IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 October 2024, accepted 5 November 2024, date of publication 8 November 2024, date of current version 18 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3494044

RESEARCH ARTICLE

Validating Unsupervised Machine Learning Techniques for Software Defect Prediction With Generic Metamorphic Testing

PAK YUEN PATRICK CHAN AND JACKY KEUNG, (Senior Member, IEEE)
Department of Computer Science, City University of Hong Kong, Hong Kong, China

Corresponding author: Pak Yuen Patrick Chan (ppychan2-c@my.cityu.edu.hk)

This work was supported in part by the General Research Fund of the Research Grants Council of Hong Kong, and in part by the Research Funds of the City University of Hong Kong under Grant 6000796.

ABSTRACT In the realm of software defect prediction, unsupervised models often step in when labelled datasets are scarce, despite facing the challenge of validating models without prior knowledge of data. Addressing this, we proposed a novel approach leveraging generic metamorphic testing to validate such models effectively, bypassing the need for expert-derived metamorphic relations. Our method identifies five categories of generic metamorphic relations, further divided into twenty-one individual generic metamorphic relations, all formulated through generic Data Mutation Operators. This framework enables us to generate follow-up datasets from the source datasets, training respective software defect prediction models. By comparing predictions between the source and follow-up software defect prediction models and identifying inconsistencies, we can assess the model's sensitivity to generic metamorphic relations as a form of validation. This approach was rigorously evaluated across twenty software defect prediction models, incorporating fourteen different machine learning algorithms and twenty high-dimensional public datasets. Remarkably, the robustness of our generic MT model was confirmed, showing substantial effectiveness in validating software defect prediction models, independent of whether they were supervised or unsupervised. Software defect prediction models, using Agglomerative clustering and Density-Based Spatial Clustering of Applications with Noise algorithms, did not violate any metamorphic relation, and nineteen software defect prediction models did not significantly violate the generic metamorphic relation ''Shrinkage and Expansion''. Our findings suggest that employing generic metamorphic relations, especially ''Shrinkage and Expansion'', can universally enhance the validation of defect prediction models. Furthermore, our model can be applied to continuously monitor software defect prediction models.

INDEX TERMS Clustering, machine learning, metamorphic relation, metamorphic testing, software defect prediction, validation.

I. INTRODUCTION

As the complexity and scale of software modules are growing, defect prediction has become an active research area in software engineering [1], [2], [3], and defect prediction with machine learning has become an essential technique in the software reliability field [4]. Supervised learning models with labelled software defect datasets, which are Supervised Software Defect Prediction models (SSDPs), are widely used [1]. However, creating labelled software defect datasets is challenging when the software projects have limited historical software data [1]. SSDPs cannot be applied to datasets that do not have labels. Thus, Unsupervised Software Defect Prediction models (USDPs) become the solution as USDPs do not need labelled datasets [1], [5]. Although USDPs can mitigate the issue of insufficient labelled defect datasets, unsupervised models are challenging to validate as there is no prior knowledge of the unlabelled datasets, making

The associate editor coordinating the review of this manuscript and approving it for publication was Mahmoud Elish.

this an untestable situation, referred to as the test oracle problem [6].

Despite the testing challenge, Metamorphic Testing (MT) is a widely used technique to alleviate the test oracle problem [6], [7], [8], [9], [10], [11]. Instead of verifying the outputs of software programs, MT validates the expected relationships between inputs and outputs of multiple executions of the software and these relationships are called Metamorphic Relations (MRs). If any MR is violated after the execution of software, it is interpreted as a faulty implementation. Through this approach, we avoid the test oracle problem. However, identifying MRs is challenging because these expected relationships were mainly created by experts or users [6], [10], [12].

To alleviate the challenge of identifying MR, we proposed a generic and adaptive MT model for validating Software Defect Prediction models (SDPs). Different from traditional models, this model is generic and based on two proven models with no expert or user input or prior knowledge needed. Five generic types of MRs are identified from the ''METamorphic Testing approach to assessing and validating unsupervised machine Learning'' model (METTLE) [6]. They are mapped with Data mutation operators (DMOs) of the μMT model [12], [13] and proposed DMOs to establish the generic MT model with twenty-one adaptable MRs in this study. This paper experimented with twenty SDPs using fourteen different machine learning algorithms on twenty publicly available high-dimensional datasets. We applied the MRs to generate follow-up datasets to train follow-up SDPs for testing. The predictions of follow-up SDPs would be compared with predictions of SDPs trained by the source datasets. If there is no violation, which means no inconsistent prediction between them, the SDP is insensitive to the related MR and that MR could be used for validation. It is because if SDP is insensitive to data transformation, then any violation of MR can be interpreted as a fault in the program.

The results show that the proposed model can effectively validate SDPs, and we conduct robust statistical tests to confirm the effectiveness of the model. The main contributions of this study include:

(1) We validate twenty SDPs implementing fourteen clustering algorithms with our generic MT model. To our best knowledge, this is the first study to introduce a generic MT model for validating SDPs utilizing fourteen different clustering algorithms with high-dimensional data.

(2) Our generic MT model, which incorporates five types of MRs mapping with DMOs, proves to be a versatile tool in the validation of SDPs. The fact that two USDPs did not violate any MR of the model, and nineteen SDPs did not significantly violate the generic MR ''Shrinkage and Expansion'', demonstrates the potential for our model to be extended into other domains using a variety of machine learning algorithms.

(3) Experiments on USDPs and SSDPs demonstrate the effectiveness of our model in labelled or unlabelled dataset situations.

(4) We followed the future direction recommended by the study of [6] and successfully demonstrated the feasibility of the METTLE model in high-dimensional data experiments.

(5) We have developed a generic validation model that researchers and practitioners can use even without expertise or input from experts or users. This model can be applied to any numeric data studied by machine learning based systems, allowing researchers and practitioners to validate targets without prior knowledge of the machine learning based systems.

The rest of this paper is organized as follows. Section II introduces the underlying concepts of software defect prediction using the machine learning approach and MT. It also discusses the challenges in test oracle problem and briefly discusses the recent related works. Section III describes our proposed generic MT model. Section IV describes the experimental setup to determine the effectiveness of the proposed model in validating SDPs and discusses the results. Section V discusses the threats to the validity of this study. Section VI concludes this paper with future directions.

II. BACKGROUND

This section briefly introduces software defect prediction using machine learning approaches, test oracle problem and the preliminary knowledge of MT and related works.

A. SOFTWARE DEFECT PREDICTION

Software Defect Prediction helps to identify defect tendencies in software modules, and there are different approaches to predicting defects. Xu et al. [4] proposed a supervised technique called the Augmented-Code Property Graph-based defect prediction method, which uses code's graph representation combined with a neural network to generate a graph of defect region candidates to predict certain defect types. Wang et al. [3] used supervised gated hierarchical long short-term memory networks which extract semantic features of abstract syntax trees of source code and traditional features provided by the PROMISE repository. Kumar et al. [1] proposed an unsupervised software defect prediction approach using threshold derivation for clustering and labelling the data points.

Supervised and unsupervised machine learning techniques are the two main types used in software defect prediction [14], [15]. There are comparison studies between them for seeking improvements. Chen et al. [14] study compared supervised and unsupervised machine learning methods and found that unsupervised methods based on the ''Lines of Code'' metric performed better than or the same as the supervised methods in their study. Li et al. [15] conducted a systematic literature review and found that unsupervised and supervised classifiers performed as well as each other. Xu et al. [5] empirical study

also indicated that clustering-based unsupervised models are worse than supervised models.

Software defect prediction is based on the features extracted from software modules to make recommendations, and ''Code metrics'' and ''Process metrics'' are the two commonly used software feature metrics for defect prediction [3], [16]. These metrics collect data from software modules for training the defect prediction models using machine learning [17].

B. MACHINE LEARNING AND TEST ORACLE

Supervised machine learning technique is learning with labelled data to make predictions. However, labelled software defect data for training can be unavailable [15]. Therefore, unsupervised techniques become popular because it is learning with unlabelled data [14], [15]. Unsupervised machine learning systems perform as well as the supervised machine learning systems and clustering-based prediction approaches dominate USDPs [15], [18]. However, USDPs do not have a ground truth to refer to, which is like a test oracle problem to validate the USDPs. MT is a widely used approach to address the test oracle problem [6], [7], [8], [9], [10], [11].

C. METAMORPHIC TESTING

MT has been used to validate the ''expected'' relationships between inputs and outputs of program executions. These ''expected'' relationships are expressed as metamorphic relations (MRs), and they are the core element of MT. If the output results in various software executions violate an MR, then a fault is revealed [6], [8], [11], [19]. So, by examining these expected relationships between inputs and outputs, which are MRs, we can validate the program under testing and avoid the test oracle problem. As machine learning is composed of programs and datasets, if the machine learning model is insensitive to data transformation, then any violation of MR can be interpreted as a fault in the program.

The idea of MR can be illustrated by a mathematical property of the *sine* function, which is ''$\sin(x) = \sin(\pi - x)$''. For example, if the corresponding MR has two inputs, x1 and x2, that satisfy ''$x1 + x2 = \pi$'', then two function outputs should be equal, i.e., $\sin(x1) = \sin(x2)$. Based on MR, x2 is constructed based on $x1$ and $\sin(x1)$. Therefore, x1 is the source input, and x2 is the follow-up input. Such MR can be used to test the program as if the outputs of $\sin(x1)$ and $\sin(x2)$ are different, then the implementation of function ''sin()'' must have faults as it violates the above MR [10].

Chua et al. [20] addressed the oracle problem in software testing in System of Systems by applying MT successfully. Xiao et al. [7] applied an MT framework designed for Deep Learning compilers to identify incorrect compilations. Zhang et al. [10] study successfully applied MT to validate systems implementing ''class integration test order generation''. There are many domains of applications and compilers have been applied MT to alleviate test oracle problems [21], such as bioinformatic software [22], machine learning classifiers [18], [23], [24], [25], [26], [27], cryptographic software [28], and online search engines [19].

In a situation in which software modules have limited historical data and limited experts' or users' inputs, MRs could be challenging to determine [6], [10], [12], and a generic approach can be applied. Zhou et al. [11] further the concept of metamorphic relations and defined Metamorphic Relation Patterns (MRP) to derive multiple concrete metamorphic relations. Segura et al. [19] presented a list of MRP to assist in identifying MRs in testing query-based systems. Chua et al. [20] study on generic MR generation was based on abstract MRP and symmetry. Studies proved that the μMT model, which used DMOs to construct MRs, is feasible [12]. Identifying MRs can be achieved by providing a set of DMOs and mapping rules of μMT [13].

Therefore, motivated by the necessity of more advanced testing techniques for validating machine learning algorithms that suffer from the test oracle problem and situations in which software modules have limited historical data and limited experts' or users' inputs, this study targets to develop a generic MT validation model and demonstrate the feasibility of model validating software defect prediction models through a comprehensive scaled experiment.

III. PROPOSED VALIDATION MODEL

This section describes the validation model and how the generic MT model is designed.

A. METAMORPHIC TESTING AS VALIDATION MODEL

This study adopts MT to validate SDPs. In order to establish a generic and adaptable MT model, we followed [6]'s future direction to experiment with a generic METTLE model on high-dimensional datasets with two non-scalable labels. We identified five generic types of MRs from the METTLE model [6]. Then, we mapped the first three types of MRs with generic DMOs of μMT [12], [13] and mapped the rest with our proposed DMOs to establish the generic MT model. This adaptation enhances the model's versatility, allowing users to apply it to numeric data features studied by various machine learning algorithms without understanding the underlying models. The proposed model does not include MR ''manipulating the coordinate system'' of METTLE because it is challenging to ensure the correctness of transforming the coordinates of high-dimensional datasets and determining the outcomes. In this regard, the proposed model is more adaptable and accessible.

B. METAMORPHIC RELATIONS

This study adopts twenty-one individual generic MRs from five generic types of MRs of METTLE mapping with DMOs of μMT and our proposed DMOs (Table 1).

1) MR1: MANIPULATING OBJECT ORDER

This relation assumes that the order of objects should not affect the prediction outcomes [6]. This MR could use

TABLE 1. Mapping table of MRs and DMOs.

MR of METTLE	DMO mappings	Source of DMOs
MR1: Manipulating object order	MR1.1 - Swap(x1, x2)	μMT
MR2: Manipulating the distinctness Among cluster in dataset	MR2.1 - Half(Half(x))	μMT
	MR2.2 - Half(x)	
	MR2.3 - Double(x)	
	MR2.4 - Double(Double(x))	
MR3: Manipulating the object density of one or more clusters in the dataset	MR3.1 - x ***AND*** Half(Half(x))	μMT
	MR3.2 - x ***AND*** Half(x)	
	MR3.3 - x ***AND*** Double(x)	
	MR3.4 - x ***AND*** Double(Double(x))	
	MR3.5 - x ***AND*** Half(x) ***AND*** Half(Half(x))	
	MR3.6 - x ***AND*** Double(x) ***AND*** Double(Double(x))	
	MR3.7 - x ***AND*** Half(x) ***AND*** Half(Half(x)) ***AND*** Double(x) ***AND*** Double(Double(x))	
MR4: Manipulating attributes	MR4.1 - Remove the highest co-related feature	Proposed
	MR4.2 - Remove 10% of total features which are the highest correlated	
MR5: Manipulating outliers	MR5.1 – $\overline{x} \pm 3\sigma$	Proposed
	MR5.2 – $\overline{x} \pm 4\sigma$	
	MR5.3 – $\overline{x} \pm 5\sigma$	
	MR5.4 – $\overline{x} \pm 6\sigma$	
	MR5.5 – $\overline{x} \pm 3\sigma$ ***AND*** $\overline{x} \pm 4\sigma$	
	MR5.6 – $\overline{x} \pm 3\sigma$ ***AND*** $\overline{x} \pm 4\sigma$ ***AND*** $\overline{x} \pm 5\sigma$	
	MR5.7 – $\overline{x} \pm 3\sigma$ ***AND*** $\overline{x} \pm 4\sigma$ ***AND*** $\overline{x} \pm 5\sigma$ ***AND*** $\overline{x} \pm 6\sigma$	

"Swap" DMO of μMT [12], [13]. This MR reverses the order of the source dataset to create the follow-up dataset.

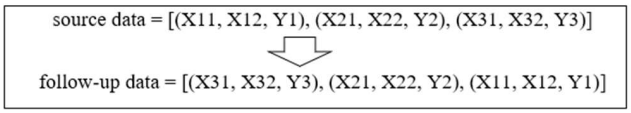


FIGURE 1. Illustration of MR1 data transformation.

2) MR2: MANIPULATING THE DISTINCTNESS AMONG CLUSTER IN DATASET

This relation assumes that shrinking and expanding proportionally to some or all clusters towards to or away from their centroids in the dataset should not affect the prediction outcomes [6]. We chose "Half" and "Double" DMO of μMT [12], [13], which means half and double all the values of all features, for this MR. In addition, this study applied the same DMO again to increase the levels of shrinking and expanding, which means a quarter times and four times all the values of all features.

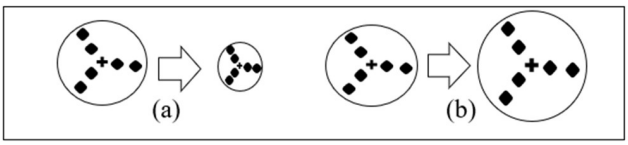


FIGURE 2. (a) Illustration of MR2 shrinking in proportion. (b) Illustration of MR2 expanding in proportion.

3) MR3: MANIPULATING THE OBJECT DENSITY OF ONE OR MORE CLUSTERS IN THE DATASET

This relation assumes that adding objects into clusters in the dataset to increase the object densities of these clusters should not affect the prediction outcomes [6]. This MR reused the follow-up datasets created in MR2 and added them to the source dataset to create different levels of densities for testing.

4) MR4: MANIPULATING ATTRIBUTES MATCHED WITH REMOVING HIGHLY CORRELATED FEATURES

This relation assumes a high correlation indicates that an attribute can be considered redundant and, hence, to be removed [6]. This study proposed using the removal of highly correlated features as DMO. Two MRs are proposed. One is to remove the highest correlated attribute in the dataset. The other one is to remove about 10% of total attributes, which are the highest correlated attributes.

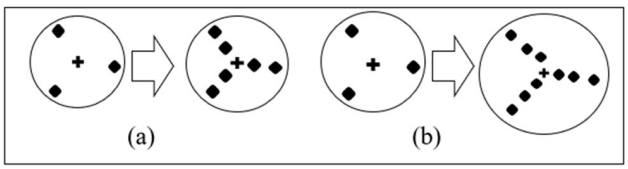
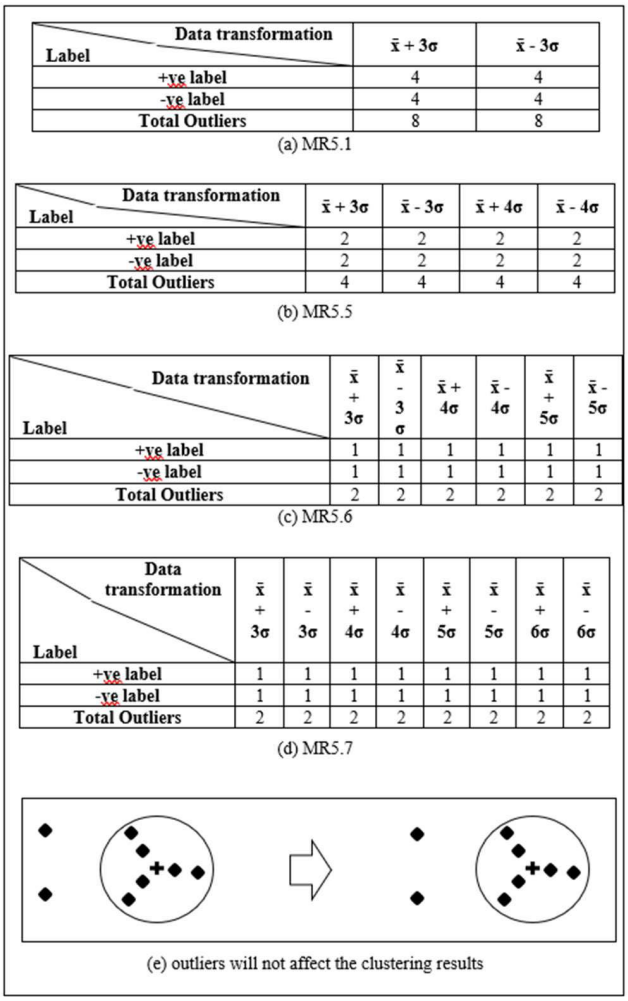


FIGURE 3. (a) Illustration of MR3 increases density with instances that are shrinking in proportion. (b) Illustration of MR3 increases density with instances that expanding in proportion.

5) MR5: MANIPULATING OUTLIERS

This relation assumes the clustering system will handle outliers by filtering them or assigning new cluster labels; thus, it should not affect the prediction outcomes [6]. This study proposed using the mean values and different levels of standard deviation values of features as DMO to create outliers. We created the same number of outliers with positive and negative labels to reduce the effect of imbalanced labels.

For example, MR5.1 used transformation $\overline{x} \pm 3\sigma$, which means adding or subtracting three times the standard deviation values of features to the mean values of features, in order



Label \ Data transformation	$\bar{x}+3\sigma$	$\bar{x}-3\sigma$
+ve label	4	4
-ve label	4	4
Total Outliers	8	8

(a) MR5.1

Label \ Data transformation	$\bar{x}+3\sigma$	$\bar{x}-3\sigma$	$\bar{x}+4\sigma$	$\bar{x}-4\sigma$
+ve label	2	2	2	2
-ve label	2	2	2	2
Total Outliers	4	4	4	4

(b) MR5.5

Label \ Data transformation	$\bar{x}+3\sigma$	$\bar{x}-3\sigma$	$\bar{x}+4\sigma$	$\bar{x}-4\sigma$	$\bar{x}+5\sigma$	$\bar{x}-5\sigma$
+ve label	1	1	1	1	1	1
-ve label	1	1	1	1	1	1
Total Outliers	2	2	2	2	2	2

(c) MR5.6

Label \ Data transformation	$\bar{x}+3\sigma$	$\bar{x}-3\sigma$	$\bar{x}+4\sigma$	$\bar{x}-4\sigma$	$\bar{x}+5\sigma$	$\bar{x}-5\sigma$	$\bar{x}+6\sigma$	$\bar{x}-6\sigma$
+ve label	1	1	1	1	1	1	1	1
-ve label	1	1	1	1	1	1	1	1
Total Outliers	2	2	2	2	2	2	2	2

(d) MR5.7

(e) outliers will not affect the clustering results

FIGURE 4. Illustration of MR5 (a)MR5.1 (b)MR5.5 (c)MR5.6 (d)MR5.7 (e)Outliers will not affect the clustering results due to the distances.

to create outliers (Fig. 4a). MR5.6 has six data transformations, so we created six lines of outliers with a positive label and six lines of outliers with a negative label (Fig. 4c).

IV. EXPERIMENT AND DISCUSSION

This section describes the setup of the experiment. It defines the research objective and research questions of our experiment. Then, it describes the subject systems and data used in the experiment. Thereafter, it discusses detailed experimental design, including environment configuration, measurement, experimental procedure, dataset preparation, and parameter setting. Finally, we present our experimental results and discuss, analyze, and interpret the results for research questions.

A. RESEARCH OBJECTIVE AND QUESTIONS

The main objective of the experiment is to demonstrate the feasibility of the proposed model for validating USDPs using clustering algorithms and SSDPs. This study aims to answer the following research questions.

RQ1: Could the generic MT model validate SDPs? Different MRs have different impacts on MT validation capacity for different machine learning algorithms. We aim to evaluate the effectiveness of different MRs in validating different machine learning algorithms by counting the occurrence of violations related to MRs. These numbers can indicate the effectiveness of the generic MT model.

RQ2: What are the underlying causes of violations? When a violation occurs, we analyze the results with the violation patterns, the characteristics of MRs, and the characteristics of SDPs, in order to seek any relationship that leads to the violations.

RQ3: What are the differences in applying generic MT model to validate USDPs and SSDPs? We further analyze the effectiveness of MT on supervised and unsupervised machine learning algorithms by comparing the differences in prediction inconsistencies and accuracies between USDPs and SSDPs.

B. SUBJECT MACHINE LEARNING ALGORITHM

References [26], [27] successfully applied MT to validate unsupervised machine learning algorithms; however, their studies only involved three machine learning algorithms. This study explores the feasibility of applying the proposed model to validate different SDPs utilizing different machine learning algorithms. Thus, our experiment involved eighteen USDPs and two SSDPs utilizing fourteen different machine learning algorithms (Table 2). They were obtained from the ''scikit-learn'' library, a popular machine learning library for Python, as [29] mentioned in their study. Version 0.24.2 of the ''scikit-learn'' library is used in this experiment.

1) UNSUPERVISED SOFTWARE DEFECT PREDICTION MODELS

USDPs are using some of the most widely used unsupervised clustering algorithms [1], [5], [15], [30], [31], and we occupied clustering classifications of Xu et al. [5] to group the subject USDPs as the following.

Model-Based Clustering(MBC) assumes the probability model for each cluster and arranges instances to the cluster that best fits the model [5]. This study selected two expectation-maximization (EM) methods, including the ''Gaussian Mixture Model'' (GMM) and the ''Bayesian Gaussian Mixture Model'' (BayGM).

GMM uses a probabilistic assignment of data points to clusters where each cluster has its separate Gaussian distribution [31]. In the ''scikit_learn'' library, the ''GaussianMixture'' function has a mandatory option for choosing the type of covariance [32], [33]. This study treats each GMM model with different covariance options as separate SDP, so there are three GMM models. They are GMM with ''diagonal'' covariance matrix option (GMM(diag)), GMM with ''spherical'' covariance matrix option (GMM(spherical)), and GMM with ''full'' covariance matrix option (GMM(full)).

BayGM is a variant of GMM and uses the Bayesian inference framework to estimate the posterior distribution of the parameters and the hyperparameters of the distribution [34]. In the ''scikit_learn'' library, the ''BayesianGaussianMixture'' function has a mandatory option for choosing the type of covariance [32], [35]. This study treats each BayGM model with different covariance options as separate SDP, so there are three BayGM models. They are BayGM with ''diagonal'' covariance matrix option (BayGM(diag)), BayGM with ''spherical'' covariance matrix option (BayGM(spherical)), and BayGM with ''full'' covariance matrix option (BayGM(full)).

Partition-Based Clustering (PBC) randomly selects partitions of instances as clusters and assigns the remaining instances to clusters whose center point is closest to those instances. It updates the center point of each cluster and relocates the clusters of other instances iteratively until the center points of the clusters remain unchanged [5]. This study selects ''K-Means'', ''Mini-Batch K-Means'', and ''Bisecting K-Means'' clustering methods.

K-Means uses squared Euclidean distance as the similarity measure to select instances in the clusters. The number of clusters is typically chosen by the user, and it aims to minimize the sum of the squared distances between each data point and its assigned cluster center [5], [30]. The ''scikit_learn'' library has a mandatory EM algorithm option for the K-Means model [36]. This study treats each K-Means model with different EM algorithm options as separated SDP, so there are two K-Means models. They are the K-Means model using the ''lloyd'' EM algorithm (Kmean (full)) and the K-Means model using the ''elkan'' EM algorithm (Kmean(elkan)) [36], [37].

Mini-Batch K-Means (MiniKM) is a variant of K-Means that uses shorter batches of datasets to calculate similarity, in order to reduce the computation time [5].

Bisecting K-Means is a variant of K-Means and works by recursively dividing clusters into two sub-clusters until the desired number of clusters is reached [38]. The ''scikit_learn'' library has a mandatory EM algorithm option for the Bisecting K-Means model [39]. This study treats each Bisecting K-Means model with different EM algorithm options as separated SDP, so there are two Bisecting K-Means models, and they are Bisecting K-Means model using ''lloyd'' EM algorithm (BisectKM(full)), and Bisecting K-Means model using ''elkan'' EM algorithm (BisectKM(elkan)) [37], [39].

Graph-Theory-Based Clustering (GTBC) constructs a weighted graph where each node represents an instance, and the weight of the edge denotes the similarity measure of its two nodes. It iteratively divides graphs until it separates all instances into non-overlapped clusters [5]. This study selected ''Spectral Clustering'' (Spectral) and ''Affinity Propagation Clustering'' (Affinity).

Spectral transforms the data into a lower-dimensional space using the eigenvectors of a similarity matrix. By using the similarity matrix, it constructs a connected graph, in which every pair of points is connected by a weighted edge, and treats the clustering as a graph partitioning problem [5], [40], [41].

Affinity regards all the data points as potential cluster centers and the negative value of the Euclidean distance between two data points as the affinity. The core idea means that the larger the sum of the affinity of one data point, the higher probability of this data point being the cluster center [5], [15], [40], [42].

Hierarchy-Based Clustering (HBC) creates a hierarchical decomposition of the data. This study selected ''Agglomerative Clustering'' (Agglo) and ''Balanced iterative reducing and clustering using hierarchies'' (Birch) clustering [5].

Agglo uses bottom-up decomposition to build clusters. It firstly treats each instance as a separate cluster and iteratively merges each instance to the closest cluster. Secondly, the process continues until all instances are merged into clusters or a predefined condition is met [5].

Birch, on the other hand, uses top-down decomposition to find clusters. It, firstly, treats all instances as one initial cluster and iteratively splits the cluster into smaller ones. Secondly, this process continues until each instance belongs to one cluster or a predefined condition is met [5].

Density-Based Clustering (DBC) uses the notion of data distribution density to form dense regions as clusters [5]. This study selected ''Density-Based Spatial Clustering of Applications with Noise'' (DBscan), ''Ordering Points to Identify the Clustering Structure'' (Optics), and ''MeanShift'' clustering (Meanshift) [40].

DBscan clusters each instance based on their proximity and density. Although the performance of DBscan is related to the input parameters, DBscan starts by randomly selecting one data point and expands its neighbourhood until it contains the minimum number of data to form a cluster [5], [43], [44].

Optics builds a reachability graph that represents the density-based connectivity of the data points in the feature space. Then it creates a hierarchical clustering of the data points based on their reachability distances. This way allows Optics to identify clusters of different densities and shapes, while being robust to noise and outliers [40], [43], [44].

Meanshift assumes that datasets of different cluster classes conform to different probability density distributions. It finds clusters by iteratively moving the centroid of a kernel density estimate towards high-density regions by calculating the mean of offset of the current data point to find the direction where the density of any sample point increases the most [5], [40], [41].

2) SUPERVISED SOFTWARE DEFECT PREDICTION MODELS

We include two SSDPs based on two widely used and proven machine learning algorithms [14], [17], [45] in this study for the purpose of comparing the results with USDPs. They are as follows:

Random Forests (RF). There are many supervised clustering-based algorithms for software defect predictions, and RF is one of the most used and well-performed algorithms in this area [14], [17], [45], [46]. RF uses hyper bagging or averaging methods of the decision trees to give prediction results to improve performance and reduce the chance of over-fitting problems [14], [47].

K-Nearest Neighbours (KNN) is another type of SSDP. KNN is a commonly used supervised clustering-based algorithm for software defect predictions [17], [45]. It uses the distance function to measure the similarity between instances and assigns the most common class of processing instance's "k" nearest neighbours to the processing instance [47], [48], [49]. "k" is the most crucial hyper-parameter in KNN as this could lead to the model to under-fitting or over-fitting the data [47].

C. SUBJECT DATASET

This study has chosen publicly available and widely used software defect datasets of NASA[1] and PROMISE[2] data repositories as subject datasets [45], [50]. NASA datasets were created based on software metrics and process metrics, which are used to estimate software stages of enhancement, including Cohesion, Coupling and Complexity [16]. PROMISE data repository is designed for "research to predict effort and defect of a software and then developed to be model-based requirements engineering and value-based software engineering" [50]. Thus, they serve the intent of this study, and we have selected five datasets from NASA and fifteen datasets from PROMISE (Table 3).

D. ENVIRONMENT

This experiment environment included using "Intel Core" i7 CPU with 64 GB RAM, "Windows 10" operating systems, "Jupyter Notebook" development platform and "Python" programming language and related libraries, such as "scikit_learn", for machine learning algorithms. Standard parameters were applied to all SDPs, with the cluster number set to two, and the rest options were set to default values. As the datasets are high-dimensional, in order to improve the speed without compensating the data integrity, this experiment used "Principal component Analysis" (PCA) for dimensionality reduction and for preserving as much variability as possible, in order to give the best possible representation of the high-dimensional datasets [51]. We used the PCA function from the "scikit_learn" library to reduce datasets over thirty features into ten principal components to preserve over 85% of the variance in the data explained by ten principal components. Additionally, we used the PCA

[1]NASA Datasets were downloaded on 2022/07/11 and from https://github.com/klainfo/NASADefectDataset

[2]PROMISE datasets were downloaded on 2022/07/11 and from https://figshare.com/articles/dataset/Software_Defect_Prediction_Dataset/13536506

TABLE 2. List of subject machine learning algorithms.

Type	Machine Learning family	Machine learning algorithm	Abbreviation
Unsupervised	Model-Based Clustering (MBC)	Gaussian Mixture Model with "diagonal" covariance matrix option	GMM(diag)
		Gaussian Mixture Model with "spherical" covariance matrix option	GMM(spherical)
		Gaussian Mixture Model with "full" covariance matrix option	GMM(full)
		Bayesian Gaussian Mixture Model with "diagonal" covariance matrix option	BayGM(diag)
		Bayesian Gaussian Mixture Model with "spherical" covariance matrix option	BayGM(spherical)
		Bayesian Gaussian Mixture Model with "full" covariance matrix option	BayGM(full)
	Partition-Based Clustering (PBC)	K-Means model using "lloyd" EM algorithm	Kmean(full)
		K-Means model using "elkan" EM algorithm	Kmean(elkan)
		Mini-Batch K-Means	MiniKM
		Bisecting K-Means model using "lloyd" EM algorithm	BisectKM(full)
		Bisecting K-Means model using "elkan" EM algorithm	BisectKM(elkan)
	Graph-Theory-Based Clustering (GTBC)	Spectral Clustering	Spectral
		Affinity Propagation Clustering	Affinity
	Hierarchy-Based Clustering (HBC)	Agglomerative Clustering	Agglo
		Balanced iterative reducing and	Birch

TABLE 2. *(Continued.)* **List of subject machine learning algorithms.**

		clustering using hierarchies	
	Density-Based Clustering (DBC)	Density-Based Spatial Clustering of Applications with Noise	DBscan
		Ordering Points to Identify the Clustering Structure	Optics
		MeanShift clustering	Meanshift
Supervised		Random Forests	RF
		K-Nearest Neighbours	KNN

TABLE 3. **Information of subject datasets.**

Name of dataset	Repository	Number of features	Total number of instances
CM1	NASA	37	330
KC3	NASA	39	195
MW1	NASA	37	250
PC1	NASA	37	680
KC1	NASA	21	1165
ant-1.7	PROMISE	21	745
camel-1.6	PROMISE	21	965
data_arc	PROMISE	21	225
data_prop-6	PROMISE	21	645
data_ivy-2.0	PROMISE	21	355
data_redaktor	PROMISE	21	175
jedit-4.2	PROMISE	21	370
log4j-1.1	PROMISE	21	110
lucene-2.0	PROMISE	21	195
poi-2.0	PROMISE	21	315
synapse-1.2	PROMISE	21	260
velocity-1.6	PROMISE	21	230
xalan-2.4	PROMISE	21	725
xerces-1.2	PROMISE	21	440
xerces-1.3	PROMISE	21	455

function to reduce datasets with twenty-one features into five principal components to preserve over 85% of the variance in the data explained by five principal components.

E. MEASUREMENT

This experiment uses the ''Re-clustering percentage'' (RP) in the METTLE study to ''measure the inconsistency between a source output and its corresponding follow-up output'' [6]. Source outputs were the prediction outcomes of SDPs trained by the source datasets, and follow-up outputs were those of SDPs trained by follow-up datasets (section IV-F). If there is no violation between source output and follow-up output, RP is zero, and the SDP is insensitive to the associated MR, which means the MR could be used for validation.

Let source dataset be $D = [x_1, x_2 \ldots x_n]$ and follow-up dataset be $D_{mr} = [x_{m1}, x_{m2} \ldots x_{mn}]$. SDP was trained by D to be SDP_d. SDP was trained by D_{mr} as SDP_{mr}. Let test dataset be $D_{test} = [x_{t1}, x_{t2} \ldots x_{tz}]$. The prediction outcomes of SDP_d with D_{test} be $Y_d = [y_{dt1}, y_{dt2} \ldots y_{dtz}]$ and the prediction outcomes of SDP_{mr} with D_{test} be $Y_{mr} = [y_{mrt1}, y_{mrt2} \ldots y_{mrtz}]$. Let *diff* be the number of cases with cluster label differences between Y_{mr} and Y_d. RP is calculated as Equation (1).

$$RP = diff / D_{test} \tag{1}$$

The above calculation indicates that if *diff* is zero, which means there is no difference between SDP_d and SDP_{mr}, RP will be zero, which is considered as zero violation to this MR. If *diff* is larger than zero, then RP will be larger than zero, and violations have occurred. Regarding validating clustering systems, if RP is zero, no violation is caused by MR and vice versa.

This study also employs the ''Violation rate'' (VR) in the METTLE study to measure how many trials have violation occurred [6]. For example, if the RP of the trial is not zero, the trial is counted as a violated trial. This measurement is used with RP to indicate how sensitive the models react to MR. If both VR and RP are high, this could explain why the model is highly sensitive to the MR. On the other hand, if VR is high and RP is low, this could be interpreted as the model is moderately sensitive to the MR. Let the total number of test trials be T_{total} and the violated test trials be $T_{violated}$. VR is calculated as Equation (2).

$$VR = T_{violated} / T_{total} \tag{2}$$

As the subject datasets are labelled, we can employ the prediction accuracy rate (ACC) and Matthew correlation coefficient (MCC) [52]. These measurements can show any relationship between MRs and prediction accuracy. Let T_p be true positives that are the actual positives and correctly predicted, T_n be true negatives that are the actual negatives and correctly predicted, F_p be false positives that are the actual negatives and wrongly predicted as positives, and F_n be false negatives are the actual positives and wrongly predicted negatives. ACC is calculated as Equation (3), and MCC is calculated as Equation (4)

$$ACC = (T_p + T_n) / (T_p + T_n + F_p + F_n) \tag{3}$$

$$MCC = \frac{(T_p \times T_n - F_p \times F_n)}{\sqrt{((T_p + F_p) \times (T_p + F_n) \times (T_n + F_p) \times (T_n + F_n))}} \tag{4}$$

Apart from the above measurements, this study also examined the violation pattern related to MRs to understand better the underlying cause of the violation and the behaviour of SDPs. Inspired by the study of Xie et al. [6], this study has defined categories of violation patterns (Table 4).

TABLE 4. Violation pattern of relabelling after apply MRs.

Violation Pattern	Description
Reverse	Over 75% of results are relabelled
New Cluster created	New Cluster(s) is created
Noise	Only outliers are relabelled
No change	No relabeling
Border	Relabeling occurred on the edge between clusters
Merge and Split	Clusters are merged or cluster(s) is split
Less Cluster	Less cluster(s) is created

F. EXPERIMENT DESIGN

In the context of scenarios characterized by the scarcity of labelled datasets, USDPs are applied. Consequently, this study assumes that the testing datasets do not have labels, and it is difficult to validate with unlabelled testing datasets solely. Thus, we trained the SDPs with data generated by MRs, which should not affect the prediction outcomes (section III). We then used the outcomes from various trained SDPs to assess the consistency and validate the SDPs. Our experiment includes four steps as follows:

Step 1. We use twenty subject datasets (section IV-C) as source datasets and randomly select twenty per cent of the data of each source dataset to create test datasets for comparison used in step 4. Test datasets will not be used for training.

Step 2. We apply the proposed MRs to source datasets (section III-B) and generated twenty-one follow-up datasets for each subject dataset.

Step 3. SDPs train with source datasets and become ''Source SDPs''. SDPs train with follow-up datasets and become ''Follow-up SDPs''.

Step 4. ''Source SDPs'' make predictions on test datasets to create ''source outputs''. ''Follow-up SDPs'' make predictions on the same test datasets to create ''follow-up outputs''. Then, we compare source and follow-up outputs and analyze their differences. A failure is revealed if the relation is violated, which means differences occur between the outputs (Fig. 5).

G. RESULT AND DISCUSSION

We first studied the violations of all MRs with all subject SDPs to evaluate the effectiveness of our proposed model (RQ1). Then, we further analyzed the violation patterns to seek more understanding of the causes (RQ2). At last, we analyzed the performances between the SSDPs and USDPs to examine the relationship between labelled datasets and our proposed model (RQ3).

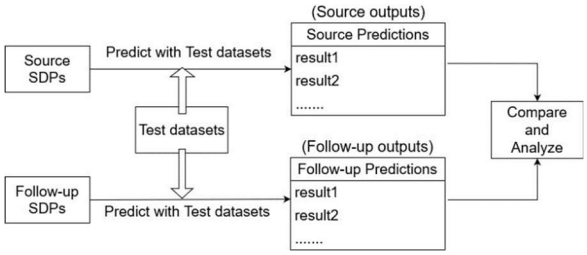


FIGURE 5. Illustration of the experiment.

1) RQ1: COULD THE GENERIC MT MODEL VALIDATE SDPS?

To answer this RQ, we reviewed the results of ''Zero violation with all MRs'', ''Zero violation with MR2'', ''Violated all MRs'', and ''Violated MR1, MR3, MR4 and MR5''.

Zero violation with all MRs. Table 6 shows the average RP, VR, ACC and MCC values of all SDPs in this study. Agglo and DBscan were two USDPs that had zero RP value (zero violation) with all twenty-one MRs and all twenty test datasets. These two USDPs are highly adaptable and robust to our model. However, the average ACC of DBscan was only 22.33%, and the average ACC of Agglo was 63.85%, with the third-highest MCC of all USDPs (Table 6). Agglo was the only SDP that made predictions better than random guesses (over 50%) and was robust to our model.

Agglo and DBscan use the ''bottom-up'' approach and do not randomly select an initial sample of points as they treat all single instances as clusters in the beginning [5], [53]. As long as the transformations do not alter the values of part of features and do not alter density distributions, Agglo and DBscan can produce consistent results.

MR1 reverses the order of data without changing the values of features or density distributions. MR2 only shrinks and expands values of all the features proportionally, so there is no change in values of part of features or density distributions of data. MR3 adds the datasets created by MR2 to source datasets to create different levels of densities. Since MR2 does not change the values of some features or density distributions, adding them to the source data should not affect their performance. MR4 removes highly correlated features, which indicates that an attribute can be considered redundant from datasets. As they are considered redundant, removing them should not affect the values of overall features or density distributions of data. MR5 adds different combinations of outliers to data, and this study created outliers of at least three standard deviations of all values of all features. Agglo and DBscan can detect added outliers, making both algorithms insensitive to MR5. Therefore, the proposed model can serve as a validation tool for SDPs using Agglo and DBscan with high-dimensional data.

Zero violation with MR2. Table 7 shows that twelve out of twenty SDPs did not violate MR2. Fig. 6a shows MR2 had a much lower average RP than the other MRs, as the maximum average RP of MR2 were below all the lower quartiles of MR1, MR3, MR4 and MR5. To examine whether

the differences were significant, we ran multiple paired t-tests with a significance level of 0.01 for the mean of differences in average RP between Agglo (zero-violated SDP) and other SDPs, which violated MR2. Table 5 shows that only the mean of differences in average RP of Affinity was significantly different from Agglo. Thus, twelve SDPs had zero violation with MR2, and seven SDPs did not significantly violate MR2; this concludes that MR2 is a must property of the proposed model.

TABLE 5. Multiple paired t-tests with significance level at 0.01 for mean of difference in average RP values of MR2 between Agglo and Affinity which were significantly different.

Paired T-test group	MR	P(T<=t) 2 tail (alpha=0.01)	Reject H0: mean diff. = 0 (p-value < 0.01)
Agglo vs Affinity	MR2.1	0.0099	YES
	MR2.2	0.0062	YES
	MR2.3	**0.0268**	**NO**
	MR2.4	0.0028	YES

Violated all MRs. Three SDPs violated all MRs and they are Spectral, Optics and Affinity.

Spectral violated only the test dataset of MW1, and the average RP in each MR is from 36% to 63%. We ran multiple paired t-tests with a significance level of 0.01 for the mean of differences in RP between Agglo and Spectral, and test results show the differences were not significant with all MRs. Thus, we conclude that these violations are not related to MRs.

Optics violated all MRs with seven test datasets; each test dataset was at the same RP as all MRs (Table 8). So, we conclude that the seven datasets cause the violations.

Affinity was sensitive to all MRs. Fig. 6b shows Affinity violated all MRs at an extensive range of average RP. We further studied the violation patterns of Affinity. We found that the predictions of all trials are either classifying all instances as class zero or class one, which leads to either 100% or 0% RP. That explains why most of the interquartile ranges of MRs in Fig. 6b are 100%.

Violated MR1, MR3, MR4 and MR5. Fifteen SDPs violated all MRs except MR2, including MBC algorithms, PBC algorithms, Meanshift, Birch, K-Nearest Neighbour, and Random forest.

Meanshift had moderate average RP (Fig. 6c). We studied the violation patterns of Meanshift and found that most violation patterns had new clusters created for predictions after data transformation. Thus, except MR2, other MRs caused Meanshift to consider instances in the test dataset with new clusters.

Birch was sensitive to all MRs except MR2 (Fig. 6d). We studied the result of Birch and found most RP of all trials are either below twenty per cent or above eighty per cent, so this explains why most of the interquartile ranges are large (Fig. 6d). We further studied the violation patterns of Birch and found that most violation patterns are ''Reverse'', which means, except MR2, data transformations caused Birch to change predictions.

The average RP of K-Nearest Neighbour and Random Forrest are moderate (Fig. 6e and Fig. 6f). We discovered their violation patterns were mostly ''Border'' and ''Merge and Split'' cases, which means the data transformations changed their predictions on instances that are on the decision edge of class zero and class one. We ran multiple paired t-tests with a significance level of 0.01 for the mean of differences in RP between Agglo and RF and between Agglo and KNN. The results showed that all the differences are significant with MR1, MR3, MR4 and MR5.

Kmean(full) (Fig. 6g), Kmean(elkan), BisectKM(full) and BisectKM(elkan) were sensitive to MR1, MR3, MR4 and MR5. We discovered that Kmean(full), Kmean(elkan), BisectKM(full) and BisectKM(elkan) had mostly ''Reverse'' violation patterns with MR1, MR3 and MR5, and had ''Border'' and ''Reverse'' violation patterns with MR4. We also ran multiple paired t-tests with a significance level of 0.01 for the mean of differences in RP between Agglo and PBC algorithms, and all the differences are significant, which matched the violation patterns, except MiniKM in MR4 (Table 9).

MiniKM had a lower average RP with MR4 than with other MRs (Fig. 6h). It had mostly ''Reverse'' violation patterns with MR1, MR3 and MR5 and ''No change'' patterns with MR4.1 and MR4.2. These results align with MiniKM, which did not significantly violate MR4 (Table 9). Thus, PBC algorithms were sensitive to MR1, MR3 and MR5, and MiniKM did not significantly violate MR2 and MR4.

GMM(diag), GMM(spherical) (Fig. 6i), GMM(full), BayGM(diag) and BayGM(full) were sensitive to MR1, MR3, MR4 and MR5. We discovered that GMM(diag), GMM(spherical), and GMM(full) had mostly ''Merge and split'' and ''Reverse'' violation patterns with MR1, MR3, MR4 and MR5. BayGM(diag) mostly had ''No Change'' and ''Border'' patterns with MR4, and ''Merge and split'' patterns with MR1, MR3, and MR5. BayGM(full) mostly had ''Merge and split'' patterns with MR1, MR3, MR4, and MR5. We ran multiple paired t-test results for the mean of differences in RP between Agglo and MBC algorithms. All the differences are significant, even with BayGM(diag), except BayGM(spherical) with MR1 and MR4 (Table 10).

BayGM(spherical) had lower average RP with MR1, MR4.1 and MR5.1 than with other MRs (Fig. 6j). BayGM(spherical) has mostly ''Merge and split'' patterns with MR3 and MR5, mostly ''No Change'' and ''Border'' patterns with MR1 and MR4. The patterns match the multiple paired t-test results (Table 10), which showed that BayGM(spherical) did not significantly violate MR1 and MR4. Thus, MBC algorithms are sensitive to MR1, MR3, and MR5, except MR2 and BayGM(spherical), which did not significantly violate MR1, MR2, and MR4.

Answering RQ1: Agglo and DBscan did not violate any MR of the generic MT model. Nineteen SDPs did not significantly violate MR2, four SDPs did not significantly violate MR4, and three SDPs did not significantly violate MR1. Thus, results conclude that the generic MT model can be used as a validation tool for SDPs, and MR2 is a must property

TABLE 6. Average RP, VR, ACC and MCC values of SDPs with all MRs and all test datasets.

	GMM (diag)	GMM (sph.)	GMM (full)	Bay-GM (diag)	Bay-GM (sph.)	Bay-GM (full)	Kmean (full)	Kmean (elkan)	Mini-KM	Bisect-KM (full)	Bisect-KM (elkan)	Mean-shift	Spec-tral	Affini-ty	Agglo	Birch	Optics	DBscan	KNN	RF
RP	39.2%	42.6%	39.6%	23.4%	21.4%	37.5%	34.1%	34.1%	32.6%	36.1%	36.3%	6.1%	2.4%	53.4%	**0.0%**	36.8%	7.9%	**0.0%**	8.8%	10.5%
VR	80.5%	75.0%	75.2%	78.3%	74.8%	79.3%	70.9%	71.0%	71.0%	71.4%	71.2%	79.5%	5.0%	53.5%	**0.0%**	67.0%	30.0%	**0.0%**	78.0%	71.5%
ACC	52.6%	53.7%	52.1%	61.9%	65.5%	52.9%	61.0%	60.9%	60.4%	60.0%	59.6%	75.4%	65.4%	36.7%	**63.8%**	60.6%	21.5%	**22.3%**	84.7%	82.4%
MCC	3.7%	4.7%	5.3%	13.5%	15.6%	10.6%	8.7%	8.6%	6.5%	7.7%	7.7%	14.8%	10.4%	0.0%	**11.4%**	0.5%	3.0%	**5.7%**	35.6%	29.2%

TABLE 7. Average RP values of MR2 of SDPs with all test datasets.

MR	GMM (diag)	GMM (sph.)	GMM (full)	Bay-GM (diag)	Bay-GM (sph.)	Bay-GM (full)	Kmean (full)	Kmean (elkan)	Mini-KM	Bisect-KM (full)	Bisect-KM (elkan)	Mean-shift	Spec-tral	Affini-ty	Agglo	Birch	Optics	DBscan	KNN	RF
MR2.1	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**1.00%**	**1.00%**	0.00%	**4.00%**	**1.97%**	0.00%	**2.68%**	**28.00%**	0.00%	0.00%	**9.25%**	0.00%	0.00%	**0.21%**
MR2.2	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**2.00%**	**2.00%**	0.00%	**3.00%**	**3.98%**	0.00%	**2.72%**	**32.00%**	0.00%	0.00%	**9.25%**	0.00%	0.00%	**0.09%**
MR2.3	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**1.00%**	0.00%	**2.00%**	**1.03%**	0.00%	**3.16%**	**22.00%**	0.00%	0.00%	**9.25%**	0.00%	0.00%	**0.09%**
MR2.4	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**1.00%**	**2.00%**	0.00%	**4.00%**	**2.07%**	0.00%	**2.32%**	**37.00%**	0.00%	0.00%	**9.25%**	0.00%	0.00%	**0.16%**

TABLE 8. Average RP of optics with violated test datasets.

Test dataset	MR1	MR2.1	MR2.2	MR2.3	MR2.4	MR3.1	MR3.2	MR3.3	MR3.4	MR3.5	MR3.6	MR3.7	MR4.1	MR4.2	MR5.1	MR5.2	MR5.3	MR5.4	MR5.5	MR5.6	MR5.7
CM1	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%
PC1	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%
KC1	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%
ant-1.7	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%
camel-1.6	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%
data_prop-6	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%
xalan-2.4	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%

TABLE 9. Multiple Paired t-tests with significance level at 0.01 for mean of difference in RP values of MR1, MR3, MR4 and MR5 between Agglo and MiniKM.

Paired T-test group	MR	P(T<=t) 2 tail (alpha=0.01)	Reject H0: mean diff. = 0 (p-value < 0.01)
Agglo vs MiniKM	MR1	0.000404	Yes
	MR3.1	0.000018	Yes
	MR3.2	0.000072	Yes
	MR3.3	0.000416	Yes
	MR3.4	0.000480	Yes
	MR3.5	0.000080	Yes
	MR3.6	0.000098	Yes
	MR3.7	0.000608	Yes
	MR4.1	**0.064163**	**NO**
	MR4.2	**0.010795**	**NO**
	MR5.1	0.000048	Yes
	MR5.2	0.000063	Yes
	MR5.3	0.000014	Yes
	MR5.4	0.000042	Yes
	MR5.5	0.000058	Yes
	MR5.6	0.000036	Yes
	MR5.7	0.000024	Yes

TABLE 10. Multiple paired t-tests with significance level at 0.01 for mean of difference in RP values of MR1, MR3, MR4 and MR5 between Agglo and BayGM(spherical).

Paired T-test group	MR	P(T<=t) 2 tail (alpha=0.01)	Reject H0: mean diff. = 0 (p-value < 0.01)
Agglo vs BayGM (spherical)	**MR1**	**1.35E-01**	**NO**
	MR3.1	8.95E-06	Yes
	MR3.2	1.58E-04	Yes
	MR3.3	5.03E-04	Yes
	MR3.4	9.63E-06	Yes
	MR3.5	2.76E-08	Yes
	MR3.6	2.25E-08	Yes
	MR3.7	2.58E-11	Yes
	MR4.1	**7.09E-02**	**NO**
	MR4.2	**2.50E-02**	**NO**
	MR5.1	8.60E-03	Yes
	MR5.2	3.42E-03	Yes
	MR5.3	1.95E-03	Yes
	MR5.4	1.32E-03	Yes
	MR5.5	4.07E-03	Yes
	MR5.6	3.61E-03	Yes
	MR5.7	1.52E-03	Yes

of the generic MT model. In addition, the average ACC of DBscan was below 50% (Table 6), so Agglo is the only SDP insensitive to all MRs and can make predictions better than random guesses (over 50%) in the study.

2) RQ2: WHAT ARE THE UNDERLYING CAUSES OF VIOLATIONS?

This section discusses the three underlying causes of violations with the violated SDPs.

Caused by characteristics of data. Results showed that the characteristics of data cause violations for Spectral and Optics. Spectral had violations with all MRs in the test dataset generated from dataset MW1 only. We compared dataset MW1 with another two zero-violated datasets, which were also from NASA and had the same features and similar size, PC1 and CM1 (Table 3). We ran multiple paired t-tests with a significance level 0.01 between those three source datasets.

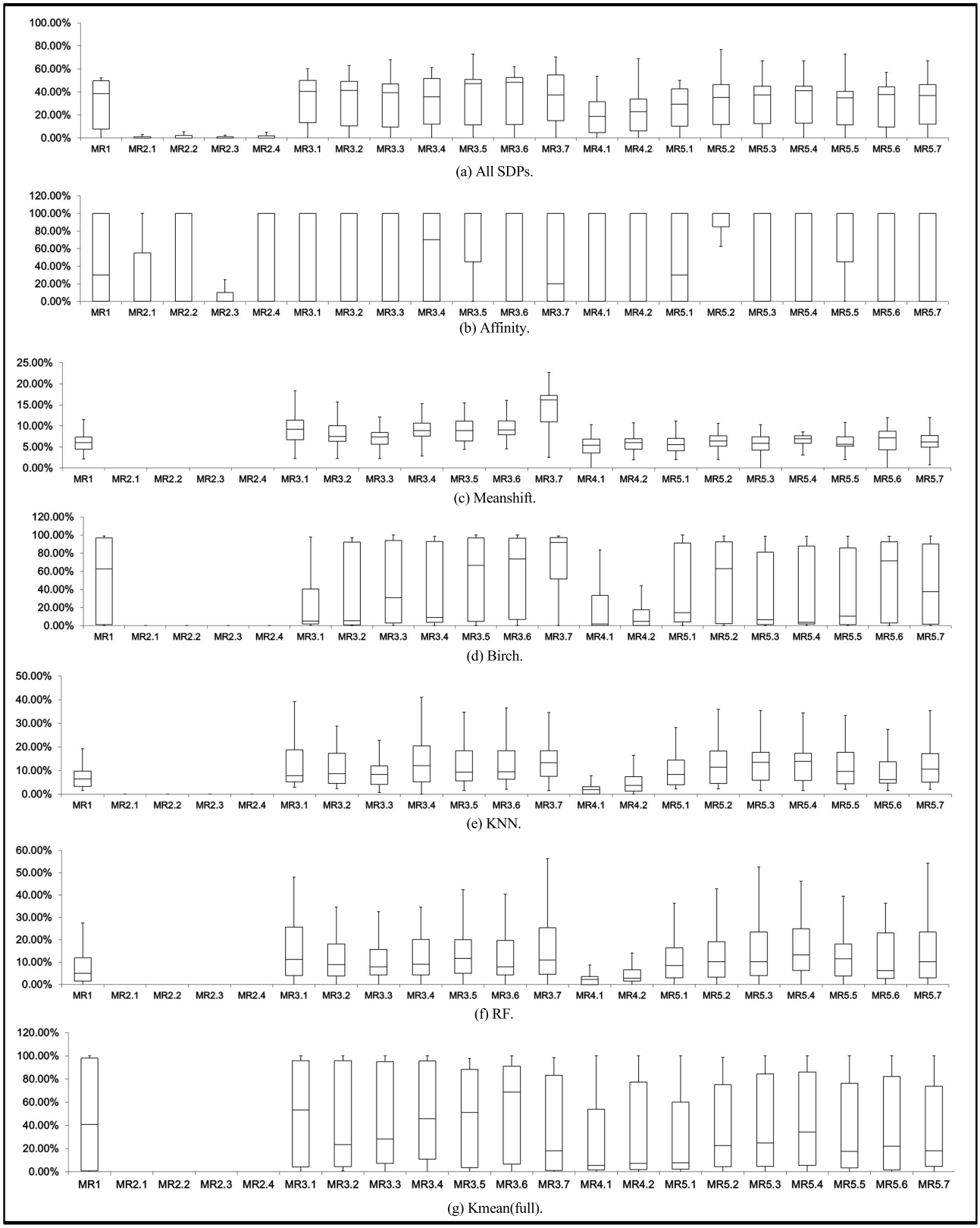


FIGURE 6. Box and whisker diagram of average RP values of all datasets with (a) all SDPs. (b) Affinity algorithm. (c) Meanshift algorithm. (d) Birch algorithm. (e) KNN algorithm. (f) RF algorithm. (g) Kmean(full) algorithm.

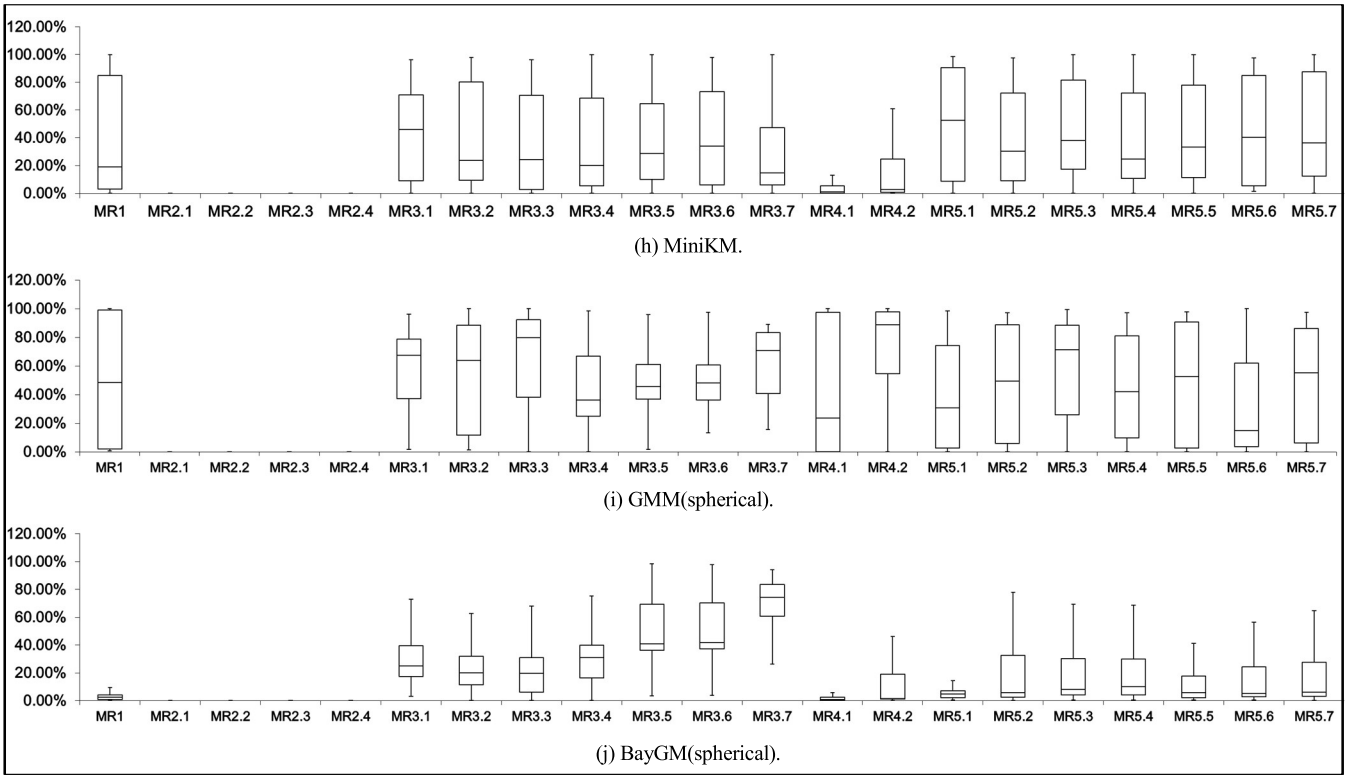


FIGURE 6. *(Continued.)* **Box and whisker diagram of average RP values of all datasets with (h) MiniKM algorithm. (i) GMM(spherical) algorithm. (j) BayGM(spherical) algorithm.**

We found that the values of the twelve features are significantly different between PC1 and MW1, between CM1 and MW1, and not significantly different between PC1 and CM1. In other words, the values of these twelve features in MW1 significantly differ from those in PC1 and CM1. Table 11 shows the statistic of the most different feature between "CM1 and MW1", and "PC1 and MW1". The ranges of values in PC1 and CM1 are more extensive than in MW1, which means the variations of values in PC1 and CM1 are more significant than those in MW1 (Table 11).

TABLE 11. The statistic of the most different feature "HALSTEAD_ DIFFICULTY" between CM1, PC1 and MW1.

	HALSTEAD_ DIFFICULTY (PC1)	HALSTEAD_ DIFFICULTY (CM1)	HALSTEAD_ DIFFICULTY (MW1)
Mean	20.3746	20.0586	12.3334
Standard Error	0.6747	0.8351	0.5187
Median	15.8400	15.2000	10.2100
Mode	16.0000	12.0000	5.5000
Standard Deviation	**17.5820**	**15.1011**	**8.2006**
Sample Variance	309.1259	228.0419	67.2501
Range	**268.6600**	**124.2300**	**45.0300**
Minimum	2.0000	2.3300	2.5000
Maximum	**270.6600**	**126.5600**	**47.5300**
Count	679	327	250

As the variations of values in the twelve features of MW1 were smaller than PC1 and CM1, the dataset of MW1 was more difficult for Spectral to form an accurate "similarity matrix of the data to construct a connected graph and treating clustering as a graph partitioning problem" [5] than PC1 and CM1. If the ranges and standard deviations of feature values in the datasets are too small, the distances between data points will be very short. This could lead Spectral to consider those data points as similar and belonging to the same cluster, when in fact they are different. Thus, we conclude that Spectral violated all MRs with the test dataset of MW1 because the variations of values of features in dataset MW1 are too small for Spectral to form an accurate similarity matrix for classification.

Optics had violations with seven test datasets and their average RP were the same with all MRs in each of the seven test datasets (Table 8). Violated test datasets are generated from datasets CM1, PC1, KC1, ant-1.7, camel-1.6, data_prop-6 and xalan-2.4 and six of them were the largest datasets in the study (Table 3). CM1 was the smallest of the seven violated test datasets, with only 330 instances and 37 features (Table 3). Table 8 shows that the RP of CM1 was only 3.03% across all MRs, which is also the smallest, and the rest are from 24.83% to 45.6%.

PROMISE dataset xerces-1.3 has fewer instances than PROMISE dataset data_prop-6 (Table 3), and Optics has no violation with the test dataset xerces-1.3. NASA datasets KC3 and MW1 have fewer instances than CM1, PC1 and KC1 (Table 3), and Optics also has no violation with the test

datasets of KC3 and MW1. We found that the size of datasets could affect Optics' sensitivity to MRs, and this is reasonable for Optics, which stores an order list for each object based on its density. Suppose the datasets contain a large number of high-dimensional instances. In that case, it will be hard to create index structures to efficiently support the hypersphere range queries needed by the Optics algorithm [43], [44], [54]. Another potential factor to consider is the shape of the data distribution. Optics tend to cluster data better when the data distribution has a spherical shape, as opposed to other shapes [5]. After performing data transformations with MRs, the shape of the data distribution may change to a form that Optics cannot effectively handle. Therefore, violations of Optics may be caused by both the size of the datasets and the shapes of the data distributions.

Further investigations are necessary to identify the root causes related to the data characteristics that lead to violations for Spectral and Optics.

Caused by configurations of algorithms. Fig. 6b shows Affinity violated all MRs with all the datasets, so data should not cause the violations. Although Affinity does not need to set the number of clusters to predict, it needs to set two parameters. They are ''damping'' and ''preference'' [55]. ''damping'' is set to avoid numerical oscillations in the updating process, and the range is from 0.5 to 1. This study sets 0.75 for damping, and the ''preference'' factor is set to ''-1e05''. These numbers made Affinity create two clusters for test datasets of source datasets after a few trials. However, if the follow-up SDPs did not suit these configurations, the predictions of follow-up Affinity would be affected [56], [57]. Therefore, we conclude that the configurations of Affinity cause the violation, and our generic MT model is unsuitable for Affinity as this is a validation study, and no cross-validation of configurations should be used.

Meanshift shifts each instance iteratively in the dataset until the top of its kernel density estimation surface reaches the nearest peak [5]. However, Meanshift relies on the bandwidth configurations for better clustering results [41]. We used the ''estimate_bandwidth'' function of the ''scikit-learn'' library with configurations ''quantile = 0.5'' and ''n_sample = 500'' to get the bandwidth. These configurations did not work with different follow-up datasets, except MR2. Therefore, results show that only MR2 can validate Meanshift without tuning bandwidth configurations. Further investigations are necessary to identify the influence of configuration on Affinity and Meanshift that caused the violation.

Caused by characteristics of algorithms. Birch treats all instances as an initial cluster at the beginning and then successively splits the cluster into smaller ones until each instance belongs to one cluster or a predefined condition is met [5]. However, Birch is sensitive to the order of the data [30], so Birch will give different clustering results for MR1 (reversing the order of data), MR3 and MR5 (both are adding more data to change the order). In addition, Birch works well with the spherical shape of clusters [30], and MR4 (remove features) can change the shape of data distributions, which affects Birch's predictions.

MiniKM is insensitive to MR4.1 (Table 9). This could be explained by the fact that ''mini-batches tend to have lower stochastic noise than individual examples in stochastic gradient descent and allow convergence to better solutions'' [58]. However, the characteristics of PBC algorithms are sensitive to noisy data and outliers [30] and randomly select instances as the initial center points [5]; therefore, it is rational that MR1 (reversing order of data) could affect the selection of the initial center points for PBC, MR3 (changing the density of data), MR4 (removing features) and MR5 (adding outliers) could make the data noisy to PBC.

The superior performance of BayGM compared to GMM (Table 6) can be attributed to BayGM's utilization of smaller batches or truncated distribution, which tends to result in less noise and more accurate solutions [58]. In addition, BayGM(spherical) did not significantly violate MR1 and MR4 (Table 10), which could be explained by the shapes of the clusters identified by source BayGM(spherical) and follow-up BayGM(spherical) of MR1 and MR4 are similar in spherical shapes. This similarity in shapes led to consistent prediction outcomes. Conversely, MBC algorithms are sensitive to noisy data and outliers [59]. Since MR3, and MR5 can make data noisy, it is rational that MBC is sensitive to them. In addition, MBC algorithms use the EM algorithm to initialize the model, which obtains an initial estimate of the model parameters randomly [31], and MR1 reverses the order of data can affect the selection of the initial estimates, which makes MBC sensitive to MR1. Thus, MBC algorithms are sensitive to MR1, MR3, and MR5 because of the characteristics of the algorithms.

The average RP of RF were lower than half of the SDPs' average RP, and its average ACC and MCC were among the second highest (Table 6). In addition, RF had a high average VR and a low average RP, which shows that RF is moderately sensitive to MRs (Table 6). However, RF has several hyper-parameters that influence the performance [60], and as RF ensembles several decision trees and each of them learns from a randomly selected sample of data [48], resulting in prediction inconsistencies. It is because MR1 reverses the order of data, which can affect the selection of samples for RF. MR3, MR4 and MR5 can make the data noisy, which can also affect the selection of samples for RF. Although further tunings for the hyper-parameters of RF can improve the performance of RF, this study prioritizes validating SDPs and the successful tuning of parameters could mask actual errors in program. Thus, due to its characteristics and configurations, RF is sensitive to MR1, MR3, MR4 and MR5.

KNN demonstrated the highest average ACC and MCC, high average VR and low average RP, indicating its moderate sensitivity to MRs (Table 6). However, KNN is sensitive to noise in the dataset. Standardization and normalization of the dataset must be implemented for better

performance [54], [61]. The inconsistency of KNN could be explained as MR3, MR4, and MR5 can make the data noisy. Furthermore, KNN is sensitive to MR1 because the standardization process involves standardizing the order of data to make KNN stable. KNN also suffers from the influence of ''k'' hyper-parameter [47]. We chose seven as the ''k'' value because we used dataset ''log4j-1.1'', which is the smallest in size (Table 3), to estimate that each cluster should at least have seven neighbours, in order to make KNN works for all the datasets. Although further tunings for parameters can improve the performance of KNN, this study focuses on validating SDPs and excessive tuning of parameters may obscure actual errors in program. Thus, due to its characteristics and configurations, KNN is sensitive to MR1, MR3, MR4, and MR5.

Answering RQ2: After analyzing the violated SDPs, we found three underlying causes of violations: characteristics of data, configurations of algorithms and characteristics of algorithms. These causes show the importance of understanding the data and the algorithms before applying the proposed model.

3) RQ3: WHAT ARE THE DIFFERENCES IN APPLYING GENERIC MT MODEL TO VALIDATE USDP AND SSDP?

This section discusses the differences in applying MT to validate USDPs and SSDPs. The main difference between them is that SSDP uses labelled datasets, and USDP does not, so this section focuses on the effect of labelled datasets.

MR2. All SSDPs and seventeen USDPs did not significantly violate MR2 (Tables 5 and 7), and MR2 shrinks and expands all feature values in datasets proportionally without changing the labels. Thus, the labelling of MR2 follow-up datasets did not affect SSDPs, and this shows that labelled data does not make a difference in applying our MR2 to validate USDPs and SSDPs.

Correlation between prediction accuracies and inconsistencies. Results show that SSDPs had the highest average ACC and MCC out of all SDPs (Table 6), which is expected as SSDPs have labelled data to revise their predictions. Even Agglo and DBscan had zero violation, their average ACC and MCC were lower than SSDPs (Table 6). In addition, Table 6 shows that the average RP of SSDPs were lower than thirteen USDPs; however, the average VR of SSDPs were over 70%, similar to other highly violated USDPs. We also ran a correlation analysis with average values of ACC, MCC, VR and RP of all SDPs. Table 12 shows that ''ACC and MCC'' are correlated, and ''VR and RP'' are correlated; however, ''ACC and RP'', and ''MCC and RP'' are not highly correlated. These results show that prediction inconsistencies, such as VR and RP, are independent of prediction accuracies for SSDPs and USDPs. Thus, labelled datasets can only improve prediction accuracies and make no difference in applying our generic MT model to SSDPs and USDPs, focusing on prediction inconsistencies.

Answering RQ3: Results show that labelled datasets make no difference when used in our generic MT model for SSDPs and USDPs, as they are insensitive to MR2. Although SSDPs have higher prediction accuracies, which can be affected by labelled data, the prediction inconsistencies are independent of prediction accuracies.

TABLE 12. Correlation analysis based on average values of all SDPs.

	ACC	MCC	VR	RP
ACC	1.00			
MCC	0.72	1.00		
VR	0.43	0.19	1.00	
RP	-0.14	-0.50	0.61	1.00

4) FURTHER ANALYSIS

The results show that although the model is generic because users can apply this model with any numeric data feature studied by machine learning algorithms, Nineteen SDPs did not significantly violate MR2. Thus, MR2 is the generic MR that can serve as a validation tool for any numeric feature of data studied by thirteen machine learning algorithms (except Affinity), even without prior knowledge of the validating systems.

Agglo is the only SDP insensitive to all MRs and can make predictions better than random guesses (over 50%) in the study. Thus, the whole model can be applied as a validation tool for systems that employ the Agglo algorithm, even without prior knowledge of the validating systems.

As the above results show, our generic MT model can be applied without prior knowledge of the validating systems, and our model can be used for continuous monitoring of machine learning based systems. If there are any updates to the machine learning based systems, our model can validate the updated systems. Moreover, if the system violates the MR of our model, errors in the updated systems can be detected at an early stage.

The experiment results did not show any trend from MR3.1 to MR3.7, which did not show any relationship between the density level and the level of re-clustering. Similar to MR5.1 to MR5.7, the results did not show any relationship between the number of outliers and the level of re-clustering. Although four SDPs did not violate MR4.1 and two SDPs did not violate MR4.2, results did not show any relationship between the number of removed features and the level of re-clustering.

V. THREATS TO VALIDITY

This section discusses the threats to validity that are faced by this study.

A. INTERNAL VALIDITY

One internal threat to the experiment is the randomness of clustering systems, as this can cause result variations. This study fixed the random seed for subject SDPs, making the results reproducible in each execution run. Another internal threat to our experiment is related to parameter configuration. Standard parameters, which are mentioned in

section IV-D, were applied to all SDPs. We built SDPs using the ''scikit-learn'' library of Python and created the follow-up datasets using Microsoft Excel. These tools and technologies should be considered reliable since they have been widely used. White-box testing has also been conducted so the source codes are error-free. Finally, this study focused on the relationship between MRs and measurements mentioned in section IV-E. The prediction outcomes of all SDPs were listed and checked to ensure that the measurements mentioned in section IV-E were calculated accurately.

B. EXTERNAL VALIDITY

METTLE and μMT were proven models [6], [12], [13], and this study is based on these models to develop the generic MT model. The potential threat to the external validity of this study is low. Another external threat is the generality of our model. As we applied DMOs of μMT [13] and proposed DMOs to the implementation of the proposed model, the DMOs are all generic and can be applied to any numeric datasets. This study has chosen twenty software defect datasets of NASA and PROMISE, which vary in size and features (Table 3). Twenty SDPs using fourteen different algorithms were employed. This study ran five trials for each SDP with each MR and dataset (section IV-F). The results should be generalized. Although the datasets used for assessment may vary case by case, the subject SDPs assessed and evaluated by our approach are rather general. Thus, we argue that the effect of different datasets on the effectiveness of the proposed model should be small.

C. CONSTRUCT VALIDITY

This study follows Xie et al. RP [6] to measure the effectiveness of our generic MT model. The independent variables of this study are prediction outcomes of the same test datasets of different SDPs, which were trained by different follow-up datasets. The dependent variable is the differences between the independent variables and the prediction outcomes of the same test datasets of SDPs trained by the source dataset, which is the RP. As this study focuses on validating SDPs, RP could show the violations. Suppose there is no violation between source prediction outcomes and follow-up prediction outcomes, which means no inconsistent prediction between them. In that case, RP is zero, and the SDP is insensitive to the related MR. That means the MR could be used for validation. As SDP is combined with programs and trained data, a violation with the MR could reveal an error in programs because the related SDP should be insensitive to the MR. That can correctly measure the intent of this study. To sum up, the construct validity of the independent variables and dependent variables of this article should be acceptable.

VI. CONCLUSION

Although USDP is a potential solution for insufficient labelled defect training datasets for SSDP, it is still faced with the test oracle problem. MT is a widely used technique to alleviate the test oracle problem; however, identifying MRs is challenging as these expected relationships are mainly created by experts or users. Thus, this study proposed a generic and adaptable MT model to alleviate the challenge of identification of MR. We propose a generic MT model that adapted twenty-one generic MRs from five generic types of MRs of METTLE [6] mapping with DMOs of μMT [12], [13] and proposed DMOs. We experimented with the proposed model with twenty publicly available high-dimensional software defect datasets of NASA and PROMISE, and twenty SDPs implementing fourteen different machine learning algorithms.

In conclusion, this study proved that the proposed generic MT model is a valuable validation tool for SDPs, even with limited knowledge of validating targets and no inputs from experts or users. Firstly, Agglo and DBscan did not violate any MR, which indicated that using a ''bottom-up'' clustering machine learning approach with no random initial sample selection is highly adaptable and robust to the proposed model. Thus, we recommend using the proposed model for validating SDPs using Agglo and DBscan. Secondly, MR ''Shrinkage and Expansion'' (MR2) is the most generic MR and a necessary property, as nineteen SDPs did not significantly violate this MR. Thirdly, all the violations were caused by characteristics of data, configurations of algorithms and characteristics of algorithms. Thus, future applications of the proposed model must consider these three factors. Fourthly, the prediction inconsistencies are independent of prediction accuracies, so labelled datasets make no difference in using our model to validate SSDPs and USDPs. Lastly, as the results show, our generic MT model can be applied without prior knowledge of validating machine learning based SDPs; our model can be used to monitor machine learning based systems continuously.

A. FUTURE DIRECTIONS

Some aspects could extend this research. Firstly, future studies can replicate the experiment of this study on different domains to explore relationships between characteristics of algorithms and MRs, and to explore how to define good generic MRs for validating different machine learning algorithms. Secondly, future studies could focus on exploring MR2 for a more precise generic MT model. Thirdly, future studies can extend this model to incorporate the latest machine learning algorithms, such as neural networks and transformer-based algorithms, to further improve our generic MT model regarding characteristics of data, configurations of algorithms, and characteristics of algorithms. Fourthly, since our generic MT model can be applied without prior knowledge of the systems, future studies can focus on applying it to areas that require continuous monitoring. Machine learning models can be trained with source and MR-generated datasets to create and update different machine learning models continuously. Prediction results from different trained machine learning models can be continuously compared to monitor inconsistencies in real-time. Fifthly, this research has

experimented with the generic model on high-dimensional datasets with a binary classification problem, and the results show that our approach can effectively handle algorithms that classify test data with a new cluster (new label). Therefore, it is plausible for future studies to replicate the experiment on high-dimensional datasets with multi-label classification problems; however, if future studies intend to replicate this experiment on larger high-dimensional datasets with a multi-label classification problem, researchers must carefully consider the sufficiency of computer resources. Sixthly, future studies could expand our model to experiment with validating various software systems using machine learning algorithms, such as autonomous car systems. This could also involve exploring different software issues, like validating software quality prediction, to showcase the ease of applying our model and its versatility. Seventhly, cross-validation studies can be conducted on our generic model using a variety of algorithm configurations, dimensional datasets, distribution shapes of datasets, dataset sizes, and variations in feature values. This approach will enable us to identify the causes that led to violations for different machine learning algorithms with our generic model. Lastly, more studies are needed to compare our model with other MT models to explore further improvements.

B. LIMITATIONS

The proposed model could only apply to features with numeric values, and this study only experimented on a two-labelled classification problem and employed two supervised software defect prediction models using machine learning algorithms. Additionally, due to the limited computer resources, this study only selected subject datasets of less than 1200 instances (section IV-C).

COMPETING INTERESTS

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this article.

ACKNOWLEDGMENT

The authors declare that they have not used any generative artificial intelligence (AI) and AI-assisted technologies in the writing process.

REFERENCES

[1] R. Kumar, A. Chaturvedi, and L. Kailasam, "An unsupervised software fault prediction approach using threshold derivation," *IEEE Trans. Rel.*, vol. 71, no. 2, pp. 911–932, Jun. 2022, doi: 10.1109/TR.2022.3151125.

[2] Z. Wan, X. Xia, A. E. Hassan, D. Lo, J. Yin, and X. Yang, "Perceptions, expectations, and challenges in defect prediction," *IEEE Trans. Softw. Eng.*, vol. 46, no. 11, pp. 1241–1266, Nov. 2020, doi: 10.1109/TSE.2018.2877678.

[3] H. Wang, W. Zhuang, and X. Zhang, "Software defect prediction based on gated hierarchical LSTMs," *IEEE Trans. Rel.*, vol. 70, no. 2, pp. 711–727, Jun. 2021.

[4] J. Xu, J. Ai, J. Liu, and T. Shi, "ACGDP: An augmented code graph-based system for software defect prediction," *IEEE Trans. Rel.*, vol. 71, no. 2, pp. 850–864, Jun. 2022, doi: 10.1109/TR.2022.3161581.

[5] Z. Xu, L. Li, M. Yan, J. Liu, X. Luo, J. Grundy, Y. Zhang, and X. Zhang, "A comprehensive comparative study of clustering-based unsupervised defect prediction models," *J. Syst. Softw.*, vol. 172, Feb. 2021, Art. no. 110862, doi: 10.1016/j.jss.2020.110862.

[6] X. Xie, Z. Zhang, T. Y. Chen, Y. Liu, P.-L. Poon, and B. Xu, "METTLE: A METamorphic testing approach to assessing and validating unsupervised machine learning systems," *IEEE Trans. Rel.*, vol. 69, no. 4, pp. 1293–1322, Dec. 2020.

[7] D. Xiao, Z. Liu, Y. Yuan, Q. Pang, and S. Wang, "Metamorphic testing of deep learning compilers," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 6, no. 1, pp. 1–28, Feb. 2022, doi: 10.1145/3508035.

[8] C.-A. Sun, B. Liu, A. Fu, Y. Liu, and H. Liu, "Path-directed source test case generation and prioritization in metamorphic testing," *J. Syst. Softw.*, vol. 183, Jan. 2022, Art. no. 111091, doi: 10.1016/j.jss.2021.111091.

[9] Z. Ying, D. Towey, A. Bellotti, Z. Q. Zhou, and T. Y. Chen, "Preparing SQA professionals: Metamorphic relation patterns, exploration, and testing for big data," in *Proc. Int. Conf. Open Innov. Educ. (ICOIE)*, 2021, pp. 22–30.

[10] M. Zhang, J. W. Keung, T. Y. Chen, and Y. Xiao, "Validating class integration test order generation systems with metamorphic testing," *Inf. Softw. Technol.*, vol. 132, Apr. 2021, Art. no. 106507.

[11] Z. Q. Zhou, L. Sun, T. Y. Chen, and D. Towey, "Metamorphic relations for enhancing system understanding and use," *IEEE Trans. Softw. Eng.*, vol. 46, no. 10, pp. 1120–1154, Oct. 2020.

[12] C.-A. Sun, Y. Liu, Z. Wang, and W. K. Chan, "μMT: A data mutation directed metamorphic relation acquisition methodology," in *Proc. IEEE/ACM 1st Int. Workshop Metamorphic Test. (MET)*, May 2016, pp. 12–18.

[13] C.-A. Sun, H. Jin, A. Fu, Z. Wang, and W. Chan. (2022). *Identifying Metamorphic Relations: A Data Mutation Directed Approach*. [Online]. Available: https://ssrn.com/abstract=4040492

[14] X. Chen, D. Zhang, Y. Zhao, Z. Cui, and C. Ni, "Software defect number prediction: Unsupervised vs supervised methods," *Inf. Softw. Technol.*, vol. 106, pp. 161–181, Feb. 2019, doi: 10.1016/j.infsof.2018.10.003.

[15] N. Li, M. Shepperd, and Y. Guo, "A systematic review of unsupervised learning techniques for software defect prediction," *Inf. Softw. Technol.*, vol. 122, Jun. 2020, Art. no. 106287, doi: 10.1016/j.infsof.2020.106287.

[16] E. Sreedevi, V. PremaLatha, S. Sivakumar, and S. R. Nayak, "A comparative study on new classification algorithm using NASA MDP datasets for software defect detection," in *Proc. Int. Conf. Intell. Sustain. Syst. (ICISS)*, Feb. 2019, pp. 312–317, doi: 10.1109/ISS1.2019.8908096.

[17] M. J. Hemández-Molinos, Á. J. Sánchez-García, and R. E. Barrientos-Martínez, "Classification algorithms for software defect prediction: A systematic literature review," in *Proc. 9th Int. Conf. Softw. Eng. Res. Innov. (CONISOFT)*, Oct. 2021, pp. 189–196.

[18] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, "Testing machine learning based systems: A systematic mapping," *Empirical Softw. Eng.*, vol. 25, no. 6, pp. 5193–5254, Nov. 2020.

[19] S. Segura, A. Durán, J. Troya, and A. Ruiz-Cortés, "Metamorphic relation patterns for query-based systems," in *Proc. IEEE/ACM 4th Int. Workshop Metamorphic Test. (MET)*, May 2019, pp. 24–31.

[20] K. K. A. Chua, D.-H. Bae, and E. Jee, "Metamorphic testing for reliability in system of systems," in *Proc. 28th Asia–Pacific Softw. Eng. Conf. (APSEC)*, Dec. 2021, pp. 390–400, doi: 10.1109/APSEC53868.2021.00046.

[21] S. Segura, D. Towey, Z. Q. Zhou, and T. Y. Chen, "Metamorphic testing: Testing the untestable," *IEEE Softw.*, vol. 37, no. 3, pp. 46–53, May 2020.

[22] B. Stacy, J. Hauzel, M. Lindvall, A. Porter, and M. Pop, "Metamorphic testing in bioinformatics software: A case study on metagenomic assembly," in *Proc. IEEE/ACM 7th Int. Workshop Metamorphic Test. (MET)*, May 2022, pp. 31–33.

[23] J. D. Ellis, R. Iqbal, and K. Yoshimatsu, "Verification of the neural network training process for spectrum-based chemical substructure prediction using metamorphic testing," *J. Comput. Sci.*, vol. 55, Oct. 2021, Art. no. 101456.

[24] X. Xie, J. W. K. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," *J. Syst. Softw.*, vol. 84, no. 4, pp. 544–558, Apr. 2011, doi: 10.1016/j.jss.2010.11.920.

[25] P. Saha and U. Kanewala, "Fault detection effectiveness of metamorphic relations developed for testing supervised classifiers," in *Proc. IEEE Int. Conf. Artif. Intell. Test. (AITest)*, Apr. 2019, pp. 157–164, doi: 10.1109/AITEST.2019.00019.

[26] F. U. Rehman and C. Izurieta, "An approach for verifying and validating clustering based anomaly detection systems using metamorphic testing," in *Proc. IEEE Int. Conf. Artif. Intell. Test. (AITest)*, Aug. 2022, pp. 12–18, doi: 10.1109/AITest55621.2022.00011.
[27] F. Ur Rehman and C. Izurieta, "MT4UML: Metamorphic testing for unsupervised machine learning," in *Proc. 9th Swiss Conf. Data Sci. (SDS)*, Jun. 2022, pp. 26–32, doi: 10.1109/SDS54800.2022.00012.
[28] S. Pugh, M. S. Raunak, D. R. Kuhn, and R. Kacker, "Systematic testing of post-quantum cryptographic implementations using metamorphic testing," in *Proc. IEEE/ACM 4th Int. Workshop Metamorphic Test. (MET)*, May 2019, pp. 2–8.
[29] S. Herbold and T. Haar, "Smoke testing for machine learning: Simple tests to discover severe bugs," *Empirical Softw. Eng.*, vol. 27, no. 2, p. 45, Mar. 2022, doi: 10.1007/s10664-021-10073-7.
[30] I. A. Venkatkumar and S. J. K. Shardaben, "Comparative study of data mining clustering algorithms," in *Proc. Int. Conf. Data Sci. Eng. (ICDSE)*, Aug. 2016, pp. 1–7.
[31] E. Patel and D. S. Kushwaha, "Clustering cloud workloads: K-means vs Gaussian mixture model," *Proc. Comput. Sci.*, vol. 171, pp. 158–167, Jan. 2020.
[32] SciKit-Learn. *2.1. Gaussian Mixture Models*. Accessed: 2022. [Online]. Available: https://scikit-learn.org/stable/modules/mixture.html
[33] SciKit-Learn. *Sklearn.Mixture.GaussianMixture*. Accessed: 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html
[34] H. Attias, "A variational Baysian framework for graphical models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 12, 1999, pp. 1–7.
[35] SciKit-Learn. *Sklearn.Mixture.BayesianGaussianMixture*. Accessed: 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.mixture.BayesianGaussianMixture.html
[36] SciKit-Learn. *Sklearn.Cluster.KMeans*. Accessed: 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
[37] J. Peksa, "An automated algorithm implementation to fill missing points with Euclidean approach," in *Proc. 4th Int. Conf. Inf. Comput. Technol. (ICICT)*, Mar. 2021, pp. 79–83, doi: 10.1109/ICICT52872.2021.00020.
[38] V. Rohilla, M. S. S. Kumar, S. Chakraborty, and M. S. Singh, "Data clustering using bisecting K-means," in *Proc. Int. Conf. Comput., Commun., Intell. Syst. (ICCCIS)*, Oct. 2019, pp. 80–83.
[39] SciKit-Learn. *Sklearn.Cluster.BisectingKMeans*. Accessed: 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.BisectingKMeans.html
[40] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Ann. Data Sci.*, vol. 2, no. 2, pp. 165–193, Jun. 2015.
[41] H. He, Y. He, F. Wang, and W. Zhu, "Improved K-means algorithm for clustering non-spherical data," *Expert Syst.*, vol. 39, no. 9, Nov. 2022, Art. no. e13062.
[42] K. Zhang and X. Gu, "An affinity propagation clustering algorithm for mixed numeric and categorical datasets," *Math. Problems Eng.*, vol. 2014, no. 1, pp. 1–8, 2014.
[43] P. Bhattacharjee and P. Mitra, "A survey of density based clustering algorithms," *Frontiers Comput. Sci.*, vol. 15, no. 1, pp. 1–27, Feb. 2021.
[44] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *ACM SIGMOD Rec.*, vol. 28, no. 2, pp. 49–60, Jun. 1999.
[45] Y. Z. Bala, P. A. Samat, K. Y. Sharif, and N. Manshor, "Current software defect prediction: A systematic review," in *Proc. Appl. Informat. Int. Conf. (AiIC)*, May 2022, pp. 117–121.
[46] L. H. Son, N. Pritam, M. Khari, R. Kumar, P. T. M. Phuong, and P. H. Thong, "Empirical study of software defect prediction: A systematic mapping," *Symmetry*, vol. 11, no. 2, p. 212, Feb. 2019.
[47] H. Chen, Z. Zhang, W. Yin, C. Zhao, F. Wang, and Y. Li, "A study on depth classification of defects by machine learning based on hyper-parameter search," *Measurement*, vol. 189, Feb. 2022, Art. no. 110660.
[48] M. T. Sathe and A. C. Adamuthe, "Comparative study of supervised algorithms for prediction of students' performance," *Int. J. Modern Educ. Comput. Sci.*, vol. 13, no. 1, pp. 1–21, 2021.
[49] L. Jiang, Z. Cai, D. Wang, and S. Jiang, "Survey of improving K-nearest-neighbor for classification," in *Proc. 4th Int. Conf. Fuzzy Syst. Knowl. Discovery (FSKD)*, vol. 1, 2007, pp. 679–683.
[50] Meiliana, S. Karim, H. L. H. S. Warnars, F. L. Gaol, E. Abdurachman, and B. Soewito, "Software metrics for fault prediction using machine learning approaches: A literature review with PROMISE repository dataset," in *Proc. IEEE Int. Conf. Cybern. Comput. Intell. (CyberneticsCom)*, Nov. 2017, pp. 19–23.
[51] I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Phil. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 374, no. 2065, Apr. 2016, Art. no. 20150202.
[52] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F_1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, p. 6, Dec. 2020, doi: 10.1186/s12864-019-6413-7.
[53] Z. Xu, J. Xuan, J. Liu, and X. Cui, "MICHAC: Defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering," in *Proc. IEEE 23rd Int. Conf. Softw. Anal., Evol., Reeng. (SANER)*, vol. 1, Mar. 2016, pp. 370–381, doi: 10.1109/SANER.2016.34.
[54] D. Papakyriakou and I. S. Barbounakis, "Data mining methods: A review," *Int. J. Comput. Appl.*, vol. 183, no. 48, pp. 5–19, Jan. 2022, doi: 10.5120/ijca2022921884.
[55] SciKit-Learn. *Sklearn.Cluster.AffinityPropagation*. Accessed: 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AffinityPropagation.html
[56] X. Liu, M. Yin, J. Luo, and W. Chen, "An improved affinity propagation clustering algorithm for large-scale data sets," in *Proc. 9th Int. Conf. Natural Comput. (ICNC)*, Jul. 2013, pp. 894–899.
[57] A. M. Serdah and W. M. Ashour, "Clustering large-scale data based on modified affinity propagation algorithm," *J. Artif. Intell. Soft Comput. Res.*, vol. 6, no. 1, pp. 23–33, Jan. 2016, doi: 10.1515/jaiscr-2016-0003.
[58] D. Sculley, "Web-scale K-means clustering," in *Proc. 19th Int. Conf. World Wide Web*, Apr. 2010, pp. 1177–1178.
[59] F. Alharithi, A. Almulihi, S. Bourouis, R. Alroobaea, and N. Bouguila, "Discriminative learning approach based on flexible mixture model for medical data categorization and recognition," *Sensors*, vol. 21, no. 7, p. 2450, Apr. 2021.
[60] Y. Ao, H. Li, L. Zhu, S. Ali, and Z. Yang, "The linear random forest algorithm and its advantages in machine learning assisted logging regression modeling," *J. Petroleum Sci. Eng.*, vol. 174, pp. 776–789, Mar. 2019.
[61] S. Ray, "A quick review of machine learning algorithms," in *Proc. Int. Conf. Mach. Learn., Big Data, Cloud Parallel Comput. (COMITCon)*, Feb. 2019, pp. 35–39.

PAK YUEN PATRICK CHAN received the B.Com. and M.Com. degrees from the University of Canterbury, New Zealand. He is currently pursuing the Ph.D. degree with the Department of Computer Science, City University of Hong Kong. His research interest includes AI for software quality assurance and validation.

JACKY KEUNG (Senior Member, IEEE) received the B.Sc. degree (Hons.) in computer science from The University of Sydney and the Ph.D. degree in software engineering from the University of New South Wales, Australia. Before joining the City University of Hong Kong, he was then a Research Scientist with the Software Systems Research Group, NICTA (now DATA61, CSIRO) based in Sydney, Australia. His research interests encompass a wide range of domains, reflecting his expertise in various areas. These include software and systems engineering, data science, AI, FinTech, machine learning, blockchain systems for FinTech applications, deep learning, LLM applications, defect prediction, and empirical modeling and evaluation of complex systems. His diverse expertise has led to fruitful collaborations with numerous software companies and government departments across the Asia Pacific Region. He has produced more than 150 journal and conference research papers in top venues.