source dataset by using the average metric values [54]. To evaluate the performance of the metric compensation, Watanabe et al. collected two defect datasets with the same metric set (8 object-oriented metrics) from two software projects and then conducted CPDP [54].

Rahman et al. evaluated the CPDP performance in terms of cost-effectiveness and confirmed that the prediction performance of CPDP is comparable to WPDP [43]. For the empirical study, Rahman et al. collected 9 datasets with the same process metric set [43].

Fukushima et al. conducted an empirical study of just-in-time defect prediction in the CPDP setting [9]. They used 16 datasets with the same metric set [9]. The 11 datasets were provided by Kamei et al. but 5 projects were newly collected with the same metric set of the 11 datasets [9, 20].

However, collecting datasets with the same metric set might limit CPDP. For example, if existing defect datasets contain object-oriented metrics such as CK metrics [2], collecting the same object-oriented metrics is impossible for projects that are written in non-object-oriented languages.

Turhan et al. proposed the nearest-neighbour (NN) filter to improve the performance of CPDP [51]. The basic idea of the NN filter is that prediction models are built by source instances that are nearest-neighbours of target instances [51]. To conduct CPDP, Turhan et al. used 10 NASA and SOFTLAB datasets in the PROMISE repository [35, 51].

Ma et al. proposed Transfer Naive Bayes (TNB) [29]. The TNB builds a prediction model by weighting source instances similar to target instances [29]. Using the same datasets used by Turhan et al., Ma et al. evaluated the TNB models for CPDP [29, 51].

Since the datasets used in the empirical studies of Turhan et al. and Ma et al. have heterogeneous metric sets, they conducted CPDP using the common metrics [29, 51]. There is another CPDP study with the top-K common metric subset [17]. However, as explained in Section 1, CPDP using common metrics is worse than WPDP [17, 51].

Nam et al. adapted a state-of-the-art transfer learning technique called Transfer Component Analysis (TCA) and proposed TCA+ [37]. They used 8 datasets in two groups, ReLink and AEEEM, with 26 and 61 metrics respectively [37].

However, Nam et al. could not conduct CPDP between ReLink and AEEEM because they have heterogeneous metric sets. Since the project pool with the same metric set is very limited, conducting CPDP using a project group with the same metric set can be limited as well. For example, at most 18% of defect datasets in the PROMISE repository have the same metric set [35]. In other words, we cannot directly conduct CPDP for the 18% of the defect datasets by using the remaining (82%) datasets in the PROMISE repository [35]. CPDP studies conducted by Canfora et al. and Panichella et al. use 10 Java projects only with the same metric set from the PROMISE repository [4, 35, 39]

Zhang et al. proposed the universal model for CPDP [57]. The universal model is built using 1398 projects from SourceForge and Google code and leads to comparable prediction results to WPDP in their experimental setting [57].

However, the universal defect prediction model may be difficult to apply for the projects with heterogeneous metric sets since the universal model uses 26 metrics including code metrics, object-oriented metrics, and process metrics. In other words, the model can only be applicable for target datasets with the same 26 metrics. In the case where the target project has not been developed in object-oriented languages, a universal model built using object-oriented metrics cannot be used for the target dataset.

He et al. addressed the limitations due to heterogeneous metric sets in CPDP studies listed above [18]. Their approach, CPDP-IFS, used distribution characteristic vectors of an instance as metrics. The prediction performance of their best approach is comparable to or helpful in improving regular CPDP models [18].

However, the approach by He et al. is not compared with WPDP [18]. Although their best approach is helpful to improve regular CPDP models, the evaluation might be weak since the prediction performance of a regular CPDP is usually very poor [59]. In addition, He et al. conducted experiments on only 11 projects in 3 dataset groups [18].

We propose HDP to address the above limitations caused by projects with heterogeneous metric sets. Contrary to the study by He et al. [18], we compare HDP to WPDP, and HDP achieved better or comparable prediction performance to WPDP in about 68% of predictions. In addition, we conducted extensive experiments on 28 projects in 5 dataset groups. In Section 3, we explain our approach in detail.

3. APPROACH

Figure 2 shows the overview of HDP based on metric selection and metric matching. In the figure, we have two datasets, Source and Target, with heterogeneous metric sets. Each row and column of a dataset represents an instance and a metric, respectively, and the last column represents instance labels. As shown in the figure, the metric sets in the source and target datasets are not identical (X_1 to X_4 and Y_1 to Y_7 respectively).

When given source and target datasets with heterogeneous metric sets, for metric selection we first apply a feature selection technique to the source. Feature selection is a common approach used in machine learning for selecting a subset of features by removing redundant and irrelevant features [13]. We apply widely used feature selection techniques for metric selection of a source dataset as in Section 3.1 [10, 47].

After that, metrics based on their similarity such as distribution or correlation between the source and target metrics are matched up. In Figure 2, three target metrics are matched with the same number of source metrics.

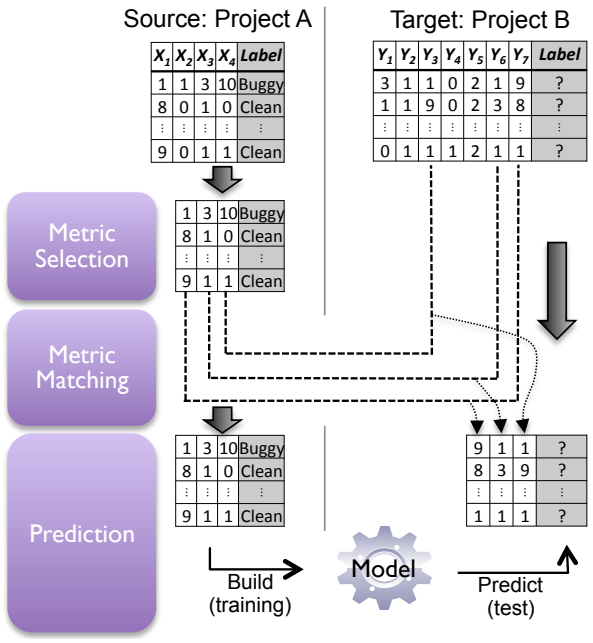

Figure 2: Heterogeneous defect prediction

After these processes, we finally arrive at a matched source and target metric set. With the final source dataset, HDP builds a model and predicts labels of target instances.

In the following subsections, we explain the metric selection and matching in detail.

3.1 Metric Selection in Source Datasets

For metric selection, we used various feature selection approaches widely used in defect prediction such as gain ratio, chi-square, relief-F, and significance attribute evaluation [10, 47]. According to benchmark studies about various feature selection approaches, a single best feature selection approach for all prediction models does not exist [5, 15, 28]. For this reason, we conduct experiments under different feature selection approaches. When applying feature selection approaches, we select top 15% of metrics as suggested by Gao et al. [10]. In addition, we compare the prediction results with or without metric selection in the experiments.

3.2 Matching Source and Target Metrics

To match source and target metrics, we measure the similarity of each source and target metric pair by using several existing methods such as percentiles, Kolmogorov-Smirnov Test, and Spearman's correlation coefficient [30, 49]. We define the following three analyzers for metric matching:

- Percentile based matching (PAnalyzer)
- Kolmogorov-Smirnov Test based matching (KSAnalyzer)
- Spearman's correlation based matching (SCoAnalyzer)

The key idea of these analyzers is computing matching scores for all pairs between the source and target metrics. Figure 3 shows a sample matching. There are two source metrics (X_1 and X_2) and two target metrics (Y_1 and Y_2). Thus, there are four possible matching pairs, (X_1,Y_1), (X_1,Y_2), (X_2,Y_1), and (X_2,Y_2). The numbers in rectangles between

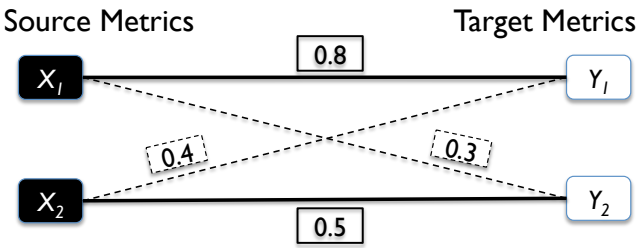

Figure 3: An example of metric matching between source and target datasets.

matched source and target metrics in Figure 3 represent matching scores computed by an analyzer. For example, the matching score between the metrics, X_1 and Y_1, is 0.8.

From all pairs between the source and target metrics, we remove poorly matched metrics whose matching score is not greater than a specific cutoff threshold. For example, if the matching score cutoff threshold is 0.3, we include only the matched metrics whose matching score is greater than 0.3. In Figure 3, the edge (X_1,Y_2) in matched metrics will be excluded when the cutoff threshold is 0.3. Thus, all the candidate matching pairs we can consider include the edges (X_1,Y_1), (X_2,Y_2), and (X_2,Y_1) in this example. In Section 4, we design our empirical study under different matching score cutoff thresholds to investigate their impact on prediction.

We may not have any matched metrics based on the cutoff threshold. In this case, we cannot conduct defect prediction. In Figure 3, if the cutoff threshold is 0.9, none of the matched metrics are considered for HDP so we cannot build a prediction model for the target dataset. For this reason, we investigate target prediction coverage (i.e. what percentage of target datasets could be predicted?) in our experiments.

After applying the cutoff threshold, we used *the maximum weighted bipartite matching* [31] technique to select a group of matched metrics, whose sum of matching scores is highest, without duplicated metrics. In Figure 3, after applying the cutoff threshold of 0.30, we can form two groups of matched metrics without duplicated metrics. The first group consists of the edges, (X_1,Y_1) and (X_2,Y_2), and another group consists of the edge (X_2,Y_1). In each group, there are no duplicated metrics. The sum of matching scores in the first group is 1.3 (=0.8+0.5) and that of the second group is 0.4. The first group has a greater sum (1.3) of matching scores than the second one (0.4). Thus, we select the first matching group as the set of matched metrics for the given source and target metrics with the cutoff threshold of 0.30 in this example.

Each analyzer for the metric matching scores is described below.

3.2.1 PAnalyzer

PAnalyzer simply compares 9 percentiles (10th, 20th,..., 90th) of ordered values between source and target metrics.

First, we compute the difference of n-th percentiles in source and target metric values by the following equation:

$$P_{ij}(n) = \frac{sp_{ij}(n)}{bp_{ij}(n)} \quad (1)$$

, where $P_{ij}(n)$ is the comparison function for n-th percentiles of i-th source and j-th target metrics, and $sp_{ij}(n)$ and $bp_{ij}(n)$ are smaller and bigger percentile values respectively at n-th percentiles of i-th source and j-th target metrics. For example, if the 10th percentile of the source metric values is 20 and that of target metric values is 15, the difference is 0.75 ($P_{ij}(10) = 15/20 = 0.75$).

Using this percentile comparison function, a matching score between source and target metrics is calculated by the following equation:

$$M_{ij} = \frac{\sum_{k=1}^{9} P_{ij}(10 \times k)}{9} \quad (2)$$

, where M_{ij} is a matching score between i-th source and j-th target metrics. The best matching score of this equation is 1.0 when the values of the source and target metrics of all 9 percentiles are the same.

3.2.2 KSAnalyzer

KSAnalyzer uses a p-value from the Kolmogorov-Smirnov Test (KS-test) as a matching score between source and target metrics. The KS-test is a non-parametric two sample test that can be applicable when we cannot be sure about the normality of two samples and/or the same variance [27, 30]. Since metrics in some defect datasets used in our empirical study have exponential distributions [36] and metrics in other datasets have unknown distributions and variances, the KS-test is a suitable statistical test to compute p-values for these datasets. In statistical testing, a p-value shows the probability of whether two samples are significantly different or not. We used the *KolmogorovSmirnovTest* implemented in the *Apache commons math* library.

The matching score is:

$$M_{ij} = p_{ij} \quad (3)$$

, where p_{ij} is a p-value from the KS-test of i-th source and j-th target metrics. A p-value tends to be zero when two metrics are significantly different.

3.2.3 SCoAnalyzer

In SCoAnalyzer, we used the Spearman's rank correlation coefficient as a matching score for source and target metrics [49]. Spearman's rank correlation measures how two samples are correlated [49]. To compute the coefficient, we used the *SpearmansCorrelation* in the *Apache commons math* library. Since the size of metric vectors should be the same to compute the coefficient, we randomly select metric values from a metric vector that is of a greater size than another metric vector. For example, if the sizes of the source and target metric vectors are 110 and 100 respectively, we randomly select 100 metric values from the source metric to agree to the size between the source and target metrics. All metric values are sorted before computing the coefficient.

The matching score is as follows:

$$M_{ij} = c_{ij} \quad (4)$$

, where c_{ij} is a Spearman's rank correlation coefficient between i-th source and j-th target metrics.

3.3 Building Prediction Models

After applying metric selection and matching, we can finally build a prediction model using a source dataset with selected and matched metrics. Then, as a regular defect prediction model, we can predict defects on a target dataset with metrics matched to selected source metrics.

Table 1: The 28 defect datasets from five groups.

Group	Dataset	# of instances		# of metrics	Prediction Granularity
		All	Buggy(%)		
AEEEM [8, 37]	EQ	325	129(39.69%)	61	Class
	JDT	997	206(20.66%)		
	LC	399	64(9.26%)		
	ML	1862	245(13.16%)		
	PDE	1492	209(14.01%)		
ReLink [56]	Apache	194	98(50.52%)	26	File
	Safe	56	22(39.29%)		
	ZXing	399	118(29.57%)		
MORPH [40]	ant-1.3	125	20(16.00%)	20	Class
	arc	234	27(11.54%)		
	camel-1.0	339	13(3.83%)		
	poi-1.5	237	141(59.49%)		
	redaktor	176	27(15.34%)		
	skarbonka	45	9(20.00%)		
	tomcat	858	77(8.97%)		
	velocity-1.4	196	147(75.00%)		
	xalan-2.4	723	110(15.21%)		
	xerces-1.2	440	71(16.14%)		
NASA [35, 45]	cm1	327	42(12.84%)	37	Function
	mw1	253	27(10.67%)		
	pc1	705	61(8.65%)		
	pc3	1077	134(12.44%)		
	pc4	1458	178(12.21%)		
SOFTLAB [51]	ar1	121	9(7.44%)	29	Function
	ar3	63	8(12.70%)		
	ar4	107	20(18.69%)		
	ar5	36	8(22.22%)		
	ar6	101	15(14.85%)		

4. EXPERIMENTAL SETUP

4.1 Research Questions

To systematically evaluate heterogeneous defect prediction (HDP) models, we set three research questions.

- RQ1: Is heterogeneous defect prediction comparable to WPDP (Baseline1)?
- RQ2: Is heterogeneous defect prediction comparable to CPDP using common metrics (CPDP-CM, Baseline2)?
- RQ3: Is heterogeneous defect prediction comparable to CPDP-IFS (Baseline3)?

RQ1, RQ2, and RQ3 lead us to investigate whether our HDP is comparable to WPDP (Baseline1), CPDP-CM (Baseline2), and CDDP-IFS (Baseline3) [18].

4.2 Benchmark Datasets

We collected publicly available datasets from previous studies [8, 37, 40, 51, 56]. Table 1 lists all dataset groups used in our experiments. Each dataset group has a heterogeneous metric set as shown in the table. Prediction Granularity in the last column of the table means the prediction granularity of instances. Since we focus on the distribution or correlation of metric values when matching metrics, it is beneficial to be able to apply the HDP approach on datasets even in different granularity levels.

We used five groups with 28 defect datasets: AEEEM, ReLink, MORPH, NASA, and SOFTLAB.

AEEEM was used to benchmark different defect prediction models [8] and to evaluate CPDP techniques [18, 37]. Each AEEEM dataset consists of 61 metrics including object-oriented (OO) metrics, previous-defect metrics, entropy metrics of change and code, and churn-of-source-code metrics [8].

Datasets in ReLink were used by Wu et al. [56] to improve the defect prediction performance by increasing the quality of the defect data and have 26 code complexity metrics extracted by the Understand tool [52].

The MORPH group contains defect datasets of several open source projects used in the study about the dataset privacy issue for defect prediction [40]. The 20 metrics used in MORPH are McCabe's cyclomatic metrics, CK metrics, and other OO metrics [40].

NASA and SOFTLAB contain proprietary datasets from NASA and a Turkish software company, respectively [51]. We used five NASA datasets, which share the same metric set in the PROMISE repository [35, 45]. We used cleaned NASA datasets (DS′ version) [45]. For the SOFTLAB group, we used all SOFTLAB datasets in the PROMISE repository [35]. The metrics used in both NASA and SOFTLAB groups are Halstead and McCabe's cyclomatic metrics but NASA has additional complexity metrics such as *parameter count* and *percentage of comments* [35].

Predicting defects is conducted across different dataset groups. For example, we build a prediction model by Apache in ReLink and tested the model on velocity-1.4 in MORPH (Apache⇒velocity-1.4).[1]

We did not conduct defect prediction across projects in the same group where datasets have the same metric set since the focus of our study is on prediction across datasets with heterogeneous metric sets. In total, we have 600 possible prediction combinations from these 28 datasets.

4.3 Matching Score Cutoff Thresholds

To build HDP models, we apply various cutoff thresholds for matching scores to observe how prediction performance varies according to different cutoff values. Matched metrics by analyzers have their own matching scores as explained in Section 3. We apply different cutoff values (0.05 and 0.10, 0.20,. . . ,0.90) for the HDP models. If a matching score cutoff is 0.50, we remove matched metrics with the matching score $\leq$ 0.50 and build a prediction model with matched metrics with the score $>$ 0.50. The number of matched metrics varies by each prediction combination. For example, when using KSAnalyzer with the cutoff of 0.05, the number of matched metrics is four in cm1⇒ar5 while that is one in ar6⇒pc3. The average number of matched metrics also varies by analyzers and cutoff values; 4 (PAnalyzer), 2 (KSAnalyzer), and 5 (SCoAnalyzer) in the cutoff of 0.05 but 1 (PAnalyzer), 1 (KSAnalyzer), and 4 (SCoAnalyzer) in the cutoff of 0.90 on average.

4.4 Baselines

We compare HDP to three baselines: WPDP (Baseline1), CPDP using common metrics (CPDP-CM) between source and target datasets (Baseline2), and CPDP-IFS (Baseline3).

We first compare HDP to WPDP. Comparing HDP to WPDP will provide empirical evidence of whether our HDP models are applicable in practice.

We conduct CPDP using only common metrics (CPDP-CM) between source and target datasets as in previous CPDP studies [18, 29, 51]. For example, AEEEM and MORPH have OO metrics as common metrics so we use them to build prediction models for datasets between AEEEM and MORPH. Since using common metrics has been adopted to address the limitation on heterogeneous metric sets in previous CPDP studies [18, 29, 51], we set CPDP-CM as a baseline to evaluate our HDP models. The number of matched metrics varies across the dataset group. Between AEEEM

[1]Hereafter a rightward arrow (⇒) denotes a prediction combination.

and ReLink, only one common metric exists, *LOC*. NASA and SOFTLAB have 28 common metrics. On average, the number of common metrics in our datasets are about five.

We include CPDP-IFS proposed by He et al. as a baseline [18]. CPDP-IFS enables defect prediction on projects with heterogeneous metric sets (Imbalanced Feature Sets) by using the 16 distribution characteristics of values of each instance such as mode, median, mean, harmonic mean, minimum, maximum, range, variation ratio, first quartile, third quartile, interquartile range, variance, standard deviation, coefficient of variance, skewness, and kurtosis [18]. The 16 distribution characteristics are used as features to build a prediction model.

4.5 Experimental Design

For the machine learning algorithm, we use Logistic regression, which is widely used for both WPDP and CPDP studies [34, 37, 46, 59]. We use Logistic regression implemented in Weka with default options [14].

For WPDP, it is necessary to split datasets into training and test sets. We use 50:50 random splits, which are widely used in the evaluation of defect prediction models [22, 37, 41]. For the 50:50 random splits, we use one half of the instances for training a model and the rest for test (round 1). In addition, we use the two splits in a reverse way, where we use the previous test set for training and the previous training set for test (round 2). We repeat these two rounds 500 times, i.e. 1000 tests, since there is a randomness in selecting instances for each split [1]. Simply speaking, we repeat the two-fold cross validation 500 times.

For CPDP-CM, CPDP-IFS, and HDP, we build a prediction model by using a source dataset and test the model on the same test splits used in WPDP. Since there are 1000 different test splits for a within-project prediction, the CPDP-CM, CPDP-IFS, and HDP models are tested on 1000 different test splits as well.

4.6 Measures

To evaluate the prediction performance, we use the area under the receiver operating characteristic curve (AUC). The AUC is known as a useful measure for comparing different models and is widely used because AUC is unaffected by class imbalance as well as being independent from the cutoff probability (prediction threshold) that is used to decide whether an instance should be classified as positive or negative [12, 25, 43, 48]. Mende confirmed that it is difficult to compare the defect prediction performance reported in the defect prediction literature since prediction results come from the different cutoffs of prediction thresholds [33]. However, the receiver operating characteristic curve is drawn by both the true positive rate (recall) and the false positive rate on various prediction threshold values. The higher AUC represents better prediction performance and the AUC of 0.5 means the performance of a random predictor [43].

To compare HDP by our approach to baselines, we also use the Win/Tie/Loss evaluation, which is used for performance comparison between different experimental settings in many studies [23, 26, 53]. As we repeat the experiments 1000 times for a target project dataset, we conduct the Wilcoxon signed-rank test ($p<0.05$) for all AUC values in baselines and HDP [55]. If an HDP model for the target dataset outperforms a corresponding baseline result after the statistical test, we mark this HDP model as a 'Win'. In a similar way, we mark an HDP model as a 'Loss' when the results of a baseline outperforms that of our HDP approach with statistical significance. If there is no difference between a baseline and HDP with statistical significance, we mark this case as a 'Tie'. Then, we count the number of wins, ties, and losses for HDP models. By using the Win/Tie/Loss evaluation, we can investigate how many HDP predictions it will take to improve baseline approaches.

Table 2: Comparison results among WPDP, CPDP-CM, CPDP-IFS, and HDP by KSAnalyzer with the cutoff of 0.05 in a median AUC.

Target	WPDP (Baseline1)	CPDP-CM (Baseline2)	CPDP-IFS (Baseline3)	HDP KSAnalyzer cutoff=0.05
EQ	0.583	0.776	0.461	0.783
JDT	0.795	0.781	0.543	0.767
LC	0.575	0.636	0.584	0.655
ML	**0.734**	0.651	0.557	0.692*
PDE	0.684	0.682	0.566	0.717
Apache	0.714	0.689	0.635	0.717*
Safe	0.706	0.749	0.616	**0.818***
Zxing	0.605	0.619	0.530	**0.650***
ant-1.3	0.609	0.590	0.500	0.835
arc	0.670	0.611	0.523	0.701
camel-1.0	0.550	0.590	0.500	0.639
poi-1.5	0.707	0.676	0.606	0.701
redaktor	0.744	0.500	0.500	0.537
skarbonka	0.569	0.736	0.528	**0.694***
tomcat	0.778	0.746	0.640	0.818
velocity-1.4	0.725	0.609	0.500	0.391
xalan-2.4	0.755	0.658	0.499	0.751
xerces-1.2	0.624	0.453	0.500	0.489
cm1	0.653	0.622	0.551	**0.717***
mw1	0.612	0.584	0.614	0.727
pc1	0.787	0.675	0.564	0.752*
pc3	**0.794**	0.665	0.500	0.738*
pc4	**0.900**	0.773	0.589	0.682*
ar1	0.582	0.464	0.500	**0.734***
ar3	0.574	0.862	0.682	**0.823***
ar4	0.657	0.588	0.575	**0.816***
ar5	0.804	0.875	0.585	0.911*
ar6	0.654	0.611	0.527	0.640
All	0.657	0.636	0.555	**0.724***

5. RESULTS

In this section, we report the experimental results of the HDP approach by significance attribute selection for metric selection and KSAnalyzer with the cutoff threshold of 0.05.

Among different metric selections, significance attribute selection led to the best prediction performance overall. In terms of analyzers, KSAnalyzer led to the best prediction performance. Since the KSAnalyzer is based on the p-value of a statistical test, we chose a cutoff of 0.05 which is a commonly accepted significance level in the statistical test [7].

5.1 Comparison Result with Baselines

Table 2 shows the prediction performance (a median AUC) of baselines and HDP by KSAnalyzer with the cutoff of 0.05, for each target as well as all targets (the last row in the table). Baseline1 represents the WPDP results of a target project and Baseline2 shows the CPDP results using common metrics (CPDP-CM) between source and target projects. Baseline3 shows the results of CPDP-IFS proposed by He et al. [18]. The last column shows the HDP results by KSAnalyzer with the cutoff of 0.05. If there are better results between Baseline1 and our approach with statistical

Table 3: Median AUCs of baselines and HDP in KSAnalyzer (cutoff=0.05) by each source group.

Source	WPDP (Baseline1)	CPDP-CM (Baseline2)	CPDP-IFS (Baseline3)	HDP KS,0.05	Target Coverage of HDP
AEEEM	0.654	0.736	0.528	**0.739***	48%
ReLink	0.654	0.665	0.500	0.702*	88%
MORPH	0.657	0.667	0.590	**0.736***	100%
NASA	0.654	0.527	0.500	**0.734***	52%
SOFTLAB	0.695	0.612	0.554	0.708*	100%

significance (Wilcoxon signed-rank test [55], $p<0.05$), the better AUC values are in bold font as shown in Table 2. Between Baseline2 and our approach, better AUC values with statistical significance are underlined in the table. Between Baseline3 and our approach, better AUC values with statistical significance are shown with an asterisk (*).

We observed the following results about RQ1:

- In 25 out of 28 targets, HDP by KSAnalyzer with the cutoff of 0.05 leads to better or comparable results against WPDP with statistical significance. (The WPDP results in only ML, pc3, and pc4 are in bold font.)
- HDP by KSAnalyzer with the cutoff of 0.05 outperforms WPDP with statistical significance when considering results from all targets (*All* in the last row in the table) together in our experimental settings.

The following results are related to RQ2:

- HDP by KSAnalyzer with the cutoff of 0.05 leads to better or comparable results to CPDP-CM with statistical significance. (no underlines in CPDP-CM of Table 2)
- HDP by KSAnalyzer with the cutoff of 0.05 outperforms CPDP-CM with statistical significance when considering results from *All* targets in our experimental settings.

In terms of RQ3, we observed the following results:

- HDP by KSAnalyzer with the cutoff of 0.05 leads to better or comparable results to CPDP-IFS with statistical significance. (no asterisks in CPDP-IFS of Table 2)
- HDP by KSAnalyzer with the cutoff of 0.05 outperforms CPDP-IFS with statistical significance when considering results from *All* targets in our experimental settings.

5.2 Target Prediction Coverage

Target prediction coverage shows how many target projects can be predicted by the HDP models. If there are no feasible prediction combinations for a target because of there being no matched metrics between source and target datasets, it might be difficult to use an HDP model in practice.

For target prediction coverage, we analysed our HDP results by KSAnalyzer with the cutoff of 0.05 by each source group. For example, after applying metric selection and matching, we can build a prediction model by using EQ in AEEEM and predict each of 23 target projects in four other dataset groups. However, because of the cutoff value, some predictions may not be feasible. For example, EQ⇒Apache was not feasible because there are no matched metrics whose matching scores are greater than 0.05. Instead, another source dataset, JDT, in AEEEM has matched metrics to Apache. In this case, we consider the source group, AEEEM, covered Apache. In other words, if any dataset in a source group can be used to build an HDP model for a target, we count the target prediction is as covered.

Table 3 shows the median AUCs and prediction target coverage. The median AUCs were computed by the AUC values of the feasible HDP predictions and their corresponding predictions of WPDP, CPDP-CM, and CPDP-IFS. We conducted the Wilcoxon signed-rank test on results between WPDP and baselines [55]. Like Table 2, better results between baselines and our approach with statistical significance are in bold font, underlined, and/or with asterisks.

First of all, in each source group, we could observe HDP outperforms or is comparable to WPDP with statistical significance. For example, target projects were predicted by some projects in ReLink and the median AUC for HDP by KSAnalyzer is 0.702 while that of WPDP is 0.654. In addition, HDP by KSAnalyzer also outperforms or had a comparable prediction performance against CPDP-CM. There are no better results in CPDP-CM than those in HDP by KSAnalyzer with statistical significance (no underlined results in third column in Table 3). In addition, HDP by KSAbalyzer outperforms CPDP-IFS in all source groups.

The target prediction coverage in the MORPH and SOFTLAB groups yielded 100% as shown in Table 3. This implies our HDP models may conduct defect prediction with high target coverage even using datasets which only appear in one source group. AEEEM, ReLink, and NASA groups have 48%, 88%, and 52% respectively since some prediction combinations do not have matched metrics because of low matching scores ($\leq$0.05). Thus, some prediction combinations using matched metrics with low matching scores can be automatically excluded. In this sense, our HDP approach follows a similar concept to the two-phase prediction model [21]: (1) checking prediction feasibility between source and target datasets, and (2) predicting defects.

5.3 Win/Tie/Loss Results

To investigate our performance evaluation in detail, we report the Win/Tie/Loss results of HDP by KSAnalyzer with the cutoff of 0.05 against WPDP (Baseline1), CPDP-CM (Baseline2), and CPDP-IFS (Baseline3) in Table 4.

KSAnalyzer with the cutoff of 0.05 conducted 222 out of 600 prediction combinations since 378 combinations do not have any matched metrics because of the cutoff threshold. In Table 4, the target dataset, EQ, was predicted in four prediction combinations and our approach, HDP, outperforms Baseline1 and Baseline3 in the four combinations (i.e. 4 Wins). However, HDP outperforms Baseline2 in only two combinations of the target, EQ (2 Wins).

Against Baseline1, the six targets such as EQ, Zxing, skarbonka, tomcat, ar3, and ar4 have only Win results. In other words, defects in those six targets could be predicted better by other source projects using HDP models by KSAnalyzer compared to WPDP models.

However, the eight targets such as JDT, ML, redaktor, velocity-1.4, xalan-2.4, xerces-1.2, pc3, and pc4 have no Wins at all against Baseline1. In addition, other targets still have Losses even though they have Win or Tie results.

Overall, the numbers of Win and Tie results are 147 and 3 respectively out of all of the 222 prediction combinations. This means that about 67.6% of prediction combinations by our HDP models achieve better or comparable prediction performance than those in WPDP.

The Win/Tie/Loss results against Baseline2 and Baseline3 show a similar trend. The HDP results in the 161 (72.5%) out of 222 prediction combinations show HDP out-

Table 4: Win/Tie/Loss results of HDP by KS-Analyzer (cutoff=0.05) against WPDP (Baseline1), CPDP-CM (Baseline2), and CPDP-IFS (Baseline3).

Target	Against								
	WPDP (Baseline1)			CPDP-CM (Baseline2)			CPDP-IFS (Baseline3)		
	Win	Tie	Loss	Win	Tie	Loss	Win	Tie	Loss
EQ	4	0	0	2	2	0	4	0	0
JDT	0	0	5	3	0	2	5	0	0
LC	6	0	1	3	3	1	3	1	3
ML	0	0	6	4	2	0	6	0	0
PDE	3	0	2	2	0	3	5	0	0
Apache	6	0	5	8	1	2	9	0	2
Safe	14	0	3	12	0	5	15	0	2
Zxing	8	0	0	6	0	2	7	0	1
ant-1.3	6	0	1	6	0	1	5	0	2
arc	3	1	0	3	0	1	4	0	0
camel-1.0	3	0	2	3	0	2	4	0	1
poi-1.5	2	0	2	3	0	1	2	0	2
redaktor	0	0	4	2	0	2	3	0	1
skarbonka	11	0	0	4	0	7	9	0	2
tomcat	2	0	0	1	1	0	2	0	0
velocity-1.4	0	0	3	0	0	3	0	0	3
xalan-2.4	0	0	1	1	0	0	1	0	0
xerces-1.2	0	0	3	3	0	0	1	0	2
cm1	7	1	2	8	0	2	9	0	1
mw1	5	0	1	4	0	2	4	0	2
pc1	1	0	5	5	0	1	6	0	0
pc3	0	0	7	7	0	0	7	0	0
pc4	0	0	7	2	0	5	7	0	0
ar1	14	0	1	14	0	1	11	0	4
ar3	15	0	0	5	0	10	10	2	3
ar4	16	0	0	14	1	1	15	0	1
ar5	14	0	4	14	0	4	16	0	2
ar6	7	1	7	8	4	3	12	0	3
Total	147 66.2%	3 1.4%	72 32.4%	147 66.2%	14 6.3%	61 27.5%	182 82.0%	3 1.3%	37 16.7%

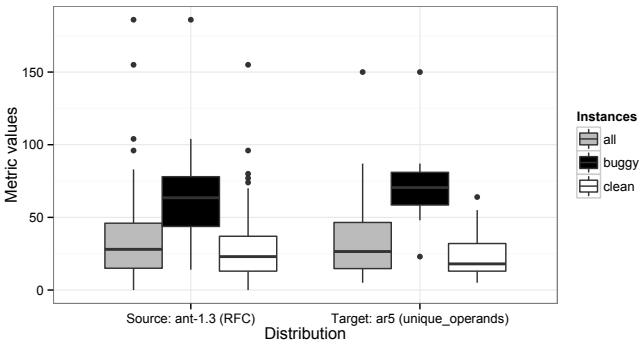

Figure 4: Distribution of metrics (matching score=0.91) from ant-1.3⇒ar5 (AUC=0.946).

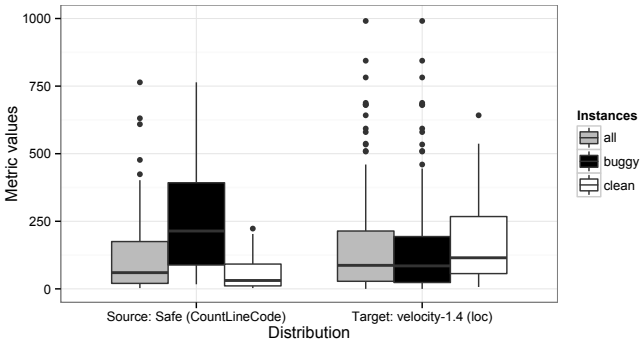

Figure 5: Distribution of metrics (matching score=0.45) from Safe⇒velocity-1.4 (AUC=0.391).

performs and is comparable to CPDP-CM. Against Baseline3, 185 (83.3%) predction combinations are Win or Tie results.

The Win/Tie/Loss results show that with our HDP model by KSAnalyzer there is a higher possibility of getting a better prediction performance.

However, there are still about 32% Loss results against WPDP. In Section 6, we discuss and analyze why Loss results happen.

6. DISCUSSION

6.1 Why Matched Metric Works?

In Figure 4, we use box plots to represent distributions of matched metrics. The gray, black, and white box plots shows distributions of matched metrics in all, buggy, and clean instances respectively. The three box plots on the left-hand side represent distributions of a source metric while the three box plots on the right-hand side represent those of a target metric. The bottom and top of the boxes represent the first and third quartiles respectively. The solid horizontal line in a box represents the median value in each distribution. Black points in the figure are outliers.

Figure 4 explains how the prediction combination of ant-1.3⇒ar5 can have a high AUC, 0.946. Suppose that a simple model predicts that an instance is buggy when the metric value of the instance is more than 40 in the case of Figure 4. In both datasets, approximately 75% or more buggy and clean instances will be predicted correctly. In Figure 4, the matched metrics in ant-1.3⇒ar5 are the response for class (*RFC*: number of methods invoked by a class) [6] and the number of unique operands (*unique_operands*) [16], respectively. The *RFC* and *unique_operands* are not the same metric so it might look like an arbitrary matching. However, they are matched based on their similar distributions as shown in Figure 4. Typical defect prediction metrics have tendencies in which higher complexity causes more bug-proneness [8, 36, 42]. In Figure 4, instances with higher values of *RFC* and *unique_operands* have the tendency to be more bug-prone. For this reason, the model using the matched metrics could achieve such a high AUC (0.938). We could observe this bug-prone tendency in other Win results. Since matching metrics is based on similarity of source and target metric distributions, HDP also addresses several issues related to a dataset shift such as the covariate shift and domain shift discussed by Turhan [50].

Some prediction combinations have Loss results in Table 4. We investigated matched metrics in these prediction combinations. In velocity-1.4 of Table 4, all results are Loss. Thus, as a representative loss result, we discuss the prediction combination, Safe⇒velocity-1.4, whose AUC is 0.391.

As observed, Loss results were usually caused by different tendencies of bug-proneness between source and target metrics. Figure 5 shows how the bug-prone tendencies of source and target metrics are different. Interestingly, the matched source and target metric by the KSAnalyzer is the same as *LOC* (*CountLineCode* and *loc*) in both. As we observe in the figure, the median of buggy instance values of the source metric is higher than that of clean instances in

Table 5: Prediction performance (a median AUC and % of Win) in different metric selections.

Approach	Against						HDP
	WPDP		CPDP-CM		CPDP-IFS		
	AUC	Win%	AUC	Win%	AUC	Win%	AUC
Gain Ratio	0.657	63.7%	0.645	63.2%	0.536	80.2%	0.720
Chi-Square	0.657	64.7%	0.651	66.4%	0.556	82.3%	0.727
Significance	0.657	66.2%	0.636	66.2%	0.553	82.0%	0.724
Relief-F	0.670	57.0%	0.657	63.1%	0.543	80.5%	0.709
None	0.657	47.3%	0.624	50.3%	0.536	66.3%	0.663

that the more *LOC* implies the higher bug-proneness in the case of Safe. However, the median of buggy instance values in a target metric is lower than that of clean instance values in that the less *LOC* implies the higher bug-proneness in velocity-1.4. This inconsistent tendency of bug-proneness between the source and target metrics could degrade the prediction performance although they are the same metric.

We regard the matching that has an inconsistent bug-prone tendency between source and target metrics as a noisy metric matching. We could observe this kind of noisy metric matching in prediction combinations in other Loss results.

However, it is very challenging to filter out the noisy metric matching since we cannot know labels of target instances in advance. If we could design a filter for the noisy metric matching, the Loss results would be minimized. Thus, designing a new filter to mitigate these Loss results is an interesting problem to address. Investigating this new filter for the noisy metric matching will remain as future work.

Figure 5 also explains why CPDP-CM did not show reasonable prediction performance. Although the matched metrics are same as *LOC*, its bug-prone tendency is inconsistent. Thus, this matching using the common metric was noisy and was not helpful for building a prediction model.

We also investigated whether the performance of HDP can be affected by the size of a target dataset. If the size of a target dataset gets smaller, it might be difficult to precisely compute distribution similarity between source and target metrics. As in Table 1, the sizes of datasets vary from 36 (ar5) and 1862 (ML), and the results of HDP in Table 4 also vary. We could not find any relation between the size of a target and the prediction performance. In ar5 of Table 4, 14 out of 18 predictions led to Win results, while in ML, all six predictions led to Loss results. In velocity-1.4 that has a relatively small number (196) of instances compared to other datasets, all three predictions had Loss results. However, tomcat that is of a bigger size (858) than velocity-1.4 had only Win results. As discussed, the prediction performance of HDP is dependent on the bug-prone tendencies of the matched metrics. If the matched metrics are consistent in terms of the bug-prone tendency, HDP can lead to promising results regardless of the size of the target dataset.

6.2 Performance in Different Metric Selections

Table 5 shows prediction results on various metric selection approaches including with no metric selection ('None'). We compare the median AUCs of the HDP results by KSAnalyzer with the cutoff of 0.05 to those of WPDP, CPDP-CM, or CPDP-IFS, and report % of Win results.

Overall, we could observe metric selection to be helpful in improving prediction models in terms of AUC. When applying metric selection, the Win results account for more than about 63% in most cases against WPDP and CPDP-CM. Against CPDP-IFS, the Win results of HDP account for more than 80% after appying the metric selection approaches. This implies that the metric selection approaches can remove irrelevant metrics to build a better prediction model. In addition, this result confirms the previous studies that we can build prediction models better than or comparable to WPDP models with even a few key metrics [10, 17]. However, the percentages of Win results in 'None' were lower than those in applying metric selection. Among metric selection approaches, 'Chi-Square' and 'Significance' based approaches lead to the best performance in terms of the percentage of the Win results (64.7%-66.2%) against WPDP.

Table 6: Prediction performance in other analyzers with the matching score cutoffs, 0.05 and 0.90. (Anz=Analyzer, TgtCov=Target coverage)

Anz	Cut off	Against						HDP	Tgt Cov
		WPDP		CPDP-CM		CPDP-IFS			
		AUC	Win%	AUC	Win%	AUC	Win%	AUC	
P	0.05	**0.684**	30.3%	0.640	45.2%	0.511	54.5%	0.617*	100%
P	0.90	0.657	54.2%	0.622	65.1%	0.535	78.3%	0.692*	96%
KS	0.05	0.657	66.2%	0.636	66.2%	0.553	82.4%	**0.724***	100%
KS	0.90	0.657	100%	0.761	71.4%	0.624	100.0%	**0.852***	21%
SCo	0.05	**0.684**	28.5%	0.640	37.3%	0.511	46.3%	0.542*	100%
SCo	0.90	**0.684**	29.0%	0.639	36.6%	0.511	48.4%	0.547	100%

6.3 Performance in Various Analyzers

In Table 6, we compare the prediction performance in other analyzers with the matching score cutoff thresholds, 0.05 and 0.90. HDP's prediction results by PAnalyzer, with a cutoff of 0.90, are comparable to WPDP. This implies that comparing 9 percentiles between source and target metrics can evaluate the similarity of them well with a threshold of 0.90. However, PAnalyzer with the cutoff of 0.90 did not achieve 100% target coverage and is too simple an approach to lead to better prediction performance than KSAnalyzer. In KSAnalyzer with a cutoff of 0.05, the AUC (0.724) outperforms it (0.657) in WPDP with statistical significance.

HDP by KSAnalyzer with a cutoff of 0.90 could lead to significant improvement in the AUC value (0.852) compared to that (0.724) with the cutoff of 0.05. However, the target coverage is just 21%. This is because some prediction combinations are automatically filtered out since poorly matched metrics, whose matching score is not greater than the cutoff, are ignored. In other words, defect prediction for 79% of targets was not conducted since the matching scores of matched metrics in prediction combinations for the targets are not greater than 0.90 so that all matched metrics in the combinations were ignored.

An interesting observation in PAnalyzer and KSAnalyzer is that AUC values of HDP by those analyzers improved when a cutoff threshold increased. As the cutoff threshold increased as 0.05, 0.10, 0.20,..., and 0.90, we observed prediction results by PAnalyzer and KSAnalyzer gradually improved from 0.617 to 0.692 and 0.724 to 0.852 in AUC, respectively. This means these two analyzers can filter out negative prediction combinations well. As a result, the percentage of Win results are also significantly increased.

Results by SCoAnalyzer were worse than WPDP results. In addition, prediction performance rarely changed regardless of cutoff thresholds; results by SCoAnalyzer in different cutoffs from 0.05 to 0.90 did not vary as well. A possible

Table 7: Win/Tie percentages of HDP by KSAnalyzer (cutoff=0.05) against WPDP, CPDP-CM, and CPDP-IFS by different machine learners.

HDP Learners	Against					
	WPDP		CPDP-CM		CPDP-IFS	
	Win	Tie	Win	Tie	Win	Tie
Logistic	66.2%	1.4%	66.2%	6.3%	82.0%	2.7%
RandomForest	10.4%	2.3%	42.3%	1.4%	65.8%	2.2%
BayesNet	34.7%	4.1%	45.9%	2.7%	66.2%	2.7%
SVM	24.3%	23.0%	27.5%	0.0%	32.9%	14.9%
J48	30.2%	11.7%	32.4%	1.4%	41.9%	12.1%
SimpleLogistic	45.5%	2.7%	69.4%	6.8%	84.2%	3.2%
LMT	42.8%	3.2%	64.4%	6.3%	79.7%	3.2%

reason is that SCoAnalyzer does not directly compare the distribution between source and target metrics. This result implies that the similarity of distribution between source and target metrics is a very important factor for building a better prediction model.

6.4 Performance in Various Learners

To investigate if HDP works with other machine learners, we built HDP models (KSAnalyzer and the cutoff of 0.05) with various learners used in defect prediction literature such as Random Forest, BayesNet, SVM, J48 Decision Tree, Simple Logistic, and Logistic Model Trees (LMT) [8, 11, 24, 25, 37, 48]. Table 7 shows Win/Tie results.

Logistic regression (Logistic) led to the best results among various learners. The Logistic regression models works well when there is a linear relationship between a predictor variable (a metric) and the logit transformation of the outcome variable (bug-proneness) [3]. In our study, this linear relationship is related to the bug-prone tendency of a metric, that is, a higher complexity causes more bug-proneness [8, 36, 42]. As the consistent bug-prone tendency of matched metrics is important in HDP, the HDP models built by Logistic regression can lead to the best prediction performance.

HDP models built by other learners such as Simple logistic and LMT led to comparable results to Logistic regression against CPDP-CM and CPDP-IFS. Against Baseline2, Win results (69.4% and 64.4%) in Simple Logistic and LMT are comparable to Win results (66.2%) in Logistic. Simple logistic also uses the logit function and LMT adopts Logistic regression at the leaves of decision tree [11]. In other words, both learners consider the linear relationship like Logistic regression [3]. In our experimental settings, HDP tends to work well with the learners based on the linear relationship between a metric and a label (bug-proneness).

6.5 Practical Guidelines for HDP

We proposed the HDP models to enable defect prediction on software projects by using training datasets from other projects even with heterogeneous metric sets. When we have training datasets in the same project or in other projects with the same metric set, we can simply conduct WPDP or CPDP using recently proposed CPDP techniques respectively [4, 29, 37, 39, 44]. However, in practice, it might be that no training datasets for both WPDP and CPDP exist. In this case, we can apply the HDP approach.

In Section 5 and Table 6, we confirm that the overall result from HDP by KSAnalyzer with the cutoff of 0.05 outperforms the WPDP and shows 100% target coverage. Since KSAnalyzer can match similar source and target metrics, we guide the use of KSAnalyzer for HDP. In terms of the matching score cutoff threshold, there is a trade-off between prediction performance and target coverage. Since a cutoff of 0.05 that is the widely used level of statistical significance [7], we can conduct HDP using KSAnalyzer with the cutoff of 0.05. However, we observe some Loss results in our empirical study. To minimize the percentage of Loss results, we can sacrifice the target coverage by increasing the cutoff as Table 6 shows KSAnalyzer with the cutoff of 0.90 led to 100% Win results in feasible predictions against WPDP.

6.6 Threats to Validity

We cannot generalize our conclusions since 28 datasets for the experiments may not be representative. However, we tried to select various datasets used in papers published in top software engineering venues [8, 25, 37, 40, 51, 56]. In addition, we used datasets from both open-source (AEEEM [8, 37], ReLink [56], and MORPH [40]) and proprietary projects (NASA [25, 51] and SOFTLAB [51]).

We evaluated our HDP models in AUC. AUC is known as a good measure for comparing different prediction models [12, 25, 43, 48]. However, validating prediction models in terms of both precision and recall is also required in practice. To fairly compare WPDP and HDP models in precision and recall, we need to identify a proper threshold of prediction probability. Identifying the proper threshold is a challenging issue and remains as future work.

We computed matching scores using all source and target instances for each prediction combination. However, with the matching scores, we tested prediction models on a test set from the 50:50 random splits because of the WPDP models as explained in Section 4.5. To conduct WPDP with all instances of a project dataset as a test set, we need a training dataset from the previous release of the same project. However, the training dataset is not available for our subjects. Instead of using the 50:50 random splits, we additionally conducted experiments about HDP against CPDP-CM and CPDP-IFS by testing a prediction model on a test set using all instances of a target dataset in each prediction combination. In this setting, the median AUC of HDP was 0.723 and the median AUCs of the corresponding results in CPDP-CM and CPDP-IFS were 0.636 and 0.555 respectively. These results are similar to those from the 50:50 random splits in Section 5. In other words, as long as the bug-prone tendency of matched metrics is consistent, HDP yields promising results.

7. CONCLUSION

Cross-project defect prediction cannot be conducted across projects with heterogeneous metric sets. To address this limitation, we proposed heterogeneous defect prediction (HDP) based on metric matching using statistical analysis [30]. Our empirical evaluation showed the proposed HDP models are feasible and yield promising results.

HDP is very promising as it permits potentially all heterogeneous datasets of software projects to be used for defect prediction on new projects or projects lacking in defect data. In addition, it may not be limited to defect prediction. This technique can potentially be applicable to all prediction and recommendation based approaches for software engineering problems. As future work, we will explore the feasibility of building various prediction and recommendation models using heterogeneous datasets.

8. REFERENCES

[1] A. Arcuri and L. Briand. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 1–10, New York, NY, USA, 2011. ACM.

[2] V. R. Basili, L. C. Briand, and W. L. Melo. A validation of object-oriented design metrics as quality indicators. *IEEE Trans. Softw. Eng.*, 22:751–761, October 1996.

[3] J. Bruin. newtest: command to compute new test, http://www.ats.ucla.edu/stat/stata/ado/analysis/, Feb. 2011.

[4] G. Canfora, A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella. Multi-objective cross-project defect prediction. In *Software Testing, Verification and Validation, 2013 IEEE Sixth International Conference on*, March 2013.

[5] C. Catal and B. Diri. Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. *Information Sciences*, 179(8):1040 – 1058, 2009.

[6] S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Trans. Softw. Eng.*, 20:476–493, June 1994.

[7] G. W. Corder and D. I. Foreman. *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. New Jersey: Wiley, 2009.

[8] M. D'Ambros, M. Lanza, and R. Robbes. Evaluating defect prediction approaches: a benchmark and an extensive comparison. *Empirical Software Engineering*, 17(4-5):531–577, 2012.

[9] T. Fukushima, Y. Kamei, S. McIntosh, K. Yamashita, and N. Ubayashi. An empirical study of just-in-time defect prediction using cross-project models. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 172–181, New York, NY, USA, 2014. ACM.

[10] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya. Choosing software metrics for defect prediction: An investigation on feature selection techniques. *Softw. Pract. Exper.*, 41(5):579–606, Apr. 2011.

[11] B. Ghotra, S. McIntosh, and A. E. Hassan. Revisiting the impact of classification techniques on the performance of defect prediction models. In *Proc. of the 37th Int'l Conf. on Software Engineering (ICSE)*, ICSE '15, pages 789–800, 2015.

[12] E. Giger, M. D'Ambros, M. Pinzger, and H. C. Gall. Method-level bug prediction. In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 171–180, New York, NY, USA, 2012. ACM.

[13] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, Mar. 2003.

[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18, November 2009.

[15] M. Hall and G. Holmes. Benchmarking attribute selection techniques for discrete class data mining. *Knowledge and Data Engineering, IEEE Transactions on*, 15(6):1437–1447, Nov 2003.

[16] M. H. Halstead. *Elements of Software Science (Operating and Programming Systems Series)*. Elsevier Science Inc., New York, NY, USA, 1977.

[17] P. He, B. Li, X. Liu, J. Chen, and Y. Ma. An empirical study on software defect prediction with a simplified metric set. *Information and Software Technology*, 59(0):170 – 190, 2015.

[18] P. He, B. Li, and Y. Ma. Towards cross-project defect prediction with imbalanced feature sets. *CoRR*, abs/1411.4228, 2014.

[19] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang. An investigation on the feasibility of cross-project defect prediction. *Automated Software Engineering*, 19(2):167–199, 2012.

[20] Y. Kamei, E. Shihab, B. Adams, A. Hassan, A. Mockus, A. Sinha, and N. Ubayashi. A large-scale empirical study of just-in-time quality assurance. *Software Engineering, IEEE Transactions on*, 39(6):757–773, June 2013.

[21] D. Kim, Y. Tao, S. Kim, and A. Zeller. Where should we fix this bug? a two-phase recommendation model. *Software Engineering, IEEE Transactions on*, 39(11):1597–1610, Nov 2013.

[22] M. Kläs, F. Elberzhager, J. Münch, K. Hartjes, and O. von Graevemeyer. Transparent combination of expert and measurement data for defect prediction: an industrial case study. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, pages 119–128, New York, NY, USA, 2010. ACM.

[23] E. Kocaguneli, T. Menzies, J. Keung, D. Cok, and R. Madachy. Active learning and effort estimation: Finding the essential content of software effort estimation data. *Software Engineering, IEEE Transactions on*, 39(8):1040–1053, 2013.

[24] T. Lee, J. Nam, D. Han, S. Kim, and I. P. Hoh. Micro interaction metrics for defect prediction. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, 2011.

[25] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *Software Engineering, IEEE Transactions on*, 34(4):485–496, 2008.

[26] M. Li, H. Zhang, R. Wu, and Z.-H. Zhou. Sample-based software defect prediction with active and semi-supervised learning. *Automated Software Engineering*, 19(2):201–230, 2012.

[27] H. W. Lilliefors. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 62(318):pp. 399–402, 1967.

[28] H. Liu, J. Li, and L. Wong. A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome Informatics*, 13:51–60, 2002.

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

[30] F. J. Massey. The kolmogorov-smirnov test for

goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

[31] J. Matouek and B. Gärtner. *Understanding and Using Linear Programming (Universitext)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[32] T. McCabe. A complexity measure. *Software Engineering, IEEE Transactions on*, SE-2(4):308–320, Dec 1976.

[33] T. Mende. Replication of defect prediction studies: Problems, pitfalls and recommendations. In *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, pages 5:1–5:10, New York, NY, USA, 2010. ACM.

[34] A. Meneely, L. Williams, W. Snipes, and J. Osborne. Predicting failures with developer networks and social network analysis. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 13–23, 2008.

[35] T. Menzies, B. Caglayan, Z. He, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan. The promise repository of empirical software engineering data, June 2012.

[36] T. Menzies, J. Greenwald, and A. Frank. Data mining static code attributes to learn defect predictors. *IEEE Trans. Softw. Eng.*, 33:2–13, January 2007.

[37] J. Nam, S. J. Pan, and S. Kim. Transfer defect learning. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 382–391, Piscataway, NJ, USA, 2013. IEEE Press.

[38] T. Ostrand, E. Weyuker, and R. Bell. Predicting the location and number of faults in large software systems. *Software Engineering, IEEE Transactions on*, 31(4):340–355, April 2005.

[39] A. Panichella, R. Oliveto, and A. De Lucia. Cross-project defect prediction models: L'union fait la force. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on*, pages 164–173, Feb 2014.

[40] F. Peters and T. Menzies. Privacy and utility for defect prediction: experiments with morph. In *Proceedings of the 2012 International Conference on Software Engineering*, pages 189–199, Piscataway, NJ, USA, 2012. IEEE Press.

[41] M. Pinzger, N. Nagappan, and B. Murphy. Can developer-module networks predict failures? In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 2–12, New York, NY, USA, 2008. ACM.

[42] F. Rahman and P. Devanbu. How, and why, process metrics are better. In *Proceedings of the 2013 International Conference on Software Engineering*, Piscataway, NJ, USA, 2013. IEEE Press.

[43] F. Rahman, D. Posnett, and P. Devanbu. Recalling the "imprecision" of cross-project defect prediction. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, New York, NY, USA, 2012. ACM.

[44] D. Ryu, O. Choi, and J. Baik. Value-cognitive boosting with a support vector machine for cross-project defect prediction. *Empirical Software Engineering*, pages 1–29, 2014.

[45] M. Shepperd, Q. Song, Z. Sun, and C. Mair. Data quality: Some comments on the nasa software defect datasets. *Software Engineering, IEEE Transactions on*, 39(9):1208–1215, Sept 2013.

[46] E. Shihab, A. Mockus, Y. Kamei, B. Adams, and A. E. Hassan. High-impact defects: a study of breakage and surprise defects. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 300–310, New York, NY, USA, 2011. ACM.

[47] S. Shivaji, E. J. Whitehead, R. Akella, and S. Kim. Reducing features to improve code change-based bug prediction. *IEEE Transactions on Software Engineering*, 39(4):552–569, 2013.

[48] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu. A general software defect-proneness prediction framework. *Software Engineering, IEEE Transactions on*, 37(3):356–370, 2011.

[49] C. Spearman. The proof and measurement of association between two things. *International Journal of Epidemiology*, 39(5):1137–1150, 2010.

[50] B. Turhan. On the dataset shift problem in software engineering prediction models. *Empirical Software Engineering*, 17(1-2):62–74, 2012.

[51] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano. On the relative value of cross-company and within-company data for defect prediction. *Empirical Softw. Eng.*, 14:540–578, October 2009.

[52] Understand 2.0. http://www.scitools.com/products/.

[53] G. Valentini and T. G. Dietterich. Low bias bagged support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 752–759. AAAI Press, 2003.

[54] S. Watanabe, H. Kaiya, and K. Kaijiri. Adapting a fault prediction model to allow inter languagereuse. In *Proceedings of the 4th International Workshop on Predictor Models in Software Engineering*, pages 19–24, New York, NY, USA, 2008. ACM.

[55] F. Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, Dec. 1945.

[56] R. Wu, H. Zhang, S. Kim, and S. Cheung. Relink: Recovering links between bugs and changes. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, 2011.

[57] F. Zhang, A. Mockus, I. Keivanloo, and Y. Zou. Towards building a universal defect prediction model. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 182–191, New York, NY, USA, 2014. ACM.

[58] T. Zimmermann and N. Nagappan. Predicting defects using network analysis on dependency graphs. In *Proceedings of the 30th international conference on Software engineering*, pages 531–540, 2008.

[59] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 91–100, New York, NY, USA, 2009. ACM.