Noname manuscript No.
(will be inserted by the editor)

The Impact of Automated Feature Selection Techniques on the Interpretation of Defect Models

Jirayus Jiarpakdee · Chakkrit Tantithamthavorn · Christoph Treude

Received: date / Accepted: date

Abstract The interpretation of defect models heavily relies on software metrics that are used to construct them. Prior work often uses feature selection techniques to remove metrics that are correlated and irrelevant in order to improve model performance. Yet, conclusions that are derived from defect models may be inconsistent if the selected metrics are inconsistent and correlated. In this paper, we systematically investigate 12 automated feature selection techniques with respect to the consistency, correlation, performance, computational cost, and the impact on the interpretation dimensions. Through an empirical investigation of 14 publicly-available defect datasets, we find that (1) 94–100% of the selected metrics are inconsistent among the studied techniques; (2) 37–90% of the selected metrics are inconsistent among training samples; (3) 0–68% of the selected metrics are inconsistent when the feature selection techniques are applied repeatedly; (4) 5–100% of the produced subsets of metrics contain highly correlated metrics; and (5) while the most important metrics are inconsistent among correlation threshold values, such inconsistent most important metrics are highly-correlated with the Spearman correlation of 0.85–1. Since we find that the subsets of metrics produced by the commonly-used feature selection techniques (except for AutoSpearman) are often inconsistent and correlated, these techniques should be avoided when interpreting defect models. In addition to introducing AutoSpearman which mitigates correlated metrics better than commonly-used feature selection techniques, this paper opens up new research avenues in the automated selection of features for defect models to optimise for interpretability as well as performance.

Jirayus Jiarpakdee and Chakkrit Tantithamthavorn are with the Faculty of Information and Technology, Monash University, Melbourne, Australia
E-mail: jirayus.jiarpakdee@monash.edu, chakkrit@monash.edu
Christoph Treude is with the School of Computer Science,
The University of Adelaide, Adelaide, Australia
E-mail: christoph.treude@adelaide.edu.au

1 Introduction

Defect models are statistical or machine learning models that are used to investigate the impact of software metrics (e.g., lines of code) on defect-proneness and to identify defect-prone software modules. The interpretation of defect models is used to validate hypotheses to develop empirical theories related to software quality, which are essential to chart quality improvement plans (Jiarpakdee *et al.* 2018a; Jiarpakdee *et al.* 2020).

The conclusions of defect models heavily rely on the studied software metrics. However, software metrics often have strong correlation among themselves (Gil and Lalouche 2017; Jiarpakdee *et al.* 2016; Tantithamthavorn *et al.* 2016b; Zhang *et al.* 2017) and some metrics are irrelevant to defect models (Menzies 2018; Shepperd *et al.* 2013). For example, Jiarpakdee *et al.* (2016) find that many metrics in defect datasets are *correlated* (e.g., the `branch_count` metric is linearly proportional to the `decision_count` metric in some NASA datasets). Shepperd *et al.* (2013) find that many metrics in the NASA datasets are *irrelevant to defect models* (e.g., constant metrics).

To address these concerns, feature selection techniques are often applied in the defect prediction domain (Arisholm *et al.* 2010; D'Ambros *et al.* 2012; Elish and Elish 2008; Kaur and Malhotra 2008; Menzies *et al.* 2007; Okutan and Yıldız 2014; Shivaji *et al.* 2013). However, prior work (Arisholm *et al.* 2010; D'Ambros *et al.* 2012; Elish and Elish 2008; Kaur and Malhotra 2008; Menzies *et al.* 2007) often relies on one feature selection technique which may pose a threat to the construct validity, i.e., conclusions may not hold true if another feature selection technique is applied. In practice, feature selection techniques should be applied only on training samples to avoid producing optimistically biased performance estimates and interpretation (Li *et al.* 2017). Yet, the conclusions (e.g., the most important metric) that are derived from defect models may be *unreliable* (i.e., produce misleading importance rankings of metrics) if the produced subsets of metrics are (1) inconsistent among feature selection techniques, (2) inconsistent among different training samples, (3) inconsistent among repetitions, and (4) inconsistent among different model specifications.

In this paper, we investigate 11 commonly-used feature selection techniques and our contribution AutoSpearman along five dimensions: (1) the consistency of the produced subsets of metrics; (2) the correlation of the produced subsets of metrics; (3) the performance; (4) the computational cost; and (5) the impact on the interpretation. Through an empirical investigation of 14 publicly-available defect datasets of systems that span both proprietary and open source domains, we address the following eight research questions:

(RQ1) Do feature selection techniques consistently produce the same subset of metrics when applied on the same training sample?
Given the same training samples from the same defect datasets, *different studied feature selection techniques* produce inconsistent subsets of metrics. As many as 94-100% of the metrics are inconsistently selected among the studied feature selection techniques, suggesting that the conclusions of prior work could be altered when another feature selection technique is applied.

(RQ2) Do feature selection techniques consistently produce the same subset of metrics when applied across different training samples?
Given *different training samples from the same defect dataset*, the studied feature selection techniques produce inconsistent subsets of metrics. We find that 37-90% of the selected metrics are inconsistent, indicating that models that are trained on different training samples produce different ranking of the most important metrics. Such inconsistent subsets of metrics restrict an application of post-hoc multiple comparison analyses (e.g., a Scott-Knott test) to identify the most important metrics when interpreting defect models.

(RQ3) Do feature selection techniques which make use of random seeds consistently produce the same subset of metrics when they are applied repeatedly?
Given the same training sample from the same dataset *but different random seeds*, Recursive Feature Elimination produces inconsistent subsets of metrics. Such inconsistency amounts to 68% and 11% of the subsets of metrics for logistic regression and random forest, respectively. This finding indicates that even if Recursive Feature Elimination is applied on the same training sample, simply altering a random seed can lead the technique to produce a different subset of metrics.

(RQ4) Do feature selection techniques which make use of model specifications consistently produce the same subset of metrics when reordering the model specification of a defect model (e.g., from $y \sim x_1 + x_2$ to $y \sim x_2 + x_1$)?
Given the same training sample from the same defect dataset *but different model specifications*, regardless of the search directions, Stepwise Regression is the only studied feature selection technique that produces inconsistent subsets of metrics, suggesting that Stepwise Regression is sensitive to the model specifications of defect models.

(RQ5) Do feature selection techniques mitigate correlated metrics?
Given the same training sample from the same defect datasets, the subsets of metrics produced by all of the studied feature selection techniques (except for AutoSpearman) *contain correlated metrics, i.e., collinearity and multicollinearity.* In other words, AutoSpearman is the only studied feature selection technique which was designed to mitigate both collinearity and multicollinearity, and as our experiments show, none of the other techniques achieves these goals. These findings suggest that the interpretation of defect models constructed using the subsets of metrics that are produced by the studied feature selection techniques (except for AutoSpearman) may be misleading.

(RQ6) What is the impact of feature selection techniques on the performance of defect models?
The results of the Mann-Whitney U test show that most of the performance differences are not statistically significant (i.e., p-values > 0.05).

The results of the Kruskal-Wallis H test indicate that all of the performance differences across feature selection techniques are not statistically significant with p-values above 0.05. Finally, the results of the Cliff's $|\delta|$ effect size test show that such performance differences are negligible to small for the AUC, F-measure, and MCC measures (except for Information Gain and Chi-Squared-based). These results from statistical and effect size tests suggest that none of the studied feature selection techniques has a greater impact on the performance of defect models than the others.

(RQ7) What is the computational cost of applying feature selection techniques?
The computational cost of filter-based feature selection techniques and AutoSpearman is cheap, while such cost is expensive for wrapper-based feature selection techniques and the consistency-based feature selection technique. The computational cost of wrapper-based feature selection techniques is as high as 7 hours and 15 minutes for RFE-RF to find the best subset of metrics from one training sample of the Eclipse Platform 3.0 defect dataset. Such expensive computation cost makes the application of wrapper-based feature selection techniques undesirable, particularly, when validating with model validation techniques that require several repetitions (e.g., bootstrap validation technique).

(RQ8) Do correlation threshold values have an impact on the interpretation of defect models?
No. Although the most important metrics are inconsistent among correlation threshold values, such inconsistent most important metrics are highly correlated with the Spearman correlation of 0.85–1, suggesting that correlation threshold values do not impact the interpretation of defect models.

Our results lead us to conclude that the subsets of metrics produced by the commonly-used automated feature selection techniques are (1) inconsistent among automated feature selection techniques; (2) inconsistent among training samples; (3) inconsistent among repetitions; and (4) highly-correlated. These findings suggest that software quality improvement plans that are derived from defect models that are trained on subsets of metrics produced by commonly-used automated feature selection techniques may be inconsistent and inaccurate, leading to wasting time and resources when applying such QA plans in practice.

Since we find that the subsets of metrics produced by the commonly-used feature selection techniques (except for AutoSpearman) are often inconsistent and correlated, these techniques should be avoided when interpreting defect models. In addition to introducing AutoSpearman which mitigates correlated metrics better than commonly-used feature selection techniques, this paper opens up new research avenues in the automated selection of features for defect models to optimise for interpretability as well as performance.

This paper is an extension to our recent work (Jiarpakdee *et al.* 2018b) (i.e., RQs 1, 2, 5, and 6). This extension makes the following five contributions:

(1) An investigation of the consistency of subsets of metrics among repetitions (RQ3).

(2) An investigation of the consistency of subsets of metrics among model specifications (RQ4).
(3) An investigation of the computational cost of feature selection techniques (RQ7).
(4) An investigation of the impact of the correlation threshold values when producing subsets of metrics on the interpretation of defect models (RQ8).
(5) A series of illustrative examples to provide an in-depth demonstration of the results of our research questions. We also provide R code snippets to reproduce all of the illustrative examples in the online appendix (RQ1, RQ2, RQ3, RQ4, RQ5, and RQ8) (Jiarpakdee *et al.* 2018c).

Paper Organisation. Section 2 provides a background of the studied feature selection techniques. Section 3 elaborates on how we situate and formulate each research question from prior studies. Section 4 describes the experimental setup. Section 5 presents the approach, results, and illustrative examples with respect to the eight research questions. Section 6 discusses the trends of the commonly-selected correlated metrics. Section 7 elaborates on the threats to the validity of our study. Finally, Section 8 draws conclusions.

2 Background

Feature selection is a data preprocessing technique for selecting a subset of the best software metrics prior to constructing a defect model. There is a plethora of feature selection techniques that can be applied (Guyon and Elisseeff 2003), e.g., filter-based, wrapper-based, and embedded-based families. Since it is impractical to study all of these techniques, we would like to select a manageable set of feature selection techniques for our study. Similar to Ghotra *et al.* (2017), we select two commonly-used families of feature selection techniques, i.e., filter-based feature selection techniques and wrapper-based feature selection techniques. Thus, embedded-based feature selection techniques are excluded from our analysis, as they are rarely explored in software engineering.

In this study, we select 11 commonly-used feature selection techniques from three families (i.e., 5 filter-based, 5 wrapper-based feature selection techniques, and one hybrid-based feature selection techniques), and our own contribution AutoSpearman (Jiarpakdee *et al.* 2018b) for evaluation. For filter-based feature selection techniques, we select correlation-based (CFS), information gain (IG), chi-squared-based (χ^2), consistency-based (CON), and findCorrelation. For wrapper-based feature selection techniques, we select Recursive Feature Elimination with two classification techniques (i.e., logistic regression (RFE-LR) and random forest (RFE-RF)), and three directions of Stepwise Regression (i.e., forward (Step-FWD), backward (Step-BWD), and both (Step-BOTH)). For a hybrid-based feature selection technique, we use a combination of chi-squared based filter-based and wrapper-based using logistic regression as provided by the `HybridFS` R package (Pandari *et al.* 2019). We select these feature selection techniques since they are the most commonly used ones in previous work on defect prediction (Arisholm *et al.* 2010; D'Ambros *et al.* 2012; Elish and Elish 2008; Kaur and Malhotra 2008; Menzies *et al.* 2007; Okutan and Yıldız 2014; Shivaji *et al.* 2013). Table 1 provides a summary of the detailed implementation for the twelve studied feature selection

techniques. Below, we provide the description of each studied feature selection technique.

2.1 Filter-based feature selection techniques

Filter-based feature selection techniques search for the best subset of metrics according to an evaluation criterion regardless of model construction. Since constructing models is not required, the use of filter-based feature selection techniques is considered low cost and widely used in the defect prediction literature (Arisholm *et al.* 2010; Cahill *et al.* 2013; Elish and Elish 2008; Kaur and Malhotra 2008; Menzies *et al.* 2007; Okutan and Yıldız 2014). There are many variants of filter-based feature selection techniques, which we describe below.

Correlation-based feature selection (Hall 1999) is a deterministic feature selection technique that searches for the best subset of metrics that shares the strongest relationship with the outcome, while having a low correlation among themselves.

Information gain feature selection (Mitchell 1997) is a deterministic feature selection technique that ranks metrics according to the information gain with respect to the outcome. The information gain is measured by how much information of the outcome is provided by a metric.

Chi-Squared-based feature selection (McHugh 2013) is a deterministic feature selection technique that assesses the importance of metrics with the χ^2 statistic which is a non-parametric statistical test of independence.

Consistency-based feature selection (Dash *et al.* 2000) is a deterministic feature selection technique that uses the consistency measure (i.e., inconsistency rate) to evaluate a subset of metrics. The technique finds the optimal subset of metrics whose inconsistency rate approximates the inconsistency rate of all metrics.

findCorrelation is a deterministic feature selection technique that reduces pair-wise correlations among metrics using a correlation matrix. The technique can be applied on any correlation matrix (e.g., Pearson and Spearman). In our paper, we choose the Spearman rank correlation test instead of other correlation tests (e.g., Pearson) since the test is resilient to non-normal distributions as commonly present in defect datasets.

Table 1 summarises the detailed implementation of the studied filter-based feature selection techniques.

2.2 Wrapper-based Feature Selection Techniques

Wrapper-based feature selection techniques (John *et al.* 1994; Kohavi and John 1997) use classification techniques to assess each subset of metrics and find the best subset of metrics according to an evaluation criterion. Wrapper-based feature selection is made up of three steps, which we described below.

(Step 1) Generate a subset of metrics. Since it is impossible to evaluate all possible subsets of metrics, wrapper-based feature selection often uses search techniques (e.g., best first, greedy hill climbing) to generate candidate subsets of metrics for evaluation.

Table 1: A summary of the detailed implementation for the twelve studied feature selection techniques.

Type	Technique	R Package	R Function	Abbreviation
Filter-based Feature Selection Techniques	Correlation-based	FSelector (Romanski and Kotthoff 2013)	`cfs(class~metrics, dataset)`	CFS
	Information Gain		`information.gain(class~metrics, dataset)`	IG
	Chi-Squared-based (χ^2)		`chi.squared(class~metrics, dataset)`	Chisq
	Consistency-based		`consistency(class~metrics, dataset)`	CON

	findCorrelation	Hmisc (Harrell Jr 2013) caret (Kuhn *et al.* 2017)	`correlation.matrix = rcorr(as.matrix(dataset[, metrics]), type = 'spearman')$r`
			`correlated.metrics = findCorrelation(correlation.matrix, cutoff = 0.7, exact = TRUE)`
			`metrics[-correlated.metrics]`
				findCorrelation

Wrapper-based Feature Selection Techniques	Recursive Feature Elimination (Logistic Regression)	caret (Kuhn *et al.* 2017)	`lrFuncs.AUC = lrFuncs`
			`lrFuncs.AUC$summary = twoClassSummary`
			`control = rfeControl(functions = lrFuncs.AUC, method = "boot", number = iterations)`
			`rfe(x = dataset[, metrics], y = dataset[, class], rfeControl = control, metric = "ROC")`
				RFE-LR

	Recursive Feature Elimination (Random Forest)		`rfFuncs.AUC = rfFuncs`
			`rfFuncs.AUC$summary = twoClassSummary`
			`control = rfeControl(functions = rfFuncs.AUC, method = "boot", number = iterations)`
			`rfe(x = dataset[, metrics], y = dataset[, class], rfeControl = control, metric = "ROC")`
				RFE-RF

	Stepwise Regression (Forward Direction)	stats (Team and contributors worldwide 2017)	`null.model = glm(class~1, data = dataset, family = binomial())`
			`full.model = glm(class~metrics, data = dataset, family = binomial())`
			`step(null.model, scope = list(upper = full.model), data = dataset, direction = "fwd")`
				Step-FWD

	Stepwise Regression (Backward Direction)		`full.model = glm(class~metrics, data = dataset, family = binomial())` `step(full.model, data = dataset, direction = "bwd")`	Step-BWD

	Stepwise Regression (Both Directions)		`null.model = glm(class~1, data = dataset, family = binomial())`
			`full.model = glm(class~metrics, data = dataset, family = binomial())`
			`step(null.model, scope = list(upper = full.model), data = dataset, direction = "both")`
				Step-BOTH

Hybrid (Filter and Wrapper)	HybridFS	HybridFS (Pandari *et al.* 2019)	`HybridFS(dataset, class)`	Hybrid
	AutoSpearman	Rnalytica (Tantithamthavorn and Jiarpakdee 2018)	`AutoSpearman(dataset, metrics, spearman.threshold = 0.7, vif.threshold = 5)`	AutoSpearman

(Step 2) Construct a classifier using a subset of metrics with a predetermined classification technique. Wrapper-based feature selection constructs a classification model using a candidate subset of metrics for a given classification technique (e.g., logistic regression and random forest).

(Step 3) Evaluate the classifier according to a given evaluation criterion. Once the classifier is constructed, wrapper-based feature selection evaluates the classifier using a given evaluation criterion (e.g., Akaike Information Criterion).

For each candidate subset of metrics, wrapper-based feature selection repeats Steps 2 and 3 in order to find the best subset of metrics according to the evaluation criterion. Finally, it provides the best subset of metrics that yields the highest performance according to the evaluation criterion.

In this study, we select two commonly-used variants of wrapper-based feature selection techniques, which we describe below.

Recursive Feature Elimination (RFE) (Guyon and Elisseeff 2003) searches for the best subset of metrics by recursively eliminating the least important metric. First, RFE constructs a model using all metrics and ranks metrics according to their importance score (e.g., Breiman's Variable Importance for random forest). In each iteration, RFE excludes the least important metric and reconstructs a model. Finally, RFE provides the subset of metrics which yields the best performance according to an evaluation criterion (e.g., AUC). In our study, we select the AUC measure since it measures the discriminatory power of models, as suggested by recent research (Ghotra *et al.* 2015; Lessmann *et al.* 2008; Rahman and Devanbu 2013; Tantithamthavorn *et al.* 2019a). We use the implementation of the recursive feature elimination using the `rfe` function as provided by the `caret` R package (Kuhn *et al.* 2017).

Stepwise Regression (Chambers 1992) finds the best subset of metrics by individually assessing each metric and adding (or removing) a metric if it improves an evaluation criterion (e.g., Akaike Information Criterion). The process is repeated until there is no improvement from adding or removing a metric. In this paper, we study three directions (i.e., forward, backward, and both directions) of Stepwise Regression. We use the implementation of Stepwise Regression using the `step` function as provided by the `stats` R package (Team and contributors worldwide 2017).

2.3 Hybrid-based Feature Selection Techniques

Hybrid-based feature selection techniques (Hsu *et al.* 2011) combines filter-based and wrapper-based feature selection techniques to search for the best subset of metrics. First, a subset of candidate metrics is selected according to the evaluation criteria of filter-based feature selection techniques. Then, the subset of candidate metrics is further refined with wrapper-based feature selection techniques to identify the best subset of metrics. Hybrid-based feature selection techniques have been used in many fields of studies, e.g., medical diagnosis (Alzubi *et al.* 2017).

AutoSpearman In our recent work (Jiarpakdee *et al.* 2018b), we introduce AutoSpearman, an automated metric selection approach based on the Spearman rank correlation test and the VIF (Variance Inflation Factor) analysis for statistical inference. Below, we describe AutoSpearman using Algorithm 1, where S is a set of Spearman coefficients for each pair of metrics, C_S is a set of Spearman coefficients

Algorithm 1: AutoSpearman

Input : M is a set of studied metrics,
$sp.t$ is a threshold value for a Spearman rank correlation test,
$vif.t$ is a threshold value for a Variance Inflation Factor analysis.

Output: M' is a set of non-correlated metrics based on a Spearman rank correlation test and a Variance Inflation Factor analysis.

```
1  M' = M
2  S = Spearman(M, M)
3  C_S = {c(m_i, m_j) ∈ S | abs(c(m_i, m_j)) ≥ sp.t}
4  C_S = sort(C_S)
5  for c(m_i, m_j) in C_S do
6  |   selected.metric = min(
   |     mean(abs(Spearman(m_i, M − {m_i, m_j}))),
   |     mean(abs(Spearman(m_j, M − {m_i, m_j}))))
7  |   removed.metric ={m_i, m_j}−selected.metric
8  |   C_S = {c(m_i, m_j) ∈ C_S |
   |     m_i ≠ removed.metric ∧ m_j ≠ removed.metric}
9  |   M' = M' − removed.metric
10 end
11 repeat
12 |   V = VIF(M')
13 |   C_V = {v(m_i) ∈ V | v(m_i) ≥ vif.t}
14 |   removed.metric = {m_i |
   |     v(m_i) ∈ C_V ∧ v(m_i) = max(C_V)
15 |   M' = M' − removed.metric
16 until |C_V| = 0;
17 return M'
```

that are above a Spearman threshold value ($sp.t$), V is a set of VIF scores of metrics, C_V is a set of VIF scores of metrics that are above a VIF threshold value ($vif.t$), and M' is a set of non-correlated metrics based on the Spearman rank correlation test and the Variance Inflation Factor analysis. The high-level concept of AutoSpearman can be summarised into 2 parts:

(Part 1) Automatically select non-correlated metrics based on a Spearman rank correlation test. We first measure the correlation of all metrics using the Spearman rank correlation test (ρ) (*cf.* Line 2). We use the interpretation of correlation coefficients ($|\rho|$) as provided by Kraemer *et al.* (2003)—i.e., a Spearman correlation coefficient of above or equal to 0.7 is considered a strong correlation. Thus, we only consider the pairs that have an absolute Spearman correlation coefficient of above or equal to the threshold value ($sp.t$) of 0.7 (*cf.* Line 3).

To automatically select non-correlated metrics based on the Spearman rank correlation test, we start from the pair that has the highest Spearman correlation coefficient (*cf.* Line 4). Since the two correlated metrics under examination can be linearly predicted with each other, one of these two metrics must be removed. Thus, we select the metric that has the lowest average values of the absolute Spearman correlation coefficients of the other metrics that are not included in the pair (*cf.* Line 6). That means the removed metric is another metric in the pair that is not selected (*cf.* Line 7). Since the removed metric may be correlated with the other metrics, we remove any pairs of metrics that are correlated with the removed metric (*cf.* Line 8). Finally, we exclude the removed metric from the set

of the remaining metrics (M') (*cf.* Line 9). We repeat this process until all pairs of metrics have their Spearman correlation coefficient below a threshold value of 0.7 (*cf.* Line 5).

(Part 2) Automatically select non-correlated metrics based on a Variance Inflation Factor analysis. We first measure the magnitude of multicollinearity of the remaining metrics (M') from `Part 1` using the Variance Inflation Factor analysis (*cf.* Line 12). We use a VIF threshold value ($vif.t$) of 5 to identify the presence of multicollinearity, as suggested by Fox (2015) and prior work (Bettenburg and Hassan 2010; Jiarpakdee *et al.* 2016, 2018a; McIntosh *et al.* 2014) (*cf.* Line 13).

To automatically remove correlated metrics from the Variance Inflation Factor analysis, we identify the removed metric as the metric that has the highest VIF score (*cf.* Line 14). We then exclude the removed metric from the set of the remaining metrics (M') (*cf.* Line 15). We apply the VIF analysis on the remaining metrics until none of the remaining metrics have their VIF scores above or equal to the threshold value (*cf.* Line 16). Finally, AutoSpearman produces a subset of non-correlated metrics based on the Spearman rank correlation test and the VIF analysis (M') (*cf.* Line 17).

Similar to filter-based feature selection techniques, *Part 1* of AutoSpearman measures the correlation of all metrics using the Spearman rank correlation test regardless of model construction. Similar to wrapper-based feature selection techniques, *Part 2* of AutoSpearman constructs linear regression models to measure the magnitude of multicollinearity of metrics. Thus, we consider AutoSpearman as a hybrid-based feature selection technique (both filter-based and wrapper-based).

We also include another hybrid-based feature selection technique for comparison. We use the implementation as provided by the `HybridFS` R package (Pandari *et al.* 2019) which uses a combination of chi-squared based filter-based and wrapper-based using logistic regression. Table 2 provides an overview comparison of the studied feature selection techniques.

3 Research Questions and Related Work

Feature selection has been widely used in software engineering to remove *irrelevant* metrics (i.e., metrics that do not share a strong relationship with the outcome) (Menzies 2018; Shepperd *et al.* 2013) and *correlated* metrics (i.e., metrics that share a strong correlation with one or more metrics) (Gil and Lalouche 2017; Jiarpakdee *et al.* 2016; Tantithamthavorn *et al.* 2016b; Zhang *et al.* 2017). While several feature selection techniques have been proposed in literature, little is known about the best feature selection techniques for model interpretation.

Many ML research efforts propose feature selection techniques and investigate the impact of such techniques on the performance of prediction models (Dash *et al.* 2000; Hall and Smith 1997; Kohavi and John 1997). For example, Dash *et al.* (2000) introduce the consistency-based feature selection technique and investigated the impact of 5 search strategies of such technique on 10 benchmark datasets as provided by the UC Irwin Machine Learning repository (Blake and Merz 1998) using C4.5 and backpropagation neural network. Hall and Smith (1997) propose the correlation-based feature selection technique and evaluate the proposed technique compared with a wrapper-based feature selection technique on 15 benchmark datasets as provided by the UC Irwin Machine Learning repository (Blake

Table 2: An overview comparison of the studied feature selection techniques

Feature Selection Technique	(Step 1) Select relevant metrics			(Step 2) Construct models and select metrics based on a selection measure						Analyses
	Correlation among metrics	Correlation between metrics and the outcome	Selection Measure	Unsupervised	Supervised	Classification Technique	Validation Technique	Selection Measure	Direction	Stepwise	
Correlation-based FS	✓	✓	Pearson correlation	-	-	-	-	-	-	-	Consistency (Agreement of the selected metrics) and Performance (Accuracy) (Hall 1999)
Information Gain	✗	✓	Information Gain	-	-	-	-	-	-	-	Performance (Precision and Recall) (Lewis and Ringuette 1994)
Chi-Squared-based FS	✗	✓	Chi-squared statistics	-	-	-	-	-	-	-	Performance (Accuracy) (Huan Liu and Setiono 1995)
Consistency-based FS	✗	✓	Consistency measure	-	-	-	-	-	-	-	Computational Cost (Time) and Performance (Error rate) (Dash *et al.* 2000)
findCorrelation	✓	✗	Spearman correlation	-	-	-	-	-	-	-	Correlation (Collinearity), Performance (RMSE, R-squared, and MAE), and Interpretation (Correlation threshold values and the ranking of the most important metrics) (Aldkmin *et al.* 2019)
Recursive Feature Elimination (Logistic Regression)	-	-	-	✗	✓	Logistic Regression	Bootstrap validation	Area Under the receiver operator characteristic Curve (AUC)	Backward	✓	Performance (Accuracy, Sensitivity, Specificity, and Stability) (Yan and Zhang 2015)
Recursive Feature Elimination (Random Forest)	-	-	-	✗	✓	Random Forest	Bootstrap validation	Area Under the receiver operator characteristic Curve (AUC)	Backward	✓	
Stepwise Regression (Forward Direction)	-	-	-	✗	✓	Logistic Regression	Whole data	Akaike Information Criterion (AIC)	Forward	✓	Performance (Accuracy) (Cai *et al.* 2010)
Stepwise Regression (Backward Direction)	-	-	-	✗	✓	Logistic Regression	Whole data	Akaike Information Criterion (AIC)	Backward	✓	
Stepwise Regression (Both Directions)	-	-	-	✗	✓	Logistic Regression	Whole data	Akaike Information Criterion (AIC)	Forward and backward	✓	
HybridFS	✗	✓	Chi-squared statistics	✗	✓	Logistic Regression	Cross validation	Area Under the receiver operator characteristic Curve (AUC)	Backward	✓	-
AutoSpearman	✓	✗	Spearman correlation	✓	✗	Logistic Regression	Whole data	Variance Inflation Factor (VIF)	Backward	✓	Consistency (Agreement on the selected metrics), Correlation (Collinearity and Multicollinearity), Performance (AUC), Computational Cost (Time), and Interpretation (Consistency and Correlation of the most important metrics)

Table 3: An overview comparison of our study with respect to prior work.

Study	#Datasets	#Feature Selection Techniques	#Classification Techniques	Analyses
Ghotra *et al.* (2017)	18	29	21	Performance (AUC)
Lu *et al.* (2014)	3	5	1	Performance (Precision, Recall, Accuracy, and AUC)
Osman *et al.* (2018)	5	2	5	Consistency (Agreement on the selected metrics) and Performance (RMSE)
Rodríguez *et al.* (2007)	5	5	2	Performance (Accuracy and F-measure)
Xu *et al.* (2016)	15	20	1	Performance (AUC)
Our study	14	12	2	Consistency (Agreement on the selected metrics), Correlation (Collinearity and Multicollinearity), Performance (AUC), Computational Cost (Time), and Interpretation (Consistency and Correlation of the most important metrics)

and Merz 1998) using 3 classification techniques (i.e., IB1, Naive Bayes, and C4.5). Kohavi and John (1997) propose the wrapper-based feature selection and investigate the model performance produced by the technique comparing with 4 other filter-based feature selection techniques on 14 benchmark datasets as provided by the UC Irwin Machine Learning repository (Blake and Merz 1998) using 5 classification techniques (i.e., ID3, Naive Bayes, and C4.5). Nevertheless, little is known whether these findings hold true for the context of software engineering.

Prior studies investigate the impact of feature selection techniques on the performance of defect models (Ghotra *et al.* 2017; Lu *et al.* 2014; Osman *et al.* 2018; Rodríguez *et al.* 2007; Xu *et al.* 2016) (see Table 3). For example, Ghotra *et al.* (2017) investigate the impact of 29 feature selection techniques (as provided by Weka (Garner *et al.* 1995)) on 18 PROMISE (Menzies *et al.* 2012) and NASA (Shepperd *et al.* 2013) datasets using 21 classification techniques. Lu *et al.* (2014) investigate the impact of 4 feature selection techniques (i.e., Information Gain, Correlation-based, Forward selection, and Backward selection) on 3 releases of Eclipse datasets as provided by Zimmermann *et al.* (2007) using random forests. Osman *et al.* (2018) investigate the impact of correlation-based and wrapper-based feature selection techniques on 5 defect datasets as provided by D'Ambros *et al.* (2012) using 5 classification techniques. Rodríguez *et al.* (2007) investigate the impact of 5 feature selection techniques on 5 PROMISE datasets (Menzies *et al.* 2012) using naive bayes and C4.5. Xu *et al.* (2016) investigate the impact of 19 feature selection techniques on 11 NASA (Shepperd *et al.* 2013) and 4 AEEEM (D'Ambros *et al.* 2012) datasets using random forests. Yet, no prior work has investigated the impact of feature selection techniques when interpreting defect models. Thus, in this paper, we investigate 11 commonly-used feature selection techniques and our own contribution AutoSpearman along five dimensions: (1) the consistency of the produced subsets of metrics; (2) the correlation of the produced subsets of metrics; (3) the performance; (4) the computational cost; and (5) the impact on the interpretation.

The conclusions of prior work often rely on one feature selection technique (Arisholm *et al.* 2010; D'Ambros *et al.* 2012; Elish and Elish 2008; Kaur and Malhotra 2008; Menzies *et al.* 2007) which may pose a threat to the construct validity, i.e., conclusions may not hold true if another feature selection technique is applied. Nevertheless, little is known about whether feature selection techniques produce

the same subset of metrics among training samples and among feature selection techniques. We therefore formulate the following research question:

(RQ1) Do feature selection techniques consistently produce the same subset of metrics when applied on the same training sample?

In practice, *feature selection techniques should only be applied on training samples* because of the unavailability of defect labels in testing samples. Since training samples are often randomly generated from model validation techniques (e.g., out-of-sample bootstrap validation or 10-folds cross-validation), feature selection techniques may produce different subsets of metrics for each training sample. Different subsets of metrics among training samples may pose a critical threat to the validity when analysing and identifying the most important metrics. For example, prior work often applies a post-hoc multiple comparison test (e.g., a Scott-Knott test) on the distributions of importance scores to identify statistically distinct ranks of the most important metrics (Jiarpakdee *et al.* 2018a; Tantithamthavorn *et al.* 2019a; Tian *et al.* 2015). Nevertheless, little is known about whether feature selection techniques produce the same subset of metrics among training samples. We therefore formulate the following research question:

(RQ2) Do feature selection techniques consistently produce the same subset of metrics when applied across different training samples?

Some feature selection techniques involve randomisation in the process of sampling data for validation to produce the best subset of metrics. Such randomisation may lead a feature selection technique to produce different subsets of metrics when the technique is applied repeatedly. Nevertheless, little is known about whether feature selection techniques produce the same subset of metrics when they are applied repeatedly. We therefore formulate the following research question:

(RQ3) Do feature selection techniques which make use of random seeds consistently produce the same subset of metrics when they are applied repeatedly?

In theory, feature selection techniques should produce the same subset of metrics regardless of the ordering of metrics in a model specification. For example, given a model specification of $y \sim x_1 + x_2$, which has an order of x_1 and x_2. When reordering the model specification (e.g., change from $y \sim x_1 + x_2$ to $y \sim x_2 + x_1$), feature selection techniques may produce the different subset of metrics. However, in practice, some feature selection techniques consider whether to include (or exclude) a metric in the output subset of metrics based on the ordering of metrics in a model specification. Such process may lead a feature selection technique to produce different subsets of metrics, if different model specifications are applied. Nevertheless, little is known about whether feature selection techniques produce the same subset of metrics across different model specifications (when reordering the model specification of a defect model). We therefore formulate the following research question:

(RQ4) Do feature selection techniques which make use of model specifications consistently produce the same subset of metrics when reordering the model specification of a defect model (e.g., from $y \sim x_1 + x_2$ to $y \sim x_2 + x_1$)?

The conclusions of prior defect studies rely on the usage of built-in interpretation techniques of classification techniques (e.g., ANOVA for logistic regression, and Breiman's Variable Importance for random forest). However, recent work points out that such interpretation techniques are sensitive to correlated metrics (Berry 1993; Jiarpakdee *et al.* 2018a; Strobl *et al.* 2008; Tantithamthavorn *et al.* 2016b; Zhang *et al.* 2017). For example, Jiarpakdee *et al.* (2018a) show that the interpretation of ANOVA Type-I can be altered by simply rearranging the model specification (e.g., from $y \sim m_1 + m_2$ to $y \sim m_2 + m_1$ if m_1 and m_2 are correlated). Despite posing a threat to the validity of previous work's conclusion, little is known about whether feature selection techniques mitigate correlated metrics. We therefore formulate the following research question:

(RQ5) Do feature selection techniques mitigate correlated metrics?

Prior research effort has shown the benefits of applying feature selection techniques to defect prediction models (Ghotra *et al.* 2017; Lu *et al.* 2014; Xu *et al.* 2016). For example, Ghotra *et al.* (2017), Lu *et al.* (2014), and Xu *et al.* (2016) investigate the impact of feature selection techniques on the performance of defect models. Nevertheless, their findings on the best feature selection techniques (i.e., lead to the model with the highest model performance) are not always consistent. Prior studies also often use different defect datasets. While our goal is model interpretation, practitioners may question the interpretation of inaccurate defect models. We set out to investigate the impact of feature selection techniques on the performance of defect models, particularly, on defect datasets from which we can accurately derive interpretations through the following research question:

(RQ6) What is the impact of feature selection techniques on the performance of defect models?

Due to the variation in metric selection, feature selection techniques may incur different computational costs. For example, filter-based feature selection techniques search for the best subset of metrics regardless of model construction. Wrapper-based feature selection techniques rely on classification techniques to assess each subset of metrics and find the best subset of metrics. Thus, we set out to investigate the computational cost of feature selection techniques through the following research question:

(RQ7) What is the computational cost of applying feature selection techniques?

Literature has suggested a variety of correlation threshold values to indicate strong correlations among metrics. For example, Kraemer *et al.* (2003) suggested the use of Spearman correlation of 0.7 to indicate strong correlations between metrics. Similarly, Hinkle *et al.* (2003) suggested that the Spearman correlation of 0.7–0.9 indicates strong correlations between metrics. Mason and Perreault Jr (1991), Fox and Monette (1992), and Hair *et al.* (2006) suggested the use of VIF threshold values of 3, 5, and 10 to indicate strong correlations between metrics, respectively. Such contradictory suggestions about the correlation threshold may produce different subsets of metrics and interpretation of defect models. Particularly, a stricter correlation threshold value identifies a higher number of correlated metrics and produces a smaller subset of metrics. On the other hand, a more relaxed correlation threshold value identifies a less number of correlated metrics and

produces a larger subset of metrics. Thus, we set out to investigate the impact of correlation threshold values on the interpretation of defect models through the following research question:

(RQ8) Do correlation threshold values have an impact on the interpretation of defect models?

4 Experimental Setup

In this section, we describe our experimental setup with respect to the studied datasets.

Studied Datasets. In selecting the studied datasets, we identify three important criteria that need to be satisfied:

Criterion 1—Publicly-available defect datasets. Prior work raises concerns about the replicability of software engineering studies (Robles 2010). In order to foster future replication of our work, we focus on publicly-available defect datasets.

Criterion 2—Datasets that are reliable and of high quality. Defect models rely greatly on the quality of the datasets that are used to construct them. Shepperd *et al.* (2013) raise concerns related to data quality in the NASA datasets. Furthermore, Petrić *et al.* (2016) show that problematic data remain in the cleaned NASA datasets. Thus, the quality of the NASA datasets is questionable. To ensure that the studied datasets are reliable and of high quality, we exclude the NASA datasets from our study.

Criterion 3—Datasets from which we can accurately derive interpretations. Analysts would only consider models that fit the data well (i.e., AUC > 0.7) and are stable (i.e., EPV > 10) (Tantithamthavorn *et al.* 2017). Hence, we only focus on datasets that produce such accurate and stable models. To identify datasets that produce accurate models, we generate 100 sets of training and testing samples using the out-of-sample bootstrap validation technique. While we use training samples to construct defect models (i.e., logistic regression and random forests), we use testing samples to evaluate such models using the AUC measure. However, classification techniques that are constructed on imbalanced data are often in favour of the majority class (Tantithamthavorn *et al.* 2019a). For example, if defective modules are the majority class, defect models are likely to produce overly optimistic performance estimates. Thus, we investigate the defective ratio (i.e., the proportion of defective modules in a dataset) of the 101 publicly-available defect datasets. Figure 1 shows the distribution of defective ratio of the 101 publicly-available defect datasets. We find that the values of defective ratio range from 0.02 to 0.99 with a median value of 0.2, suggesting that publicly-available defect datasets are often imbalanced. As suggested by Agrawal and Menzies (2018), we apply the SMOTE imbalance technique (Chawla *et al.* 2002) to all of the generated 100 sets of training samples prior to constructing defect models. We use the implementation of the `SMOTE` function as provided by the `DMwR` R package (Torgo and Torgo 2015). We compute the average AUC estimation of all models and exclude datasets that produce models with the average AUC estimation of below 0.7. To identify datasets that produce stable models, we compute the EPV measure for each dataset. EPV (Event-Per-Variables) is a measure of the risk of overfitting.

EPV is calculated as a ratio of the number of occurrences of the least frequently occurring class of the dependent variable (i.e., the number of defective modules) to the number of software metrics that are used to train the model. We then exclude datasets that have the EPV values of below 10 since models that are constructed using such datasets have a high risk of overfitting (Tantithamthavorn *et al.* 2017).

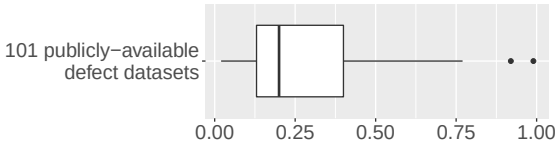


Fig. 1: The distributions of the defective ratio of the 101 publicly-available defect datasets.

To satisfy criterion 1, similar to prior work (Tantithamthavorn *et al.* 2016a), we begin our study using a collection of the 101 publicly-available defect datasets that are collected from 5 different corpora, i.e., 76 datasets from the Tera-PROMISE Repository (Menzies *et al.* 2012), 12 clean NASA datasets as provided by Shepperd *et al.* (2013), 5 datasets as provided by Kim *et al.* (2011), 5 datasets as provided by D'Ambros *et al.* (2010, 2012), and 3 datasets as provided by Zimmermann *et al.* (2007). To satisfy criterion 2, we exclude 12 datasets for which their data quality is questionable. Finally, to satisfy criterion 3, we exclude 75 datasets which have an EPV value below 10 and produce models with an AUC value below 0.7 after applying the SMOTE technique. Hence, we focus on 14 defect datasets from 4 corpora. In particular, 6 datasets (i.e., *Apache POI 2.5*, *Apache POI 3.0*, *Apache Xalan 2.6*, *Apache Xerces 1.4*, *Proprietary 1*, and *Proprietary 4*) as provided by Jureczko and Madeyski (2010); 3 datasets (i.e., *Apache Lucene 2.4*, *Eclipse JDT*, and *Eclipse Mylyn*) as provided by D'Ambros *et al.* (2010, 2012); 3 datasets (i.e., *Eclipse Platform 2*, *Eclipse Platform 2.1*, and *Eclipse Platform 3*) as provided by Zimmermann *et al.* (2007); and 2 datasets (i.e., *Eclipse Debug* and *Eclipse SWT 3.4*) as provided by Kim *et al.* (2011).

Table 4 shows a statistical summary of the 14 studied datasets.

5 Experimental Results

In this section, we present the approach, results, and illustrative examples with respect to the eight research questions.

(RQ1) Do feature selection techniques consistently produce the same subset of metrics when applied on the same training sample?

Approach. To address RQ1, we investigate the consistency of subsets of metrics that are produced by feature selection techniques. We first generate training samples. Then, we apply feature selection techniques on each training sample. Finally,

Table 4: A statistical summary of the studied datasets.

Project	Dataset	Modules	Metrics	Defective Ratio	EPV	$\mathbf{AUC}_{\mathrm{LR}}$	$\mathbf{AUC}_{\mathrm{RF}}$
Apache	Lucene 2.4	340	20	60	10	0.79	0.85
	POI 2.5	385	20	64	12	0.79	0.85
	POI 3.0	442	20	64	14	0.79	0.85
	Xalan 2.6	885	20	46	21	0.79	0.85
	Xerces 1.4	588	20	74	22	0.79	0.85
Eclipse	Debug 3.4	1,065	17	25	15	0.72	0.81
	JDT	997	15	21	14	0.81	0.82
	Mylyn	1,862	15	13	16	0.78	0.74
	Platform 2	6,729	32	14	30	0.82	0.84
	Platform 2.1	7,888	32	11	27	0.77	0.78
	Platform 3	10,593	32	15	49	0.79	0.81
	SWT 3.4	1,485	17	44	38	0.87	0.97
Proprietary	Prop 1	18,471	20	15	137	0.75	0.79
	Prop 4	8,718	20	10	42	0.74	0.72

we analyse the consistency of subsets of metrics that are produced by the studied feature selection techniques. We describe each step below.

(Step 1) Generate training samples. To generate training samples, we use the out-of-sample bootstrap validation technique that (1) leverages aspects of statistical inference (Efron and Tibshirani 1993; Friedman *et al.* 2001; Harrell Jr 2015); and (2) produces the least bias and variance of performance estimates for defect prediction (Tantithamthavorn *et al.* 2017). We randomly generate a bootstrap sample of size N with replacement from an original dataset, where N is the size of the original dataset. On average, 36.8% of the original dataset will not be selected, since a bootstrap sample is selected with replacement (Efron and Tibshirani 1993). We repeat the out-of-sample bootstrap process 100 times.

(Step 2) Apply feature selection techniques. We only apply feature selection techniques on *a training sample*, instead of the whole dataset in order to avoid producing optimistically biased performance estimates and interpretation (Li *et al.* 2017). The subsets of metrics that are produced by each studied feature selection technique for all studied datasets are available in the online appendix (Jiarpakdee *et al.* 2018c).

(Step 3) Analyse the consistency of subsets of metrics among different feature selection techniques. Ideally, feature selection techniques should consistently produce the same subset of metrics. We measure the consistency as a percentage of the unique metrics that consistently appeared among all of the studied feature selection techniques compared to all of the unique metrics for all studied feature selection techniques. We write the following equation to describe the consistency of subsets of metrics among different feature selection techniques (C_{FS}).

$$C_{\mathrm{FS}} = \frac{|S_{\mathrm{FS}_1\mathrm{TS}_j} \cap S_{\mathrm{FS}_2\mathrm{TS}_j} ... \cap S_{\mathrm{FS}_9\mathrm{TS}_j}|}{|S_{\mathrm{FS}_1\mathrm{TS}_j} \cup S_{\mathrm{FS}_2\mathrm{TS}_j} ... \cup S_{\mathrm{FS}_9\mathrm{TS}_j}|} \quad (1)$$

where $S_{\mathrm{FS}_i\mathrm{TS}_j}$ is a subset of metrics that is produced by a feature selection technique (FS_i) when applied on a training sample (TS_j). We present the consistency percentage using boxplots in Figure 2.

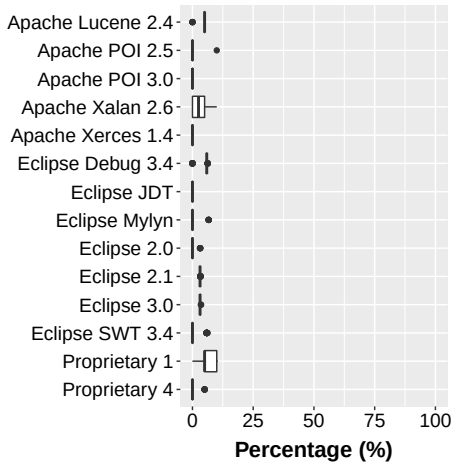


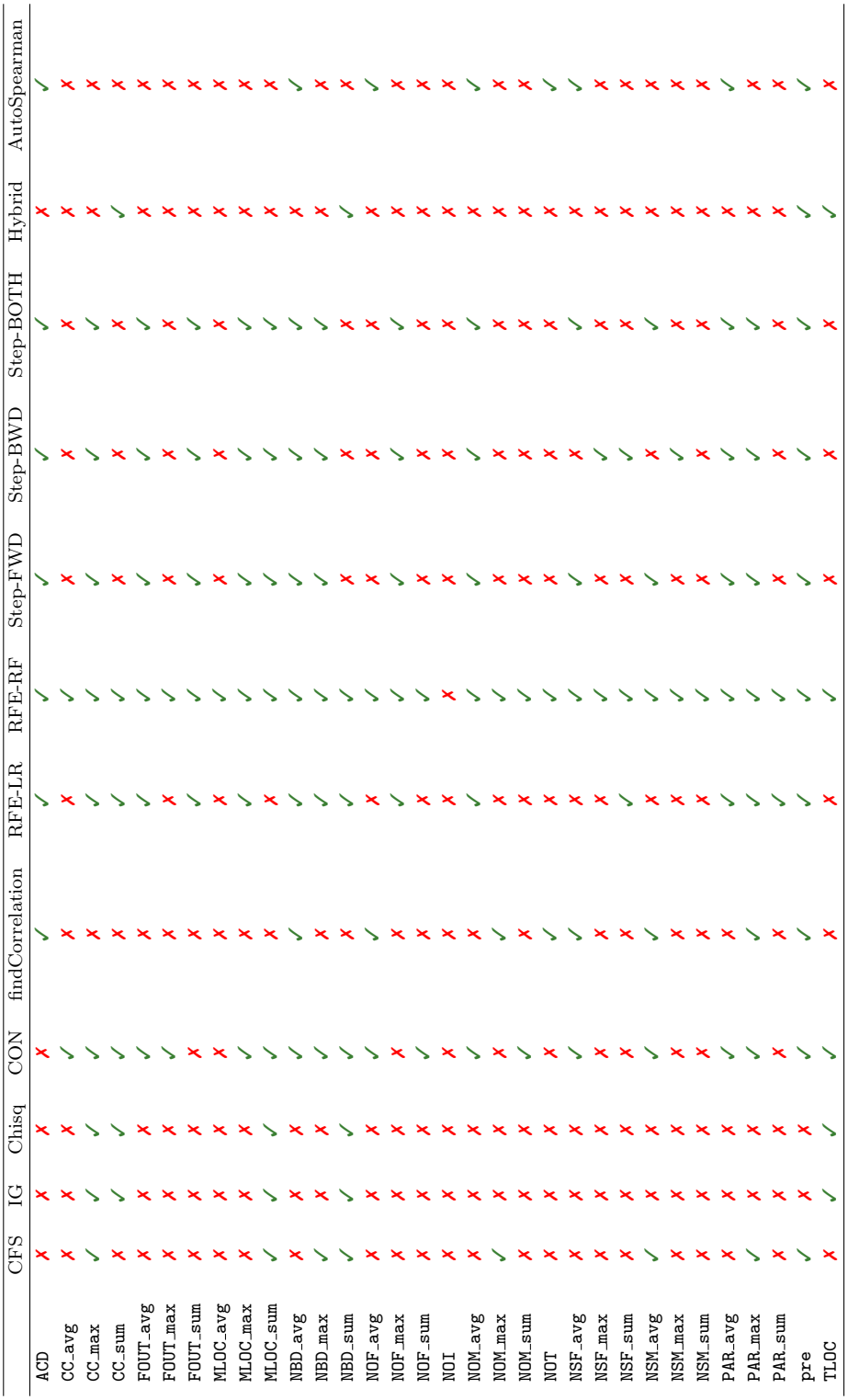
Fig. 2: The percentage of metrics that are consistently selected when applying feature selection techniques to the same training sample for all defect datasets.

Results. **When applying the studied feature selection techniques to the same training sample, only 0-6% of the metrics are consistently selected.** Figure 2 shows the percentage of metrics that are consistently selected when applying feature selection techniques on each training sample of each defect dataset. We observe that, at the median, only 0-6% of the metrics are consistently selected among the studied feature selection techniques. In other words, as many as 94-100% of the metrics are inconsistently selected among the studied feature selection techniques, suggesting that feature selection techniques select different metrics even if they are applied on the same training sample. We provide an illustrative example below.

Illustrative Example. We select the Eclipse Platform 2 dataset as the subject of this example, since it is widely used in a large number of defect prediction studies (Bird *et al.* 2009; Moser *et al.* 2008; Zimmermann *et al.* 2009). We first draw a bootstrap training sample (*cf.* Step 1 of RQ1) and apply all studied feature selection techniques (*cf.* Step 2 of RQ1). Unfortunately, we observe that none of the metrics is consistently selected across all studied feature selection techniques. The most commonly-selected metric is `pre`, which is selected by the 10 studied feature selection techniques except for information gain and chi-squared-based. This observation raises concerns related to the external and the conclusion validity of prior work that their conclusions could be altered when another feature selection technique is applied. We report the produced subsets of metrics for this illustrative example in Table 5.

Given the same training samples from the same defect datasets, different studied feature selection techniques *produce inconsistent subsets of metrics. As many as 94-100% of the metrics are inconsistently selected among the studied feature selection techniques, suggesting that the conclusions of prior work could be altered when another feature selection technique is applied.*

Table 5: The subsets of metrics that are produced by all of the twelve studied feature selection techniques on a training sample from the Eclipse Platform 2 dataset. While a ✓ mark indicates that a metric is selected by a feature selection technique, a ✗ mark indicates that a metric is not selected by a feature selection technique.



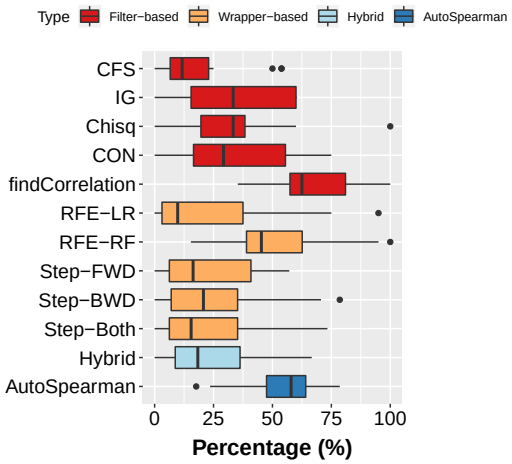


Fig. 3: The percentage of metrics that are consistently selected when applying feature selection techniques to different training samples from the same dataset.

(RQ2) Do feature selection techniques consistently produce the same subset of metrics when applied across different training samples?

Approach. To address RQ2, we investigate the consistency of subsets of metrics across different training samples. Similar to RQ1, we start from the subsets of metrics that are produced by all studied feature selection techniques for each training sample of each studied dataset from RQ1 (*cf.* Step 2 of RQ1). Ideally, each feature selection technique should produce the same subset of metrics for all of the 100 training samples. We compute the consistency as a percentage of the unique metrics that consistently appeared among all of the 100 training samples compared to all of the unique metrics for all training samples. We write the following equation to describe the consistency of subsets of metrics across different training samples (C_{TS}).

$$C_{\text{TS}} = \frac{|S_{\text{FS}_i\text{TS}_1} \cap S_{\text{FS}_i\text{TS}_2} ... \cap S_{\text{FS}_i\text{TS}_{100}}|}{|S_{\text{FS}_i\text{TS}_1} \cup S_{\text{FS}_i\text{TS}_2} ... \cup S_{\text{FS}_i\text{TS}_{100}}|} \quad (2)$$

where $S_{\text{FS}_i\text{TS}_j}$ is a subset of metrics that is produced by a feature selection technique (FS_i) when applied on a training sample (TS_j). Finally, we present the consistency percentage using boxplots in Figure 3.

Results. **Surprisingly, when applying the studied feature selection techniques to different training samples from the same dataset, 10-63% of the metrics are consistently selected.** Figure 3 shows the percentage of metrics that are consistently selected when applying feature selection techniques to different training samples. We find that, at the median, 12%, 33%, 33%, 29%, 63%, 10%, 45%, 16%, 21%, 15%, 18%, and 58% of the metrics are consistently selected when applying CFS, IG, Chisq, CON, findCorrelation, RFE-LR, RFE-RF, Step-FWD, Step-BWD, Step-BOTH, Hybrid, and AutoSpearman to different training samples, respectively. In other words, the selected metrics are 37-90% inconsis-

Table 6: The subsets of metrics that are produced by the correlation-based feature selection technique (CFS) when applied on different training samples. While the green texts represent metrics that are consistently selected, the red texts represent inconsistently selected metrics.

Training Sample	The Subset of Metrics that is produced by CFS
Sample 1 (TS_1)	pre, MLOC_sum, NBD_sum, PAR_max, CC_max, NBD_max, NOM_max, NSM_avg
Sample 2 (TS_2)	pre, MLOC_sum, NBD_sum, PAR_max, CC_max, FOUT_sum, NBD_avg, NSM_max, PAR_sum

tent when applying the studied feature selection techniques to different training samples from the same dataset.

Although the training samples are drawn from the same original dataset, none of the commonly-used feature selection techniques consistently produce the same subset of metrics. Figure 4 shows a Venn diagram of the number of overlapping unique instances between two bootstrap samples. According to the figure, we find that the inconsistency of subsets of metrics among training samples has to do with the differences in the characteristics of data among such training samples. In this example, only 46% of the unique samples are overlapping, while 54% of them are different. Thus, different training samples may have different correlation among metrics, leading the correlation-based feature selection technique to produce different subsets of metrics that share the strongest relationship with the outcome while having a low correlation among themselves for such different training samples.

The inconsistency of subsets of metrics among training samples suggests that the randomisation of training samples could produce defect models that are constructed from different subsets of metrics even if the training samples are drawn from the same dataset. Below, we provide a detailed illustrative example to demonstrate such inconsistency of subsets of metrics in practice.

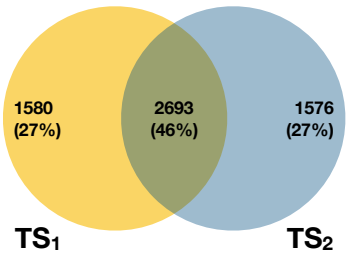


Fig. 4: A Venn diagram of the number of overlapping unique instances between two bootstrap samples.

Illustrative Example. Similar to prior illustrative examples in RQ1, we use the Eclipse Platform 2 dataset and the correlation-based feature selection technique

(CFS) for this illustrative example. First, we draw two bootstrap training samples (*cf.* Step 1 of RQ1) and apply CFS on these samples. Table 6 shows the subsets of metrics that are produced by the correlation-based feature selection techniques when applied on the two bootstrap training samples. We observe that only 5 of 12 metrics are consistently selected. This inconsistency in subsets of metrics restricts an application of post-hoc multiple comparison analyses (e.g., a Scott-Knott test) to identify the most important metrics when interpreting defect models since post-hoc multiple comparison analyses require balanced and completed data (i.e., all of the studied metrics must have the same number of instances without any missing values). Thus, when the importance score of a metric is not available due to the metric not being selected by a feature selection technique, it is not feasible to apply multiple comparison analyses.

Given different training samples from the same defect dataset*, the studied feature selection techniques produce inconsistent subsets of metrics. We find that 37-90% of the selected metrics are inconsistent, indicating that models that are trained on different training samples produce different ranking of the most important metrics. Such inconsistent subsets of metrics restrict an application of post-hoc multiple comparison analyses (e.g., a Scott-Knott test) to identify the most important metrics when interpreting defect models.*

(RQ3) Do feature selection techniques which make use of random seeds consistently produce the same subset of metrics when they are applied repeatedly?

Approach. To address RQ3, we investigate the consistency of subsets of metrics when feature selection techniques are applied repeatedly. Unlike Step 1 of RQ1 and RQ2, for each studied defect dataset, we use only one bootstrap training sample. To foster future replication studies, we set random seeds prior to applying feature selection techniques. Thus, we apply feature selection techniques on one training sample for all studied defect datasets with different random seeds. In RQ3, we focus on feature selection techniques which make use of random seeds and thus perform the analysis only for Recursive Feature Elimination. Finally, we analyse the consistency of subsets of metrics that are produced by feature selection techniques. We describe each step below.

(Step 1) Apply feature selection techniques on the training sample with different random seeds. Since the goal of this research question is to measure the consistency across different random seeds, we use 100 random seeds that range from 1 to 100 prior to applying feature selection techniques. We then apply feature selection techniques on the training sample and produce 100 subsets of metrics for each technique and each defect dataset.

(Step 2) Analyse the consistency of subsets of metrics across different random seeds. We start from the 100 subsets of metrics that are produced in Step 1 for each technique and each defect dataset. Ideally, each technique should produce the same subset of metrics for all 100 random seeds. We compute the consistency as a percentage of the unique metrics that consistently appeared among all of the 100 random seeds compared to all of the unique metrics for all random seeds. We write the following equation to describe the consistency of subsets of metrics when

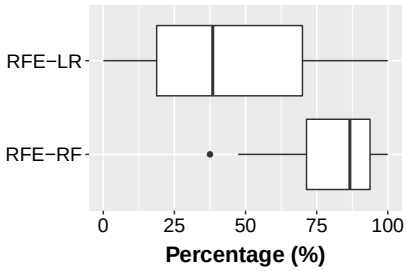


Fig. 5: The percentage of metrics that are consistently selected when applying feature selection techniques with different random seeds.

a feature selection technique is applied repeatedly (C_{RS}).

$$C_{\text{RS}} = \frac{|S_{\text{FS}_i\text{RS}_1\text{TS}} \cap S_{\text{FS}_i\text{RS}_2\text{TS}} ... \cap S_{\text{FS}_i\text{RS}_{100}\text{TS}}|}{|S_{\text{FS}_i\text{RS}_1\text{TS}} \cup S_{\text{FS}_i\text{RS}_2\text{TS}} ... \cup S_{\text{FS}_i\text{RS}_{100}\text{TS}}|} \tag{3}$$

, where $S_{\text{FS}_i\text{RS}_j\text{TS}}$ is a subset of metrics that is produced by a feature selection technique (FS_i) when applied with a random seed (RS_j) on a training sample (TS)). We present the consistency percentage using boxplots in Figure 5.

Results. **Recursive Feature Elimination produces inconsistent subsets of metrics when applied repeatedly with different random seeds regardless of the studied classification techniques.** Figure 5 shows the percentage of metrics that are consistently selected when repeatedly applying feature selection techniques with different random seeds. We observe that, at the median, 32% and 89% of the metrics are consistently selected when repeatedly applying Recursive Feature Elimination with logistic regression and random forest (i.e., RFE-LR and RFE-RF, respectively) with different random seeds. In other words, the selected metrics are 68% and 11% inconsistent for RFE-LR and RFE-RF, respectively. Similar to RQ2, we observe a higher consistency of subsets of metrics when applying Recursive Feature Elimination with random forests compared to logistic regression. We suspect that such a higher consistency of subsets of metrics has to do with the process of constructing and using multiple trees for a random forests model. For example, unlike logistic regression that constructs only one model, random forest constructs multiple trees using out-of-bag samples generated from training data and uses the aggregated results of these trees to generate predictions. Thus, the construction of multiple trees and the use of aggregated results lead random forests to produce a higher consistency of subsets of metrics than logistic regression when applied repeatedly.

The inconsistency of subsets of metrics when applied repeatedly with different random seeds suggests that although these variants of Recursive Feature Elimination are applied on the same sample, simply altering a random seed can lead them to produce a different subset of metrics. We suspect that the inconsistency in subsets of metrics has to do with the randomisation in the process of sampling data for validation in Recursive Feature Elimination. We discuss and provide a detailed illustrative example below.

Illustrative Example. Similar to prior illustrative examples in RQ1 and RQ2, we use the Eclipse Platform 2 dataset as the subject of this example. First, we

Table 7: The subsets of metrics that are produced by the Recursive Feature Elimination technique with logistic regression (RFE-LR) when applied with different random seeds on the same training sample. While the green texts represent metrics that are consistently selected, the red texts represent inconsistently selected metrics.

Random Seed	The Subset of Metrics that is produced by RFE-LR
RS_1 (`set.seed(1)`)	ACD, CC_max, CC_sum, FOUT_avg, FOUT_sum, MLOC_max, NBD_avg, NBD_max, NBD_sum, NOF_max, NOM_avg, NSF_sum, PAR_avg, PAR_max, PAR_sum, pre
RS_2 (`set.seed(2)`)	ACD, CC_max, CC_sum, FOUT_avg, FOUT_sum, MLOC_max, NBD_avg, NBD_max, NBD_sum, NOF_max, NOM_avg, NSF_sum, PAR_avg, PAR_max, PAR_sum, pre, NOF_avg, NOF_sum, NSM_avg, NSF_max

draw a bootstrap training sample (unlike Step 1 of RQ1, we use only one training sample) and apply RFE-LR with two different random seeds, i.e., `set.seed(1)` and `set.seed(2)`. We find that RFE-LR selects 16 metrics when applied with one random seed, while selecting 20 metrics when applied with another random seed. Across these subsets of metrics, 16 metrics are consistently selected. Such inconsistency in subsets of metrics when repeatedly applying Recursive Feature Elimination technique raises concerns related to the reliability and the construct validity. We provide the subsets of metrics that are produced by RFE-LR when applied with two different random seeds for this illustrative example in Table 7.

Given the same training sample from the same dataset but different random seeds*, Recursive Feature Elimination produces inconsistent subsets of metrics. Such inconsistency amounts to 68% and 11% of the subsets of metrics for logistic regression and random forest, respectively. This finding indicates that even if Recursive Feature Elimination is applied on the same training sample, simply altering a random seed can lead the technique to produce a different subset of metrics.*

(RQ4) Do feature selection techniques which make use of model specifications consistently produce the same subset of metrics when reordering the model specification of a defect model (e.g., from $y \sim x_1 + x_2$ to $y \sim x_2 + x_1$)?

Approach. To address RQ4, we investigate the consistency of subsets of metrics across different model specifications. Similar to RQ3, for each studied defect dataset, we use only one bootstrap training sample. Then, we apply feature selection techniques on the training sample with different model specifications for all studied defect datasets. In RQ4, we focus on feature selection techniques which make use of model specifications and thus perform the analysis only for the 3 variants of Stepwise Regression. Finally, we analyse the consistency of subsets of

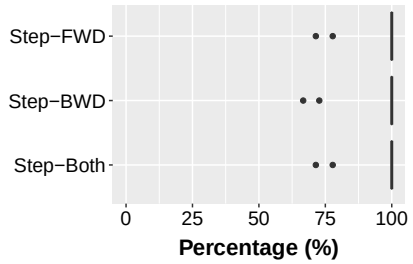


Fig. 6: The percentage of metrics that are consistently selected when applying feature selection techniques with different model specifications.

metrics that are produced by feature selection techniques. We describe each step below.

(Step 1) Apply feature selection techniques on the training sample with different model specifications. Since it is difficult to measure the consistency across all possible model specifications (e.g., 10 metrics would lead to 10! different model specifications), in this study, we generate 100 model specifications by randomly rearranging the ordering of metrics of each studied defect dataset differently for 100 times. For example, $y \sim x_1 + x_2 + ...$, $y \sim x_2 + x_1 + ...$, and $y \sim x_3 + x_1 + ...$ We then apply feature selection techniques on the training sample with 100 model specifications and produce 100 subsets of metrics for each technique and each defect dataset.

(Step 2) Analyse the consistency of subsets of metrics across different model specifications. We start from the subsets of metrics in Step 1 that are produced with 100 variations of model specification on the training sample for each feature selection technique of each studied dataset. Ideally, each feature selection technique should produce the same subset of metrics for all 100 variations of model specification regardless of the ordering of metrics, e.g., $\text{FS}(y \sim x_1 + x_2) = \text{FS}(y \sim x_2 + x_1)$. We compute the consistency as a percentage of the unique metrics that consistently appeared among all of the 100 variations of the model specification compared to all of the unique metrics for all model specifications. We write the following equation to describe the consistency of subsets of metrics across different model specifications (C_{MS}).

$$C_{\text{MS}} = \frac{|S_{\text{FS}_i\text{MS}_1\text{TS}} \cap S_{\text{FS}_i\text{MS}_2\text{TS}} ... \cap S_{\text{FS}_i\text{MS}_{100}\text{TS}}|}{|S_{\text{FS}_i\text{MS}_1\text{TS}} \cup S_{\text{FS}_i\text{MS}_2\text{TS}} ... \cup S_{\text{FS}_i\text{MS}_{100}\text{TS}}|} \tag{4}$$

, where $S_{\text{FS}_i\text{MS}_j\text{TS}}$ is a subset of metrics that is produced by a feature selection technique (FS_i) when applied with a model specification (MS_j) on a training sample (TS)). We present the consistency percentage using boxplots in Figure 6.

Results. **All of the 3 variants of Stepwise Regression produce inconsistent subsets of metrics across model specifications.** Figure 6 shows the percentage of metrics that are consistently selected when applying feature selection techniques with different model specifications. We observe an inconsistency of subsets of metrics that are produced by Stepwise Regression when reordering the model specification in the Eclipse SWT 3.4 dataset. We discuss and provide a detailed illustrative example below.

Table 8: The subsets of metrics that are produced by the forward direction Stepwise Regression (Step-FWD) when applied with different model specifications on the same training sample. While the green texts represent metrics that are consistently selected, the red texts represent inconsistently selected metrics.

Model Specification	The Subset of Metrics that is produced by Step-FWD
Specification 1 (MS_1)	CBO, change_times, DIT, IFANIN, NCM, NIM, NIV, RFC
Specification 2 (MS_2)	CBO, change_times, DIT, IFANIN, NCM, NIV, RFC, WMC

```
Step:  AIC=1150.72
isDefective ~ RFC + change_times + NCM + NIV + CBO + IFANIN +
    DIT

                    Df Deviance    AIC
+ NIM                1   1131.6 1149.6
+ WMC                1   1131.6 1149.6
+ LCOM               1   1132.1 1150.1
<none>                   1134.7 1150.7
+ NCV                1   1133.7 1151.7
+ minus              1   1134.0 1152.0
+ plus               1   1134.4 1152.4
+ NOC                1   1134.4 1152.4
+ LOC                1   1134.7 1152.7
+ MaximumComplexity  1   1134.7 1152.7
+ AverageComplexity  1   1134.7 1152.7
```

Fig. 7: The execution log of Stepwise Regression.

Illustrative Example. We select the Eclipse SWT 3.4 dataset, the problematic defect dataset, as the subject of this example. Similar to the prior illustrative example in RQ3, we draw only one bootstrap training sample and apply forward direction Stepwise Regression with two different model specifications. Table 8 shows the subsets of metrics that are produced by forward direction Stepwise Regression when applied with two different model specifications. We observe that, while 7 metrics are consistently selected, Stepwise Regression either selects `NIM` or `WMC` depending on which metric appears first in the ordering of metrics in a model specification. For example, given a model specification of `defect` $\sim$ `NIM` + `WMC` + ..., Stepwise Regression will select `NIM` rather than `WMC` since the metric appears first in the model specification. The execution log of Stepwise Regression shows that including either of these metrics equally improves the AIC value of a regression model. This finding leads us to conclude that, when including (or excluding) either of the two metrics can equally improve the AIC value of a regression model, Stepwise Regression includes (or excludes) the first metric that appears in the model specification. We further investigate the correlation of these problematic metrics and find that their Spearman correlation coefficient is as high as 0.87, suggesting that they are highly-correlated metrics. This finding raises concerns that correlated metrics may introduce the inconsistency of subsets of metrics when applying Stepwise Regression with different model specifications. We provide the subset of metrics for this illustrative example in Table 8 and the mentioned part of the execution log of Stepwise Regression in Figure 7.

Given the same training sample from the same defect dataset but different model specifications*, regardless of the search directions, Stepwise Regression is the only studied feature selection technique that produces inconsistent subsets of metrics, suggesting that Stepwise Regression is sensitive to the model specifications of defect models.*

(RQ5) Do feature selection techniques mitigate correlated metrics?

Approach. To identify correlated metrics, we apply correlation analyses on subsets of metrics that are produced by feature selection techniques. Similar to RQ1 and RQ2, we start from the subsets of metrics that are produced by all studied feature selection techniques for each training sample of each studied dataset from RQ1 (*cf.* Step 2 of RQ1). Then, we analyse the correlation among metrics for each subset of metrics that are produced by the studied feature selection techniques. In this paper, we focus on two types of correlation among metrics, i.e., collinearity and multicollinearity. Collinearity is a phenomenon in which one metric can be linearly predicted by another metric. On the other hand, multicollinearity is a phenomenon in which one metric can be linearly predicted by a combination of two or more metrics. We describe each step below.

(Step 1) Analyse collinearity. To analyse collinearity, we use a Spearman rank correlation test (ρ) to measure the correlation between metrics. We choose the Spearman test instead of other correlation tests (e.g., Pearson) since the Spearman test is resilient to non-normal distributions as commonly present in defect datasets. We use the interpretation of correlation coefficients ($|\rho|$) as provided by Kraemer *et al.* (2003), i.e., a Spearman correlation coefficient of above *0.7 is considered as a strong correlation.* Thus, two metrics which have their Spearman correlation coefficient of above 0.7 are considered correlated. We use the implementation of the `rcorr` function as provided by the `Hmisc` R package (Harrell Jr 2013).

(Step 2) Analyse multicollinearity. To analyse multicollinearity, we use the Variance Inflation Factor analysis (VIF) (Fox and Monette 1992). VIF determines how well a metric can be linearly predicted by a combination of other metrics through a construction of a regression model. A VIF score of a metric under examination is an R^2 goodness-of-fit of the model that is constructed by the other metrics to predict the metric under examination where a VIF score is $\frac{1}{1-R^2}$. We use a VIF threshold value of 5 to identify the presence of multicollinearity, as suggested by Fox (2015) and prior work (Bettenburg and Hassan 2010; Jiarpakdee *et al.* 2016, 2018a; McIntosh *et al.* 2014). Thus, metrics that have their VIF score above 5 are considered correlated. We use the implementation of the Variance Inflation Factor analysis using the `vif` function as provided by the `rms` R package (Harrell Jr 2017). Finally, we present the results using boxplots in Figure 8.

Results. **All of the commonly-used feature selection techniques do not mitigate correlated metrics except for AutoSpearman.** Figure 8 presents the percentage of subsets of metrics that contain correlated metrics for each studied feature selection technique. The studied feature selection techniques produce, at the median, 100% of subsets of metrics with collinearity except for findCorrelation (0%) and AutoSpearman (0%). Furthermore, except for AutoSpearman, the studied feature selection techniques produce, at the median, 5-100% of subsets of metrics with multicollinearity.

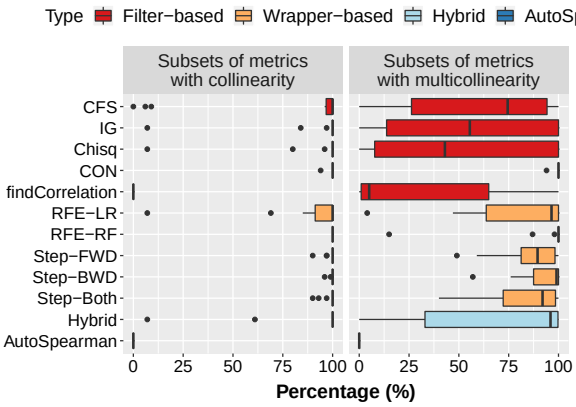


Fig. 8: The percentage of subsets of metrics that contain correlated metrics for each studied feature selection technique. The left boxplots present the percentage of subsets of metrics with collinearity, while the right boxplots present the percentage of subsets of metrics with multicollinearity.

Surprisingly, although CFS searches for the best subset of metrics that share the highest correlation with the outcome (e.g., defect-proneness) while having the lowest correlation among each other, our results show that correlated metrics are prevalent in subsets of metrics that are produced by CFS. We observe that CFS tends to focus on the correlation of each metric and the outcome more than the correlation between metrics. Thus, when CFS is applied, correlated metrics are often selected if these correlated metrics share a strong relationship with the outcome.

Furthermore, while findCorrelation can often mitigate collinearity, the technique cannot mitigate multicollinearity as its percentage of subsets of metrics with multicollinearity is as high as 100% in some studied defect datasets. We observe that findCorrelation only reduces pair-wise correlations among metrics using a correlation matrix (e.g., Spearman correlation), leading to the mitigation of collinearity. Unfortunately, the usage of pair-wise correlations among metrics of findCorrelation fails to detect and mitigate multicollinearity (i.e., a phenomenon in which one metric can be linearly predicted by two or more metrics), suggesting that more advanced techniques (e.g., Variance Inflation Factor analysis) should be applied. Below, we provide detailed illustrative examples.

Illustrative Example. Similar to prior illustrative examples in RQ1, RQ2, and RQ3, we select the Eclipse Platform 2 dataset as the subject of this example. We first draw a bootstrap training sample (*cf.* Step 1 of RQ1) and analyse the correlation among all software metrics, i.e., collinearity and multicollinearity. Then, we apply three feature selection techniques that consider the correlation among metrics when selecting metrics, i.e., a correlation-based feature selection technique (CFS—one of the most commonly-used feature selection technique in the defect prediction domain), findCorrelation, and AutoSpearman. Finally, we analyse the correlation among the metrics in the subsets of metrics that are produced by CFS and findCorrelation.

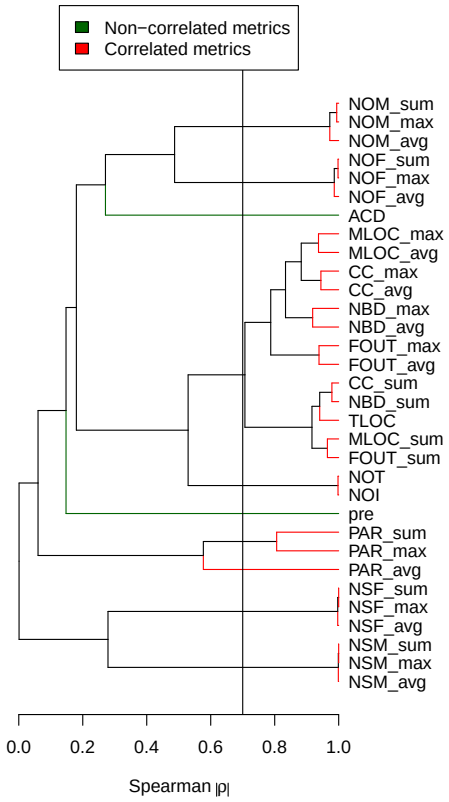
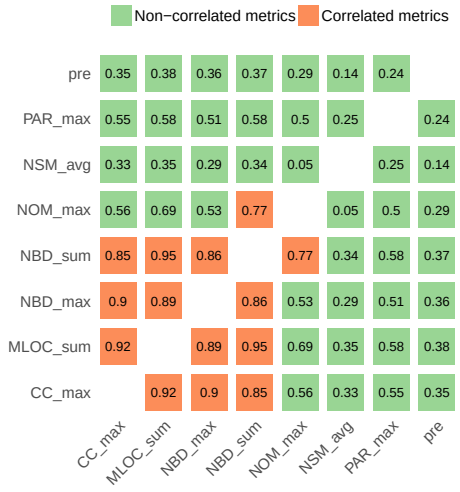
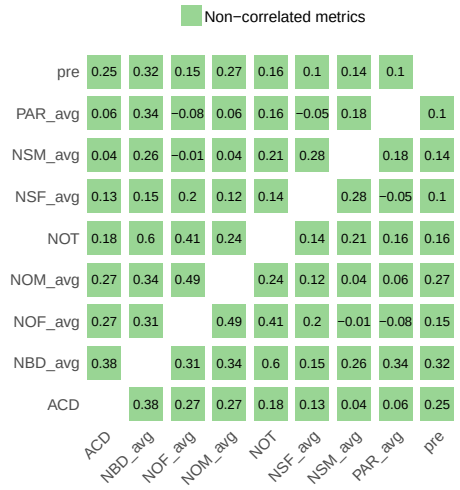


Fig. 9: The hierarchical cluster view of the correlation analysis of all metrics in the Eclipse Platform 2 dataset. Correlated metrics that can be linearly predicted by another metric and have their Spearman correlation coefficient above 0.7 are highlighted in red, while non-correlated metrics are highlighted in green.

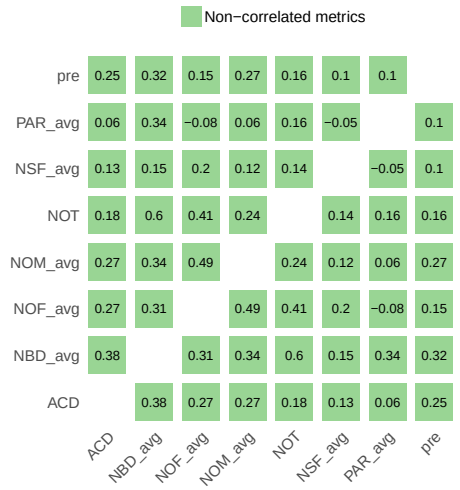
According to Figure 9, among the 32 metrics in the Eclipse Platform 2 dataset, we find that 30 metrics are correlated with a Spearman correlation coefficient above 0.7. We find that CFS selects 8 metrics that share the highest correlation with defect-proneness while having the lowest correlation among each other; findCorrelation selects 9 metrics through the reduction of pair-wise Spearman correlations among metrics; and AutoSpearman selects 8 metrics based on the results of correlation analyses. Unfortunately, the results of correlation analyses show that CFS and findCorrelation cannot mitigate correlated metrics. According to Figure 10a, as many as 5 of the 8 metrics that are selected by CFS can be linearly predicted by another metric and have their Spearman correlation coefficient above 0.7. Furthermore, according to Table 9, 3 of the 8 metrics that are selected by CFS and 2 of 9 the metrics that are selected by findCorrelation can be linearly predicted by a combination of other metrics and have their VIF score above 5. These findings suggest that (1) CFS, while considering the correlation among metrics, does not



(a) CFS



(b) findCorrelation



(c) AutoSpearman

Fig. 10: The Spearman rank correlation test on the subsets of metrics that are produced by CFS, findCorrelation, and AutoSpearman, respectively. Correlated metrics that can be linearly predicted by another metric and have their Spearman correlation coefficient above 0.7 are highlighted in red, while non-correlated metrics are highlighted in green.

Table 9: The Variance Inflation Factor analysis of the subsets of metrics that are produced by the correlation-based feature selection technique (CFS), findCorrelation, and AutoSpearman, respectively. Correlated metrics that can be linearly predicted by a combination of other metrics and have their VIF score above 5 are highlighted in red.

CFS	
Metric	VIF score
NBD_sum	18.02
MLOC_sum	12.37
NOM_max	5.32
CC_max	3.24
NBD_max	2.18
PAR_max	1.40
pre	1.32
NSM_avg	1.17

findCorrelation	
Metric	VIF score
NSM_avg	7.24
NSF_avg	7.17
NBD_avg	2.10
NOT	1.84
pre	1.29
ACD	1.26
PAR_avg	1.15
NOM_avg	1.12
NOF_avg	1.11

AutoSpearman	
Metric	VIF score
NBD_avg	2.10
NOT	1.83
pre	1.29
ACD	1.26
PAR_avg	1.13
NOM_avg	1.12
NOF_avg	1.11
NSF_avg	1.02

mitigate correlated metrics; and (2) findCorrelation can mitigate collinearity but not multicollinearity.

On the other hand, accroding to Figure 10c and Table 9, AutoSpearman mitigates correlated metrics for both collinearity and multicollinearity. We observe that the collinearity analysis of AutoSpearman selects one representative metric from each group of correlated metrics. Furthermore, the multicollinearity analysis of AutoSpearman further mitigates correlated metrics that cannot be detected by collinearity analysis (i.e., the analysis of pair-wise correlation among metrics).

Given the same training sample from the same defect datasets, the subsets of metrics produced by all of the studied feature selection techniques (except for AutoSpearman) contain correlated metrics, i.e., collinearity and multicollinearity. *In other words, AutoSpearman is the only studied feature selection technique which was designed to mitigate both collinearity and multicollinearity, and as our experiments show, none of the other techniques achieves these goals. These findings suggest that the interpretation of defect models constructed using the subsets of metrics that are produced by the studied feature selection techniques (except for AutoSpearman) may be misleading.*

(RQ6) What is the impact of feature selection techniques on the performance of defect models?

Approach. To address RQ6, we analyse the performance of defect models that are constructed using the subsets of metrics that are produced by the twelve studied feature selection techniques, and a baseline (i.e., all metrics of a defect dataset). Since Figure 1 and Table 4 show that the defective ratio of the studied defect datasets is imbalanced and may lead to models that are often in favour of the majority class, we apply the SMOTE imbalance technique to each training sample for all studied defect datasets. We then apply feature selection techniques on all of the balanced training samples of the studied defect datasets to generate the subsets

of metrics (*cf.* Step 2 of RQ1). We construct defect models using these subsets of metrics and evaluate their performance. Consequently, we analyse the impact of each studied feature selection technique on model performance. We describe each step below.

(Step 1) Generate training samples. Similar to Step 1 of RQ1, we use the out-of-sample bootstrap validation technique to generate 100 sets of training and testing samples for each studied defect dataset.

(Step 2) Apply the SMOTE imbalance technique. For each training sample, we apply the SMOTE imbalance technique (Chawla *et al.* 2002) to mitigate the imbalanced nature of defect datasets. We use the implementation of the `SMOTE` function as provided by the `DMwR` R package (Torgo and Torgo 2015).

(Step 3) Apply feature selection techniques. Similar to Step 2 of RQ1, we only apply feature selection techniques on training samples to generate subsets of metrics of each studied feature selection techniques. The subsets of metrics that are produced by each studied feature selection technique for all studied datasets are available in the online appendix (Jiarpakdee *et al.* 2018c).

(Step 4) Construct defect models. For each training sample, we construct logistic regression and random forest models using subsets of metrics that are produced by the twelve studied feature selection techniques, and all metrics of a defect dataset. We use the implementation of logistic regression as provided by the `glm` function of the `stats` R package (Team and contributors worldwide 2017) with the default parameter setting. We use the implementation of random forest as provided by the `randomForest` function of the `randomForest` R package (Breiman *et al.* 2006) with the default `ntree` value of 100, since recent studies (Tantithamthavorn *et al.* 2016a, 2019b) show that the performance of random forest models is insensitive to the parameter setting. To ensure that the training and testing corpora share similar characteristics and avoid any potential impact on the interpretation of defect models, we do not re-balance nor do we re-sample the training data (Tantithamthavorn *et al.* 2019a).

(Step 5) Evaluate defect models. In our study, we evaluate defect models using three performance measures. First, we use the Area Under the receiver operator characteristic Curve (AUC) to measure the discriminatory power of our models, as suggested by recent research (Ghotra *et al.* 2015; Lessmann *et al.* 2008; Rahman and Devanbu 2013; Tantithamthavorn and Hassan 2018). The *AUC* is a threshold-independent performance measure that evaluates the ability of classifiers in discriminating between defective and clean modules. The values of AUC range between 0 (worst performance), 0.5 (no better than random guessing), and 1 (best performance) (Hanley and McNeil 1982). Second, we use the F-measure, i.e., a threshold-dependent measure. F-measure is a harmonic mean (i.e., $\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision}+\text{recall}}$) of precision ($\frac{\text{TP}}{\text{TP}+\text{FP}}$) and recall ($\frac{\text{TP}}{\text{TP}+\text{FN}}$). Similar to prior studies (Arisholm *et al.* 2010; Zimmermann *et al.* 2009), we use the default probability value of 0.5 as a threshold value for the confusion matrix, i.e., if a module has a predicted probability above 0.5, it is considered defective; otherwise, the module is considered clean. Third, we use the Matthews Correlation Coefficient (MCC) measure, i.e, a threshold-dependent measure, as suggested by prior studies (Matthews 1975; Shepperd *et al.* 2014). MCC is a balanced measure based on true and false positives and negatives that is computed using the following equation: $\frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}}$.

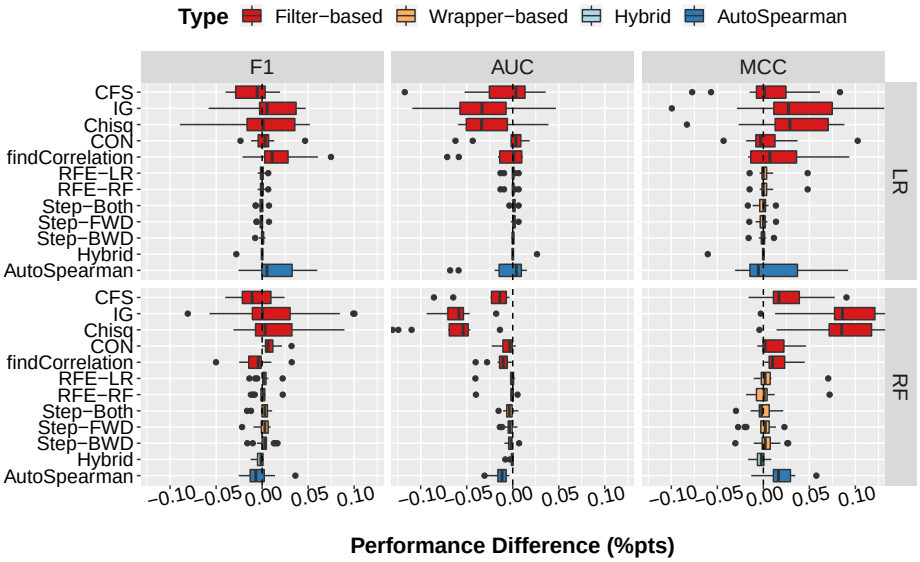


Fig. 11: The distributions of the performance difference (%pts) between defect models that are constructed using subsets of metrics that are produced by the twelve studied feature selection techniques and all metrics of a defect dataset, i.e., $P_{\text{FS}} - P_{\text{All}}$.

(Step 6) Analyse the impact on the model performance. We analyse the performance difference between defect models that are constructed using subsets of metrics that are produced by feature selection techniques (i.e., the twelve studied feature selection techniques) and all metrics of a defect dataset ($P_{\text{FS}} - P_{\text{All}}$). We also set out to investigate whether such performance difference is statistically significant using the Mann-Whitney U test and its extension, the Kruskal and Wallis H test. A p-value of the Mann-Whitney U test of below 0.05 suggests that the difference between two input distributions is statistically significant, while a p-value of the Kruskal and Wallis H test of below 0.05 suggest that the differences across multiple input distributions are statistically significant. Finally, in order to measure the effect size of the performance difference, we use Cliff's $|\delta|$ effect size to analyse the magnitude of the difference between these distributions. Cliff's $|\delta|$ effect size ranges from 0 to +1, where a zero value indicates that two distributions are identical. We use the interpretation of Cliff's $|\delta|$ estimates by Romano *et al.* (2006) which maps Cliff's $|\delta|$ to Cohen's significance levels as follows:

$$\begin{gathered}\text{Negligible: } |\delta| < 0.147 \\ \text{Small: } 0.147 \leq |\delta| < 0.33 \\ \text{Medium: } 0.33 \leq |\delta| < 0.474 \\ \text{Large: } 0.474 \leq |\delta|\end{gathered}$$

We present the results using boxplots in Figure 11.

Results. The studied feature selection techniques impact the performance of defect models by up to 9%pts. Figure 11 shows the performance

difference (%pts) between defect models that are constructed using subsets of metrics that are produced by the twelve studied feature selection techniques and all metrics of a defect dataset, i.e., $P_{\text{FS}} - P_{\text{All}}$. We make the three following observations: (1) filter-based feature selection techniques (except for CON) have the highest impact on the performance of defect models by up to 9%pts (at the median); (2) wrapper-based feature selection techniques have the least impact on the performance of defect models by up to 1%pt (at the median); and (3) AutoSpearman impacts the performance of defect models by up to 2%pts (at the median). The results of the Mann-Whitney U test show that most of the performance differences are not statistically significant with p-values above 0.05. The results of Kruskal-Wallis H test suggest that all of the performance differences are not statistically significant with p-values above 0.05. Finally, the results of the Cliff's $|\delta|$ effect size test show that such performance differences are negligible to small for the AUC, F-measure, and MCC measures (except for Information Gain and Chi-Squared-based). We suspect that the statistically insignificant impact on model performance of most of the studied feature selection techniques has to do with the low number of irrelevant metrics while most relevant metrics are highly correlated as shown in the results of RQ5. Also, the high number of highly correlated (RQ5) and relevant metrics (RQ6) may lead feature selection techniques to produce different optimal subsets of metrics as observed in RQ1. Although the inconsistency of the optimal subsets of metrics among feature selection techniques may not impact the performance of defect models, such inconsistency has a substantial impact on practitioners (e.g., developers, managers, and QA analysts) when developing software quality improvement plans. For example, given two metrics `CC_max` (code complexity) and `MLOC_sum` (method lines of code) that are highly correlated as shown in the illustrative example of RQ5, the use of these two metrics for training defect models may produce similar accuracy. On the other hand, the insights that are derived from defect models that are trained on correlated metrics may be inaccurate and unstable. In particular, QA plans generated from one feature selection technique may suggest minimizing the risks of having defects by reducing the code complexity, while QA plans generated by another feature selection technique may suggest minimizing the file size (i.e., lines of code). Such an inconsistency produced by widely-used automated feature selection techniques may distort the software quality improvement plans that are derived from defect models, leading to wasting time and QA resources.

The results of the Mann-Whitney U test show that most of the performance differences are not statistically significant (i.e., p-values > 0.05). The results of the Kruskal-Wallis H test indicate that all of the performance differences across feature selection techniques are not statistically significant with p-values above 0.05. Finally, the results of the Cliff's $|\delta|$ effect size test show that such performance differences are negligible to small for the AUC, F-measure, and MCC measures (except for Information Gain and Chi-Squared-based). These results from statistical and effect size tests suggest that none of the studied feature selection techniques has a greater impact on the performance of defect models than the others.

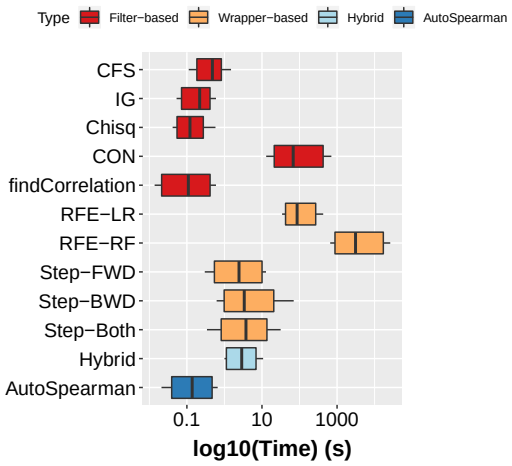


Fig. 12: The computational cost analysis of the twelve studied feature selection techniques.

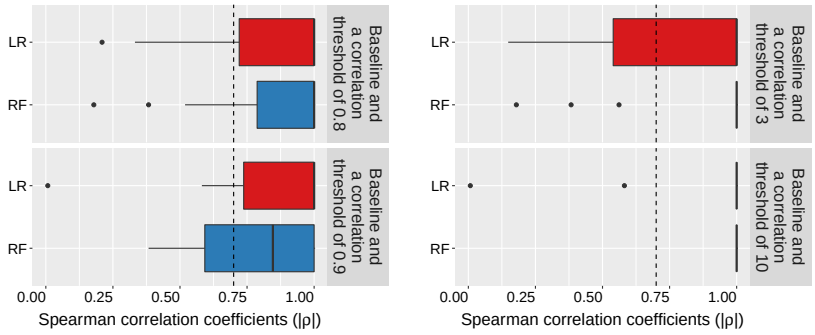
(RQ7) What is the computational cost of applying feature selection techniques?

Approach. To address RQ7, we analyse the computational cost of the twelve studied feature selection techniques. For each defect dataset, we use only one bootstrap training sample. Then, we measure the computational cost of applying the twelve studied feature selection techniques on a training sample for each defect dataset. We measure the computational cost using the `microbenchmark` function of the `microbenchmark` R package (Mersmann *et al.* 2018). Finally, we present the results using boxplots in Figure 12. Due to the enormous difference in the computation cost across the studied feature selection techniques, we report the computational cost (x-axis) using a `log10` scale.

Results. **The computational cost of wrapper-based feature selection techniques and the consistency-based feature selection technique is expensive.** Figure 12 shows the computational cost analysis of the studied feature selection techniques using a `log10` scale. We observe that, at the median, the computational cost of the studied feature selection techniques is 0.5s, 0.2s, 0.1s, 105s, 0.1s, 102s, 5021s (1 hour and 23 minutes), 4s, 3.8s, 6.8s, 3.4s, and 0.2s for CFS, IG, Chisq, CON, findCorrelation, RFE-LR, RFE-RF, Step-FWD, Step-BWD, Step-BOTH, Hybrid, and AutoSpearman, respectively. In particular, wrapper-based feature selection techniques and the consistency-based feature selection technique are expensive to compute, and can cost up to 7 hours and 15 minutes for RFE-RF to find the best subset of metrics from one training sample of the Eclipse Platform 3.0 defect dataset. Such expensive computation cost makes the application of wrapper-based feature selection techniques undesirable, particularly, when validating with model validation techniques that require several repetitions (e.g., bootstrap validation technique).

The computational cost of filter-based feature selection techniques and AutoSpearman is cheap, while such cost is expensive for wrapper-based feature selection techniques and the consistency-based feature selection technique. The computational cost of wrapper-based feature selection techniques is as high as 7 hours and 15 minutes for RFE-RF to find the best subset of metrics from one training sample of the Eclipse Platform 3.0 defect dataset. Such expensive computation cost makes the application of wrapper-based feature selection techniques undesirable, particularly, when validating with model validation techniques that require several repetitions (e.g., bootstrap validation technique).

(RQ8) Do correlation threshold values have an impact on the interpretation of defect models?



(a) Spearman correlation (b) VIF

Fig. 13: The distributions of Spearman correlation coefficients of the most important metrics produced by the baseline and correlation threshold values.

Approach. To address RQ8, we analyse the impact of correlation threshold values when producing subsets of metrics with AutoSpearman on the interpretation of defect models along 2 dimensions, i.e., consistency and correlation. AutoSpearman relies on two threshold values, i.e., the Spearman correlation threshold and the VIF threshold. While we use the Spearman correlation threshold value of 0.7 as a baseline as suggested by Kraemer *et al.* (2003) and the VIF threshold value of 5 as a baseline as suggested by Fox and Monette (1992) to indicate strong correlations between metrics, we perform sensitivity analyses for both threshold values. For the sensitivity analysis of the Spearman correlation threshold, we use the Spearman correlation threshold value of 0.7 and compare the interpretation produced by the Spearman correlation threshold values of 0.8 and 0.9, as suggested by Hinkle *et al.* (2003), with those produced by the baseline. Similarly, for the sensitivity analysis of the VIF threshold, we use the VIF threshold value of 5 and compare the interpretation produced by the VIF threshold values of 3 (Mason and Perreault Jr 1991) and 10 (Hair *et al.* 2006) with those produced by the baseline. Below, we

Table 10: The percentage of the studied defect datasets in which their most important metrics are consistent among correlation threshold values.

Classification Techniques	Model Interpretation Techniques	Spearman Correlation Threshold		VIF Threshold	
		0.8	0.9	3	10
Logistic Regression	ANOVA Type 2	86%	69%	100%	100%
Random Forests	Scaled Permutation Importance	86%	46%	92%	100%

explain how we generate the interpretation of defect models and how we analyse the impact of correlation threshold values on such interpretation.

To generate the interpretation of defect models, we adopted the defect modelling workflow used in prior studies (Jiarpakdee *et al.* 2018a). First, we mitigate correlated metrics using our proposed feature selection technique, AutoSpearman. Second, we generate training and testing samples using the out-of-sample bootstrap validation technique. Third, we use training samples to construct defect models (i.e., logistic regression and random forests). Fourth, we generate the importance score of metrics using the ANOVA Type-II analysis (Fox 2015) for logistic regression and the scaled permutation importance analysis (Breiman 2001) for random forests. Finally, we identify the importance ranking of metrics using the improved Scott-Knott Effect Size Difference (ESD) test (Tantithamthavorn 2017).

We analyse the consistency and correlation of the most important metrics among correlation threshold values. To do so, we compute the percentage of the studied datasets in which their most important metrics are consistent among correlation threshold values. We also compute the Spearman correlation of the most important metrics produced by different correlation threshold values.

Results. **Different correlation threshold values may produce different most important metrics.** Table 10 shows the percentage of the studied defect datasets in which their most important metrics are consistent among correlation threshold values. We find that 86% and 46-69% of the studied defect datasets have their most important metrics produced by the Spearman correlation threshold values of 0.8 and 0.9 consistent with the baseline (the Spearman correlation threshold value of 0.7), respectively. We also find that 92-100% and 100% of the studied defect datasets have their most important metrics produced by the VIF threshold values of 3 and 10 consistent with the baseline (the VIF threshold value of 5), respectively. In other words, 14-54% of these studied datasets produce different most important metrics among Spearman correlation threshold values, while such inconsistency among VIF threshold values are 0-8%.

The most important metrics produced by different correlation threshold values are highly correlated. Figure 13 shows the distributions of Spearman correlation coefficients of the most important metrics produced by the baseline and correlation threshold values. We find that, at the median, Spearman correlation coefficients of the most important metrics produced by the baseline and the Spearman correlation threshold values of 0.8 and 0.9 are 1 and 0.85-1 for both logistic regression and random forests. Similarly, at the median, Spearman correlation coefficients of the most important metrics produced by the baseline and the VIF threshold values of 3 and 10 are 1 for logistic regression and random

Table 11: The top-5 most important metrics according to the percentage of the studied defect datasets in which their most important metrics are consistent among correlation threshold values.

Rank	AutoSpearman(0.7)	AutoSpearman(0.9)
1	NOM_avg	NBD_sum
2	NBD_avg	NBD_avg
3	pre	NOM_avg
4	PAR_avg	FOUT_avg
5	NOF_avg	pre

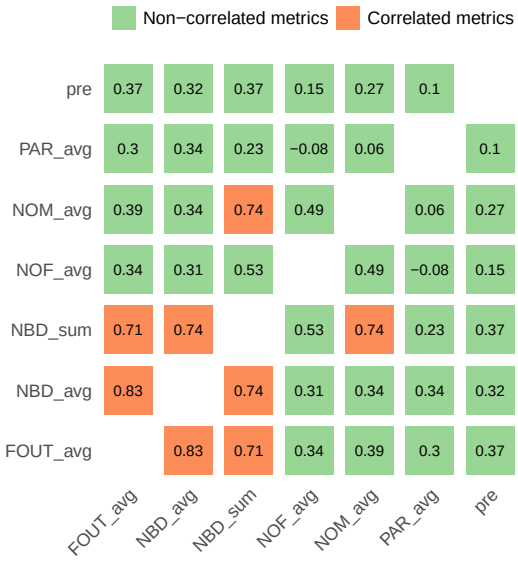


Fig. 14: The Spearman rank correlation test on the top-5 important metrics according to AutoSpearman using the correlation threshold values of 0.7 and 0.9. Correlated metrics that can be linearly predicted by another metric and have their Spearman correlation coefficient above 0.7 are highlighted in red, while non-correlated metrics are highlighted in green.

forests, respectively. These findings show that while the most important metrics are inconsistent among different correlation threshold values, such inconsistent most important metrics are highly correlated. The experimental results lead us to conclude that correlation threshold values do not impact the interpretation of defect models.

Illustrative Example. Similar to previous research questions, we use the Eclipse Platform 2 dataset as the subject of this example. Following the approach to generate the interpretation of defect models as explained above, we identify the importance rankings of metrics produced by AutoSpearman with the correlation threshold values of 0.7 (baseline) and 0.9 as shown in Table 11. According to Figure 14, we

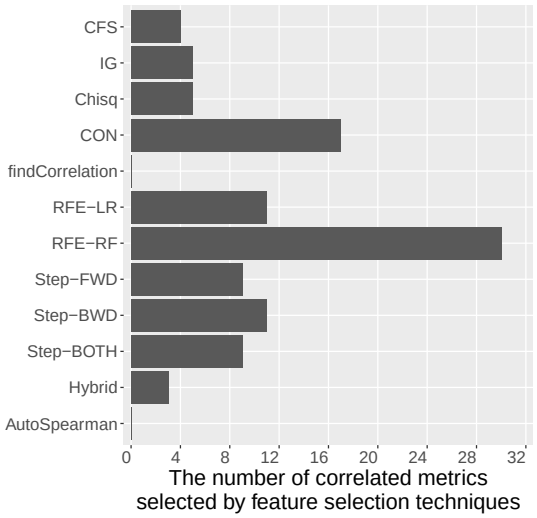


Fig. 15: The number of correlated metrics selected by each studied feature selection technique for an illustrative analysis of the Eclipse Platform 2 dataset.

find that `NOM_avg` (the most important metric according to AutoSpearman(0.7)) and `NBD_sum` (the most important metric according to AutoSpearman(0.9)) are highly correlated with the Spearman correlation of 0.73. This example further illustrates the little impact of correlation threshold values on the interpretation of defect models.

No. Although the most important metrics are inconsistent among correlation threshold values, such inconsistent most important metrics are highly correlated with the Spearman correlation of 0.85–1, suggesting that correlation threshold values do not impact the interpretation of defect models.

6 Discussion

In this section, we discuss the trends of correlated metrics that are selected by feature selection techniques.

Approach. To investigate the trends of correlated metrics that are selected by commonly-used feature selection techniques, similar to the illustrative example in RQ1, we first select the Eclipse Platform 2 dataset as the subject of this analysis. We draw a bootstrap training sample and apply all studied feature selection techniques to generate subsets of metrics for each studied technique. Then, we analyse the correlation among metrics for both collinearity and multicollinearity to find correlated metrics in each subset of metrics produced by the studied feature selection techniques (*cf.* Steps 1 and 2 of RQ5). Finally, we identify (1) the number of correlated metrics selected by each feature selection technique and (2) the number of feature selection techniques that select each correlated metric.

Results. **Most of the correlated metrics selected by commonly-used feature selection techniques are aggregated metrics produced by metric**

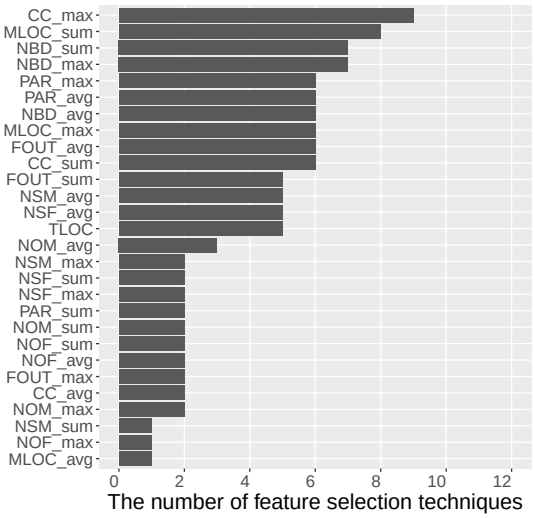


Fig. 16: The number of studied feature selection techniques that select each correlated metric for an illustrative analysis of the Eclipse Platform 2 dataset.

aggregation schemes. Figure 16 shows the number of studied feature selection techniques that select each correlated metric for an illustrative analysis of the Eclipse Platform 2 dataset. We find that correlated metrics are selected by 1–9 studied feature selection techniques. We observe that most of the correlated metrics are aggregated metrics—metrics that are extracted at the method level and are summarised to the file level using metric aggregation schemes, e.g., minimum, average, and maximum. This observation is consistent with that of Zhang *et al.* (2017) where metric aggregation schemes often produce correlated metrics.

7 Threats to Validity

We now discuss threats to the validity of our study.

Construct Validity. Plenty of prior work shows that the parameters of classification techniques have an impact on the performance of defect models (Fu *et al.* 2016; Koru and Liu 2005; Mende 2010; Mende and Koschke 2009; Tantithamthavorn *et al.* 2016a). While we use a default `ntree` value of 100 for random forest models, recent studies (Jiang *et al.* 2008; Tantithamthavorn *et al.* 2016a; Tosun and Bener 2009) show that the performance of random forest models is insensitive to this parameter setting. Thus, we believe that this threat is not a major limitation of our work.

The concept of non-correlated metrics in our paper relies on threshold values of correlation analyses (i.e., 0.7 for a Spearman rank correlation test and 5 for a Variance Inflation Factor analysis). To mitigate this threat, we perform an in-depth sensitivity analysis in RQ8 and find that correlation threshold values do not impact the interpretation of defect models.

Internal Validity. Prior studies raise concerns related to replicability (Robles 2010), data quality (Petrić *et al.* 2016; Shepperd *et al.* 2013), mislabelling (Tantithamthavorn *et al.* 2015; Yathish *et al.* 2019), and the risk of overfitting (Tantithamthavorn *et al.* 2017) in defect datasets. We conduct a highly-controlled experiment where we apply 3 dataset selection criteria to mitigate these concerns. Thus, our study focuses on 14 defect datasets from 4 corpora. In particular, 6 datasets (i.e., *Apache POI 2.5*, *Apache POI 3.0*, *Apache Xalan 2.6*, *Apache Xerces 1.4*, *Proprietary 1*, and *Proprietary 4*) as provided by Jureczko and Madeyski (2010); 3 datasets (i.e., *Apache Lucene 2.4*, *Eclipse JDT*, and *Eclipse Mylyn*) as provided by D'Ambros *et al.* (2010, 2012); 3 datasets (i.e., *Eclipse Platform 2*, *Eclipse Platform 2.1*, and *Eclipse Platform 3*) as provided by Zimmermann *et al.* (2007); and 2 datasets (i.e., *Eclipse Debug* and *Eclipse SWT 3.4*) as provided by Kim *et al.* (2011).

External Validity. We studied a limited number of feature selection techniques. Thus, our results may not generalise to other feature selection techniques. Nonetheless, other feature selection techniques can be explored in future work. We provide a detailed methodology for others who would like to re-examine our findings using unexplored feature selection techniques.

The studied defect datasets are part of several systems (e.g., Eclipse) that span both proprietary and open source domains. However, we studied a limited number of defect datasets. Thus, the results may not generalise to other datasets and domains. Replication studies are needed.

The dimensions (i.e., the number of metrics) of the studied defect datasets are relatively low when comparing to other domains. Thus, the findings may be altered in other contexts of high-dimensional datasets (e.g., bioinformatics or textual features). Future studies could revisit our study with datasets from other domains which may vary in dimensions and characteristics.

The conclusions of our case study rely on one defect prediction scenario (i.e., within-project defect models). However, there are a variety of defect prediction scenarios in the literature (e.g., cross-project defect prediction (Canfora *et al.* 2013; Zimmermann *et al.* 2009), just-in-time defect prediction (Kamei *et al.* 2013), heterogenous defect prediction (Nam *et al.* 2017)). Therefore, the conclusions may differ for other scenarios. Future research should revisit our study in other scenarios of defect models.

8 Conclusions

In this paper, we investigate 11 commonly-used feature selection techniques and our own contribution AutoSpearman along five dimensions: (1) the consistency of the produced subsets of metrics; (2) the correlation of the produced subsets of metrics; (3) the performance; (4) the computational cost; and (5) the impact on the interpretation. Through a case study of 14 publicly-available defect datasets of systems that span both proprietary and open source domains, we find that (1) 94-100% of the selected metrics are inconsistent among the studied techniques; (2) 37-90% of the selected metrics are inconsistent among training samples; (3) 0-68% of the selected metrics are inconsistent when the application of the feature selection techniques is repeated; and (4) 5-100% of the produced subsets of metrics contain correlated metrics—suggesting that the commonly-used automated

feature selection techniques are often unreliable. We also find that (5) although the most important metrics are inconsistent among correlation threshold values, such inconsistent most important metrics are highly-correlated with the Spearman correlation of 0.85–1.

This is the first paper to provide evidence that subsets of metrics produced by commonly-used feature selection techniques in the defect prediction domain are inconsistent in nature. Such inconsistent nature of these feature selection techniques (1) restricts an application of post-hoc multiple comparison analyses (e.g., a Scott-Knott test) to identify the most important metrics when interpreting defect models; and (2) has a negative impact on the interpretation of defect models due to the presence of correlated metrics. Thus, to mitigate these concerns, the results of our empirical investigations suggest that AutoSpearman should be used in future studies which aim at ensuring high consistency and the automated mitigation of correlated metrics.

Finally, we would like to emphasise that the goal of this work is not to claim the generalisation of our results for every dataset and every model in software engineering. In addition, the best subset of metrics that one should include in studies depends on the goal of the studies. For example, if the goal of the study is prediction (i.e., aiming to achieve the highest predictive performance), one might prioritise resources on improving the model performance regardless of the correlation among metrics. On the other hand, if the goal of the study is model interpretation (i.e., aiming to examine the impact of various phenomena on software quality), one should avoid using commonly-used feature selection techniques when interpreting defect models. To mitigate the inconsistent nature of automated feature selection techniques, we recommend AutoSpearman be used in future studies when interpreting defect models.

Acknowledgements

C. Tantithamthavorn is supported by the Australian Research Council's Discovery Early Career Researcher Award (DECRA) funding scheme (DE200100941). C. Treude is supported by the Australian Research Council's Discovery Early Career Researcher Award (DECRA) funding scheme (DE180100153).

References

Agrawal A, Menzies T (2018) Is" better data" better than" better data miners"? In: 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE), IEEE, pp 1050–1061

Alckmin G, Kooistra L, Lucieer A, Rawnsley R (2019) Feature Filtering and Selection for Dry Matter Estimation on Perrenial Ryegrass: A Case Study of Vegetation Indices. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 42(2/W13)

Alzubi R, Ramzan N, Alzoubi H, Amira A (2017) A Hybrid Feature Selection Method for Complex Diseases SNPs. IEEE Access 6:1292–1301, DOI 10.1109/ACCESS.2017.2778268

Arisholm E, Briand LC, Johannessen EB (2010) A Systematic and Comprehensive Investigation of Methods to Build and Evaluate Fault Prediction Models. Journal of Systems and Software 83(1):2–17

Berry WD (1993) Understanding Regression Assumptions, vol 92. Sage Publications

Bettenburg N, Hassan AE (2010) Studying the Impact of Social Structures on Software Quality. In: Proceedings of the International Conference on Program Comprehension (ICPC), pp 124–133

Bird C, Bachmann A, Aune E, Duffy J, Bernstein A, Filkov V, Devanbu P (2009) Fair and Balanced?: Bias in Bug-fix Datasets. In: Proceedings of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering (ESEC/FSE), pp 121–130

Blake C, Merz C (1998) Uci repository of machine learning databases. University of California, Irvine, CA 55

Breiman L (2001) Random forests. Machine learning 45(1):5–32

Breiman L, Cutler A, Liaw A, Wiener M (2006) randomForest: Breiman and Cutler's Random Forests for Classification and Regression. R package version 4.6-12. Software available at URL: `https://cran.r-project.org/package=randomForest`

Cahill J, Hogan JM, Thomas R (2013) Predicting Fault-prone Software Modules with Rank Sum Classification. In: Proceedings of the Australian Software Engineering Conference (ASWEC), pp 211–219

Cai Y, Chow M, Lu W, Li L (2010) Statistical Feature Selection From Massive Data in Distribution Fault Diagnosis. IEEE Transactions on Power Systems 25(2):642–648

Canfora G, De Lucia A, Di Penta M, Oliveto R, Panichella A, Panichella S (2013) Multi-objective Cross-project Defect Prediction. In: Proceedings of the International Conference on Software Testing, Verification and Validation (ICST), pp 252–261

Chambers JM (1992) Statistical Models in S. Wadsworth. Pacific Grove, California

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: Synthetic Minority Oversampling TEchnique. Journal of Artificial Intelligence Research 16:321–357

D'Ambros M, Lanza M, Robbes R (2010) An Extensive Comparison of Bug Prediction Approaches. In: Proceedings of the International Conference on Mining Software Repositories (MSR), pp 31–41

D'Ambros M, Lanza M, Robbes R (2012) Evaluating Defect Prediction Approaches: A Benchmark and an Extensive Comparison. Empirical Software Engineering (EMSE) 17(4-5):531–577

Dash M, Liu H, Motoda H (2000) Consistency based Feature Selection. In: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pp 98–109

Efron B, Tibshirani RJ (1993) An Introduction to the Bootstrap. Springer US, Boston, MA

Elish KO, Elish MO (2008) Predicting Defect-prone Software Modules using Support Vector Machines. Journal of Systems and Software 81(5):649–660

Fox J (2015) Applied regression analysis and generalized linear models. Sage Publications

Fox J, Monette G (1992) Generalized Collinearity Diagnostics. Journal of the American Statistical Association (JASA) 87(417):178–183

Friedman J, Hastie T, Tibshirani R (2001) The Elements of Statistical Learning, vol 1. Springer series in statistics

Fu W, Menzies T, Shen X (2016) Tuning for Software Analytics: Is it really necessary? Information and Software Technology 76:135–146

Garner SR, *et al.* (1995) Weka: The Waikato Environment for Knowledge Analysis. In: Proceedings of the New Zealand Computer Science Research Students Conference (NZCSRSC), pp 57–64

Ghotra B, McIntosh S, Hassan AE (2015) Revisiting the Impact of Classification Techniques on the Performance of Defect Prediction Models. In: Proceedings of the International Conference on Software Engineering (ICSE), pp 789–800

Ghotra B, McIntosh S, Hassan AE (2017) A large-scale study of the impact of feature selection techniques on defect classification models. In: Proceedings of the 14th International Conference on Mining Software Repositories, pp 146–157

Gil Y, Lalouche G (2017) On the Correlation between Size and Metric Validity. Empirical Software Engineering (EMSE) 22(5):2585–2611

Guyon I, Elisseeff A (2003) An Introduction to Variable and Feature Selection. Journal of Machine Learning Research 3:1157–1182

Hair JF, Black WC, Babin BJ, Anderson RE, Tatham RL, *et al.* (2006) Multivariate data analysis (vol. 6)

Hall MA (1999) Correlation-based feature selection for machine learning. PhD thesis, University of Waikato Hamilton

Hall MA, Smith LA (1997) Feature Subset Selection: A Correlation Based Filter Approach

Hanley Ja, McNeil BJ (1982) The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143(4):29–36

Harrell Jr FE (2013) Hmisc: Harrell miscellaneous. R package version 3.12-2. Software available at URL: http://cran.r-project.org/web/packages/Hmisc

Harrell Jr FE (2015) Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis. Springer

Harrell Jr FE (2017) rms: Regression Modeling Strategies. R package version 5.1-1. Software available at URL: http://cran.r-project.org/web/packages/rms

Hinkle DE, Wiersma W, Jurs SG (2003) Applied Statistics for the Behavioral Sciences, vol 663. Houghton Mifflin College Division

Hsu HH, Hsieh CW, Lu MD (2011) Hybrid Feature Selection by Combining Filters and Wrappers. Expert Systems with Applications 38(7):8144–8150, DOI 10.1016/j.eswa.2010.12.156, URL http://dx.doi.org/10.1016/j.eswa.2010.12.156

Huan Liu, Setiono R (1995) Chi2: feature selection and discretization of numeric attributes. In: Proceedings of the International Conference on Tools with Artificial Intelligence, pp 388–391

Jiang Y, Cukic B, Menzies T (2008) Can Data Transformation Help in the Detection of Fault-prone Modules? In: Proceedings of the International Workshop on Defects in Large Software Systems (DEFECTS), pp 16–20

Jiarpakdee J, Tantithamthavorn C, Ihara A, Matsumoto K (2016) A Study of Redundant Metrics in Defect Prediction Datasets. In: Proceedings of the International Symposium on Software Reliability Engineering Workshops (ISSREW), pp 51–52

Jiarpakdee J, Tantithamthavorn C, Hassan AE (2018a) The Impact of Correlated Metrics on Defect Models. arXiv preprint arXiv:180110271 p To Appear

Jiarpakdee J, Tantithamthavorn C, Treude C (2018b) AutoSpearman: Automatically Mitigating Correlated Software Metrics for Interpreting Defect Models. In: Proceedings of the International Conference on Software Maintenance and Evolution (ICSME), pp 92–103

Jiarpakdee J, Tantithamthavorn C, Treude C (2018c) Online Appendix for "Should Automated Feature Selection Techniques be Applied when Interpreting Defect Models?". https://github.com/software-analytics/autospearman-extension-appendix

Jiarpakdee J, Tantithamthavorn C, Dam HK, Grundy J (2020) An empirical study of model-agnostic techniques for defect prediction models. Transactions on Software Engineering (TSE) pp 1–1

John GH, Kohavi R, Pfleger K (1994) Irrelevant Features and the Subset Selection Problem. In: Proceedings of the International Conference on Machine Learning (ICML), pp 121–129

Jureczko M, Madeyski L (2010) Towards Identifying Software Project Clusters with Regard to Defect Prediction. In: Proceedings of the International Conference on Predictive Models in Software Engineering (PROMISE), p 9

Kamei Y, Shihab E, Adams B, Hassan AE, Mockus A, Sinha A, Ubayashi N (2013) A Large-Scale Empirical Study of Just-In-Time Quality Assurance. Transactions on Software Engineering (TSE) 39(6):757–773

Kaur A, Malhotra R (2008) Application of Random Forest in Predicting Fault-prone Classes. In: Proceedings of International Conference on the Advanced Computer Theory and Engineering (ICACTE), pp 37–43

Kim S, Zhang H, Wu R, Gong L (2011) Dealing with Noise in Defect Prediction. In: Proceedings of the International Conference on Software Engineering (ICSE), pp 481–490

Kohavi R, John GH (1997) Wrappers for Feature Subset Selection. Artificial Intelligence 97(1-2):273–324

Koru AG, Liu H (2005) An Investigation of the Effect of Module Size on Defect Prediction Using Static Measures. Software Engineering Notes (SEN) 30:1–5

Kraemer HC, Morgan GA, Leech NL, Gliner JA, Vaske JJ, Harmon RJ (2003) Measures of Clinical Significance. Journal of the American Academy of Child & Adolescent Psychiatry (JAACAP) 42(12):1524–1529

Kuhn M, Wing J, Weston S, Williams A, Keefer C, Engelhardt A, Cooper T, Mayer Z, Kenkel B, Team R, *et al.* (2017) caret: Classification and regression training. R package version 6.0–78. Software available at URL: https://cran.r-project.org/package=caret

Lessmann S, Baesens B, Mues C, Pietsch S (2008) Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings. Transactions on Software Engineering (TSE) 34(4):485–496

Lewis DD, Ringuette M (1994) A comparison of two learning algorithms for text categorization. In: Annual Symposium on Document Analysis and Information Retrieval, vol 33, pp 81–93

Li J, Cheng K, Wang S, Morstatter F, Trevino RP, Tang J, Liu H (2017) Feature Selection: A Data Perspective. ACM Computing Surveys (CSUR) 50(6):94

Lu H, Kocaguneli E, Cukic B (2014) Defect Prediction between Software Versions with Active Learning and Dimensionality Reduction. In: Proceedings of the International Symposium on Software Reliability Engineering (ISSRE), pp 312–322

Mason CH, Perreault Jr WD (1991) Collinearity, Power, and Interpretation of Multiple Regression Analysis. Journal of Marketing Research (JMR) pp 268–280

Matthews BW (1975) Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochimica et Biophysica Acta (BBA)-Protein Structure 405(2):442–451

McHugh ML (2013) The Chi-square Test of Independence. Biochemia Medica 23(2):143–149

McIntosh S, Kamei Y, Adams B, Hassan AE (2014) The Impact of Code Review Coverage and Code Review Participation on Software Quality. In: Proceedings of the International Conference on Mining Software Repositories (MSR), pp 192–201

Mende T (2010) Replication of Defect Prediction Studies: Problems, Pitfalls and Recommendations. In: Proceedings of the International Conference on Predictive Models in Software Engineering (PROMISE), pp 1–10

Mende T, Koschke R (2009) Revisiting the Evaluation of Defect Prediction Models. Proceedings of the International Conference on Predictive Models in Software Engineering (PROMISE) pp 7–16

Menzies T (2018) The Unreasonable Effectiveness of Software Analytics. IEEE Software 35(2):96–98, DOI 10.1109/MS.2018.1661323

Menzies T, Greenwald J, Frank A (2007) Data Mining Static Code Attributes to Learn Defect Predictors. Transactions on Software Engineering (TSE) 33(1):2–13

Menzies T, Caglayan B, Kocaguneli E, Krall J, Peters F, Turhan B (2012) The Promise Repository of Empirical Software Engineering Data

Mersmann O, Beleites C, Hurling R, Friedman A (2018) microbenchmark: Accurate Timing Functions. R package version 1.4-6. Software available at URL: `https://cran.r-project.org/package=microbenchmark`

Mitchell TM (1997) Machine Learning. McGraw Hill

Moser R, Pedrycz W, Succi G (2008) A Comparative Analysis of the Efficiency of Change Metrics and Static Code Attributes for Defect Prediction. In: Proceedings of the International Conference on Software Engineering (ICSE), pp 181–190

Nam J, Fu W, Kim S, Menzies T, Tan L (2017) Heterogeneous Defect Prediction. Transactions on Software Engineering (TSE) p In Press

Okutan A, Yıldız OT (2014) Software Defect Prediction using Bayesian Networks. Empirical Software Engineering (EMSE) 19(1):154–181

Osman H, Ghafari M, Nierstrasz O (2018) The impact of feature selection on predicting the number of bugs. arXiv preprint arXiv:180704486

Pandari Y, Thangavel P, Senthamaraikannan H, Jagadeeswaran S (2019) HybridFS: A Hybrid Filter-Wrapper Feature Selection Method. R package version 0.1.3. Software available at URL: `https://cran.r-project.org/package=HybridFS`

Petrić J, Bowes D, Hall T, Christianson B, Baddoo N (2016) The Jinx on the NASA Software Defect Data Sets. In: Proceedings of the International Conference on Evaluation and Assessment in Software Engineering (EASE), pp 13–17

Rahman F, Devanbu P (2013) How, and Why, Process Metrics are Better. In: Proceedings of the International Conference on Software Engineering (ICSE), pp 432–441

Robles G (2010) Replicating MSR: A Study of the Potential Replicability of Papers Published in the Mining Software Repositories Proceedings. In: Proceedings of the International Conference on Mining Software Repositories (MSR), pp 171–180

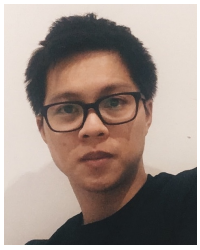
Rodríguez D, Ruiz R, Cuadrado-Gallego J, Aguilar-Ruiz J (2007) Detecting Fault Modules Applying Feature Selection to Classifiers. In: Proceedings of the International Conference on Information Reuse and Integration (IRI), pp 667–672

Romano J, Kromrey JD, Coraggio J, Skowronek J (2006) Appropriate Statistics for Ordinal Level Data: Should we really be using T-test and Cohen's d for Evaluating group differences on the NSSE and other surveys. In: Annual meeting of the Florida Association of Institutional Research (FAIR), pp 1–33

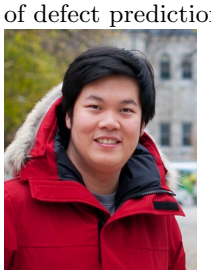
Romanski P, Kotthoff L (2013) FSelector: Selecting attributes. R package version 0.19. Software available at URL: `https://cran.r-project.org/package=FSelector`

Shepperd M, Song Q, Sun Z, Mair C (2013) Data Quality: Some Comments on the NASA Software Defect Datasets. Transactions on Software Engineering (TSE) 39(9):1208–1215
Shepperd M, Bowes D, Hall T (2014) Researcher Bias: The Use of Machine Learning in Software Defect Prediction. Transactions on Software Engineering (TSE) 40(6):603–616
Shivaji S, Whitehead EJ, Akella R, Kim S (2013) Reducing Features to Improve Code Change-Based Bug Prediction. Transactions on Software Engineering (TSE) 39(4):552–569
Strobl C, Boulesteix AL, Kneib T, Augustin T, Zeileis A (2008) Conditional Variable Importance for Random Forests. BMC Bioinformatics 9(1):307
Tantithamthavorn C (2017) ScottKnottESD: The Scott-Knott Effect Size Difference (ESD) Test. R package version 2.0. Software available at URL: `https://cran.r-project.org/web/packages/ScottKnottESD`
Tantithamthavorn C, Hassan AE (2018) An Experience Report on Defect Modelling in Practice: Pitfalls and Challenges. In: In Proceedings of the International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), pp 286–295
Tantithamthavorn C, Jiarpakdee J (2018) Rnalytica: An R package of the Miscellaneous Functions for Data Analytics Research. `https://github.com/software-analytics/Rnalytica`
Tantithamthavorn C, McIntosh S, Hassan AE, Ihara A, Matsumoto K (2015) The Impact of Mislabelling on the Performance and Interpretation of Defect Prediction Models. In: Proceeding of the International Conference on Software Engineering (ICSE), pp 812–823
Tantithamthavorn C, McIntosh S, Hassan AE, Matsumoto K (2016a) Automated Parameter Optimization of Classification Techniques for Defect Prediction Models. In: Proceedings of the International Conference on Software Engineering (ICSE), pp 321–332
Tantithamthavorn C, McIntosh S, Hassan AE, Matsumoto K (2016b) Comments on "Researcher Bias: The Use of Machine Learning in Software Defect Prediction". Transactions on Software Engineering (TSE) 42(11):1092–1094
Tantithamthavorn C, McIntosh S, Hassan AE, Matsumoto K (2017) An Empirical Comparison of Model Validation Techniques for Defect Prediction Models. Transactions on Software Engineering (TSE) 43(1):1–18
Tantithamthavorn C, Hassan AE, Matsumoto K (2019a) The Impact of Class Rebalancing Techniques on The Performance and Interpretation of Defect Prediction Models. Transactions on Software Engineering (TSE) p Preprints
Tantithamthavorn C, McIntosh S, Hassan AE, Matsumoto K (2019b) The Impact of Automated Parameter Optimization on Defect Prediction Models. Transactions on Software Engineering (TSE) 45(7):683–711
Team RC, contributors worldwide (2017) stats: The R Stats Package. R Package. Version 3.4.0
Tian Y, Nagappan M, Lo D, Hassan AE (2015) What Are the Characteristics of High-Rated Apps? A Case Study on Free Android Applications. In: Proceedings of the International Conference on Software Maintenance and Evolution (ICSME), pp 301–310
Torgo L, Torgo ML (2015) DMwR: Functions and Data for Data Mining with R. R package version 0.4.1. Software available at URL: `https://cran.r-project.org/package=DMwR`
Tosun A, Bener A (2009) Reducing False Alarms in Software Defect Prediction by Decision Threshold Optimization. In: Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM), pp 477–480
Xu Z, Liu J, Yang Z, An G, Jia X (2016) The Impact of Feature Selection on Defect Prediction Performance: An Empirical Comparison. In: Proceedings of the International Symposium on Software Reliability Engineering (ISSRE), pp 309–320
Yan K, Zhang D (2015) Feature selection and analysis on correlated gas sensor data with recursive feature elimination. Sensors and Actuators B: Chemical 212:353–363
Yathish S, Jiarpakdee J, Thongtanunam P, Tantithamthavorn C (2019) Mining Software Defects: Should We Consider Affected Releases? In: In Proceedings of the International Conference on Software Engineering (ICSE), p To Appear
Zhang F, Hassan AE, McIntosh S, Zou Y (2017) The Use of Summation to Aggregate Software Metrics Hinders the Performance of Defect Prediction Models. Transactions on Software Engineering (TSE) 43(5):476–491
Zimmermann T, Premraj R, Zeller A (2007) Predicting Defects for Eclipse. In: Proceedings of the International Workshop on Predictor Models in Software Engineering (PROMISE), pp 9–19
Zimmermann T, Nagappan N, Gall H, Giger E, Murphy B (2009) Cross-project Defect Prediction. In: Proceedings of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering (ESEC/FSE), pp 91–100

Biography



Jirayus Jiarpakdee is a Ph.D. candidate at Monash University, Australia. His research interests include empirical software engineering and mining software repositories (MSR). The goal of his Ph.D. is to apply the knowledge of statistical modelling, experimental design, and software engineering to improve the explainability of defect prediction models.



Chakkrit Tantithamthavorn is an ARC DECRA Fellow and a lecturer in the Faculty of Information Technology, Monash University, Australia. His research interests focused on developing practical and explainable analytics for preventing software defects. His work has been published at several top-tier software engineering venues (e.g., TSE, ICSE, EMSE). He received the B.E. degree from Kasetsart University, Thailand, the M.E. and Ph.D. degrees from NAIST, Japan. More about Chakkrit and his work is available online at `http://chakkrit.com`.

Christoph Treude is an ARC DECRA Fellow and a Senior Lecturer in the School of Computer Science at the University of Adelaide, Australia. He received his Ph.D. in computer science from the University of Victoria, Canada. The goal of his research is to advance collaborative software engineering through empirical studies and the innovation of tools and processes that explicitly take the wide variety of artefacts available in a software repository into account. He currently serves on the editorial board of the Empirical Software Engineering journal and as general co-chair for the IEEE International Conference on Software Maintenance and Evolution 2020.