

# A recommendation system of training data selection method for cross-project defect prediction

Benyamin Langgu Sinaga<sup>1,2</sup>, Sabrina Ahmad<sup>1</sup>, Zuraida Abal Abas<sup>1</sup>, Intan Ermahani A. Jalil<sup>1</sup>

<sup>1</sup>Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka, Durian Tunggal, Malaysia

<sup>2</sup>Department of Informatics, Universitas Atma Jaya Yogyakarta, Yogyakarta, Indonesia

---

## Article Info

### Article history:

Received Apr 12, 2022

Revised Jun 6, 2022

Accepted Jun 20, 2022

---

### Keywords:

Cross-project defect prediction

Meta-learning

Recommendation system

Training data selection

---

## ABSTRACT

Cross-project defect prediction (CPDP) has been a popular approach to address the limited historical dataset when building a defect prediction model. Directly applying cross-project datasets to learn the prediction model produces an unsatisfactory predictive model. Therefore, the selection of training data is essential. Many studies have examined the effectiveness of training data selection methods, and the best-performing method varied across datasets. While no method consistently outperformed the others across all datasets, predicting the best method for a specific dataset is essential. This study proposed a recommendation system to select the most suitable training data selection method in the CPDP setting. We evaluated the proposed system using 44 datasets, 13 training data selection methods, and six classification algorithms. The findings concluded that the recommendation system effectively recommends the best method to select training data.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



---

## Corresponding Author:

Sabrina Ahmad

Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka

Durian Tunggal, Melaka, Malaysia

Email: sabrinaahmad@utem.edu.my

---

## 1. INTRODUCTION

Cross-project defect prediction (CPDP) is a well-studied topic in software defect prediction (SDP) studies. CPDP has become a popular strategy to deal with the lack of historical datasets. CPDP builds an SDP model using the dataset from the other project and predicts defects in the target project [1]. However, CPDP has its drawbacks since merely using cross-project datasets to learn a prediction model produces a model with unsatisfactory predictive performance. It is due to the divergence in data distribution between training and testing datasets [1]–[3].

Several approaches have been proposed to address this problem, such as data transformation [3]–[5], data normalization [3], and training data selection. Data transformation and normalization use all training instances to train an SDP model, possibly containing irrelevant and noisy data. Zhang *et al.* [6] found that choosing a suitable transformation for a particular pair of training and testing instances is open to question. Prior studies [3], [7] show that the effect of the transformation on the model performance varies on the same dataset. Meanwhile, studies [2], [8], [9] reported that the prediction model developed using selected cross-project data has a satisfied predictive performance. Therefore, selecting the relevant data for training an SDP model becomes an important and challenging task [9].

Training data selection is a method for selecting appropriate training instances based on the target instances. Correct identification of suitable training data is critical, as using irrelevant training instances can detrimentally impact the performance of an SDP model. Recent studies have proposed numerous methods for

filtering training data classified into several groups (i.e., instance, project, and multi-granularity) based on training data granularity [10]. The researchers compared the effect of using training data selection on defect prediction performance [11]–[13]. However, they found that no single training data selection method provides optimal results on all data selection problems. This result is caused by different datasets, classification algorithms, and performance metrics used in each study. As a result, a method performs better on a particular dataset but worse on the other datasets. The development of a new method does not necessarily guarantee better performance. In addition, applying all training data selection methods to all problems to identify the optimal method is impractical. It raises the issue of whether we can predict an optimal training data selection method for a given dataset.

To answer this question, we proposed an automatic recommendation system using meta-learning to select the optimal training data selection method for a particular dataset. Bradzil *et al.* [14] stated that meta-learning relates the characteristics of a problem to the relative performance of algorithms, which enables us to select the optimal method for a given dataset. However, no meta-learning approach has been implemented in selecting training data methods. The proposed system is an instance-based meta-learning that works according to the similarity of training and target datasets.

This study contributes in three ways. Firstly, we proposed a recommender for selecting proper training data selection methods, validated with thirteen methods. To the best of our knowledge, it is the first experiment in the defect prediction area to address the training data selection method. Secondly, we proposed data-related statistical meta-features suitable for choosing a method to select relevant cross-project training data. Finally, we compared the effectiveness of various sets of unsupervised meta-features for training data selection meta-learning. This paper is structured as follows: section 2 summarizes the related works, section 3 describes the proposed recommendation system and the experimental settings, section 4 reports and discusses the results, and section 5 draws on the conclusion.

## 2. RELATED WORKS

### 2.1. Training data selection

Training data selection refers to the process of selecting suitable training instances based on the target instance. The selection of appropriate training data is critical, as using improper training instances can harm the performance of an SDP model. Although selecting training data eliminates some source instances, the model trained with the selected training data performs as expected. Numerous methods for filtering training data have been studied and classified into several groups (i.e., instance, project, and multi-granularity) based on training data granularity [10].

At the instance-level selection, several studies proposed distance-based [8], [15], [16] and cluster-based filters [17], [18]. The distance-based filter utilized the nearest-neighbor algorithm to choose the relevant training data. Peters *et al.* [15] used source instances to guide the selection, while Turhan *et al.* [8] and He *et al.* [16] employed testing instances to select the training data. He *et al.* [16] introduced a method that uses the similarity of training and target instances and the defect number in each training instance to perform the selection. Meanwhile, Kawata *et al.* [17] and Menzies *et al.* [18] utilized a cluster-based technique for selecting training data.

Other studies investigated the effectiveness of selecting training data at the project level [9], [19]. Herbold [9] used the kNN algorithm and distributional characteristics (i.e., standard deviation and mean) to select the relevant source data based on the Euclidean distance between distributional characteristic vectors. Unlike Herbold, He *et al.* [19] used performance accuracy to calculate the distance between the training and testing datasets. Selected training data are instances from datasets that exceed a pre-defined cutoff in terms of accuracy.

Studies [20], [21] introduced multi-granularity approaches that integrate project-level and instance-level methods. At the project level, they used Herbold's method to filter training data. Meanwhile, at the instance level, He *et al.* [20] proposed two strategies that use the Burak filter [8] and Peter filter [15], while Li *et al.* [21] used K-Means to create clusters of similar instances.

Several comparative studies have evaluated the effect of training data selection on defect prediction performance [11], [13], [21]. They discovered inconsistencies in the study findings. It could be due to differences in experimental settings, such as datasets, classifiers, or levels of analysis. Previous study [12] found that the optimal method for each dataset varies. A method performs better on one dataset but not on others. As a result, we cannot find the best training data selection method across all datasets. In addition, the development of a new method does not necessarily guarantee better performance across datasets. Thus, this research focuses on predicting the optimal method for selecting training data for a particular dataset.

## 2.2. Meta-learning

Many methods for selecting training data have emerged from research on software defect prediction. Such methods may be ideal for some problems but not for others since the method might have a different bias and is dependent on the problem. No single method works for all problems. Choosing a suitable training data selection method for a given dataset becomes an intriguing issue. Method selection is difficult because machine learning algorithms have many hyperparameters, resulting in massive configurations [22]. Due to time constraints, it is impossible to run every possible combination of algorithms and configurations. Hence, predicting the suitable method with performance similar to the best method in a reasonable time is the rational solution. It is the goal of meta-learning, a popular solution to algorithm selection issues [14].

Several studies in software defect prediction have investigated the use of meta-learning [23]–[26]. Studies [23]–[25] studied meta-learning to select a suitable method for the within-project defect prediction (WPDP) approach. WPDP uses training and testing datasets from the same project. Both datasets have target labels, enabling the extraction of the meta feature in a supervised approach. Dores *et al.* [23] and Nucci *et al.* [24] proposed a recommendation system to select suitable classifiers for a particular dataset. Sun *et al.* [25] investigated a recommender for selecting a resampling algorithm. Later, Porto *et al.* [26] proposed a meta-learning to select the most suitable transfer learning algorithm in the CPDP setting. They used two sets of meta-features and the random forest as the meta-learning algorithm. However, since meta-learning usually has a limited number of meta-examples, the use of the random forest as the meta-learner may be problematic since the random forest is challenging with a small number of instances [27]. Our study also used a CPDP approach, but we focused on recommending the training data selection method and used kNN as the meta-learning algorithm. Table 1 summarizes the different points of the related meta-learning studies in the software defect prediction area.

Table 1. Summary of related meta-learning studies on software defect prediction

Ref	Goal of recommendation	Datasets	Approach	Meta learner	#set of metafeatures	# compared methods
[23]	Classifier	71 (PROMISE)	WPDP	RF, Ensemble	2	7 classifiers
[24]	Classifier	30 (PROMISE)	WPDP	RF	-	5 classifiers
[26]	Transfer learning method	47 (PROMISE)	CPDP	RF	2	6 methods
[25]	Imbalance handling method	20 (PROMISE)	WPDP	kNN	1	12 methods
Our study	Training data selection method	44 (PROMISE)	CPDP	kNN	5	13 methods

Within the CPDP approach, it is assumed that target instances are unlabeled. Thus, the traditional characterization of datasets (supervised meta-feature) is not applicable. Instead, unsupervised meta-features are used, the extraction of which is independent of the target label. Since there is limited research on unsupervised meta-features in software defect prediction, we proposed a small set of unsupervised meta-features based on: i) statistical characteristics of the used datasets and ii) issues related to CPDP data.

## 2.3. Meta-features for CPDP

The main challenge in developing a meta-learning system is choosing the right meta-features. Castiello *et al.* [28] noted that the quality of meta-features considerably influences meta-learner effectiveness; hence, identifying the appropriate meta-features is critical for meta-learning [29]. In this study, we proposed a small set of unsupervised meta-features based on i) statistical characteristics of the used datasets and ii) issues related to CPDP data. Hosseini *et al.* [30] concluded that there are several data-related issues in cross-project defect prediction, i.e., class imbalance, redundant and multi-collinearity among features, and noise in data. We did not propose meta-features related to class imbalance since the class information is needed, whereas, in the CPDP setting, the target datasets are unlabeled, meaning that the class information is unavailable. Hence, we deemed only the last two issues, i.e., redundant and multi-collinearity among features and noise in data.

### 2.3.1. Redundant and multi-collinearity among features

We investigated the extent to which cross-project datasets have redundant features and the correlation among features. We used Gothra *et al.*'s idea [31] to extract features from datasets using principal component analysis (PCA), accounting for 95% of the variance of the datasets. We then compared the PCA dimensions to the dimensions of the original datasets. Figure 1(a) displays the ratio between PCA dimensions and original dataset dimensions (i.e., PCA\_ratio).

We observed that the ratio between the dimension of PCA to that of original datasets is small for training datasets. The lower the value, the fewer original features are required to characterize the variability of the original datasets. A small value implies that the original datasets have redundant features in the original datasets. According to Lorena *et al.* [32], PCA dimensions could be viewed as an estimate of the dimensionality of a dataset after the correlation between features is reduced. It indirectly indicates that the used training datasets may have a correlation between the pairs of features. Based on this analysis, we proposed to use three meta-features, i.e., correlation and covariance among a pair of attributes and the number of highly correlated attributes, described as in Table 2.

**2.3.2. Noise in data**

We examined the degree to which our multi-source cross-project datasets contain noise. A dataset is considered noisy if it has an instance of data with a high deviation from the normal range of the feature space (i.e., outlier) [30] or it contains redundant or inconsistent instances [33]. We also explored the skewness and kurtosis of the used datasets, Figure 1(b). It can be observed from Figure 1(a) that 80 percent of the attributes of the used datasets contain outliers. In addition, 20 percent of the instances are redundant in each dataset. If we see the second y-axis, the multi-source cross-project datasets have high skewness and kurtosis within the range [6.5–8.5] and [120–170], respectively.

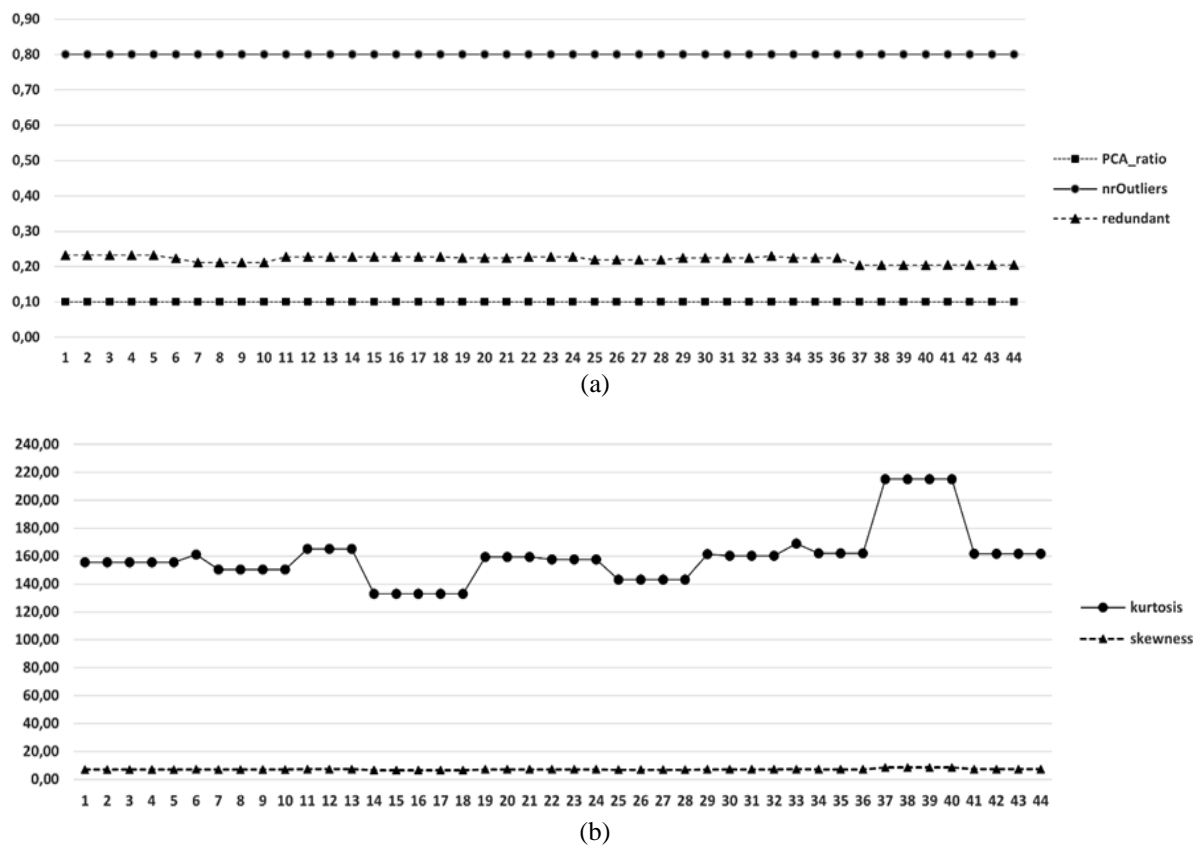


Figure 1. Several data-related issues in multi-source cross-project dataset used in the experiment (a) redundant and multi-collinearity among features and (b) skewness and kurtosis

Meanwhile, De Carlo [34] stated that kurtosis could be used to detect the presence of an outlier. According to Davies [35], kurtosis is a measure of outlier potential, with a high value indicating many outliers and a low value indicating the opposite. Davies also pointed out that skewness describes the preference for many or few outliers in regions less than or greater than the mean. Based on this evaluation, we proposed skewness and kurtosis as the meta-features. We prefer to propose a small number of meta-features since we used kNN as the meta-learner. The large meta-features are deemed to affect the kNN algorithm negatively [36]. We refer to our proposed meta-features as SStats meta-features.

Table 2. The proposed SStats meta-features

Meta-feature	Description	How to compute	Ref
cor	cor measures the mean of correlation between a pair of attributes	$cor_{x,y} = \frac{cov_{x,y}}{sd_x sd_y}$ , where $cov_{x,y} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{n-1}$ , and $sd_x = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}}$	[29], [37]
cov	cov measures the mean of covariance between a pair of attributes	$cov_{x,y} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{n-1}$ ,	[29], [37]
nrCorAttr	nrCorAttr represents the proportion of highly correlated attributes.	$nrCorAttr = \frac{2}{d(d-1)} \sum_{i=1}^{d-1} \sum_{j=i+1}^d \mathbf{1}( cor_{x,y}  \geq \tau)$ , where $\tau$ is a threshold within [0..1], usually $\tau = 0.5$ , $x$ and $y$ are numerical attributes.	[29], [38]
kurtosis	kurtosis measures the mean of absolute kurtosis	$kurtosis_{x_i} = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^4}{sd_x^4} - 3$	[28], [29], [37]
skewness	skewness measures the mean of absolute skewness	$skewness_{x_i} = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^3}{sd_x^3}$	[28], [29], [37]

We also utilized several sets of meta-features to find the most effective meta-features for the proposed recommendation system, i.e., distributional characteristics [2], distance [39], correlation [40] and itemset [41]. Distributional characteristics meta-features: We used distributional characteristics meta-features in this research. Several defect prediction studies have used distributional characteristics. We calculate the min, mean, max, mode, and median of 16 data characteristics from He *et al.* [2]. Thus, we have 80 meta-features. However, we did not use all 80 meta-features. Instead, we performed the meta-feature selection, and the results are used as the meta-features in this experiment. We refer to these meta-features as DC meta-features.

Distance-based meta-features: Ferrari and Castro [39] proposed distance-based meta-features for clustering problems. They calculate dissimilarity between two instances using the Euclidean distance and then create a dissimilarity vector among instances. The vector values are normalized into the interval [0, 1]. From this vector, 19 meta-features are extracted, which are referred to as Distance meta-features (please refer to [39] for detail).

Correlation meta-features: Pimentel and Carvalho [40] improved distance meta-features by combining dissimilarity and correlation among instances. First, the dissimilarity between two instances is calculated following [39]. Next, the Spearman rank correlation coefficient between each instance is computed. Later, dissimilarity and correlation among instances are combined into a single vector. Ferrari and Castro [39], 19 meta-features are extracted from the vector after the values are normalized into the interval [0, 1]. We refer to these meta-features as Correlation meta-features.

Structural meta-features: Song *et al.* [41] proposed meta-features used to develop a classification algorithm recommendation system. For a given dataset, its attribute values are changed to the corresponding binary dataset. After that, two attribute vectors are retrieved from the binary dataset, i.e., a one-item vector indicating the distribution of values for each feature and a two-item vector representing the correlation between two features. These vectors are then concatenated into one vector and sorted in ascending order. Finally, the meta-features are obtained by calculating the minimum, maximum, and seven octiles of the concatenated vector. We refer to these meta-features as Itemset meta-features.

### 3. RESEARCH METHOD

#### 3.1. Proposed framework

Recent studies on SDP found that using suitable training data can improve prediction performance [8], [9], [15]–[17], [21]. They also reported that no single training data selection method performs consistently well with various classifiers on all datasets. Also, each method has different predictive performance across datasets. Thus, selecting the best training data selection method for a given dataset becomes an interesting issue.

Previous research on meta-learning found that dataset characteristics are ties to classification algorithm performance [25], [41]–[44]. Thus, if datasets have identical characteristics, classification algorithms applied to the datasets tend to perform similarly. In other words, the data characteristics of training data tie closely to the prediction results, which corroborates the finding of He *et al.* [2].

Based on the results discussed, we proposed a meta-learning approach that uses instance-based learning to recommend the best training data selection methods in CPDP settings. The proposed approach estimates the rank of all training data selection methods on a particular dataset. Figure 2 displays the proposed recommendation framework that includes two parts: meta-database construction and method recommendation.

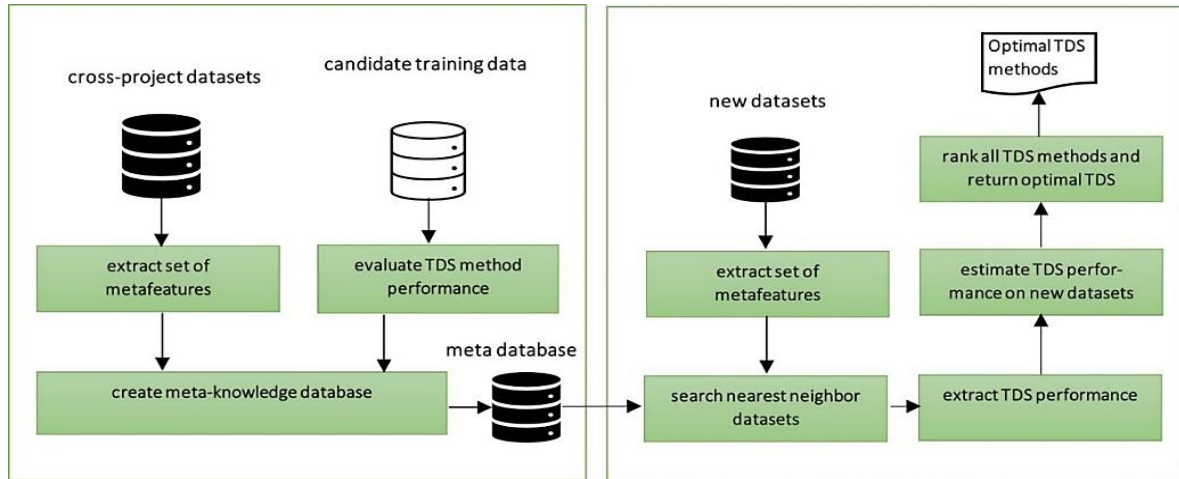


Figure 2. The framework of the recommendation of training data selection methods

### 3.2. Construction of meta database

The meta-database contains meta-target and meta-features. The meta-target represents prediction performance on a target dataset, while the meta-features represent the characteristics of training datasets. We applied each method to all training datasets and measured the performance (i.e., AUC). We used the success rate ratio (SRR) between training data selection methods as the meta-target [45]. This measure is better than the other ranking methods in the meta-learning comparative study [43].

The success rate ratio defines the relative advantage of a method over the others. We modify the formulation of the success rate ratio in [45] since we used AUC instead of error. When using the error as a performance measure, the smaller the error, the better is the method. In contrast, as for AUC, the bigger the AUC, the better is the method. Thus, the success rate of a method  $m_j$  over  $m_k$  for a datasets  $D_i$  as (1).

$$SRR_{m_j, m_k}^{D_i} = \frac{pv_{m_j}^{D_i}}{pv_{m_k}^{D_i}} \tag{1}$$

Where  $pv_{m_j}^{D_i}$  is the performance value (i.e., AUC) of method  $m_j$  on dataset  $D_i$ . Generally, the number of candidate methods is more than two. Therefore, multiple comparisons of training data selection must be performed. We then calculate the performance of a method  $m_j$  for a given dataset  $D_i$  as (2).

$$SRR_{m_j}^{D_i} = \frac{\sum_{k=1}^N SRR_{m_j, m_k}^{D_i}}{N-1} \tag{2}$$

where  $N$  is the number of candidate methods for training data selection.

After extracting meta-features and calculating the SRR, we create a meta-database that consists of meta-instances. Each meta-instance is formed by a tuple containing meta-features and SRR values. The steps to construct the meta-database are available in Procedure 1.

**Procedure 1. Create Metadatabase**

Input:                   ▪ Cross-project benchmark datasets  $\{D_1, D_2, \dots, D_i \mid 1 \leq i \leq n\}$

Output:                 meta-database

- 1     extract meta-features of each dataset  $D_i$ , using meta-feature described in Section 2.3.
- 2     evaluate the performance of each candidate of the training data selection method.
- 3     calculate the success rate ratio of each candidate method on every single dataset  $D_i$ , using eq. (2)
- 4     create meta-database

### 3.3. Recommendation of training data selection method

We used kNN as the meta-learner since it is widely used in meta-learning studies [41], [43], [46], [47] and it is effective and efficient [39], [43], [47], [48]. In addition, the number of meta-examples in meta-learning is relatively small. Therefore, it is challenging to induce a model, particularly using an algorithm

that generates a sharp threshold [43]. Moreover, the kNN allows us to select multiple suitable methods, i.e., multi-target prediction [14], as it might be possible for a particular dataset to have more than one optimal method.

When a target dataset  $D_{tar}$  comes, the meta-learning extracts the target meta-features and calculates the distance of all training datasets to the target dataset. It then selects the  $k$  nearest datasets to the target dataset. Identifying a training dataset similar to the target dataset  $D_{tar}$  is performed by computing the distance between both datasets using meta-features characterizing them. We used L1 distance to calculate the distance since it can measure similarity well between two datasets [43], [49]. A distance between two datasets  $D_i$  and  $D_j$ , is formulated as (3).

$$dist(D_i, D_j) = \sum_{k=1}^n |M_{i,k} - M_{j,k}| \quad (3)$$

Where  $M_{i,k}$  denotes the  $k^{th}$  value of meta-features of  $D_i$ ,  $n$  is meta-feature length. Since the range of meta-feature value may be quite different, we normalize all meta-feature values into the same range  $[0,1]$  to avoid the dominance of meta-feature having large values.

After identifying the most similar training datasets, we compute the SRR of all methods on the  $k$  nearest dataset. The performance of the candidate methods for the new target datasets is estimated by aggregating the SRR values of all identified candidate methods. We did not give the same contribution to all the nearest neighbor datasets. Instead, following the recommendation [43], we weigh the contribution of each nearest dataset differently based on its distance to the target dataset.

Let  $SRR_{i,m}$  be the performance measure of the  $m^{th}$  method on training dataset  $D_i$ . Each  $SRR_{i,m}$  is given a weight depending on the distance between  $D_i$  and the target dataset,  $D_{tar}$ . The performance of the  $m^{th}$  method on the target datasets  $D_{tar}$  denoted as  $SRR_{tar,m}$  is estimated as (4).

$$SRR_{tar,m} = \sum_{i=1}^k SRR_{i,m} w_i \quad (4)$$

Where  $k$  is the number of the nearest training datasets. A  $w_i$  represents the similarity of the dataset  $D_i$  to the target dataset  $D_{tar}$ .

A small distance represents a high similarity between dataset  $D_i$  and the target dataset  $D_{tar}$ . Therefore, adopting [49], we use the reciprocal distance to determine the weight of each training dataset. If  $dist(D_i, D_{tar})$  be the distance of dataset  $D_i$  to the target dataset  $D_{tar}$ , then  $1/dist(D_i, D_{tar})$  is the reciprocal distance of the  $D_i$  and  $D_{tar}$  and  $\sum_{i=1}^k 1/dist(D_i, D_{tar})$  is sum of reciprocal distance between  $k$  nearest neighbor datasets to the target dataset  $D_{tar}$ . The weight,  $w_i$ , is calculated as (5). Having calculated the performance of all methods on the target dataset, using equation 4, we rank all candidates of the training data selection method and pick the top three methods as the most suitable methods for a particular target dataset. The procedure of recommending the suitable methods is shown in Procedure 2.

$$w_i = \frac{1/dist(D_i, D_{tar})}{\sum_{i=1}^k 1/dist(D_i, D_{tar})} \quad (5)$$

Procedure 2. Get Recommended Methods

Input:   ▪ meta-database  $\{ \langle mf_{i,1}, mf_{i,2}, \dots, mf_{i,p}, mt_{i,1}, mt_{i,2}, \dots, mt_{i,q} \rangle \mid 1 \leq i \leq n \}$ , where  $mf$  and  $mt$  are the meta-features and meta-target of dataset  $D_i$ .  $p$  and  $q$  denote the number of meta-features and candidate methods, respectively.  
       ▪ the number of test datasets ( $n$ ); the number of nearest-neighbor datasets ( $k$ ); target dataset ( $D_{tar}$ ).

Output: recommended methods

```

1  extract meta-features of  $D_{tar}$ 
2  for  $i$  in 1 to  $n$  do
    calculate  $dist(D_i, D_{tar})$  based on the meta-features, using eq. (3)
    end for
3  select  $k$  nearest neighbor datasets
4  extract performance of all candidate methods on all  $k$  nearest datasets
5  for  $i$  in 1 to  $k$  do
    adjusted the performance of each method on the nearest dataset
     $D_i$ ,  $SRR_{i,m} = SRR_{i,m} w_i$ .  $w_i$  calculated using eq.(5).
    end for
6  estimate the performance of candidate methods on the target dataset by aggregating
    the performance of each method on all  $k$  nearest datasets, using eq. (4).
7  rank all candidate methods obtained in step 6
8  return the top three candidate methods.
```

### 3.4. Datasets

We experimented on 44 dataset releases widely used in defect prediction studies, i.e., PROMISE datasets. Table 3 presents the description of each release. The PROMISE dataset [50] contains several projects with various releases. Each release (version) has 20 independent attributes and a class attribute representing the defect number found in that release. Herbold [9] did not use proprietary datasets to avoid the influence of mixing open-source and proprietary datasets on the experimental results. He also eliminated small projects by choosing releases having at least 100 instances. Following his arguments, we selected 44 releases from 14 projects available from this repository.

### 3.5. Performance evaluation methods

We conducted a two-level evaluation of the proposed method. At the base level, we used strict cross-project defect prediction that utilizes multiple source datasets for training a model and a single dataset for testing the model. A dataset release is selected as the testing dataset for each cross-project defect prediction, while the remaining releases from other projects are used as candidate training datasets. We employed this approach since several training data selection methods at the project level [9], [20], [21] required multiple source training data. Using the experimental setup mentioned, we only need to execute the experiment once for each pair of training data selection and classifier [51], except for approaches with a random component (i.e., Menzies11, Peters13b, Peters15 and ZHe13). We undertook the experiment 10 times for such approaches and used each performance mean value for comparison with the other approaches. At the meta-level, we adopted the leave-one-out strategy to validate the recommendation performance. Each release or release serves as the test dataset in this strategy, while the remaining releases from the other projects serve as candidate training datasets.

Table 3. The description of benchmark datasets

No	release	#instances	#defective	%defective	No	release	#instances	#defective	%defective
1	ant-1.3	125	20	16%	23	lucene-2.2	247	144	58%
2	ant-1.4	178	40	22%	24	lucene-2.4	340	203	60%
3	ant-1.5	293	32	11%	25	poi-1.5	237	141	59%
4	ant-1.6	351	92	26%	26	poi-2.0	314	37	12%
5	ant-1.7	745	166	22%	27	poi-2.5	385	248	64%
6	arc	234	27	12%	28	poi-3.0	442	281	64%
7	camel-1.0	339	11	3%	29	redaktor	176	27	15%
8	camel-1.2	608	216	36%	30	synapse-1.0	157	16	10%
9	camel-1.4	872	145	17%	31	synapse-1.1	222	60	27%
10	camel-1.6	965	188	19%	32	synapse-1.2	256	86	34%
11	ivy-1.1	111	63	57%	33	tomcat	858	77	9%
12	ivy-1.4	241	16	7%	34	velocity-1.4	196	147	75%
13	ivy-2.0	352	40	11%	35	velocity-1.5	214	142	66%
14	jedit-3.2	272	90	33%	36	velocity-1.6	220	78	35%
15	jedit-4.0	306	75	25%	37	xalan-2.4	723	110	15%
16	jedit-4.1	312	79	25%	38	xalan-2.5	803	387	48%
17	jedit-4.2	367	48	13%	39	xalan-2.6	885	411	46%
18	jedit-4.3	492	11	2%	40	xalan-2.7	909	898	99%
19	log4j-1.0	135	34	25%	41	xerces-init	162	77	48%
20	log4j-1.1	109	37	34%	42	xerces-1.2	440	71	16%
21	log4j-1.2	205	189	92%	43	xerces-1.3	453	69	15%
22	lucene-2.0	195	91	47%	44	xerces-1.4	588	437	74%

### 3.6. Classification algorithm

We used six classifiers to assess the effectiveness of training data selection on defect prediction performance. We selected the classifiers based on [12], [52] and considered different "families" of previously successful classifiers in detecting defects [53], [54]. C4.5 (decision tree), LR (regression function), multi-layer perceptron (neural networks), Naive Bayes (probabilistic), random forest (ensemble method), and support vector machine were chosen as base classifiers. When experimenting, we used classifiers provided by WEKA with default parameters.

### 3.7. Performance measures

This experiment used two performance metrics, i.e., prediction performance and recommendation performance. The former indicates the suitability of training data selection methods for a given classification algorithm. Whereas the latter evaluates the proposed meta-learning approach. For the prediction performance, we used the AUC since the AUC is unaffected by the imbalanced class problem, is threshold-independent [22], and is widely used in SDP research [55]. For the recommendation performance, we used



the Spearman rank correlation coefficient (SRC), hit ratio (HR), and average recommendation accuracy (ARA).

Spearman rank correlation coefficient measures the ranking accuracy by evaluating the agreement between the rank of recommended methods and the optimal method on each dataset. SRC is calculated using (6) [56].

$$SRC = \frac{6 * \sum_{i=1}^n (r_i - i_i)^2}{n^3 - n} \quad (6)$$

Where  $i_i$  and  $r_i$  denote ideal and recommended ranking,  $n$  represents the number of candidate methods. A value of +1 denotes a perfect match, and a value of -1 denotes a complete disagreement between the two rankings. Several meta-learning studies [39], [40], [43], [57] used this metric.

Hit ratio evaluates the matching between the recommended method with the optimal method for each dataset. Let  $S_{opt, D}$  be a set of optimal methods whose classification performance falls within an optimal margin. Let  $S_{rec, D}$  be the recommended method for a particular dataset. If  $S_{rec, D}$  is the subset of  $S_{opt, D}$ , then hit count (Hit) is 1, indicating that the recommended method is deemed optimal. HR represents the proportion of total hit count to the number of recommendations (i.e., test datasets, denoted as  $N$ ), i.e.,

$$HR = \frac{\sum_{n=1}^N Hit_n}{N} * 100\% \quad (7)$$

We adopt [58], [59] to define the optimal margin relative to the performance of the best method on a dataset. It was also used in [57]. If an optimal range is defined as (8).

$$opt.range = \sqrt{pf_{best} (1 - pf_{best}) / N_T} \quad (8)$$

Then the acceptable optimal margin is set as (9).

$$opt.margin = [pf_{best} - k * opt.range, pf_{best}] \quad (9)$$

Where  $pf_{best}$  is the best classification performance (AUC) on a particular dataset,  $N_T$  is the size of the test dataset, and  $k$  is the margin size. Thus, each training data selection method with classification performance within this optimal margin is considered optimal. We set  $k$  to 3, corresponding to a 95 percent confidence level [58].

Recommendation accuracy measures the difference in performance between the recommended and the best method. Given that  $M_{rec, D}$ ,  $M_{best, D}$ , and  $M_{worst, D}$  be the recommended, the best, and the worst training data selection method on a dataset  $D$ , respectively, then this evaluation metric is defined (10).

$$RA = \frac{P_{M_{rec, D}} - P_{M_{worst, D}}}{P_{M_{best, D}} - P_{M_{worst, D}}} * 100\% \quad (10)$$

Where  $P_{M_{rec, D}}$ ,  $P_{M_{best, D}}$ , and  $P_{M_{worst, D}}$  represent the classification performance of  $M_{rec, D}$ ,  $M_{best, D}$ , and  $M_{worst, D}$ . RA measures the recommendation accuracy for a single recommendation. Hence, the recommendation accuracy over all test datasets is defined as ARA, calculated using (11).

$$ARA = \frac{\sum_{n=1}^N RA_n}{N} * 100\% \quad (11)$$

Where  $RA_n$  is recommendation accuracy on dataset  $n$ , while  $N$  is the number of test datasets.

### 3.8. Training data selection methods

We considered strict cross-project defect prediction [60] because it is prevalent in defect prediction studies [12]. Likewise, this approach does not require a labeled target data set, which an organization may not have when building an SDP model. The representative methods used in this study were chosen based on several factors, including their frequent use in SDP comparative studies [11], [13], [21], the granularity of the training data, and the selection strategy. Table 4 lists the included methods.

Table 4. Overview of the included training data selection methods

Ref	Granularity	Characteristics	Selection methods	Label name (ID*)
[8]	Instance-level	K-nearest neighbors	Distance to target instances, using Euclidean.	Turhan09 (11)
[15]	Instance-level	K-nearest neighbors	Distance to target instances, one source instance per target instance.	Peters13a (4)
[61]	Instance-level	K-nearest neighbors	Distance to target instances, using Hamming.	Ryu15 (10)
[62]	Instance-level	K-nearest neighbors	Distance to nearest unlike neighbors.	Peters15 (6)
[16]	Instance-level	K-nearest neighbors	Distance to target instances, considering number of defects	PHe18 (9)
[18]	Instance-level	Clustering	local cluster-based selection.	Menzies11 (3)
[17]	Instance-level	Clustering	instances in the same cluster as target instance.	Kawata15 (2)
[63]	Instance-level	Ranking-based	10% source instances with highest power.	Peters13b (5)
[9]	Project-level	K-nearest neighbors	Distance to target dataset, based on data characteristics.	Herbold13 (1)
[19]	Project-level	Ranking-based	Top 10 datasets rank based on predictive accuracy.	ZHe13 (13)
[21]	Multi-granularity	K-nearest neighbors+clustering	Distance to target datasets (Herbold13)+instances in the same cluster as target instance.	YL17 (12)
[20]	Multi-granularity	K-nearest neighbors+clustering	Distance to target datasets (Herbold13)+instances in the same cluster as target instance (Peters13a).	PHe0214 (8)
[20]	Multi-granularity	K-nearest neighbors at both level	Distance to target datasets (Herbold13)+Distance to target instances (Turhan09).	PHe0114 (7)

\* used as y-axis in Figure 5

### 3.9. Research question (RQ): Does the proposed recommendation system effectively select the optimal training data selection method for a particular dataset?

Recent studies on cross-project defect prediction reported different results about the most effective training data selection methods. A training data selection method can perform well on one dataset but poorly on another. Hence, we proposed a meta-learning approach to recommend the optimal method for selecting training data for a particular dataset. However, whether the proposed meta-learning approach outperforms the baseline method across datasets is unclear. Therefore, we studied the feasibility of using meta-learning to recommend a suitable training data selection method.

## 4. RESULTS AND DISCUSSION

We proposed a recommendation system for selecting the training data selection method for a specified dataset, with the kNN as the meta-learning algorithm. We adopted a leave-one-out strategy to validate the recommendation performance. Each release serves as the test dataset, while the remaining releases serve as candidate training datasets. We used 44 different release datasets from 14 different projects. We varied the value of k to evaluate the proposed system. Hence, for each k, the experiment is repeated 44 times. The recommendation performance is calculated for each repetition. Once all values are obtained, the average value across 44 releases is computed. The recommendation performance is measured using SRC, HR, and ARA.

To examine the performance difference among the meta-models, we performed a multiple comparison test to show the difference between all pairwise meta-models that are statistically significant. We conducted the Friedman test followed by Wilcoxon signed-rank test at a 5% significance level. Subsequently, we applied the Bonferroni correction to reduce the cumulative error caused by multiple comparisons.

### 4.1. Spearman rank correlation

Table 5 summarizes the overall meta-learning recommendation in terms of SRC on various sets of meta-features. Value in brackets denotes the ranks of SRC among the meta-features. The baseline is the average ranking on the entire dataset [43], often used as a benchmark for recommendation quality [14]. A score higher than the baseline value suggests that the recommendation is of higher quality.

To evaluate the meta-learning, SRC values are compared to baseline and critical values. Table 5 shows that the proposed meta-learning outperforms the baseline. This experiment tried to find the best training dataset selection method out of 13 available methods. With a 5% significance level, two rankings are said to be positively correlated when the critical value for ranking is 0.484 [64]. Hence, the ranking accuracy of the meta-learning approach is better than the critical value.

Table 6 shows that SStats statistically outperforms itemset, distance, correlation, and the baseline. However, it performs similarly to DC. Thus, SStats and DC sit in the same position. In other words, the SStats and DC are the most promising meta-features to induce the correct meta-model for recommending suitable training data selection methods.

Table 5. SRC of the meta-learning on various sets of meta-features

Meta-feature	SStats	DC	Itemset	Correlation	Distance	Baseline
AUC	$0.645 \pm 0.175$ (1)	$0.625 \pm 0.187$ (2)	$0.561 \pm 0.205$ (4)	$0.587 \pm 0.202$ (3)	$0.553 \pm 0.207$ (5)	0.419 (6)

Table 6. Wilcoxon signed-rank test results of the SRC for all pairwise comparisons between meta-features

Meta-feature	SStats	DC	Itemset	Correlation	Distance	Baseline
SStats	X	= (0.3191)	+ (0.0000)	+ (0.0000)	+ (0.0000)	+ (0.0000)
DC	= (0.3191)	X	+ (0.0000)	+ (0.0000)	+ (0.0000)	+ (0.0000)
Itemset	-- (0.0000)	-- (0.0000)	X	-- (0.0222)	= (1.0000)	+ (0.0000)
Correlation	-- (0.0000)	-- (0.0000)	+ (0.0222)	X	+ (0.0000)	+ (0.0000)
Distance	-- (0.0000)	-- (0.0000)	= (1.0000)	-- (0.0000)	X	+ (0.0000)
Baseline	-- (0.0000)	-- (0.0000)	-- (0.0000)	-- (0.0000)	-- (0.0000)	X

Table 5 also shows a high SRC variance. This high variation may be attributed to the multi-source strict cross-project defect prediction characteristics [3], [12], [60]. In this approach, each software release is selected as the target dataset. When ant-1.3 is the target dataset, the other releases from the same project (i.e., ant-1.4–ant-1.7) are excluded from the training datasets. All the remaining releases from the other projects are combined as the candidate for training datasets. As a result, all target datasets from the same project use the same training datasets. It leads to a metadata generation issue. Since all target datasets from the same project have the same training datasets, the meta-feature values for those target datasets are the same. However, the meta-target values are different. Consequently, it creates inconsistent metadata, where several meta-instances have identical meta-feature values but different target attribute values. This situation complicates meta-learning.

#### 4.2. Hit ratio and average recommendation accuracy

Table 7 and Table 8 summarize the overall meta-learning recommendation based on HR and ARA on various meta-features sets. Alg<sub>1</sub>, Alg<sub>2</sub>, and Alg<sub>3</sub> denote the recommendation options, i.e., the first, the second, and the third recommended method, respectively, while Alg[1,2,3] is the top three recommended methods. The value in brackets denotes the rank of the recommendation option among the meta-features.

Table 7. HR of the proposed meta-learning on different set of meta-features

Rec. method	HR				
	Stats	DC	Itemset	Distance	Correlation
Alg1	92.39 (1)	90.87 (2)	87.12 (5)	88.37 (4)	90.15 (3)
Alg2	65.42 (2)	65.57 (1)	62.99 (4)	62.92 (5)	63.90 (3)
Alg3	56.33 (2)	56.82 (1)	53.26 (4)	52.65 (5)	55.68 (3)
Alg[1,2,3]	97.73 (1)	96.93 (3)	96.21 (5)	96.44 (4)	97.20 (2)

Table 8. ARA of proposed meta-learning on different set of meta-features

Rec. method	ARA				
	Stats	DC	Itemset	Distance	Correlation
Alg1	89.58 (1)	88.11 (2)	85.05 (5)	86.38 (4)	87.08 (3)
Alg2	70.69 (1)	70.43 (2)	68.98 (4)	68.46 (5)	69.65 (3)
Alg3	64.08 (1)	63.87 (2)	61.02 (5)	62.11 (4)	63.20 (3)

Based on the recommendation options, the first recommended method (Alg<sub>1</sub>) consistently has the highest performance for both HR and ARA. The second (Alg<sub>2</sub>) and the third (Alg<sub>3</sub>) methods are then suggested. The difference in HR and ARA between Alg<sub>1</sub> and the other two options (i.e., Alg<sub>2</sub> and Alg<sub>3</sub>) is significant. The HR and ARA of the first recommended method are consistently higher than 87% and 85%, respectively. With the increased number of recommended algorithms, HR is also better. The HR for the top three recommended methods is over 96%, meaning that 96% of the top three recommended methods are the best.

We then compare the performance among the meta-features for both HR and ARA as well. We only include the first and the top three recommended methods. The recommendation model using SStats achieves the highest HR, 92.39%. At the same time, DC and correlation reach the second-best and the third-best position. Whereas distance and itemset consistently ranked last. The difference in HR between the meta-features is nearly identical, within the range of [87.12–92.39]. As for the top three recommendation options, SStats has the best hit ratio, reaching 97.73%. The SStats has the highest ARA among the meta-features for

all different recommendation options, i.e., 89.58%. Highlighting the first recommendation only, the ARA of SStats, DC, and correlation steadily in the first, second, and third position. At the same time, itemset and distance stand in the last two positions. The recommendation accuracy between the meta-features is almost similar.

The Friedman test found a significant difference in performance between pairwise meta-features. Hence, we proceed with the Wilcoxon signed-rank test and the Bonferroni correction. Table 9 displays the results of multiple comparison tests. The adjusted p-values for each comparison are shown in brackets. A '+' symbol indicates that the approach in a row is statistically better than the one in the column. A '--' denotes the opposite side, while '=' indicates an insignificant difference. Although SStats is better than DC for all possible pairwise comparisons (see Table 7 and Table 8) the difference is statistically insignificant. In all cases, SStats also outperforms itemset and distance.

Table 9. Wilcoxon signed-rank test results of HR and ARA for all pairwise comparisons between various meta-features

classifier meta-feature	HR					ARA				
	Stats	DC	Itemset	Distance	Correlation	Stats	DC	Itemset	Distance	Correlation
SStats	X	=(0.6851)+	(0.0000)+	(0.0057)=	(0.2229)	X	=(0.2867)	+(0.0000)	+(0.0000)	+(0.0088)
DC	=(0.6851)	X	+(0.0002)=	(0.0541)=	(1.0000)	=(0.2867)	X	+(0.0000)	=(0.2038)	=(1.0000)
Itemset	--(0.0000)--	(0.0002)X	=(0.6772)--	(0.0022)	--(0.0000)--	(0.0000)	X	=(0.4961)	--(0.0033)	
Distance	--(0.0057)=	(0.0541)=	(0.6772)X	=(0.1335)	--(0.0000)=	(0.2038)	=(0.4961)	X	=(0.2702)	
Correlation	=(0.2229)	=(1.0000)+	(0.0022)=	(0.1335)X	--(0.0088)=	(1.0000)	+(0.0033)	=(0.2702)	X	

Table 10 compiles all evaluations at the meta-level. The '=' in brackets denotes that the corresponding meta-feature is statistically comparable to SStats, while '+' means that SStats statistically outperform this meta-feature. We observed three points from Table 10. Firstly, the ranking accuracy (SRC) is better than the critical value meaning that the recommended and actual rankings are positively correlated. Thus, the proposed meta-learning approach works well in that the recommended methods have recommended rankings matching the actual rankings. Secondly, the proposed meta-learning has a good HR and ARA. It can effectively recommend the optimal method. The first recommended method has an HR of over 85% and an ARA of over 88 percent. Thirdly, the SStats outperform other meta-features for both HR and ARA, despite being statistically insignificant in some cases. It means SStats is better than others at selecting training data method.

Table 10. The complete recommendation performance of the proposed meta-learning

Recommendation measure	SStats	DC	Itemset	Distance	Correlation
SRC	0.645	0.625 (=)	0.561 (+)	0.553 (+)	0.587 (+)
HR	92.39	90.87 (=)	87.12 (+)	88.37 (+)	90.15 (=)
ARA	89.58	88.11 (=)	85.05 (+)	86.38 (+)	87.08 (+)

### 4.3. Classification Performance of the recommended method

At the base level, we compare the classification performance of recommended training data selection method to that of baseline methods. We choose the SStats as the meta-feature for the proposed meta-learning since it is the best meta-features based on the previous analysis. For baseline methods, we used Turhan09, Herbold13, PHe0114, and the majority method. Turhan09 represents the instance-based training data selection methods, while Herbold13 and PHe01 14 represent the project-level and multi granularity-level methods. The majority method is a training data selection method that most frequently appears as the best method for the entire datasets. We rank the method candidates for each dataset based on their performance and then count how often a training data selection method obtains rank 1. The method with the most rank 1 is the majority method.

We plot SRC for different k values to see how the number of nearest neighbor datasets affects recommendation performance. We identify the k value where meta-learning performs optimally. Figure 3 shows that the SRC improves with more nearest neighbors. After k=6, the performance gradually decreases, and at a particular point, it is the default ranking. In most cases, the recommendation system outperforms the default ranking.

We then investigate k=6 because it is the optimal number for the proposed meta-learning. The results are shown in Table 11. Header %freq denotes how often a particular method performs best for whole datasets. The sum of %freq is higher than 100% since, for some datasets, more than one approach achieves

the optimal performance. Table 11 shows that the proposed meta-learning (MTL.rec1) outperforms the baseline methods (p-value is less than 0.05). The improvement of MTL.rec1 over the baseline method varies. It achieves a small improvement over the majority (i.e.,  $\delta=0.180$ ), large improvement over Turhan09, Herbold13, and PHe0114 (i.e.,  $\delta=0.624, 0.584, 0.630$ , respectively, which higher than 0.474).

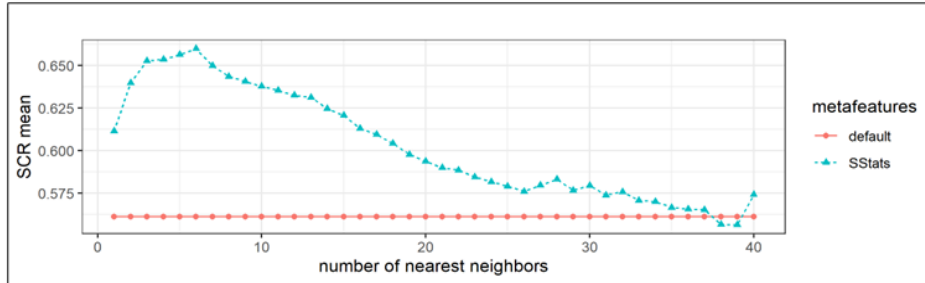


Figure 3. The recommendation performance of SStats for different values of k in terms of AUC

Table 11. Comparison of classification accuracy between the recommended method and the baseline methods

Methods	Statistical comparison results			
	Mean $\pm$ stdv	%freq.	p-value	Effect size
MTL.rec1	0.7129 $\pm$ 0.0960	80.68	--	--
majority	0.6938 $\pm$ 0.1156	67.80	2.726e-06	0.180
Turhan09	0.6302 $\pm$ 0.1114	9.95	2.437e-10	0.624
Herbold13	0.6148 $\pm$ 0.1056	14.39	3.465e-10	0.684
PHe0114	0.6293 $\pm$ 0.1082	7.58	7.685e-11	0.630

For further analysis, we compare the recommended and the majority method for each dataset, plotted in Figure 4. In addition, Figure 5 displays the recommended method for each classification algorithm. The ‘o’ denotes a hit, which is a correct recommended method, whereas ‘x’ represents the opposite side.

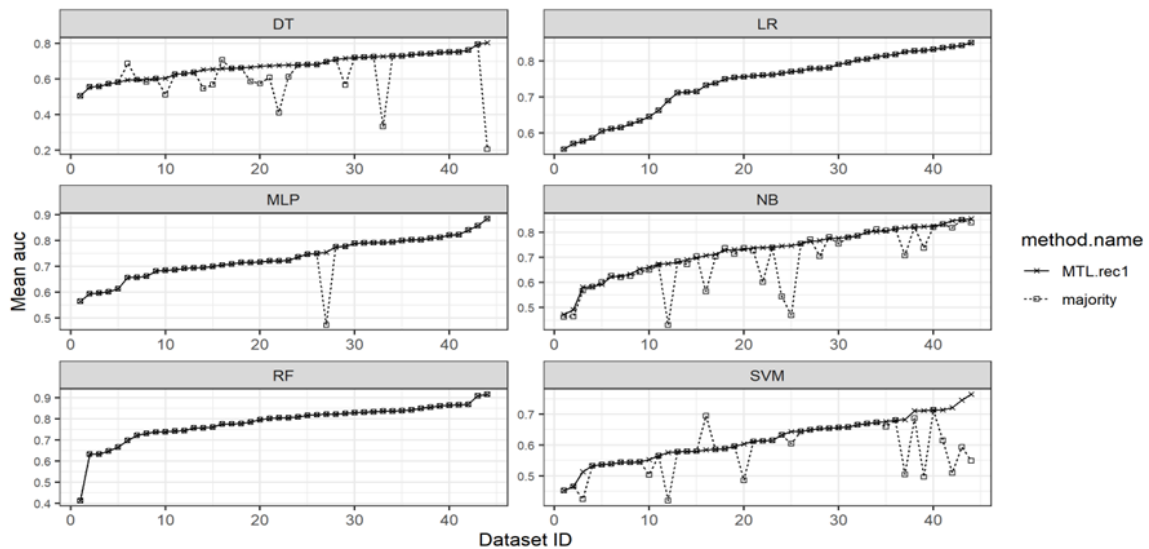


Figure 4. Performance comparison between the recommended method and the majority method

Figure 4 shows that the recommended method performs the same as the majority when using LR and RF as classification algorithms. Therefore, we suspect that a method is dominantly superior to the others on most datasets, and it also happens to be the first recommended method. It is confirmed in Figure 5. However, the performance difference between the two examined methods is detected when using the other classifiers,

i.e., DT, NB, and SVM. It suggests that the majority method is not dominant, and the proposed meta-learning can identify the other method as the best-performing method for a given dataset. Since the performance of each method in Table 11 is the aggregate performance of all classifiers and the performance in some classifiers is nearly identical, the average performance of the recommended method and the majority method is not significantly different.

Figure 4 and Figure 5 raise a further question of whether the proposed system is still necessary. The dominant method only occurs in LR and RF but not for other classification algorithms. For the other classification algorithms, it appears that there is no dominant method, and the proposed system has succeeded in identifying the optimal method for a particular dataset. Hence, it indicates that the proposed system works properly and is still needed.

**4.4. Answer to RQ**

The meta-level analysis shows that the proposed meta-learning can effectively recommend the appropriate training data selection methods. The high values of the hit ratio and the average recommendation accuracy of the first recommended method, higher than 87% and 85%, indicate this. The base-level analysis also found that the recommended method outperformed the baseline methods.

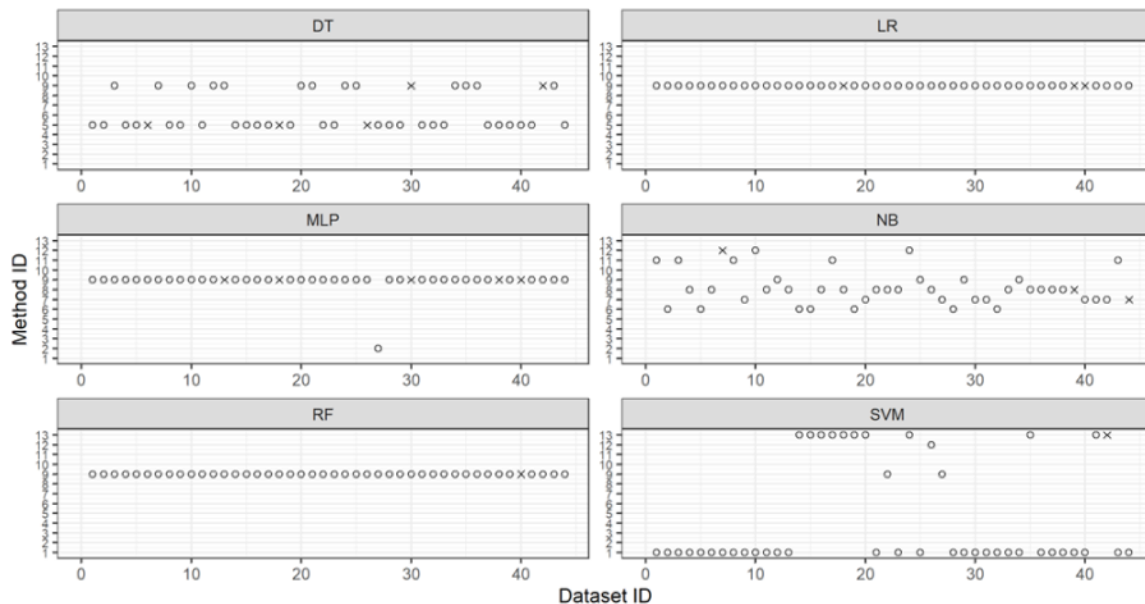


Figure 5. Recommended training data selection method for each classification algorithm

**5. CONCLUSION**

The performance of an SDP model could be improved if it is trained using suitable training datasets. Hence, the correct identification of suitable training data is important. There are many methods to select the relevant training datasets. However, prior research found no single method performs uniformly across all datasets. Therefore, recommending the optimal training data selection method for a particular problem become essential. This study proposed a recommendation system to choose the appropriate training data selection methods for a particular dataset in a CPDP setting.

The proposed system works by first extracting meta-features of training datasets. It then evaluates the performance of each training data selection method on training datasets. Afterward, it creates a meta-knowledge database that relates dataset characteristics to the method performance, enabling us to select the optimal training data selection method. When an unseen dataset appears, the proposed system identifies the nearest neighbors datasets and the corresponding performances. It subsequently estimates the performance of training data selection methods on the unseen dataset and recommends the optimal method. We evaluated the proposed system using 44 datasets, 13 training data selection methods, and six classifiers under four performance metrics, i.e., the SRC, hit ratio (AR), average recommendation accuracy, and AUC. The results concluded that the recommendation system effectively recommended suitable training data selection methods. Also, the proposed method outperformed the baseline methods.

## ACKNOWLEDGEMENTS

This study has been supported by (1). The Faculty of Information and Communication Technology Universiti Teknikal Malaysia Melaka, Malaysia, (2). The Technical Implementing Service unit of Information System, Faculty of Industrial Technology, Universitas Atma Jaya Yogyakarta, Indonesia, and (3). The office of Information System, Universitas Atma Jaya Yogyakarta, Indonesia.

## REFERENCES

- [1] 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 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering on European software engineering conference and foundations of software engineering symposium*, 2009, p. 91, doi: 10.1145/1595696.1595713.
- [2] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, "An investigation on the feasibility of cross-project defect prediction," *Automated Software Engineering*, vol. 19, no. 2, pp. 167–199, Jun. 2012, doi: 10.1007/s10515-011-0090-3.
- [3] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *Proceedings - International Conference on Software Engineering*, May 2013, pp. 382–391, doi: 10.1109/ICSE.2013.6606584.
- [4] A. E. C. Cruz and K. Ochimizu, "Towards logistic regression models for predicting fault-prone code across software projects," *2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009*, pp. 460–463, 2009, doi: 10.1109/ESEM.2009.5316002.
- [5] S. Watanabe, H. Kaiya, and K. Kajiri, "Adapting a fault prediction model to allow inter languagereuse," in *Proceedings of the 4th international workshop on Predictor models in software engineering - PROMISE '08*, 2008, pp. 19–24, doi: 10.1145/1370788.1370794.
- [6] F. Zhang, I. Keivanloo, and Y. Zou, "Data transformation in cross-project defect prediction," *Empirical Software Engineering*, vol. 22, no. 6, pp. 3186–3218, Dec. 2017, doi: 10.1007/s10664-017-9516-2.
- [7] Y. Jiang, B. Kukic, and T. Menzies, "Can data transformation help in the detection of fault-prone modules?," in *DEFECTS'08: 2008 International Symposium on Software Testing and Analysis - Proceedings of the 2008 Workshop on Defects in Large Software Systems 2008, DEFECTS'08*, 2008, pp. 16–20, doi: 10.1145/1390817.1390822.
- [8] 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 Software Engineering*, vol. 14, no. 5, pp. 540–578, Oct. 2009, doi: 10.1007/s10664-008-9103-7.
- [9] S. Herbold, "Training data selection for cross-project defect prediction," in *Proceedings of the 9th International Conference on Predictive Models in Software Engineering*, Oct. 2013, pp. 1–10, doi: 10.1145/2499393.2499395.
- [10] B. L. Sinaga, S. Ahmad, and Z. A. Abas, "A review of training data selection in software defect prediction," *Journal of Theoretical and Applied Information Technology*, vol. 98, no. 12, pp. 2092–2108, 2020.
- [11] Y. Bin, K. Zhou, H. Lu, Y. Zhou, and B. Xu, "Training data selection for cross-project defection prediction: which approach is better?," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Nov. 2017, vol. 2017-Novem, pp. 354–363, doi: 10.1109/ESEM.2017.49.
- [12] S. Herbold, A. Trautsch, and J. Grabowski, "A comparative study to benchmark cross-project defect prediction approaches," *IEEE Transactions on Software Engineering*, vol. 44, no. 9, pp. 811–833, Sep. 2018, doi: 10.1109/TSE.2017.2724538.
- [13] H. Luo, H. Dai, W. Peng, W. Hu, and F. Li, "An empirical study of training data selection methods for ranking-oriented cross-project defect prediction," *Sensors*, vol. 21, no. 22, pp. 1–18, 2021, doi: 10.3390/s21227535.
- [14] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning: applications to data mining*, Heidelberg: Springer, 2009, doi: 10.1007/978-3-540-73263-1.
- [15] F. Peters, T. Menzies, and A. Marcus, "Better cross company defect prediction," in *IEEE International Working Conference on Mining Software Repositories*, 2013, pp. 409–418, doi: 10.1109/MSR.2013.6624057.
- [16] P. He, Y. He, L. Yu, and B. Li, "An improved method for cross-project defect prediction by simplifying training data," *Mathematical Problems in Engineering*, vol. 2018, pp. 1–18, Jun. 2018, doi: 10.1155/2018/2650415.
- [17] K. Kawata, S. Amasaki, and T. Yokogawa, "Improving relevancy filter methods for cross-project defect prediction," in *2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence*, Jul. 2015, pp. 2–7, doi: 10.1109/ACIT-CSI.2015.104.
- [18] T. Menzies, A. Butcher, A. Marcus, T. Zimmermann, and D. Cok, "Local vs global models for effort estimation and defect prediction," in *2011 26th IEEE/ACM International Conference on Automated Software Engineering, ASE 2011, Proceedings*, 2011, pp. 343–351, doi: 10.1109/ASE.2011.6100072.
- [19] Z. He, F. Peters, T. Menzies, and Y. Yang, "Learning from open-source projects: an empirical study on defect prediction," in *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, Oct. 2013, pp. 45–54, doi: 10.1109/ESEM.2013.20.
- [20] P. He, B. Li, D. Zhang, and Y. Ma, "Simplification of training data for cross-project defect prediction," May 2014, [Online]. Available: <http://arxiv.org/abs/1405.0773>.
- [21] Y. Li, Z. Huang, Y. Wang, and B. Fang, "Evaluating data filter on cross-project defect prediction: comparison and improvements," *IEEE Access*, vol. 5, pp. 25646–25656, 2017, doi: 10.1109/ACCESS.2017.2771460.
- [22] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "The impact of automated parameter optimization on defect prediction models," *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 683–711, Jul. 2019, doi: 10.1109/TSE.2018.2794977.
- [23] S. N. D. Dôres, L. Alves, D. D. Ruiz, and R. C. Barros, "A meta-learning framework for algorithm recommendation in software fault prediction," *Proceedings of the ACM Symposium on Applied Computing*, vol. 04-08-April, pp. 1486–1491, 2016, doi: 10.1145/2851613.2851788.
- [24] D. D. Nucci, F. Palomba, R. Oliveto, and A. D. Lucia, "Dynamic selection of classifiers in bug prediction: an adaptive method," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 3, pp. 202–212, 2017, doi: 10.1109/TETCI.2017.2699224.
- [25] Z. Sun, J. Zhang, H. Sun, and X. Zhu, "Collaborative filtering based recommendation of sampling methods for software defect prediction," *Applied Soft Computing Journal*, vol. 90, 2020, doi: 10.1016/j.asoc.2020.106163.
- [26] F. Porto, L. Minku, E. Mendes, and A. Simao, "A systematic study of cross-project defect prediction with meta-learning," Feb. 2018, [Online]. Available: <http://arxiv.org/abs/1802.06025>.




- [27] M. A. Muñoz, L. Villanova, D. Baatar, and K. Smith-Miles, "Instance spaces for machine learning classification," *Machine Learning*, vol. 107, no. 1, pp. 109–147, Jan. 2018, doi: 10.1007/s10994-017-5629-5.
- [28] C. Castiello, G. Castellano, and A. M. Fanelli, "Meta-data: characterization of input features for meta-learning," in *MDAI 2005: Modeling Decisions for Artificial Intelligence*, 2005, pp. 457–468, doi: 10.1007/11526018\_45.
- [29] A. Rivolli, L. P. F. Garcia, C. Soares, J. Vanschoren, and A. C. P. L. F. de Carvalho, "Meta-features for meta-learning," *Knowledge-Based Systems*, vol. 240, p. 108101, 2022, doi: 10.1016/j.knsys.2021.108101.
- [30] S. Hosseini, B. Turhan, and D. Gunarathna, "A systematic literature review and meta-analysis on cross project defect prediction," *IEEE Transactions on Software Engineering*, vol. 45, no. 2, pp. 111–147, Feb. 2019, doi: 10.1109/TSE.2017.2770124.
- [31] B. Ghotra, S. McIntosh, and A. E. Hassan, "A large-scale study of the impact of feature selection techniques on defect classification models," in *IEEE International Working Conference on Mining Software Repositories*, 2017, pp. 146–157, doi: 10.1109/MSR.2017.18.
- [32] A. C. Lorena, L. P. F. Garcia, J. Lehmann, M. C. P. Souto, and T. K. A. M. Ho, "How complex is your classification problem?: A survey on measuring classification complexity," *ACM Computing Surveys*, vol. 52, no. 5, 2019, doi: 10.1145/3347711.
- [33] Z. Sun, J. Li, and H. Sun, "An empirical study of public data quality problems in cross project defect prediction," in *Proceedings of 12th International Symposium on Empirical Software Engineering and Measurement*, May 2018, vol. 26, no. 2, pp. 203–209.
- [34] L. T. DeCarlo, "On the meaning and use of kurtosis," *Psychological Methods*, vol. 2, no. 3, pp. 292–307, 1997, doi: 10.1037/1082-989x.2.3.292.
- [35] C. Davies, "Meta-feature taxonomy for supporting automatic machine learning," M.S. Thesis, University of Calgary, 2019.
- [36] T. Mitchell, *Machine learning*. New York, NY, USA: Mc Graw Hill, 1997.
- [37] J. Vanschoren, "Meta-learning," F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham: Springer International Publishing, 2019, pp. 35–61, doi: 10.1007/978-3-030-05318-5\_2.
- [38] M. A. Salama, A. E. Hassanien, and K. Revett, "Employment of neural network and rough set in meta-learning," *Memetic Computing*, vol. 5, no. 3, pp. 165–177, 2013, doi: 10.1007/s12293-013-0114-6.
- [39] D. G. Ferrari and L. N. de Castro, "Clustering algorithm selection by meta-learning systems: a new distance-based problem characterization and ranking combination methods," *Information Sciences*, vol. 301, pp. 181–194, Apr. 2015, doi: 10.1016/j.ins.2014.12.044.
- [40] B. A. Pimentel and A. C. P. L. F. de Carvalho, "A new data characterization for selecting clustering algorithms using meta-learning," *Information Sciences*, vol. 477, pp. 203–219, Mar. 2019, doi: 10.1016/j.ins.2018.10.043.
- [41] Q. Song, G. Wang, and C. Wang, "Automatic recommendation of classification algorithms based on data set characteristics," *Pattern Recognition*, vol. 45, no. 7, pp. 2672–2689, 2012, doi: 10.1016/j.patcog.2011.12.025.
- [42] D. W. Aha, "Generalizing from case studies: a case study," in *Machine Learning Proceedings 1992*, 1992, pp. 1–10, doi: 10.1016/B978-1-55860-247-2.50006-1.
- [43] P. B. Brazdil, C. Soares, and J. P. da Costa, "Ranking learning algorithms: using IBL and meta-learning on accuracy and time results," *Machine Learning*, vol. 50, no. 3, pp. 251–277, 2003, doi: 10.1023/A:1021713901879.
- [44] G. Wang, Q. Song, and X. Zhu, "An improved data characterization method and its application in classification algorithm recommendation," *Applied Intelligence*, vol. 43, no. 4, pp. 892–912, 2015, doi: 10.1007/s10489-015-0689-3.
- [45] P. B. Brazdil and C. Soares, "A comparison of ranking methods for classification algorithm selection," in *Machine Learning: ECML 2000*, 2000, vol. 1810, pp. 63–75, doi: 10.1007/3-540-45164-1\_8.
- [46] L. Li, Y. Wang, Y. Xu, and K. Y. Lin, "Meta-learning based industrial intelligence of feature nearest algorithm selection framework for classification problems," *Journal of Manufacturing Systems*, 2021, doi: 10.1016/j.jmsy.2021.03.007.
- [47] J. Kanda, A. de Carvalho, E. Hruschka, C. Soares, and P. Brazdil, "Meta-learning to select the best meta-heuristic for the traveling salesman problem: a comparison of meta-features," *Neurocomputing*, vol. 205, pp. 393–406, 2016, doi: 10.1016/j.neucom.2016.04.027.
- [48] X. Chu, F. Cai, C. Cui, M. Hu, L. Li, and Q. Qin, "Adaptive recommendation model using meta-learning for population-based algorithms," *Information Sciences*, vol. 476, pp. 192–210, 2019, doi: 10.1016/j.ins.2018.10.013.
- [49] G. Wang, Q. Song, H. Sun, X. Zhang, B. Xu, and Y. Zhou, "A feature subset selection algorithm automatic recommendation method," *Journal of Artificial Intelligence Research*, vol. 47, pp. 1–34, 2013, doi: 10.1613/jair.3831.
- [50] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering - PROMISE '10*, 2010, p. 1, doi: 10.1145/1868328.1868342.
- [51] D. Bowes, T. Hall, and J. Petrić, "Software defect prediction: do different classifiers find the same defects?," *Software Quality Journal*, vol. 26, no. 2, pp. 525–552, 2018, doi: 10.1007/s11219-016-9353-3.
- [52] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Applied Soft Computing Journal*, vol. 27, pp. 504–518, Feb. 2015, doi: 10.1016/j.asoc.2014.11.023.
- [53] A. Panichella, R. Oliveto, and A. De Lucia, "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*, Feb. 2014, pp. 164–173, doi: 10.1109/CSMR-WCRE.2014.6747166.
- [54] A. Iqbal *et al.*, "Performance analysis of machine learning techniques on software defect prediction using NASA datasets," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 5, pp. 300–308, 2019, doi: 10.14569/ijacsa.2019.0100538.
- [55] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276–1304, Nov. 2012, doi: 10.1109/TSE.2011.103.
- [56] C. Spearman, "The proof and measurement of association between two things," *The American Journal of Psychology*, vol. 100, no. 3/4, p. 441, 1987, doi: 10.2307/1422689.
- [57] X. Zhang, R. Li, B. Zhang, Y. Yang, J. Guo, and X. Ji, "An instance-based learning recommendation algorithm of imbalance handling methods," *Applied Mathematics and Computation*, vol. 351, pp. 204–218, 2019, doi: 10.1016/j.amc.2018.12.020.
- [58] P. Brazdil, J. Gama, and B. Henery, "Characterizing the applicability of classification algorithms using meta-level learning," in *Machine Learning: ECML-94*, 1994, pp. 83–102, doi: 10.1007/3-540-57868-4\_52.
- [59] L. Todorovski and S. Džeroski, "Experiments in meta-level learning with LLP," in *PKDD 1999: Principles of Data Mining and Knowledge Discovery*, 1999, pp. 98–106, doi: 10.1007/978-3-540-48247-5\_11.
- [60] S. Herbold, A. Trautsch, and J. Grabowski, "Global vs. local models for cross-project defect prediction: A replication study," *Empirical Software Engineering*, vol. 22, no. 4, pp. 1866–1902, 2017, doi: 10.1007/s10664-016-9468-y.
- [61] D. Ryu, J.-I. Jang, and J. Baik, "A hybrid instance selection using nearest-neighbor for cross-project defect prediction," *Journal of Computer Science and Technology*, vol. 30, no. 5, pp. 969–980, Sep. 2015, doi: 10.1007/s11390-015-1575-5.






- [62] F. Peters, T. Menzies, and L. Layman, "LACE2: better privacy-preserving data sharing for cross project defect prediction," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, May 2015, vol. 1, pp. 801–811, doi: 10.1109/ICSE.2015.92.
- [63] F. Peters, T. Menzies, L. Gong, and H. Zhang, "Balancing privacy and utility in cross-company defect prediction," *IEEE Transactions on Software Engineering*, vol. 39, no. 8, pp. 1054–1068, 2013, doi: 10.1109/TSE.2013.6.
- [64] J. H. Zar, "Significance testing of the spearman rank correlation coefficient," *Journal of the American Statistical Association*, vol. 67, no. 339, pp. 578–580, 1972, doi: 10.1080/01621459.1972.10481251.

## BIOGRAPHIES OF AUTHORS






**Benjamin Langgu Sinaga**    is currently with Department of Informatics Universitas Atma Jaya Yogyakarta, Indonesia. He received bachelor's degree in electrical engineering from Department of Electrical Engineering, Gadjah Mada University, Yogyakarta, Indonesia in 1994. He obtained master degree in computer science from School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia in 2000. He currently pursuing Ph.D. degree at Faculty of Information and Communication Technology, Universiti Teknikal Malaysia, Melaka. His research interests include software engineering, information system adoption, machine learning. He can be contacted at email: benjamin.sinaga@uajy.ac.id.






**Sabrina Ahmad**    is an Associate Professor and the Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka. She has qualifications and undergone formal trainings in the area of software engineering and development. She is specialized in requirements engineering in both research and practice. She obtained her Ph.D. in Computer Science from The University of Western Australia in 2012. She endeavors to maintain the bridge between academia and practitioners and therefore continue strengthening software engineering curriculum and apply the knowledge in the industries through consultation projects. Therefore, she continues to upgrade her skills and knowledge through professional training and certification. She obtained Professional Certification in Requirements Engineering, a Certified Tester and a Certified Professional IT Architect. She can be contacted at email: sabrinaahmad@utem.edu.my.



**Zuraida Abal Abas**    is currently an Associate Professor at the Department of Intelligent Computing & Analytics, Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka. She obtained her Ph.D. in Mathematics at Universiti Teknologi Malaysia; her MSc in Operational Research at London School of Economics, United Kingdom; and her BSc in Industrial Mathematics at Universiti Teknologi Malaysia. Her research interests are operational research, analytics, modeling and simulation. She can be contacted at email: zuridaa@utem.edu.my.



**Intan Ermahani A. Jalil**    is a Senior Lecturer at Faculty of Information and Communication Technology (FTMK), Universiti Teknikal Malaysia Melaka (UTeM). She is a member of Computational Intelligence Technologies (CIT) Lab research group at FTMK, UTeM and also a member of Malaysia Board of Technologists (MBOT). She received her Bachelor Degree in Computer from Universiti Teknologi Malaysia (UTM) and Master of Science in Software Engineering from University of Brighton, United Kingdom. She has also received her Ph.D. in Computer from Universiti Teknologi Malaysia (UTM). Her research interest includes soft computing, pattern recognition, image processing, software engineering and software testing. She is certified by INFOSYS and also in Certified Tester Foundation Level (CTFL). She can be contacted at email: ermahani@utem.edu.my.