A Comprehensive Investigation of the Impact of Class Overlap on Software Defect Prediction

Lina Gong, Haoxiang Zhang, Jingxuan Zhang, Mingqiang Wei, *Senior Member*, *IEEE*, and Zhiqiu Huang

Abstract—Software Defect Prediction (SDP) is one of the most vital and cost-efficient operations to ensure the quality of software under developed. The performance of SDP heavily relies on the characteristics of experimental datasets (or say SDP datasets). However, there often exists the phenomenon of class overlap in the SDP datasets, i.e., defective modules and non-defective modules are similar in terms of values of metrics. Class overlap hinders the smooth performance as well as the use of SDP models. Even though efforts have been made to investigate the impact of overlapping instance removing techniques on the performance of SDP, many open issues are still challenging yet unknown. For example, 1) how to effectively identify the overlapping instances? 2) Whether is the phenomenon of class overlap universal in the SDP datasets? 3) What are the impacts of class overlap on the performance and interpretation of SDP models? Questions like these are very important but have not been fully explored yet. In this paper, we conduct an empirical study to comprehensively investigate the impact of class overlap on SDP. Specifically, we first propose an overlapping instances identification approach by analyzing the class distribution in the local neighborhood of a given instance. Based on the approach, we then investigate the impact of class overlap on the performance and the interpretation of seven representative SDP models. Finally, we investigate the impact of two common overlapping instance handling techniques (i.e., removing and separating techniques) on the performance of SDP models. Through an extensive case study on 230 datasets that span across industrial and open-source software projects, we observe that: i) 70.0% of SDP datasets contain overlapping instances; ii) different levels of class overlap have different impacts on the performance of SDP models. The class overlap ratio and the number of instances seriously affect the stability of the performance of SDP models; iii) class overlap affects the rank of the important feature list of SDP models, particularly the feature lists at the top 2 and top 3 ranks; IV) Class overlap handling techniques could statistically significantly improve the performance of SDP models trained on datasets with over 12.5% overlap ratios. Therefore, on the basis of these findings we suggest that future work in SDP should apply our proposed KNN method to: i) identify whether the overlap ratios of their defect datasets are greater than 12.5% before building SDP models; ii) remove the overlapping instances to find the more consistent guiding significance metrics; iii) combine RF classifier and class overlap handling techniques when reducing the efforts to review codes.

Index Terms—Software defect prediction, class overlap, data quality, local analysis, K-nearest neighbourhood, software metrics

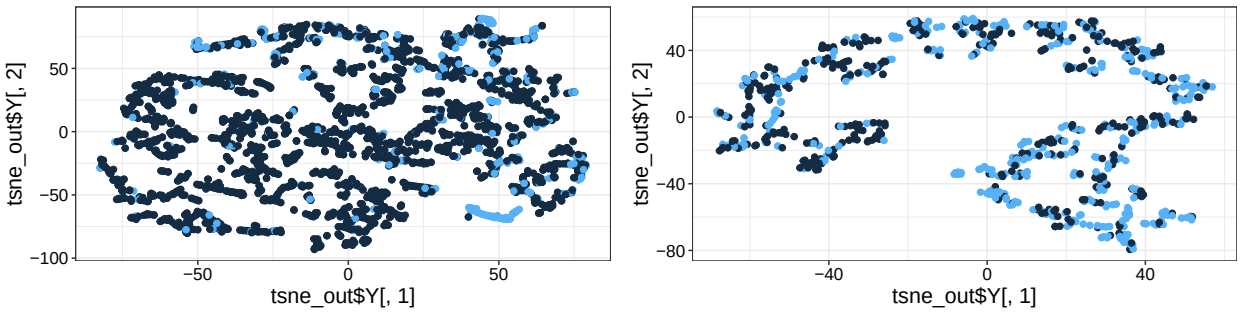
1 Introduction

Software Defect Prediction (SDP) aims to identify modules that are defect prone in the Testing Phase of the Software Development Life Cycle (SDLC). SDP plays an increasingly important role in ensuring the quality of software being developed [17, 37, 59, 80] in the era of Artificial Intelligence. The performance of SDP models heavily relies on the quality of SDP (experimental) datasets. Typically, the SDP datasets include a large number of modules (instances), which are expressed by a series of metrics (e.g., code metrics and process metrics) and labels (e.g., defect and non-defect). However, when collecting SDP datasets, there is a phenomenon that software modules with similar structure and size belong to different classes. The common region combined by these software modules in SDP datasets would lead to class overlap, since class overlap occurs when instances of more than one class share a common region in the data space [74].

In the domain of machine learning, Prati et al. [52] firstly conducted a systematic study to investigate the relationship between class overlap and the performance of classifiers. They found that the loss of the performance of classifiers was not directly caused by class imbalance, but was related to the degree of class overlap. After that, Denil and Trappenberg [15], García et al. [16] have revealed that class overlap seriously hinders the prediction ability of classifiers. However, in the context of SDP, only several researchers [11, 20] have raised concerns about removing overlapping instances from the SDP datasets. For example, to eliminate the effect of class overlap on the SDP model, Chen et al. [11] applied neighbor cleaning to remove overlapping non-defective instances. In addition, our prior study [20] exploited an improved K-Means clustering cleaning approach (IKMCCA) to remove overlapping defective and non-defective instances. The two studies have shown that the performance of SDP models can be improved by removing overlapping instances. That is to say, the class overlap is harmful to the performance of SDP models.

However, some important issues have not been fully investigated in terms of overlapping instances and their impacts on the SDP models. First, ***none*** of the prior studies attempts to quantify the distribution and prevalence of overlapping instances in the SDP datasets. For example, for the commonly used SDP datasets *prop-5-40* and *xalan-2.5*, we employ the t-distributed Stochastic Neighbor Embedding (t-SNE) method to project them into 2-dimensional space and visualize the overlapping instances among independent features in Figure 1, where X-axis and Y-axis represent the 2-dimensional features projected by t-SNE respectively. Blue dots represent the defective instances, while black dots represent the non-defective instances. From the visualization

- *Lina Gong, Jingxuan Zhang, Mingqiang Wei and Zhiqiu Huang are with College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. E-mail: {gonglina,jxzhang,mqwei,zqhuang}@nuaa.edu.cn. Haoxiang Zhang is with Software Analysis and Intelligence Lab (SAIL), School of Computing, Queen's University, Canada. E-mail: haoxiang.zhang@acm.org*
- *Lina Gong and Jingxuan Zhang are the co-corresponding authors.*



(a) prop-5-40 (b) xalan-2.5

Fig. 1: The visualization of prop-5-40 and xalan-2.5.

of these two projects, we can observe that they have different ratios and distributions of overlapping instances. Unfortunately, none of the existing studies has quantified them. Second, even though existing studies have shown that removing overlapping instances could improve the performance of SDP models with one evaluation metric, they do not investigate the relationship between overlapping instances and the performance as well as the interpretation of SDP models from different and comprehensive evaluation metrics. Third, there exists two types of overlapping instances handling techniques, i.e., removing and separating. However, existing studies have only explore the impact of the removing technique on the performance of SDP models without investigating the separating technique and comparing the two techniques.

To fill these research gaps, we first propose a K-nearest neighborhood-based method to identify overlapping instances. We then systemically investigate the impact of class overlap on the performance and interpretation of SDP models. Through a large-scale empirical study of 230 SDP datasets, we structure our study by answering the following Research Questions (RQs):

• **RQ1: How effective our proposed method is in identifying overlapping instances in the SDP datasets?**
For most of the studied 230 SDP datasets, our proposed overlapping identification method could identify more than 70% of overlapping instances identified by our compared methods. In contrast, the comparison methods identify less than 60% of overlapping instances identified by our proposed overlapping identified method. Therefore, we recommend researchers and practitioners use our proposed K-nearest neighborhood-based method to identify overlapping instances before they build their SDP models.

• **RQ2: How overlapped are the SDP datasets?**
Our statistics illustrate that 70.0% of SDP datasets have a class overlap ratio above 5%. Indeed, 32.6% of SDP datasets have a class overlap ratio above 12.5%, especially the frequently used defect prediction datasets in NASA, AEEEM, PROMISE, ReLink and SOFTLAB. It shows that overlapping instances are ubiquitous in the SDP datasets. Therefore, we recommend researchers and practitioners identify whether there are overlapping instances in their studied datasets before building the SDP models.

• **RQ3: How do different levels of class overlap impact the performance of defect prediction models?**
Experimental results indicate that the levels of class overlap have different effects on the performance of SDP models. The class overlap ratio that is over 12.5% seriously affects the stability of the performance of SDP models. Meanwhile, the number of instances in the training datasets would affect the stability of the performance of SDP models trained on the training datasets with over 12.5% overlap ratio. Therefore, we recommend that researchers and practitioners should check the overlap ratio of their defect prediction datasets and repeatedly run their models to prevent unstable results and findings.

• **RQ4: How does class overlap impact the interpretation of defect prediction models?**
Experimental results indicate that features at the top 1 rank (i.e, important feature ranked in Top-1 lists calculated with Section 4.7) in original and non-overlap datasets have a strong agreement, features at the top 2 rank have a partial agreement, while features at the top 3 rank vary considerably. Meanwhile, removing overlapping instances could help to find more consistent guiding significance metrics for a given dataset and different releases of a given project. Therefore, we recommend that researchers and practitioners should remove the overlapping instances to find the more consistent guiding significance metrics.

• **RQ5: How do different class overlap handling techniques impact the performance of defect prediction models?**
Our study shows that class overlap handling techniques could statistically significantly improve the performance of SDP models trained on datasets with over 12.5% ratios. Removing and Separating techniques have similar impacts on the performance of SDP models. The RF classifier can obtain better performance on both the original datasets and the datasets handled by removing or separating techniques. Therefore, we recommend researchers and practitioners apply RF classifier with removing or separating techniques to build SDP models.

In summary, the main contributions of this work are fourfold:

1) We propose a novel method for identifying overlapping instances in the SDP datasets, which is based on analyzing the class distribution in the local neighborhood of the given instances.

2) We first quantify the degree of overlapping instances across proprietary and open source available in the SDP datasets.

3) We study the exact relationship between the experimental factors (i.e., overlap ratio, class overlap techniques, and classifiers) and the performance as well as the interpretation of SDP models.

4) We have released a replication package in the GitHub repository [1] to provide transparency into our process.

1. https://github.com/glnmzx888/SDP_overlap

For the remainder of this paper, Section 2 presents the motivation and related work of our study. Section 3 shows the details of the identifying overlapping instances method, which is based on analyzing a class distribution in a local neighbourhood of the given instances. Section 4 provides our study design, including studied datasets and experimental setup, while Section 5 presents and discusses our experimental results. Section 7 presents the implications that can be inferred from our results. Section 8 discusses the threats to the validity of our observations, followed by the conclusions of our study in Section 9.

2 Motivation and Related work

2.1 Motivation

During software development, developers often encounter some software modules with similar values of metrics but have different states of the defect (i.e., defective or non-defective). The SDP datasets collected from these software modules would include many overlapping instances. Prati et al. [52], García et al. [16] and Denil and Trappenberg [15] have corroborated that class overlap is important for classification performance.

Indeed, the SDP models built with classifiers also need high-quality SDP datasets. If the SDP models are trained on class overlap datasets, they may achieve poor performance and interpretation. In SDP, there exist only a few researchers which consider the class overlap as data quality or noise detection. For example, Tang and Khoshgoftaar [63] proposed a clustering-based noise detection method with noise factor metric to remove p% of noise instances of datasets in the NASA corpus. Experimental results indicated that removing noise instances could improve the performance of the Decision Tree (DT) classifier. Kim et al. [33] investigated the impact of noise instances on the performance of SDP models by manually adding different percentages of noise instances. They also proposed a closest list noise identification method to remove noise instances. Meanwhile, there exist two studies that considered both class imbalance and class overlap. For example, Chen et al. [11] proposed the neighborhood cleaning learning approach to only remove the overlapping non-defective instances. Gong et al. [20] proposed an improved K-Means clustering cleaning approach and investigated the impact of different removing methods on both the within-project and cross-project SDP models.

The prior studies still have many unexplored avenues.

First, none of the existing studies investigates the nature of class overlap in the SDP datasets. The four studies [11, 20, 33, 63] only proposed different removing methods to demonstrate that class overlap is harmful to the performance of SDP models. They do not quantify the degrees of overlapping instances and investigate whether class overlap is prominent in the SDP datasets.

Second, the relationship between overlap ratio in the SDP datasets and the performance of SDP models is not clearly established. Kim et al. [33] just analyzed the relationship between the percentage of noise instances and the performance of SDP models by adding different numbers of noise instances. However, the phenomenon of class overlap in the SDP datasets is not clearly represented.

Third, the impact of class overlap on the interpretation of SDP models is still unknown. An important role of SDP models is to help developers understand which software metrics are more likely to cause defects. Such insights could be necessary to improve software quality. However, such an impact of class overlap on the interpretation of SDP models needs to be investigated.

Finally, the prior studies just applied overlapping instances removing techniques to eliminate the impact of class overlap without considering other techniques. In the domain of machine learning, besides removing techniques, there are other techniques (i.e., the separating technique) to eliminate the impact of class overlap on the performance of classifiers. However, none of the existing studies investigates how the separating technique impacts the performance of SDP models.

2.2 Related work

In this subsection, we describe related studies about the SDP models and class overlap techniques.

Studies on the SDP models: The SDP models usually apply machine learning algorithms to mine the rules between software metrics and defect-proneness of software modules, which have been an active research topic in software engineering [25, 59, 60]. In these prior studies, software defect datasets and machine learning algorithms are two important factors. Software defect datasets include metrics that present the nature of software projects (i.e., code metrics and process metrics) and labels that present the defect or non-defect of software modules [1, 5, 24, 83]. In the last few decades, researchers have exposed lots of software defect datasets, such as PROMISE [42], AEEEM [12], SOFTLAB [73], JIRA [80], NASA [58], to name a few. Based on these software defect datasets, some studies [24, 83] have compared the effectiveness of various metrics on identifying the defective modules.

Different machine learning algorithms have been applied to build the SDP models, e.g., support vector machine (SVM) [21], Naive Bayes (NB) [71], Decision tree (DT) [45], K-Nearest Neighbors (KNN), Random Forest (RF) [70], Logistic Regression (LR) [53], and Gradient Boosting Method (GBM) [35]. We herein employ all of these seven classifiers to investigate the impact of class overlap on the SDP models.

Studies on class overlap techniques: Class overlap hinders the performance of classifiers. There are some class overlap handling techniques in the literature [4], e.g., the removing technique and separating technique. The aim of removing techniques is to remove the overlapping instances, while classifiers are trained on the non-overlapping instances. Separating techniques train classifiers on overlapping and non-overlapping instances separately, which can produce two classifiers. Lee and Kim [36] proposed an overlap-sensitive margin (OSM) classifier to separate a dataset into soft- and hard-overlap regions and classify each separated region, respectively. Thus, in our study, we employ both removing and separating techniques to investigate their impacts on the performance of SDP models.

3 Overlapping Instance Identification

Class overlap appears when instances of more than one class share a common region in the data space. The determination of overlapping instances depends on the scope of regions, which means there are no "real" ground truths of overlapping instances. But it does not mean that this range can be expanded indefinitely. Therefore, it is necessary to choose the best method to quantify overlapping instances in the SDP datasets.

Figure 1 visualizes the distribution of instances in projects of prop-5-40 and xalan-2.5, which could prove the occurrence of

TABLE 1: Overview of the studied datasets.

Project	DR(%)	#modules	#metrics	#FACRA	EPV	OR
camel-1.2	35.5	608	20	10	10.8	0.268
derby-10.2.1.6	33.7	1963	54	23	12.2	0.200
derby-10.3.1.4	30.3	2206	54	23	12.4	0.189
eclipse34_debug	24.7	1065	17	10	15.5	0.170
prop1-44	9.2	4081	24	13	15.7	0.059
prop1-92	35.1	3670	24	15	53.6	0.169
prop1-164	9.0	3541	24	15	13.3	0.070
prop2-256	30.9	2025	24	14	26.0	0.153
prop3-318	14.9	2440	24	14	15.2	0.079
prop4-355	32.9	2802	24	14	38.5	0.168
prop5-4	7.5	3514	24	13	11.0	0.057
prop5-40	12.2	3815	24	15	19.4	0.069
prop5-85	26.5	3509	24	15	38.7	0.151
prop5-121	12.3	3445	24	15	17.7	0.096
prop5-157	12.8	2863	24	14	15.3	0.095
prop5-185	8.2	3260	24	13	11.2	0.071
xalan-2.5	48.2	803	20	11	19.4	0.253
xalan-2.6	46.4	885	20	11	20.55	0.210

DR:Defective Ratio.
FACRA:Features After Correlation and Redundancy Analysis.
OR: Overlap Ratio

overlapping instances in the SDP datasets. However, the visualization method cannot be directly used to identify the overlapping instances. As we know, overlapping instances are defined as the instances from different classes (defective or non-defective) with very similar values of metrics. That is to say, they usually reside in overlapping regions in the feature space, which could allow us to identify the overlapping instances by studying the class labels of local neighborhoods of instances. Therefore, to identify overlapping instances for evaluating the popularity and impacts of class overlap in the SDP datasets, we propose a universal method that relies on the natural distribution of instances rather than on a specific classifier.

We propose to employ K-nearest neighborhoods to identify overlapping instances. Based on the above analysis, we identify overlapping instances by comparing the class labels of their K-nearest neighbors. Constructing this type of neighborhood needs both a proper value of K and a distance function. For example, a smaller value (e.g., 1 or 3) may poorly distinguish the nature of instances, while a higher value would be inconsistent with the locality of instances (especially the defective instances are fewer than non-defective instances in the SDP datasets). Chawla et al. [10] applied local analysis to the proposed SMOTE method, where K was set to 5 as typical options. Meanwhile, Napierala and Stefanowski [48] also stayed with k=5 as the appropriate value. In the SDP datasets, Kim et al. [33] found when K was set to 5, classifiers could produce the best values. Thus, in our study, we also use K=5 as the appropriate value. The specific execution process of identifying overlapping instances is shown in Appendix A.

4 Study Design

To answer the RQs, we propose an investigation workflow in Figure 2. We will detail each step as follows.

4.1 Experimental datasets

We select experimental datasets from public software defect repositories, which are widely used in SDP. We collect a total of 230 publicly-available SDP datasets, including 65 datasets from PROMISE [42], 3 datasets from ReLink [33] and Wu et al. [75], 5 datasets from AEEEM [12], 5 datasets from SOFTLAB [73], 32 datasets from JIRA [80], 3 datasets collected by Zimmermann et al. [84], 12 datasets from NASA [58], 78 datasets from Github [72], and 27 datasets from industrial software projects provided by Madeyski and Jureczko [39]. We identify the overlapping instances (see Section 3) and calculate the overlap ratios of these 230 SDP datasets to assess whether class overlap is prominent in the SDP datasets to answer RQ1.

Meanwhile, for RQ2, RQ3, RQ4, and RQ5, we evaluate the performance of SDP models under the effort-aware scenario (the effort required to identify defects), which requires specific number of defects in each software module. Thus, we employ the following two criteria to select datasets (similar to Rajbahadur et al. [53] and Tantithamthavorn et al. [67]):

Criterion 1: Defective ratio$< 50\%$**:** If the defective ratio of a dataset is more than 50%, that is to say, there are more than a half of modules in a project being defective, which would not happen in real software development. Therefore, in our study, we select the datasets whose defective ratios are less than 50% [67, 70].

Criterion 2: Events Per Variable (EPV)> 10**:** EPV [53] is the ratio of the frequency of the least occurring class of the dependent variable to the number of independent variables involved in training the classifiers. Tantithamthavorn et al. [68] demonstrated that the EPV values of datasets affect the performance of SDP models. Meanwhile, for the datasets with low EPV values, the performance of SDP models would be unstable. Therefore, in our study, we select the datasets whose EPV values are larger than 10 (same as Rajbahadur et al. [53]) to assure the stability of results.

As a result, 18 datasets remain that satisfy the two criteria. Table 1 provides a detailed overview of our studied datasets for RQ2, RQ3, RQ4, and RQ5.

4.2 Correlation analysis and redundancy analysis

Correlation analysis: Jiarpakdee et al. [29] and Tantithamthavorn and Hassan [66] point out that correlated metrics impact the interpretation of SDP models. Indeed, correlated metrics would impact the results when performing overlapping instance analysis. Therefore, we need to perform correlation analysis before doing overlap analysis. We follow the same procedures as Rajbahadur et al. [53] and Tantithamthavorn et al. [68] to perform this analysis.

Redundancy analysis: Except for correlated metrics, there exist redundant metrics interfering with each other [81]. To eliminate the impact of redundant metrics, we also remove redundant metrics before doing overlapping instance analysis. We perform redundant analysis the same as Rajbahadur et al. [53] and Tantithamthavorn et al. [68].

4.3 Label overlapping instances

To analyze the nature and impacts of class overlap in the SDP datasets, we should firstly identify the overlapping instances regardless of defective or non-defective in the SDP datasets. We label overlapping instances using our proposed method described in Section 3. Meanwhile, We also compare the overlapping instances identified by our method with K-Means [20], Soft Margin Ratio (SMR) [22], and Support Vector Data Description (SVDD) [76] methods to validate the effectiveness of our proposed method. We follow the detailed execution procedures of these comparison methods to implement them for comparison. KNN and K-Means are independent of any classifiers, while SMR and SVDD are

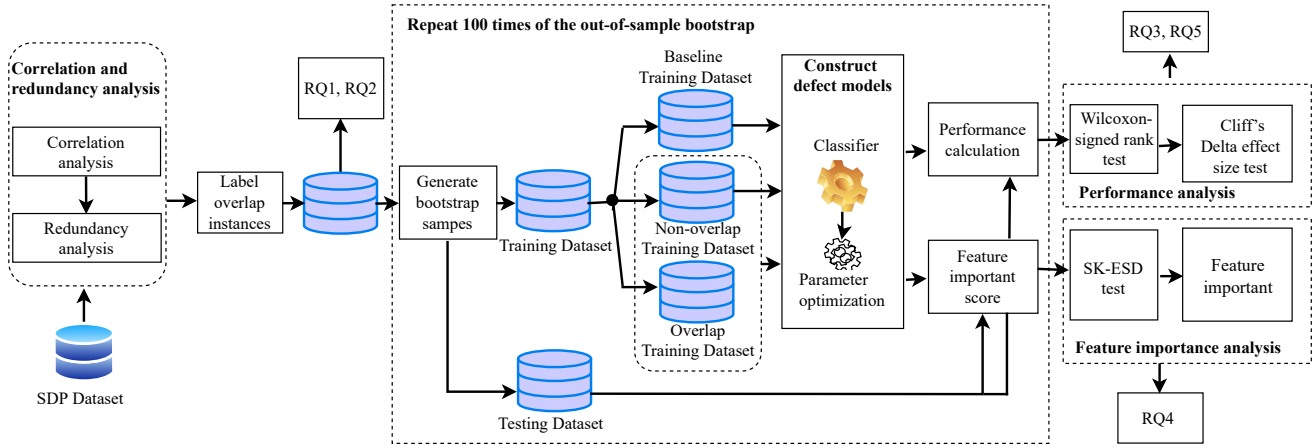


Fig. 2: Overview of our study design.

TABLE 2: Studied classifiers and their hyperparameters.

Abbr	Classifier	Python	Parameters	Range
SVM	Support Vector Machine	svc	C	(0.0, 10)
			kernel	['linear', 'poly', 'rbf', 'rbf', 'sigmoid']
			gamma	(0.0, 10)
NB	Naive Bayes	GaussianNB		
DT	Decision tree	DecisionTree Classifier	criterion	["gini", "entropy"]
			max_depth	[1,15]
			min_samples_leaf	([1,5]]
KNN	K-Nearest Neighbors	KNeighbors Classifier	n_neighbors	[1,20]
RF	Random Forest	RandomForest Classifier	n_estimators	[30,500]
			max_depth	[1,15]
LR	Logistic Regression	Logistic Regression	C	(0, 4)
			penalty	["l1", "l2"]
GBM	Gradient Boosting Method	GradientBoosting Classifier	n_estimators	[30,500]
			max_depth	[1,15]
			max_features	[1,np.sqrt(no.of features)]

dependent on the SVM classifier. The specific execution process of compared identifying overlapping instances methods are shown in Appendix B.

4.4 Generating bootstrap samples

In our study, since the out-of-sample bootstrap validation [68] technique leverages aspects of the statistical inference [9], we use this technique for each studied dataset to ensure the robust conclusions of SDP models. The out-of-sample bootstrap validation technique produces the training dataset with the same size as the original dataset by randomly sampling instances with replacements. Efron [9] pointed out that about 36.8% of instances would not appear in the bootstrap sample. Therefore, the non-sampled instances are set aside as the test dataset. We repeated 100 times of the out-of-sample bootstrap process to produce 100 combinations of the train and test datasets.

4.5 Constructing defect models

Class overlap handling techniques: To investigate the impact of class overlap on the SDP models, we apply two class overlap techniques (i.e., Removing and Separating as described in Section 2.2). It should be noted that in these two class overlap handling techniques, we directly apply our proposed KNN method to identify overlapping instances in the training dataset, which could avoid the leakage of the test instances. The specific execution process of removing technique and separating technique are shown in Appendix C.

Studied Classifiers: Many classification techniques have been applied to build the SDP models [18, 23]. Therefore, to ensure the generalizability and applicability of our results, we select the classifiers which are widely used. As a result, we construct our SDP models with seven classifiers, including Support Vector Machine (SVM) [21], Naive Bayes (NB) [71], Decision Tree (DT) [45], K-Nearest Neighbors (KNN), Random Forest (RF) [70], Logistic Regression (LR) [53], and Gradient Boosting Method (GBM) [35]. Meanwhile, prior studies [2, 67] pointed out automated parameter optimization affected the performance of classifiers. Thus, we apply `Hyperopt` from the pypi[2] Python package to tune the hyper-parameters of these seven studied classifiers. Table 2 lists the studied classifiers and python package used in our study.

4.6 Performance calculation

To investigate the impact of class overlap on the performance of SDP models, we evaluate both the Defect-classification models and Effort-aware models.

Evaluate Defect-classification models: Defect-classification models aim to detect whether a software module is defective or non-defective. We employ threshold-independent performance measures of Area Under the Reciever-operator Characteristic Curve (AUC) [37], Brier score [57], and threshold-dependent performance measure of Recall, False alarm to evaluate the performance of Defect-classification models. High values in terms of AUC and Recall indicate good performance, while a low value in terms of Brier-score and False alarm indicates good performance. We refer the readers to the study by Ni et al. [50], Tantithamthavorn et al. [70] for a more detailed explanation of these performance measures.

Evaluate Effort-aware models: Effort-aware models aim to consider the effort required to identify defects. In our study, we employ P_{opt} [31, 79] to evaluate the Effort-aware models. P_{opt}

2. https://pypi.org/project/hyperopt/

is the normalized version of effort-aware performance measures provided by Mende and Koschke [41]. The specific definition of P_{opt} is shown in Appendix D.

Performance analysis: To estimate whether there are statistical significance differences between compared SDP models, we apply the Wilcoxon-signed rank test [61] to statistically analyze the performance between compared SDP models at a confidence level of 95%. Moreover, since we perform multiple pairwise-comparisons, we also correct the obtained p-values from the Wilcoxon-signed rank test with Benjamini-Hochberg correction [6, 7] to control for false positives. Meanwhile, to further estimate the differences between compared SDP models, We then apply the Cliff's Delta effect size test [78] to measure the effect size. We do so since Wilcoxon-signed rank test and Cliff's Delta effect size test do not assume the data is normal [14, 53]. Note that to avoid some potential that differences between approaches are overestimated, we do the statistical significance analysis on the median values not directly on the 100 out-of-sample bootstrap iterations. For each performance measure, we first compute the median value of 100 values of each dataset generated by 100 bootstrap iterations. We then conduct the Wilcoxon-signed rank test with Benjamini-Hochberg correction and the Cliff's Delta effect size test (0.147 (small), 0.33 (medium), and 0.474 (large)) on the median values of the 18 studied datasets at a confidence level of 95%. Therefore, if the adjusted $P - value < 0.05$, we consider there are statistically significant differences between the compared results. Meanwhile, if the Cliff's Delta ≥ 0.33, we consider there is a greater than medium effect size between the compared results.

4.7 Feature importance analysis

To investigate the interpretation of SDP models, we calculate the importance of each feature, rank features based on their scores, and calculate the evaluation metrics to analyze feature importance of seven classifiers trained on original, non-overlap, and overlap datasets.

Calculate the feature important score: Similar to Rajbahadur et al. [53] and Tantithamthavorn et al. [69], we apply the permutation feature importance [3] to evaluate the importance of features, since it can be applied to any classifier. We refer the readers to the study by Tantithamthavorn et al. [69] for a more detailed explanation of calculating the permutation feature importance. In our study, we apply `PermutationImportance` from the Scikit-learn[3] Python package to calculate the feature important score of each feature in each dataset.

Rank feature: After obtaining the importance of each feature in each dataset, we apply the improved Scott-Knott Effect Size Difference (SK-ESD) [64] to obtain the rank of each feature that is the same as Rajbahadur et al. [53] and Tantithamthavorn et al. [69]. Thus, we can generate three feature important rank lists for all the seven classifiers.

Calculate the evaluation metrics: After generating the three feature important rank lists, we measure their differences to evaluate how much they agree with each other. Thus, we use three evaluation metrics of Top K overlap [54], where K is set to 1, 2, and 3.

3. https://eli5.readthedocs.io/en/latest/blackbox/permutation_importance.html

Top K overlap: This evaluation metric analyzes features that exist at the top k ranks in both the compared rank lists. The formula is defined as

$$Top \quad K \quad overlap = \frac{\bigcap_{i\geq 2}^{n} Features \quad at \quad top \quad k \quad ranks}{\cup_{i\geq 2}^{n} Features \quad at \quad top \quad k \quad ranks} \tag{1}$$

where n is the number of compared rank lists. The features at the top ranks would be more important than others in distinguishing defective modules from non-defective modules [38]. Therefore, in our study, we select K=1, 2, and 3 to calculate Top 1 overlap, Top 2 overlap and Top 3 overlap, which is similar to Rajbahadur et al. [53] and Tantithamthavorn et al. [69]. Note that for Top 1 overlap, Top 2 overlap, and Top 3 overlap, a higher value indicates a lower disagreement.

5 Results

In this section, we present the results of the following five RQs.

5.1 (RQ1) How effective our proposed method is in identifying overlapping instances in the SDP datasets?

Approach. Correct quantization of overlapping instances is a prerequisite for studying the impact of class overlap on the performance and interpretability of SDP models. Therefore, in our study, one quantification method is to apply the same overlapping instances identified by these different methods. In this case, overlapping instances identified by other methods are treated as "ground truths" that can help us measure the overlapping instances identified by different methods. If more overlapping instances identified by a method are also identified by another method, the overlapping identification method would be better.

We compare overlapping instances identified by our method with the K-Means, SMR, and SVDD algorithms (refer to Section 4.3). Specifically, we first remove the correlated and redundant metrics in each SDP dataset. We then identify overlapping instances by our method and the comparison methods respectively. Finally, for each method, we calculate the percentages of overlapping instances that are also identified by other methods.

Furthermore, to validate the effectiveness of our proposed method, we compare the performance of the SDP models trained on removing overlapping instances identified by each method. Meanwhile, since different overlapping identification methods would identify different overlapping instances, we also compare the combining method that considers the instance identified by at least three of the four methods (KNN, K-Means, SMR, and SVDD) to be an overlapping instance. For each method, we repeated 100 times of the out-of-sample bootstrap process to produce 100 combinations of the train and test datasets (refer to Section 4.4). For each train and test dataset, we use the Hyperopt method to tune the hyperparameters of the RF classifier since RF could achieve the highest performance. In addition, we calculate the Wilcoxon-signed rank test and Cliff's Delta effect size test between the models trained on original datasets and datasets removing overlapping instances identified by each of compared methods followed by by performance evaluation in Section 4.6.

Results. Figure 3 presents the boxplot of the percentages of the same overlapping instances identified by any two methods across 230 SDP datasets. In this figure, each facet (such as K-Means) represents the proportions of overlapping instances identified by K-Means that are also identified by other methods (i.e., KNN,

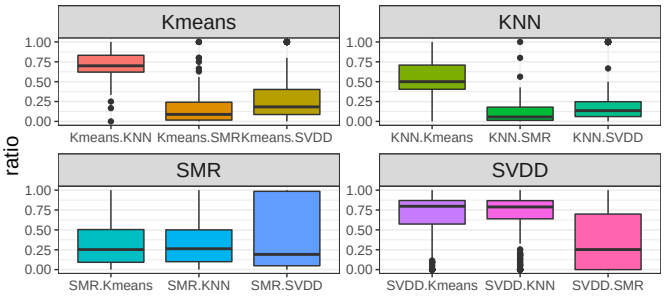


Fig. 3: The percentages distribution of the same overlapping instances identified by any two methods across 230 SDP datasets.

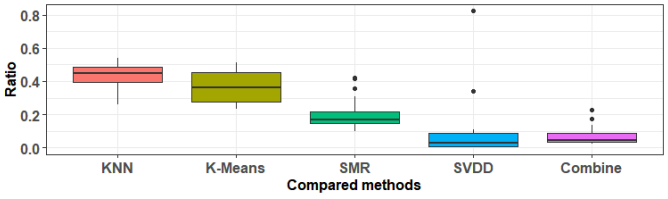


Fig. 4: The Boxplot of the proportion of overlapping instances identified by different methods in the instances distributed in the boundary of SDP models.

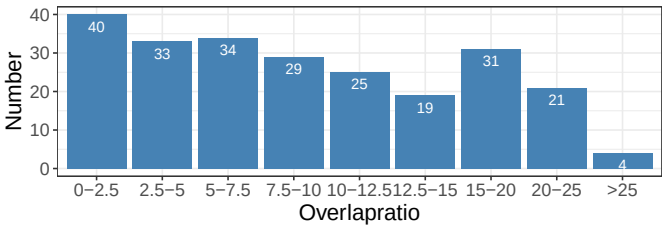


Fig. 5: The histogram of the class overlap ratio of the 230 publicly-available SDP datasets.

SVDD, SMR) to the total overlapping instances identified by K-Means. Tables 7 and 8 in Appendix E present the median values of SDP models trained on removing overlapping instances identified by different methods. Based on these figures, we can obtain the following observations.

Observation 1) Our proposed method employing K-nearest neighborhoods could identify more precise overlapping instances compared with the K-Means, SMR, and SVDD methods. From Figure 3, we can observe that for most of our studied 230 SDP datasets, our proposed method (KNN) could identify 70% (or 75%) of overlapping instances identified by K-Means (or SVDD). On the contrary, the K-Means (or SVDD) method only identifies 50% (or 25%) overlapping instances identified by our proposed KNN method.

On the one hand, compared with the KNN and K-Means methods that are independent of any classifiers, the SMR and SVDD methods are dependent on the SVM classifier. Since the SVM classifier is not so good for software defect prediction that could not identify defective modules [17], the overlapping instances identified by SMR and SVDD are not precise, which could be proved in Figure 3.

On the other hand, we hypothesize that our proposed KNN method can identify more precise overlapping instances since the KNN method identifies more overlapping instances distributed in the boundary of SDP models by analyzing the class distribution in the local neighborhood of a given instance. We do it since instances with different label distribution in the boundary of SDP models are the main reason for the poor performance of the SDP models. To do so, we first apply the method proposed by Migut et al. [44] to identify the decision boundaries of SDP models. We then use the Euclidean distance to identify the instances distributed in the boundary of SDP models. We finally calculate the proportion of overlapping instances identified by different methods in the instances distributed in the boundary of SDP models in Figure 4. From Figure 4, we can observe that our KNN method can identify more overlapping instances distributed in the boundary of SDP models. Therefore, KNN can identify more precise overlapping instances compared with the other methods.

Observation 2) SDP models trained on datasets removing overlapping instances identified by KNN method achieve the highest values in terms of AUC and P_{opt}. From Tables 7 and 8 in Appendix E, we observe that SDP models trained on removing overlapping instances identified by KNN method achieve the highest values of 12/18 projects in terms of AUC and 15/18 projects in terms of Popt. Furthermore, we can observe that there are statistically differences in terms of AUC, Recall, and Popt between models trained on original datasets and datasets removing overlapping instances by KNN method. Furthermore, we observe that if overlapping instances are not well defined, it may cause performance degradation. Therefore, we claim that our proposed method can identify more precise overlapping instances and can be thought of as a benchmark.

Summary

Our proposed method employing KNN could identify more precise overlapping instances in SDP datasets than existing methods. Furthermore, SDP models trained on datasets removing overlapping instances identified by KNN method improve the AUC, Recall, and Popt performance. Therefore, we recommend researchers and practitioners to use our proposed KNN method to identify overlapping instances before they build their SDP models.

5.2 (RQ2) How overlapped are the SDP datasets?

Approach. To investigate the popularity of class overlap in SDP datasets, we analyze the ratios of overlapping instances in 230 publicly-available SDP datasets. First, we remove the correlated and redundant metrics in each dataset. Then, we use our proposed overlapping instances identification method to label the overlapping instances in each dataset. Finally, we calculate the class overlap ratio of each dataset.

Results. Figure 5 reports the histogram of the class overlap ratio of these 230 SDP datasets. Based on this figure, we have the following observation.

Observation 3) Class overlap is prominent in SDP datasets. From Figure 5, we can observe that SDP datasets have various levels of class overlap ratios. 70.0% of SDP datasets have a class overlap ratio above 5%. Indeed, 38.3% of SDP datasets have a class overlap ratio between 5% and 12.5%. Meanwhile, 32.6% of SDP datasets have a class overlap ratio higher than 12.5%.

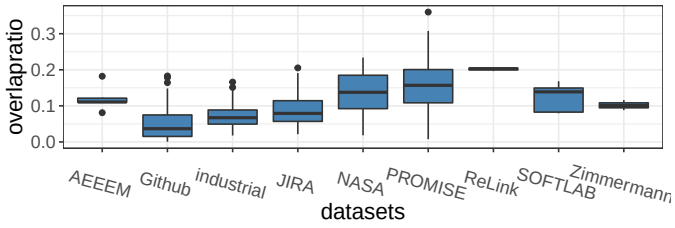


Fig. 6: The boxplot of the distribution of the class overlap ratio of defect prediction data in each dataset.

Furthermore, we analyze the distribution of class overlap ratio in different types of SDP datasets, i.e., PROMISE, ReLink, AEEEM, SOFTLAB, JIRA, Zimmermann, NASA, Github, and industrial software. As shown in figure 6, we observe that the frequently-used SDP datasets (e.g., NASA, AEEEM, PROMISE, ReLink, and SOFTLAB) have more than 12.5% of overlapping instances, which motivates us to pay attention to the impact of class overlap when constructing SDP models.

Summary

Class overlap is ubiquitous in SDP datasets. Especially the frequently used datasets (i.e., NASA, AEEEM, PROMISE, ReLink and SOFTLAB) have more than 12.5% overlapping instances. Therefore, we recommend that researchers and practitioners should pay attention to the class overlap issue when they construct a study on software defect prediction.

5.3 (RQ3) How do different levels of class overlap impact the performance of defect prediction models?

Approach. To better answer RQ3, we generate a series of training datasets with different overlap ratios by flipping the label of instances. To generate training datasets with different class overlap ratios, we firstly apply the 100-out-of-bootstrap approach to generate 100 pairs of training and testing datasets for each studied dataset. We then randomly sample defective and non-defective instances with the range of $\{10\%, 20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%, 90\%\}$ for each training dataset, respectively. For each selected instance, we apply the Euclidean Distance to find three nearest neighbors and flip the labels of three nearest neighbors relying on the rule, if they are not the same as the label of the selected instance. Finally, we repeat 10 times for the two steps to generate 1000 ($100 * 10$) training datasets for each test dataset. In this RQ, we choose camel-1.2, derby-10.3.1.4, and prop-1-92 as our experimental datasets, since these three datasets have enough defective instances to generate different overlapping ratios and span different fields.

After generating training datasets with different overlap ratios, we train SDP models built by 7 classifiers (described in Section 4.5) on these generated training datasets. For each SDP model, we could obtain 1000 results for each studied project. Finally, we analyze the correlation between the measured performance and different class overlap ratios calculated by our proposed method in Section 3.

Results. Figure 7 presents the performance of SDP models trained on the camel-1.2 project with different levels of class overlap.

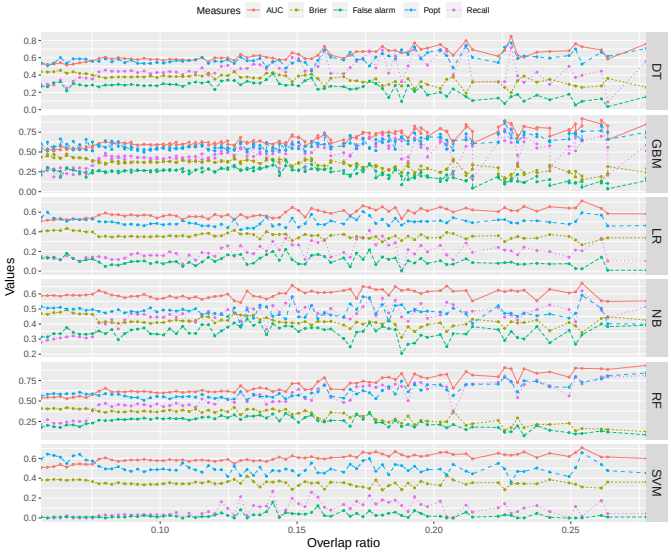


Fig. 7: Performance of SDP models trained on the camel-1.2 project with different levels of class overlap.

Meanwhile, the performance of SDP models trained on the derby-10.3.1.4 and Prop-1-92 projects with different levels of class overlap can be found in Appendix E, respectively. We can obtain the following observations from these figures.

Observation 4) Different levels of class overlap ratios have different effects on the performance of SDP models. Especially the class overlap ratio that is over 12.5% seriously affects the stability of the performance of SDP models. From Figures 7, 17, and 18 in Appendix E, we can observe that these 7 SDP models trained on training datasets with the range of 7.5%-12.5% overlap ratio would achieve relatively stable performance values in terms of all studied measures. However, when the overlap ratio is larger than 12.5%, all of these 7 models generate very turbulent performance values in terms of all the studied measures. For example, the RF classifier trained on the camel-1.2 dataset with a 25% overlap ratio only achieves the AUC value of 0.51, while the RF classifier trained on the camel-1.2 dataset with 25.6% would achieve the AUC value of 0.798. That is to say that, the performance of SDP models trained on datasets with a higher than 12.5% overlap ratio is not stable. Thus, the stability of the performance of SDP models is threatened by the class overlap ratio, especially when the overpal ratio is above 12.5%.

Observation 5) The number of instances in training datasets could affect the stability of the performance of SDP models trained on training datasets with above 12.5% overlap ratio. From Figures 7, 17, and 18 in Appendix E, we can observe that the fluctuation of the performance of SDP models trained on derby-10.3.1.4 and prop-1-92 are smaller than the performance trained on camel-1.2 when the overlap ratio is higher than 12.5%. For example, for the camel-1.2 project with 608 instances, the disparities between the maximum and minimum performance values of SDP models are 0.748, 0.494, 0.399, and 0.501 in terms of Recall, AUC, Brier-score, and Popt, respectively, while for prop-1-92 (derby-10.3.1.4) with 3670 (2206), the disparities between the maximum and minimum performance values of SDP models are 0.717 (0.741), 0.316 (0.397), 0.251 (0.272), and 0.326 (0.432) in terms of Recall, AUC, Brier-score, and Popt, respectively. That is

to say that, when the overlap ratio is similar, increasing the number of instances would help to improve the stability of performance.

Furthermore, we apply the 100-out-of-sample bootstrap approach to train SDP models built by 7 classifiers on the selected original 18 datasets (described in 4.1). Then, we analyze the disparity between the maximum and minimum values in the 100 results of each dataset. Figure 8 presents the distribution of the disparity between the maximum and minimum values of SDP models trained on original datasets.

From Figure 8, we could observe that the disparity of the maximum and minimum values of SDP models trained on datasets with an overlap ratio above 12.5% is much larger than that of SDP models trained on datasets with overlap ratio ranging from 7.5% to 12.5%. Meanwhile, the larger number of instances in training datasets, the more stable the performance of SDP models. For example the projects prop-4-355 (overlap ratio :16.8%, No.: 2802), prop-1-92 (overlap ratio :16.9%, No.: 3670), and eclipse34_debug (overlap ratio :17.0%, No.: 1065):

1) Comparing prop-1-92 and prop-4-355 with the same overlap ratio but different number of instances, the disparity of studied SDP models trained on prop-1-92 is smaller than that of studied SDP models trained on prop-4-35 in terms of studied measures.

2) Comparing prop-1-92 and eclipse34_debug with the similar overlap ratio but different number of instances, the disparity of studied SDP models trained on prop-1-92 is much smaller than that of studied SDP models trained on eclipse34_debug in terms of studied measures.

Therefore, the disparity of SDP models trained on original SDP datasets further prove that the overlap ratio (especially above 12.5%) and number of instances would seriously affects the stability of the performance of SDP models. Researchers and practitioners should check the overlap ratio of their SDP datasets and repeatedly run their models to prevent unstable results and final conclusion.

Summary

The class overlap ratio that is over 12.5% seriously affects the stability of the performance of SDP models. Furthermore, the number of instances in training datasets would affect the stability of the performance of SDP models trained on the datasets with a similar overlap ratio. Therefore, we recommend researchers and practitioners should check the overlap ratio of their SDP datasets and repeatedly run their models to prevent unstable results and findings.

5.4 (RQ4) How does class overlap impact the interpretation of defect prediction models?

Approach. To answer RQ4, we first apply the permutation importance to calculate the 100 important scores for each feature of the 7 studied classifiers on each of the studied 18 datasets with original and non-overlap datasets with the 100-out-of-sample bootstrap to avoid uncertainty [43]. Then, we use the SK-ESD test to rank and obtain a feature list of each dataset on each classifier. Finally, to estimate the impact of overlapping instances on the interpretation of SDP models, we compute the Top K overlap (K=1, 2, and 3) (described in Section 4.7) in the ranks of features that appear in the original and non-overlap datasets.

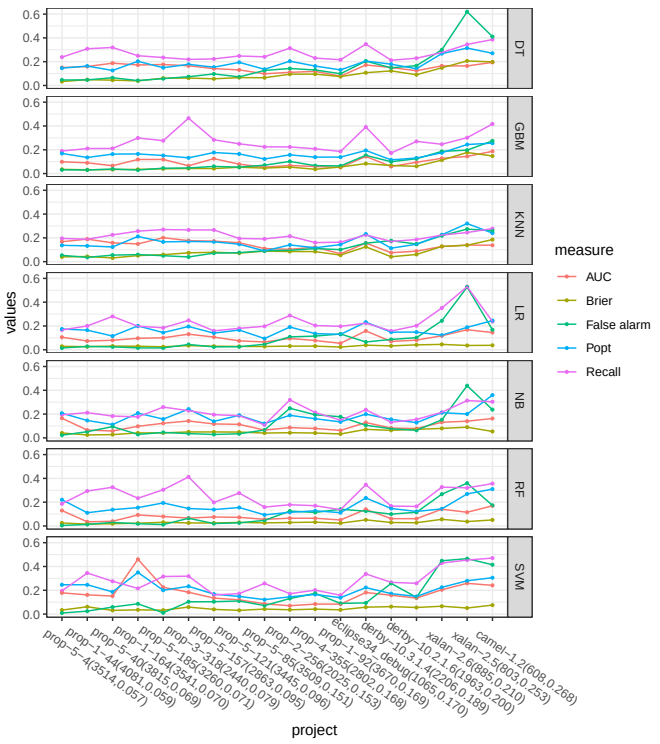


Fig. 8: Distribution of disparities between the maximum and minimum values of SDP models trained on original datasets (Projects are labeled according to their overlap ratios. Note that The x-axis is in ascending order of overlap ratio of datasets.

Furthermore, it is valuable to check whether the different releases for the same projects have the same metrics at the Top -1 rank for future defect prediction work. To do so, we first get the derby, prop1, prop2, prop5, and xalan projects in Table 1 that have different releases. We then compute the Top-1 overlap (refer to Eq. 1) among different releases of the same project on each of the studied classifiers.

Results. Tables 13, 14, and 15 in Appendix E present the important features in TOP-1, 2 and 3 ranks with the studied seven classifiers for each of the studied SDP datasets generated by Section 4.7. Figure 9 shows the density plot of Top K overlap (K=1, 2, and 3) for each of 7 studied classifiers trained on original and non-overlap datasets. In Figure 9, each density plot is a representation of the distribution of the Top K overlap on Top k feature lists between original and non-overlap datasets generated by Section 4.7. For instance, considering the density plot with DT classifier on the Top-1 overlap as an example, except for prop-1-164 and prop-5-4, the other 16 SDP datasets are clustered at the position of 1.0, since these datasets get the Top-1 overlap value of 1. Based on these results, we can obtain the following observations.

Observation 6) Features at the top 1 rank in original and non-overlap datasets have a strong agreement, features at the top 2 rank in original and non-overlap datasets have a partial agreement, while features at the top 3 rank vary considerably. From Figure 9, we can observe that all of the 7 studied classifiers have a strong agreement on the importance feature list at the Top 1 rank since the top 1 overlap values of most datasets on these 7 classifiers are almost 1. In addition, 4/7 studied classifiers have a strong agreement on the important feature lists of 9/18 datasets at

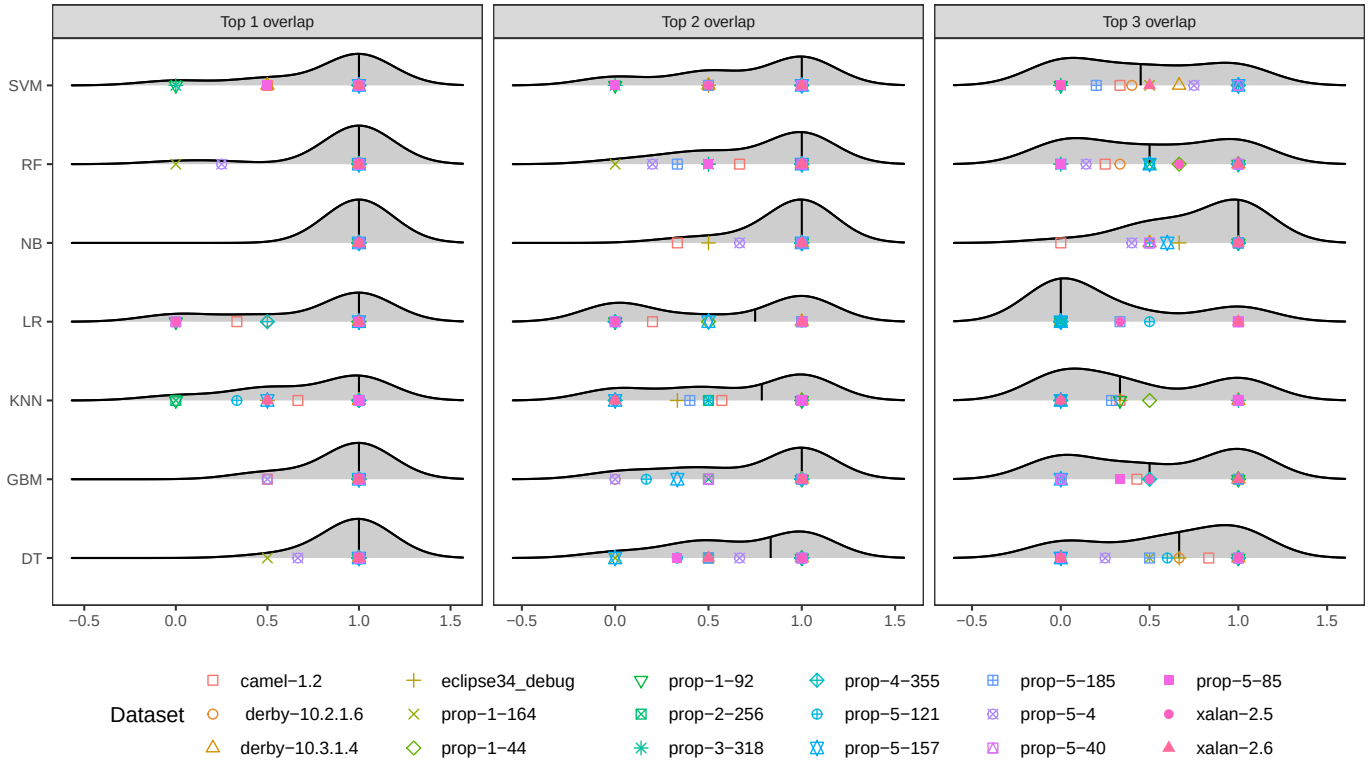


Fig. 9: A density plot of Top K overlap (K=1, 2, and 3) for each studied classifier trained on original and non-overlap datasets.

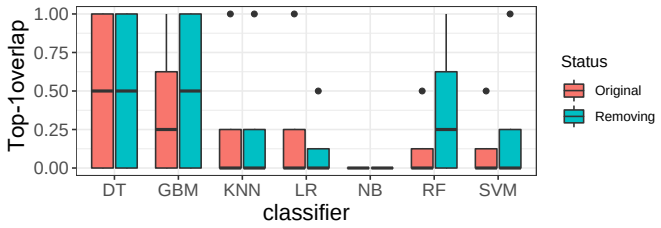


Fig. 10: The distribution of Top 1 overlap among different releases of same projects for each studied classifier trained on original and non-overlap datasets.

the Top 2 ranks. Meanwhile, only 1/7 of the studied classifier has a strong agreement on the important feature lists at the Top 3 ranks. Furthermore, from Tables 13, 14, and 15 in Appendix E, we can observe that for the same classifiers, despite the high agreement on the top-1 and 2 important features, the ranking of the important features would vary between models trained on original datasets and models trained on removing overlapping instances. That is to say, overlapping instances can change the important feature rank of SDP models, which shift the interpretation of SDP models.

Observation 7) Class overlap has different impacts on the interpretation of SDP models constructed by different classifiers. From Figure 9, we can observe that the NB classifier has a strong agreement on important feature lists at all Top-1, Top-2 and Top-3 ranks, while the LR classifier has no agreement on the important features of most datasets at top2 and top 3 ranks. In addition, for RF, GBM, SVM, DT, and KNN classifiers, the levels of agreement between original and non-overlapping instances are also different at top 1, top 2, and top 3. That is to say, the impact of class overlap on the interpretation of SDP models depends on the studied classifiers.

Observation 8) For a given SDP dataset, it is easier to find agreement feature lists among different classifiers after removing the overlapping instances. From Tables 13, 14, and 15 in Appendix E, we can observe that after removing the overlapping instances, features at the Top-1 and 2 ranks in non-overlap datasets with different classifiers have a strong agreement. Consider Prop-4-355 as an example, after removing overlapping instances, the top -1 features for the studied seven classifiers are fixed on amc, the top-2 features for the studied seven classifiers are fixed on mfa. That is to say, removing overlapping instances could help to find more consistent guiding significance metrics for future defect prediction work.

Observation 9) Compared with original datasets, GBM, RF and SVM classifiers trained on datasets with overlapping instances removed can identify more of the same metrics at the Top-1 rank among the different releases for the same projects. From Figure 10, we can observe that after removing the overlapping instances, the median top-1 overlap of GBM improves from 0.25 to 0.5, while the median top-1 overlap of RF improves from 0.0 to 0.25. Therefore, we recommend that researchers and practitioners should remove the overlapping instances to find the same important metrics when they want to build SDP models on the cross-version context.

Summary

Class overlap has different impacts on interpretation of SDP models constructed by different classifiers, especially for the metrics listed at the Top-2 and 3 ranks. Meanwhile, removing overlapping instances could help to find more consistent guiding significance metrics for a given dataset and different releases of a given project. Therefore, we recommend that researchers and practitioners should remove the overlapping instances to find the more consistent guiding significance metrics.

5.5 (RQ5) How do different class overlap handling techniques impact the performance of defect prediction models?

Approach. To better answer RQ5, we firstly apply two class overlap handling techniques (i.e., removing and separating in Section 4.5) to handle overlapping instances. Specifically, we firstly divide the dataset into ten datasets with over 12.5% overlap ratio and eight datasets with lower than 12.5% overlap ratio. We then compare SDP models with two class overlap handling techniques with SDP models trained on original datasets on the studied seven classifiers in Section 4.5 for two sets of studied datasets.

In addition, we also apply two unsupervised learning methods (CLA and CLAMI [46]) which do not consider the label information to build SDP models, since Xu et al. [77] recently observed that CLA and CLAMI [46] achieve competitive performance compared with typical supervised models. Therefore, we can obtain 23 ($7 \times 3 + 2$) SDP models and evaluate the performance of these 23 SDP models by performance evaluation in Section 4.6.

Results. Table 3 reports the AUC results of the 23 SDP models based on different class overlap handling techniques for each of the studied SDP datasets, respectively. Meanwhile, Table 3 also lists the statistical results for the studied SDP datasets based on different handling techniques compared with models trained on original datasets. Note that for the CLA and CLAMI methods, Table 3 presents the statistical results for studied SDP datasets between models trained on the studied seven classifiers and models with CLA and CLAMI methods. Furthermore, Appendix E presents the other evaluating measures (i.e., Recall, Brier-score, False alarm, and P_{opt}) of 23 SDP models based on different class overlap handling techniques. According to these tables, we can achieve the following observations.

Observation 10) Class overlap handling techniques have different effects on the performance of SDP models trained on datasets with different overlap ratios. Class overlap handling techniques could improve the performance of SDP models trained on datasets with over 12.5% overlap ratios. From Table 3 and Tables 9, 10, 11 and 12 in Appendix E, we can observe that for SDP datasets with over 12.5% overlap ratio, SDP models trained on datasets handled by removing or separating techniques outperform the SDP models trained on original datasets on 7 (6, 4, 5, 7) out of 7 studied classifiers in terms of AUC (Recall, Brier-score, False alarm and P_{opt}). For the KNN, LR, SVM, GBM, and RF classifiers, there are statistically significant different improvements in terms of different evaluated measures. On the contrary, for SDP datasets with lower than 12.5% overlap ratio, SDP models trained on datasets handled by removing and separating techniques do not outperform the SDP models trained on original datasets. Especially for the DT and NB classifiers, the performance in terms of AUC and False alarm are lower than the SDP models trained on original datasets.

Observation 11) Though there are some overlapping instances in SDP datasets, SDP models trained on datasets handled by removing and separating techniques outperform the SDP models trained on datasets without labels (i.e., CLA and CLAMI methods). From Tables 3 and Appendix E, we can observe that SDP models with the studied seven classifiers trained on datasets handled by both removing and separating techniques can obtain statistically significant different performance compared with CLA (or CLAMI) method in terms of AUC, Brier-score, False alarm and p_{opt} (AUC, Recall and Brier-score). Furthermore, the values of cliff's delta are more than median. Therefore, we recommend researchers and practitioners still apply supervised learning methods with class overlap handling techniques to build software defect prediction models.

Observation 12) The RF classifier can obtain better performance values both on the original datasets and on the datasets handled by removing or separating techniques. From Table 3 and Tables 9, 10, 11 and 12 in Appendix E, we can observe that SDP models with the RF classifier can obtain the best performance values for 18 (5, 18, 11, and 3) out of the 18 studied SDP datasets in terms of AUC (Recall, Brier-score, False alarm, and P_{opt}). Even if Recall and P_{opt} are not the best values, they are still relatively good values, especially for SDP models with the RF classifier trained on datasets handled by removing technique. Therefore, even on SDP datasets with overlapping instances, we recommend researchers and practitioners to apply the RF classifier to build SDP models.

Summary

Class overlap handling techniques could statistically significantly improve the performance of SDP models trained on datasets with over 12.5% overlap ratios. The RF classifier can obtain better performance values both on the original datasets and on the datasets handled by removing or separating techniques. Therefore, we recommend researchers and practitioners to apply the RF classifier with removing technique to build SDP models.

6 Discussion

In Section 5, we observe that overlapping instances are ubiquitous in SDP datasets and affect the performance and interpretation of SDP models. Our proposed overlapping identification method can identify more precise overlapping instances to improve the performance of SDP models. However, our proposed overlapping identification method may be affected by parameters (i.e., the proportion of neighbors, distance method, and clustering method). In this section, we study the impact of different parameters on identifying overlapping instances and the performance of SDP models.

6.1 The Impact of parameter proportion of neighbors (P)

In this experiment, we change the value of the parameter proportion of neighbors (p) in the range of 3, 4, 5 to investigate the impact

TABLE 3: Comparisons between different class overlap handling techniques on 18 datasets in terms of AUC.

Overlapping	Project	Handling techniques	DT	NB	KNN	LR	SVM	GBM	RF	CLA	CLAMI
$R \geq 12.5\%$	Camel-1.2	Original	0.553	0.565	0.588	0.550	0.542	0.589	0.616	0.545$(L)^{***}$	0.521$(L)^{***}$
		Removing	0.595$(M)^{**}$	0.577$(M)^{***}$	0.602**	0.589$(L)^{***}$	0.602$(M)^{***}$	0.635$(L)^{***}$	**0.682**$(L)^{***}$		
		Separating	0.594$(M)^{**}$	0.568$(M)^{***}$	0.605**	0.580$(M)^{***}$	0.603$(M)^{***}$	0.627$(L)^{***}$	0.680$(L)^{***}$		
	derby-10.2.16	Original	0.678	0.704	0.685	0.766	0.748	0.735	0.774	0.684$(L)^{***}$	0.535$(L)^{***}$
		Removing	0.727$(M)^{**}$	0.707	0.749$(L)^{***}$	0.771*	0.763$(M)^{***}$	0.791$(L)^{***}$	**0.794**$(L)^{***}$		
		Separating	0.721$(M)^{**}$	0.678	0.734$(L)^{***}$	0.757	0.753	0.779$(L)^{***}$	0.790$(L)^{***}$		
	derby-10.3.1.4	Original	0.676	0.716	0.692	0.769	0.761	0.769	0.748	0.678$(L)^{***}$	0.546$(L)^{***}$
		Removing	0.724$(M)^{*}$	0.719	0.752$(L)^{***}$	0.773	0.760	0.790$(L)^{***}$	**0.798**$(L)^{***}$		
		Separating	0.721$(M)^{**}$	0.684	0.743$(L)^{***}$	0.766	0.754	0.782$(L)^{***}$	0.783$(L)^{***}$		
	eclipse34_debug	Original	0.623	0.746	0.666	0.689	0.687	0.714	0.780	0.644$(L)^{***}$	0.533$(L)^{***}$
		Removing	0.654$(M)^{***}$	0.741	0.698**	0.710*	0.712$(M)^{***}$	0.749$(L)^{***}$	**0.794**$(M)^{***}$		
		Separating	0.651$(M)^{**}$	0.687	0.697**	0.714$M)^{**}$	0.708*	0.746$(L)^{***}$	0.786		
	prop-1-92	Original	0.752	0.711	0.778	0.775	0.790	0.850	0.836	0.509$(L)^{***}$	0.527$(L)^{***}$
		Removing	0.786$(M)^{**}$	0.714	0.813$(M)^{***}$	0.781*	0.818$(L)^{***}$	0.869$(L)^{***}$	**0.888**$(L)^{***}$		
		Separating	0.779**	0.710	0.806$(M)^{***}$	0.780*	0.813$(L)^{***}$	0.863$(L)^{***}$	0.873$(L)^{***}$		
	prop-2-256	Original	0.797	0.724	0.828	0.769	0.755	0.878	0.865	0.514$(L)^{***}$	0.508$(L)^{***}$
		Removing	0.814*	0.740$(M)^{***}$	0.838*	0.813$(L)^{***}$	0.861$(L)^{***}$	0.895$(L)^{***}$	**0.920**$(L)^{***}$		
		Separating	0.785	0.733$(M)^{**}$	0.843*	0.819$(L)^{***}$	0.872$(L)^{***}$	0.901$(L)^{***}$	0.876*		
	prop-4-355	Original	0.795	0.716	0.801	0.760	0.774	0.874	0.839	0.513$(L)^{***}$	0.516$(L)^{***}$
		Removing	0.773	0.717	0.818$(M)^{**}$	0.801$(L)^{***}$	0.826$(L)^{***}$	0.881*	**0.900**$(L)^{***}$		
		Separating	0.768	0.705	0.822$(M)^{**}$	0.801$(L)^{***}$	0.829$(L)^{***}$	0.882$(M)^{***}$	0.890$(L)^{***}$		
	prop-5-85	Original	0.779	0.715	0.793	0.792	0.786	0.851	0.836	0.627$(L)^{***}$	0.582$(L)^{***}$
		Removing	0.752	0.702	0.804*	0.818$(L)^{***}$	0.824$(L)^{***}$	0.861$(M)^{***}$	**0.886**$(L)^{***}$		
		Separating	0.738	0.699	0.800*	0.812$(L)^{***}$	0.824$(L)^{***}$	0.856*	0.845		
	xalan-2.5	Original	0.621	0.582	0.657	0.579	0.616	0.697	0.706	0.578$(L)^{***}$	0.527$(L)^{***}$
		Removing	0.664**	0.595$(L)^{***}$	0.687$(L)^{***}$	0.643$(L)^{***}$	0.681$(L)^{***}$	0.742$(L)^{***}$	**0.778**$(L)^{***}$		
		Separating	0.602	0.663$(L)^{***}$	0.679$(L)^{***}$	0.628$(L)^{***}$	0.678$(L)^{***}$	0.731$(L)^{***}$	0.763$(L)^{***}$		
	xalan-2.6	Original	0.733	0.729	0.750	0.788	0.782	0.789	0.836	0.592$(L)^{***}$	0.544$(L)^{***}$
		Removing	0.775$(L)^{**}$	0.742$(M)^{***}$	0.784$(L)^{***}$	0.796*	0.788*	0.826$(L)^{***}$	**0.860**$(L)^{***}$		
		Separating	0.773$(L)^{**}$	0.736$(M)^{***}$	0.770$(L)^{***}$	0.781	0.791*	0.814$(L)^{***}$	0.855$(L)^{***}$		
$R < 12.5\%$	prop-1-164	Original	0.695	0.734	0.739	0.756	0.669	0.813	**0.817**	0.689$(L)^{***}$	0.595$(L)^{***}$
		Removing	0.706	0.739$(L)^{***}$	0.707	0.755	0.686$(M)^{***}$	0.783$(L)^{***}$	0.821		
		Separating	0.706	0.720$(L)^{***}$	0.730	0.779$(M)^{***}$	0.710$(L)^{***}$	0.801*	0.823		
	prop-1-44	Original	0.854	0.809	0.812	0.789	0.739	0.853	**0.951**	0.630$(L)^{***}$	0.517$(L)^{***}$
		Removing	0.893*	0.775	0.755	0.834$(L)^{***}$	0.843$(L)^{***}$	0.938$(L)^{***}$	0.956*		
		Separating	0.858	0.804	0.782	0.848$(L)^{***}$	0.858$(L)^{***}$	0.943$(L)^{***}$	0.937		
	prop-3-318	Original	0.829	0.925	0.888	0.904	0.896	0.925	**0.960**	0.595$(L)^{***}$	0.526$(L)^{***}$
		Removing	0.898$(M)^{**}$	0.922	0.897*	0.932$(M)^{***}$	0.918*	0.940*	0.968		
		Separating	0.845**	0.883	0.898*	0.930$(M)^{***}$	0.919*	0.940$(M)^{***}$	0.922		
	prop-5-121	Original	0.716	0.659	0.721	0.784	0.705	0.788	0.812	0.684$(L)^{***}$	0.631$(L)^{***}$
		Removing	0.625	0.639	0.713	0.788	0.695	0.803$(M)^{***}$	**0.833**$(L)^{***}$		
		Separating	0.626	0.665	0.721	0.797*	0.708	0.805$(M)^{***}$	**0.833**$(L)^{***}$		
	prop-5-157	Original	0.696	0.651	0.724	0.784	0.755	0.793	0.791	0.691$(L)^{***}$	0.618$(L)^{***}$
		Removing	0.597	0.648	0.647	0.779	0.681	0.739	**0.807**		
		Separating	0.586	0.630	0.680	0.788	0.715	0.760	0.805		
	prop-5-185	Original	0.719	0.725	0.741	0.794	0.695	0.811	0.830	0.668$(L)^{***}$	0.500$(L)^{***}$
		Removing	0.608	0.726	0.698	0.793	0.717$(M)^{***}$	0.795	**0.838**		
		Separating	0.600	0.692	0.700	0.799	0.724$(M)^{***}$	0.798	0.826		
	prop-5-4	Original	0.668	0.726	0.705	0.755	0.649	0.807	0.795	0.632$(L)^{***}$	0.635$(L)^{***}$
		Removing	0.626	0.690	0.663	0.744	0.683$(M)^{***}$	0.773	**0.812**		
		Separating	0.628	0.683	0.674	0.755	0.699$(L)^{***}$	0.781	0.798		
	prop-5-40	Original	0.849	0.841	0.824	0.846	0.843	0.876	**0.941**	0.597$(L)^{***}$	0.516$(L)^{***}$
		Removing	0.736	0.808	0.780	0.845	0.831	0.899$(L)^{***}$	0.931		
		Separating	0.707	0.798	0.786	0.845	0.836	0.897$(L)^{***}$	0.903		

(1) *** means $p < 0.0001$, ** means $p < 0.001$, * means $p < 0.05$.
(2) L/M: Large/Medium effect size according to Cliff's delta.
(3) The statistical results of CLA and CLAMI are calculated with models with RF classifier trained with removing overlapping instances.

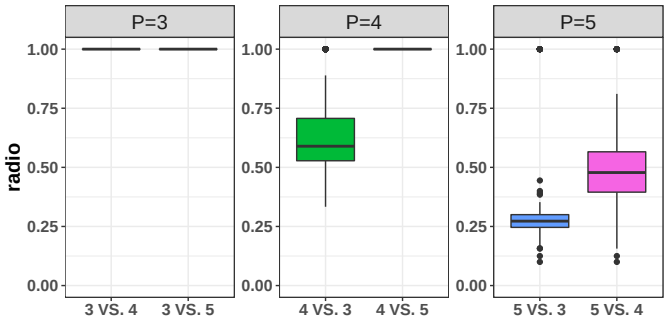


Fig. 11: The boxplot of the percentages of the same overlapping instances identified by different p values across 230 SDP datasets.

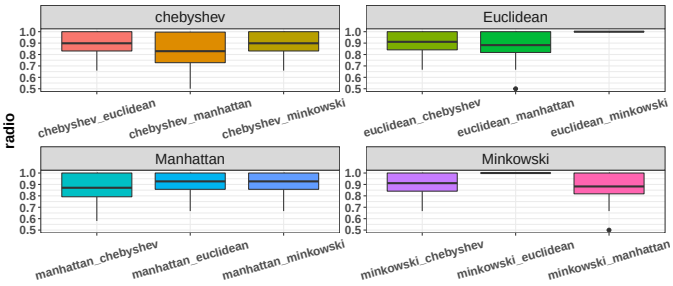


Fig. 12: The boxplot of the percentages of the same overlapping instances identified by different distance methods across 230 SDP datasets.

of p on identifying overlapping instances and the performance of SDP models. Figure 11 presents the percentages of the same overlapping instances identified by different p values across 230 SDP datasets. Table 4 presents the median results for each studied SDP dataset based on different p values.

From Figure 11 and Table 4, we observe that although more instances are identified as overlapping instances when p is equal to 4 and 5, the performance of the SDP model degrades. On the contrary, when p is set to 3, our approach at least 16/18 projects achieves the best performance in terms of AUC, Recall, Brier-score, and Popt.

Furthermore, to indicate the effectiveness of the P values, we calculate the Wilcoxon-signed rank test [61] and Cliff's Delta effect size test [78] between the models trained on removing overlapping instances identified by P=3 and each of the other P values in Table 4. From Table 4, we observe that the performance of the SDP model trained on removing overlapping instances identified by p=3 has statistical differences from other p values. Therefore, we recommend researchers and practitioners use p=3 as our identified overlapping method.

6.2 The Impact of distance methods

To investigate the impact of the distance method on identifying overlapping instances and performance of SDP models, we use different distance methods (i.e., euclidean, manhattan, chebyshev, and minkowski) to identify overlapping instances while fixing the p to 3.

Figure 12 presents the percentages of the same overlapping instances identified by different distance methods across 230 SDP datasets. Table 5 presents the median results for each studied SDP

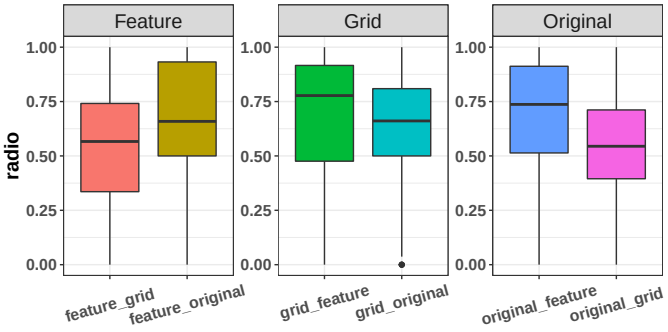


Fig. 13: The boxplot of the percentages of the same overlapping instances identified by the compared methods across 230 SDP datasets.

dataset based on different distance methods. Furthermore, Table 5 presents the results of Wilcoxon-signed rank test and Cliff's Delta effect size test between the models trained on removing overlapping instances identified by euclidean and other distance methods.

From these results, we observe that percentages of the same overlapping instances identified by any two distance methods across 230 SDP datasets are more than 0.85. Meanwhile, models trained on removing overlapping instances identified by the Euclidean method could get the highest performance in terms of AUC, Recall, Brier-score, and Popt for at least 14/18 projects. Furthermore, compared with other distance methods, models trained on removing overlapping instances identified by the Euclidean method are statistically different. Therefore, we recommend researchers and practitioners to use the euclidean method as the distance method of our identified overlapping method.

6.3 The Impact of the number of features and cluster method

In this experiment, we study the impact of the number of features and cluster method on our proposed overlapping identification method, since the number of features could change how we calculcate distance and Nam et al. [47] reported that SDP datasets could be reduced to 2-3 metrics. We use the Grid-Clustering and feature selection method proposed by Papakroni [51] to compare with our proposed overlapping identification method based on metrics with correlation and redundancy analysis in Section 4.2.

Figure 13 presents the percentages of the same overlapping instances identified by the compared methods across 230 SDP datasets. Table 6 presents the median results for each of the studied SDP datasets based on the compared methods. Note that in Figure 13 and Table 6, the term of original represents our proposed overlapping identification method based on metrics with correlation and redundancy analysis, the term of feature represents our proposed overlapping identification method based on 2-3 metrics, the term of grid represents the grid-clustering method to identify the overlapping instances.

From these results, we observe that median percentages of the same overlapping instances identified by the compared methods across 230 SDP datasets are less than 0.75. Meanwhile, models trained on removing overlapping instances identified by our original method could get the highest performance for at least 15/18 projects in terms of AUC, Recall, Brier-score, and Popt.

TABLE 4: The Median results for each studied SDP datasets based on different p values.

Project	$AUC^{\uparrow}$			$Recall^{\uparrow}$			$Brier-score^{\downarrow}$			$False\ \ alarm^{\downarrow}$			$P_{opt}{}^{\uparrow}$		
	3	4	5	3	4	5	3	4	5	3	4	5	3	4	5
Camel-1.2	**0.682**	$0.635(M)^{***}$	$0.595(L)^{***}$	**0.235**	$0.105(L)^{***}$	$0.040(L)^{***}$	**0.226**	$0.264(L)^{***}$	$0.290(L)^{***}$	0.056	0.028	**0.007**	**0.560**	$0.492(M)^{***}$	$0.490(M)^{***}$
derby-10.2.16	**0.794**	$0.776(M)^{***}$	$0.765(L)^{***}$	**0.519**	0.491^{*}	$0.421(L)^{***}$	**0.176**	$0.192(L)^{***}$	$0.209(L)^{***}$	0.120	0.121	**0.105**	**0.460**	0.436^{*}	$0.408(L)^{***}$
derby-10.3.1.4	**0.798**	0.776^{**}	$0.756(L)^{***}$	**0.454**	0.410	$0.405(M)^{***}$	**0.166**	0.182^{*}	$0.195(L)^{***}$	0.091	**0.085**	0.096	**0.484**	$0.446(M)^{***}$	$0.410(L)^{***}$
eclipse34_debug	**0.794**	0.775^{**}	$0.704(L)^{***}$	**0.333**	$0.258(M)^{***}$	$0.144(L)^{***}$	**0.152**	0.176^{*}	$0.200(L)^{***}$	0.047	0.049	**0.026**	**0.486**	$0.419(M)^{***}$	$0.407(L)^{***}$
prop-1-164	**0.821**	$0.784(L)^{***}$	$0.781(L)^{***}$	**0.289**	0.163^{*}	$0.111(L)^{***}$	**0.060**	0.069	0.072	0.006	0.002	**0.000**	**0.687**	0.633^{*}	0.634^{*}
prop-1-44	**0.956**	$0.871(L)^{***}$	$0.797(L)^{***}$	**0.372**	$0.284(L)^{***}$	$0.224(L)^{***}$	**0.046**	0.061^{*}	0.066^{*}	0.005	0.006	**0.004**	**0.793**	$0.704(L)^{***}$	$0.664(L)^{***}$
prop-1-92	**0.888**	0.864^{**}	$0.851(L)^{***}$	0.703	**0.719**	0.685^{*}	**0.126**	$0.140(M)^{***}$	$0.149(M)^{***}$	**0.109**	$0.148(L)^{***}$	$0.147(L)^{***}$	**0.725**	0.666^{*}	$0.654(L)^{***}$
prop-2-256	**0.920**	$0.887(L)^{***}$	$0.869(L)^{***}$	**0.729**	0.672^{*}	$0.657(L)^{***}$	**0.101**	0.122^{*}	0.136^{*}	**0.076**	0.096^{**}	$0.129(L)^{***}$	**0.779**	0.729^{*}	$0.664(L)^{***}$
prop-3-318	**0.968**	0.958^{*}	$0.903(L)^{***}$	**0.604**	$0.433(L)^{***}$	$0.296(L)^{***}$	**0.053**	0.066^{*}	$0.089(M)^{***}$	0.017	0.005	**0.000**	**0.859**	0.829^{*}	$0.792(M)^{***}$
prop-4-355	**0.900**	0.874^{**}	$0.846(L)^{***}$	**0.688**	$0.564(L)^{***}$	$0.531(L)^{***}$	**0.118**	0.135^{**}	$0.152(L)^{***}$	0.091	**0.074**	**0.074**	**0.718**	0.664^{*}	0.667^{*}
prop-5-121	**0.833**	0.812^{*}	$0.765(L)^{***}$	**0.202**	$0.080(L)^{***}$	$0.036(L)^{***}$	**0.087**	0.101	0.109	0.007^{*}	0.004	**0.002**	**0.662**	$0.604(M)^{***}$	$0.550(L)^{***}$
prop-5-157	**0.807**	$0.767(M)^{***}$	$0.697(L)^{***}$	**0.173**	$0.090(L)^{***}$	$0.065(L)^{***}$	**0.097**	0.108	0.113	0.008	0.003	**0.002**	**0.508**	0.459^{*}	0.473^{*}
prop-5-185	**0.832**	$0.788(L)^{***}$	$0.693(L)^{***}$	**0.102**	$0.051(L)^{***}$	0.000	**0.064**	0.071	0.076	0.004	0.001	**0.000**	**0.619**	0.563^{*}	$0.536(L)^{***}$
prop-5-4	**0.812**	$0.716(L)^{***}$	$0.670(L)^{***}$	**0.199**	$0.073(L)^{***}$	$0.073(L)^{***}$	**0.056**	0.065^{*}	0.066^{*}	0.002	**0.000**	**0.000**	**0.710**	0.668^{*}	0.694^{*}
prop-5-40	**0.931**	$0.900(L)^{***}$	$0.855(L)^{***}$	**0.432**	0.378^{*}	$0.326(L)^{***}$	**0.061**	0.068	0.076^{*}	0.006	0.002	**0.000**	**0.788**	0.732^{*}	0.751^{*}
prop-5-85	**0.886**	$0.861(M)^{***}$	$0.856(L)^{***}$	**0.567**	$0.405(L)^{***}$	$0.404(L)^{***}$	**0.115**	$0.131(M)^{***}$	$0.138(M)^{***}$	0.048	**0.029**	0.032	**0.705**	0.662^{*}	0.659^{*}
xalan-2.5	**0.778**	0.754^{**}	$0.749(M)^{***}$	**0.739**	$0.663(L)^{***}$	$0.558(L)^{***}$	**0.196**	0.208	0.212^{*}	0.316	0.287	**0.227**	**0.498**	0.442^{*}	0.463^{*}
xalan-2.6	**0.860**	0.846^{*}	$0.827(L)^{***}$	**0.675**	0.643^{*}	0.669	**0.157**	0.165	$0.175(M)^{***}$	0.133	**0.132**	$0.186(L)^{***}$	**0.327**	0.290^{*}	$0.277(L)^{***}$

(1) *** means $p < 0.0001$, ** means $p < 0.001$, * means $p < 0.05$.
(2) L/M: Large/Medium effect size according to Cliff's delta.

TABLE 5: The Median results for each studied SDP datasets based on different distance methods.

Project	$AUC^{\uparrow}$				$Recall^{\uparrow}$				$Brier-score^{\downarrow}$				$False\ \ alarm^{\downarrow}$				$P_{opt}^{\uparrow}$			
	Eu	Ma	Ch	Mi	Eu	Ma	Ch	Mi	Eu	Ma	Ch	Mi	Eu	Ma	Ch	Mi	Eu	Ma	Ch	Mi
Camel-1.2	**0.682**	0.675	0.663^{*}	0.674	**0.235**	$0.188(M)^{***}$	$0.167(M)^{***}$	$0.157(M)^{***}$	0.226	**0.224**	0.225	0.226	0.056	0.044	0.043	**0.034**	**0.560**	0.535^{*}	0.535^{*}	0.540^{*}
derby-10.2.16	**0.794**	0.790	0.781^{*}	0.786	**0.519**	0.493^{*}	0.482^{*}	0.498^{*}	**0.176**	0.179	0.184	0.183	0.120	**0.117**	0.120	0.121	**0.460**	0.447^{*}	$0.439(M)^{**}$	$0.437(M)^{**}$
derby-10.3.1.4	**0.798**	0.794	0.782^{*}	0.789	**0.454**	0.422^{*}	$0.397(M)^{***}$	0.419^{*}	**0.166**	**0.166**	0.172	0.169	0.091	0.080	**0.078**	0.088	**0.484**	0.464^{*}	0.460^{*}	0.471^{*}
eclipse34_debug	0.794	**0.796**	0.794	0.791	**0.333**	$0.267(M)^{***}$	$0.284(M)^{***}$	$0.282(M)^{***}$	**0.152**	0.159	0.158	0.159	0.047	**0.044**	0.050	0.046	**0.486**	$0.448(M)^{***}$	$0.441(M)^{***}$	$0.449(M)^{***}$
prop-1-164	**0.821**	0.820	0.819	0.818	**0.289**	$0.178(L)^{***}$	$0.163(L)^{***}$	$0.175(L)^{***}$	**0.060**	0.063	0.065	0.063	0.006	**0.002**	**0.002**	**0.002**	**0.687**	$0.649(M)^{**}$	$0.642(M)^{**}$	$0.648(M)^{**}$
prop-1-44	0.956	**0.968**	0.943^{*}	0.957	**0.372**	0.347^{*}	$0.323(M)^{***}$	$0.338(M)^{***}$	**0.046**	**0.046**	0.053^{*}	0.051	0.006	**0.004**	**0.004**	0.005	0.793	**0.799**	$0.748(M)^{**}$	0.773^{*}
prop-1-92	**0.888**	0.873^{*}	0.873^{*}	0.873	0.719	**0.725**	0.713	0.699^{*}	**0.126**	0.135^{*}	0.135^{*}	0.135^{*}	**0.109**	$0.137(M)^{**}$	$0.141(M)^{**}$	$0.132(M)^{**}$	**0.725**	0.698^{*}	$0.672(M)^{**}$	$0.672(M)^{**}$
prop-2-256	**0.920**	0.904^{*}	$0.899(M)^{***}$	$0.899(M)^{***}$	**0.729**	0.690^{*}	$0.662(M)^{**}$	$0.679(M)^{***}$	**0.101**	0.111	0.116^{*}	0.116^{*}	**0.076**	0.090^{*}	0.085^{*}	0.087^{*}	**0.779**	0.763^{*}	$0.735(M)^{**}$	$0.745(M)^{**}$
prop-3-318	**0.968**	0.966	0.965	0.965	**0.604**	0.551^{*}	$0.516(L)^{***}$	$0.519(L)^{***}$	**0.053**	0.056	0.058	0.059	0.017	**0.011**	0.013	0.012	**0.859**	0.842^{*}	0.842^{*}	0.842^{*}
prop-4-355	**0.900**	0.887^{*}	0.880^{*}	0.882^{*}	**0.688**	0.661	$0.615(M)^{***}$	$0.637(M)^{***}$	**0.118**	0.125^{*}	0.129^{*}	0.129^{*}	0.091	0.097	**0.083**	0.091	**0.718**	0.672^{*}	$0.659(M)^{**}$	$0.662(M)^{**}$
prop-5-121	0.833	**0.838**	0.827	0.837	**0.202**	$0.124(M)^{***}$	$0.109(L)^{***}$	$0.125(M)^{***}$	**0.087**	0.091	0.093	0.090	0.007	**0.004**	**0.004**	**0.004**	**0.662**	0.635^{*}	$0.616(M)^{**}$	0.637^{*}
prop-5-157	**0.807**	0.806	0.797	0.792^{*}	**0.173**	$0.136(M)^{***}$	$0.114(M)^{***}$	$0.117(M)^{***}$	**0.097**	**0.097**	0.100	0.101	0.008	0.006	**0.004**	0.005	0.508	**0.509**	0.485^{*}	0.483^{*}
prop-5-185	**0.838**	0.835	0.826	0.833	0.102	**0.119**	0.096	0.112	**0.064**	**0.064**	0.067	0.066	0.005	0.005	**0.004**	0.005	**0.631**	0.629	$0.606(M)^{**}$	0.619^{*}
prop-5-4	**0.812**	0.783^{*}	$0.769(M)^{***}$	0.780	**0.199**	0.114	$0.067(L)^{***}$	$0.105(M)^{***}$	**0.056**	0.061	0.063	0.061	0.002	0.002	**0.001**	0.002	**0.710**	$0.664(M)^{**}$	$0.655(M)^{**}$	$0.667(M)^{**}$
prop-5-40	0.931	0.929	**0.934**	0.930	**0.432**	0.410^{*}	0.403^{**}	0.401^{**}	**0.061**	0.062	0.063	0.063	0.006	0.005	**0.004**	**0.004**	**0.788**	0.760	0.763	0.764
prop-5-85	**0.886**	0.873	0.869^{*}	0.867^{*}	**0.567**	$0.479(M)^{***}$	$0.442(L)^{***}$	$0.457(L)^{***}$	**0.115**	**0.119**	0.124	0.122	0.048	0.037	**0.035**	0.037	**0.705**	0.690	0.673	0.676
xalan-2.5	**0.778**	0.766^{*}	0.762^{*}	0.766^{*}	**0.739**	$0.687(M)^{***}$	0.717^{**}	$0.714(M)^{**}$	**0.196**	0.199	0.201	0.198	0.316	**0.301**	0.323	0.321	**0.498**	0.464^{*}	0.468^{*}	0.467^{*}
xalan-2.6	**0.860**	**0.860**	0.848^{*}	0.855	**0.675**	$0.631(M)^{***}$	$0.639(M)^{***}$	$0.649(M)^{***}$	0.157	**0.155**	0.159	0.157	0.133	**0.100**	0.121	0.122	**0.327**	0.305^{*}	0.300^{*}	0.301^{*}

(1) Eu:Euclidean; Ma:Manhattan; Ch:Chebyshev; Mi:Minkowski.
(2) *** means $p < 0.0001$, ** means $p < 0.001$, * means $p < 0.05$.
(3) L/M: Large/Medium effect size according to Cliff's delta.

Therefore, we recommend researchers and practitioners to identify overlapping instances with our parameter settings (i.e., K=5, p=3, distance=euclidean).

7 IMPLICATIONS

According to the experimental results, we outline and present the implications that could guide researchers and practitioners to build their SDP models.

Implication 1) **Researchers and practitioners set parameter (i.e., K=5, p=3, distance=euclidean) as the default parameter when applying our proposed KNN method to identify overlapping instances.** From the experimental results in Section 6, we can observe that different parameter setting of our proposed KNN methods could generate different results. SDP models trained on removing overlapping instances identified by our proposed KNN method with the K=5, p=3, and distance=euclidean generate the highest performance. Therefore, we recommend researchers and practitioners to identify overlapping instances with our parameter settings (i.e., K=5, p=3, distance=euclidean) in defect datasets.

Implication 2) **Researchers and practitioners should first apply our proposed KNN method to identify whether the overlap ratios of their defect datasets are greater than 12.5% before building SDP models.** From the experimental results in RQ2, we can observe that overlapping instances are ubiquitous in SDP datasets. Indeed, different levels of class overlap have different impacts on the performance of SDP models. When the class overlap ratio is over 12.5%, it seriously affects the stability of the performance of SDP models, while the frequently used SDP datasets in NASA, AEEEM, PROMISE, ReLink, and SOFTLAB always have more than 12.5% overlapping instances. However, as far as we know, none of the related and similar studies consider the class overlap problem before constructing SDP models [8, 26, 27, 62, 70]. Therefore, we urge researchers and practitioners to first apply our proposed KNN method to identify overlapping ratios in their studies before constructing SDP models.

Implication 3) **For defect datasets with overlap ratio greater than 12.5%, RF classifiers with class overlap handling techniques should be considered when quality assurance teams want to reduce the efforts needed to review the code.** The experimental results of RQ5 indicate that class overlap handling techniques (i.e., removing and separating techniques) could statistically significantly improve the performance of SDP models trained on datasets with over 12.5% overlap ratios. The RF classifier can obtain better performance values both on the original datasets and on the datasets handled by removing techniques. Therefore, we urge the researchers and practitioners to apply the RF classifier with the removing technique to construct SDP models for defect datasets with overlap ratios greater than 12.5%.

Implication 4) **For interpretation of defect prediction models, researchers and practitioners should first apply our proposed KNN method to remove the overlapping instances to find the more consistent guiding significance metrics.** From the experimental results of RQ4, we can observe that class overlap shifts the important feature lists of SDP models, particularly the features listed at the top 3 rank. Meanwhile, we observe that compared with original datasets, models trained on removing overlapping instances can identify more of the same metrics at the Top-1 rank among the different releases of the same projects. Therefore, we urge the researchers and practitioners to remove

TABLE 6: The Median results for each studied SDP datasets based on compared methods.

Project	$AUC^{\uparrow}$			$Recall^{\uparrow}$			$Brier-score^{\downarrow}$			$False\ \ alarm^{\downarrow}$			${}^{\uparrow}\mathbf{P}^{\uparrow}_{opt}$		
	Original	feature	Grid	Original	feature	Grid	Original	feature	Grid	Original	feature	Grid	Original	feature	Grid
Camel-1.2	**0.682**	0.564$(L)^{***}$	0.593$(M)^{***}$	0.235	0.200*	**0.274**	**0.226**	0.237	0.271$(M)^{***}$	**0.056**	0.147$(L)^{***}$	0.143$(L)^{***}$	**0.560**	0.498$(M)^{***}$	0.503$(M)^{***}$
derby-10.2.16	**0.794**	0.707$(M)^{***}$	0.729$(M)^{***}$	**0.519**	0.391$(L)^{***}$	0.471*	**0.176**	0.198	0.205*	**0.120**	0.117	0.166$(M)^{***}$	**0.460**	0.439*	0.451
derby-10.3.1.4	**0.798**	0.696$(L)^{***}$	0.717$(M)^{***}$	**0.454**	0.287$(L)^{***}$	0.380$(L)^{***}$	**0.166**	0.190*	0.198*	**0.091**	0.078	0.134*	**0.484**	0.419$(M)^{***}$	0.409$(M)^{***}$
eclipse34_debug	**0.794**	0.775*	0.704$(L)^{***}$	**0.333**	0.258$(L)^{***}$	0.144$(L)^{***}$	**0.152**	0.176	0.200	0.047	0.049	**0.026**	**0.486**	0.419	0.407
prop-1-164	**0.821**	0.649$(L)^{***}$	0.639$(L)^{***}$	**0.289**	0.058$(L)^{***}$	0.068$(L)^{***}$	**0.060**	0.082	0.080	0.006	**0.003**	0.008	**0.687**	0.519$(L)^{***}$	0.519$(L)^{***}$
prop-1-44	**0.956**	0.625$(L)^{***}$	0.630$(L)^{***}$	**0.372**	0.116$(L)^{***}$	0.092$(L)^{***}$	**0.046**	0.090	0.085	0.006	0.020	**0.005**	**0.793**	0.524$(L)^{***}$	0.525$(L)^{***}$
prop-1-92	**0.888**	0.713$(L)^{***}$	0.703$(L)^{***}$	**0.719**	0.475$(L)^{***}$	0.546$(L)^{***}$	**0.126**	0.234$(L)^{***}$	0.254$(L)^{***}$	**0.109**	0.114	0.182$(M)^{***}$	**0.725**	0.569$(L)^{***}$	0.630$(M)^{***}$
prop-2-256	**0.920**	0.697$(L)^{***}$	0.792$(L)^{***}$	**0.729**	0.579$(L)^{***}$	0.651$(M)^{***}$	**0.101**	0.234$(L)^{***}$	0.179$(L)^{***}$	**0.076**	0.19$(L)^{***}$6	0.142$(L)^{***}$	**0.779**	0.655$(L)^{***}$	0.713$(M)^{***}$
prop-3-318	**0.968**	0.632$(L)^{***}$	0.632$(L)^{***}$	**0.604**	0.307$(L)^{***}$	0.021$(L)^{***}$	**0.053**	0.165$(L)^{***}$	0.143$(L)^{***}$	0.017	0.078	**0.007**	**0.859**	0.543$(L)^{***}$	0.345$(L)^{***}$
prop-4-355	**0.900**	0.713$(L)^{***}$	0.718$(L)^{***}$	**0.688**	0.539$(L)^{***}$	0.577$(L)^{***}$	**0.118**	0.241$(L)^{***}$	0.225$(L)^{***}$	**0.091**	0.164$(L)^{***}$	0.165$(L)^{***}$	**0.718**	0.668*	0.668*
prop-5-121	**0.833**	0.693$(L)^{***}$	0.703$(L)^{***}$	**0.202**	0.069$(L)^{***}$	0.046$(L)^{***}$	**0.087**	0.111	0.109	**0.007**	0.013*	0.014*	**0.662**	0.504$(L)^{***}$	0.454$(L)^{***}$
prop-5-157	**0.807**	0.689$(L)^{***}$	0.683$(L)^{***}$	**0.173**	0.058	0.089	**0.097**	0.113	0.111	0.008	**0.007**	0.017	**0.508**	0.456*	0.446*
prop-5-185	**0.838**	0.687$(L)^{***}$	0.707$(L)^{***}$	**0.102**	0.057$(M)^{***}$	0.058$(M)^{***}$	**0.064**	0.074	0.072	0.005	**0.004**	0.007	**0.631**	0.513$(L)^{***}$	0.470$(L)^{***}$
prop-5-4	**0.812**	0.607$(L)^{***}$	0.616$(L)^{***}$	**0.199**	0.011$(L)^{***}$	0.021$(L)^{***}$	**0.056**	0.071	0.071	**0.002**	**0.002**	0.003	**0.710**	0.449$(L)^{***}$	0.395$(L)^{***}$
prop-5-40	**0.931**	0.900*	0.855$(L)^{***}$	**0.432**	0.378*	0.326$(L)^{***}$	**0.061**	0.068	0.076	0.006	0.002	**0.000**	**0.788**	0.732*	0.751*
prop-5-85	**0.886**	0.736$(L)^{***}$	0.712$(L)^{***}$	**0.567**	0.395$(L)^{***}$	0.353$(L)^{***}$	**0.115**	0.210$(M)^{***}$	0.200$(M)^{***}$	**0.048**	0.116$(L)^{***}$	0.104$(L)^{***}$	**0.705**	0.603$(L)^{***}$	0.554$(L)^{***}$
xalan-2.5	**0.778**	0.650$(L)^{***}$	0.649$(L)^{***}$	**0.739**	0.685*	0.605$(L)^{***}$	**0.196**	0.285$(L)^{***}$	0.287$(L)^{***}$	**0.316**	0.402$(L)^{***}$	0.377$(L)^{***}$	**0.498**	0.394$(L)^{***}$	0.376$(L)^{***}$
xalan-2.6	**0.860**	0.812$(M)^{***}$	0.808$(M)^{***}$	0.675	**0.685**	0.669	**0.157**	0.192	0.195	**0.133**	0.191	0.222	0.327	0.355	**0.358**

(1) Eu:Euclidean; Ma:Manhattan; Ch:Chebyshev; Mi:Minkowski.
(2) *** means $p < 0.0001$, ** means $p < 0.001$, * means $p < 0.05$.
(3) L/M: Large/Medium effect size according to Cliff's delta.

the overlapping instances to find the more consistent guiding significance metrics.

8 Threats to Validity

Experimental design would affect the experimental results [65]. Therefore, in this section, we identify the threats to the validity of our presented results.

Internal Validity. In our study, we apply KNN with Euclidean distance and k=5 to identify the neighborhoods of a given instance for detecting overlapping instances, other parameters of K are not investigated. However, prior studies [11, 33, 48] have proved the validity under k=5 and Euclidean distance in software defect prediction. Meanwhile, we compared our proposed method with other overlapping identification methods (K-Means, SMR, and SVDD) and found that our proposed method employing KNN could identify more precise overlapping instances compared with K-Means, SMR, and SVDD methods. Therefore, we argue that our results under the current parameter are still valid. Furthermore, we evaluate our proposed method with fixed range values for different parameters (e.g., distance methods (i.e., euclidean, manhattan, chebyshev, and minkowski)) in Section 6.2 to find the best possible parameter values for the implication of our proposed KNN method. However, we acknowledge that investigating the impact of different parameter settings is an interesting avenue for future research.

Another threat to internal validity is that we only study the removing and separating techniques to handle the class overlap problem. Other class overlap handling techniques in machine learning (e.g., ClusBUS [13], active learning methods [82], Spill trees [40]) are not investigated in our study. Therefore, it would be valuable to extend our study by employing other class overlap handling techniques.

The third possible threat to internal validity is that we apply artificially to change their labels from defective to non-defective or from non-defective to defective to generate datasets with different overlap ratios. Kim et al. [33] have artificially changed the labels of instances to generate different noise for defect prediction models. However, it is possible to introduce its own conflations into the analysis. Therefore, we encourage future studies to explore the impact of other generating datasets with different overlap ratio on the performance of defect prediction models.

External Validity. The first possible external validity would be our studied datasets. In our experimental study, we select 230 datasets from industrial and open-source software projects and exclude datasets whose EPV are lower ($EPV < 10$) or defective ratio are high ($> 50\%$). Thus, our results might not be generalized to other SDP datasets. However, these datasets are characterized by different types of software projects, with different sizes of modules and different levels of overlapping instances (see Table 1), which enables our findings to be reliable. In addition, we analyze the effectiveness of our proposed KNN methods on the projects with the fixed stages of development. Although our studied datasets include a variety of stages of the development of the projects, studying what stage of the development of the project to apply our proposed KNN method in the SDP would be a valuable avenue for future work.

The second possible external validity would be the SDP contexts and scenarios. In our study, we only consider the Defect-classification and Effort-aware scenarios in the within-project context, the other SDP contexts such as cross-project context [85] and just-in-time context [32] and other defect prediction scenario such as the defect-count scenario [34, 49] are not explored. Although our prior study [20] has demonstrated the effectiveness of removing overlapping instances in cross-item scenarios, it would be valuable to extend our study to other SDP scenarios and contexts.

Another threat to external validity is that we analyze the impact of the different class overlap ratios on the defect prediction models on three software projects. Though the studied projects are diverse in domain, size, and metrics, there is a threat that our findings may not generalize for projects with different dimensions. Therefore, we encourage future studies to study the impact of different overlap ratios on the defect prediction models across different software projects.

Construct Validity. The first possible construct validity would be stochastic data processing techniques, such as dividing datasets into training and test datasets. In our study, we apply 100 times of the out-of-sample bootstrap process to produce 100 combinations of training and test datasets. Tantithamthavorn et al. [68] have proved that the out-of-sample bootstrap validation technique could generate the best balance between the bias and variance in the SDP context. Thus, the out-of-sample bootstrap validation technique would be a well-established stochastic data processing technique for comparing different SDP models.

Another possible construct validity would be the studied classifiers. In our study, we select 7 classifiers that are widely used in the SDP context. Meanwhile, these 7 studied classifiers are from different classifier families [19]. Thus, we suspect that our findings

in these studied classifiers would be valuable to researchers and practitioners.

The third possible threat to construct validity is that we do not consider multiple model interpretation techniques for defect prediction models. Prior studies [30, 55] have found that the studied model-agnostic techniques exhibit a strong agreement on the features reported at top-1 and top-3 ranks. Therefore, in our study, we only apply permutation method to calculate the 100 important scores for each feature of the 7 studied classifiers on each of the studied 18 datasets with original and non-overlapping instances. However, it is possible that the observed results on permutation method might not generalize for other model agnostic techniques such as SHAP and LIME [56]. Therefore, we encourage future studies to explore the impact of other model agnostic techniques on the model interpretation of defect prediction models across the original datasets and non-overlap datasets.

9 Conclusions

Class overlap is the phenomenon that datasets contain some instances with different classes having similar values in feature space, which hinders the performance of machine learning techniques [15, 16, 52]. In the SDP context, ***none*** of the prior studies attempts to quantify overlapping instances and the relationship between class overlap and the performance of SDP models. To fill these research gaps, we set out a comprehensive investigation to explore the impacts of class overlap on SDP models in this study. Specifically, we propose a KNN method to identify overlapping instances and investigate the overlapping level in 230 SDP datasets. Indeed, we investigate the impacts of the levels of class overlap and class overlap handling techniques (removing and separating) on the performance and the interpretation of SDP models. We construct our SDP models with 7 representative classifier techniques (SVM, RF, KNN, DT, GBM, NB, and LR) over 18 datasets across 4 performance measures. We outline the following observations:

- Overlapping instances are ubiquitous in SDP datasets. Meanwhile, the level of class overlap has different impacts on the performance of SDP models, especially the class overlap ratio that is over 12.5% seriously affects the stability of the performance of SDP models.
- Class overlap shifts the important feature lists of SDP models, particularly the features lists at the top 2 and 3 ranks. Meanwhile, removing overlapping instances could help to find more consistent guiding significance metrics for a given dataset and different releases of a given project.
- Class overlap handling techniques could statistically significantly improve the performance of SDP models trained on datasets with over 12.5% overlap ratios. RF classifier can obtain better performance values both on the original datasets and on the datasets handled by removing techniques.

Based on these findings, we have the following implications for researchers and developers:

- Researchers and practitioners should first apply our proposed KNN method to identify whether the overlap ratios of defect datasets are greater than 12.5% before building SDP models.
- For defect datasets with overlap ratio greater than 12.5%, RF classifiers with class overlap handling techniques should be considered when quality assurance teams want to reduce the efforts needed to review the code.
- For the interpretation of SDP models, researchers and practitioners should first apply our proposed KNN method to remove the overlapping instances to find the more consistent guiding significance metrics.

Acknowledgments

This paper would not have been possible without the generous support by the National Natural Science Foundation of China under grant No. 62202223, and the Natural Science Foundation of Jiangsu Province, China under grant No. BK20220881.

References

[1] A. Agrawal and T. Menzies, "Is "better data" better than "better data miners"?" in *the International Conference on Software Engineering*, 2018, pp. 1050–1061.

[2] A. Agrawal, W. Fu, and T. Menzies, "What is wrong with topic modeling? and how to fix it using search-based software engineering," *Information and Software Technology*, vol. 98, pp. 74–88, 2018.

[3] A. Altmann, L. Tolosi, O. Sander, and T. Lengauer, "Permutation importance: a corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010.

[4] H. X. andJunjie Wu and L. Liu, "Classification with class overlapping: A systematic study," in *The 2010 International Conference on E-Business Intelligence*, 2010, pp. 491–497.

[5] E. Arisholm, L. C. Briand, and E. B. Johannessen, "A systematic and comprehensive investigation of methods to build and evaluate fault prediction models," *Journal of Systems and Software*, vol. 83, no. 1, pp. 2–17, 2010.

[6] Y. Benjamini and Y. Hochberg, "The control of the false discovery rate in multiple testing under dependency," *Journal of the Royal Statistical Society: Series B (Methodological).*, vol. 57, no. 1, pp. 289–300, 1995.

[7] Y. Benjamini and D. Yekutieli, "The control of the false discovery rate in multiple testing under dependency," *Ann. Statist.*, vol. 29, no. 4, pp. 1165–1188, 2001.

[8] K. E. Bennin, J. Keung, P. Phannachitta, A. Monden, and S. Mensah, "MAHAKIL: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction," *IEEE Transactions on Software Engineering*, vol. 44, no. 6, pp. 534–550, 2018.

[9] BradleyEfron, "Estimating the error rate of a prediction rule: Improvement on cross-validation," *Journal of the American Statistical Association*, vol. 78, no. 382, pp. 316–331, 1983.

[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[11] L. Chen, B. Fang, Z. Shang, and Y. Tang, "Tackling class overlap and imbalance problems in software defect prediction," *Software Quality Journal*, vol. 26, no. 1, pp. 97–125, 2018.

[12] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: a benchmark and an extensive comparison," *Empirical Software Engineering*, vol. 17, pp. 531–577, 2012.

[13] B. Das, N. C. Krishnan, and D. J. Cook, "Handling class overlap and imbalance to detect prompt situations in smart homes," in *2013 IEEE 13th International Conference on Data Mining Workshops*, 2013, pp. 266–273.

[14] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[15] M. Denil and T. Trappenberg, "Overlap versus imbalance," in *Advances in Artificial Intelligence*, 05 2010, pp. 220–231.

[16] V. García, J. S. Sánchez, and R. Mollineda, "An empirical study of the behavior of classifiers on imbalanced and overlapped data sets," in *Iberoamerican Congress on Pattern Recognition*, 11 2007, pp. 397–406.

[17] B. Ghostra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," in *the International Conference on Software Engineering*, 2015, pp. 789–800.

[18] B. Ghotra, S. Mclntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," in *7th IEEE International Conference on Software Engineering*, 2015, pp. 789–800.

[19] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation," *Journal of Computational and Graphical Statistics*, vol. 24, no. 1, pp. 44–65, 2015.

[20] L. Gong, S. Jiang, R. Wang, and L. J. and, "Empirical evaluation of the impact of class overlap on software defect prediction," in *34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 11 2019, pp. 1–211.

[21] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "Using the support vector machine as a classification method for software defect prediction with static code metrics," in *International Conference on Engineering Applications of Neural Networks*, 2009, pp. 223–234.

[22] S. Gupta and A. Gupta, "Handling class overlapping to detect noisy instances in classification," *The Knowledge Engineering Review*, vol. 33, no. e8, pp. 1–13, 2018.

[23] T. Hall, S. Beecham, D. Bowes, and D. Gray, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276–1304, 2012.

[24] A. E. Hassan, "Predicting faults using the complexity of code changes," in *The International Conference on Software Engineering*, 2009, pp. 78–88.

[25] 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*, vol. 59, pp. 170–190, 2015.

[26] 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, 2018.

[27] 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, 2019.

[28] L. Hu, M. Huang, S. Ke, and C. Tsai, "The distance function effect on k-nearest neighbor classification for medical datasets," *SpringerPlus*, vol. 5, pp. 1–9, 12 2016.

[29] J. Jiarpakdee, C. Tantithamthavorn, and A. E. Hassan, "Which process metrics can significantly improve defect prediction models? an empirical study," *The Impact of Correlated Metrics on Defect Models (Early Access)*, pp. 1–16, 2019.

[30] J. Jiarpakdee, C. Tantithamthavorn, H. Dam, and J. Grundy, "An empirical study of model-agnostics techniques for defect prediction models," *IEEE Transactions on Software Engineering*, vol. 48, no. 1, pp. 166–185, Jan. 2022.

[31] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi, "A large-scale empirical study of just-in-time quality assurance," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 757–773, 2013.

[32] Y. Kamei, T. Fukushima, S. Mclntosh, K. Yamashita, N. Ubayashi, and A. E. Hassan, "Studying just-in-time defect prediction using cross-project models," *Empirical Software Engineering*, vol. 21, pp. 2072–2106, 2016.

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

[34] W. B. Langdon, J. Dolado, F. Sarro, and M. Harman, "Exact mean absolute error of baseline predictor, marp0," *Information and Software Technology*, vol. 73, no. 5, pp. 16–18, 2016.

[35] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Information and Software Technology*, vol. 58, pp. 388–402, 2015.

[36] H. K. Lee and S. B. Kim, "An overlap-sensitive margin classifier for imbalanced and overlapping data," *Expert Systems with Applications*, vol. 98, no. 12, pp. 72–83, 2018.

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

[38] C. Liu, S. Gong, C. C. Loy, and X. Lin, "Person re-identification: What features are important?" in *Computer Vision – ECCV 2012. Workshops and Demonstrations*, 2012, pp. 391–401.

[39] L. Madeyski and M. Jureczko, "Which process metrics can significantly improve defect prediction models? an empirical study," *Software Quality Journal*, vol. 23, no. 1, pp. 393–422, 2015.

[40] B. McFee and G. Lanckriet, "Large-scale music similarity search with spatial trees," in *Proceedings of the IEEE/ACM International Conference on International Society for Music Information Retrieval*, 2011, pp. 1–6.

[41] T. Mende and R. Koschke, "Effort-aware defect prediction models," in *14th European Conference on Software Maintenance and Reengineering*, 2010, pp. 107–116.

[42] T. Menzies, C. Page, R. Krishna, and M. Rees-Jones, "The promise repository of empirical software engineering data," 2015, http://openscience.us/repo.

[43] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmermann, "Local versus global lessons for defect prediction and effort estimation," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 822–834, 2013.

[44] M. A. Migut, M. Worring, and C. J. Veenman, "Visualizing multi-dimensional decision boundaries in 2d," *Data Mining and Knowledge Discovery*, vol. 29, pp. 273–295, 2015.

[45] R. Moser, W. Pedrycz, and G. Succi, "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction," in *30th international conference on Software engineering*, 2008, pp. 181–190.

[46] J. Nam and S. Kim, "Clami: Defect prediction on unlabeled datasets," in *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015, pp. 452–463.

[47] J. Nam, W. Fu, S. Kim, T. Menzies, and L. Tan, "Heterogeneous defect prediction," *IEEE Transactions on Software Engineering*, vol. 44, no. 9, pp. 874–896, 2018.

[48] K. Napierala and J. Stefanowski, "Types of minority class examples and their influence on learning classifiers from imbalanced data," *Journal of Intelligent Information Systems*, vol. 46, pp. 563–597, 2015.

[49] T. H. D. Nguyen, B. Adams, and A. E. Hassan, "Studying the impact of dependency network measures on software quality," in *The International Conference on Software Maintenance*, 2010.

[50] C. Ni, X. Xia, D. Lo, X. Chen, and Q. Gu, "Revisting supervised and unsupervised methods for effort-aware cross-project defect prediction," *IEEE Transactions on Software Engineering*, vol. 48, no. 3, pp. 786–802, 2022.

[51] V. Papakroni, "Data carving: Identifying and removing irrelevancies in the data," Master's thesis, West Virginia University, 2013.

[52] R. C. Prati, G. E. Batista, and M.-C. Monard, "Class imbalances versus class overlapping: An analysis of a learning system behavior," in *Mexican International Conference on Artificial Intelligence*, 1 2004, pp. 312–321.

[53] G. K. Rajbahadur, S. Wang, Y. Kamei, and A. E. Hassan, "The impact of using regression models to build defect classifiers," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories*, 2017, pp. 135–145.

[54] G. K. Rajbahadur, S. Wang, G. A. Oliva, Y. Kamei, and A. E. Hassan, "The impact of feature importance methods on the interpretation of defect classifiers," *IEEE Transactions on Software Engineering (Under review)*, pp. 1–20, 2020.

[55] G. K. Rajbahadur, S. Wang, G. Ansaldi, Y. Kamei, and A. E. Hassan, "The impact of feature importance methods on the interpretation of defect classifiers," *IEEE Transactions on Software Engineering*, pp. 1–1, 2021.

[56] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the IEEE/ACM International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1–10.

[57] K. Rufibach, "Use of brier score to assess binary predictions," *Journal of Clinical Epidemiology*, vol. 63, no. 8, pp. 938–939, 2010.

[58] M. Shepperd, Q. Song, Z. Sun, and G. Mair, "Data quality: Some comments on the nasa software defect datasets," *IEEE Transactions on Software Engineering*, vol. 39, no. 9, pp. 1208–1215, 2013.

[59] M. Shepperd, D. Bowes, and T. Hall, "Researcher bias: The use of machine learning in software defect prediction," *IEEE Transactions on Software Engineering*, vol. 40, no. 6, pp. 603–616, 2014.

[60] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, "A general software defect-proneness prediction framework," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 356–370, 2010.

[61] ——, "A general software defect-proneness prediction framework," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 356–370, 2011.

[62] Q. Song, Y. Guo, and M. Shepperd, "A comprehensive investigation of the role of imbalanced learning for software defect prediction," *IEEE Transactions on Software Engineering*, vol. 45, no. 12, pp. 1253–1269, 2019.

[63] W. Tang and T. Khoshgoftaar, "Noise identification with the k-means algorithm," in *16th IEEE International Conference on Tools with Artificial Intelligence*, 2004, pp. 373–378.

[64] C. Tantithamthavorn, "Scottknottesd: The scott-knott effect size difference (esd) test," 2016, https://cran.r-project.org/web/ packages/ScottKnottESD/index.html.

[65] ——, "Towards a better understanding of the impact of experimental components on defect prediction modelling," in *38th International Conference on Software Engineering Companion*, 2016, p. 867–870.

[66] C. Tantithamthavorn and A. E. Hassan, "An experience report on defect modelling in practice: pitfalls and challenges," in *the 40th International Conference on Software Engineering: Software Engineering in Practice*, 2018, pp. 286–295.

[67] C. Tantithamthavorn, S. Mclntosh, A. E. Hassan, and K. Matsumoto, "Automated parameter optimization of classification techniques for defect prediction models," in *38th International Conference on Software*

Engineering, 2016, pp. 321–332.
[68] ——, "An empirical comparison of model validation techniques for defect prediction models," *IEEE Transactions on Software Engineering*, vol. 43, no. 1, pp. 1–18, 2017.
[69] C. Tantithamthavorn, A. E. Hassan, and K. Matsumoto, "The impact of class rebalancing techniques on the performance and interpretation of defect prediction models," *IEEE Transactions on Software Engineering (Early Access)*, pp. 1–20, 2018.
[70] 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, 2019.
[71] A. Tosun, B. Turhan, and A. Bener, "Validation of network measures as indicators of defective modules in software systems," in *ACM International Conference Proceeding Series*, 01 2009, pp. 1–9.
[72] Z. Tóth, P. Gyimesi, and R. Ferenc, "A public bug database of github projects and its application in bug prediction," in *International Conference on Computational Science and Its Applications*, 2016, pp. 625–636.
[73] B. Turhan, T. Menzies, A. B.Bener, and J. D. Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Software Engineering*, vol. 14, pp. 540–578, 2009.
[74] P. Vuttipittayamongkol, EyadElyan, and AndreiPetrovski, "On the class overlap problem in imbalanced data classification," *Knowledge-Based Systems*, vol. 212, pp. 1–17, 2021.
[75] R. wu, H. Zhang, S. Kim, and S. Cheung, "Relink: Recovering links between bugs and changes," in *the 19th ACM SIGSOFT Symposium on Foundations of Software Engineering*, 09 2011, pp. 15–25.
[76] H. Xiong, J. Wu, and LuLiu, "Classification with classoverlapping: A systematic study," in *Proceedings of the 1st International Conference on E-Business Intelligence*, 2010, pp. 491–497.
[77] Z. Xu, L. Li, M. Yan, J. Liu, X. Luo, J. Grundy, Y. Zhang, and X. Zhang, *The Journal of Systems and Software*, vol. 172, pp. 1–22, 2021.
[78] X. Yang, D. Lo, X. Xia, and J. Sun, "Tlel: A two-layer ensemble learning approach for just-in-time defect prediction," *Information and Software Technology*, vol. 87, pp. 206–220, 2017.
[79] Y. Yang, Y. Zhou, J. Liu, Y. Zhao, H. Lu, L. Xu, B. Xu, and H. Leung, "Effort-aware just-in-time defect prediction: simple unsupervised models could be better than supervised models," in *24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 157–168.
[80] S. Yatish, J. Jiarpakdee, P. Thongtanunam, and C. Tantithamthavorn, "Mining software defects: Should we consider affected releases?" in *2019 IEEE/ACM 41st International Conference on Software Engineering*, 05 2019, pp. 1–12.
[81] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," *Journal of Machine Learning Research*, vol. 5, pp. 1205–1224, 2004.
[82] Z. Yu, C. Theisen, L. Williams, and T. Menzies, "Improving vulnerability inspection efficiency using active learning," *IEEE Transactions on Software Engineering*, vol. 47, no. 1, pp. 2401–2420, 2021.
[83] T. Zimmermann and N. Nagappan, "Predicting defects using network analysis on dependency graphs," in *The International Conference on Software Engineering*, 2008.
[84] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," in *Third International Workshop on Predictor Models in Software Engineering*, 05 2007, pp. 9–20.
[85] 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 *7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, 2009, pp. 91–100.

Lina Gong is currently an assistant professor in the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. She received her Ph.D. degree in the Computer software and theory from China University of Mining and Technology, China, her BE degree in Software Engineering from China University of Petroleum, China. She also studied as a visitor one year in the Software Analysis and Intelligence Lab (SAIL), School of Computing, Queen's University, Canada. Her research interests include machine learning, software analysis, software testing and mining software repositories. More information at: https://lina-gong.github.io/.

Haoxiang Zhang is a research fellow at Queen's University, Canada. His research interests include empirical software engineering, mining software repositories, and intelligent software analytics. He received a PhD in Computer Science from Queen's University, Canada. He received a PhD in Physics and MSc in Electrical Engineering from Lehigh University, and obtained his BSc in Physics from the University of Science and Technology of China. Contact haoxiang.zhang@acm.org. More information at: https://haoxianghz.github.io/.

Jingxuan Zhang is an assistant professor of College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. Zhang received the Ph.D. degree in software engineering from the Dalian University of Technology, China. His current research interests include mining software repositories and software data analytics.

Mingqiang Wei received his Ph.D degree (2014) in Computer Science and Engineering from the Chinese University of Hong Kong (CUHK). He is a professor at the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA). Before joining NUAA, he served as an assistant professor at Hefei University of Technology, and a postdoctoral fellow at CUHK. He was a recipient of the CUHK Young Scholar Thesis Awards in 2014. He is now an Associate Editor for the Visual Computer Journal, Journal of Electronic Imaging, Journal of Image and Graphics, and a Guest Editor for IEEE Transactions on Multimedia. His research interests focus on 3D vision, computer graphics, software engineering, pre-training model, and deep learning.

Zhiqiu Huang is a professor of College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. He received the Ph.D. degree in computer science from the Nanjing University of Aeronautics and Astronautics in 1999. He has authored over 80 papers in referred journals and proceedings in the areas of software engineering and knowledge engineering. His research interests include software engineering, formal methods, and knowledge engineering.

APPENDIX A
THE SPECIFIC EXECUTION PROCESS OF IDENTIFYING OVERLAPPING INSTANCE

Figure 14 shows the specific execution process of our proposed algorithm in Section 2.2. Specifically, we firstly produce the five nearest neighbour through Euclidean Distance in line 3 in figure 14. We then get the number of neighbours that are with the same label as the current instance in line 5-9 in figure 14. Finally, we get the overlap status of each instance based on the results generated by previous step in lines 10-14 in figure 14. As a result, the proportion of neighbours with the same label against neighbours with the opposite label ranges from 5:0 (all five neighbours have the same label with the given instance) to 0:5 (all five neighbour have the opposite label with the given instance). Depending on the proportion, we define proportions of the neighbours 5:0, 4:1 or 3:2 as non-overlapping instance; while 2:3, 1:4 or 0:5 as overlapping instance. Based on this method, we can identify all overlapping instances in SDP datasets.

Alogrithm 1: Idenfitying overlapping instances method

Input: SDP datasets from {AEEEM, Github, Industrial, NASA, PROMISE, ReLink, SOFTLAB, Zimmemann}

```
1  for each SDP dataset in {AEEEM, Github, Industrial, NASA,
                PROMISE, ReLink, SOFTLAB, Zimmemann}:
2      for each instance i in SDP dataset:
3          Calculate euclidean distance to produce the five
               nearest neighbours List_i
4          number=0
5          for each neighbour j in List_i:
6              if label(i) == label(j):
7                  number=number+1
8              end if
9          end for
10         if number>=3:
11             Overlap_label(i)=0
12         else:
13             Overlap_label(i)=1
14         end if
15     end for
16 end for
17 return overlap_label of SDP datasets.
```

Fig. 14: The execution process of our proposed algorithm.

In addition, different distance functions would produce different K-nearest neighbors [28]. In the SDP datasets, Kim et al. [33] and Chen et al. [11] have both employed the Euclidean distance to find the K nearest neighbors, leading to a better performance. Therefore, in our study, we also employ the Euclidean distance to identify the K nearest neighbors. Furthermore, to indicate the justification of our choices about how to identify an overlap instance, we discuss the impact of different parameters (i.e., p, distance schemes, and cluster methods) in Section 6.

APPENDIX B
THE SPECIFIC EXECUTION PROCESS OF COMPARED IDENTIFYING OVERLAPPING INSTANCE METHODS

The K-Means method is motivated by our prior study [20]. In this method, instances are divided into K clusters by K-Means, where K is determined by ensuring that the number of instances in each cluster is not more than 20. Then, the overlapping status of each cluster is identified through the proportion of defective and non-defective instances (described in Eq. 2) that are similar as our proposed method.

$$Overlapping\ \ instances\ \ in\ \ each\ \ cluster = \begin{cases} defective\ \ instances & P < 0.4 \\ all\ \ instances & 0.4 \leqslant P \leqslant 0.6 \\ non-defectiveinstances & P > 0.6 \end{cases} \quad (2)$$

SMR is an overlap rate score of an instance assigned by the SVM classifier. The calculating formula is defined as

$$SMR(i) = \sum_{j=1}^{m} a_j y_j G(x_j, \chi) + b \quad (3)$$

where a_j represents a Lagrange multiplier and the instances with $a_j > 0$ are called support vectors. b represents an intercept. The higher the SMR(i) score is, the better the instance i is defined. Thus, we sort the instances in ascending order and choose the top $p\%$ instances as overlapping instances in our study.

The SVDD method was proposed by Xiong et al. [76] who used the support vector data description to find overlapping regions in datasets. In this method, SVDD was firstly used to obtain the decision function for each class (defective or non-defective). Then, instances satisfying the decision function and greater than 0 in both defective and non-defective models are identified as overlapping instances.

APPENDIX C
THE SPECIFIC EXECUTION PROCESS OF CLASS OVERLAPPING HANDLING TECHNIQUES

Figure 15 provides different execution processes of these two techniques. The removing technique directly removes the overlapping instances identified by our proposed KNN method before building the SDP models. The separating technique divides the training datasets into overlapping and non-overlapping datasets. Through this, we divide the training datasets into non-overlap training dataset (just include the non-overlapping instances) and overlap training dataset (just include the overlapping instances). Then, we build the SDP models for each dataset respectively.

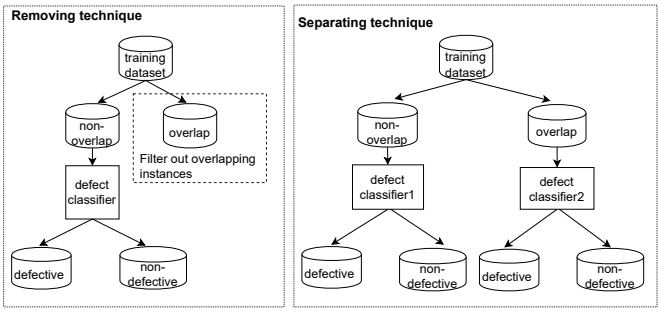


Fig. 15: The specific execution process of the removing and separating techniques.

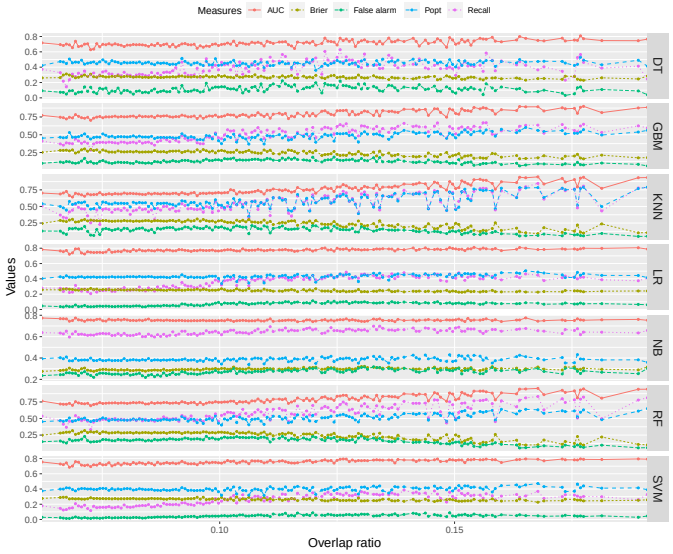


Fig. 17: Performance of models trained on derby-10.3.1.4 project with different levels of class overlap.

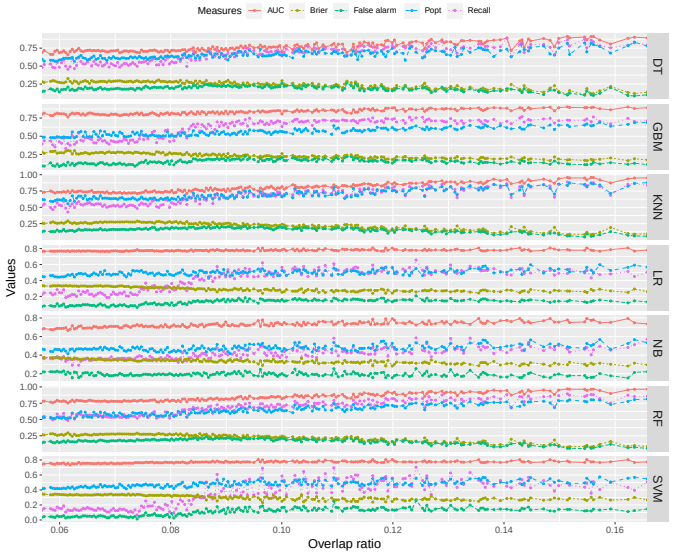


Fig. 18: Performance of defect prediction models trained on Prop-1-92 project with different levels of class overlap.

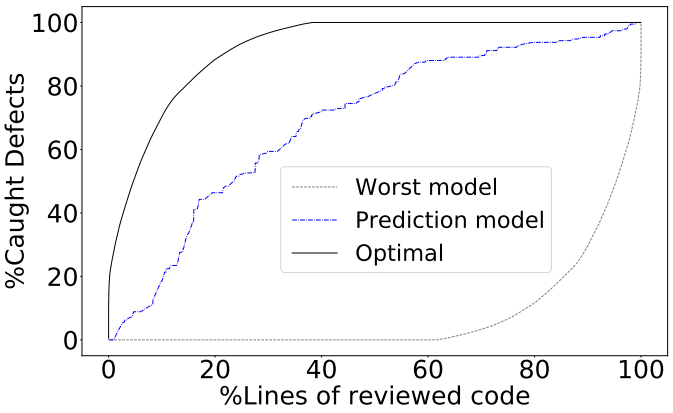


Fig. 16: Cost-Effectiveness curve for an Effort-aware model.

APPENDIX D
THE DEFINITION OF P_{opt}

Figure 16 provides an example of the performance of a prediction model, which is a Alberg diagram. The x-axis represents the cumulative percentage of reviewed code, while y-axis represents the cumulative percentage of detected defects. To calculate P_{opt}, we need two additional models: the optimal model and the worst model. These two models are both based on the actual defect densities (i.e., the number of defects / lines of code). The optimal model is sorted in decreasing order, and the worst model is sorted in ascending order. Thus, P_{opt} can be defined as

$$P_{opt} = \frac{Area_{\pi}(m) - Area_{\pi}(Worst)}{Area_{\pi}(Optimal) - Area_{\pi}(Worst)} \quad (4)$$

where a high value of P_{opt} indicates good performance.

APPENDIX E
EXPERIMENTAL RESULTS

Tables 7 and 8 present the comparison performance of SDP models trained on removing overlapping instances identified by different methods. Furthermore, in the Tables 7 and 8 present the statistical results for studied SDP datasets based on different overlapping identification methods calculated by statistical methods in Section 4.

Figure 17 presents the performance of defect prediction models trained on derby-10.3.1.4 project with different levels of class overlap. Note that the different levels of class overlap are generated by the approach in Section 5.3.

Figure 18 presents the performance of defect prediction models trained on derby-10.3.1.4 project with different levels of class overlap. Note that the different levels of class overlap are generated by the approach in Section 5.3.

Tables 9, 10, 11 and 12 present the comparisons between different class overlap handling techniques on 18 datasets in terms of Recall, Brier-score, False alarm and P_{opt}. Note that for CLA and CLAMI method, the statistical results presented in columns CLA and CLAMI are calculated between models trained on the RF classifiers and models with CLA and CLAMI methods. The statistical results presented in columns of the DT, NB, KNN, LR, SVM, GBM and RF are calculated among the models trained on original datasets and models trained on datasets with different handling overlapping techniques (i.e., removing and separating).

Tables 13, 14, and 15 show the important features in TOP-1, 2 and 3 ranks with the studied seven classifiers for each of studied SDP datasets.

TABLE 7: The comparison performance of SDP models trained on removing overlapping instances identified by different methods.

Project	Measures	Original	KNN	K-Means	SMR	SVDD	Combine
Camel−1.2	$AUC^{\uparrow}$	0.616	**0.682** $(L)^{***}$	$0.673(M)^{***}$	0.618	0.607	0.639^{**}
	$Recall^{\uparrow}$	0.115	$0.235(M)^{***}$	**0.542**$(L)^{***}$	0.098	0.129	0.115
	$Brier-score^{\downarrow}$	**0.218**	0.226	0.235	**0.218**	0.220	**0.218**
	$Falsealarm^{\downarrow}$	0.036	0.056	0.305	**0.023**	0.034	0.034
	$P_{opt}{}^{\uparrow}$	0.492	**0.560**$(M)^{***}$	**0.560**$(M)^{***}$	0.527^{*}	0.529^{*}	0.512
derby−10.2.1.6	$AUC^{\uparrow}$	0.774	**0.794** $(M)^{***}$	0.791 $(S)^{**}$	0.775	0.721	0.775
	$Recall^{\uparrow}$	0.453	$0.519(L)^{***}$	**0.660**$(L)^{***}$	0.457^{*}	0.377	$0.468(M)^{***}$
	$Brier-score^{\downarrow}$	**0.175**	0.176	0.186	0.177	0.189	0.179
	$Falsealarm^{\downarrow}$	0.114	0.120	0.210	0.111	**0.084**$(M)^{***}$	0.123
	$P_{opt}{}^{\uparrow}$	0.439	0.460^{*}	0.445^{*}	0.451^{*}	0.439	0.438
derby−10.3.1.4	$AUC^{\uparrow}$	0.777	**0.798**$(L)^{***}$	0.796^{*}	0.776	0.778	0.782^{*}
	$Recall^{\uparrow}$	0.368	$0.454(L)^{***}$	**0.591**$(L)^{***}$	$0.405(M)^{***}$	0.360	0.391^{*}
	$Brier-score^{\downarrow}$	0.168	0.166	0.174	0.169	0.166	**0.165**
	$Falsealarm^{\downarrow}$	0.075	0.091	0.160	0.095	**0.073**	0.081
	$P_{opt}{}^{\uparrow}$	0.429	**0.484**$(L)^{***}$	$0.458(M)^{***}$	0.432	0.434	0.427
eclipse34−debug	$AUC^{\uparrow}$	0.780	0.794^{**}	0.789^{*}	0.777	0.781	0.786
	$Recall^{\uparrow}$	0.224	$0.333(L)^{***}$	**0.429**$(L)^{***}$	0.082	0.218	0.250^{**}
	$Brier-score^{\downarrow}$	**0.151**	0.152	0.158	0.156	0.152	0.152
	$Falsealarm^{\downarrow}$	0.037	0.047	0.098	**0.016**$(M)^{***}$	0.038	0.044
	$P_{opt}{}^{\uparrow}$	0.417	0.486^{**}	0.432	0.420	0.433	0.424
prop−1−164	$AUC^{\uparrow}$	0.817	0.821	**0.829** $(M)^{***}$	0.828^{**}	0.808	0.824^{**}
	$Recall^{\uparrow}$	0.163	0.289^{**}	0.269	0.081	0.151	0.102
	$Brier-score^{\downarrow}$	0.063	**0.060**	0.057	0.066	0.065	0.065
	$Falsealarm^{\downarrow}$	0.004	0.006	0.013	**0.000**	0.004	0.003
	$P_{opt}{}^{\uparrow}$	0.623	**0.687**$(L)^{***}$	0.611	$0.667(L)^{***}$	0.622	0.619
prop−1−44	$AUC^{\uparrow}$	0.951	0.956	**0.970**$(M)^{***}$	0.959	0.950	0.957
	$Recall^{\uparrow}$	0.290	$0.372(M)^{***}$	**0.514**$(L)^{***}$	0.180	0.146	0.280
	$Brier-score^{\downarrow}$	0.053	**0.046**	0.053	0.054	0.056	0.052
	$Falsealarm^{\downarrow}$	0.004	0.006	0.010	**0.001**	0.005	0.004
	$P_{opt}{}^{\uparrow}$	0.760	0.793^{*}	0.751	**0.825**$(L)^{***}$	0.741	0.786
prop−1−92	$AUC^{\uparrow}$	0.836	$0.888(L)^{***}$	**0.890**$(L)^{***}$	0.826	0.835	0.837
	$Recall^{\uparrow}$	0.583	$0.719(L)^{***}$	**0.805**$(L)^{***}$	0.494	0.577	0.611
	$Brier-score^{\downarrow}$	0.160	**0.126**$(L)^{***}$	$0.132(M)^{***}$	0.164	0.160	0.158
	$Falsealarm^{\downarrow}$	0.117	0.109	0.162	**0.088**	0.111	0.128
	$P_{opt}{}^{\uparrow}$	0.622	**0.725**$(L)^{***}$	0.618	$0.713(L)^{***}$	$0.650(L)^{***}$	0.598
prop−2−256	$AUC^{\uparrow}$	0.865	**0.920**$(L)^{***}$	$0.914(M)^{***}$	0.863	0.872	0.870
	$Recall^{\uparrow}$	0.509	$0.729(L)^{***}$	**0.812**$(L)^{***}$	$0.588(L)^{***}$	0.515	0.502
	$Brier-score^{\downarrow}$	0.143	**0.101**$(L)^{***}$	$0.111(L)^{***}$	0.164	0.160	0.158
	$Falsealarm^{\downarrow}$	0.064	0.076	0.124	0.085	0.061	**0.056**
	$P_{opt}{}^{\uparrow}$	0.695	**0.779**$(L)^{***}$	$0.724(M)^{***}$	$0.737(L)^{***}$	$0.726(M)^{***}$	0.706
prop−3−318	$AUC^{\uparrow}$	0.960	0.968^{*}	0.960	0.954	0.960	0.960
	$Recall^{\uparrow}$	0.453	$0.604(L)^{***}$	**0.72**$(L)^{***}$	0.382	0.421	0.452
	$Brier-score^{\downarrow}$	0.061	0.053^{*}	0.049	0.069	0.062	0.061
	$Falsealarm^{\downarrow}$	**0.009**	0.017	0.034	0.012	0.066	0.008
	$P_{opt}{}^{\uparrow}$	0.839	0.859^{*}	0.828	0.786	0.830	0.827
prop−4−355	$AUC^{\uparrow}$	0.839	**0.900**$(L)^{***}$	$0.899(M)^{***}$	0.834	0.842	0.836
	$Recall^{\uparrow}$	0.446	$0.688(L)^{***}$	**0.771**$(L)^{***}4$	0.406	0.452	0.446
	$Brier-score^{\downarrow}$	0.157	**0.118**$(L)^{***}$	$0.123(M)^{***}$	0.162	0.157	0.157
	$Falsealarm^{\downarrow}$	0.068	0.091	0.133	**0.056**	0.064	0.069
	$P_{opt}{}^{\uparrow}$	0.607	**0.718**$(L)^{***}$	$0.622(M)^{***}$	$0.707(L)^{***}$	$0.633(M)^{***}$	0.594

TABLE 8: The Median values of SDP models trained on removing overlapping instances identified by different methods.

Project	Measures	Original	KNN	K-Means	SMR	SVDD	Combine
prop−5−121	$AUC^{\uparrow}$	0.812	**0.833**$(L)^{***}$	$0.831(M)^{***}$	0.818	0.819	0.815
	$Recall^{\uparrow}$	0.096	$0.202(M)^{***}$	**0.353**$(L)^{***}$	0.092	0.084	0.107
	$Brier-score^{\downarrow}$	0.091	**0.087**	0.092	0.090	0.089	0.089
	$Falsealarm^{\downarrow}$	0.004	0.007	0.026	0.004	**0.003**	0.005
	$P_{opt}{}^{\uparrow}$	0.577	**0.662**$(L)^{***}$	0.574	0.583	0.581	0.583
prop−5−157	$AUC^{\uparrow}$	0.791	0.807^{*}	0.782	0.806	0.810^{*}	0.805
	$Recall^{\uparrow}$	0.115	0.173^{*}	**0.314**$(L)^{***}$	0.081	0.110	0.114
	$Brier-score^{\downarrow}$	0.094	0.097	0.097	0.094	0.093	**0.092**
	$Falsealarm^{\downarrow}$	0.007	0.008	0.035	**0.004**	0.006	0.007
	$P_{opt}{}^{\uparrow}$	0.487	0.508^{**}	0.492	0.492	0.493	0.485
prop−5−185	$AUC^{\uparrow}$	0.830	0.838	0.816	0.838	**0.839**	0.823
	$Recall^{\uparrow}$	0.065	$0.102(M)^{***}$	**0.138**$(L)^{***}$	0.069	0.044	0.061
	$Brier-score^{\downarrow}$	**0.062**	0.064	0.065	**0.062**	**0.062**	**0.062**
	$Falsealarm^{\downarrow}$	0.002	0.005	0.007	0.003	**0.001**	0.002
	$P_{opt}{}^{\uparrow}$	0.566	**0.631**$(L)^{***}$	0.563	$0.599(M)^{***}$	0.580^{**}	0.558
prop−5−4	$AUC^{\uparrow}$	0.795	0.812	0.791	0.815	0.817	**0.819**
	$Recall^{\uparrow}$	0.038	$0.199(M)^{***}$	**0.278**$(L)^{***}$	0.000	0.056	0.034
	$Brier-score^{\downarrow}$	0.063	**0.056**	0.075	0.063	0.061	0.062
	$Falsealarm^{\downarrow}$	**0.000**	0.002	0.002	**0.000**	0.002	0.001
	$P_{opt}{}^{\uparrow}$	0.574	**0.710**$(L)^{***}$	0.614^{*}	0.598	0.604^{*}	$0.643(L)^{***}$
prop−5−40	$AUC^{\uparrow}$	0.941	0.931	0.926	0.944	0.945	**0.946**
	$Recall^{\uparrow}$	0.390	0.432^{*}	**0.438**$(M)^{***}$	0.355	0.392	0.390
	$Brier-score^{\downarrow}$	**0.058**	0.061	0.060	0.060	**0.057**	**0.057**
	$Falsealarm^{\downarrow}$	0.004	0.006	0.008	**0.003**	**0.003**	0.004
	$P_{opt}{}^{\uparrow}$	0.776	0.788	0.768	0.784	**0.792**	0.786
prop−5−85	$AUC^{\uparrow}$	0.836	**0.886**$(L)^{***}$	$0.875(M)^{***}$	0.841	0.842	0.840
	$Recall^{\uparrow}$	0.325	**0.567**$(L)^{***}$	$0.413(L)^{***}$	0.334	0.328	0.329
	$Brier-score^{\downarrow}$	0.138	**0.115**$(L)^{***}$	0.133	0.136	0.137	0.137
	$Falsealarm^{\downarrow}$	0.025	0.048	0.047	0.026	0.025	**0.023**
	$P_{opt}{}^{\uparrow}$	0.642	**0.705**$(L)^{***}$	0.646	0.647	$0.656(M)^{***}$	0.638
xalan−2.5	$AUC^{\uparrow}$	0.706	**0.778**$(L)^{***}$	$0.752(M)^{***}$	0.699	0.703	0.707
	$Recall^{\uparrow}$	0.617	$0.739(L)^{***}$	**0.911**$(L)^{***}$	0.586	0.624	0.635
	$Brier-score^{\downarrow}$	0.223	**0.196**$(L)^{***}$	0.260	0.225	0.223	0.223
	$Falsealarm^{\downarrow}$	0.313	0.316	0.535	**0.282**	0.323	0.321
	$P_{opt}{}^{\uparrow}$	0.394	**0.498**$(L)^{***}$	$0.443(M)^{***}$	0.320	0.316	0.380
xalan−2.6	$AUC^{\uparrow}$	0.836	**0.860**$(L)^{***}$	0.843^{*}	0.817	0.824	0.837
	$Recall^{\uparrow}$	0.634	0.675^{**}	**0.838**$(L)^{***}$	0.471	0.627	0.631
	$Brier-score^{\downarrow}$	0.171	**0.157**$(L)^{***}$	0.189	0.190	0.174	0.167
	$Falsealarm^{\downarrow}$	0.141	0.133	0.338	**0.094**	0.142	0.141
	$P_{opt}{}^{\uparrow}$	0.275	$0.327(L)^{***}$	$0.335(L)^{***}$	**0.457**$(L)^{***}$	0.304^{**}	0.277

(1) *** means $p < 0.0001$, ** means $p < 0.001$, * means $p < 0.05$.
(2) L/M: Large/Medium effect size according to Cliff's delta.
(3) '↓' indicates 'the smaller the better';'↑' indicates 'the larger the better'.

TABLE 9: Comparisons between different class overlap handling techniques on 18 datasets in terms of Recall.

Overlapping	Project	Handling techniques	DT	NB	KNN	LR	SVM	GBM	RF	CLA	CLAMI
$R \geq 12.5\%$	Camel-1.2	Original	0.222	0.159	0.387	0.085	0.131	0.354	0.115	**0.522**	0.145
		Removing	0.268	0.212$(L)^{***}$	0.240	0.059	0.225$(L)^{***}$	0.254	0.235$(L)^{***}$		
		Separating	0.289$(M)^{***}$	0.233$(L)^{***}$	0.333	0.212$(L)^{***}$	0.327$(L)^{***}$	0.333	0.310$(L)^{***}$		
	derby-10.2.16	Original	0.433	0.478	0.473	0.407	0.539	0.501	0.453	**0.691**	0.091$(L)^{***}$
		Removing	0.476$(L)^{***}$	0.477	0.434	0.440$(M)^{***}$	0.560$(L)^{***}$	0.513*	0.519$(L)^{***}$		
		Separating	0.475$(M)^{***}$	0.487*	0.432	0.446$(M)^{***}$	0.538	0.508	0.512$(L)^{***}$		
	derby-10.3.1.4	Original	0.428	0.455	0.460	0.378	0.327	0.480	0.368	**0.693**	0.140$(L)^{***}$
		Removing	0.427	0.436	0.392	0.383*	0.460$(L)^{***}$	0.458	0.454$(L)^{***}$		
		Separating	0.428	0.447	0.410	0.410$(M)^{***}$	0.467$(L)^{***}$	0.470	0.442$(L)^{***}$		
	eclipse34_debug	Original	0.427	0.255	0.381	0.147	0.200	0.380	0.224	**0.615**	0.098$(L)^{***}$
		Removing	0.362	0.227	0.290	0.150	0.240$(L)^{***}$	0.322	0.333$(L)^{***}$		
		Separating	0.381	0.257	0.316	0.239$(L)^{***}$	0.276$(L)^{***}$	0.351	0.345$(L)^{***}$		
	prop-1-92	Original	0.674	0.311	0.659	0.509	0.622	0.683	0.583	0.381$(L)^{***}$	0.091$(L)^{***}$
		Removing	0.685*	0.315	0.642	0.577$(M)^{***}$	0.630	0.694*	**0.719**$(L)^{***}$		
		Separating	0.672	0.342$(L)^{***}$	0.641	0.598$(M)^{***}$	0.608	0.687	0.704$(L)^{***}$		
	prop-2-256	Original	0.717	0.210	0.681	0.530	0.248	0.689	0.509	0.340$(L)^{***}$	0.059$(L)^{***}$
		Removing	0.707	0.313$(L)^{***}$	0.670	0.506	0.603$(L)^{***}$	0.697	*0.729$(L)^{***}$		
		Separating	0.746$(L)^{***}$	0.338$(L)^{***}$	0.684	0.612$(L)^{***}$	0.530$(L)^{***}$	0.717$(M)^{***}$	**0.760**$(L)^{***}$		
	prop-4-355	Original	0.604	0.152	0.671	0.351	0.422	0.681	0.446	0.265$(L)^{***}$	0.070$(L)^{***}$
		Removing	0.678$(L)^{***}$	0.182$(L)^{***}$	0.631	0.438$(M)^{***}$	0.533$(L)^{***}$	0.669	0.688$(L)^{***}$		
		Separating	0.678$(L)^{***}$	0.197$(L)^{***}$	0.649	0.485$(L)^{***}$	0.577$(L)^{***}$	0.692	**0.694**$(L)^{***}$		
	prop-5-85	Original	0.492	0.460	0.563	0.350	0.313	0.580	0.325	0.499$(L)^{***}$	0.238$(L)^{***}$
		Removing	0.589$(L)^{***}$	0.456	0.505	0.398$(M)^{***}$	0.440$(L)^{***}$	0.554	0.567$(L)^{***}$		
		Separating	**0.593**$(L)^{***}$	0.510$(L)^{***}$	0.542	0.450$(L)^{***}$	0.475$(L)^{***}$	0.583	0.571$(L)^{***}$		
	xalan-2.5	Original	0.554	0.402	0.577	0.380	0.479	0.606	0.617	0.535$(L)^{***}$	0.179$(L)^{***}$
		Removing	0.659$(L)^{***}$	0.466$(L)^{***}$	0.610*	0.541$(L)^{***}$	0.592$(L)^{***}$	0.668$(M)^{***}$	**0.739**$(L)^{***}$		
		Separating	0.648$(L)^{***}$	0.443$(L)^{***}$	0.585	0.530$(L)^{***}$	0.585$(L)^{***}$	0.643$(M)^{***}$	0.709$(L)^{***}$		
	xalan-2.6	Original	0.611	0.599	0.635	0.613	0.578	0.671	0.634	0.510$(L)^{***}$	0.193$(L)^{***}$
		Removing	0.636$(M)^{***}$	0.585	0.606	0.613	0.594$(M)^{***}$	0.649	**0.675**$(L)^{***}$		
		Separating	0.638$(M)^{***}$	0.583	0.633	0.652$(L)^{***}$	0.625$(L)^{***}$	0.662	0.655$(M)^{***}$		
$R < 12.5\%$	prop-1-164	Original	0.430	0.450	0.328	0.127	0.156	0.323	0.163	**0.749**	0.197$(L)^{***}$
		Removing	0.252	0.357	0.134	0.095	0.215$(M)^{***}$	0.258	0.289$(L)^{***}$		
		Separating	0.268	0.371	0.244	0.215$(L)^{***}$	0.302$(L)^{***}$	0.324	0.310$(L)^{***}$		
	prop-1-44	Original	**0.734**	0.514	0.407	0.110	0.251	0.409	0.290	0.641	0.098$(L)^{***}$
		Removing	0.386	0.449	0.306	0.161$(M)^{***}$	0.315$(L)^{***}$	0.378	0.372$(L)^{***}$		
		Separating	0.571	0.705$(L)^{***}$	0.260	0.443$(L)^{***}$	0.381$(L)^{***}$	0.453$(M)^{**}$	0.411		
	prop-3-318	Original	0.709	0.854	0.577	0.533	0.504	0.605	0.453	0.486$(L)^{***}$	0.079$(L)^{***}$
		Removing	0.508	0.772	0.400	0.543	0.541$(L)^{***}$	0.549	0.604$(L)^{***}$		
		Separating	0.723*	**0.901**$(L)^{***}$	0.488	0.596$(M)^{***}$	0.587$(L)^{***}$	0.589	0.656		
	prop-5-121	Original	0.206	0.390	0.282	0.139	0.047	0.285	0.096	**0.645**	0.336
		Removing	0.286$(L)^{***}$	0.351	0.111	0.119	0.153$(L)^{***}$	0.199	0.202$(L)^{***}$		
		Separating	0.336$(L)^{***}$	0.526$(L)^{***}$	0.193	0.188$(M)^{***}$	0.214$(L)^{***}$	0.254	0.145$(L)^{***}$		
	prop-5-157	Original	0.206	0.388	0.228	0.191	0.011	0.252	0.115	**0.659**	0.299
		Removing	0.225*	0.356	0.081	0.122	0.119$(L)^{***}$	0.163	0.173$(L)^{***}$		
		Separating	0.258$(M)^{***}$	0.392*	0.180	0.221$(M)^{***}$	0.213$(L)^{***}$	0.260	0.130⋆		
	prop-5-185	Original	0.196	0.000	0.216	0.132	0.047	0.202	0.065	**0.730**	0.000$(L)^{***}$
		Removing	0.240$(M)^{***}$	0.000	0.115	0.074	0.133$(L)^{***}$	0.151	0.102$(L)^{***}$		
		Separating	0.292$(L)^{***}$	0.224$(L)^{***}$	0.162	0.152	0.178$(L)^{***}$	0.191	0.160$(L)^{***}$		
	prop-5-4	Original	0.140	0.359	0.256	0.068	0.038	0.249	0.038	**0.572**	0.342
		Removing	0.253$(L)^{***}$	0.301	0.121	0.039	0.078	0.171	0.199$(L)^{***}$		
		Separating	0.304$(L)^{***}$	0.443$(L)^{***}$	0.175	0.098	0.133$(L)^{***}$	0.228	0.296$(L)^{***}$		
	prop-5-40	Original	0.499	**0.698**	0.454	0.147	0.093	0.457	0.390	0.553	0.097$(L)^{***}$
		Removing	0.486	0.624	0.373	0.408	0.405$(L)^{***}$	0.424	0.432$(M)^{***}$		
		Separating	0.528*	0.680	0.410	0.379$(L)^{***}$	0.440$(L)^{***}$	0.453	0.474$(L)^{***}$		

(1) *** means $p < 0.0001$, ** means $p < 0.001$, * means $p < 0.05$.
(2) L/M: Large/Medium effect size according to Cliff's delta.
(3) The statistical results of CLA and CLAMI are calculated with models with RF classifier trained with removing overlapping instances.

TABLE 10: Comparisons between different class overlap handling techniques on 18 datasets in terms of Brier-score.

Overlapping	Project	Handling techniques	DT	NB	KNN	LR	SVM	GBM	RF	CLA	CLAMI
$R \geq 12.5\%$	Camel-1.2	Original	0.283	0.228	0.290	0.228	0.232	0.296	**0.218**	$0.449(L)^{***}$	$0.272(L)^{***}$
		Removing	0.310	0.248	0.294	0.247	0.245	$0.261(L)^{***}$	0.226		
		Separating	0.318	0.266	0.308	0.260	0.255	0.275^{*}	0.230		
	derby-10.2.16	Original	0.231	0.232	0.229	0.177	0.185	0.220	**0.175**	$0.319(L)^{***}$	$0.233(L)^{***}$
		Removing	0.222	0.251	$0.197(L)^{***}$	0.188	0.190	$0.196(L)^{***}$	0.176		
		Separating	0.223	0.282	$0.210(L)^{***}$	0.197	0.197	0.207^{*}	**0.175**		
	derby-10.3.1.4	Original	0.223	0.211	0.222	0.168	0.174	0.198	0.168	$0.329(L)^{***}$	$0.234(L)^{***}$
		Removing	0.205^{*}	0.223	$0.185(L)^{***}$	0.178	0.183	$0.188(M)^{***}$	**0.166**		
		Separating	0.208^{*}	0.249	$0.194(L)^{***}$	0.184	0.188	0.195	0.172		
	eclipse34_debug	Original	0.278	0.170	0.205	0.167	0.172	0.187	**0.151**	$0.337(L)^{***}$	$0.208(L)^{***}$
		Removing	$0.216(L)^{***}$	0.185	$0.180(M)^{***}$	0.177	0.181	0.178	0.152		
		Separating	$0.223(L)^{***}$	0.220	0.192^{*}	0.180	0.190	0.189	0.160		
	prop-1-92	Original	0.220	0.234	0.201	0.182	0.179	0.160	0.160	$0.454(L)^{***}$	$0.237(L)^{***}$
		Removing	$0.193(M)^{***}$	0.245	$0.173(L)^{***}$	0.186	$0.169(L)^{***}$	0.156	**0.126**$(L)^{***}$		
		Separating	$0.197(M)^{***}$	0.242	$0.180(L)^{***}$	0.185	$0.170(L)^{***}$	0.164	$0.134(L)^{***}$		
	prop-2-256	Original	0.176	0.197	0.158	0.177	0.195	0.137	0.143	$0.421(L)^{***}$	$0.234(L)^{***}$
		Removing	0.159^{*}	0.185	$0.145(L)^{***}$	0.170	$0.145(L)^{***}$	$0.126(L)^{***}$	**0.101**$(L)^{***}$		
		Separating	0.197	0.198	$0.143(L)^{***}$	0.170	$0.136(L)^{***}$	$0.124(L)^{***}$	0.143		
	prop-4-355	Original	0.173	0.215	0.178	0.183	0.188	0.141	0.157	$0.440(L)^{***}$	$0.237(L)^{***}$
		Removing	0.194	0.214	$0.161(M)^{***}$	$0.172(L)^{***}$	$0.156(L)^{***}$	0.142	**0.118**$(L)^{***}$		
		Separating	0.200	0.215	$0.162(M)^{***}$	$0.170(L)^{***}$	$0.155(L)^{***}$	0.142	$0.125(L)^{***}$		
	prop-5-85	Original	0.157	0.210	0.161	0.152	0.150	0.139	0.138	$0.315(L)^{***}$	$0.214(L)^{***}$
		Removing	0.173	0.221	$0.142(L)^{***}$	$0.145(L)^{***}$	$0.140(L)^{***}$	0.133	**0.115***		
		Separating	0.195	0.261	$0.153(L)^{***}$	0.149^{*}	$0.142(L)^{***}$	0.141	0.149		
	xalan-2.5	Original	0.293	0.249	0.280	0.247	0.243	0.273	0.223	$0.421(L)^{***}$	$0.282(L)^{***}$
		Removing	0.319	0.254	0.271^{*}	0.244	0.247	$0.243(L)^{***}$	**0.196**$(L)^{***}$		
		Separating	0.319	0.249	0.278^{*}	0.248	0.247	$0.248(M)^{***}$	$0.201(L)^{***}$		
	xalan-2.6	Original	0.233	0.211	0.230	0.187	0.185	0.218	0.171	$0.401(L)^{***}$	$0.271(L)^{***}$
		Removing	0.219^{*}	0.210	$0.212(L)^{***}$	0.189	0.196	0.202	**0.157**$(L)^{***}$		
		Separating	0.221^{*}	0.210	0.224^{*}	0.194	0.196	0.209	$0.159(L)^{***}$		
$R < 12.5\%$	prop-1-164	Original	0.105	0.108	0.080	0.072	0.077	0.068	0.063	$0.359(L)^{***}$	$0.172(L)^{***}$
		Removing	$0.076(L)^{***}$	0.091	0.072	0.076	$0.071(L)^{***}$	0.068	**0.060**		
		Separating	$0.081(L)^{***}$	0.099	0.073	0.074	$0.070(L)^{***}$	0.070	0.063		
	prop-1-44	Original	0.050	0.097	0.062	0.070	0.072	0.059	0.053	$0.355(L)^{***}$	$0.517(L)^{***}$
		Removing	0.056	0.082	0.062	0.066^{*}	$0.067(L)^{***}$	0.056^{*}	**0.046**$(L)^{***}$		
		Separating	0.096	0.165	0.060	$0.065(M)^{***}$	$0.064(L)^{***}$	0.054^{*}	0.055		
	prop-3-318	Original	0.084	0.131	0.071	0.076	0.071	0.072	0.061	$0.319(L)^{***}$	$0.162(L)^{***}$
		Removing	0.074	0.120	0.071	0.075	0.068	0.068	**0.053**$(L)^{***}$		
		Separating	0.142	0.159	0.070	0.077	0.069	0.070	0.102		
	prop-5-121	Original	0.108	0.146	0.106	0.094	0.100	0.095	0.091	$0.285(L)^{***}$	$0.165(L)^{***}$
		Removing	0.118	0.138	0.099	0.098	0.104	0.095	**0.087**		
		Separating	0.146	0.215	0.103	0.101	0.107	0.098	0.092		
	prop-5-157	Original	0.113	0.147	0.109	0.093	0.101	0.100	**0.094**	$0.285(L)^{***}$	$0.163(L)^{***}$
		Removing	0.126	0.138	0.109	0.105	0.109	0.105	0.097		
		Separating	0.159	0.138	0.118	0.111	0.113	0.114	0.100		
	prop-5-185	Original	0.074	0.071	0.073	0.067	0.072	0.068	**0.062**	$0.392(L)^{***}$	$0.195(L)^{***}$
		Removing	0.083	0.075	0.069	0.071	0.074	**0.070**	0.064		
		Separating	0.148	0.164	0.076	0.074	0.076	0.076	0.084		
	prop-5-4	Original	0.074	0.118	0.072	0.063	0.067	0.061	0.063	$0.337(L)^{***}$	$0.162(L)^{***}$
		Removing	0.068	0.105	0.065	0.068	0.068	0.062	**0.056**		
		Separating	0.082	0.210	0.069	0.071	0.071	0.065	0.089		
	prop-5-40	Original	0.074	0.122	0.076	0.080	0.085	0.072	**0.058**	$0.378(L)^{***}$	$0.204(L)^{***}$
		Removing	0.071	0.100	$0.072(M)^{***}$	$0.075(M)^{***}$	$0.074(L)^{***}$	0.071	0.061		
		Separating	0.173	0.149	0.076	0.079	$0.077(L)^{***}$	0.074	0.084		

(1) *** means $p < 0.0001$, ** means $p < 0.001$, * means $p < 0.05$.
(2) L/M: Large/Medium effect size according to Cliff's delta.
(3) The statistical results of CLA and CLAMI are calculated with models with RF classifier trained with removing overlapping instances.

TABLE 11: Comparisons between different class overlap handling techniques on 18 datasets in terms of False alarm.

Overlapping	Project	Handling techniques	DT	NB	KNN	LR	SVM	GBM	RF	CLA	CLAMI
$R \geq 12.5\%$	Camel-1.2	Original	0.152	0.096	0.262	0.028	0.083	0.250	**0.036**	0.435$(L)^{***}$	0.101$(L)^{***}$
		Removing	0.118(*	0.040$(L)^{***}$	0.108$(L)^{***}$	**0.014**	0.078	0.084$(L)^{***}$	0.056		
		Separating	0.152	0.165	0.182$(L)^{***}$	0.153	0.173	0.170$(L)^{***}$	0.128		
	derby-10.2.16	Original	0.160	0.188	0.183	**0.087**	0.068	0.173	0.114	0.321$(L)^{***}$	0.022
		Removing	0.141*	0.184	0.090$(L)^{***}$	0.091	0.099	0.115$(L)^{***}$	0.120		
		Separating	0.149*	0.234	0.117$(L)^{***}$	0.119	0.124	0.138$(L)^{***}$	0.121		
	derby-10.3.1.4	Original	0.147	0.176	0.169	0.079	0.047	0.145	0.075	0.338$(L)^{***}$	0.047
		Removing	0.125*	0.161	0.073$(L)^{***}$	**0.070**	0.081	0.089$(L)^{***}$	0.091		
		Separating	0.127*	0.206	0.096$(L)^{***}$	0.095	0.102	0.109$(L)^{***}$	0.110		
	eclipse34_debug	Original	0.185	0.079	0.142	0.024	0.037	0.115	0.037	0.324$(L)^{***}$	0.034
		Removing	0.085$(L)^{***}$	0.064*	0.052$(L)^{***}$	**0.021**	0.041	0.052$(L)^{***}$	0.047		
		Separating	0.100$(L)^{***}$	0.151	0.086$(L)^{***}$	0.069	0.077	0.085$(L)^{***}$	0.057		
	prop-1-92	Original	0.169	0.148	0.160	0.142	0.174	0.152	0.117	0.365$(L)^{***}$	0.031
		Removing	0.148$(L)^{***}$	0.149	0.125$(L)^{***}$	0.172	0.144$(M)^{***}$	0.125$(L)^{***}$	**0.109**		
		Separating	0.151$(L)^{***}$	0.169	0.137*	0.204	0.157$(M)^{***}$	0.138$(L)^{***}$	0.113		
	prop-2-256	Original	0.128	0.089	0.135	0.068	0.041	0.120	**0.064**	0.315$(L)^{***}$	0.047
		Removing	0.107$(M)^{***}$	0.075*	0.101*	0.096	0.099	0.088$(L)^{***}$	0.076		
		Separating	0.186	0.114	0.107$(L)^{***}$	0.103	0.099	0.094$(M)^{***}$	0.185		
	prop-4-355	Original	0.142	0.087	0.134	0.085	0.101	0.122	**0.068**	0.292$(L)^{***}$	0.038
		Removing	0.129*	0.091	0.102*	0.083	0.083	0.099$(L)^{***}$	0.091		
		Separating	0.142	0.109	0.114*	0.110	0.106	0.111	0.110		
	prop-5-85	Original	0.092	0.131	0.108	0.058	0.035	0.094	**0.025**	0.245$(L)^{***}$	0.075$(L)^{***}$
		Removing	0.086$(L)^{***}$	0.129	0.060$(L)^{***}$	0.039*	0.049	0.052$(L)^{***}$	0.048		
		Separating	0.116	0.238	0.083$(L)^{***}$	0.071	0.072	0.078	0.110		
	xalan-2.5	Original	0.340	0.300	0.350	**0.276**	0.336	0.336	0.313	0.377$(L)^{***}$	0.135
		Removing	0.348	0.334	0.324	0.396	0.358	0.310*	0.316		
		Separating	0.348	0.286	0.313	0.391	0.362	0.320	0.333		
	xalan-2.6	Original	0.203	0.297	0.221	0.173	0.135	0.233	0.141	0.321$(L)^{***}$	0.107
		Removing	0.171$(M)^{***}$	0.254$(M)^{***}$	0.137$(L)^{***}$	0.176	0.153	0.171$(L)^{***}$	0.133		
		Separating	0.173$(M)^{***}$	0.265$(M)^{***}$	0.180$(M)^{***}$	0.219	0.179	0.205$(M)^{***}$	**0.123**		
$R < 12.5\%$	prop-1-164	Original	0.059	0.102	0.030	0.008	0.006	0.021	0.004	0.370$(L)^{***}$	0.007$(L)^{***}$
		Removing	0.011$(L)^{***}$	0.063$(M)^{***}$	0.004$(L)^{***}$	**0.003**	0.006	0.008$(L)^{***}$	0.006		
		Separating	0.017$(L)^{***}$	0.075$(M)^{***}$	0.014$(L)^{***}$	0.016	0.014	0.017	0.009		
	prop-1-44	Original	0.029	0.089	0.023	0.007	0.018	0.018	**0.004**	0.351$(L)^{***}$	0.017$(L)^{***}$
		Removing	0.012$(L)^{***}$	0.056$(M)^{***}$	0.005$(L)^{***}$	0.006	0.012	0.007$(L)^{***}$	0.006		
		Separating	0.071	0.214	0.013*	0.017	0.016	0.014	0.033		
	prop-3-318	Original	0.047	0.180	0.035	0.026	0.019	0.037	0.009	0.282$(L)^{***}$	0.027$(L)^{***}$
		Removing	0.026$(L)^{***}$	0.158*	**0.006**$(L)^{***}$	0.010	0.016	0.016	0.017		
		Separating	0.128	0.269	0.017$(L)^{***}$	0.028	0.030	0.026	0.170		
	prop-5-121	Original	0.031	0.125	0.044	0.012	0.000	0.034	**0.004**	0.275$(L)^{***}$	0.072$(L)^{***}$
		Removing	0.036	0.089$(L)^{***}$	0.010$(L)^{***}$	0.005$(L)^{***}$	0.011	0.010$(L)^{***}$	0.007		
		Separating	0.066	0.230	0.023$(L)^{***}$	0.021	0.026	0.024$(L)^{***}$	0.010		
	prop-5-157	Original	0.035	0.125	0.043	0.016	0.000	0.038	0.007	0.274$(L)^{***}$	0.063$(L)^{***}$
		Removing	0.032	0.088$(L)^{***}$	**0.003**$(L)^{***}$	0.006$(L)^{***}$	0.007	0.010$(L)^{***}$	0.008		
		Separating	0.075	0.176	0.029	0.032	0.033	0.034	0.009		
	prop-5-185	Original	0.022	0.000	0.025	0.008	0.015	0.021	**0.002**	0.403$(L)^{***}$	0.000
		Removing	0.024	0.000	0.006	0.003$(L)^{***}$	0.007	0.008$(L)^{***}$	0.005		
		Separating	0.098	0.168	0.017	0.016	0.018	0.019*	0.034		
	prop-5-4	Original	0.016	0.106	0.029*	0.004	0.002	0.016	**0.000**	0.330$(L)^{***}$	0.076$(L)^{***}$
		Removing	0.012	0.074***	0.006$(L)^{***}$	0.002	0.003	0.003$(L)^{***}$	0.002		
		Separating	0.035	0.211	0.015	0.011	0.011	0.012*	0.047		
	prop-5-40	Original	0.033	0.115	0.028	0.014	0.005	0.025	**0.004**	0.370$(L)^{***}$	0.067$(L)^{***}$
		Removing	0.017$(L)^{***}$	0.076***	0.006$(L)^{***}$	0.009$(L)^{***}$	0.012	0.008$(L)^{***}$	0.006		
		Separating	0.090	0.154	0.016	0.021	0.022	0.019$(L)^{***}$	0.072		

(1) *** means $p < 0.0001$, ** means $p < 0.001$, * means $p < 0.05$.
(2) L/M: Large/Medium effect size according to Cliff's delta.
(3) The statistical results of CLA and CLAMI are calculated with models with RF classifier trained with removing overlapping instances.

TABLE 12: Comparisons between different class overlap handling techniques on 18 datasets in terms of P_{opt}.

Overlapping	Project	Handling techniques	DT	NB	KNN	LR	SVM	GBM	RF	CLA	CLAMI
$R \geq 12.5\%$	Camel-1.2	Original	0.516	0.490	0.575	0.426	0.462	0.518	0.492	$0.452(M)^{***}$	**0.666**
		Removing	$0.588(M)^{***}$	0.472	$0.616(M)^{***}$	$0.473(L)^{***}$	$0.533(L)^{***}$	$0.538(M)^{***}$	$0.560(L)^{***}$		
		Separating	$0.594(M)^{***}$	0.481	$0.607(M)^{***}$	0.501	$0.532(L)^{***}$	$0.552(M)^{***}$	$0.585(L)^{***}$		
	derby-10.2.16	Original	0.509	0.355	0.516	0.439	0.457	0.481	0.439	0.425^{*}	**0.603**
		Removing	0.454	0.354	0.481	0.431	$0.494(L)^{***}$	0.490	0.460^{*}		
		Separating	0.460	$0.415(L)^{***}$	0.499	0.451	$0.503(L)^{***}$	0.495	0.457^{*}		
	derby-10.3.1.4	Original	0.501	0.370	0.561	0.443	0.441	0.499	0.429	0.443	**0.593**
		Removing	0.477	0.377	0.498	0.443	$0.510(L)^{***}$	0.509	$0.484(M)^{**}$		
		Separating	0.477	0.387^{*}	0.512	0.453	$0.514(L)^{***}$	0.518	$0.483(M)^{**}$		
	eclipse34_debug	Original	0.611	0.532	0.554	0.364	0.480	0.524	0.417	$0.345(L)^{***}$	**0.613**
		Removing	0.584	0.521	0.529	0.395^{*}	0.460	0.495	$0.486(M)^{***}$		
		Separating	0.581	0.514	0.536	$0.426(L)^{***}$	0.483	0.511	$0.490(M)^{***}$		
	prop-1-92	Original	0.722	0.577	0.711	0.539	0.606	0.644	0.622	$0.536(L)^{***}$	$0.647(L)^{***}$
		Removing	0.705	0.577	0.729^{*}	0.539	0.637^{*}	0.676^{*}	$0.725(L)^{***}$		
		Separating	0.701	0.586^{*}	0.721	0.548	0.638^{*}	0.674^{*}	$0.706(L)^{***}$		
	prop-2-256	Original	**0.793**	0.640	0.777	0.595	0.593	0.728	0.695	$0.269(L)^{***}$	$0.656(L)^{***}$
		Removing	0.777	$0.710(L)^{***}$	0.789^{*}	0.619^{*}	$0.738(L)^{***}$	$0.754(M)^{***}$	$0.779(L)^{***}$		
		Separating	0.778	0.654^{*}	0.787^{*}	0.612^{*}	$0.735(L)^{***}$	$0.758(M)^{***}$	$0.764(L)^{***}$		
	prop-4-355	Original	0.646	0.581	**0.761**	0.505	0.593	0.717	0.607	$0.301(L)^{***}$	$0.619(L)^{***}$
		Removing	$0.756(L)^{***}$	0.584	0.744	$0.528M^{**}$	$0.641(L)^{***}$	0.689	$0.718(L)^{***}$		
		Separating	$0.758(L)^{***}$	0.574	0.755	$0.554(L)^{***}$	$0.637(L)^{***}$	0.702	$0.720(L)^{***}$		
	prop-5-85	Original	0.663	0.693	**0.702**	0.577	0.615	0.666	0.644	$0.503(L)^{***}$	0.690
		Removing	$0.748(L)^{***}$	0.689	0.727^{*}	$0.658(L)^{***}$	$0.668(L)^{***}$	$0.692(M)^{***}$	0.705		
		Separating	$0.746(L)^{***}$	0.646	0.733^{*}	$0.657(L)^{***}$	$0.667(L)^{***}$	$0.695(M)^{***}$	0.688		
	xalan-2.5	Original	0.550	0.389	0.588	0.237	0.435	0.541	0.394	0.282	0.503
		Removing	**0.645**$(L)^{***}$	0.394	0.608^{*}	$0.360(L)^{***}$	0.442	0.517	$0.498(L)^{***}$		
		Separating	**0.645**$(L)^{***}$	0.375	0.618^{*}	$0.381(L)^{***}$	0.454	0.524	$0.477(L)^{***}$		
	xalan-2.6	Original	0.471	0.260	0.483	0.241	0.292	0.306	0.275	0.623	**0.770**
		Removing	0.500^{*}	0.252	0.475	0.251	0.301	0.294	0.327		
		Separating	0.500^{*}	0.263	0.501	0.261	0.311	0.299	0.299		
$R < 12.5\%$	prop-1-164	Original	0.674	0.640	0.678	0.529	0.570	0.649	0.623	$0.591(L)^{***}$	$0.640(L)^{***}$
		Removing	0.620	0.639	0.647	$0.581(L)^{***}$	$0.650(L)^{***}$	0.651	0.687^{*}		
		Separating	0.624	0.631	0.664	$0.612(L)^{***}$	$0.671(L)^{***}$	0.668	0.681^{*}		
	prop-1-44	Original	**0.810**	0.714	0.709	0.595	0.639	0.705	0.760	$0.524(L)^{***}$	$0.559(L)^{***}$
		Removing	0.786	0.693	0.673	$0.660(L)^{***}$	$0.728(L)^{***}$	$0.786(L)^{***}$	0.793^{*}		
		Separating	0.766	0.716	0.697	$0.685(L)^{***}$	$0.757(L)^{***}$	$0.803(L)^{***}$	0.787^{*}		
	prop-3-318	Original	0.836	0.847	0.804	0.709	0.744	0.790	0.839	$0.482(L)^{***}$	$0.701(L)^{***}$
		Removing	0.841	0.849	0.816	$0.793(L)^{***}$	0.791^{*}	0.824^{*}	**0.859**		
		Separating	0.817	0.790	0.822^{*}	$0.797(L)^{***}$	0.795^{*}	0.824^{*}	0.851		
	prop-5-121	Original	0.594	0.647	0.639	0.529	0.540	0.591	0.577	0.574^{*}	**0.701**
		Removing	$0.686(L)^{***}$	0.645	0.655^{*}	$0.609(L)^{***}$	$0.635(L)^{***}$	$0.633(M)^{***}$	$0.662(L)^{***}$		
		Separating	$0.674(L)^{***}$	0.623	0.661^{*}	$0.619(L)^{***}$	$0.636(L)^{***}$	$0.637(M)^{***}$	$0.646(L)^{***}$		
	prop-5-157	Original	0.524	0.566	0.566	0.502	0.515	0.547	0.487	0.501	**0.642**
		Removing	$0.573(M)^{***}$	0.561	0.522	0.480	0.471	0.500	0.508^{*}		
		Separating	$0.584(M)^{***}$	0.552	0.565	0.525^{*}	0.517	0.542	0.524^{*}		
	prop-5-185	Original	0.621	0.521	0.627	0.494	0.546	0.610	0.566	$0.457(L)^{***}$	0.601
		Removing	**0.653**	0.530	0.626	0.527^{*}	$0.631(L)^{***}$	0.626^{*}	$0.631(L)^{***}$		
		Separating	0.623	0.474	0.634	$0.622(L)^{***}$	$0.632(L)^{***}$	0.635^{*}	$0.623(L)^{***}$		
	prop-5-4	Original	0.626	0.690	0.701	0.558	0.586	0.643	0.574	$0.471(L)^{***}$	0.711
		Removing	**0.729**$(L)^{***}$	0.673	0.723^{*}	$0.673(L)^{***}$	$0.665(L)^{***}$	$0.681(M)^{***}$	$0.710(L)^{***}$		
		Separating	$0.722(L)^{***}$	0.652	0.718^{*}	$0.678(L)^{***}$	$0.672(L)^{***}$	$0.686(M)^{***}$	$0.711(L)^{***}$		
	prop-5-40	Original	0.754	0.786	0.709	0.627	0.627	0.698	0.776	$0.488(L)^{***}$	$0.643(L)^{***}$
		Removing	0.760	**0.788**	0.731	$0.704(L)^{***}$	$0.714(L)^{***}$	$0.748(L)^{***}$	**0.788**		
		Separating	0.740	0.769	0.734	$0.699(L)^{***}$	$0.716(L)^{***}$	$0.749(L)^{***}$	0.779		

(1) *** means $p < 0.0001$, ** means $p < 0.001$, * means $p < 0.05$.
(2) L/M: Large/Medium effect size according to Cliff's delta.
(3) The statistical results of CLA and CLAMI are calculated with models with RF classifier trained with removing overlapping instances.

TABLE 13: The important features in TOP-1, 2 and 3 ranks with the studied seven classifiers for each of studied SDP datasets

Project	Classifiers	Top-1 rank		Top-2 rank		Top-3 rank	
		Original	Removing	Original	Removing	Original	Removing
Camel -1.2	DT	ca, avg_cc,lcom	ca, lcom, avg_cc	cam	cam, amc	amc, lcom3, dit, cbm, moa, noc	lcom3, cbm, dit, noc, moa
	NB	noc	noc	avg_cc	avg_cc	lcom	lcom
	KNN	cam, avg_cc	cbm, cam, avg_cc	dit, lcom, lcom3, cbm, noc, amc	dit, lcom, amc, noc, moa	moa, ca	ca, lcom3
	SVM	avg_cc, noc	avg_cc	moa, cbm	cam, cbm, noc, moa	cam, lcom	lcom3, lcom
	LR	noc, avg_cc	noc, lcom3	lcom3, moa, lcom, ca	lcom, avg_cc	cbm	ca
	GBM	ca, lcom	ca	avg_cc	lcom, avg_cc	amc, dit, cbm, moa, noc, lcom3, cam	dit, amc, cbm
	RF	ca	ca	avg_cc, noc, lcom	lcom, avg_cc	moa	noc, moa, cbm, lcom3
derby -10.2.16	DT	CountLineComment	CountLineComment	MaxInheritanceTree	MaxInheritanceTree	RatioCommentToCode CountDeclInstance-Variable, CountClassCoupled	CountDeclInstance-Variable, CountClassCoupled
	NB	CountClassCoupled	CountClassCoupled	CountLineComment, CountDeclMethod-Private	CountLineComment, CountDeclMethod-Private	CountDeclClass, CountDeclMethodPublic, CountDeclInstance-Variable	CountDeclClass, CountDeclMethodPublic, CountDeclInstance-Variable
	KNN	MaxInheritanceTree	MaxInheritanceTree	CountLineComment, CountClassCoupled	CountClassCoupled, CountLineComment	CountDeclMethodPublic, PercentLackOfCohesion	CountDeclMethodPublic, PercentLackOfCohesion
	SVM	MaxInheritanceTree	MaxInheritanceTree	CountLineComment	CountLineComment	CountDeclMethodPublic, $MaxNesting_{Mean}$, CountDeclClass, CountDeclMethod-Private, PercentLackOfCohesion	CountDeclMethod-Public, MaxNesting_Mean
	LR	CountLineComment	CountLineComment, CountClassCoupled	CountDeclMethodPrivate	CountDeclMethodPrivate	CountClassCoupled	RatioCommentToCode
	GBM	CountLineComment, MaxInheritanceTree	CountLineComment	CountDeclInstanceVariable, MaxNesting_Mean, CountDeclMethodPublic, CountDeclMethod-Private	MaxInheritanceTree	AvgLineComment, CountDeclMethodProtected, RatioCommentToCode, PercentLackOfCohesion, CountClassCoupled, CountDeclMethodDefault	CountDeclInstance-Variable, CountDeclMethodPublic, MaxNesting_Mean
	RF	CountLineComment	CountLineComment	MaxInheritanceTree	MaxInheritanceTree	CountDeclMethodPrivate, CountDeclInstanceVariable, CountClassCoupled	CountDeclMethodPrivate
derby -10.3.1.4	DT	CountLineComment	CountLineComment	MaxInheritanceTree	RatioCommentToCode	CountClassCoupled, RatioCommentToCode	RatioCommentToCode, CountClassCoupled
	NB	CountLineComment	CountLineComment	CountClassCoupled	CountClassCoupled	CountDeclMethodPrivate	CountDeclMethodPrivate
	KNN	MaxInheritanceTree	MaxInheritanceTree	CountClassCoupled	CountClassCoupled	CountLineComment	CountLineComment
	SVM	CountLineComment, MaxInheritanceTree	MaxInheritanceTree	CountClassCoupled	CountLineComment, CountClassCoupled	AvgLineComment, CountDeclMethodPrivate	MaxNesting_Mean, CountDeclMethodPrivate, AvgLineComment
	LR	CountLineComment	CountLineComment	RatioCommentToCode	RatioCommentToCode	AvgLineComment	MaxInheritanceTree
	GBM	CountLineComment	CountLineComment	MaxInheritanceTree	MaxInheritanceTree	CountDeclMethodPrivate, AvgLineComment	AvgLineComment, CountDeclMethodPrivate
	RF	MaxInheritanceTree, CountLineComment	MaxInheritanceTree, CountLineComment	CountDeclMethodPrivate, AvgLineComment	CountDeclMethodPrivate, AvgLineComment	CountDeclMethodProtected, CountOutput_Min, CountDeclClassMethod, $MaxNesting_{Mean}$, RatioCommentToCode	CountDeclClassMethod, MaxNesting_Mean, CountOutput_Min, CountDeclMethod-Protected, RatioCommentToCode
eclipse34 _debug	DT	change_times	change_times	CBO	CBO	DIT, IFANIN	IFANIN, NCV, DIT
	NB	change_times	change_times	NCV	NCV	CBO	CBO
	KNN	change_times	change_times, IFANIN	IFANIN, CBO	DIT, CBO	DIT, NCV	NCV, NIM
	SVM	change_times	change_times	CBO, NCV	CBO	NIM, IFANIN, NIV, NCM	NCV
	LR	change_times	change_times	CBO	NCV	NCV	CBO
	GBM	change_times	change_times	CBO	CBO	NCV	NCV
	RF	change_times	change_times	CBO, NCV	CBO	NIM	NCV

TABLE 14: The important features in TOP-1, 2 and 3 ranks with the studied seven classifiers for each of studied SDP datasets

Project	Classifiers	Top-1 rank		Top-2 rank		Top-3 rank	
		Original	Removing	Original	Removing	Original	Removing
prop-1 -164	DT	ndpv	amc, ndpv	amc	ndc	ndc, ce	ce
	NB	ndpv	ndpv	ndc	ndc	ca, ce	ca, ce
	KNN	ndpv	ndpv	ndc	ndc	ce	amc, lcom3
	SVM	ndpv	ndpv	ndc	ndc	lcom3, ce	ce
	LR	ndc	ndc	lcom3	lcom3	npm	ndpv
	GBM	ndpv	ndpv	ndc	ndc	ce	ce
	RF	ndpv	ndpv	ndc, ce	ndc, ce	lcom3, ca, amc	lcom3
prop-1 -44	DT	ndpv	ndpv	cam	cam	mfa	mfa
	NB	ndpv	ndpv	cam	cam	ca	ca
	KNN	ndpv	ndpv	cam	cam	mfa	mfa, lcom3
	SVM	ndpv	ndpv	cam	cam	mfa	mfa
	LR	ndpv	ndpv	mfa, cam	mfa	lcom	cam
	GBM	ndpv	ndpv	cam	cam	mfa	mfa
	RF	ndpv	ndpv	cam	cam	ce, npm, mfa	ce, npm
prop-1 -92	DT	amc	amc	mfa	mfa	lcom3, avg.cc., ce	ce, avg.cc., lcom3
	NB	avg.cc.	avg.cc.	lcom3, amc	lcom3, amc	ndc	ndc
	KNN	lcom3	amc	avg.cc.	avg.cc.	amc, ce	lcom3, ce
	SVM	ce	amc	amc, lcom3	ce	avg.cc.	lcom3
	LR	ce	lcom3	lcom3, avg.cc.	ce	ic	avg.cc.
	GBM	amc	amc	mfa	mfa	ce	ce
	RF	amc	amc	mfa	mfa	ce	ce
prop-2 -256	DT	amc	amc	dit, mfa	dit	avg.cc., npm	mfa
	NB	avg.cc.	avg.cc.	amc	amc	dit, mfa	dit, mfa
	KNN	avg.cc.	dit	amc, dit	amc	lcom3	avg.cc.
	SVM	dit	dit	amc	amc	mfa	mfa
	LR	mfa	mfa	avg.cc.	avg.cc.	amc	dit
	GBM	amc	amc	mfa	dit, mfa	dit	avg.cc.
	RF	amc	amc	dit	dit	mfa	avg.cc., mfa
prop-4 -355	DT	amc	amc	mfa	mfa	dit, lcom3	lcom3, dit
	NB	amc	amc	max.cc.	max.cc.	dit	dit
	KNN	dit	dit	amc	mfa	mfa	amc
	SVM	dit	dit	amc	amc	mfa	mfa
	LR	mfa, dit	dit	amc	mfa	dam	amc
	GBM	amc	amc	mfa	mfa	ca, dit	dit
	RF	amc	amc	mfa	mfa	dit	dit
prop-5 -121	DT	cbo	cbo	lcom3, nr, ndpv	lcom3	avg.cc., cam, mfa	nr, avg.cc., mfa, cam, ndpv
	NB	ndpv	ndpv	nr	nr	cbo	cbo
	KNN	ndpv	ndpv, mfa, lcom3	cbo, nr	nr, cbo, amc, dam	lcom3	avg.cc.
	SVM	nr, ndpv	ndpv, nr	cbo	cbo	lcom	dam, lcom3
	LR	cbo	cbo	amc	amc	nr, ndpv	ndpv
	GBM	nr, cbo	cbo, nr	ndpv, ca, avg.cc., mfa, lcom3, cam	ca	amc, npm, ic, noc, moa, lcom, dam	mfa, avg.cc., ndpv, lcom3
	RF	nr	nr	cbo	cbo	ndpv	ndpv
prop-5 -157	DT	cbo	cbo	cam, nml	amc	amc	cam, nml
	NB	ndpv	ndpv	nml	nml	cbo	cbo
	KNN	nml	cbo, nml	cbo	amc	amc	ca, npm
	SVM	nml	nml	cbo	cbo	ndpv	ndpv
	LR	cbo	cbo	nml, ca	nml	amc, dam	ca
	GBM	cbo	cbo	cam, nml, amc	amc	ic	nml, cam
	RF	cbo, cam	cbo, cam	nml	nml	amc, ndpv	amc

TABLE 15: The important features in TOP-1, 2 and 3 ranks with the studied seven classifiers for each of studied SDP datasets

Project	Classifiers	Top-1 rank		Top-2 rank		Top-3 rank	
		Original	Removing	Original	Removing	Original	Removing
prop-5 -185	DT	cbo	cbo	avg.cc.	avg.cc., cam	cam, npm	npm
	NB	cbo	cbo	ca	ca	lcom	lcom
	KNN	cbo	cbo	amc, mfa, dam, ca, avg.cc.	mfa, dam	moa, npm, lcom3, lcom, cam	moa, ca, lcom3, amc
	SVM	cbo	cbo	dam, amc, avg.cc., mfa	mfa, dam	lcom3, ca, moa	avg.cc., amc, lcom3
	LR	cbo	cbo	ca	ca	amc, lcom, cam	amc
	GBM	cbo	cbo	avg.cc.	avg.cc.	amc	amc
	RF	cbo	cbo	avg.cc., amc, cam	avg.cc.	dam, lcom3	amc, cam
prop-5 -4	DT	lcom3, cbo, ca	cbo, lcom3	npm, amc	amc, ca, npm	mfa	dam, cam, mfa, ndpv
	NB	ndpv	ndpv	cbo	cbo	ca	ca
	KNN	lcom	lcom	amc, npm, cbo	amc, cbo, npm	ca	ca
	SVM	ndpv	ndpv	cbo	cbo	ca, mfa, avg.cc.	ca, amc, avg.cc., mfa
	LR	cbo	cbo	ca	ca	ndpv, amc	ndpv, amc
	GBM	amc, cbo	amc	ca, mfa	cbo	lcom3, ndpv	ca
	RF	cam, cbo, ndpv, mfa	amc	ca, npm, dam	ca, cbo, mfa	avg.cc., lcom, lcom3, cam	ndpv, dam, lcom3, npm
prop-5 -40	DT	ndpv	ndpv	nr, cam	nr, cam	lcom	lcom
	NB	nr	nr	ndpv	ndpv	cam	cam
	KNN	nr	nr	amc	amc	cbo	cbo
	SVM	ndpv	ndpv	nr	nr	cam	cam
	LR	ndpv	nr	nr	ndpv	cam	cam
	GBM	ndpv	ndpv	nr, cam	nr	lcom	cam
	RF	ndpv	ndpv	nr	nr	cam	cam
prop-5 -85	DT	cbo, cam	cbo, cam	ndc	amc, ndc, mfa	mfa	lcom3
	NB	ndc	ndc	ndpv	ndpv	cbo	cbo
	KNN	cbo	cbo	amc	amc	lcom	lcom
	SVM	cbo, ndc	cbo	cam	ndc	mfa, amc, lcom3	cam
	LR	cbo	cbo	ndc	ndc	cam	cam
	GBM	cbo	cbo	mfa	mfa	cam	cam, ndc, amc
	RF	cbo	cbo	amc	amc, mfa	mfa	ndc
xalan -2.5	DT	amc	amc	lcom3, lcom	lcom3, lcom	ca, npm, dit	dit, ca, npm
	NB	avg_cc, npm	avg_cc, npm	noc, lcom	noc, lcom	moa, dit	moa, dit
	KNN	dit, dam	dit	avg_cc	dam	moa, lcom3, amc, npm	avg_cc
	SVM	dit	dit	dam, lcom3	dam	avg_cc, amc	lcom3
	LR	npm	avg_cc	dit, avg_cc	npm	lcom, dam, noc	dam
	GBM	amc	amc	lcom	lcom	lcom3, dit	dit, lcom3, npm, ca
	RF	ndc	amc	lcom3, lcom	lcom3, lcom	ca, dit, npm	ca, dit
xalan -2.6	DT	amc	amc	npm	npm, cam	cam	lcom3
	NB	amc	amc	lcom	lcom	npm	npm
	KNN	lcom3, amc	lcom3	cam	amc	moa, avg_cc, ic, npm, dit	cam
	SVM	amc	amc	lcom3	lcom3	cam, npm	cam
	LR	amc	amc	npm	npm	lcom3	lcom3
	GBM	amc	amc	npm, cam	npm, cam	lcom3	lcom3
	RF	dit	amc	cam	cam	npm	npm