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

The Impact of Feature Reduction Techniques on Defect Prediction Models

Masanari Kondo · Cor-Paul Bezemer · Yasutaka Kamei · Ahmed E. Hassan · Osamu Mizuno

the date of receipt and acceptance should be inserted later

Abstract Defect prediction is an important task for preserving software quality. Most prior work on defect prediction uses software features, such as the number of lines of code, to predict whether a file or commit will be defective in the future. There are several reasons to keep the number of features that are used in a defect prediction model small. For example, using a small number of features avoids the problem of multicollinearity and the so-called 'curse of dimensionality'. Feature selection and reduction techniques can help to reduce the number of features in a model. Feature selection techniques reduce the number of features in a model by selecting the most important ones, while feature reduction techniques reduce the number of features by creating new, combined features from the original features. Several recent studies have investigated the impact of feature *selection* techniques on defect prediction. However, there do not exist large-scale studies in which the impact of multiple feature *reduction* techniques on defect prediction is investigated.

Masanari Kondo, Osamu Mizuno
Software Engineering Laboratory (SEL)
Kyoto Institute of Technology, Japan
E-mail: m-kondo@se.is.kit.ac.jp, o-mizuno@kit.ac.jp

Cor-Paul Bezemer
Department of Electrical and Computer Engineering
University of Alberta, Canada
E-mail: bezemer@ualberta.ca

Yasutaka Kamei
Principles of Software Languages group (POSL)
Kyushu University, Japan
E-mail: kamei@ait.kyushu-u.ac.jp

Ahmed E. Hassan
Software Analysis and Intelligence Lab (SAIL), School of Computing
Queen's University
Kingston, Ontario, Canada
E-mail: ahmed@cs.queensu.ca

In this paper, we study the impact of eight feature reduction techniques on the performance and the variance in performance of five supervised learning and five unsupervised defect prediction models. In addition, we compare the impact of the studied feature reduction techniques with the impact of the two best-performing feature selection techniques (according to prior work).

The following findings are the highlights of our study: (1) The studied correlation and consistency-based feature selection techniques result in the best-performing supervised defect prediction models, while feature reduction techniques using neural network-based techniques (restricted Boltzmann machine and autoencoder) result in the best-performing unsupervised defect prediction models. In both cases, the defect prediction models that use the selected/generated features perform better than those that use the original features (in terms of AUC and performance variance). (2) Neural network-based feature reduction techniques generate features that have a small variance across both supervised and unsupervised defect prediction models. Hence, we recommend that practitioners who do not wish to choose a best-performing defect prediction model for their data use a neural network-based feature reduction technique.

1 Introduction

Software developers have limited time to test their software. Hence, developers need to be selective when deciding where to focus their testing effort. Defect prediction models help developers focus their limited testing effort on components that are the most likely to be defective. Because detecting defects at an early stage can considerably reduce the development cost [74], defect prediction models have received widespread attention in software engineering research.

Many software features (e.g., software complexity features) can be used in defect prediction models [4, 11, 27, 36, 47]. However, it is important to carefully select the set of features that is used in such models, as using a set of features that is too large does not automatically result in better defect prediction. For example, prior studies showed that reducing the number of features avoids the problem of multicollinearity [15] and the curse of dimensionality [5]. Hence, many of the existing defect prediction models used feature selection or reduction techniques [8, 11, 16, 18–20, 23, 29, 33, 34, 48, 52–54, 61–64, 69, 76, 81].

Feature selection techniques reduce the number of features in a model by selecting the most important ones, while feature reduction techniques reduce the number of features by creating new, combined features from the original features. Recent studies [18, 76] investigated the impact of feature *selection* techniques on the performance of defect prediction models. However, this paper describes the first large-scale study on multiple feature *reduction* techniques and their impact on a large number of prediction models.

In this paper, we compared the impact of the original features, features that are generated using *traditional feature reduction techniques* (i.e., PCA [11], FastMap [14], feature agglomeration [65], random projections [6], TCA [57] and TCA+ [54]), and features that are generated using *neural network-based feature reduction techniques* (i.e., restricted Boltzmann machine (RBM) [70] and autoencoder (AE) [30]) on defect prediction models. In addition, we compared the impact of features that are generated using feature reduction techniques with features that are selected using the best-performing feature selection techniques in prior work (correlation and consistency-based feature selection) [18, 76]. We compared the features along two dimensions: their performance (area under the receiver operating characteristic curve (AUC)), and their performance variance (interquartile range (IQR)). The receiver operating characteristic curve is created by plotting the true positive rate against the false positive rate, hence, the AUC represents the balance between the true positive rate and the false positive rate. The IQR represents the variance of a data distribution.

We conducted experiments on three publicly available datasets that contain software defects (the PROMISE [33], cleaned NASA [66], and AEEEM [11] datasets). Our ultimate goal is to identify which feature reduction techniques yield new, powerful features that preserve the predictive power of the original features, and improve the prediction performance compared to feature selection techniques. We studied the impact of feature reduction techniques on five supervised learning and five unsupervised learning models for defect prediction in our experiments. In particular, we focus on the following research questions:

RQ1: **What is the impact of feature reduction techniques on the performance of defect prediction models?**
Motivation: Reducing the number of features in a model can address the multicollinearity problem [15] and the curse of dimensionality [5]. In this RQ, we studied how feature reduction techniques impact the performance of supervised and unsupervised defect prediction models.
Results: Feature agglomeration and TCA can reduce the number of features, while preserving an AUC that is as good as that of the original features for supervised models. In addition, the AUC of unsupervised defect prediction models is significantly better when preprocessing the features with neural network-based feature reduction techniques than other feature reduction techniques.

RQ2: **What is the impact of feature reduction techniques on the variance of the performance across defect prediction models?**
Motivation: The AUC for a dataset can vary across defect prediction models. Hence, it can be challenging for practitioners to choose the defect prediction model that performs best on their data. If all defect prediction models have a small performance variance for a dataset, practitioners can avoid having to make this challenging choice.
Results: Neural network-based feature reduction techniques (RBM and AE) generate features that improve the variance of the performance across different defect prediction models in many cases when used in a supervised or unsupervised defect prediction model. In addition, almost all feature reduction techniques (except PCA) generate features that improve the variance of the performance across different defect prediction models in many cases when used in an unsupervised defect prediction model.

RQ3: **How do feature selection techniques compare to feature reduction techniques when applied to defect prediction?**

Motivation: Prior work [18, 76] showed that several feature selection techniques outperform the original models. In this RQ, we studied how feature selection techniques compare to feature reduction techniques.
Results: For the supervised defect prediction models, the studied feature selection techniques (correlation and consistency-based feature selection) outperform all the studied feature reduction techniques. However, for the unsupervised defect prediction models, the neural network-based feature reduction techniques (RBM and AE) outperform the other studied feature selection/reduction techniques.

Our results provide practitioners with advice on which feature selection/reduction technique to use in combination with a defect prediction model. In particular, we recommend to use a feature selection technique when using a supervised defect prediction model, and a neural network-based feature reduction technique when using an unsupervised defect prediction model, as these feature selection/reduction techniques improve the variance across defect prediction models, while outperforming the other feature reduction techniques.

The organization of our paper is as follows. Section 2 introduces related work. Section 3 presents our methodology. Section 4 presents experimental setup. Section 5 presents the results of our experiment. Section 6 discusses these results. Section 7 describes the threats to the validity of our findings. Section 8 presents the conclusion.

2 Related work

In this section, we discuss related work on defect prediction, and feature selection and reduction.

2.1 Defect prediction

Defect prediction approaches can be divided in two categories: approaches that use supervised learning and unsupervised learning. Supervised defect prediction models need training data and test data. Usually, the training data and test data are collected from the same, or very similar projects from the same organization (within-project defect prediction). However, sometimes it is difficult to collect sufficient training data from the same project or organization (e.g., when the project is new).

In such cases, cross-project defect prediction can be a viable alternative solution. Cross-project defect prediction uses training datasets that contain data that comes from multiple projects, or even from multiple organizations. Cross-project defect prediction offers a solution for the problem of within-project defect prediction since small datasets can be extended with data from other projects. Hence, cross-project defect prediction is one of the important research themes in defect prediction research. However, cross-project defect prediction has several challenges [3, 28, 29, 50, 53]. For instance, cross-project defect prediction has the problem of heterogeneous original features across datasets [53]. There exists one solution for this challenge [51], however; converging datasets collected from multiple companies to one dataset when

Table 1 Overview of prior work that uses feature selection or reduction techniques in combination with defect prediction.

Technique	Family	Example of Techniques	References
Feature Selection	Filter-based Feature Ranking	Chi-Square Correlation	[16, 18–20, 33, 45, 48, 61, 69, 76]
	Filter-based Subset Selection	Correlation-based Feature Subset Selection	[8, 16, 18, 48, 61, 76]
	Wrapper-based Subset Selection	Logistic Regression	[18, 61, 63, 64, 69, 76]
	Greedy-based Feature Selection	Greedy Forward Selection Algorithm	[48]
	Distance-based Feature Selection	EM Algorithm	[29]
	State-of-the-Art	MVS	[52]
	Others	Significance	[34, 53, 69, 81]
Feature Reduction	PCA	PCA	[8, 11, 18, 49, 50, 56, 61, 62, 76]
	State-of-the-Art	TCA	[54, 62]

they contain different features still remains an open challenge and further investigation is needed.

One way to overcome the challenge of heterogeneity is to use unsupervised defect prediction models [2, 7, 52, 77, 78, 80]. Unsupervised models have the advantage that they do not need a training dataset, and therefore, are not affected by the problem of heterogeneous features [2, 7, 52, 77, 78, 80]. Recently, Zhang et al. [78] summarized the accuracy of several supervised and unsupervised models for defect prediction. They concluded that connectivity-based unsupervised models have an accuracy that is as good as that of supervised models. Therefore, unsupervised models are a viable alternative to supervised models for defect prediction. Note that unsupervised models do not need training data, and are therefore always within-project defect prediction models.

2.2 Feature selection and reduction

Using feature selection or reduction technique has the advantage of addressing the *curse of dimensionality* [5]. This problem is originally considered a dynamic optimization problem. However, machine learning models also need to consider this problem. The problem occurs when having a large number of features yet a small sample size in machine learning models. In this case, the sample size is not large enough to search the representations of all the combinations of features, and to generalize their parameters (resulting in overfitting) [31]. Hence, the prediction performance of these models for unseen data would be worse, and could lead to a classification error. Feature selection or reduction techniques can address this problem by removing or combining redundant and irrelevant features. In addition, prior work [45, 46] reported that the prediction performance of a model is often determined by only a few features. Therefore, feature selection or reduction techniques can often be applied without negatively affecting the prediction performance.

In this paper, we define feature selection and reduction as follows:

- Feature selection techniques reduce the number of features by selecting a subset of the original features.
- Feature reduction techniques reduce the number of features by combining original features into new features.

Several researchers studied the impact of feature selection techniques on defect prediction models [8, 16, 18–20, 29, 33, 34, 45, 48, 52, 53, 61, 63, 64, 69, 76, 81] . Table 1 gives an overview of prior work that uses feature selection or reduction techniques. For instance, Ghotra et al. [18] summarized the impact of feature selection techniques for defect prediction. They reported that correlation-based feature selection outperforms the other feature selection techniques. In addition, they showed that the impact of feature selection techniques varies across the studied datasets. Xu et al. [76] also summarized the impact of feature selection techniques on defect prediction models. They also reported that the effectiveness of feature selection techniques exhibits significant differences across studied datasets. Menzies et al. [45] used the information gain measure to rank features based on their importance in a defect prediction model. They found that defect prediction can often be done using only a small set of features. For example, they reduced the number of features in the CM1 project from 38 to 3 without affecting the prediction performance.

The impact of feature reduction techniques on defect prediction models has not been studied as extensively. Most researchers use principal component analysis (PCA) [8, 11, 18, 49, 50, 56, 61, 62, 76], and only a few researchers use other feature reduction techniques [54, 62]. For instance, D'Ambros et al. [11] compared class-level defect prediction models to present a benchmark for defect prediction. They used PCA to avoid the problem of multicollinearity [15] among the independent variables. Nagappan et al. [49] predicted post-release defects using complexity features. They built prediction models using PCA to avoid the problem of multicollinearity. Neumann [56] proposed PCA-ANN which is a combination of PCA and artificial neural networks. Neumann also used PCA to avoid the problem of multicollinearity. Challagulla et al. [8] compared several prediction models for identifying defects. In addition, they compared PCA with feature selection techniques such as feature subset selection. They concluded that feature selection techniques are better than PCA, and that PCA did not add any advantages for defect prediction. Rathore et al. [61] compared the performance of feature selection techniques. In this comparison, they also used PCA. They found that PCA is one of the best-performing techniques in this comparison. Ren et al. [62] extended PCA to address class imbalance problem for defect prediction. Nam et al. [54] applied transfer component analyses (TCA and TCA+) to training and test data to convert the data to be closer than the original data. This process addressed the challenge of heterogeneity of training data and test data in cross-project defect prediction. Nam's approach significantly improved cross-project defect prediction performance.

Despite the amount of prior work on feature reduction techniques (mostly on PCA), no prior work has conducted a systematic study of the impact of feature reduction techniques on defect prediction models. In this paper, we provide such a systematic study of the impact of eight feature reduction and two best-performing

Table 2 Description of studied projects

Studied Dataset	Project	# of Entities	# of Defective	% Defective	# of Features*	# of Studied Features*
PROMISE	Ant v1.7	745	166	22.3	24	20
	Camel v1.6	965	188	19.5	24	20
	Ivy v1.4	241	16	6.6	24	20
	Jedit v4.0	306	75	24.5	24	20
	Log4j v1.0	135	34	25.2	24	20
	Lucene v2.4	340	203	59.7	24	20
	POI v3.0	442	281	63.6	24	20
	Tomcat v6.0	858	77	9.0	24	20
	Xalan v2.6	885	411	46.4	24	20
	Xerces v1.3	453	69	15.2	24	20
NASA	CM1	327	42	12.8	38	37
	JM1	7,782	1,672	21.5	22	21
	KC3	194	36	18.6	40	39
	MC1	1,988	46	2.3	39	38
	MC2	125	44	35.2	40	39
	MW1	253	27	10.7	38	37
	PC1	705	61	8.7	38	37
	PC2	745	16	2.1	37	36
	PC3	1,077	134	12.4	38	37
	PC4	1,287	177	13.8	38	37
	PC5	1,711	471	27.5	39	38
AEEEM	Eclipse JDT Core	997	206	20.7	291	212
	Equinox	324	129	39.8	291	212
	Apache Lucene	691	64	9.3	291	212
	Mylyn	1,862	245	13.2	291	212
	Eclipse PDE UI	1,497	209	14.0	291	212

* We removed features that are not related to source code. For instance, the name of the file, name of the class and the version. Hence, the number of studied features are different from the total number of features.

feature selection techniques on five supervised and five unsupervised defect prediction models.

Finally, Peters et al. [58] showed that often not only the features can be reduced without negatively affecting the performance, but also the amount of the rows in the data.

3 Methodology

In this section, we describe our methodology. In particular, we discuss our studied datasets, defect prediction models, feature selection techniques, feature reduction techniques, evaluation measure, our preprocessing steps, and our validation scheme.

3.1 Studied datasets

In our work, we used three publicly available datasets (the PROMISE [33], cleaned NASA [55] and AEEEM [11] datasets) that were used by prior work [78] on supervised and unsupervised defect prediction models. Table 2 describes the studied datasets. All datasets contain popular software features for measuring source code complexity (see Table 3 for a summary of the used features). Each entity in a dataset is labelled as defective or clean.

Table 3 Studied features

Studied Dataset	Features	Reference
PROMISE	CK	Chidamber et al. [9]
	OO	Basili et al. [4]
NASA	McCabe	McCabe [43]
	Halstead	Halstead [24]
AEEEM	CK	Chidamber et al. [9]
	OO	Basili et al. [4]
	number of previous defects	Kim et al. [36]
	change features	Moser et al. [47]
	complexity code change features	Hassan [27]
	churn of CK and OO	D'Ambros et al. [11]
	entropy of CK and OO	D'Ambros et al. [11]

The PROMISE dataset contains several types of projects. We chose the 10 projects that were used by prior work [78], to ease the comparison of our results with prior work. All studied PROMISE projects have the same number of features. The PROMISE dataset contains the Chidamber and Kemerer (CK) features [9] and an additional set of object-oriented (OO) features [4].

The NASA dataset [55] contains 11 projects. Each project in the NASA dataset has a different number of features. The NASA dataset contains McCabe features [43] and Halstead features [24]. We used the cleaned version [66] of the NASA dataset, because prior studies [59, 66] showed that the original version of the NASA dataset contains inconsistent and mislabeled data.

The AEEEM dataset [11] contains five projects. All projects have the same number of features. Like the PROMISE dataset, the AEEEM dataset contains the CK and OO features. However, the AEEEM data also contains the number of previous defects [36], change features [47], complexity code change features [27], and the churn and entropy of the CK and OO [11] features.

3.2 Studied defect prediction models

We focused on defect prediction models that were used by prior work [78], to make our results easier to compare. We studied five supervised models and five unsupervised models. Below we give a brief overview of the ideas behinds these models. For a detailed overview, we refer the reader to the original papers in which these models were introduced. We studied the following supervised defect prediction models:

- *Logistic Regression (LR) [44]:* LR is one of the most commonly used models for defect prediction. LR expresses the relationship between one or more independent variables (i.e., the original features) and one dependent variable (i.e., defective or clean) using a polynomial expression and a *sigmoid function* [25].
- *Decision Tree (J48) [60]:* J48 is a decision tree implementation in WEKA [21]. The decision tree uses a tree structure to decide on the dependent variable. In this tree, each node corresponds to one of the independent variables with a condition.

J48 traverses the tree from the root to a leaf, while taking into account the input entity and the conditions in the tree. Each leaf corresponds to a value of the dependent variable.
- *Random Forest (RF) [32]:* RF is a popular *ensemble learning* model. RF builds multiple decision trees based on subsets of training data that are randomly selected. RF decides on a value of the dependent variable by taking the result of a majority of the decision trees.
- *Naive Bayes (NB) [79]:* NB is a probability-based classifier that follows Bayes' theorem. Bayes' theorem describes the probability of an event, given knowledge of conditions that could be related to the event.
- *Logistic Model Tree (LMT) [39]:* LMT is a classifier which combines a decision tree and a logistic regression model. Like the decision tree, LMT follows a tree structure. However, LMT uses logistic regressions instead of values in the leaves.

We used the `caret` library in `R` [38] to optimize the parameters of the supervised models as suggested by Tantithamthavorn et al. [72].

We studied the following unsupervised defect prediction models:

- *Spectral Clustering (SC) [75]:* SC labels entities using a graph that is based on similarities across entities. In this graph, each node is an entity and each edge represents the similarity of the entities it connects. SC cuts sparse edges in this graph by classifying eigenvectors of the *Laplacian matrix* [75] of the graph. Following this process, SC divides the entities into two groups.
- *k-means (KM) [26]:* KM is a popular clustering approach. KM classifies entities based on the distances between entities and the center of a class (i.e., the mean of all entities in that class). In this paper, we used the Euclidean distance as the distance metric.
- *Partition Around Medoids (PAM) [35]:* PAM is an approach that is similar to KM. While KM uses the center of a class, PAM uses *medoids*. A medoid is an entity of which the sum of all the distances to the other entities in a class is at its minimum. Because PAM uses the medoid instead of the center, PAM is less likely to be affected by outliers than KM.
- *Fuzzy C-Means (FCM) [13]:* FCM is also an approach that is similar to KM. While KM classifies each entity to only one class in its process, FCM allows entities to be a member of more than one class. The membership is expressed as a probability.
- *Neural Gas (NG) [42]:* NG is an approach that is similar to a *self-organizing map* [37]. NG generates *weighted points* which have random features. Hence, the weighted points are distributed across the feature space. For each learning iteration, the features of the weighted points are updated by distances to closer entities. Finally, the weighted points become the class centers.

We used the default parameters of the implementations for the unsupervised models. We set the number of clusters to two, as defect prediction is a binary problem. Table 4 shows the libraries that we used for the implementation of the defect prediction models.

Labeling technique in the unsupervised models: The unsupervised models classify the data in two unlabeled clusters. We adopted the following heuristic to

Table 4 Packages used for experiments

Defect Prediction Models	Packages
LR	The `caret` package in `R`
RF	The `caret` package in `R`
NB	The `caret` package in `R`
J48	The `caret` package in `R`
LMT	The `caret` package in `R`
SC	Zhang et al.'s implementation [78] (see app. A)
KM	The `cclust` package in `R`
PAM	The `cluster` package in `R`
FCM	The `e1071` package in `R`
NG	The `cclust` package in `R`

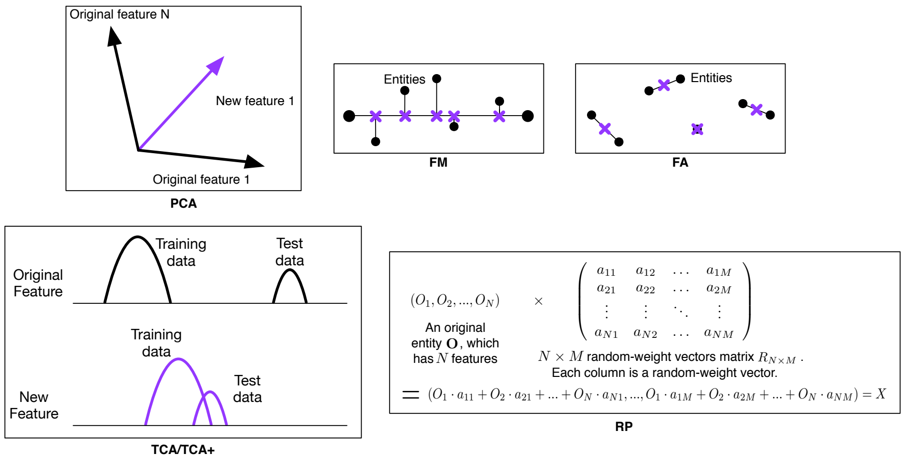

Original feature N
New feature 1
Original feature 1
PCA
Entities
FM
Entities
FA
Original Feature
Training data
Test data
Training data
Test data
New Feature
TCA/TCA+
$(O_1, O_2, ..., O_N)$
An original entity $\mathbf{O}$, which has N features
$\times$
$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1M} \\ a_{21} & a_{22} & \dots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NM} \end{pmatrix}$
$N \times M$ random-weight vectors matrix $R_{N \times M}$.
Each column is a random-weight vector.
$= (O_1 \cdot a_{11} + O_2 \cdot a_{21} + ... + O_N \cdot a_{N1}, ..., O_1 \cdot a_{1M} + O_2 \cdot a_{2M} + ... + O_N \cdot a_{NM}) = X$
RP

Fig. 1 A visual overview of the core concepts of the traditional feature reduction techniques. The black symbols represent the original features (or entities in FM and FA) and the purple symbols represent the newly-generated features (or entities in FM and FA). RP (Random projection) transforms an original entity to a new entity using *M* random-weight vectors.

identify the defective cluster: "*For most features, software entities containing defects generally have larger values than software entities without defects*" [78]. In particular, we used the sum of row average of the normalized features in each cluster, to decide which cluster contains the defects [78]. To calculate the sum of row average, we first summed the entity values in each cluster, respectively. Then, we calculated the average values for each cluster. The cluster with the larger average value was identified as the cluster with the defective entities.

3.3 Studied feature reduction techniques

In this subsection, we discuss the studied feature reduction techniques. We studied two types of feature reduction techniques: *traditional* and *neural network-based* feature reduction techniques. We give a brief overview of the core concepts of each

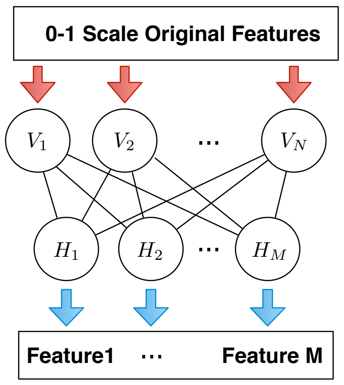


Fig. 2 An overview of neural network-based feature reduction techniques (RBM and AE). RBM and AE convert the original features (V_i), which values must range between 0 and 1, into M new features (H_i). Note that the original input data may need to be preprocessed to satisfy the 0-1 range requirement.

feature reduction technique. For more precise details, we refer to the references that are mentioned for each technique. Figure 1 shows a visualization of the traditional feature reduction techniques (PCA, FM, FA, TCA/TCA+, RP). Figure 2 gives an overview of neural network-based feature reduction techniques (RBM and AE).

3.3.1 Traditional feature reduction techniques

We studied the following traditional feature reduction techniques.

– *Principal Component Analysis (PCA):* PCA is one of the most commonly used feature reduction techniques in defect prediction [8, 11, 18, 23, 50, 61, 62, 76]. PCA reduces the number of features by projecting the original set of features on a smaller number of principal components.
– *FastMap (FM):*For N original features, FM [14] first generates a (N-1)-dimensional orthogonal hyper-plane of the line between two entities that are far from each other. Second, FM projects the other entities on this hyper-plane. Because FM projects the entities on the $N-1$ orthogonal hyper-plane, we can reduce one feature from the original features. FM repeats this procedure until we get the required number of new features. For instance, if we want three features to visualize our data from the N original features, we repeat the procedure N-3 times.
– *Feature Agglomeration (FA):* FA is a simple hierarchical clustering algorithm [65]. FA starts by creating a new feature from each original feature. Then, FA merges the two nearest (based on their Euclidean distance) features into one feature, and repeats this process until the desired number of features is reached.
– *Transfer Component Analysis (TCA and TCA+):* TCA [57] creates new features from the original features by projecting them on so-called transfer components (similar to PCA). However, the goal of TCA is not to reduce the number of features, but to reduce the gap between the distribution of the training and testing data. During this process, the number of features is often reduced. Hence, TCA

can be used as a feature reduction technique. TCA+ [54] is an extension of TCA, which optimizes the data using a preprocessing step according to the gap between the distribution of the training and testing data, such as scaling the original features between 0 and 1 instead of using the z-score.
- *Random Projection (RP):*RP projects the original N-dimensional features onto M generated features ($M \ll N$) using a $N \times M$ random-weight vectors matrix [6]. The equation of RP is as follows:

$$\mathrm{X} = \mathbf{O} \times R_{N\times M}$$

where X is a generated M-dimensional vector entity, $\mathbf{O} = (O_1, O_2, ..., O_N)$ is an original entity, and $R_{N\times M}$ is a random-weight vectors matrix. For example, if we want three features from N original features, we prepare three random-weight vectors with N random values in each of them. The random values are selected such that X represents the original features.

3.3.2 Neural network-based feature reduction techniques

We studied the following neural network-based feature reduction techniques.

- *Restricted Boltzmann Machine (RBM):* An RBM [70] automatically extracts important information from the original features as weights and biases on a two-layered neural network. Each node in the first-layer corresponds to an original feature, and each node in the second-layer corresponds to a new feature. We use the output of the second-layer as the new features.
- *Autoencoder (AE):* AE [30] and RBM are similar, but trained differently. In RBM, the network is trained based on a probability distribution. In AE, the network is trained using the difference between the original and the generated features.

3.4 Studied feature selection techniques

We studied the correlation-based (CFS) and consistency-based feature selection techniques (ConFS). These techniques were reported as the best-performing feature selection techniques in prior studies [18, 76]. Below, we give a brief overview of these techniques.

- Correlation-based feature selection (CFS) [22]: CFS selects a subset of features based on their *correlation*. The selected features have strong correlations with the class label (clean or defective), while having a low correlation with each other.
- Consistency-based feature selection (ConFS) [12]: ConFS uses the *consistency* of the class label across the entities instead of the correlation. For example, if file A has a feature set (10, 20, 40, defective) and file B has a feature set (10, 20, 30, clean), we can identify the defective and clean entities using the third feature. However, if a feature reduction technique removes the third feature, file A and file B have the same feature set except for the class label. In that case, these entities are inconsistent. Using this information, ConFS selects the best feature subset from the original features.

3.5 Area under the receiver operating characteristic curve (AUC)

We used the Area Under the receiver operating characteristic Curve (AUC) as the performance measure since AUC is not affected by the skewness of defect data [71, 78]. The receiver operating characteristic (ROC) curve is created by plotting the false positive rate (on the x-axis) and the true positive rate (on the y-axis) at various thresholds. In our experiment, the false positive rate is defined as the portion of clean entities that are identified as defective; the true positive is defined as the portion of defective entities that are identified as defective. The threshold is used to label an entity as clean or defective by checking whether its predicted probability is over the threshold. The AUC is the area under the ROC curve. The values of the AUC range between 0 and 1; a perfect classifier has an AUC of 1, while a random classifier has an AUC of 0.5.

3.6 Preprocessing

Most feature reduction techniques require the data to be preprocessed. We detail the preprocessing step below.

3.6.1 Preprocessing for traditional feature reduction techniques

The traditional feature reduction techniques require features that are normalized to a mean of 0 and a variance of 1 using the z-score [78]. The z-score is calculated as follows:

$$\mathbf{X}_z = \frac{\mathbf{X}_{org} - \mu}{\sigma} \tag{1}$$

where μ is a mean of the value of the feature for all entities and σ is the standard deviation of the value of the feature for all entities.

3.6.2 Preprocessing for neural network-based feature reduction techniques

The neural network-based feature reduction techniques require either binary features or features that are between 0 and 1. Hence, we scale the original features as follows:

$$\mathbf{X}_{scaled} = \frac{\mathbf{X}_{org} - X_{\min}}{X_{\max} - X_{\min}} \tag{2}$$

where $\mathbf{X}_{org}$ is a vector of the value of a particular feature for all entities. $X_{\min}$ is the smallest value of the feature and $X_{\max}$ is the largest value of the feature for all entities [1].

3.7 Out-of-sample bootstrap sampling

Bootstrap sampling is a validation technique that is used to estimate the performance of a model for unseen data. The technique is based on random sampling with replacement. *Out-of-sample bootstrap sampling* is a bootstrap sampling technique that estimates the future performance of a defect prediction model more accurately than a cross-validation scheme [71, 73]. Hence, we used the out-of-sample bootstrap sampling technique instead of a conventional validation technique such as 10-fold cross-validation. The process of the out-of-sample bootstrap sampling is as follows:

1. Sample N data points following the distribution of the original dataset, with N data points, while allowing for replacement.
2. Train a model using the sampled N data points, and test it using the data points that were not sampled.
3. Repeat steps 1 and 2 M times.
4. Report the average/median performance as the performance estimate.

We used the out-of-sample bootstrap sampling under the condition where $M = 100$ and we report the median performance.

4 Experimental setup

In this section, we give an overview of the setup of our experiments. The results are presented in Section 5. Figure 3 shows the steps of our experiments. We first conducted the out-of-sample bootstrap sampling on our studied datasets to generate and select features using each of the studied feature reduction and selection techniques. We then preprocessed the original features of each *bootstrap sample* as described in Section 3.6. We generated eight new feature sets (one for each feature reduction technique) for each bootstrap sample. Hence, we generated 800 new feature sets using feature reduction in total. Furthermore, the two studied feature selection techniques selected two feature subsets (one for each feature selection technique) for each bootstrap sample. Hence, we selected 200 feature subsets using feature selection in total.

The smallest number of features in the studied datasets is 20 (i.e., in the PROMISE dataset). Hence, to be able to observe the impact of a feature reduction technique, we configured each feature reduction technique to generate 10 features (H1–H10). However, PCA uses variance to decide on the number of generated features [11, 18]. Therefore, each bootstrap sample results in a different number of generated features using PCA. We configured PCA to retain 95% of the variance in the data [11, 18]. The median number of generated features by PCA in our experiments was 12 in the PROMISE dataset, 10 in the NASA dataset and 34 in the AEEEM dataset.

The experimental setup for each RQ is discussed in the next section.

5 Results

In this section, we present the results of our experiments. For each RQ, we discuss the motivation, approach and results.

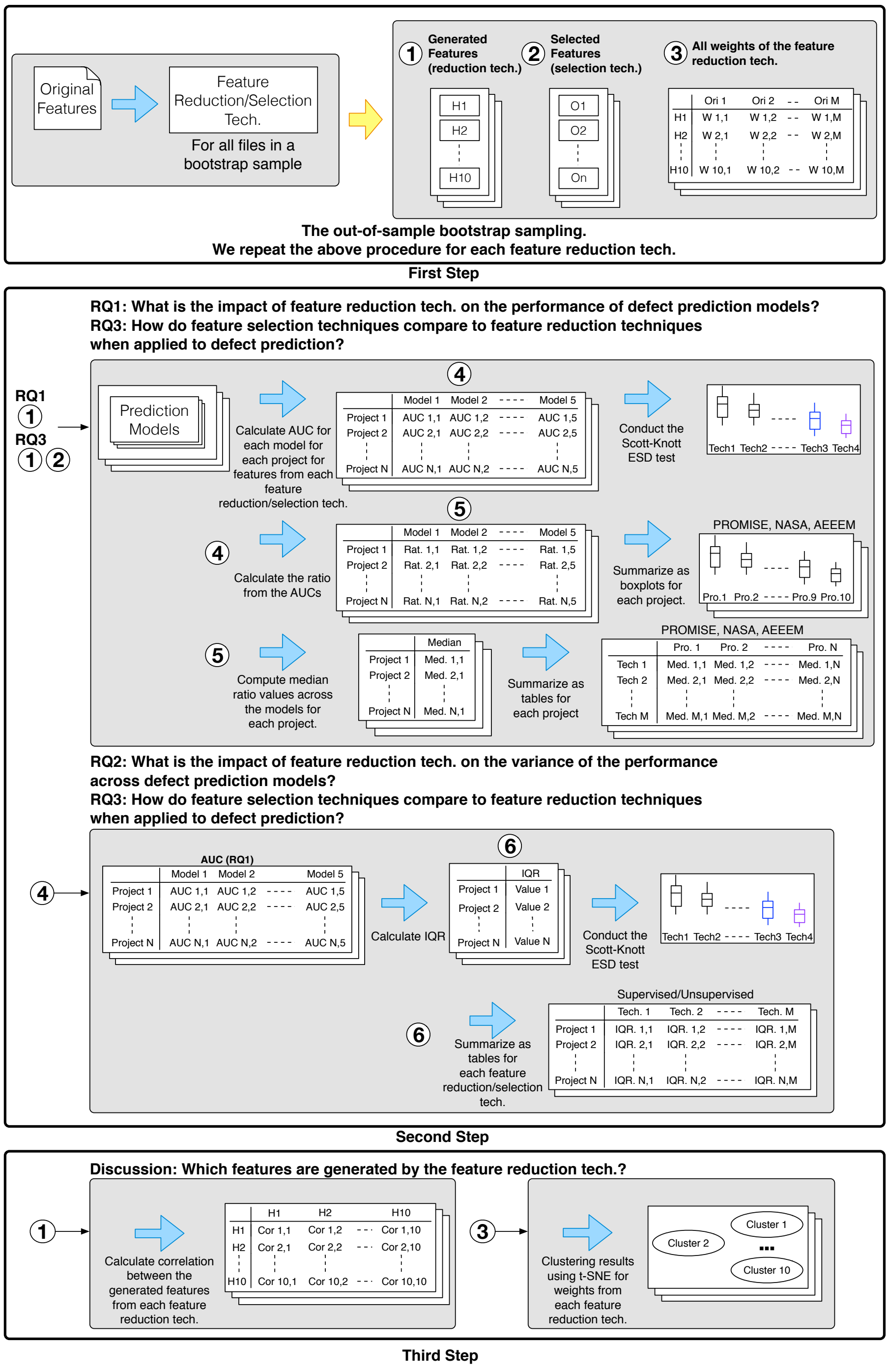

Original Features
Feature Reduction/Selection Tech.
For all files in a bootstrap sample
1 Generated Features (reduction tech.)
2 Selected Features (selection tech.)
3 All weights of the feature reduction tech.
H1 H2 H10
O1 O2 On
Ori 1 Ori 2 Ori M
W 1,1 W 1,2 W 1,M
W 2,1 W 2,2 W 2,M
W 10,1 W 10,2 W 10,M
The out-of-sample bootstrap sampling.
We repeat the above procedure for each feature reduction tech.
First Step
RQ1: What is the impact of feature reduction tech. on the performance of defect prediction models?
RQ3: How do feature selection techniques compare to feature reduction techniques when applied to defect prediction?
Prediction Models
Calculate AUC for each model for each project for features from each feature reduction/selection tech.
Calculate the ratio from the AUCs
Compute median ratio values across the models for each project.
Conduct the Scott-Knott ESD test
Summarize as boxplots for each project.
Summarize as tables for each project
PROMISE, NASA, AEEEM
Tech1 Tech2 Tech3 Tech4
Pro.1 Pro.2 Pro.9 Pro.10
RQ2: What is the impact of feature reduction tech. on the variance of the performance across defect prediction models?
RQ3: How do feature selection techniques compare to feature reduction techniques when applied to defect prediction?
AUC (RQ1)
Calculate IQR
Summarize as tables for each feature reduction/selection tech.
Supervised/Unsupervised
Second Step
Discussion: Which features are generated by the feature reduction tech.?
Calculate correlation between the generated features from each feature reduction tech.
Clustering results using t-SNE for weights from each feature reduction tech.
Cluster 1 Cluster 2 Cluster 10
Third Step

Fig. 3 Overview of our experimental design. We first generate/select 100 (the out-of-sample bootstrap) feature sets using each feature reduction/selection technique for each studied dataset. The second step is different for each RQ. We conduct correlation analysis and clustering analysis for discussion in the third step.

5.1 RQ1: **What is the impact of feature reduction techniques on the performance of defect prediction models?**

Motivation: Reducing the number of features that are used in a defect prediction model can be beneficial for addressing the curse of dimensionality and multicollinearity of the model. There exist two ways to reduce the number of features in a model: (1) by *selecting* the most important features, and (2) by *reducing* the number of features by creating new, combined features from the original features. Prior work has systematically studied the impact of feature selection techniques on defect prediction [18, 76], but no work has conducted a large-scale study of the impact of feature reduction techniques on defect prediction. Hence, in this RQ, we studied the impact of feature reduction techniques on the performance (AUC) of defect prediction models.

Approach: We used each feature set that was generated by a feature reduction technique as input to the studied 5 supervised and 5 unsupervised defect prediction models. We used the AUC as the performance measure. Because we calculated the AUC of a defect prediction model using the out-of-sample bootstrap sampling 100 times for each feature reduction technique, each model has 100 AUC values. Hence, we used the median value to represent the median performance of a defect prediction model using a certain feature reduction technique. Because we studied 26 projects, our experiments yielded 260 median AUC values for each feature reduction technique (5 supervised models*26 projects+5 unsupervised models*26 projects). For comparison, we also calculated the performance of the studied defect prediction models without applying a feature reduction technique (indicated as *ORG*). Note that we did within-project defect prediction in our experiments.

We used the Scott-Knott ESD test [73] (using a 95% significance level) to compare the median AUC values across feature reduction techniques. The Scott-Knott test is a hierarchical clustering algorithm that ranks the distributions of values. In particular, distributions that are not statistically significantly different are placed in the same rank. The Scott-Knott ESD test is an extension of the Scott-Knott test, which not only ranks based on significance, but also on Cohen's d effect size [10]. The Scott-Knott ESD test places distributions which are not significantly different, or have a negligible effect size, in the same rank. We used the `ScottKnottESD` R package[1] that was provided by Tantithamthavorn [72].

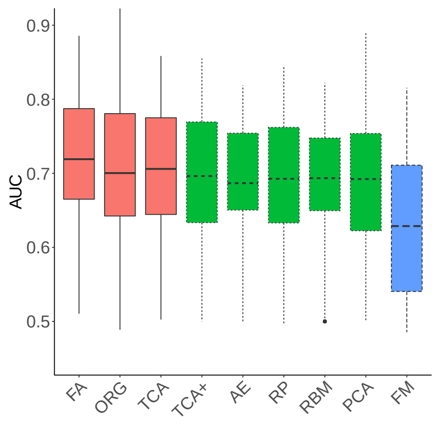
Project-level analysis: the aforementioned procedure combines the results of all projects. However, this procedure prevents us from understanding differences for each project. Hence, we also studied the performance at the project-level.

We compared the ratios of the AUCs (the median AUCs across all bootstrap samples) of each feature reduction technique to the original models. We calculated this ratio as follows:

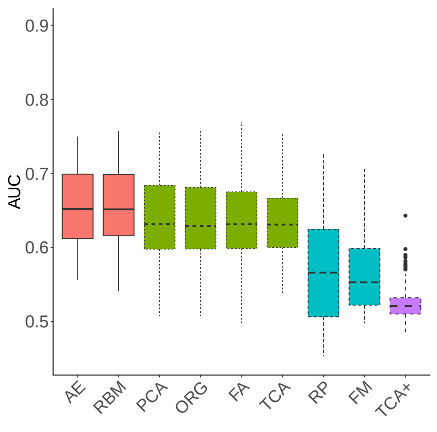
$$\text{ratio} = \frac{AUC_{FR}}{AUC_{ORG}}$$

Where AUC_{ORG} is the AUC of a prediction model using the original features, and AUC_{FR} is the AUC of a prediction model using the features that were generated by

[1] https://github.com/klainfo/ScottKnottESD



(a) The supervised models



(b) The unsupervised models

Fig. 4 The Scott-Knott ESD test results for the supervised (logistic regression, random forest, naive Bayes, decision tree, and logistic model tree) and the unsupervised (spectral clustering, *k*-means, partition around medoids, fuzzy C-means, neural-gas) models. Each color indicates a rank: models in different ranks have a statistically significant difference in performance. Each boxplot has 130 median AUC values (5 defect prediction models times 26 projects). The x-axis refers to the feature reduction techniques; the y-axis refers to the AUC values.

a particular feature reduction technique. Hence, a ratio larger than 1 indicates that the feature reduction technique improved the AUC compared to the original models, while a ratio smaller than 1 indicates that the feature reduction technique reduced the AUC. We computed the median ratio across the five studied supervised and unsupervised prediction models.

We used the aforementioned ratio to analyze performance at the project-level. The project-level analysis shows the impact of the different feature reduction techniques in every single project and dataset. Figure 6 shows the distributions of the ratios for each studied project for the supervised and unsupervised prediction models, respectively. Each boxplot contains 40 ratio values (5 prediction models * 8 feature reduction techniques). In addition, we show the median ratios for the best-performing feature reduction techniques as tables for deeper analysis (Table 5). These median ratios were computed from five AUC values (one for each studied prediction model).

Results: **FA and TCA can preserve the performance of the original defect prediction models, while at the same time reducing the number of features.** Figure 4(a) and Figure 4(b) show that the performance of the supervised and unsupervised defect prediction models does not decrease when applying FA or TCA. Hence, these feature reduction techniques can safely be applied to reap the benefits of a reduced number of features. In particular, FA and TCA work well for supervised models. Interestingly, the performance of the supervised and unsupervised defect prediction models is significantly lower when using TCA+ (which is an extension of TCA), compared to the original TCA.

The neural network-based feature reduction techniques (RBM and AE), significantly outperform traditional feature reduction techniques for the unsuper-

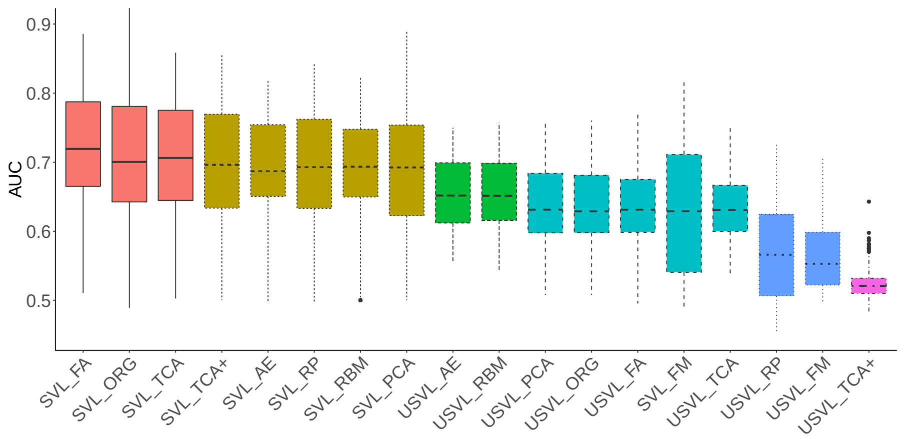


Fig. 5 The Scott-Knott ESD test results for both the supervised and unsupervised models. Each color indicates a rank: models in different ranks have a significant difference in performance. Each boxplot has 130 median AUC values (5 defect prediction models times 26 projects). The x-axis refers to the feature reduction techniques; the y-axis refers to the AUC values. In the x-axis, the "SVL_"-prefix refers to the 5 supervised defect prediction models; the "USVL_"-prefix refers to the 5 unsupervised defect prediction models.

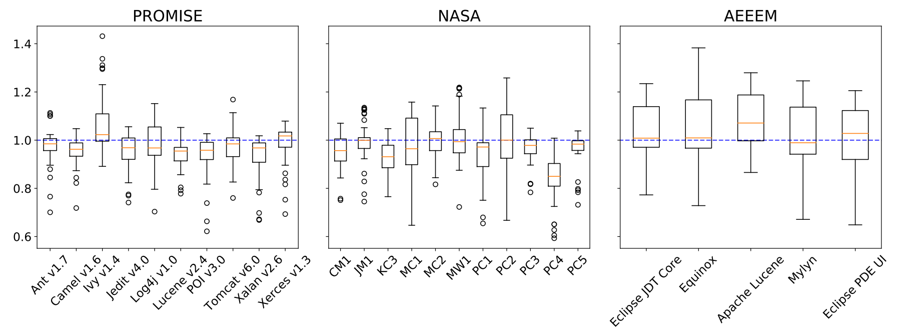
vised defect prediction models. Figure 4(b) shows the AUC values and the results of the Scott-Knott ESD test for the unsupervised models after applying the studied feature reduction techniques.

The highest rank contains only the two studied neural network-based techniques: RBM and AE. Hence, the neural network-based feature reduction techniques can significantly improve the AUC compared to the original models and other feature reduction techniques. However, these neural network-based feature reduction techniques do not outperform ORG for the supervised models. In Section 6 we further investigate why neural network-based feature reduction techniques work well for the unsupervised, but not for the supervised defect prediction models.

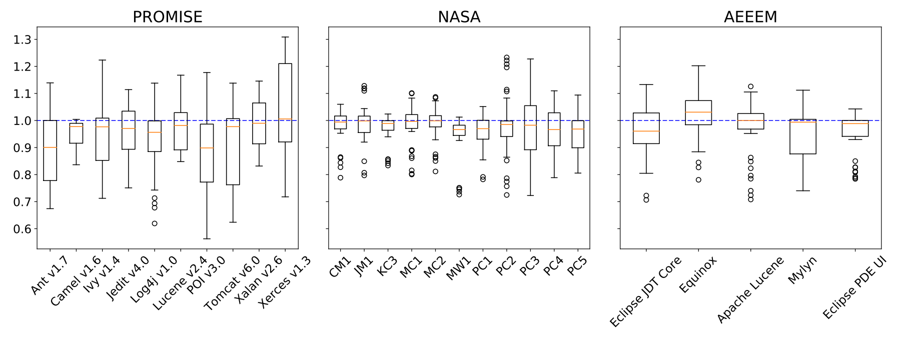
The supervised models with feature reduction techniques significantly outperform the unsupervised models with feature reduction techniques. Figure 5 shows the AUC after applying the feature reduction techniques to the supervised and unsupervised models. The supervised models significantly outperform the unsupervised models. Prior research [78] reported that spectral clustering (SC) is the only studied unsupervised defect prediction model that outperforms the supervised models.

The reason that the unsupervised models perform worse than the supervised models in Figure 5 is that we consider all the unsupervised models together, to be able to provide a more generic conclusion. However, as Figure 5 shows, some unsupervised defect prediction models perform better than others.

In the AEEEM dataset, the feature reduction techniques improve the prediction performance of the supervised models for most projects. We observe that the feature reduction techniques did not improve the prediction performance in many projects, as the median values of several boxplots in Figure 6 are lower than 1.0. How-



(a) The supervised models



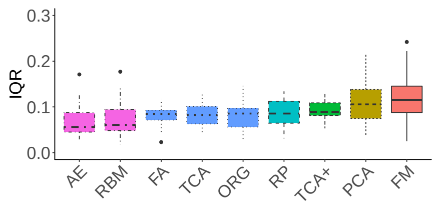
(b) The unsupervised models

Fig. 6 The ratios of the AUCs of the supervised and unsupervised prediction models. Each boxplot contains 40 ratio values (5 prediction models * 8 feature reduction techniques). The x-axis indicates the studied projects. The y-axis indicates the ratio. The dashed blue line indicates a ratio of 1.0. A ratio larger than 1.0 indicates that the feature reduction technique improved the AUC compared to the original models, while a ratio smaller than 1.0 indicates that the feature reduction technique reduced the AUC. A ratio of 1.0 (on the dashed blue line) indicates that the prediction model that uses features that were generated by a feature reduction technique had the same performance as the original model.

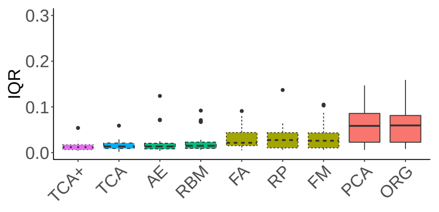
ever, the studied feature reduction techniques improved the prediction performance of the supervised models for many projects in the AEEEM dataset (Figure 6(a)). We further investigate this phenomenon in Section 5.3.1.

The neural network-based techniques improve the prediction performance of the supervised/unsupervised prediction models for most projects. Table 5 shows the median ratios for the feature reduction techniques. We observe that the neural network-based feature reduction techniques RBM and AE have the most gray cells for the supervised/unsupervised prediction models in combination with the feature reduction techniques.

However, almost all feature reduction techniques did not improve the prediction performance in the NASA dataset except for FA with the unsupervised prediction models. FA combined with the unsupervised prediction models improved over half of the projects in the NASA dataset. We further investigate why the feature reduc-



(a) The supervised models



(b) The unsupervised models

Fig. 7 The Scott-Knott ESD test results for the IQR of the supervised and unsupervised models. Each color indicates a rank: feature reduction techniques in different ranks have a significant difference in variance (IQR). Each boxplot has 26 IQR values (one for each project). The x-axis refers to the feature reduction techniques; the y-axis refers to the IQR values.

tion techniques work well for the AEEEM dataset but not for the other datasets in Section 5.3.1.

5.2 RQ2: **What is the impact of feature reduction techniques on the variance of the performance across defect prediction models?**

Motivation: A challenge in applying defect prediction for practitioners is to select the best-performing model for their data from many possible defect prediction models [17, 19]. In this RQ, we studied the variance in performance across defect prediction models of the studied feature reduction techniques for a particular dataset. If this variance is small, the practitioner does not need to worry about the choice at all, as the models perform similarly across datasets.

Approach: We used the interquartile range (IQR) which captures the performance variance across the studied defect prediction models for a given project and feature reduction technique. We used the AUC values of all the studied supervised and unsupervised models for all bootstrap samples to conduct a new *bootstrap sampling* to calculate the IQR for each reduction technique and each project. We calculated an IQR value as follows:

1. Sample 100 AUC values at random from the 100 AUC values for each studied supervised/unsupervised model allowing for replacement.
2. Compute the median AUC value across the sampled 100 AUC values.
3. Repeat steps 1 and 2 100 times.
4. Compute the IQR value for the 500 sampled median AUC values (5 supervised/unsupervised prediction models * 100 median AUC values) for each feature reduction/selection technique for each studied project.

Where the IQR values are computed as follows:

$$\text{IQR} = Q_3 - Q_1$$

where Q_1 is the first quartile of the 500 sampled median AUC values, and Q_3 is the third quartile of the 500 sampled median AUC values. The first quartile is the median

Table 5 The median AUC ratios of the feature reduction techniques. The table also contains the median ratios of the CFS and ConFS feature selection techniques, which are studied in RQ3. A ratio larger than 1 indicates that the feature reduction/selection technique improved the AUC compared to the original models, while a ratio smaller than 1 indicates that the feature reduction/selection technique reduced the AUC. The gray cells refer to the ratios that are greater than 1.0. The "Improved" row indicates the number of projects for which a feature reduction/selection technique improved the performance.

(a) The supervised models

		RBM	AE	PCA	FM	FA	RP	TCA	TCA+	CFS	ConFS
PROMISE	Ant v1.7	1.015	1.004	0.983	0.896	0.995	0.956	0.962	0.962	1.012	1.006
	Camel v1.6	0.973	0.979	0.921	0.880	0.996	0.962	0.948	0.948	0.962	0.956
	Ivy v1.4	1.065	1.081	1.005	1.005	1.026	0.999	1.036	1.020	1.062	1.037
	Jedit v4.0	1.012	0.996	0.971	0.775	1.011	0.921	0.952	0.947	0.984	0.996
	Log4j v1.0	1.047	1.034	1.000	0.960	1.012	0.895	0.961	0.954	0.993	1.002
	Lucene v2.4	1.016	0.956	0.940	0.804	0.978	0.886	0.962	0.955	0.990	0.978
	POI v3.0	0.932	0.899	0.955	0.739	0.993	0.946	0.965	0.971	1.016	1.003
	Tomcat v6.0	1.014	1.016	0.947	0.875	0.997	0.925	0.985	0.990	1.045	1.026
	Xalan v2.6	0.861	0.912	0.987	0.698	0.992	0.970	0.993	0.985	1.012	0.998
	Xerces v1.3	1.032	1.022	0.969	0.817	1.019	1.017	1.034	1.033	1.005	1.012
NASA	CM1	1.004	0.979	0.939	0.984	0.974	0.995	0.949	0.940	1.028	0.981
	JM1	1.004	1.001	0.997	0.923	0.999	1.003	1.008	0.956	1.000	1.001
	KC3	0.910	0.944	0.901	0.923	1.010	0.924	0.905	0.874	1.025	0.985
	MC1	0.956	1.021	0.932	0.901	0.999	0.915	0.980	0.932	1.001	0.973
	MC2	1.019	1.014	0.955	0.947	1.037	1.014	0.976	0.972	0.973	0.968
	MW1	1.005	1.016	0.949	0.984	1.019	0.991	0.952	0.963	1.016	1.015
	PC1	0.906	0.830	0.936	0.972	1.013	0.976	0.971	0.925	1.004	1.021
	PC2	1.039	1.019	0.931	0.994	1.020	0.999	0.935	0.933	1.169	1.028
	PC3	0.944	0.971	0.985	0.921	0.997	0.988	1.001	0.968	1.026	0.995
	PC4	0.793	0.806	0.931	0.651	0.906	0.862	0.866	0.870	1.003	0.986
	PC5	0.999	0.982	0.994	0.798	0.990	0.977	0.989	0.986	0.986	0.997
AEEEM	Eclipse JDT Core	1.028	1.010	1.019	0.874	1.007	0.959	1.015	0.986	0.993	1.012
	Equinox	1.074	1.043	1.006	0.973	1.028	0.996	1.000	1.010	1.060	1.024
	Apache Lucene	1.112	1.079	1.090	1.045	1.009	1.063	1.030	1.039	1.021	1.035
	Mylyn	1.089	1.057	0.980	0.921	1.052	0.951	1.066	1.065	1.037	1.023
	Eclipse PDE UI	1.071	1.058	1.075	0.861	1.035	0.944	1.020	0.905	0.995	1.006
	Improved	17	15	5	2	14	4	8	5	17	15

(b) The unsupervised models

		RBM	AE	PCA	FM	FA	RP	TCA	TCA+	CFS	ConFS
PROMISE	Ant v1.7	1.005	1.000	1.001	0.739	0.896	0.695	0.918	0.748	1.002	1.006
	Camel v1.6	0.989	0.980	1.002	0.894	0.979	0.848	0.984	0.933	0.977	0.988
	Ivy v1.4	1.040	1.011	1.006	0.938	0.850	0.750	0.991	0.874	0.937	0.928
	Jedit v4.0	1.055	1.038	1.000	0.895	1.033	0.888	0.966	0.802	0.939	0.962
	Log4j v1.0	0.993	0.989	1.001	0.950	0.973	0.694	0.924	0.790	0.954	0.986
	Lucene v2.4	1.110	1.131	1.001	0.870	0.989	0.894	0.966	0.858	0.984	1.004
	POI v3.0	0.919	0.924	1.002	0.755	1.036	0.875	0.610	0.755	0.962	0.982
	Tomcat v6.0	1.041	1.007	0.997	0.762	0.956	0.652	1.007	0.763	1.010	0.968
	Xalan v2.6	1.070	1.065	0.997	0.885	0.936	0.965	1.084	0.848	0.964	0.974
	Xerces v1.3	1.244	1.210	1.008	0.885	1.259	0.893	1.201	0.939	1.106	1.017
NASA	CM1	1.008	0.996	0.997	0.967	0.979	1.049	0.983	0.844	1.030	0.949
	JM1	0.992	1.023	1.002	0.953	1.036	0.997	1.011	0.849	1.008	1.001
	KC3	0.970	0.987	1.000	0.974	1.014	0.997	0.998	0.850	0.968	0.955
	MC1	1.005	1.023	1.000	0.891	1.021	1.000	0.994	0.816	1.081	1.023
	MC2	1.073	1.043	1.000	0.981	1.018	0.985	0.995	0.862	0.993	0.991
	MW1	0.939	0.971	1.000	0.973	0.966	0.990	0.957	0.746	1.017	1.019
	PC1	0.990	0.996	0.996	0.927	1.027	0.935	0.993	0.855	1.102	1.041
	PC2	0.985	0.976	0.994	0.975	0.985	0.993	0.966	0.774	1.035	0.940
	PC3	1.074	1.090	1.000	0.940	0.954	0.758	1.041	0.831	1.213	1.123
	PC4	0.986	0.982	0.998	0.902	1.062	0.829	0.961	0.818	1.123	0.991
	PC5	0.971	0.975	0.999	0.866	0.971	0.987	0.932	0.857	1.030	0.997
AEEEM	Eclipse JDT Core	1.121	1.089	0.997	0.888	0.957	0.995	1.107	0.805	1.017	1.007
	Equinox	1.029	1.105	1.000	0.913	1.069	1.033	1.073	0.900	1.025	1.033
	Apache Lucene	1.031	1.053	1.000	0.863	0.989	1.006	1.024	0.740	0.956	0.981
	Mylyn	1.006	1.015	1.001	0.829	0.996	0.892	0.994	0.827	1.009	0.996
	Eclipse PDE UI	1.015	1.024	1.000	0.827	0.980	0.977	0.993	0.793	1.003	1.024
	Improved	16	15	9	0	10	3	8	0	16	11

between the smallest and the median of the 500 AUC values, and the third quartile is the median between the median and the largest of the 500 AUC values. As we studied 26 projects, we have 26 IQR values for each feature reduction technique. We used the Scott-Knott ESD test to compare the distributions of IQRs for each feature reduction technique. Figure 7 shows the results of the Scott-Knott ESD test.

In addition, we compared the IQR values of the prediction models across the feature reduction techniques for each project. Table 6 shows the results of the IQR analysis at the project level. Each cell contains an IQR value that was computed from 500 bootstrapped median AUC values of the supervised and unsupervised prediction models.

Results: **The neural network-based feature reduction techniques, RBM and AE, generate features that result in less variance across the supervised models than the original features.** Figure 7(a) shows that the original features (ORG) are in the second rank, and RBM and AE belong to the first rank. Hence, RBM and AE significantly improve the variance of the performance across the supervised defect prediction models.

Almost all feature reduction techniques (except PCA) generate features that have a significantly smaller performance variance across the unsupervised models than the original features. Figure 7(b) shows that the unsupervised models that use features that were generated by PCA, or the original features are in the lowest rank. Hence, using the feature reduction techniques (except PCA) in combination with an unsupervised defect prediction model results in a small performance variance, which is helpful for practitioners.

The neural network-based feature reduction techniques improve the performance variance of the supervised models for the largest number of projects. Table 6(a) shows that the features that were generated by the neural network-based feature reduction techniques (RBM and AE) improved the performance variance (IQR) across the studied supervised prediction models for the largest number of projects compared to the other feature reduction techniques, and the original models. RBM and AE also belong to the first rank of the overall performance variance result (Figure 7(a)).

The neural network-based feature reduction techniques improve the performance variance of the unsupervised models for the largest number of projects. Table 6(b) shows that the features that were generated by the neural network-based feature reduction techniques (RBM and AE) improved the performance variance across the studied unsupervised prediction models for the largest number of projects. Interestingly, in terms of overall performance variance, TCA and TCA+ belong to the first and the second rank (Figure 7(b)). However, the difference with the neural network-based feature reduction techniques is only small (TCA+) and negligible (TCA), according to the Cliff's delta effect size.

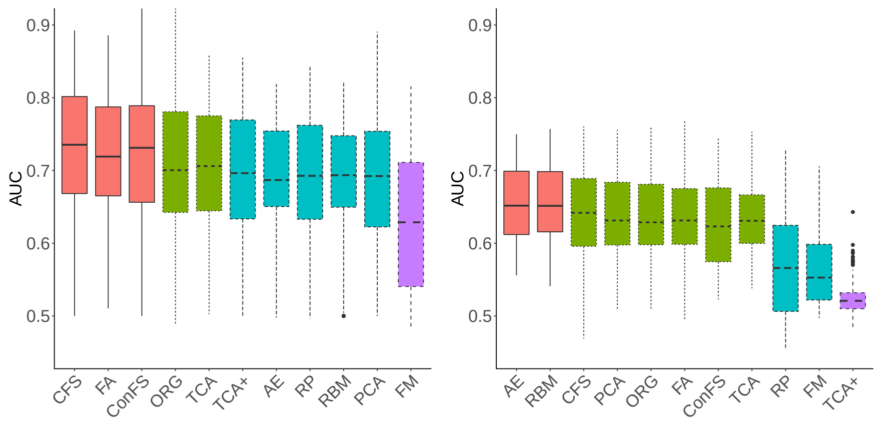
Table 6 The IQR ratio for the feature reduction techniques in the studied supervised and unsupervised prediction models. The gray cells refer to the ratios that are greater than 1.0. The "Improved" row indicates the number of gray cells in the column.

(a) The supervised models

		RBM	AE	PCA	FM	FA	RP	TCA	TCA+	CFS	ConFS
PROMISE	Ant v1.7	1.769	1.330	0.849	0.439	1.024	0.943	0.975	1.006	1.294	1.031
	Camel v1.6	3.885	3.027	0.842	0.809	0.774	0.777	1.312	1.357	1.637	1.466
	Ivy v1.4	1.841	2.964	1.978	2.197	2.001	1.151	3.641	1.233	1.340	2.072
	Jedit v4.0	1.018	1.180	0.562	0.501	0.771	0.595	0.688	0.718	0.767	0.683
	Log4j v1.0	1.335	2.643	1.030	5.212	1.388	1.898	1.252	1.189	1.361	1.377
	Lucene v2.4	1.737	0.661	0.554	0.473	0.635	1.141	0.666	0.647	0.800	0.658
	POI v3.0	0.973	0.586	0.628	0.241	0.959	1.642	0.894	0.977	1.268	1.157
	Tomcat v6.0	1.351	1.429	0.671	0.657	1.096	0.739	0.854	0.921	1.369	1.204
	Xalan v2.6	0.762	0.615	0.863	0.495	0.725	0.750	0.531	0.556	0.972	0.841
	Xerces v1.3	1.918	1.749	0.806	1.022	0.847	0.943	1.064	1.078	1.171	1.417
NASA	CM1	0.196	0.227	0.241	0.311	0.329	0.683	0.389	0.430	0.589	0.453
	JM1	2.502	2.579	1.886	0.604	0.804	0.972	1.036	0.758	0.892	0.986
	KC3	0.582	0.955	1.509	1.130	0.661	1.093	0.906	0.751	1.871	1.710
	MC1	0.639	0.670	0.605	0.734	0.980	0.810	1.227	0.967	0.831	2.011
	MC2	1.087	1.080	0.545	1.074	0.808	0.835	0.734	0.679	1.121	1.169
	MW1	1.199	1.451	0.722	4.598	2.933	1.166	2.113	1.697	1.792	1.274
	PC1	0.826	0.984	0.649	1.078	1.326	1.936	2.501	0.875	1.179	1.578
	PC2	0.541	0.565	0.454	0.788	1.067	1.317	0.937	0.898	0.796	0.389
	PC3	1.060	1.522	0.684	0.985	1.274	0.718	0.862	0.695	1.331	1.145
	PC4	0.885	1.504	0.548	0.397	0.852	1.055	0.681	0.750	5.539	1.138
	PC5	0.796	1.892	0.752	0.344	0.908	0.705	0.723	0.757	0.798	1.009
AEEEM	Eclipse JDT Core	1.332	1.433	1.246	0.434	1.556	0.998	1.373	0.970	1.072	1.393
	Equinox	1.989	2.927	1.543	0.821	1.076	0.822	1.075	1.129	1.438	1.337
	Apache Lucene	1.373	1.844	0.936	0.608	0.989	1.678	1.014	0.993	0.881	1.022
	Mylyn	5.981	4.024	0.940	0.946	5.923	1.155	1.289	1.282	1.992	2.406
	Eclipse PDE UI	1.574	1.208	1.096	0.224	0.392	0.374	0.434	0.384	0.475	0.452
	Improved	17	18	7	7	11	11	12	8	16	19

(b) The unsupervised models

		RBM	AE	PCA	FM	FA	RP	TCA	TCA+	CFS	ConFS
PROMISE	Ant v1.7	2.693	1.772	1.035	2.539	3.228	3.217	1.447	4.320	0.335	0.245
	Camel v1.6	1.224	1.715	1.331	0.961	1.659	0.778	0.503	0.680	0.209	0.301
	Ivy v1.4	2.881	4.073	0.924	2.096	3.690	3.628	3.317	3.076	2.809	2.153
	Jedit v4.0	3.221	3.229	1.012	1.704	4.387	2.213	3.346	4.496	0.671	0.790
	Log4j v1.0	3.400	3.445	1.221	1.346	0.542	0.977	1.562	0.783	0.762	0.793
	Lucene v2.4	0.460	0.982	0.653	0.796	0.577	1.865	0.945	0.560	0.176	0.163
	POI v3.0	0.764	0.582	1.152	7.759	1.425	7.589	4.332	2.842	0.415	0.927
	Tomcat v6.0	2.993	1.985	1.116	2.326	3.566	4.079	1.512	5.814	0.630	0.904
	Xalan v2.6	2.028	3.317	0.763	2.339	1.363	3.747	0.829	0.907	0.362	2.860
	Xerces v1.3	8.508	6.308	1.132	11.297	6.729	10.673	2.089	10.157	4.720	0.980
NASA	CM1	2.032	2.820	1.074	0.694	1.274	1.535	1.573	3.922	1.238	0.688
	JM1	1.206	5.044	0.988	0.983	1.360	1.304	39.761	154.852	1.210	1.009
	KC3	0.920	0.753	0.936	1.280	0.639	0.780	0.946	3.250	0.591	0.750
	MC1	5.284	3.674	1.035	0.576	3.833	1.261	5.611	3.635	1.132	0.971
	MC2	1.377	2.056	1.839	0.778	1.670	1.919	2.709	4.491	1.457	1.441
	MW1	2.059	1.207	0.952	1.043	0.897	1.008	1.325	1.520	0.814	0.816
	PC1	2.413	2.298	0.996	1.047	1.405	0.866	3.256	3.491	0.964	0.748
	PC2	2.058	1.900	1.014	2.441	5.610	1.035	5.055	7.867	2.231	1.436
	PC3	2.293	2.230	1.069	2.318	4.411	7.025	13.146	13.203	2.622	1.182
	PC4	4.032	3.431	0.917	4.188	1.266	3.476	9.617	10.161	1.653	1.177
	PC5	5.900	11.856	1.025	18.438	4.839	2.096	30.503	5.129	1.355	1.267
AEEEM	Eclipse JDT Core	15.497	15.262	0.973	1.727	1.290	3.088	20.680	8.202	1.104	1.505
	Equinox	3.504	7.741	0.684	3.573	1.155	1.420	6.460	9.422	0.676	0.601
	Apache Lucene	8.221	6.507	0.989	0.521	4.423	1.610	5.717	8.040	0.659	0.920
	Mylyn	10.940	9.597	1.038	8.177	1.099	1.036	17.600	11.010	0.980	0.889
	Eclipse PDE UI	3.237	2.806	1.068	0.690	1.235	0.465	5.627	6.956	0.389	0.366
	Improved	23	23	15	18	22	21	22	22	11	9



(a) The supervised models (b) The unsupervised models

Fig. 8 The Scott-Knott ESD test results for the supervised (logistic regression, random forest, naive Bayes, decision tree, and logistic model tree) and unsupervised (spectral clustering, k-means, partition around medoids, fuzzy C-means, neural-gas) models. Each color indicates a rank: models in different ranks have a statistically significant difference in performance. Each boxplot has 130 median AUC values (5 defect prediction models times 26 projects). The x-axis refers to the feature reduction/selection techniques; the y-axis refers to the AUC values.

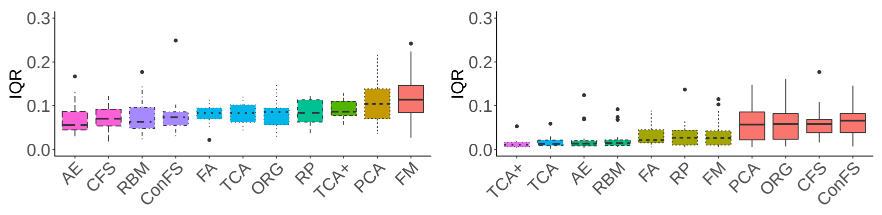
5.3 RQ3: **How do feature selection techniques compare to feature reduction techniques when applied to defect prediction?**

In this RQ, we compare feature reduction and selection techniques along two dimensions: the performance and the performance variance of the defect prediction models. We study the correlation-based (CFS) and consistency-based (ConFS) feature selection techniques, as they performed best according to prior studies [18, 76].

Motivation: In RQ1 and RQ2, we found that several feature reduction techniques (FA, RBM and AE) outperform the original features (ORG) in terms of performance or performance variance of the defect prediction models. Prior work [18, 76] showed that several feature selection techniques outperform the original models as well. In this RQ, we compare the performance (AUC) and the performance variance (IQR) of the feature reduction and selection techniques when applied to defect prediction models.

Approach: The experimental procedure is the same as the procedures of RQ1 and RQ2 (only we use the two feature selection techniques CFS and ConFS instead of the feature reduction techniques).

Results: **The feature selection techniques (correlation-based feature selection (CFS) and consistency-based feature selection technique (ConFS)) significantly outperform the original features (ORG) in the supervised models, and perform as well as the feature agglomeration (FA) reduction technique.** Figure 8(a) shows the AUC values and the results of the Scott-Knott ESD test for the supervised models



(a) The supervised models (b) The unsupervised models

Fig. 9 The Scott-Knott ESD test results for IQR in the supervised and unsupervised models. Each color indicates a rank: feature reduction/selection techniques in different ranks have a significant difference in variance (IQR). Each boxplot has 26 IQR values (one for each project). The x-axis refers to the feature reduction/selection techniques; the y-axis refers to the IQR values.

after applying the studied feature reduction and selection techniques.[2] Each boxplot shows the median AUC values for the projects using a certain feature reduction/selection technique. CFS, ConFS and FA are in the highest rank by themselves, which indicates that the subsets of features that were selected by CFS or ConFS perform as well as the feature sets that were generated by FA for the supervised models.

The neural network-based feature reduction techniques (RBM and AE) significantly outperform the feature selection techniques (CFS and ConFS) for the unsupervised defect prediction models. The highest rank contains only the two studied neural network-based feature reduction techniques (Figure 8(b)). The studied feature selection techniques (CFS and ConFS) belong to the second rank, together with the original models (ORG). Hence, the studied feature selection techniques have a worse performance than the neural network-based feature reduction techniques for the unsupervised defect prediction models.

In the supervised models, applying the neural network-based feature reduction techniques, RBM and AE, or the feature selection techniques, CFS and ConFS, significantly outperforms the original models in terms of performance variance. The original models (ORG) belong to the third rank (Figure 9(a)). The neural network-based feature reduction techniques and the feature selection techniques belong to the first or second rank, hence they have a smaller performance variance than the original models.

In the unsupervised models, all feature reduction techniques (except PCA) significantly outperform the feature selection techniques in terms of performance variance. The feature selection techniques belong to the worst rank together with the original models (ORG) and PCA (Figure 9(b)). Hence, the studied feature selection had a larger performance variance than almost all the studied feature reduction techniques for the unsupervised defect prediction models.

Our above findings for RQ3 are confirmed by our project-level analysis. Table 5 shows the median ratios of the performance of each feature reduction/selection

[2] Note that the ranks are slightly different from Figure 4 due to the fact that Scott-Knott ESD is a clustering algorithm, and hence affected by the total set of input distributions. For more information see https://github.com/klainfo/ScottKnottESD.

technique compared to the original models. We calculated this ratio as follows:

$$\text{ratio} = \frac{AUC_{FRS}}{AUC_{ORG}}$$

Where AUC_{ORG} is the AUC (the median AUC across all bootstrap samples) of a prediction model using the original features, and AUC_{FRS} is the AUC of a prediction model using the features that were generated/selected by a particular feature reduction or selection technique. We computed the median ratio across the five studied supervised and unsupervised prediction models. Table 5 confirms our above findings about the performance of the studied feature selection techniques compared to that of the feature reduction techniques.

Table 6 shows the IQR ratio values of each feature reduction/selection technique. We define this ratio as follows:

$$\text{ratio} = \frac{IQR_{FRS}}{IQR_{ORG}}$$

Where IQR_{ORG} is the IQR (the median IQR across all bootstrap samples) of a prediction model using the original features, and IQR_{FRS} is the IQR of a prediction model using the features that were generated/selected by a particular feature reduction or selection technique. We calculated the median IQR value for the supervised models using bootstrap samples as follows:

1. Sample 100 values following the distribution of the 100 AUC values for each studied supervised model while allowing for replacement.
2. Compute the median AUC value across the sampled 100 values.
3. Repeat steps 1 and 2 100 times.
4. Compute the IQR value for the 500 sampled median AUC values (5 supervised prediction models * 100 median AUC values) for each feature reduction/selection technique for each studied project.

We repeated the above procedure for the unsupervised models. Table 6(a) shows that the project-level results confirm our findings above, as the RBM and AE feature reduction techniques and the CFS and ConFS feature selection techniques improve the performance variance of most projects compared to the other techniques. In addition, Table 6(b) shows that all feature reduction techniques improve the performance variance of more projects than the CFS and ConFS feature selection techniques.

5.3.1 Why do feature reduction techniques work well in the AEEEM dataset?

Motivation: We observed that the feature reduction techniques work better for the projects in the AEEEM dataset than for the projects in the other datasets. Ghotra et al. [18] applied PCA to the data of each project to capture its richness. We use the same analysis to investigate whether the dataset richness is an explanation of why feature reduction techniques work better for the AEEEM dataset.

Approach: The idea behind Ghotra et al.'s analysis [18] is to generate features from a dataset using PCA that (together) retain at least 95% of the variance of the

Table 7 The number of generated features (principal components) that are needed to account for 95% of the data variance.

Studied Dataset	Studied Project	# of Studied Features	# of Generated Features	% of Generated Features
PROMISE	Ant v1.7	20	12	60.0
	Camel v1.6	20	12	60.0
	Ivy v1.4	20	10	50.0
	Jedit v4.0	20	12	60.0
	Log4j v1.0	20	12	60.0
	Lucene v2.4	20	12	60.0
	POI v3.0	20	12	60.0
	Tomcat v6.0	20	12	60.0
	Xalan v2.6	20	12	60.0
	Xerces v1.3	20	12	60.0
NASA	CM1	37	11	29.7
	JM1	21	8	38.1
	KC3	39	10	25.6
	MC1	38	15	39.5
	MC2	39	11	28.2
	MW1	37	11	29.7
	PC1	37	12	32.4
	PC2	36	10	27.8
	PC3	37	13	35.1
	PC4	37	14	37.8
	PC5	38	15	39.5
AEEEM	Eclipse JDT Core	212	36	17.0
	Equinox	212	31	14.6
	Apache Lucene	212	33	15.6
	Mylyn	212	46	21.7
	Eclipse PDE UI	212	38	17.9

original dataset. Ghotra et al. reason that a larger number of generated features indicates a richer dataset. Likewise, they interpret that a small number of generated features indicates redundancy in the original dataset. We applied PCA to each project and counted the number of generated features.

Results: **The PROMISE, NASA, and AEEEM datasets have different data richness characteristics, however; the characteristics of the projects within each dataset are consistent.** Table 7 shows the number of generated features. While the number of generated features is approximately the same for the PROMISE and NASA projects, the proportion of generated features compared to the number of original features is different. In addition, this proportion is even lower for the AEEEM projects. Hence, we conclude that the datasets have different characteristics in terms of data richness. However, within each dataset, the projects have approximately the same richness characteristics.

The original features of the projects in the AEEEM dataset are more diverse than the projects of the other datasets. We observe that 36 principal components are needed to cover 95% of the variance of the Eclipse JDT Core project in Figure 10, compared to 12 components for the Ant project and 11 for the CM1 project. Hence, the original features of the AEEEM dataset are much more diverse than those of the

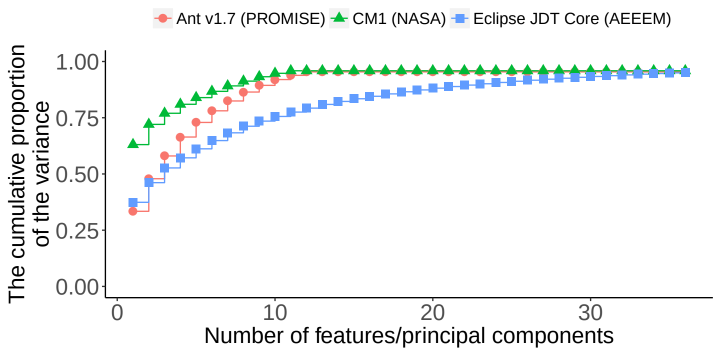


Fig. 10 The number of principal components (features that were generated by PCA) that are needed to account for the original data variance for the Ant, CM1 and Eclipse projects in the PROMISE, NASA and AEEEM datasets. The x-axis indicates the number of principal components. The y-axis indicates the cumulative proportion of the variance. The other projects of the datasets showed a similar pattern.

PROMISE and the NASA datasets. The diversity of the AEEEM dataset could be a reason why feature reduction techniques improve the performance of this dataset.

5.3.2 Comparing feature selection and reduction techniques along the dimensions of understandability and execution time

The understandability of the metrics in a defect prediction model is important, as understandable metrics make the model, and its predictions, easier to explain [68]. Feature reduction techniques combine the original features into one or more newly-generated features. Hence, these newly-generated features are by definition harder to understand than the features that are a subset (i.e., they were *selected*) of the original features. We inspected the feature sets that were generated during our experiments, and we observed that almost all generated feature sets consist of features that are a complex combination of all available original features. Hence, defect prediction models that are generated using feature reduction techniques are harder to understand than those that use feature selection.

In addition, the execution time of a feature reduction/selection technique and a defect prediction model is important – models that take too long to build or execute are not very useful in practice. To conduct our experiments in a timely manner, we ran them on a cluster of servers in parallel. Hence, it is difficult to compare the execution time of the experiments. In general, the execution time of our feature reduction/selection techniques and defect prediction models was short (i.e., in the range of minutes). Therefore, execution time is not a very problematic metric for practitioners who wish to apply feature reduction or selection to their defect prediction models.

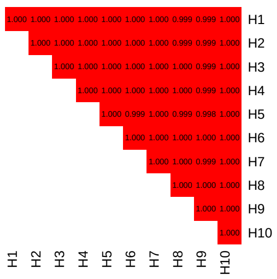
6 Discussion: Which features are generated by the feature reduction techniques?

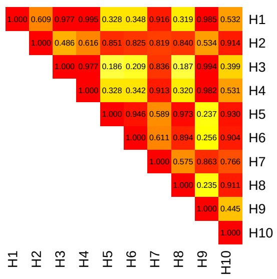
In RQ1, RQ2, and RQ3, we observed that some feature reduction/selection techniques generate/select features that perform defect prediction better and less variance than the features that were generated/selected by other feature reduction/selection techniques. In particular, we found that RBM and AE outperform the other studied feature reduction/selection techniques for the unsupervised models. However, RBM and AE are less-performing feature reduction techniques than the original models (ORG) for the supervised models. In this discussion, we take a closer look at the generated features to investigate why neural network-based feature reduction techniques perform well for the unsupervised defect prediction models, but not for the supervised models. In this section, we discuss possible explanations for the differences in AUC and variance of the performance.

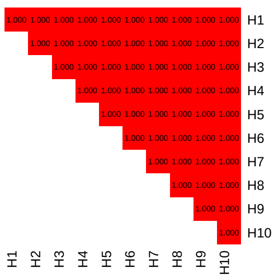
Approach: We focused our discussion on the RBM and AE feature reduction techniques, as these techniques generate new features by assigning (combinations of) weights to the original features. For example, a newly generated feature may be generated by 0.5 times original metric 1 and 0.5 times original metric 2. These weight sets allow us to study how the new features are related to the original features, and to the features that were generated by other feature reduction techniques, and extract possible explanations for the improved and small variance of the performance. RP and PCA also generate new features by assigning weights to the original features, however, the weights of RP are randomly generated, and PCA generates a different number of features for each project, which makes them difficult to compare. Hence, we focused our discussion on RBM and AE. As we generated 100 new feature sets of 10 features using RBM and AE, we generated 1,000 weight sets using these two feature reduction techniques for each project. For each feature reduction technique, we randomly selected 10 (out of the 100) generated feature sets for our investigation.

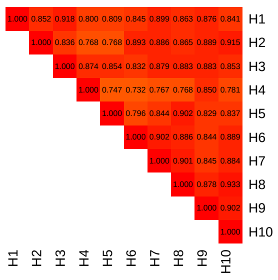
We conducted correlation analysis and clustering analysis on the studied feature sets to study their similarity within and across projects. The correlation analysis shows how independent the features that are generated for a project are. Highly correlated features can negatively impact the performance of regression models [15] and this effect can affect our supervised models as well. We first calculated the Spearman rank correlation [82] between the generated features in a feature set within a project. Each generated feature was normalized using the z-score. We chose Spearman rank correlation because it is non-parametric, and therefore requires no assumption about the distribution of the studied data.

To study the similarity of the generated feature sets across projects within a dataset, we compared the weight sets of the generated features. First, we normalized all weights using z-score normalization. Second, we used k-means to cluster the weight sets of the features, and then we used t-distributed stochastic neighbour embedding (t-SNE) [41] to visualize the clustering results. t-SNE is commonly used for visualizing high dimensional features in scatter plots [40]. In particular, t-SNE models high-dimensional objects (i.e., feature sets) by two- or three-dimensional points such that similar objects are close, and dissimilar objects are further away from each other.


(a) RBM-generated features in the Ant project in the PROMISE dataset.


(b) RBM-generated features in the Eclipse project in the AEEEM dataset.


(c) RBM-generated features in the the Eclipse project using change features in the AEEEM dataset.


(d) AE-generated features in the Log4j project in the PROMISE dataset.

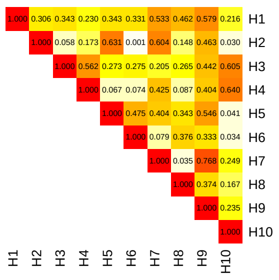
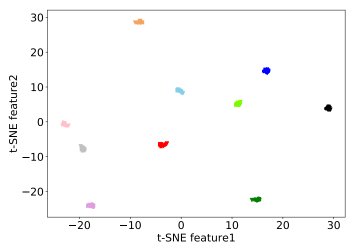

(e) AE-generated features in the Camel project in the PROMISE dataset.

Fig. 11 The Spearman rank correlation of generated features in the studied datasets. The darker colours indicate a strong absolute correlation (close or equal to one). The lighter colours indicate a weak absolute correlation (close or equal to zero).

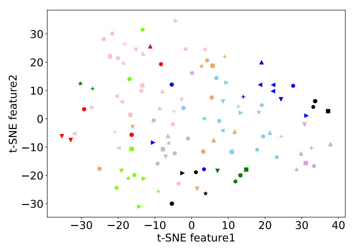
The goal of our clustering analysis is to find out how similar the generated features are across projects within a dataset. Hence, we configured *k*-means to search for 10 clusters (as we are generating 10 new features). We visualized the clustering results using the default settings of t-SNE.

Results: **RBM generates feature sets in which all features are strongly correlated with each other.** Figure 11(a) shows the Spearman rank correlation of one set of RBM-generated features for the Ant project in the PROMISE dataset using a heatmap. We observe that all correlations are close to 1 (dark red), which means that all RBM-generated features in the feature set are strongly correlated (and hence similar) to each other. We observe similar correlations for the other studied feature sets for the PROMISE and NASA datasets. RBM generates weakly correlated features for several projects in the AEEEM dataset (e.g., for the Eclipse project: Figure 11(b)). However, if we use a smaller set of original features from that dataset, such as only the change features, RBM generates strongly correlated features for these projects as well (e.g., for the Eclipse project: Figure 11(c)).

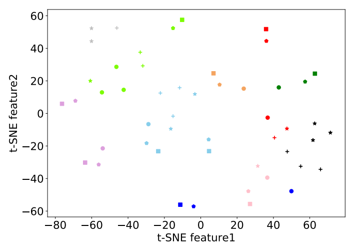
We observe similar correlations in feature sets that were generated by AE. However, the correlation within the AE-generated feature sets appeared to be linked to the specific project. For example, AE generates sets of features that are strongly cor-



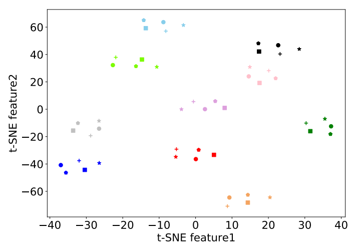
(a) RBM-generated weight sets in the PROMISE dataset.



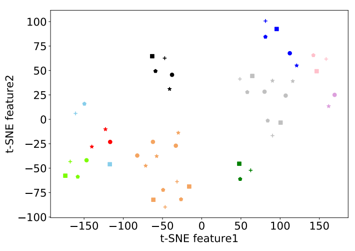
(b) RBM-generated weight sets in the NASA dataset.



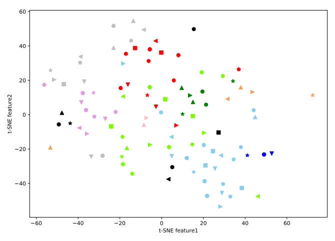
(c) RBM-generated weight sets in the AEEEM dataset.



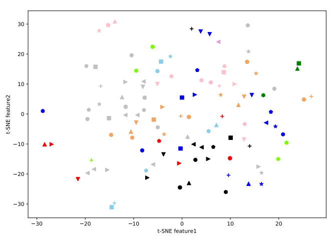
(d) RBM-generated weight sets using Change features in the AEEEM dataset.



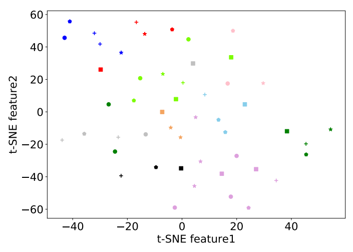
(e) RBM-generated weight sets using complexity code change features in the AEEEM dataset.



(f) AE-generated weight sets in the PROMISE dataset.



(g) AE-generated weight sets in the NASA dataset.



(h) AE-generated weight sets using Change features in the AEEEM dataset.

Fig. 12 *k*-means clustering results with t-SNE for generated weight sets in the studied datasets. Each shape represents a project, and each color represents a cluster.

related to each other for the Log4j project (see Figure 11(d)), but features that are not as strongly correlated for the Camel project (see Figure 11(e)). Hence, a possible explanation of the reason for the small variance of the performance of features that were generated by the neural network-based feature reduction techniques (i.e., RBM and AE) could be the strong correlation within the generated feature sets.

RBM mostly generates the same feature sets across projects. Figure 12(a) shows the *k*-means clustering result for one bootstrap sample of the RBM-generated weight sets in the PROMISE dataset using t-SNE [40]. Each shape refers to a project, and each color refers to a cluster that was identified by *k*-means clustering. The x-axis and y-axis refer to the t-SNE features that were generated from the 260 RBM-generated weight sets (10 weight sets on each project) by t-SNE. Hence, if there are 10 clearly identifiable groups of different shapes with the same colour in the t-SNE plot, we can conclude that the generated weight sets (and hence the features) are the same across projects. We observe that each colored cluster contains all shapes, which indicates that each cluster contains all projects. Hence, in this bootstrap sample, RBM generated the same feature sets across projects in the PROMISE dataset. We noticed a similar pattern for the other projects in the PROMISE dataset for the other bootstrap samples.

Figure 12(b) shows the result for the RBM-generated weight sets in the NASA dataset. The figure shows that RBM generated different feature sets across projects in the NASA dataset. A possible reason is that the original features in the NASA dataset are different in each project (Table 2).

Figure 12(c) shows the result for the RBM-generated weight sets in the AEEEM dataset. The figure shows that RBM generated different feature sets across projects in the AEEEM dataset. The explanation is similar to our observation during the correlation analysis. If we use one type of features, such as only the change features, Figure 12(d) shows that RBM generates the same feature sets across projects. However, if we use complexity code change features, we observe that RBM generates different feature sets across projects (Figure 12(e)). AE generated different features across projects (Figure 12(f), 12(g) and 12(h)).

From the discussion results, we can extract several possible explanations for the fact that RBM and AE significantly improve the variance of the performance across the unsupervised models, but not across the supervised models, and for why these feature reduction techniques perform well for the unsupervised models. As the studied neural network-based feature reduction techniques appear to generate strongly correlated features for a project, these feature sets suffer from multicollinearity [15], which is known to negatively affect the performance of the supervised models. However, as the unsupervised defect prediction models do not need to be trained, these models are not be affected by the multicollinearity problem. In addition, because RBM generates strongly correlated features for a project, the unsupervised models become much simpler, which seems to improve the variance of the performance across the unsupervised defect prediction models.

7 Threats to validity

7.1 External validity

With regards to the generalizability of our results, we applied our experiments to three publicly available datasets. These studied datasets (PROMISE, NASA and AEEEM) were all used in many prior defect prediction studies. The projects in these studied datasets span different domains, include both open source and industrial projects and contain different features. Future studies are necessary to investigate whether our results generalize to other projects.

In addition, we studied only a subset of the many existing feature reduction and selection techniques and defect prediction models. We carefully selected techniques and models that have been used before for defect prediction, and that have an implementation readily available. Without such an implementation, it is difficult and time-consuming to ensure that the implementation matches the one used in prior studies. Future studies are necessary to investigate whether our results apply to other feature reduction/selection techniques and defect prediction models.

7.2 Internal validity

In our experiments, we used AUC as a performance measure. AUC is a popular performance measure for defect prediction, as it does not require a threshold. However, different software project teams may have different objectives. Hence, future studies should investigate the impact of feature reduction techniques on other performance measures, while keeping in mind the possible pitfalls of studying threshold-dependent performance measures [71].

When using the out-of-sample bootstrap sampling, we encountered computational errors in two bootstrap samples. The errors occurred because two bootstrap samples violated requirements of the NB and SC models (e.g., a generated feature had a variance of zero which violates a requirement of the NB model). To mitigate this threat, we discarded these bootstrap samples and generated a new sample instead.

We provide all experimental scripts that we used in our study.[3] This replication package allows researchers and practitioners to replicate our experiments and confirm our results.

8 Conclusion

In defect prediction, reducing the number of features is an important step when building defect prediction models [5, 15, 67, 69]. Prior studies indicated that reducing the number of features avoids the problem of multicollinearity [15] and the curse of dimensionality [5]. Feature selection and reduction techniques help to reduce the number of features in a model. Feature selection techniques reduce the number of features in a model by selecting the most important ones, while feature reduction techniques

[3] `https://sailhome.cs.queensu.ca/replication/featred-vs-featsel-defectpred/`

reduce the number of features by creating new, combined features from the original features.

Prior work [18, 76] studied the impact of feature *selection* techniques on defect prediction models. Our work is the first large-scale study on the impact of feature *reduction* techniques on defect prediction models. In particular, we studied the impact of eight feature reduction techniques on five supervised and five unsupervised defect prediction models. In addition, we compared the impact of feature reduction techniques on defect prediction with the impact of the two best-performing feature selection techniques (according to prior work).

We studied the impact of feature reduction/selection techniques on defect prediction models along two dimensions:

1. The defect prediction performance (AUC) of the features that are generated by the feature reduction/selection techniques, to study whether feature reduction/selection techniques can improve the performance of defect prediction models.
2. The variance of the AUC across defect prediction models that use the features that are generated by feature reduction or selected by feature selection techniques. It is difficult to select the best performing model for each project, since the best model may change per project [17, 19]. Hence, we studied whether feature reduction or selection techniques can relieve the burden for practitioners of having to choose the best performing defect prediction model for their data.

Below, we summarize the main recommendations that follow from our work.

Recommendation 1: For the supervised defect prediction models, use the correlation-based (CFS) or consistency-based (ConFS) feature selection techniques. Our experiments in RQ3 show that, for the supervised models, CFS and ConFS outperform the feature reduction techniques (except feature agglomeration (FA)) and the original models. While FA has a similar performance, CFS and ConFS have a smaller performance variance. Hence, using CFS or ConFS in combination with a supervised defect prediction model allows practitioners to improve the performance of their defect prediction models, while making the choice for a particular defect prediction model easier as well.

Recommendation 2: For the unsupervised defect prediction models, use a neural network-based technique (Restricted Boltzmann Machine (RBM) or autoencoder (AE)). Our experiments in RQs 1 and 3 show that the RBM and AE feature reduction technique can significantly improve the performance of the unsupervised defect prediction models compared to the other feature reduction/selection techniques and the original models. In addition, we observed in RQs 2 and 3 that RBM and AE significantly improve the performance variance across the unsupervised models (except compared to the transfer component analyses (TCA and TCA+)). While the transfer component analyses (TCA and TCA+) have the smallest performance variance, they have a worse performance than RBM and AE. The effect size (Cliff's Delta) between the transfer component analyses and the neural network-based feature reduction techniques is negligible or small for the performance variance in favour of the transfer component analyses, but small (TCA) or large (TCA+) for the performance in favour of the neural network-based techniques. Hence, using a neural network-based feature reduction technique to preprocess the data of the unsupervised defect prediction models can improve both their performance and relieve the burden for practitioners of having to select the best-performing unsupervised defect prediction model for their project.

Recommendation 3: If a project has diverse data, the neural network-based techniques (Restricted Boltzmann Machine (RBM) or autoencoder (AE)) are likely to improve its defect prediction performance and performance variance. Our experiments showed that RBM and AE consistently improve the AUC and IQR of projects in the AEEEM dataset, for both supervised and unsupervised models. Practitioners should run PCA on their data to identify the diversity of their project's data (similar to what we did in Section 5.3.1). If the data turns out to be rich, RBM and AE are good options to improve the defect prediction performance and variance.

Acknowledgment

This work was partially supported by JSPS KAKENHI Grant Number 16K12415.

References

1. Scikit learn: Minmaxscaler. http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html. [Online; accessed 1-July-2018]
2. Abaei, G., Rezaei, Z., Selamat, A.: Fault prediction by utilizing self-organizing map and threshold. In: Proceedings of the International Conference on Control System, Computing and Engineering (ICCSCE), pp. 465–470. IEEE (2013)
3. Arora, I., Tetarwal, V., Saha, A.: Open issues in software defect prediction. Procedia Computer Science **46**, 906–912 (2015)
4. Basili, V.R., Briand, L.C., Melo, W.L.: A validation of object-oriented design metrics as quality indicators. IEEE Transactions on Software Engineering (TSE) **22**(10), 751–761 (1996)
5. Bellman, R.: Dynamic programming. Princeton University Press (1957)
6. Bingham, E., Mannila, H.: Random projection in dimensionality reduction: applications to image and text data. In: Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining, pp. 245–250. ACM (2001)
7. Bishnu, P.S., Bhattacherjee, V.: Software fault prediction using quad tree-based k-means clustering algorithm. IEEE Transactions on Knowledge and Data Engineering **24**(6), 1146–1150 (2012)
8. Challagulla, V.U.B., Bastani, F.B., Yen, I.L., Paul, R.A.: Empirical assessment of machine learning based software defect prediction techniques. International Journal on Artificial Intelligence Tools **17**(02), 389–400 (2008)
9. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Transactions on Software Engineering (TSE) **20**(6), 476–493 (1994)
10. Cohen, J.: Statistical power analysis for the behavioral sciences (1988)
11. D'Ambros, M., Lanza, M., Robbes, R.: An extensive comparison of bug prediction approaches. In: Proceedings of the 7th International Conference on Mining Software Repositories (MSR), pp. 31–41. IEEE (2010)

12. Dash, M., Liu, H.: Consistency-based search in feature selection. Artificial intelligence **151**(1), 155–176 (2003)
13. Dunn, J.C.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. J. Cybernet **3**, 32–57 (1973)
14. Faloutsos, C., Lin, K.I.: FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 163–174. ACM (1995)
15. Farrar, D.E., Glauber, R.R.: Multicollinearity in regression analysis: the problem revisited. The Review of Economic and Statistics pp. 92–107 (1967)
16. Gao, K., Khoshgoftaar, T.M., Wang, H., Seliya, N.: Choosing software metrics for defect prediction: an investigation on feature selection techniques. Software: Practice and Experience **41**(5), 579–606 (2011)
17. Ghotra, B., McIntosh, S., Hassan, A.E.: Revisiting the impact of classification techniques on the performance of defect prediction models. In: Proceedings of the 37th International Conference on Software Engineering (ICSE), pp. 789–800. IEEE Press (2015)
18. Ghotra, B., Mcintosh, S., Hassan, A.E.: 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 (MSR), pp. 146–157. IEEE Press (2017)
19. Gray, A.R., Macdonell, S.G.: Software metrics data analysis–exploring the relative performance of some commonly used modeling techniques. Empirical Software Engineering **4**(4), 297–316 (1999)
20. Guo, L., Cukic, B., Singh, H.: Predicting fault prone modules by the dempster-shafer belief networks. In: Proceedings of the 18th International Conference on Automated Software Engineering (ASE), pp. 249–252. IEEE (2003)
21. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. ACM SIGKDD explorations newsletter **11**(1), 10–18 (2009)
22. Hall, M.A.: Correlation-based feature selection for machine learning. Ph.D. thesis, University of Waikato Hamilton (1999)
23. Hall, M.A., Holmes, G.: Benchmarking attribute selection techniques for discrete class data mining. IEEE Transactions on Knowledge and Data Engineering **15**(6), 1437–1447 (2003)
24. Halstead, M.H.: Elements of software science, vol. 7. Elsevier New York (1977)
25. Han, J., Moraga, C.: The influence of the sigmoid function parameters on the speed of backpropagation learning. In: Proceedings of the International Workshop on Artificial Neural Networks, pp. 195–201. Springer (1995)
26. Hartigan, J.A., Wong, M.A.: Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) **28**(1), 100–108 (1979)
27. Hassan, A.E.: Predicting faults using the complexity of code changes. In: Proceedings of the 31st International Conference on Software Engineering (ICSE), pp. 78–88. IEEE Computer Society (2009)

28. He, Z., Shu, F., Yang, Y., Li, M., Wang, Q.: An investigation on the feasibility of cross-project defect prediction. Automated Software Engineering **19**(2), 167–199 (2012)
29. Herbold, S.: Training data selection for cross-project defect prediction. In: Proceedings of the 9th International Conference on Predictive Models in Software Engineering, p. 6. ACM (2013)
30. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
31. Hira, Z.M., Gillies, D.F.: A review of feature selection and feature extraction methods applied on microarray data. Advances in bioinformatics **2015** (2015)
32. Ho, T.K.: Random decision forests. In: Proceedings of the 3rd International Conference on Document Analysis and Recognition, vol. 1, pp. 278–282. IEEE (1995)
33. Jureczko, M., Madeyski, L.: Towards identifying software project clusters with regard to defect prediction. In: Proceedings of the 6th International Conference on Predictive Models in Software Engineering, p. 9. ACM (2010)
34. Kamei, Y., Fukushima, T., McIntosh, S., Yamashita, K., Ubayashi, N., Hassan, A.E.: Studying just-in-time defect prediction using cross-project models. Empirical Software Engineering **21**(5), 2072–2106 (2016)
35. Kaufman, L., Rousseeuw, P.J.: Finding groups in data: an introduction to cluster analysis, vol. 344. John Wiley & Sons (2009)
36. Kim, S., Zimmermann, T., Whitehead Jr, E.J., Zeller, A.: Predicting faults from cached history. In: Proceedings of the 29th International Conference on Software Engineering (ICSE), pp. 489–498. IEEE Computer Society (2007)
37. Kohonen, T.: The self-organizing map. Proceedings of the IEEE **78**(9), 1464–1480 (1990)
38. Kuhn, M.: Caret: classification and regression training. Astrophysics Source Code Library (2015)
39. Landwehr, N., Hall, M., Frank, E.: Logistic model trees. Machine Learning **59**(1), 161–205 (2005)
40. van der Maaten, L.: Accelerating t-SNE using tree-based algorithms. Journal of Machine Learning Research **15**(1), 3221–3245 (2014)
41. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. Journal of Machine Learning Research **9**(Nov), 2579–2605 (2008)
42. Martinetz, T., Schulten, K., et al.: A "neural-gas" network learns topologies. Artificial Neural Networks pp. 397–402 (1991)
43. McCabe, T.J.: A complexity measure. IEEE Transactions on Software Engineering (TSE) (4), 308–320 (1976)
44. McDonald, J.H.: Handbook of Biological Statistics (3rd ed.). Sparky House Publishing, Baltimore, Maryland. (2014)
45. Menzies, T., Greenwald, J., Frank, A.: Data mining static code attributes to learn defect predictors. IEEE Transactions on Software Engineering (TSE) **33**(1) (2007)
46. Menzies, T., Owen, D., Richardson, J.: The strangest thing about software. Computer **40**(1) (2007)

47. Moser, R., Pedrycz, W., Succi, G.: A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In: Proceedings of the 30th International Conference on Software Engineering (ICSE), pp. 181–190. IEEE (2008)
48. Muthukumaran, K., Rallapalli, A., Murthy, N.: Impact of feature selection techniques on bug prediction models. In: Proceedings of the 8th India Software Engineering Conference, pp. 120–129. ACM (2015)
49. Nagappan, N., Ball, T., Zeller, A.: Mining metrics to predict component failures. In: Proceedings of the 28th International Conference on Software Engineering (ICSE), pp. 452–461. ACM (2006)
50. Nam, J.: Survey on software defect prediction. HKUST PhD Qualifying Examination, Department of Compter Science and Engineerning, The Hong Kong University of Science and Technology, Tech. Rep (2014)
51. Nam, J., Fu, W., Kim, S., Menzies, T., Tan, L.: Heterogeneous defect prediction. IEEE Transactions on Software Engineering (2017)
52. Nam, J., Kim, S.: CLAMI: Defect prediction on unlabeled datasets. In: Proceedings of the 30th International Conference on Automated Software Engineering (ASE), pp. 452–463. IEEE (2015)
53. Nam, J., Kim, S.: Heterogeneous defect prediction. In: Proceedings of the 10th Joint Meeting on Foundations of Software Engineering (FSE), pp. 508–519. ACM (2015)
54. Nam, J., Pan, S.J., Kim, S.: Transfer defect learning. In: Proceedings of the 2013 International Conference on Software Engineering (ICSE), pp. 382–391. IEEE Press (2013)
55. NASA: Metrics data program. `http://openscience.us/repo/defect/mccabehalsted/`. [Online; accessed 1-September-2016]
56. Neumann, D.E.: An enhanced neural network technique for software risk analysis. IEEE Transactions on Software Engineering (TSE) **28**(9), 904–912 (2002)
57. Pan, S.J., Tsang, I.W., Kwok, J.T., Yang, Q.: Domain adaptation via transfer component analysis. IEEE Transactions on Neural Networks **22**(2), 199–210 (2011)
58. Peters, F., Menzies, T., Gong, L., Zhang, H.: Balancing privacy and utility in cross-company defect prediction. IEEE Transactions on Software Engineering **39**(8), 1054–1068 (2013)
59. Petrić, J., Bowes, D., Hall, T., Christianson, B., Baddoo, N.: The jinx on the NASA software defect data sets. In: Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, pp. 1–5. ACM (2016)
60. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers (1993)
61. Rathore, S.S., Gupta, A.: A comparative study of feature-ranking and feature-subset selection techniques for improved fault prediction. In: Proceedings of the 7th India Software Engineering Conference, p. 7. ACM (2014)
62. Ren, J., Qin, K., Ma, Y., Luo, G.: On software defect prediction using machine learning. Journal of Applied Mathematics **2014** (2014)

63. Rodríguez, D., Ruiz, R., Cuadrado-Gallego, J., Aguilar-Ruiz, J.: Detecting fault modules applying feature selection to classifiers. In: Proceedings of the 2007 International Conference on Information Reuse and Integration, pp. 667–672. IEEE (2007)
64. Rodriguez, D., Ruiz, R., Cuadrado-Gallego, J., Aguilar-Ruiz, J., Garre, M.: Attribute selection in software engineering datasets for detecting fault modules. In: Proceedings of the 2007 EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 418–423. IEEE (2007)
65. Rokach, L., Maimon, O.: Clustering methods. In: Data mining and knowledge discovery handbook, pp. 321–352. Springer (2005)
66. Shepperd, M., Song, Q., Sun, Z., Mair, C.: Data quality: Some comments on the NASA software defect datasets. IEEE Transactions on Software Engineering (TSE) **39**(9), 1208–1215 (2013)
67. Shihab, E.: Practical software quality prediction. In: Proceedings of the 2014 International Conference on Software Maintenance and Evolution (ICSME), pp. 639–644. IEEE (2014)
68. Shihab, E., Jiang, Z.M., Ibrahim, W.M., Adams, B., Hassan, A.E.: Understanding the impact of code and process metrics on post-release defects: A case study on the Eclipse project. In: Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 4:1–4:10. ACM (2010)
69. Shivaji, S., Whitehead, E.J., Akella, R., Kim, S.: Reducing features to improve code change-based bug prediction. IEEE Transactions on Software Engineering (TSE) **39**(4), 552–569 (2013)
70. Smolensky, P.: Information processing in dynamical systems: Foundations of harmony theory. Tech. rep., DTIC Document (1986)
71. Tantithamthavorn, C., Hassan, A.E.: An experience report on defect modelling in practice: Pitfalls and challenges. In: Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), pp. 286–295. ACM (2018)
72. Tantithamthavorn, C., McIntosh, S., Hassan, A.E., Matsumoto, K.: Automated parameter optimization of classification techniques for defect prediction models. In: Proceedings of the 38th International Conference on Software Engineering (ICSE), pp. 321–332. ACM (2016)
73. Tantithamthavorn, C., McIntosh, S., Hassan, A.E., Matsumoto, K.: An empirical comparison of model validation techniques for defect prediction models. IEEE Transactions on Software Engineering (TSE) **43**(1), 1–18 (2017)
74. Tassey, G.: The economic impacts of inadequate infrastructure for software testing. National Institute of Standards and Technology, RTI Project **7007**(011) (2002)
75. Von Luxburg, U.: A tutorial on spectral clustering. Statistics and computing **17**(4), 395–416 (2007)
76. Xu, Z., Liu, J., Yang, Z., An, G., Jia, X.: The impact of feature selection on defect prediction performance: An empirical comparison. In: Proceedings of the 27th International Symposium on Software Reliability Engineering (ISSRE), pp. 309–320. IEEE (2016)

77. Yang, B., Yin, Q., Xu, S., Guo, P.: Software quality prediction using affinity propagation algorithm. In: Proceedings of the International Joint Conference on Neural Networks, pp. 1891–1896. IEEE (2008)
78. Zhang, F., Zheng, Q., Zou, Y., Hassan, A.E.: Cross-project defect prediction using a connectivity-based unsupervised classifier. In: Proceedings of the 38th International Conference on Software Engineering (ICSE), pp. 309–320. ACM (2016)
79. Zhang, H.: The optimality of Naive Bayes. In: FLAIRS Conference. AAAI Press (2004)
80. Zhong, S., Khoshgoftaar, T.M., Seliya, N.: Unsupervised learning for expert-based software quality estimation. In: HASE, pp. 149–155. Citeseer (2004)
81. Zimmermann, T., Nagappan, N., Gall, H., Giger, E., Murphy, B.: Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In: Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC-FSE), pp. 91–100. ACM (2009)
82. Zwillinger, D., Kokoska, S.: CRC standard probability and statistics tables and formulae. Crc Press (1999)