METTLE: A METamorphic Testing Approach to Assessing and Validating Unsupervised Machine Learning Systems

Xiaoyuan Xie, *Member, IEEE*, Zhiyi Zhang, Tsong Yueh Chen, *Senior Member, IEEE*, Yang Liu, *Member, IEEE*, Pak-Lok Poon, *Member, IEEE*, and Baowen Xu, *Senior Member, IEEE*

***Abstract*—Unsupervised machine learning is the training of an artificial intelligence system using information that is neither classified nor labeled, with a view to modeling the underlying structure or distribution in a dataset. Since unsupervised machine learning systems are widely used in many real-world applications, assessing the appropriateness of these systems and validating their implementations with respect to individual users' requirements and specific application scenarios/contexts are indisputably two important tasks. Such assessments and validation tasks, however, are fairly challenging due to the absence of *a priori* knowledge of the data. In view of this challenge, in this article, we develop a METamorphic Testing approach to assessing and validating unsupervised machine LEarning systems, abbreviated as METTLE. Our approach provides a new way to unveil the (possibly latent) characteristics of various machine learning systems, by explicitly considering the specific expectations and requirements of these systems from individual users' perspectives. To support METTLE, we have further formulated 11 generic metamorphic relations (MRs), covering users' generally expected characteristics that should be possessed by machine learning systems. We have performed an experiment and a user evaluation study to evaluate the viability and effectiveness of METTLE. Our experiment and user evaluation study have shown that, guided by user-defined MR-based adequacy criteria, end users are able to assess, validate, and select appropriate clustering systems in accordance with their own specific needs. Our investigation has also yielded insightful understanding and interpretation of the behavior of the machine learning systems from an *end-user software engineering's perspective*, rather than a designer's or implementor's perspective, who normally adopts a theoretical approach.**

***Index Terms*—Clustering assessment, clustering validation, end-user software engineering, metamorphic testing (MT), metamorphic relation (MR), unsupervised machine learning.**

Manuscript received January 10, 2019; revised June 10, 2019 and October 23, 2019; accepted February 4, 2020. Date of publication March 13, 2020; date of current version November 30, 2020. This work was supported by the National Key R&D Program of China under Grant 2018YFB1003901, and in part by the National Natural Science Foundation of China under Grant 61572375, Grant 61972289, Grant 61832009, and Grant 61772263. Associate Editor: F. Belli. *(Corresponding author: Xiaoyuan Xie.)*

Xiaoyuan Xie and Zhiyi Zhang are with the School of Computer Science, Wuhan University, Wuhan 430072, China (e-mail: xxie@whu.edu.cn; zhiyi_whu@126.com).

Tsong Yueh Chen is with the Department of Computer Science and Software Engineering, Swinburne University of Technology, Hawthorn, VIC 3122, Australia (e-mail: tychen@swin.edu.au).

Yang Liu is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore (e-mail: yangliu@ntu.edu.sg).

Pak-Lok Poon is with the School of Engineering and Technology, Central Queensland University, Melbourne, VIC 3000, Australia (e-mail: p.poon@cqu.edu.au).

Baowen Xu is with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: bwxu@nju.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TR.2020.2972266

I. Introduction

UNSUPERVISED machine learning requires no prior knowledge and can be widely used in a large variety of applications such as market segmentation for targeting customers [1], anomaly or fraud detection in banking [2], grouping genes or proteins in biological process [3], deriving climate indices from earth science data [4], and document clustering based on content [5]. More recently, unsupervised machine learning has also been used by software testers in predicting software faults [6].

This article specifically focuses on *clustering systems* (which refer to software systems that implement clustering algorithms and are intended to be used in different domains) Such a clustering system helps users partition a given unlabeled dataset into groups (or clusters) based on some similarity measures, so that data in the same cluster are more "similar" to each other than to data from different clusters. In artificial intelligence (AI) and data mining, numerous clustering systems [7]–[9] have been developed and are available for public use. Thus, selecting the most appropriate clustering system for use is an important concern from end users. (In this article, *end users*, or simply *users*, refer to those people who are "causal" users of clustering systems. Although they have some hands-on experience on using such systems, they often do not possess a solid theoretical foundation on machine learning. These users come from different fields such as bioinformatics [10] and nuclear engineering [11]. Also, their main concern is the applicability of a clustering system in the *users' specific contexts*, rather than the detailed logic of this system.) From a user's perspective, this selection is not trivial [12], not only because end users generally do not have very solid theoretical background on machine learning, but also because the selection task involves two complex issues as follows.

(Issue 1) The correctness of the clustering results is a major concern for users. However, when evaluating a clustering system, there is not necessarily a correct solution or "ground truth"

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

that users can refer to for verifying the clustering result [13]. Furthermore, not only is the correct result difficult or infeasible to find, the interpretation of correctness varies from one user to another. This is because, although data points are partitioned into clusters based on some similarity measures, the comprehension of "similarity" may vary among individual users. Given a cluster, one user may consider that the data in it are similar, yet another user may consider not.

(Issue 2) Despite the importance of the correctness of the clustering result, in many cases, users would probably be more concerned if a clustering system produces an output that is appropriate or meaningful to their particular scenarios of applications. This view is supported by the following argument in [14].

> *... the major obstacle is the difficulty in evaluating a clustering algorithm without taking into account the context: why does the user cluster his data in the first place, and what does he want to do with the clustering afterwards? We argue that clustering should not be treated as an application-independent mathematical problem, but should always be studied in the context of its end-use.*

Regarding issue 1, it is well known as the *oracle problem* in software testing. This problem occurs when a test oracle (or simply an oracle) does not exist. Here, an *oracle* refers to a mechanism that can verify the correctness of the system output [15]. In view of the oracle problem, users of unsupervised machine learning rely on two types of validation techniques (external and internal) to evaluate clustering systems. Basically, *external* validation techniques evaluate the output clusters based on some existing benchmarks; while *internal* validation techniques adopt features inherent to the data alone to validate the clustering result.

Both external and internal validation techniques suffer from some problems, which affect their effectiveness and applicability. For external techniques, it is usually difficult to obtain sufficient relevant benchmark data for comparison [16], [17]. In most situations, the benchmarks selected for use are essentially those special cases in software verification and validation, thereby providing insufficient test adequacy, coverage, and diversity. This issue does not exist in internal validation techniques. However, since internal techniques mainly rely on the features associated with the dataset, their performance is easily affected by various data characteristics [18]. *In addition, both external and internal techniques evaluate clustering systems mainly from the "static" perspective of a dataset, without considering the changeability of input datasets or the interrelationships among different clustering results (i.e., the "dynamic" aspect).* We argue that, in reality, users require this dynamic perspective of a clustering system to be evaluated, because datasets may change due to various reasons. For instance, before the clustering process starts, a dataset may be preprocessed to filter out noises and outliers for improving the reliability of the clustering result, or the data may be normalized so that different measures use the same scale for fair and reliable comparison.

Our above argument is based on a common phenomenon that users often have some general expectations about the change in the clustering result when the dataset is changed in a particular way, for example, a better clustering result should be obtained after the noises have been filtered out from a dataset. To many users, evaluating this dynamic aspect (called the *ripple effect of dataset change or transformation*) will give them more confidence on the performance of a clustering system than a code coverage test [15]. Despite its importance, it is unfortunate that both external and internal techniques generally do not consider the dynamic aspect of dataset transformation when testing clustering systems.

We now turn to issue 2: There has not yet been a generally accepted and systematic methodology that allows end users to effectively assess the quality and appropriateness of a clustering system for their particular applications. In traditional software testing, *test adequacy* is commonly measured by code coverage criteria to unveil necessary conditions of detecting faults in the code (e.g., incorrect logic). In this regard, clustering systems are harder to assess because the logic of a machine learning model is primarily learnt from massive data. In view of this problem, a practically applicable adequacy criterion is in need to help a user assess and validate the characteristics that a clustering system should possess in a specific application scenario, so that the most appropriate system can be selected for use from this user's perspective. As a reminder, the characteristics that a clustering system is "expected" to possess may vary across different users. Needless to say, there is also no systematic methodology for users to *validate* the appropriateness of a clustering result in their own contexts.

In view of the above two challenging issues, we propose a **MET**amorphic **T**esting approach to assessing and validating unsupervised machine **LE**arning systems (abbreviated as METTLE). To alleviate Issue 1, METTLE applies the framework of metamorphic testing (MT) [15], so that users are still able to validate a clustering system even when the oracle problem occurs. In addition, MT is naturally considered to be a candidate solution for addressing the ripple effect of data transformation, since MT involves multiple inputs (or datasets), which follow a specific transformation relation. By defining a set of *metamorphic relations* (*MRs*) (which capture the relations between multiple inputs (or datasets) and their corresponding outputs (or clustering results)) to be used in MT, the dynamic perspective of a clustering system can be properly assessed. Furthermore, the defined MRs can address Issue 2 by serving as an effective vehicle for users to specify their expected characteristics of a clustering system in their specific application scenarios. The compliance of the clustering results across multiple system executions with these MRs can be treated as a practical adequacy criterion to help a user select the appropriate clustering system for use. More details about the rationale and the procedure of METTLE will be provided in later sections.

The main contributions of this article are summarized as follows.

1) We proposed an MT-based approach (METTLE) to assessing and validating unsupervised machine learning systems that generally suffer from the absence of *a priori* knowledge of the data and a test oracle. Different from traditional validation methods, our approach provides a new and lightweight mechanism to unveil the (possibly latent) characteristics of various learning systems, by explicitly

considering the specific expectations and requirements of these systems from the perspective of individual users, who do not possess a solid theoretical foundation of machine learning. In addition, METTLE can validate learning systems by explicitly considering the dynamic aspect of a dataset.

2) We developed 11 generic MRs to support METTLE, from users' generally expected characteristics of clustering systems. We conducted an experiment involving six commonly used clustering systems, which were assessed and compared against these 11 MRs through both quantitative and qualitative analysis.
3) We demonstrated a framework to help users assess clustering systems based on their own specific requirements. Guided by an adequacy criterion (with respect to those chosen generic MRs or those MRs specifically defined by users), users are able to select the appropriate unsupervised learning systems to serve their own purposes.
4) Our investigation has yielded insightful understanding and interpretation of the behaviors of some commonly used machine learning systems from a user's perspective, rather than a designer's or implementor's perspective (who normally adopts a more theoretical approach).

The rest of this article is organized as follows. Section II outlines the main concepts of clustering systems and MT. Section III discusses the challenges in clustering validation and the potential problems associated with dataset transformation in clustering. Section IV describes our METTLE methodology and a list of 11 generic MRs to support METTLE. Section V discusses our experimental setup to determine the effectiveness of METTLE in validating a set of subject clustering systems. Section VI presents a quantitative analysis of the performance of the subject clustering systems in terms of their compliance with (or violation to) the 11 generic MRs, followed by an in-depth qualitative analysis on the underlying behavior patterns and plausible reasons for the violations revealed by these MRs. Section VII illustrates how METTLE can be used as a systematic and yet easy-to-use framework (*without the requirement of having sophisticated knowledge on machine learning theories*) for assessing the appropriateness of clustering systems in accordance with a user's own specific requirements and expectations. Section VIII further evaluates the viability and effectiveness of METTLE from a user's perspective. Section IX discusses some internal and external threats to this article. Section X briefly discusses the recent related work on MT. Finally, Section XI concludes this article.

II. Background Concepts

In this section, we discuss the background concepts of clustering systems and MT. We also give some examples to illustrate how MT can be used as a software validation approach.

A. Clustering Systems

In AI, *clustering* [19], [20] is the task of partitioning a given unlabeled dataset into clusters based on some similarity measures, where data in the same cluster are more "similar" to each other than to data from different clusters. Thus, cluster analysis

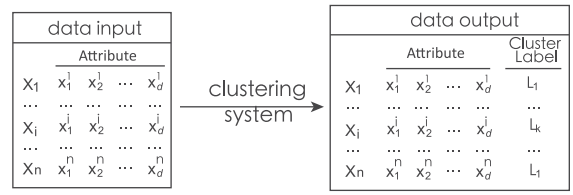

Fig. 1. Clustering system.

involves the discovery of the latent structure or distribution of data in a dataset. The clustering problem can be formally defined as follows.

Definition 1 (Clustering): Assuming that dataset $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ contains n instances; each instance $\mathbf{x}_i = (x_1^i, x_2^i, \ldots, x_d^i)$ has d-dimensional attributes. A clustering system divides D into k clusters $\mathbb{C} = \{C_1, C_2, \ldots, C_k\}$ with label L_k be the label for cluster C_k, where $\bigcup_{i=1}^{k} C_i = D, C_i \neq \varnothing, C_i \cap C_j = \varnothing\ (i \neq j, 1 \leq i, j \leq k)$.

Fig. 1 describes the input and output of a clustering system. It is well known that validating clustering systems will encounter the oracle problem (i.e., the absence of an oracle). For instance, it is argued in [13] that

> *The problem is that there isn't necessarily a 'correct' or ground truth solution that we can refer to it if we want to check our answers ...you will come to the inescapable conclusion is that there is no 'true' number of clusters (though some numbers feel better than others) [therefore a definite correct clustering result, or an oracle, does not exist], and that the same dataset is appropriately viewed at various levels of granularity depending on analysis goals.*

In view of the oracle problem, users of machine learning generally rely on two types (internal and external) of techniques to validate clustering systems. Both types, however, are not satisfactory because of their own limitations. These limitations have been briefly outlined in Section I, and will be further elaborated in Section III-A.

B. Metamorphic Testing

To alleviate the oracle problem, MT [15], [21] has been proposed to verify and validate the "expected" relationships between inputs and outputs across *multiple* software executions. These "expected" relationships are expressed as MRs [22], [23]. If the output results across multiple software executions violate an MR, then a fault is revealed. Below, we give an example to illustrate the main concepts of MT and MR.

Consider a program S that calculates the value of the $\sin(x)$ function. It is extremely difficult to verify the correctness of the output from S because an oracle is extremely difficult to compute, except that x is a special value (such as π where $\sin(\pi) = 1$). MT can help alleviate this oracle problem. Consider, for example, the mathematical property $\sin(x) = \sin(\pi - x)$. Based on this property, we can define an MR in MT: "If $y = \pi - x$, then $\sin(x) = \sin(y)$." With reference to this MR, S is executed twice: first, with any angle x as a *source test case*; and, then, with the angle y, such that $y = \pi - x$, as a *follow-up test case*. In this case, even the correct and precise value of $\sin(x)$ is unknown, if the two execution results (one

with input x and the other with input y) are *different* so that the above MR is violated, we can conclude that S is faulty. The above example illustrates an important feature of MT—it involves *multiple* software executions.

MT was initially proposed as a verification technique. For example, Murphy *et al.* [24] applied MT to several machine learning applications (e.g., MartiRank) and successfully revealed several defects. Different types of metamorphic properties were also categorized to provide a foundation for determining the relationships and transformations that can be used for conducting MT in machine learning applications [24]. Another study has successfully demonstrated that MT can be extended to support validation of supervised machine learning software [25]. In their study, Xie *et al.* [25] presented a series of MRs (which may not be the necessary properties of the relevant algorithm) generated from the anticipated behaviors of supervised classifiers. Violations to the MRs may indicate that the relevant classifier is unsuitable to the current application scenario, even if the algorithm is correctly implemented.

Later, Zhou *et al.* [26] applied MT to validate online search services. They adopted logical consistency relations as a measure of users' perceived quality of search services, and used this measure to validate the performance of four popular search engines such as Google and Bing. In this work [26], Zhou *et al.* compared four search engines with respect to different scenarios and factors, thereby providing users and developers with a more comprehensive understanding of how to choose a proper search engine for better searching services with clear and definite objectives. Olsen and Raunak [27] applied MT for simulation validation, involving two prevalent simulation approaches: agent-based simulation and discrete-event simulation. Guidelines were also provided for identifying MRs for both simulation approaches. Case studies [27] showed how MT can help increase users' confidence in the correctness of the simulation models.

MT has also been recently applied to validate a deep learning framework for automatically classifying biology cell images that involves a convolutional neural network and a massive image dataset [28]. This work has demonstrated the effectiveness of MT for ensuring the quality of deep learning (especially involving massive training data). Moreover, this MT-based validation approach can be further extended for checking the quality of other deep learning applications. Other recent works [29], [30], [31] have also been conducted to validate autonomous driving systems where MRs were leveraged to automatically generate test cases to reflect real-world scenes.

III. Motivation

Recall that users of the machine learning community often rely on certain validation techniques (which mainly focus on the "static" aspect of a dataset) to evaluate clustering systems. Moreover, these validation techniques suffer from several problems, which affect their effectiveness and applicability (e.g., unable to validate the "dynamic" aspect of a dataset, that is, the effect of changing the input datasets on the clustering results). Section III-A below discusses in detail the limitations of most existing cluster validation techniques. Section III-B, then, presents some potential problems associated with data transformation that should be addressed when validating clustering systems.

A. Challenges in Clustering Validation

In unsupervised machine learning, *clustering* is a technique to divide a group of data samples into clusters such that data samples within the same cluster are "similar" to each other; while data samples of different clusters show "distinct" features from each other. Because clustering attempts to discover hidden patterns in data with no prior knowledge, it is difficult to evaluate the correctness or quality of the clustering results (see Issues 1 and 2 in Section I).

Generally speaking, there are two major types of techniques (external and internal) for validating the clustering result. Both of them, however, have their own limitations.

1) External Validation Techniques: The basic idea is to compare the clustering result with an external benchmark or measure, which corresponds to a prespecified data structure. For external validity measures, there are several essential criteria to follow such as cluster homogeneity and completeness [32]. Consider, for instance, the widely adopted *F-measure* [33]. It considers two important aspects: *recall* (how many samples within a category are assigned to the same cluster) and *precision* (how many samples within a cluster are in one category). It is well known that good and relevant external benchmarks are hard to obtain. This is because, in most situations, the data structure specified by the predefined class labels or other users is unknown. As a result, without prior knowledge, it is generally very expensive and difficult to obtain an appropriate external benchmark for comparing with the clustering structure generated by a clustering system.

2) Internal Validation Techniques: This type of techniques validate the clustering result by adopting features inherent to the dataset alone. Many internal validity indices were proposed based on two aspects: intercluster compactness and intracluster separation. For example, one of the widely adopted indices—the *silhouette coefficient*—was proposed based on the concept of distance/similarity [34]. If this coefficient (which ranges from -1 to $+1$) of a data sample is close to $+1$, it means that this data sample is well matched to its own cluster and poorly matched to neighboring clusters. When compared with external techniques, on one hand, internal techniques are more practical because they can be applied without an oracle. On the other hand, internal techniques are less robust because they mainly rely on the features associated with the dataset, that is, data compactness and data separation. Hence, the performance of internal techniques could be easily affected by various data characteristics such as noise, density, and skewed distribution [18].

In addition to the specific limitations of external and internal validation techniques mentioned above, both types of techniques validate clustering systems mainly from a "static" perspective, without considering the changeability of input datasets or the interrelationships among different clustering results.

To address the limitations of external and internal validation techniques with respect to the "dynamic" perspective of clustering, based on the notion of *cluster stability* [35], various

resampling techniques have been developed to complement the external and internal techniques. A core concept of these resampling techniques (and cluster stability) is that *independent* sample sets drawn from the same underlying statistical distribution should produce similar clustering results. Various resampling techniques [36]–[38] have been proposed to generate independent sample sets. An example of these resampling techniques is Bootstrap (a representative nonparametric resampling technique) [39], which obtains samples by drawing a certain number of data points randomly with replacement from the original samples, and calculates a sample variance to estimate the population variance. Another example is Jittering [37], which generates copies of the original sample by randomly adding noises to the dataset in order to simulate the influence of measurement errors. As a reminder, although Jittering considers noises and outliers, it does not explicitly investigate the changing trend of clusters.

To some extent, resampling techniques complement the external and internal validation techniques by comparing *multiple* clustering results. However, it is not difficult to see from Boobstrap [39] and Jittering [37] discussed above that resampling techniques do not provide a comprehensive validation on the dynamic perspective of clustering systems, because they mainly deal with *independent* sample sets. In reality, datasets may change constantly in various manners, involving *interrelated* datasets [40]. Thus, estimating cluster stability without considering these *interrelated* datasets may result in incomprehensive clustering validation.

We argue that, in most cases, users of machine learning are particularly concerned whether a clustering system produces an output that is appropriate or meaningful to their specific scenarios of applications: Our argument is supported by AI researchers [13], [14]. For example, it is argued in [14] that "clustering should not be treated as an application-independent mathematical problem, but should always be studied in the context of its end-use." Therefore, given a particular clustering system, one user may consider it useful, while another user may not, because of their different "expectations" or "preferences" on the clustering result. In spite of the need for catering for different users' preferences, existing clustering validation techniques (external, internal, and resampling) generally do not allow users to specify and validate their unique preferences when evaluating clustering systems (see Issue 2 in Section I). Furthermore, even if we consider a particular user, it is possible that none of the existing available clustering systems fulfills all their preferences on a clustering system. If this happens, users can only choose a particular clustering system that can fulfill their preferences the best.

It has been reported that a general, systematic, and objective assessment and validation approach for *all* clustering problems does not exist [14]. Although many cluster validation methods with a range of desired characteristics have been developed, most of them are based on statistical testing and analysis. There are still other desired characteristics that existing cluster validation methods have not been addressed. In view of this problem, rather than proposing a cluster validation method, which is "generic" enough to evaluate *every* desired characteristic from *all* possible users on a clustering system (which is intuitively infeasible), our strategy is to propose a "flexible, systematic, and easy-to-use" evaluation framework (i.e., METTLE) so that users are able to define their own sets of desired characteristics and, then, use these sets to validate the appropriateness of a clustering system in their specific application scenarios.

B. Potential Problems Associated With Dataset Transformation

In reality, datasets may be changed now and then. For example, before clustering commences, we may need to preprocess a dataset to filter out noises and outliers, in order to make the clustering result more reliable. We may also need to normalize the data so that different measures use the same scale for the sake of comparison. In this regard, whether data transformation may result in some unknown and undesirable ripple effect on the clustering result is a definite concern for most users.

Often, users have some general expectations about the impact on the clustering result when the dataset is changed in a particular manner (i.e., the "dynamic" perspective of a dataset). Consider, for example, the filtering of noises and outliers from the dataset before clustering starts. Not only users expect the absence of the ripple effect, they also expect a better clustering result after the filtering process. Another example is that users generally expect that a clustering system is not susceptible to the input order of data. However, we observe that some clustering systems, such as k-means (KM) [41], do not meet this expectation. This is because KM and some other clustering systems are, in fact, sensitive to the input order of data due to the choice of the original cluster centroid, thus even a slight offset in distance will have a noticeable effect on the clustering result.

One may argue that KM is a popular clustering system, so users are likely to be aware of its above characteristic with respect to the input order of data. As a result, users will consider this issue when evaluating whether KM should be used for clustering in their own contexts. We argue, however, as more and more new clustering systems are developed, it is practically infeasible for users to be knowledgeable about the potential ripple effect of data transformation for every method (particularly the newly developed ones), so that the most appropriate one could be selected for use.

IV. Our Methodology: METTLE

This section introduces our approach for cluster assessment and validation. Section IV-A outlines some key features and core concepts associated with METTLE from the perspective of *end-user software engineering* [42], [43]. Section IV-B gives the relevant definitions used in METTLE. Section IV-C, then, presents a list of 11 generic MRs (which are based on some common end users' expectations on a clustering system) developed to support METTLE.

A. Key Features and Core Concepts

To alleviate the challenges and potential problems mentioned in Sections III-A and III-B, we propose an MT-based

methodology (METTLE) for "users" to assess and validate unsupervised machine learning systems. In this article, as explained in Section I, *users* refer to those "causal" users with some hands-on experience on using clustering systems in their specific application scenarios (e.g., biomedicine, market segmentation, and document clustering), but do not possess a solid theoretical foundation on machine learning. Thus, these users often have little interest on the internal logic of clustering systems. Rather, they are more concerned with the applicability of these systems in their own usage contexts. Consider, for example, users in bioinformatics consider using a clustering system to perform predictive analysis. These bioinfomaticians often have good domain knowledge about complex biological applications, but they are not experts in machine learning, or do not care much about the detailed theories in machine learning. For these users, there is a vital demand for an effective and yet easy-to-use validation method (but without the need for having sophisticated knowledge about machine learning) to help them select an appropriate clustering system for their own use.

Some key features of METTLE are listed below.

1) It alleviates the oracle problem in validating clustering systems (see Issue 1 in Section I).
2) It allows users to comprehensively assess and validate the "dynamic" perspectives of clustering systems related to dataset transformation. In other words, it enables users to test the impact on the clustering result when the input dataset is changed in a particular way for a given clustering system. Thus, METTLE works well with interrelated datasets.
3) It allows users to assess and validate their expected characteristics (expressed in the form of MRs) of a clustering system. In addition, during assessing and validating, users are able to assign weighted scores to defined MRs in accordance with their relative importance from the users' perspectives. As such, an MR-based adequacy criterion, by means of a set of user's defined MRs, can be derived to help users select an appropriate clustering system for their own use (see Issue 2 in Section I).
4) METTLE is supported with an initial suite of 11 MRs, which are fairly generic and are expected to be applicable across many different application scenarios and contexts. As a reminder, in reality, users may ignore some of these MRs that are irrelevant or inapplicable in a specific application scenario.

Features 1) to 3) of METTLE are made available by allowing users to define a set of MRs, with each MR capturing a relation between multiple inputs (datasets) and outputs (clusters) across different clustering tasks. These user-defined MRs, together with the generic MRs in the initial suite (feature 4) above), are assigned with weighted scores to reflect their relative importance from the user's perspective (feature 3) above). Such "ranked" MRs, thus, allow users to specify their expected characteristics of a clustering system. If a clustering system generates results from multiple executions which violate an MR, it indicates that this system does not fulfill the expected characteristic corresponding to this MR. Thus, the set of user-defined specific MRs and user-chosen generic MRs essentially serves as a test adequacy criterion for users to evaluate candidate clustering systems, with a view to selecting the most appropriate one for use.

B. Definitions

1) MR for Cluster Validation: Given a clustering system A and a dataset D. Let $R_s = A(D)$ denote the clustering result. Assume that a transformation T is applied to D and generates D^T. Let $R_f = A(D^T)$ denote the new clustering result. An MR defines the expectation from users about the changing trend of A's behaviors after transforming D by T, that is, the expected relation $\mathbb{R}^T$ between R_s and R_f after T.

We call the original dataset D and the result R_s as the *source input* (source sample set) and the *source output* (source clustering result), respectively; call the transformed D^T and the result R_f as the *follow-up input* (follow-up sample set) and the *follow-up output* (follow-up clustering result), respectively; and call the clustering processes with D and D^T as the *source execution* and the *follow-up execution*, respectively.

2) Output Relations: An MR for validation may not be a necessary property of the system under test, especially for machine learning systems. Also, clustering results may vary due to randomness. Thus, we will not simply check whether or not an MR holds across multiple clustering results, as normally done in MT. If the clustering results violate an MR, we will investigate the reason(s) for such violations. To facilitate this, we will analyze and investigate an output relation across different clustering results in the following aspects.

1) *Changes on the returned cluster label for each sample object in the source data input $\boldsymbol{D}$:* For each MR, map each sample object $\mathbf{x}_i^s \in D$ to a new object $\mathbf{x}_i^f \in D^T$ (with changed or unchanged attribute values). To understand how the clustering result changes after applying the data transformation T, it is necessary to compare the returned label for each object $\mathbf{x}_i^s \in D$ and its corresponding object $\mathbf{x}_i^f \in D^T$.
2) *Consistency between the expected label and the actual label for each newly added sample in $\boldsymbol{D^T}$:* Apart from mapping source data objects into their corresponding follow-up data objects, some MRs may also involve creating new objects such that users may have different expectations for the behaviors of these newly inserted objects. The newly added objects may share the same label with their neighbors, or may be assigned a new label. We will illustrate different expectations in corresponding MRs in Section IV-C.

In view of the above two aspects, we propose the notion of *reclustering percentage* (RP), which measures the inconsistency between a source output and its corresponding follow-up output. This notion is formally defined as follows.

3) Reclustering Percentage: Given a clustering system A, an MR, and a source input dataset $D = \{\mathbf{x}_1^s, \mathbf{x}_2^s, \ldots, \mathbf{x}_n^s\}$, by applying the data transformation T to D with respect to this MR, we obtain the corresponding follow-up input dataset $D^T = \{\mathbf{x}_1^f, \mathbf{x}_2^f, \ldots, \mathbf{x}_m^f\}$ (where $n = m$ if no new objects are added; $n < m$ if there are new objects inserted into D^T). Let d_{old} denote

the number of cases where $\mathbf{x}_i^f$ has a cluster label different from that of $\mathbf{x}_i^s$ (where $1 \leqslant i \leqslant n$); d_{new} denote the number of cases where a newly added object $\mathbf{x}_j^f$ has a different cluster label than expected (where $n < j \leqslant m$); $|D|$ denote the size of the source input dataset; and $|D^T|$ denote the size of the follow-up input dataset. RP is defined as

$$\text{RP} = \frac{d_{\text{old}} + d_{\text{new}}}{|D^T|}.$$

Obviously, $\text{RP} = 0$ if no violation to MR is observed between this pair of source and follow-up executions.

It should be reminded that, in the above definition:

1) We do not adopt some general similarity coefficients, such as Jaccard that calculates the intersection over union, because the RP measure we defined above serves our purpose more precisely.
2) The above definition may extend beyond the necessary properties of a clustering system, because our purpose is to validate the characteristics of a clustering system instead of detecting the source code faults in its corresponding implementation. In particular, if the clustering results generated from two related datasets do not follow the specified relation in an MR definition, a violation is said to be revealed and the characteristics of the corresponding system should be evaluated in detail to identify how and why these characteristics affect the clustering results.

Also, it is not difficult to see from the above that, by configuring the transformation T with various operations, the various behaviors of a clustering system can be validated.

C. Generic MRs

METTLE aims to provide an effective vehicle for end users without the need for a theoretical background of clustering to assess their expected characteristics for a clustering system, and validate the appropriateness of a clustering result in their own context.

To support METTLE, we developed an initial suite of 11 generic MRs. Each of these generic MRs is defined based on users' general expectations on the clustering result when a dataset changes in a particular way. These expectations are not gained from the theoretical background of any particular machine learning system, but from intrinsic and intuitive requirements of a clustering task. In other words, METTLE is primarily developed for users, without the need for a solid theoretical foundation of machine learning.

These 11 generic MRs fall into six different aspects of properties of a clustering system, and are expected to be applicable across various users' perspectives. Note that these generic MRs do not cover every possible property of a clustering system that is expected by all users, because different users may have different sets of expectations of a clustering system. This problem, however, is not an issue in METTLE because users can, at their own will, simply adopt any of these 11 generic MRs, and also define additional, more specific MRs for their specific scenarios of applications.

In contrast to a purely theoretical analysis on the properties of a clustering system, METTLE takes a lightweight and more practical approach to its application. METTLE helps users determine the relative "usefulness" among a set of clustering systems in different specific scenarios. This, in turn, facilitates the comparison and selection of the most appropriate clustering system from the user's perspective. Below, we discuss these 11 generic MRs we developed.

1) *Manipulating the sample object order in the dataset:* Reordering sample objects is a frequently performed operation, and users often assume that this operation is trivial and, hence, does not affect the clustering result. However, this assumption is not held for some clustering systems, such as KM [41] as discussed in Section III-B. To validate whether or not this assumption is held for a clustering system, MR1.1 and MR1.2 are defined as follows.

   MR1.1—Changing the object order: If we permute the order of the sample objects in the dataset, the new clustering result (R_f) will remain the same as the original result (R_s).

   MR1.2—Changing the object order but keeping the same set of starting centroids: If we permute the order of the sample objects in the dataset but keeping the same set of starting centroids, we will have $R_f = R_s$.

   In MR1.2, *starting centroids* are those objects that are randomly selected by a clustering system when it starts execution. Thus, by fixing the starting centroids, we can alleviate the randomness problem (i.e., the same dataset gives rise to different clustering results) associated with system execution. Consider, for example, KM. It randomly selects k objects from D as the initial cluster centroids, then assigns each object to the cluster with the closest centroid. Clusters are, then, formed by recomputing cluster centroids and reassigning data objects. With respect to this property of the system, if we fix k initial objects when it starts execution, followed by shuffling the other objects in D, it is generally expected that $R_f = R_s$, leading to MR1.2 above.

   It should be noted that MR1.1 differs from MR1.2 in that the former may or may not involve changing the starting centroids, but the latter keeps the starting centroids unchanged.

2) *Manipulating the distinctness among clusters in the dataset:* Users often expect that the distinctness among clusters will affect the clustering result. First, we consider the impact on the clustering result by shrinking some or all of the clusters towards their centroids in the dataset [see Fig. 2(a)]. MR2.1 is defined accordingly as follows.

   MR2.1—Shrinking one or more clusters towards their centroids: If some or all of the clusters in the dataset are shrunk toward their centroids, we will have $R_f = R_s$.

   The rationale behind MR2.1 is obvious and needs no explanation. With respect to MR2.1, for each cluster C_k in R_s to be shrunk, we first identify its centroid $\mathbf{m}_k$ returned by the clustering system. Then, for each $\mathbf{x}_i$ in C_k, we

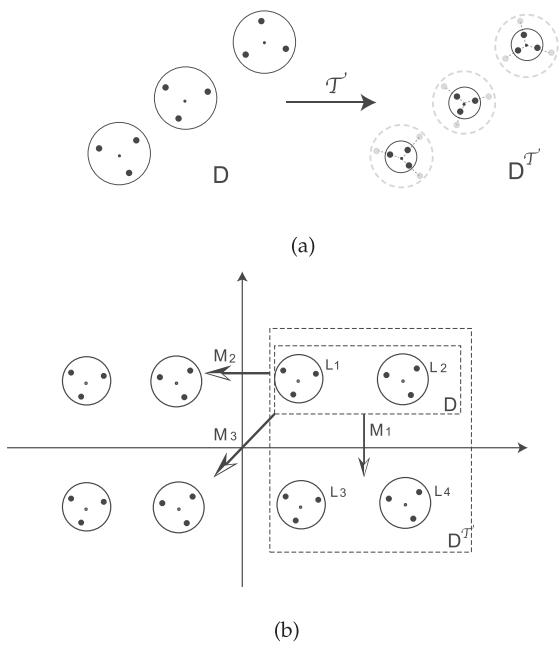

Fig. 2. Illustration on MR2.1 and MR2.2. (a) MR2.1. (b) MR2.2.

compute the middle point (denoted as $\mathbf{x}_i^k$) from $\mathbf{m}_k$ to $\mathbf{x}_i$. D^T is constructed by replacing all $\mathbf{x}_i$ in D with $\mathbf{x}_i^k$. Another aspect related to changing the distinctness among clusters is data mirroring, which is related to the following MR.

> *MR2.2—Data mirroring:* Given an initial dataset D such as its corresponding R_s contains k clusters in the *same* quadrant of the space. If we mirror all these k clusters in R_s to other N quadrants of the space so that clusters have approximately equal distance to each other, then $((N+1)*k)$ clusters will be formed in R_f. Furthermore, the newly formed clusters in R_f will include the original k clusters in R_s.

To illustrate MR2.2, let us consider a two-dimensional (2-D) space in Fig. 2(b). Suppose, after the first execution of a clustering system A, R_s contains two clusters L_1 and L_2. We, then, segment the space into four quadrants, where L_1 and L_2 are in the *same* quadrant. With the mirroring operation M_1, we mirror L_1 and L_2 (and the sample objects contained in them) in D to an adjacent quadrant to create new "mirroring" clusters L_3 and L_4. A new dataset D^T is created, containing the original clusters (L_1 and L_2 before mirroring) and the newly formed "mirroring" clusters (L_3 and L_4 after mirroring). We, then, perform two more mirroring operations (M_2 and M_3) in Fig. 2(b) similar to M_1 to create additional "mirroring" clusters. Finally, we perform another execution of A, and compare the clusters in R_s and R_f to see whether or not MR2.2 is violated.

3) *Manipulating the object density of one or more clusters in the dataset:* Suppose additional sample objects are added into some clusters in the dataset D to increase the object densities of these clusters (see Fig. 3). With respect to this action, users will expect that every newly object added to a cluster L (before executing the clustering system) will indeed be assigned to L by the clustering system after its execution. In reality, however, not every clustering system meets such user's expectation. To validate the behavior of a clustering system with respect to the change in the object densities of clusters, we define the following MR.

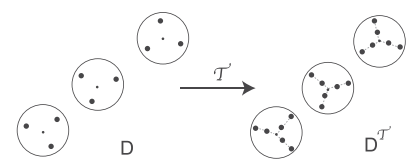

Fig. 3. Illustration on MR3.1.

> *MR3.1—Adding sample objects around cluster centers:* If we add new sample objects to a cluster L in R_s so that they are closer to the centroid of L than some existing objects in L, followed by executing the clustering system again, then: 1) all the clusters appearing in R_s will also appear in R_f, and 2) these newly added sample objects will also appear in R_f and with L as their cluster.

MR3.1 can be validated in a similar way as to validating MR2.1 but with some changes. First, similar to validating MR2.1, we create a new object $\mathbf{x}_i^k$ for an existing object $\mathbf{x}_i$ in a given cluster C_k of R_s, such that $\mathbf{x}_i^k$ is the middle point between $\mathbf{x}_i$ and the centroid $\mathbf{m}_k$. However, for validating MR3.1, we do not create $\mathbf{x}_i^k$ for each $\mathbf{x}_i$. Rather, we randomly select $\mathbf{x}_i$. Second, the newly created $\mathbf{x}_i^k$ is added as a new element, instead of replacing the original $\mathbf{x}_i$ as for validating MR2.1.

MR3.1 can be slightly revised to create another MR (MR3.2); the latter involves adding sample objects near the boundary of a cluster.

> *MR3.2—Adding sample objects near a cluster's boundary:* If we randomly add new sample objects on the edge of the convex hull[1] of the objects whose cluster is L, followed by executing the clustering system again, then 1) all the clusters appearing in R_s will also appear in R_f, and 2) these newly added objects will also appear in R_f and with L as their cluster.

4) *Manipulating attributes:* Attributes in a dataset may be occasionally changed. We consider two possible types of transformation on attributes. First, new attributes may be added to a dataset, if they are considered representative for distinguishing sample objects. In view of this addition, MR4.1 is defined as follows.

> *MR4.1—Adding informative attributes:* We define an *informative attribute* as the one whose value for each object $\mathbf{x}_i = \{x_1^i, x_2^i, \ldots, x_d^i\}$ is the corresponding returned cluster name l_i in R_s (l_i could be any $C_k \in R_s$). D^T is constructed by adding this new informative attribute to D, that is, each object $\mathbf{x}_i^T$ in D^T becomes $\mathbf{x}_i^T = \{x_1^i, x_2^i, \ldots, x_d^i, l_i\}$. Then, we will have $R_f = R_s$.

[1]In mathematics, the *convex hull* of a set X of points in the Euclidean plane is the smallest convex set that contains X.

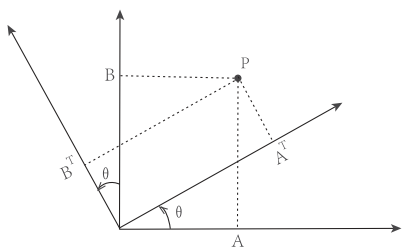

Fig. 4. Illustration on MR5.1.

Next, we consider the second type of data transformation. An attribute is generally considered as *redundant* if it can be derived from another attribute [32]. Redundancy is a critical issue in data integration, and its occurrence can be detected by correlation analysis. Han *et al.* argued that a high correlation generally indicates that an attribute can be considered redundant and, hence, to be removed. To define an MR related to redundant attributes, we adopt a widely used Pearson's product moment coefficient to measure the degree of correlation between attributes, and construct D^T by removing redundant attributes (if any). Intuitively speaking, we expect removing redundant attributes will not affect the clustering result. This expectation leads to the following MR.

> *MR4.2—Removing redundant attributes:* If we remove one or more redundant attributes from the dataset D and, then, execute the clustering system again, we will have $R_f = R_s$.

5) *Manipulating the coordinate system:* Several ways exist for manipulating the coordinate system such as rotation, scaling, and translation. These ways of changing the coordinate system shall not affect the spatial distribution of sample objects, thereby leading to the next two MRs.

> *MR5.1—Rotating the coordinate system:* Suppose the original coordinates are (A, B). We perform a transformation T by rotating the coordinate system by a random degree θ (where $\theta \in [0°, 90°]$) anticlockwise. After performing T, we get the new coordinates (A^T, B^T). The same set of clusters will appear in both R_s and R_f.

Fig. 4 depicts the transformation T. The formula below can be used to transform the existing coordinates in D into the corresponding new coordinates in D^T

$$\begin{pmatrix} A^T & B^T \end{pmatrix} = \begin{pmatrix} A & B \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

A scaling transformation changes the sizes of clusters. Scaling is performed by multiplying the original coordinates of objects with a scaling factor.

> *MR5.2—Scaling the coordinate system:* Suppose the original coordinates are (A, B); the scaling factors for the two axes are S_a and S_b, respectively; and the new coordinates after scaling are (A^T, B^T) (the mathematical representation of this scaling transformation is shown in the formula below). When $S_a = S_b$, we will have $R_f = R_s$
>
> $$\begin{pmatrix} A^T & B^T \end{pmatrix} = \begin{pmatrix} A & B \end{pmatrix} \begin{pmatrix} S_a & 0 \\ 0 & S_b \end{pmatrix}.$$

6) *Manipulating outliers:* An *outlier* is a data object that acts quite different from the rest of the objects, as if it were generated by a different mechanism [32]. It is generally expected that a clustering system will handle outliers by either filtering them or assigning new cluster labels to them. In this article, we mainly focus on *global outliers*, which do not follow the same distribution as other sample objects and significantly deviate from the rest of the dataset [32].

> *MR6—Inserting outliers:* To generate D^T, we add a sample object X_o to the dataset D so that the distance from X_o to any cluster is much larger than the average distance between clusters (in order to make X_o not associated with any predefined clusters in D). After this operation, the following properties must be met: 1) every object (except X_o) has the same cluster label in both D and D^T, and 2) X_o does not occur in R_f, or if X_o occurs in R_f, then X_o has a new cluster label, which is not associated with all the other objects.

V. Experimental Setup

This section outlines the setup of our experiment, which follows the guidelines by Wohlin *et al.* [44] as far as possible. In what follows, we first define the main objective and research questions of our experiment. This is followed by discussing the subject clustering systems used in the experiment. Thereafter, we discuss the detailed experimental design, including environment configuration, experimental procedures, dataset preparation, and parameter setting.

A few properties corresponding to some of the generic MRs discussed in Section IV-C were *individually* investigated in some previous studies. For example, it has been reported in [45] that the performance of KM depends on the initial dataset conditions. More specifically, some initial dataset conditions may cause KM to produce suboptimal clustering results. As another example, density-based clustering systems are found to be generally efficient at separating noises and outliers [46]. However, few studies have been conducted to provide a systematic, practical, and lightweight approach for validating a set of clustering systems with reference to various properties (defined from the user's perspective) in a comprehensive and holistic manner.

A. Research Objective and Questions

The main objective of our experiment is to demonstrate, by means of quantitative and qualitative analyses, the feasibility and practicality of METTLE for assessing and validating clustering systems with respect to a set of system characteristics as defined by users. In this article, we do not intend to conduct a comparative experiment with other "traditional" cluster validation techniques. This is because most of these techniques take a statistical perspective while METTLE focuses on the users' perspective; this difference in perspective renders a comparison meaningless.

In view of the above research objective, the following two research questions have been set.

1) RQ1: What are the levels of compliance of the subject clustering systems with the 11 generic MRs?
2) RQ2: What are the underlying behaviors of the subject clustering systems that cause violations to the relevant MRs (if any)?

B. Subject Clustering Systems

Our experiment involved six popular clustering systems obtained from the open source software Weka (version 3.6.6) [47]. These six subject systems fall into three categories: prototype-based, hierarchy-based, and density-based.

1) Prototype-Based Systems: Given a dataset $D = \{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n\}$ that contains n instances; each instance has d attributes. The main task of prototype-based systems is to find a number of representative data objects (known as *prototypes*) in the data space. More specifically, an initial partition of data is built first, then a prototype-based system will minimize a given criterion by iteratively relocating data points among clusters. In this category, we specifically considered the following three methods:

a) k-means (KM): Let $m^{(t)}$ denote the cluster centroid of each cluster, where t is the number of iterations. In essence, KM [41] involves the following major steps.

1) Randomly choose k data points as the initial cluster centroids.
2) Assign each data point to the nearest centroid, using the following formula (in which $\|\|$ means the L2 norm)

$$C_i{}^{(t)} = \{\mathrm{x}_p : \|\mathrm{x}_p - \mathrm{m}_i{}^{(t)}\| \leq \|\mathrm{x}_p - \mathrm{m}_j{}^{(t)}\|\}.$$

The above formula follows the notation in Definition 1 in Section II-A, where $C_i{}^{(t)}$ denotes the ith cluster in the tth iteration, and C_i denotes a set of points whose label is the current cluster.

3) Recalculate the centroid of each cluster, the new centroid is as follows:

$$m_i{}^{(t+1)} = \frac{1}{|C_i{}^{(t)}|} \sum_{\mathrm{x}_j \in C_i{}^{(t)}} x_j$$

where m_i is the centroid of cluster C_i, and $\mathrm{m}_i^{(t+1)}$ is the new centroid.

4) Repeat steps 2) and 3) above until there is no further change in clusters or the predefined maximum number of iterations is reached.

b) x-means (XM): This system addresses two weaknesses of KM: 1) poor calculation ability, and 2) the need for foreknowing the value of k and the local minima [48]. Unlike KM, XM only needs users to specify a range of k values so that XM can arrive at an optimal cluster number. The major steps of XM are as follows.

1) Run conventional KM, where k equals to the lower bound $k_{\min}$ of the specified range.
2) Split some centroids into two by calculating the value of the Bayesian Information Criterion [49].
3) Repeat 1) and 2) until $k > k_{\max}$, where $k_{\max}$ is the upper bound of the specified range.

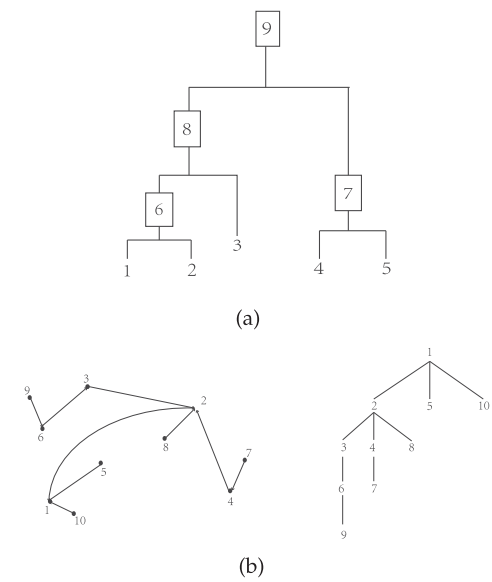

Fig. 5. Examples of clustering results generated by hierarchy-based clustering systems. (a) AN. (b) FF.

c) Expectation-Maximization (EM): This system aims at finding the maximum likelihood of parameters in a statistical model [50]. EM consists of the following major steps.

1) Initialize the distribution parameter θ.
2) E-step: Calculate the expected value of the unobserved variable $z^{(i)}$ with respect to the current estimate of the parameter θ; thereby, indicating the class to which the data object i belongs

$$Q_i(z^{(i)}) = p(z^{(i)}|x^{(i)}; \theta).$$

3) M-step: Find the parameter that maximizes the log likelihood function using the following formula:

$$\theta = \arg\max_\theta \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) log \frac{p(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}.$$

2) Hierarchy-Based Systems: This category of systems aims at building a hierarchy of clusters by merging or splitting data partitions, and the results are usually presented as a dendrogram. Fig. 5 shows the resulting clusters generated by two popular hierarchy-based methods: agglomerative nesting (AN) and farthest-first traversal (FF).

a) Agglomerative Nesting: This system adopts a bottom-up approach, where each data object is initially considered as a cluster in its own and, then, pairs of clusters are successively merged where appropriate. The clustering process has the following steps.

1) Assign each data point to a single cluster.
2) Evaluate the pairwise distance between clusters by a distance metric (e.g., the Euclidean distance) and a linkage criterion.
3) Merge the closest two clusters into one cluster according to the calculated distance.
4) Repeat steps 2) and 3) above until all relevant clusters have been merged into a single cluster that contains all data

points. The clustering result of AN is typically visualized as a dendrogram as shown in Fig. 5(a).

The linkage criterion (denoted as linkType) used in step 2) determines the distance between sets of observations as a function of the pairwise distances between these observations. Some commonly used criteria for linkType are single-linkage, complete-linkage, and average-linkage. Single-linkage could lead to a bad behavior known as "chaining," while complete-linkage, being an opposite extreme of single-linkage, suffers from the problem of "crowding" [51]. Take average-linkage as an example, the distance between two clusters is calculated as follows:

$$d(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x_p \in C_i} \sum_{x_q \in C_j} d(\mathbf{x}_p, \mathbf{x}_q).$$

AN does not require a prespecified number of clusters (i.e., k). However, the dendrogram should be cut at some point if we want a partition of disjoint clusters. Some criteria can be used to determine the cut point such as similarity level, or just a specific k, which is preferred in our approach.

b) Farthest-First Traversal: It consists of the following three main steps [51].

1) Randomly pick a point from n data points as a starting point and label it as 1.
2) Number the remaining points using FF: For $i = 2, 3, \ldots, n$, find the unlabeled point furthest from the set $\{1, 2, \ldots, i-1\}$ and label it as i (using the standard notion of distance from a point to a set: $d(x, S) = \min_{y \in S} d(x, y)$). For point i, let: $\pi(i) = \text{argmin}_{j<i} d(i, j)$ be its parent, and $R_i = d(i, \pi(i))$ be its distance to $\pi(i)$. A tree T^π is, then, constructed on nodes $\{1, 2, \ldots, n\}$, rooted at 1 and with an edge between each point i and its parent $\pi(i)$. An example is shown in Fig. 5(b).
3) Obtaining the ordering of points (i.e., T^π) from step 2). The first k points are regarded as k cluster centers, where the remaining points in T^π are assigned to their closest centers.

3) Density-Based Systems: Many clustering systems are distance-based, thereby exhibiting the limitation on discovering nonconvex clusters. On the other hand, density-based clustering systems (implemented under a data connectivity criterion) can efficiently identify clusters of arbitrary shape. We found two density-based clustering systems in Weka 3.6.6: DS and Ordering Points To Identify the Clustering Structure (OPTICS). Since OPTICS does not deliver the clustering result explicitly, we only chose DS in our experiment.

a) Density-Based Spatial Clustering of Applications With Noise (DS): Given a dataset with n points in a space, DS groups data points in high density areas. Data points are labeled one of the following three types.

1) *Core points:* A point m is labeled as *core* if there exist at least a minimum number of points (minPts) that are within the specific distance eps of m. Also, these points are said to be directly reachable from m. The number of points whose distances from m are smaller than eps is called *density*.
2) *Density-reachable points:* A point n is said to be *density-reachable* from m if there exists a path of points $p = \{t_1, t_2, \ldots, t_k\}$, where $t_1 = m$, $t_k = n$, and for any t_i in p, t_{i+1} is directly reachable from t_i.
3) *Noisy points:* A point is marked as *noise* if it is unreachable from any other points.

DS involves the following three main steps.

1) Randomly select an unvisited point from the dataset.
2) If the selected point is a core point, then label all its density-reachable points as one cluster. Otherwise, exit.
3) Repeat steps 1) and 2) above until all points have been visited.

C. Experimental Design

1) Environment Configuration: The experimental environment was configured as follows. Hardware environment: Intel Core i7 CPU with 8 GB memory. Operating system: Windows 10 X64. Software development platform: Java.

2) Experimental Procedures: Our experiment involved two main steps as follows.

Step 1: We evaluated the performance of each subject clustering system with respect to the 11 generic MRs as discussed in Section IV-C (*RQ1*). In particular, for each system, we measured the extent of violations to these generic MRs. In general, the fewer the violations an MR reveals, the better a clustering system fits the requirement (which is expressed in that MR) of a user. To measure the extent of violation, we used two metrics: 1) *violation rate (VR)*—it is the ratio of the number of violated trials to the total number of trials; 2) *RP*—it is the ratio of the number of objects being reassigned to the total number of objects within the follow-up dataset (previously defined in Section IV-B). We used the mean value of *RP* across all trials (with different dataset per trial) to measure the extent of a clustering system that violates an MR. To reduce the effect of irrelevant factors on the measurement, we followed the "blocking" design principle [44] in our experiment.

Step 2: For any violation to an MR, we investigated and analyzed the underlying behaviors of subject clustering systems that cause such violation, and analyzed the plausible reasons (*RQ2*). Here, we carefully examined the clustering results of both source and follow-up executions, with a view to identifying their corresponding clustering patterns. This facilitated us (and users) to better understand the relevant anomalous execution behaviors of a clustering system. The investigation result was, then, used to develop a list of strengths and weaknesses (with respect to the 11 generic MRs) for the six subject clustering systems.

3) Dataset Preparation and Parameter Setting: For the rest of the article, we call the dataset used for the *first* execution of a clustering system the *source dataset*, and the dataset (that has been changed according to a particular MR) used for the *second* system execution the *follow-up dataset*.

After selecting the subject clustering systems, we prepared a source dataset with clustered samples using the function make_blobs in Scikit-learn [52]. This function generates isotropic Gaussian blobs for clustering; that is, each cluster is a Gaussian distribution around a center point to ensure that the whole dataset is well clustered.

Let cluster_std denote the standard deviation of the clusters, centers denote the number of centers to generate (default $= 3$), n_features denote the number of features for each sample, and n_sample denote the total number of points equally divided among clusters. We set cluster_std, centers, and n_features to 0.5, 3, and 2, respectively. We also set n_sample to a valid range of [50, 200] because the larger the dataset was, the more likely violations to MRs were revealed.

Note that there were some special cases with specific arrangements. For MR2.2, only two well-separated clusters were generated with cluster_std $= 0.5$, and were mapped to the adjacent quadrant. As a result, altogether four distinctive clusters were generated in the follow-up dataset. For MR4.2, we generated an extra correlated attribute (A') with a particular Pearson correlation coefficient: each sample object is three-dimensional and is denoted as $(A,\ B,\ A')$, and Pearson$(A,\ A') = 0.8$. Let $\mathbf{x}_i^s = (A,\ B,\ A')$ be a source sample, and $\mathbf{x}_i^f = (A,\ B)$ be its follow-up sample. Note that, given a follow-up sample, the correlated attribute A' was removed from it to form its corresponding source sample. In MR5.2, S_a was randomly selected from the range $[0.2, 5.0]$ and S_b was set to the same value as S_a.

Based on each identified MR, follow-up datasets were derived from the corresponding source datasets. We ensured the object orders in the source datasets and follow-up datasets were properly aligned (except for MR1.1 and MR1.2 since both MRs involved changing the object orders). Because our experiment did not focus on the effect of the input parameters of the clustering systems, we fixed the parameters in each batch of experiments: 1) Euclidean distance was taken as the distance function for systems that require a distance metric; 2) linkType was set to "AVERAGE" (i.e., average-linkage criteria); 3) eps and minPts were set to 0.1 and 8, respectively, for DS; and 4) random seed, which is used for random number generation in Weka implementation, was also fixed across multiple executions of each subject system involving the source datasets and their corresponding follow-up datasets, in order to ensure the clustering results were reproducible and the initial conditions were properly aligned.

As explained in Section V-B, EM and DS do not need a prespecified cluster number. For the other four subject clustering systems, we set the parameter k as follows.

1) KM: Since centers was set to 3 (i.e., three source clusters), k was also set to 3 for all MRs except MR2.2.
2) XM: The permissible range was set to $[k-1,\ k]$. k was the actual number of clusters in a dataset and was set to 3 except MR2.2.
3) AN and FF: k was set to 3 for all MRs except MR2.2 and MR6.

VI. Experimental Results

This section presents our experimental results for the two research questions RQ1 and RQ2. Section VI-A addresses RQ1 by providing and discussing the relevant quantitative statistics. Section VI-B addresses RQ2 by providing an in-depth qualitative analysis, framed by a set of clustering patterns that we observed in the experiment. In addition, Section VI-C summarizes our observations and provides further analysis and interpretation of the results.

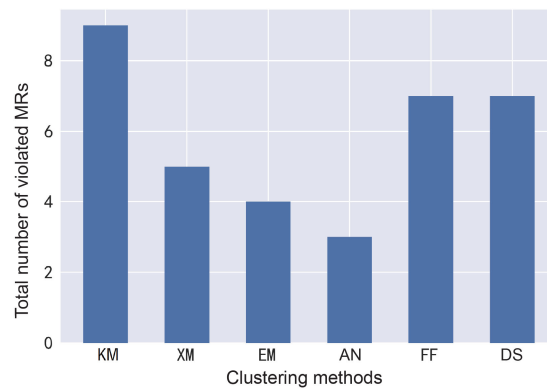

Fig. 6. Total number of violated MRs of each clustering system.

TABLE I
VR Values for Subject Clustering Systems With Respect to Generic MRs

Type of MRs	MR	Prototype-based			Hierarchy-based		Density-based
		KM	XM	EM	AN	FF	DS
1	1.1	5%	8%	0	0	90%	8%
	1.2	0	0	N/A	N/A	0	N/A
2	2.1	26%	0	0	N/A	0	N/A
	2.2	35%	0	0	0	62%	100%
3	3.1	7%	9%	5%	14%	95%	28%
	3.2	6%	10%	11%	15%	85%	21%
4	4.1	36%	0	0	0	8%	0
	4.2	17%	0	0	0	0	7%
5	5.1	57%	54%	9%	47%	91%	61%
	5.2	0	0	0	0	0	0
6	6	11%	12%	39%	0	92%	10%

A. Compliance of Clustering Systems With 11 Generic MRs (RQ1)

With respect to each of the six subject clustering systems, we conducted 100 trials (with different datasets in each trial) for each of the 11 generic MRs defined in Section IV-C. When validating a clustering system against an MR, an experimental trial was said to cause a "violation," if its corresponding RP was greater than zero (see Section IV-B for the details). This result indicated that there was at least one sample reclustered "unexpectedly" in the current trial. Also, a method was said to violate an MR if there was one or more violations in all the 100 experimental trials.

Fig. 6 summarizes the total number of violated MRs of each system. The figure shows that KM had the worst performance in that it violated nine MRs. It was followed by FF and DS—each of them violated seven MRs. XM and EM violated five and four MRs, respectively. AN performed the best because it had the smallest number of violations ($= 3$). Recall that every generic MR defined in Section IV-C involves data transformation in a certain way. Thus, in general, Fig. 6 indicates that KM is the most sensitive to data transformation, whereas AN is least sensitive.

Furthermore, we noted that even if two systems both violated the same MR, the chance of revealing a violation could be quite diverse. Therefore, we define the concept "VR" to facilitate a deeper analysis. Basically, *VR* is defined as the number of violation trials to all the 100 trials. Table I shows the values of VR for all methods with respect to each generic MR.

Consider, for example, in this table, VR $= 26\%$ for KM with respect to MR2.1. It indicates that, among the 100 experimental

trials, 26 of them had their RP values greater than zero. Consider another example. VR $= 0$ for XM with respect to MR2.1, indicating that none of the 100 experimental trials violated MR2.1. As a reminder, if "N/A" is indicated for a particular MR in Table I, it means that this MR is not applicable for the relevant system(s). For instance, MR2.1 requires cluster centroids to be returned by a system. Since AN and DS do not return any cluster centroid, so their corresponding VR values are labeled as "N/A."

1) Zero violation: Several systems had zero VR values for some MRs in Table I. These zero-violation cases not only indicate a high adaptability and robustness of the corresponding systems with respect to particular types of data transformation, but they also imply that the relevant MRs may be *necessary* properties of these systems and, hence, can be used for verification [25]. Consider, for example, the zero VR value of AN with respect to MR1.1. We can indeed prove that MR1.1 is a necessary property of AN. In this system, each data point is first considered a single cluster. Then, AN calculates the distances between clusters and incrementally merges two closest clusters. It is obvious that the distance calculation and the way of merging clusters are unrelated to the order of the data in the dataset. In addition, Table I shows that no violation to MR5.2 occurred across all the six subject systems. Thus, it can be argued that MR5.2 can be considered a necessary property of the six systems. Since this article mainly focuses on validation rather than verification, therefore, the formal proofs and analyses for zero-violation cases are excluded from this article. Note that this article mainly focuses on the nonzero violation cases.

2) Nonzero Violation: Table I shows that the nonzero VR values spread across a wide range from 5% to 100%. Intuitively speaking, with respect to an MR: 1) a high VR value indicates that a system is very sensitive to the type of data transformation corresponding to this MR, and the clustering result is likely to vary unexpectedly; and 2) a low VR value indicates that a system is relatively robust to the corresponding data transformation, and violations to this MR occur sporadically among all the experimental trials. Consider, for example, the values of VR of KM (5%) and FF (90%) with respect to MR1.1. The result indicates that KM violated MR1.1 in only five trials out of 100, while FF violated as many as 90 trials out of 100. Thus, the result shows that FF is far more sensitive to the type of data transformation corresponding to MR1.1 (i.e., changing the object order) than KM.

By examining how a system reclusters transformed data samples in each violated case, we observed that different cases had different levels of inconsistency as measured by RP. In other words, the nonzero RP values exhibited a diverse range. As an example for illustration, among the five violations to MR1.1 for KM (VR $= 5\%$ in Table I), we observed five diverse RP values (in ascending order): 0.55%, 0.67%, 0.93%, 46.67%, and 48.19% (mean $= 19.40\%$). Table II shows the mean values of RP for the nonzero violation cases for each system with respect to each generic MR. Due to page limitation, the table shows the mean values of RP rather than their individual values.

Note that Tables I and II show the results in different perspectives. Table I counts the numbers of violated cases; while Table II focuses on the mean numbers of inconsistencies among those violated cases. Also note that a high VR value does not necessarily imply a high RP value. Take FF under MR3.1 as an example. Here, reclustering occurred for 95 times among all the 100 trials (VR $= 95\%$). However, the mean percentage of reclustering was less than 8% (mean number of RP $= 7.94\%$). Thus, the results indicate that, although MR3.1 was often violated by FF, the extent of reclustering in these violations was quite marginal on average. In contrast to FF, although XM violated MR3.1 only nine times (VR $= 9\%$), this method had a mean value of RP of 35.75%.

TABLE II
MEAN VALUES OF RP FOR SUBJECT CLUSTERING SYSTEMS WITH RESPECT TO GENERIC MRS

Type of MRs	MR	Prototype-based			Hierarchy-based		Density-based
		KM	XM	EM	AN	FF	DS
1	1.1	19.40%	11.81%	0	0	7.40%	1.11%
	1.2	0	0	N/A	N/A	0	N/A
2	2.1	49.77%	0	0	N/A	0	N/A
	2.2	23.36%	0	0	0	8.51%	16.31%
3	3.1	7.62%	35.75%	1.22%	1.92%	7.94%	2.74%
	3.2	18.16%	17.28%	0.83%	1.67%	7.91%	4.48%
4	4.1	46.34%	0	0	0	7.46%	0
	4.2	5.74%	0	0	0	0	0.64%
5	5.1	3.84%	2.37%	3.79%	16.15%	15.84%	8.48%
	5.2	0	0	0	0	0	0
6	6	11.53%	11.83%	1.99%	0	7.93%	1.97%

† Each figure in the table denotes the mean value of RP over the violated trials with respect to the relevant MR.

We now turn to Fig. 7, which combines the results in Tables I and II in one figure. In this figure, each horizontal bar corresponds to a violation to a particular MR by a system. In each subfigure, the largest value of RP shown in the y-axis is 70%, because this was the largest RP value we observed across all the 11 MRs and all the six subject clustering systems in our experiment. In Fig. 7, we can easily observe the "density" of the occurrences of reclustering over a certain range. Consider, for example, the set of horizontal bars related to MR2.2 and DS in Fig. 7(f). By looking at the distribution pattern of the horizontal bars, we know that the values of RP had a larger deviation in the higher-value ranges (closer to 70%) than in the lower-value ranges (closer to zero percentage).

Below, we summarize the above findings.

1) The 11 generic MRs have different capabilities to help a user detect "unexpected" behavior in clustering systems (from the user's perspective). More specifically
   a) MR5.1 (related to the rotation of the coordinate system) is the most effective MR in identifying the corresponding "unexpected" behavior across all the six subject methods.
   b) Some generic MRs, particularly MR5.2, could be necessary properties of clustering systems and, as such, no violation has been observed.
2) The robustness of handling each type of data transformation (as represented by the relevant generic MR) varied across the clustering systems, in terms of the VR and RP measures. More specifically
   a) KM and FF had the worst performance across the 11 generic MRs.
   b) On the other hand, EM and AN stayed relatively robust, yielding more desired results.

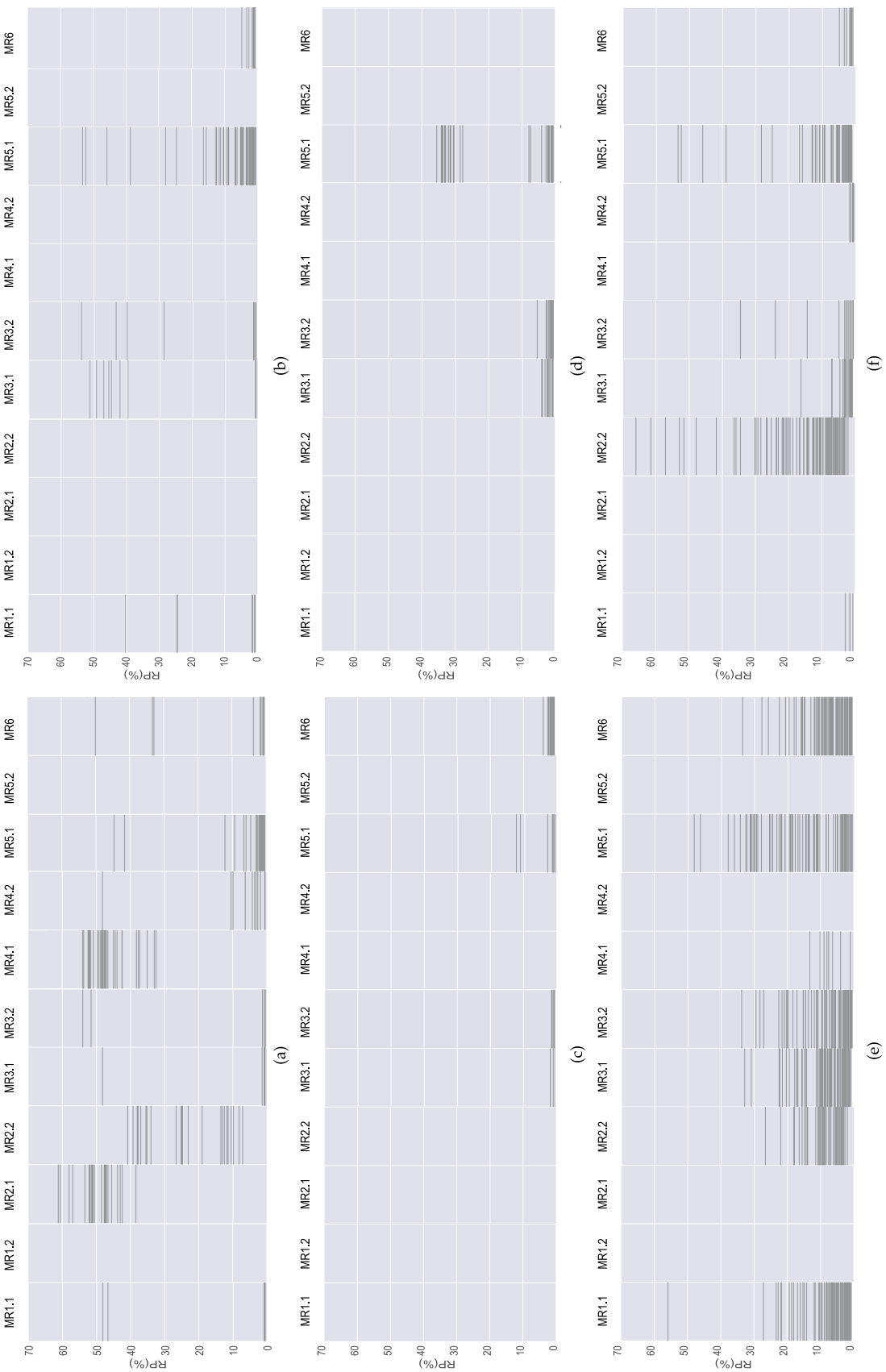

Fig. 7. RP distributions across the 11 generic MRs for each method. (a) KM. (b) XM. (c) EM. (d) AN. (e) FF. (f) DS.

TABLE III
DIFFERENT TYPES OF CLUSTERING PATTERNS AND THEIR RELATED CLUSTERING SYSTEMS

Pattern type	Description	Related Clustering Systems
BORDER	Data objects near the boundary of one cluster in the source dataset are reassigned to *different* clusters in the follow-up dataset.	All
MERGE & SPLIT	Two source clusters are merged into one follow-up cluster, *and* another source cluster is split into two smaller follow-up clusters.	KM, FF
SPLIT	One or more source clusters are split into smaller follow-up clusters.	EM, AN
NOISE	Reclustering mainly occurs for those objects that are considered "noise".	DS
NUM	The numbers of clusters differ between the results after the source and the follow-up executions.	DS

B. Underlying Behaviors that Cause Violations (RQ2)

This section complements Section VI-A by drilling down to the underlying behaviors and plausible reasons for the violations to each generic MR.

For each violation, we carefully inspected the results of both source and follow-up executions, by visualizing their clustering patterns. Such patterns are fairly evident and immediate to users; these patterns could easily and intuitively comprehend the anomalous behaviors of the subject clustering systems. Five types of clustering patterns were identified and shown in Table III. (Note that Table III only shows the pattern types that we observed in our experiment, rather than all the different possible pattern types.) In this table, each pattern type may be associated with more than one "similar" pattern with nonidentical data distributions and numbers of clusters. In what follows, we will illustrate the observed clustering pattern types and the underlying causes for their occurrence.

1) Violations Related to KM and XM: Fig. 7(a) and (b) shows the distributions of the RP values for KM and XM, respectively, with respect to all the 11 generic MRs. For both systems, their RP values generally varied across a wide range (between 0% and 70%). For all the violations related to both systems, we took a close examination of the clustering results, and revealed two types of clustering patterns. In other words, these two pattern types occurred for both KM and XM. For the rest of Section VI-B1, to avoid lengthy discussion, we mainly discuss the results related to KM, followed by a short discussion on the results related to XM.

a) BORDER: For those violations related to KM with relatively low RP values (e.g., RP $<$ 10%), some data points near the boundaries of clusters were reassigned to *different* clusters in the follow-up dataset, as shown in Fig. 8. For simple illustration, this figure only shows one data point (enclosed in a small box) reassigned from one cluster (near its boundary) to an adjacent cluster. [Note that the data point "■" in the box in Fig. 8(a) has become the data point "•" in the box in Fig. 8(b)]. However, in our experiment, more than one data points were observed to be reassigned to different clusters.

This pattern type was observed in the violations to MR1.1, MR3.1, MR3.2, MR5.1, and MR6. Some statistics on the violations related to BORDER and their RP values are provided as follows: (MR1.1) 60% violations (three out of five, where RP $<$ 1%). (MR3.1) 86% violations (six out of seven, where RP $<$ 1%). (MR3.2) 67% violations (four out of six, where RP $<$ 1%). (MR5.1) 96% violations (55 out of 57, where RP $<$ 10%). (MR6) 73% (eight out of 11, where RP $<$ 5%).

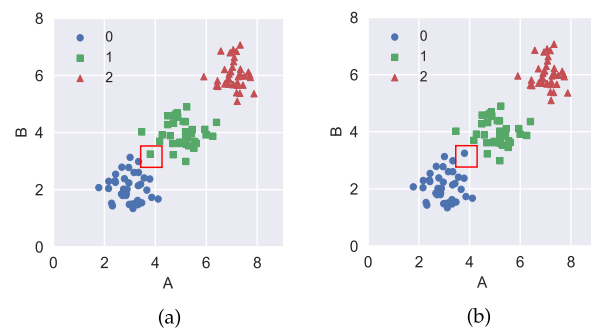

Fig. 8. Pattern type BORDER for KM. (a) Source dataset. (b) Follow-up dataset.

Apparently, some users may think that KM is sensitive to the initialization condition (i.e., the selection of starting centroids). Thus, even a slight change on the starting centroids caused by data transformation (such as reordering or adding data samples) could lead to fairly different clustering results. Below, we use MR1.1 (changing the object order) and MR5.1 (rotating the coordinate system) as examples to explain how data transformation affects the clustering results generated by KM.

Consider MR1.1 first. Reordering data samples has no effect on data distribution, but is likely to change the randomly initialized (starting) cluster centroids. We argue that, with the gradual relocation of cluster centroids following each iteration of reclustering, KM may finally generate a different set of data clusters. Our argument was validated by MR1.2 that no violation occurred if we changed the object order but keeping the same set of starting centroids. With respect to MR1.1, we carefully checked the clustering process and confirmed that the starting centroids in the *violated* trials were actually changed after changing the object order. But, at the same time, we also observed that many *nonviolated* trials involved changing their starting centroids. Therefore, the results have suggested that KM may not be as sensitive to the starting centroids as some users initially conceive.

Next, we turn to MR5.1. Many users of KM generally expect that rotating the coordinate system will not affect the clustering result, because such rotation does not change the data

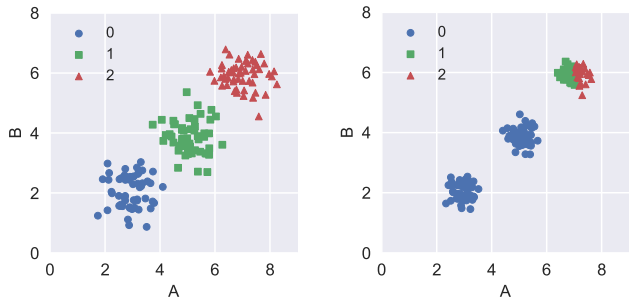

Fig. 9. Pattern type MERGE & SPLIT for KM. (a) Source dataset. (b) Follow-up dataset.

distribution pattern. However, this was not the case observed in our experiment; we found some "unexpected" violations to MR5.1. By inspecting the source code of KM collected from Weka, we found a function distance for calculating the Euclidean distance between an arbitrary object x_i and each cluster centroid. Before executing the core part of the distance computation, KM normalizes each attribute value with min-max normalization via the function norm. As such, the centroid m_k nearest to x_i will be chosen and x_i will be assigned with label k. By checking the output after each iteration of KM, we found that the normalized Euclidean distance between x_i and m_k was different between the source and the follow-up executions, although the theoretical distance remains unchanged after rotating the coordinates. Hence, a small change on the distance could result in a different decision by KM when choosing the nearest centroid. Furthermore, the impact of min-max normalization will be brought forward into subsequent iterations, thereby explaining the major reason for violating MR5.1.

b) MERGE & SPLIT: Most KM-related violations with their RP values larger than 10% were associated with this pattern type (see Fig. 9 for an example). For the MERGE & SPLIT pattern type, two source clusters are merged into one follow-up cluster, and one other source cluster is split into two smaller follow-up clusters.

This pattern type was associated with all the violations, which were related to all the 11 generic MRs except MR1.2 and MR5.2. Some statistics on the violations related to MERGE & SPLIT and their RP values are provided as follows: (MR1.1) 40% violations (two out of five, where RP $>$ 40%); (MR2.1) 100% violations (26 out of 26, where RP $>$ 38%); (MR2.2) 100% violations (35 out of 35, where RP $>$ 10%); (MR3.1) 14% violations (1 out of 7, where RP $>$ 40%); (MR3.2) 33% violations (2 out of 6, where RP $>$ 40%); (MR4.1) 100% violations (36 out of 36, where RP $>$ 30%); (MR4.2) 6% violations (one out of 17, where RP $>$ 40%); (MR5.1) 4% violations (two out of 57, where RP $>$ 40%); (MR6) 27% violations (three out of 11, where RP $>$ 30%).

It is commonly known that KM may quickly converge to a local optimum, resulting in unsatisfactory results. We conjecture that the MERGE & SPLIT pattern type occurred due to this reason. To test this conjecture, we compared the iteration numbers between the source and follow-up executions. Our rationale is

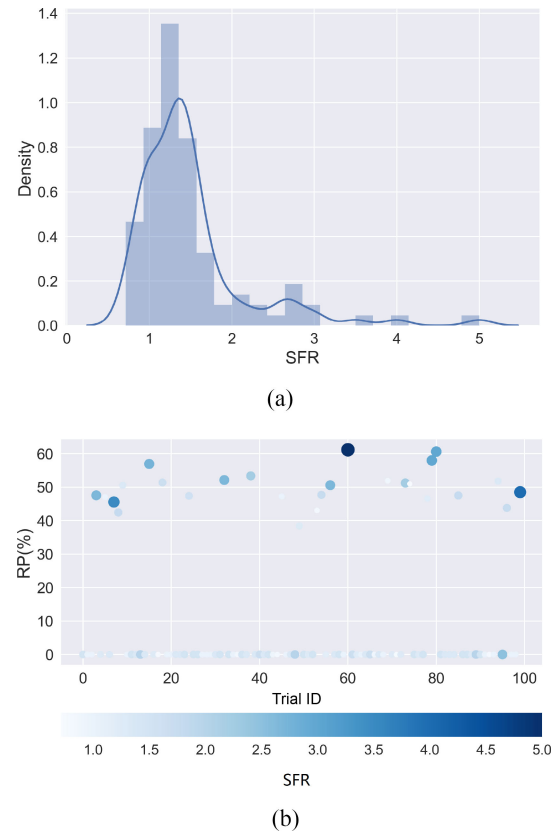

Fig. 10. Distributions of SFR and RP values related to MR2.1 (100 trials). (a) Histogram of SFR values related to MR2.1 (100 trials). (b) Distributions of RP and SFR values related to MR2.1 (100 trials).

based on the intuition that a low iteration number (i.e., an early iteration) is normally associated with high convergence speed, and high convergence speed is often a signal of prematurity, resulting in a local optimum.

Here, we use MR2.1 as an example for illustration: If a set of data samples can be well-clustered, then shrinking each cluster toward its centroid should make the clusters more compact, thereby producing an even more clearcut clustering result. Let I_s and I_f denote the iteration numbers in the source and follow-up clustering processes, respectively. Let SFR $= \frac{I_s}{I_f}$. Obviously, $SFR > 1$ indicates less iterations and a higher convergence speed in the follow-up clustering process; while SFR < 1 indicates the opposite situation. Fig. 10(a) illustrates the distribution of SFR values related to MR2.1 for 100 trials in a histogram. From this figure, we observed that among the 100 trials, 10% of them had their SFR values less 1.0, and 19% of them had their SFR values equal to 1.0. Among the remaining 71% of the trials whose SFR $>$ 1, 79% have $1 <$ SFR ≤ 2, 18% have $2 <$ SFR ≤ 3, and 3% have SFR > 3.

The main upper portion of Fig. 10(b) shows the distribution of RP values related to MR2.1 over 100 trials. Each dot at position (i, j) in the figure indicates that the ith trial had its corresponding RP $= j$. Note that the size and the darkness of the round dots are proportional to their SFR values: the larger and darker a dot is, the higher is its corresponding SFR value (and, hence, the higher is the convergence speed in the follow-up clustering process).

The horizontal bar at the bottom of Fig. 10(b) indicates the value ranges of SFR. According to the definition of RP, RP $= 0\%$ indicates no violation to the relevant MR, while RP $> 0\%$ indicates the existence of a violation. In all the violated cases related to MR2.1, data were clustered in patterns similar to Fig. 9, resulting in fairly high RP values. It can be seen from Fig. 10(b) that almost all trials with RP $= 0\%$ had their SFR values close to 1 (see the small and light dots on the horizontal line (i.e., those that are parallel and just above the x-axis) corresponding to RP $= 0\%$), indicating that the source and the follow-up processes had similar convergence speeds. On the other hand, those trials with very high RP values were most likely associated with high SFR values [see the large and dark dots on the horizontal line (i.e., those that are parallel and just above the x-axis in Fig. 10(b)], indicating that their follow-up processes were faster than the source processes. In particular, the large and dark dot for the trial ID 60 corresponds to a violated trial with RP $> 60\%$, and its follow-up process was about four times faster than its corresponding source process. Fig. 10(b) also shows a positive correlation between RP and SFR values with respect to MR2.1. All the above analyses have demonstrated that, with respect to KM, violations with high RPs were very likely due to an accelerated convergence to local optima. For other MRs with MERGE & SPLIT violation pattern in KM, we observed similar phenomena.

We now turn to XM. In terms of the violations to MR1.1, MR3.1, MR3.2, MR5.1, and MR6, XM was not better than KM. For these five MRs, the clustering pattern types observed for KM also occurred for XM. Thus, we do not repeat the discussion on the violations related to XM. However, we would like to point out that, when comparing with KM, XM was relatively more robust to the type of data transformation related to MR2.1, MR2.2, MR4.1, and MR4.2. A close examination on those violations related to these four MRs revealed that a common property existed, and that the resulting clusters were relatively more clear-cut (for MR2.1) or separated from each other (for MR4.1).

As an extension to KM, XM proposes a partial remedy for the local optimum problem [48]. Many people argue that XM is less sensitive to local optima by searching for the true number of clusters in a predefined range. This argument was validated to be valid by METTLE—XM outperformed KM in the situations where data groups were largely separated. On the other hand, in those situations where data groups were well clustered but with a lower degree of separation, XM and KM generated similar clustering results.

c) Summary: The sensitivity of KM and XM to initial conditions and noisy data was validated by our experiment. Data transformation, such as reordering data and adding noises, will result in reassigning data objects near the boundary of one cluster to another cluster, which is normally expected by users. Our experiment also revealed an important property of KM: this system tends to converge to local optima even when the clusters are sufficiently well separated, which leads to high RPs. Although XM is theoretically less sensitive to local optima than KM, our experiment results show that XM only outperforms KM when the original dataset is highly separated.

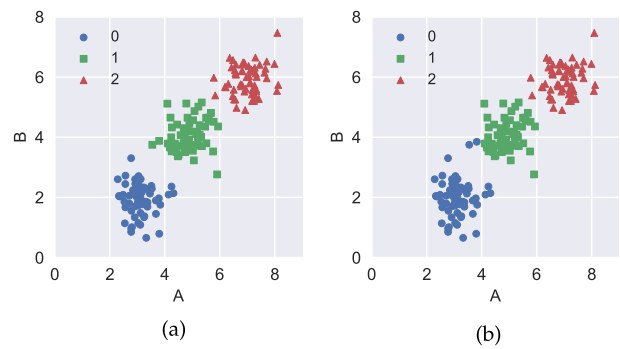

Fig. 11. Pattern type BORDER for EM. (a) Source dataset. (b) Follow-up dataset.

2) Violations Related to EM: Fig. 7(c) shows that, for EM, violations only occurred in those cases related to MR3.1, MR3.2, MR5.1, and MR6. Among the 100 trials, the numbers of violated cases were 5, 11, 9, and 39, respectively, for these four MRs. Each of these violations had a low RP value, indicating that very few data samples were reassigned from one cluster to another. Based on these results, we argue that EM is fairly robust to different types of data transformation. We also found two clustering pattern types: BORDER and SPLIT.

a) Border: We observed from Fig. 7(c) that most of the violations related to EM had fairly low RP values. As shown in Fig. 11, only several data samples near the boundaries of clusters were reassigned to other clusters by EM, and this clustering result was consistent with many users' expectation.

This pattern type was observed in those violations to MR3.1, MR3.2, MR5.1, and MR6. Some statistics on the violations related to this pattern type and their RP values are provided as follows: (MR3.1) 100% violations (5 out of 5, where RP $< 2\%$); (MR3.2) 100% violations (11 out of 11, where RP $< 1\%$); (MR5.1) 78% violations (7 out of 9, where RP $< 3\%$); (MR6) 100% violations (39 out of 39, where RP $< 3\%$).

The above statistics indicate that, although the clustering results generated by EM was affected by the types of data transformation corresponding to MR3.1, MR3.2, MR5.1, and MR6, the impact on the clustering results was fairly small (as shown by the very small RP values). Also, Fig. 7 shows that EM has the second smallest number of violated MRs ($= 4$) among all the six subject clustering systems. In this regard, EM has the best performance among the subject clustering systems according to the user's expectations (which are expressed in terms of the 11 generic MRs).

One issue is worth mentioning here. Similar to KM and XM, violations to MR5.1 (rotating the coordinate system) were also observed for EM. As we have pointed out in Section VI-B1, violations to MR5.1 (and also other generic MRs) have revealed a gap between the actual performance and the user's expectation about a method (in this case, EM). In the Weka implementation, EM initializes estimators by running KM ten times and then choosing the "best" solution with the smallest squared error for all the clusters. This chosen solution, then, serves as the basis for executing the E-step and the M-Step in EM (see

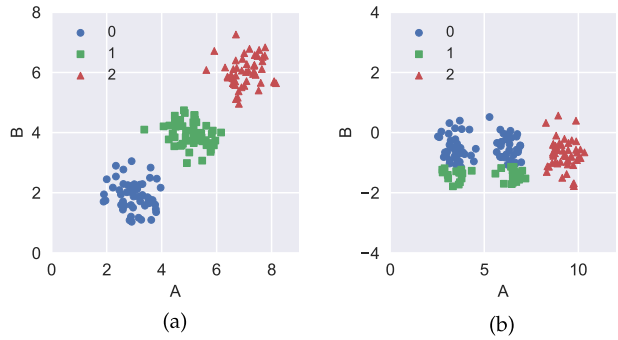

Fig. 12. Pattern type SPLIT for EM. (a) Source dataset. (b) Follow-up dataset.

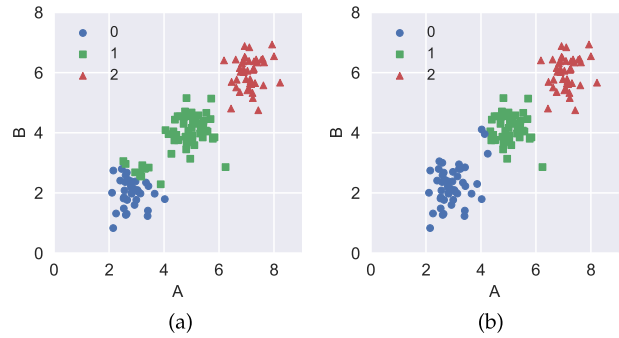

Fig. 13. Pattern type BORDER for AN. (a) Source dataset. (b) Follow-up dataset.

Section V-B1). Due to this reason, the clustering result generated by EM partially depends on KM, therefore it is not surprising to see that both EM and KM showed violations to MR5.1.

b) SPLIT: Fig. 12 shows an example of this pattern type: in the source dataset, each of the two clusters (the cluster with the "●" data points and the cluster with the "■" data points) clusters was split into two smaller clusters in the follow-up dataset; at the same time, merging of clusters in the source dataset did not occur.

This pattern type was only discovered in two out of the nine violations (=22%) to MR5.1, with their RP values over 10%. As explained above, EM partially depends on the KM solution. Thus, violations to MR5.1 by EM could occur after rotating the coordinates. After a close examination, we found that the theoretically "best" KM solution was not always what users normally expected. With respect to SPLIT, the chosen KM solution at the initialization stage was found to involve unexpected data partitions, which was similar to the pattern type shown in Fig. 12(b). This explains why EM generated poor clustering results after iterations based on the ill-initialization by KM.

c) Summary: According to the 11 generic MRs, EM is the most robust one among the six subject clustering systems. Reassigning data samples from one cluster to another cluster still occurred, which contradicted the user's expectation. However, since the RP values were very small ($<3\%$), the impact of data transformation on the clustering result was much less than the other five subject clustering systems. Although both EM and KM execute in an iterative manner, our experiment shows that EM is less sensitive to local optima than KM. Furthermore, in Weka implementation, the theoretically "best" solution chosen by EM during initialization may not be in line with the user's expectation, resulting in the poor clustering result generated by EM.

3) Violations Related to AN: It can be seen from Fig. 7(d) that AN only caused violations to three generic MRs: MR3.1, MR3.2, and MR5.1. The RP values associated with MR3.1 and MR3.2 were very low ($<5\%$ for both MRs). On the other hand, among all the RP values associated with MR5.1, some were under 10% but the others were fairly high ($>30\%$). We found two clustering pattern types from those violations related to AN.

a) Border: For those violations with low RP values ($<10\%$), the same clustering pattern type as shown in Fig. 13 was

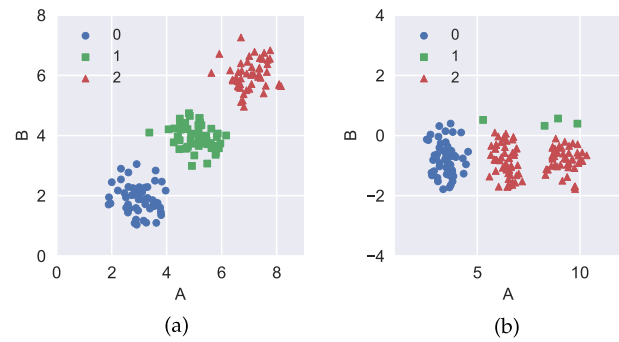

Fig. 14. Pattern type SPLIT for AN. (a) Source dataset. (b) Follow-up dataset.

observed. For these violations, only several data samples near the cluster boundaries were affected. BORDER was observed in all violations to MR3.1 and MR3.2, and in 43% (39 out of 91) violations to MR5.1.

b) SPLIT: For those violations with relatively high RP values ($>30\%$), we observed this pattern type (similar to the one shown in Fig. 14), where each of the two clusters (the cluster with the "■" data points and the cluster with the "▲" data points) in the source dataset was split into a small cluster (with the "■" data points) and a much larger cluster (with the "▲" data points) in the follow-up dataset. After checking the Weka implementation, we found that min-max normalization is also adopted in the preprocessing phase of AN, causing the violations to MR5.1.

c) Summary: As a hierarchy-based clustering system, AN is more robust to data transformation when compared with FF—only boundary points are occasionally affected. Our experiment also revealed that, similar to other systems, there exists a gap between the performance of AN and the user's expectation on this system.

4) Violations Related to FF: Fig. 7(e) shows that FF caused relatively more violations to the generic MRs when compared with other clustering systems, with the RP values ranged from 0% to 50%. We observed two clustering pattern types for FF.

a) BORDER: For the violations where RP $<30\%$, data samples near the cluster boundaries were reassigned to different clusters, as shown in Fig. 15. This pattern type appeared for MR1.1, MR2.2, MR3.1, MR3.2, MR4.1, MR5.1, and MR6. When compared with other methods, the reclustering of data

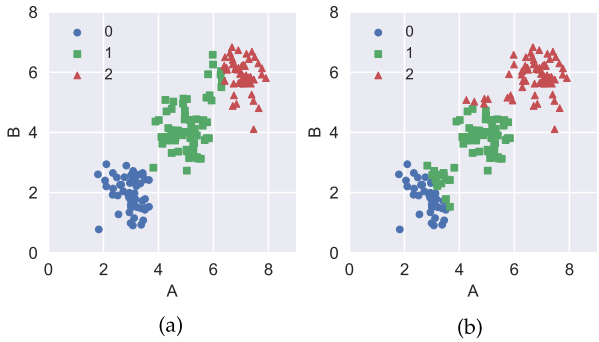

Fig. 15. Pattern type BORDER for FF. (a) Source dataset. (b) Follow-up dataset.

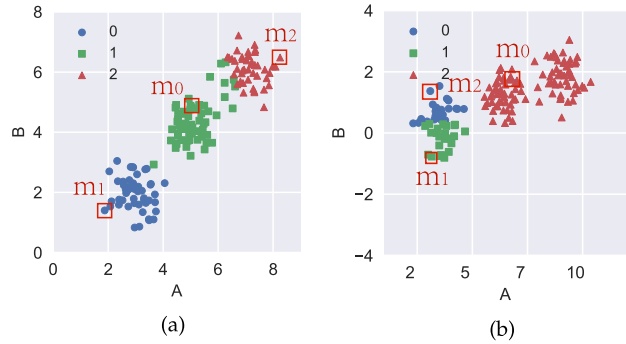

Fig. 16. Pattern type MERGE & SPLIT for FF. The points enclosed in red boxes are the cluster centroids. m_0, m_1, and m_2 denote the first, second, and third selected centroids, respectively. (a) Source dataset. (b) Follow-up dataset.

samples with this pattern type was not very precise. Consider, for example, in Fig. 15, a few data samples near the boundary of the cluster with the "■" data points in the source dataset were incorrectly reassigned to the cluster with the "▲" data points in the follow-up dataset.

It can be seen from Table I that FF caused many violations to MR1.1, with RP $= 90\%$. This result supports our analysis on FF, that the clustering process and result are largely affected by the starting centroids chosen by FF [51]. If the starting centroids selected by FF are changed by reordering the object order (MR1.1), the farthest-first traversal sequence may be affected. In addition, the fact that no violation to MR1.2 (this MR involves keeping the same set of starting centroids unchanged) was detected for FF further supports our analysis.

Similar to MR1.1, adding sample objects (MR3.1 and MR3.2) or inserting outliers (MR6) may change the farthest-first traversal sequence, thereby affecting the clustering results. For MR4.1, follow-up clusters should have better separation after adding informative attributes. However, unexpected results were still observed for FF.

For MR2.2, reclustering also occurred for the "marginal" points. Moreover, we found that the source execution generated inaccurate results where the points on the margin of one cluster were assigned to another cluster, while the follow-up execution generated four well-clustered results. This observation revealed a reclustering problem of FF with respect to MR2.2. As for MR5.1, we found that the violations were mainly due to the data normalization task during the preprocessing stage, and the effects of normalization varied across different violations.

For BORDER, only data samples near the boundaries were affected. For MERGE & SPLIT, data normalization had a greater impact on the clustering result, which will be discussed in detail below.

b) MERGE & SPLIT: This pattern type was observed in 15% violations to MR5.1 (rotating the coordinate system), with the RP values varied from 30% to 50%. We noted from the Weka implementation that min-max normalization will be applied before computing the Euclidean distance of a pair of data objects. FF will randomly select a starting centroid m_0 as the first cluster centroid, and will then select a farthest point m_1 from m_0 as the second centroid (the remaining centroids will be selected in the same way). Eventually, every data point will be assigned to its nearest centroid. By rotating the coordinates, data assignment could be different due to the slight change on the normalized distance.

Fig. 16 illustrates how the traversal sequence is affected in relation to MR5.1. We obtained the "same" (i.e., the instances with the same index) starting centroid m_0 in the source and follow-up executions by fixing the random seed during the experiment. After FF had finished the first traversal, different points were chosen as the second centroids m_1 in the source and follow-up executions. Similarly, after completing the second traversal, the third centroids m_2 in the source and follow-up executions were different. In the end, the resulting clusters turned out to be totally different between the source and follow-up datasets.

c) Summary: The traversal sequence of FF largely depends on the starting centroid. After a data object has been assigned to a cluster, it can no longer be moved around. Therefore, FF is much more sensitive to data transformation such as reordering the data sequence and inserting outliers (or noises). We found that FF is effective in recognizing an outlier and assigning it to a single cluster, without being much affected by data transformation. However, data transformation may cause data objects other than outliers to be reassigned to different clusters. Furthermore, FF occasionally does not generate clearcut and accurate clusters as expected, even when the data samples are well separated.

5) Violations Related to DS: Fig. 7(f) shows that violations to MR1.1, MR2.2, MR3.1, MR3.2, MR5.1, and MR6 occurred, with a wide range of RP values (between 0% and 70%). With further analysis, we noted that some points were "noises," representing a major difference on the clustering results between DS and other methods.

a) BORDER: This pattern type was observed in the violations to MR1.1, with RP $< 3\%$. In this pattern type, violations occurred near the cluster boundaries (see the points in the two boxes in Fig. 17), especially in those cases where clusters were close to each other. It has been reported by others (e.g., in [46]) that DS is almost independent of the order of the input data objects. In our experiment, however, we observed that, among the 100 trials with MR1.1, eight violations related to BORDER occurred. In each of these violations, a very small portion of data objects was found to be assigned to different clusters from the source

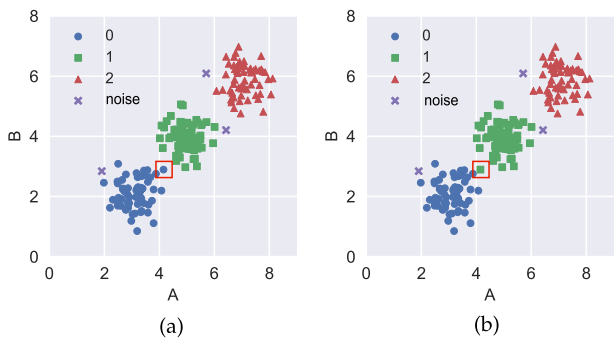

Fig. 17. Pattern type BORDER for DS. (a) Source dataset noise. (b) Follow-up dataset.

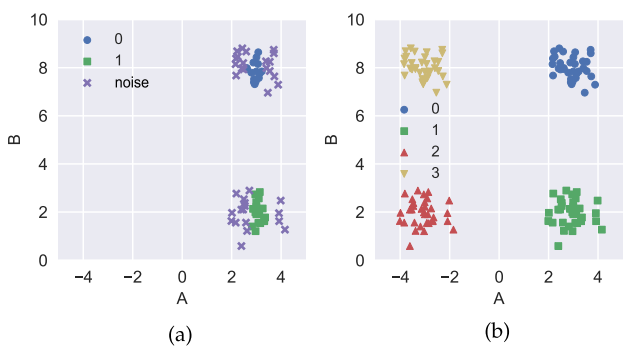

Fig. 18. Pattern type NOISE for DS. Blue dots in (a) denote the "noisy" data detected by DS, but there was no "noisy" data detected in (b). (a) Source dataset. (b) Follow-up dataset.

dataset to the follow-up dataset. These violations occurred due to a property of DS: if a data object was density-reachable from two neighbor clusters, the cluster to which this data object would be assigned was decided by the chronological sequence of detecting the clusters near that object. Nevertheless, DS was fairly robust to the type of data transformation corresponding to MR1.1 if data samples to be clustered were well separated.

b) NOISE: For DS, we observed another violation pattern type related to noisy data. This pattern type occurred in all the violations to MR2.2, MR3.1, MR4.2, and MR6.1; in 86% (18 out of 21) violations to MR3.2; and in 90% (55 out of 61) violations to MR5.1.

Consider MR2.2 (data mirroring) as an example. We noted from Fig. 18 that some points marked as "noisy" data in the source clusters turned out to be density-reachable points in the follow-up clusters. We also noted that the number of noisy data sharply dropped to zero or a tiny value after performing data mirroring as prescribed by MR2.2. We also observed that DS only generated good clustering results when the parameters eps and minPts were properly set. More specifically, when these two parameters were properly set so that noisy data did not occur in the source dataset, then no violation to MR2.2 would have occurred in the follow-up dataset. Similarly, for those violations to MR3.1, MR3.2, MR4.2, and MR6.1, the number of "noises" also decreased after performing the types of data transformation corresponding to these MRs.

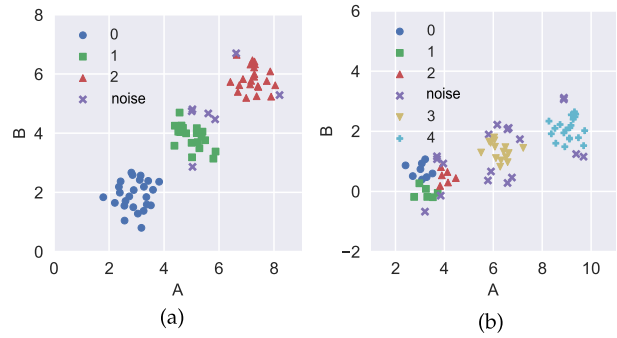

Fig. 19. Pattern type NUM of DS. (a) Source dataset. (b) Follow-up dataset.

By analyzing the implementation of DS, the above violations can be explained as follows. Suppose q denotes a density-connected point from a core point p in cluster c; o denotes a point in the eps-neighborhood of q, which is marked as "noise." After inserting new data points to c, there may be as many points as minPts within q's eps-neighborhood. Thus, q becomes a new core point, and o is density-connected to p so that o becomes a new member of cluster c. Hence, the number of noises is expected to decrease or remain unchanged after inserting new data points to a cluster. Our analysis result can be seen as a convenient quantification of the execution behavior of DS.

c) NUM: This pattern type was observed in three violations to MR3.2, and in six violations to MR5.1, where DS recognized an incorrect number of clusters. Take MR5.1 as an example. We found that DS unexpectedly divided the data samples into four or more clusters, and at the same time the number of samples labeled as "noises" [see the data points "×" in Fig. 19(a) and (b)] increased from the source dataset to the follow-up dataset. Since DS in the Weka implementation includes an embedded data normalization routine, the generated clustering result could be affected by even a slight change on the normalized distance among data objects.

d) Summary: Although the clustering result of DS is generally considered as not being affected by the input order of data samples, our experiment revealed that this was not the case due to the randomness of the system itself. When compared with other clustering systems, DS is effective in recognizing outliers. We also found that "noisy" data points are sensitive to data transformation. The configuration of parameters, which may have some impacts on the clustering result, is also important. As a whole, DS is robust to different types of data transformation—this is what users expect on a clustering system.

C. Summary and Further Analysis

We learnt from the analyses and discussions in Sections VI-B1–VI-B5 that, for all subject clustering systems, data samples located near the cluster boundaries were sensitive to even a small change in data input. This can be explained by the randomness of the system during its initialization. Moreover, those systems (KM, XM, and FF), which largely depend on the initialization conditions, showed a larger impact of data transformation on the clustering result, while EM, AN, and DS showed higher

TABLE IV
SUMMARY OF COMPLIANCE AND VIOLATION PATTERNS (WITH PLAUSIBLE REASONS) WITH RESPECT TO EACH GENERIC MR

MR	Clustering Systems					
	KM	XM	EM	AN	FF	DS
MR1.1	*BORDER*: random initialization of (starting) centroids; *MERGER & SPLIT*: tends to converge to local optima	*BORDER*: random initialization of (starting) centroids; *MERGER & SPLIT*: tends to converge to local optima	✓	✓	*BORDER*: random initialization of the starting centroid	*BORDER*: chronological sequence of detecting clusters
MR1.2	✓	✓	N/A	N/A	✓	N/A
MR2.1	*MERGE & SPLIT*: tends to converge to local optima	✓	✓	N/A	✓	N/A
MR2.2	*MERGE & SPLIT*: tends to converge to local optima	✓	✓	✓	*BORDER*: change of traversal sequence	*NOISE*: noisy objects become density-reachable objects
MR3.1	*BORDER*: random initialization of (starting) centroids; *MERGER & SPLIT*: tends to converge to local optima	*BORDER*: random initialization of (starting) centroids; *MERGER & SPLIT*: tends to converge to local optima	*BORDER*: deviation in parameter estimation	*BORDER*: deviation in distance computation	*BORDER*: random initialization of the starting centroid	*NOISE*: decrease in noisy objects (increase in core objects)
MR3.2	*BORDER*: random initialization of (starting) centroids; *MERGER & SPLIT*: tends to converge to local optima	*BORDER*: random initialization of (starting) centroids; *MERGER & SPLIT*: tends to converge to local optima	*BORDER*: deviation in parameter estimation	*BORDER*: deviation in distance computation	*BORDER*: random initialization of the starting centroid	*NOISE*: increase in density-reachable points
MR4.1	*MERGE & SPLIT*: tends to converge to local optima	✓	✓	✓	*BORDER*: change of traversal sequence	✓
MR4.2	*MERGE & SPLIT*: tends to converge to local optima	✓	✓	✓	✓	*NOISE*: increase in density-reachable points
MR5.1	*BORDER*: min-max normalization; *MERGER & SPLIT*: tends to converge to local optima	*BORDER*: min-max normalization; *MERGER & SPLIT*: tends to converge to local optima	*BORDER*: min-max normalization; *SPLIT*: suboptimal KM solution during initialization	*BORDER*: min-max normalization; *SPLIT*: min-max normalization	*BORDER*: min-max normalization; *MERGE & SPLIT*: min-max normalization	*BORDER*: min-max normalization; *NUM*: min-max normalization ;
MR5.2	✓	✓	✓	✓	✓	✓
MR6	*BORDER*: random initialization of (starting) centroids; *MERGER & SPLIT*: tends to converge to local optima	*BORDER*: random initialization of (starting) centroids; *MERGER & SPLIT*: tends to converge to local optima	*BORDER*: deviation in parameter estimation	✓	*BORDER*: deviation in distance computation	*NOISE*: decrease in noisy objects

† "✓" means that no violation has been revealed by the relevant MR, and "N/A" means that the relevant MR is inapplicable to the clustering system.

robustness to such change. Undoubtedly, users normally expect that the chosen clustering system will have high robustness to relocating data samples from near the boundary of one cluster to another cluster as a result of data transformation. Thus, in this aspect, EM, AN, and DS are more preferable than the other systems.

Table IV summarizes, for each subject clustering system, its compliance with or violations to the relevant generic MRs. Furthermore, for each violation case, we give the plausible reason(s) for its occurrence. Consider, for example, the cell related to KM and MR1.1. This cell indicates that KM exhibited two types of violation patterns (BORDER and MERGE & SPLIT) with respect to MR1.1. The cause of BORDER was due to the random initialization of the cluster centroid. For MERGE & SPLIT, it occurred because KM was trapped into a local optimum. Table IV not only summarizes our assessment results of the subject clustering systems, but also serves as a useful and handy checklist for users to make informed decisions on which clustering systems to choose with respect to their expectations. Also, users are allowed to assign different weights to different violation patterns. For example, if users consider that violations with the BORDER pattern are less important than violations with the MERGE & SPLIT pattern, then MERGE & SPLIT can be assigned a higher weight than BORDER. In this way, the compliance with each generic MR (in Table IV), together with its corresponding weighted score, can be used as a test adequacy criterion to be further leveraged by users for selecting an appropriate clustering system in accordance with the users' expectations. In addition, we also analyzed and summarized the relative strengths and weaknesses of the six subject systems with respect to the 11 generic MRs in Table V, with a view to facilitating users to gain a deep understanding of these systems.

Surprisingly, our experimental results (see Tables I and II) show that all the subject systems involved many violations to MR5.1 (rotating the coordinate system). A close examination of the corresponding source code found that min-max normalization is the major cause of the observed violations. More specifically, the normalized distance among data points could be different after nonlinear data transformation such as rotating the coordinates (even if the data distribution remains unchanged). Note that data normalization is a very important step in most machine learning systems—some of these systems (e.g., those available in Weka) have embedded a data normalization routine in them. Without using METTLE, users are unlikely to get an opportunity to understand the impact of the embedded normalization routine in machine learning systems.

TABLE V
SUMMARY OF RELATIVE STRENGTHS AND WEAKNESSES OF SIX SUBJECT CLUSTERING SYSTEMS

Clustering Systems	Strengths	Weaknesses
KM	Easy to understand and use	Sensitive to the random initialization of centroids and sensitive to outliers; may easily be trapped into a local optimum which may in turn lead to the occurrence of the MERGE & SPLIT pattern
XM	A partial remedy for the local optimum problem; shows great advantages over KM when data groups are sufficiently separated	Occasionally tends to underestimate the true number of clusters given a range $[k_{min}, k_{max}]$
EM	Strongly robust to various types of data transformation; less sensitive to a local optimum than KM	Depends partially on the KM solution during the initialization stage (EM initializes estimators by running KM 10 times and then choosing the "best" solution with the smallest squared error) and, hence, the clustering result is sometimes not what the users expect
AN	Independent of the order of input data; fairly robust to data transformation; strong ability to recognize outliers	Scaling datasets will possibly change the clustering results
FF	Involves less reassignment and adjustment, which may speed up the clustering process	Very sensitive to initialization conditions; sensitive to even a small change in data input; occasionally generates inaccurate clusters even when the data samples are well separated
DS	Effective in recognizing outliers	The values of the parameters eps and $minPts$ would have a large impact on the clustering result; noisy data are often relocated to different clusters after data transformation

The above discussion shows that, apart from assessing clustering systems and facilitating their selection, METTLE also supports program comprehension and end-user software engineering [42], [43], through which users can gain a deeper understanding of the program under test without the need for using relevant complex theories.

VII. METTLE AS A FRAMEWORK FOR SELECTING CLUSTERING SYSTEMS

Apart from assessing clustering systems, another potential application of METTLE is to help users select the most appropriate clustering systems. With more and more open-source software libraries that provide ready-to-use machine learning systems, users are facing a big challenge in choosing a proper one for their application scenarios. Traditionally, users apply a *data-driven* approach to tackle this challenge, where a set of candidate systems are run against various datasets. After execution, cross-validation and statistical analyses are used to help users select the proper system to use [10], [53], [54]. However, we argue that, besides the average performance of a clustering system across various datasets, users' expectations or requirements on the system with respect to the application scenario should also be taken into account.

Following this argument, METTLE does provide an intuitively appealing and systematic framework to aid selecting proper clustering systems, by enabling users to assess the appropriateness of these systems based on their own *specific* requirements and expectations. Below, we give more detailed explanation.

First, the framework of METTLE involves a concept of "adequacy criterion." For example, a list of generic MRs derived from users' expectations is used in METTLE as an adequacy criterion. Subject clustering systems are, then, assessed by validating the compliance with each generic MR. The results of assessment are used for selecting an appropriate system in accordance with users' own specific needs.

Test adequacy plays a crucial role in traditional software testing and validation. A lot of coverage criteria from different perspectives have already been proposed to measure test adequacy, such as statement coverage, branch coverage, and path coverage [55]. The necessity for evaluating test adequacy has been gradually accepted in machine learning testing [56]. Many researchers from the software engineering community have been working on proposing suitable criteria for evaluating the test adequacy for machine learning systems with a view to gaining confidence on the testing results [57], [58]. However, until now, there have been very few generally acceptable and systematic criteria for users to assess and validate machine learning (include clustering) systems in their own contexts.

Traditional clustering assessment methods can be regarded as a type of data-oriented adequacy measurement, by exploring the "adequacy" in the input space. However, with such data-oriented adequacy criterion, users cannot easily link the input to the appropriateness of a system with respect to their own expectations and requirements. In contrast, our METTLE provides a property-oriented adequacy criterion based on MRs, which can easily address the above problem in traditional methods. In fact, this property-oriented adequacy criterion makes the first step in the potential research direction pointed out by Chen *et al.* [15], where they argue that MT can allow the development of an MR-based metric to be used as a black-box test adequacy criterion. Assessing the compliance with MRs provides useful information about the quality and appropriateness of the relevant properties and functionalities of a clustering system in a particular application domain. Thus, such an MR-based criterion in METTLE can provide more confidence to users in making decision about which clustering system to select.

Table IV summarizes the performance in terms of the compliance with each generic MR of the six subject clustering systems with respect to the 11 generic MRs. As discussed in Section VI-C, this table can be used to help users make informed decision about which clustering system to select for use in a specific scenario. In addition to adopting some or all of the 11 generic MRs, more specific MRs can be defined by users to complement the generic ones (if users have expectations that do not correspond to any of these generic MRs). Note that users are not required to have substantial and sophisticated knowledge on the candidate clustering systems. This is because defining specific MRs is primarily based on users' domain knowledge of their applications. The adopted generic MRs, together with the additional, specific MRs defined by users, form a comprehensive checklist where MR compliance and the associated weighted scores can be used as a selection criterion.

In reality, a user may not consider all selected MRs (and their corresponding types of data transformation) to be equally important. In other words, some selected MRs are considered more preferable while the others are less preferable. Consider, for example, an e-commerce firm with a fast-growing number of online customers. Each of these customers has a registered account with the e-commerce firm. Consider further the following scenarios.

Scenario 1: The marketing department of the e-commerce firm often clusters its customers into different groups to facilitate new product recommendation to the targeted groups. In this case, the marketing director may be highly concerned with the impact of adding data samples (correspond to newly registered customer accounts) near a cluster's centroid or boundary on the clustering result generated by a clustering system.

Scenario 2: The business fraud department of the e-commerce firm may concern more on how a clustering system handles outliers because they may correspond to malicious hackers.

In view of the different levels of importance on the types of data transformation (and their corresponding MRs), the overall framework to support users to select clustering systems (in the context of METTLE) is given as follows.

1) Select generic MRs or define new MRs in accordance with the user's intuitive expectations and specific requirements related to their application domains.
2) Classify all the selected MRs into two categories: "must have" and "nice to have."
3) Use METTLE to validate all the candidate clustering systems against all the selected MRs by executing each method twice (first with the source dataset, then with the follow-up dataset).
4) Construct a summary table, which summarizes the violation patterns with respect to all the selected MRs.
5) For each "nice-to-have" selected MR, assign a weight w_1 (where $0.0 < w_1 < 1.0$), so that a higher value of w_1 means that the corresponding MR is relatively more preferable or important. Then, assign a weight w_2 (where $0.0 < w_2 < 1.0$) according to the type of violation patterns related to this MR, so that a higher value of w_2 indicates more severity for the corresponding violation pattern.
6) Ignore those clustering systems which show violations to any "must-have" MR.
7) For every remaining clustering system m_i (where $1 \leq i \leq k$; $k =$ total number of remaining systems), calculate its score S_{m_i} using the following formula:

$$S_{m_i} = (w_{11} \times w_{12} \times x_1) + (w_{12} \times w_{22} \times x_2) + \cdots + (w_{1n} \times w_{2n} \times x_n)$$

where $1 \leq j \leq n$; $n =$ total number of selected MRs; w_j = the weight assigned to MR_j; $x_j = 1$ if one or more violations to MR_j occur, and $x_j = 0$ if no violation to MR_j occurs.
8) The most appropriate system to select is the m_i with the *smallest* S_{m_i}.

By means of the above selection framework, users are able to devise their own quality assessment schemes for evaluating a set of candidate clustering systems in accordance with their own preferences. An overview of METTLE is provided in Fig. 20. To facilitate the application of METTLE, a tool has been developed to support the following tasks: executing datasets, checking violations to MRs, assigning weights to individual MRs, and computing the final weighted scores for individual clustering systems. This tool is made publicly accessible to those users who are interested in METTLE.[2]

As a reminder, the individual lists of selected MRs developed by different users in the same application domain can be shared, with a view to developing a more comprehensive and effective aggregated list of selection MRs. Furthermore, a repository (e.g., in [59]) can be created to store all the selected MRs and their corresponding validation results for some clustering systems. Via this repository, even inexperienced users without much knowledge about the execution behaviors of individual clustering systems (with respect to data transformation) can still effectively evaluate and, then, select their most preferred systems.

VIII. User Evaluation

In addition to our experiment described in Sections V and VI, we further evaluated the applicability of METTLE from the users' perspectives.

A. Evaluation Settings

In essence, our user evaluation aimed to investigate the applicability of METTLE for (end) users in their own application contexts. More specifically, we investigated whether or not users were able to use METTLE to validate the appropriateness of clustering systems with respect to the users' own requirements on data transformation. In this article, we also obtained users' comments and feedback about the viability and usefulness of METTLE in assessing and selecting appropriate clustering systems.

[2][Online]. Available: https://github.com/METTLE-dev/mettle

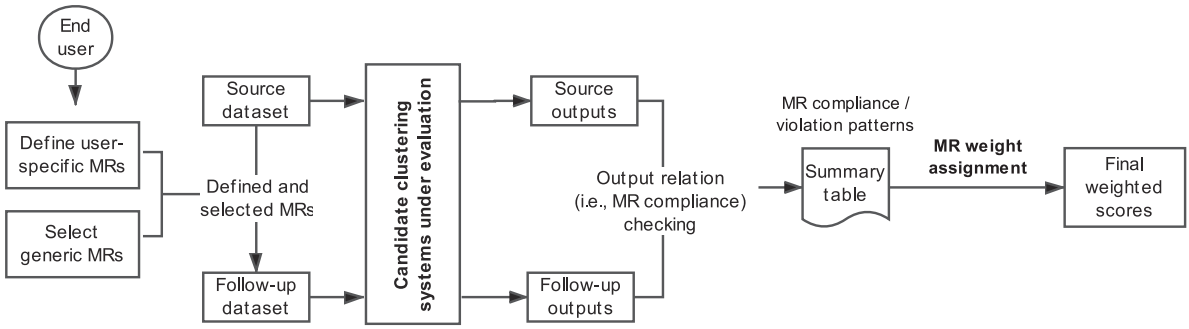

Fig. 20. Overview of METTLE.

The detailed evaluation settings and procedures are as follows.

1) We recruited a group of participants who are end users of clustering systems across a range of application domains. We collected the personal particulars of these participants, including, for example, their affiliated institutions, application domains, and experiences in using, validating, or selecting clustering systems.
2) We explained to the participants about the objectives of the evaluation, and offered a tutorial on MT and METTLE to help them familiar with the main concepts.
3) The participants were introduced (and illustrated with examples) with the 11 generic MRs discussed in Section IV-C. With reference to their application domains, the participants were asked to determine which of these 11 generic MRs were applicable for selecting appropriate clustering systems. Also, they were asked to assign a weight to each applicable MR, so that higher values indicating more preferable, relevant, or important MRs.
4) Besides selecting applicable, generic MRs, the participants were asked to define additional, more specific MRs for those "expected" data transformations, which were not covered by the generic MRs. The participants also assigned weights to these newly defined, more specific MRs in a similar way as for the applicable, generic MRs in step 3).
5) Using datasets relevant to the application domains in which the participants work, clustering results were generated for each clustering system in accordance with the selected, generic MRs [in step 3)] and the newly defined, user-specific MRs [in step 4)]. These results showed which MRs were violated by each clustering system.
6) Using the user-assigned weights for generic and user-specific MRs, for *each* clustering system, the final weighted score by *each* participant was computed. Thereafter, based on these final weighted scores by all participants, ranked lists of clustering systems for *each* application domain were generated (one ranked list per each relevant participant in each application domain).
7) We discussed the ranked lists generated in step 6) above with the participants, and asked the participants whether or not the results were consistent with their preferences on the clustering systems.

Note that each participant performed the MR selection [in step 3)] and MR definition [in step 4)] tasks on an individual basis, without discussion with other participants.

B. Evaluation Results

A total of 15 participants were recruited to participate in the evaluation study. All of them are end users of clustering systems in at least one of the following application domains: big data, computational mechanics, built environment, signal processing, remote sensing, Light Detection and Ranging (LIDAR) and surveying, and bioscience.

1) Participants' Experiences in Clustering Applications: All the 15 participants are experienced in using clustering applications. Among them, 11 participants have experiences in validating clustering systems using internal validation techniques, external validation techniques, or a cross-validation approach (see Section I). Furthermore, seven participants have experiences in comparing and selecting appropriate clustering systems in their own application domains.

2) MR Definition and Weight Assignment: The 15 participants selected those generic MRs that were applicable to their application domains, and also defined additional, user-specific MRs, which covered their own requirements. More specifically, each participant has selected 2–10 generic MRs. Furthermore, five participants have defined their own specific requirements on data transformation, which were not covered by any generic MR. Each of these five participants has defined a specific requirement; and three such requirements independently defined by three participants were the same. Hence, there were three distinct specific requirements, and each such requirement was subsequently converted into its corresponding user-specific MR. These three user-specific MRs can be classified into three types: 1) coordinate transformation (Geodetic coordinates to Cartesian coordinates), 2) feature reduction with principal component analysis (PCA), and 3) z-score normalization.

More details about these three user-specific MRs will be given in Section VIII-C2 later. Each participant also assigned a weight to every generic MR selected by and every user-specific MR defined by him/her.

3) Rankings of Clustering Systems: For every subject clustering system mentioned in Section V-B, the selected generic

TABLE VI
TOP$_k$ FREQUENCY FOR EACH APPLICATION DOMAIN

Application Domains	Clustering Systems	$k=1$	$k=2$	$k=3$
Big Data	EM	1.00	1.00	1.00
	AN	0.00	1.00	1.00
	XM	0.00	0.00	0.50
	DS	0.00	0.00	0.50
Computational Mechanics	EM	1.00	1.00	1.00
	AN	0.00	1.00	1.00
	DS	0.00	0.00	1.00
Built Environment	EM	0.00	1.00	1.00
	AN	0.00	0.00	1.00
	XM	1.00	1.00	1.00
Signal Processing	EM	0.00	1.00	1.00
	AN	0.00	0.00	1.00
	XM	1.00	1.00	1.00
Remote Sensing	EM	0.75	1.00	1.00
	AN	0.00	1.00	1.00
	XM	0.25	0.25	0.50
	DS	0.00	0.25	0.50
LIDAR and Surveying	EM	0.50	1.00	1.00
	AN	0.50	1.00	1.00
	XM	0.00	0.00	0.50
	DS	0.00	0.00	0.50
Bioscience	EM	0.67	1.00	1.00
	AN	0.33	0.67	1.00
	XM	0.00	0.00	0.33
	DS	0.00	0.33	0.67

MRs and defined user-specific MRs were used to validate this system in each of the seven application domains mentioned earlier in this section. In this process, for every clustering system in each application domain, the final weighted scores by the relevant participants were computed, from which ranked lists of clustering systems were generated.

In order to determine the overall performance of a clustering system x in each application domain D, we measured the frequency in which x was once evaluated by the relevant participants as one of the top k clustering systems in terms of its appropriateness with respect to D. For each D, let N be the total number of relevant participants, and $\boldsymbol{r}_i$ be the ranking result of the ith participant. Given a clustering system x, we use $\text{top}_k(x, \boldsymbol{r}_i)$ to denote whether or not x was one of the top k clustering systems according to $\boldsymbol{r}_i$. If yes, $\text{top}_k(x, \boldsymbol{r}) = 1$, and 0 otherwise. Then, the value of the metric top$_k$ frequency was computed as follows:

$$\text{top}_k \text{ frequency} = \frac{\sum_{i=1}^{N} \text{top}_k(x, \boldsymbol{r}_i)}{N}.$$

In this way, the above metric, thus, indicates the overall performance of x (with respect to D) as computed by METTLE, in accordance with the orders of preferences (expressed in terms of the weights) on the relevant MRs from the users' perspectives.

Table VI shows the values of top$_k$ frequency with $k = 1$, 2, or 3. In this table, those clustering systems with their top$_k$ frequency $= 0$ are not shown. There are three important observations from Table VI: 1) individual clustering systems have different performances across the seven application domains; 2) overall, EM, AN, XM, and DS were evaluated by the participants as the first, second, third, and fourth most preferred clustering systems, respectively, across the seven application domains; and 3) neither FF nor KM was evaluated as one of the top three most preferred clustering systems in any application domain.

4) End Users' Feedback: Participants were asked about whether the *recommended lists* of ranked clustering systems generated by METTLE were consistent with their own ranked lists (called the *user lists*). If the answer was "no," then participants were asked to compare both lists and to judge which list was more preferable.

With regard to the top-ranked clustering systems, most participants agreed that the recommended lists were generally consistent with the user lists. For instance, in remote sensing, EM and DS were most preferred by the participants, and both systems also appeared as two of the top-3 recommended solutions by METTLE. Interestingly, AN was among one of the top-3 solutions in the recommended lists, but this system was originally not ranked favorably by the participants. However, after the participants had checked the level of MR compliance of AN, they all said that they would consider selecting AN in their future clustering tasks. They also agreed that the information about MR compliance helps them better understand these clustering systems. This viewpoint has also been reported in a recent study [60] that MRs could enhance system understanding. Similar observations were also found in the other six application domains.

It would also be worth mentioning that, in built environment, KM did not appear in the recommended list by METTLE, but this system was commonly selected for use by the participants mainly due to its handy simulation feature. Although the participants often selected KM, they admitted that KM performed relatively poorer than other clustering systems in some application scenarios.

All in all, the participants were very pleased and enthusiastic about the applicability and effectiveness of METTLE.

C. Further Analysis and Discussion

Here, we provide further analysis and discussion on our user evaluation study.

1) Preferences on and Characteristics of MRs: In this article, the participants were asked to select generic MRs that were deemed applicable and to define additional, user-specific MRs for those requirements that were not covered by any generic MR. The participants, then, assigned different weights to these generic and user-specific MRs in accordance to the relevant importance of these MRs. In general, we observed that the weights assigned to user-specific MRs were larger than those assigned to generic MRs. We further noted the following two observations.

a) Participants From Different Application Domains Had Different Preferences on the Same MRs: Consider, for example, the big-data domain. Participants in this domain are often confronted with massive data and continuous information flows. A tough problem, thus encountered by these participants is the *curse of dimensionality* (which refers to unexpected or anomalous

phenomena that arise when analyzing and organizing data in high-dimensional spaces that do not exist in low-dimensional settings). One solution to alleviate this problem is to reduce the sample space via feature engineering. Participants from the big-data domain would, therefore, place more emphasis on those MRs related to manipulating data features (e.g., MR4.2—removing redundant features).

Now we turn to the LIDAR and surveying domain. Participants in this domain often apply clustering systems to classify the land-cover types of laser points. Since a laser point has three fixed features (latitude, longitude, and elevation), therefore those MRs related to manipulating data features (e.g., MR4.1—adding informative attributes and MR4.2—removing redundant attributes) are rarely applicable. Also, some participants in this domain expressed their expectation on data transformation: *original data points in Geodetic coordinates should always be transformed into Cartesian coordinates during the clustering analysis*. Thus, in this article, these participants selected those generic MRs related to coordinate transformation (e.g., MR5.1—rotating the coordinate system), and assigned higher weights to these MRs. Additionally, some of these participants defined a more domain-specific coordinate transformation property, corresponding to the user-specific MR1 (transforming geodetic coordinates to Cartesian coordinates) to be discussed in Section VIII-C2 below.

For the participants in the bioscience domain, they often apply clustering systems to categorize deoxyribonucleic acid sequences and cells, where selecting good features for their clustering models is a common and necessary task. Furthermore, the order of the sample data may shuffle during data preprocessing or transferring. Because of these reasons, in this article, participants from bioscience particularly selected those generic MRs related to feature (attribute) manipulation (e.g., MR4.2—removing redundant attributes) and data order manipulation (e.g., MR1.2—changing the object order but keeping the same set of starting centroids).

b) Common MRs Across Different Application Domains: For instance, when clustering environmental data (e.g., air temperature, humidity, and wind speed), participants from built environment often first perform data normalization (a participant in the study argued that: *different thermal sensations must be normalized into the same range*). Similarly, several participants from big data and computational mechanics mentioned that: *normalization is necessary to obtain nondimensionalized data*. Thus, MRs related to data normalization are more likely to be applicable in built environment, big data, and computational mechanics. These defined, user-specific MRs could be broadly classified into the generic MR-type "manipulating the coordinate system" as discussed in Section IV-C, but they were more domain-specific. For example, in our evaluation study, a more domain-specific MR (see user-specific MR3 "Z-score normalization" to be discussed in Section VIII-C2) was defined by some participants for the three application domains mentioned above.

2) Applicability of METTLE: We investigated and analyzed the applicability of METTLE for users to validate and assess clustering systems in their own contexts in the following two aspects: 1) MR definition (i.e., whether the participants were able to define their own requirements on data transformation, eventually leading to the definition of user-specific MRs), and 2) MR implementation (i.e., whether participants were able to implement programs or use some existing tools to perform the necessary data transformations in accordance with the selected or defined MRs).

a) MR Definition: In the article, all the participants were asked to define their own data transformation requirements, which were not covered by the 11 generic MRs by using the following two-step approach.

1) Given a dataset for clustering, identify a data transformation that is not covered by any generic MR.
2) For every data transformation identified in 1), specify the expected relation between the clustering results generated by the original dataset and the follow-up dataset.

Obviously, the outcome of 2) is a user-specific MR. This article showed that the above approach worked well even for participants who were inexperienced on MT.

As mentioned in Section VIII-B, five participants have defined their own specific requirements, resulting in three distinct user-specific MRs. These three MRs are discussed below.

1) For spaceborne photon-counting LIDAR, a laser point is represented in three dimensions: latitude, longitude, and elevation in the Geodetic coordinate system. A participant from LIDAR and surveying defined the following user-specific MR.

   User-Specific MR1—Transforming Geodetic Coordinates to Cartesian Coordinates: When clustering the land-cover types of laser points, if the Geodetic coordinates of these points are transformed into their corresponding Cartesian coordinates, the clustering result is expected to remain unchanged.

   Although MR1 is domain-specific and was defined by a participant, it can be broadly classified into the generic MR-type "manipulating the coordinate system" (but MR1 is different from generic MR5.1 and MR5.2) as discussed in Section IV-C.
2) In bioscience (more specifically, from neuroscience), users often apply clustering systems on the white-matter connection strength of a brain to learn different human's cognitive functions. Thus, a participant from this domain defined the following user-specific MR.

   User-Specific MR2—Feature Reduction With PCA: After feature reduction, the data dimensionality will be reduced and the cluster boundary will become clearer.

   Similar to MR1 above, although MR2 is domain-specific, it falls into the generic MR-type "manipulating attributes" (but MR2 is different from generic MR4.1 and MR4.2) as discussed in Section IV-C.
3) In the big-data, built science, and computational mechanics domains, data normalization is generally a common practice, which allows users to generate nondimensionalized data. Accordingly, three participants (one from big data, one from built environment, and one from computational mechanics) independently defined the following user-specific MR.

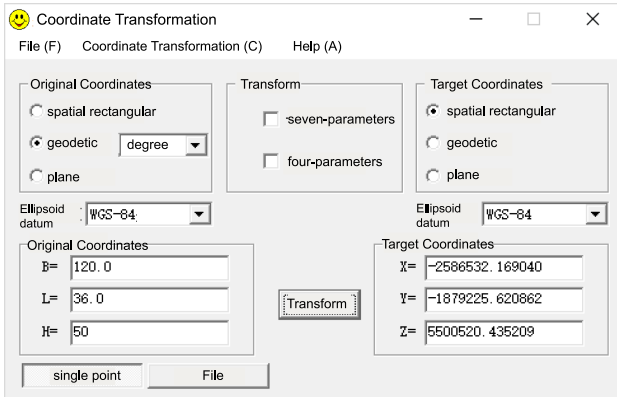

Fig. 21. Coordinate transformation tool.

> *User-Specific MR3—Z-Score Normalization:* After performing normalization to generate the follow-up dataset from the original dataset, the clustering result based on the original dataset is expected to be the same as the clustering result based on the follow-up dataset.

MR3 falls into the generic MR-type "manipulating the coordinate system" (but MR3 is different from generic MR5.1 and MR5.2) as discussed in Section IV-C.

b) MR Implementation: In our user study, participants have reported that various tools are available for transforming the original dataset into the corresponding follow-up dataset in accordance with a particular MR. For example, a participant from the LIDAR and surveying domain mentioned that: *we have some ready-to-use code snippets, and even tools to support complex transformations*. Fig. 21 shows a tool that supports coordinate transformation in the LIDAR and surveying domain. Once the two datasets are ready, participants can execute a clustering system twice: first with the original dataset, next with the follow-up dataset. These two executions will, then, generate two clustering results, from which participants can compare the results to check whether the relevant MR is violated.

The above discussion has demonstrated the applicability of METTLE from the user's perspective. Before the user evaluation study, none of the 15 participants had knowledge about MT and METTLE. Although all the participants had just received a short tutorial on MT and METTLE, this article showed that these participants were able to: 1) understand and select those generic MRs that covered their requirements on data transformation and assigned weights to these generic MRs; 2) define user-specific MRs for those requirements that were not covered by the generic MRs, and assign weights to these user-specific MRs; 3) use existing available tools to transform the original dataset into its corresponding follow-up dataset so that both datasets could be executed by the clustering systems to generate the clustering results; 4) compare the two clustering results generated from the source dataset and the follow-up dataset to see whether the relevant MR has been violated, and compute the final weighted scores for the candidate clustering systems with a view to selecting the most appropriate one by using METTLE.

3) Usefulness of METTLE: In the user evaluation study, the participants were asked about their experiences in validating and selecting clustering systems. On one hand, most participants acknowledged the importance and necessity of validating various clustering systems; on the other hand, they reported that the validation task is not trivial and, in fact, is fairly challenging. Below are some of these participants' comments.

> *Clustering as unsupervised learning has many ambiguities; it is not easy to evaluate whether the clustering results are good or bad.*
> *I could not acquire the expected outputs.*
> *It needs a lot of experiments to evaluate the performances [of the candidate clustering systems].*

At the end of the evaluation study, we asked the participants about their overall feelings of the usefulness of METTLE. All the participants unanimously showed positive feedbacks on METTLE. Some of these feedbacks are shown below.

> *I could gain a better understanding of the clustering systems from METTLE.*
> *It becomes easier for me to know which [clustering] algorithm fits better to my applications.*
> *I do not need substantial technical knowledge for using METTLE; it is convenient and easy to use.*
> *Using METTLE costs less effort than conducting preliminary experiments for choosing an appropriate clustering system.*

The above discussion has demonstrated the usefulness of METTLE from the user's perspective.

IX. Threats to Validity

In this section, we discuss some potential factors that might affect the validity of our experiment and user evaluation study.

A. Internal Validity

A main internal threat to our experiment is the randomness of clustering systems. Some of these systems will randomly select an object from the dataset in their initialization. This may lead to result variations across multiple system executions. To alleviate this threat, we fixed the random seed for the relevant systems in our experiment, so that the clustering results are reproducible in each execution run.

Another internal threat to our experiment is related to parameter configuration. Different input parameters would lead to totally different clustering results. Thus, the impact of parameters on clustering validity is definitely a further research topic for clustering assessment and validation. For example, DS has two critical parameters: the minimal number of points minPts within a specific distance eps. The clustering result generated by DS is largely affected by these two parameters. In this article, we do not attempt to evaluate the impact of different parameters on the clustering result. Thus, these parameters were not treated as independent variables and, hence, were fixed during our experiment.

B. Construct Validity

In our user evaluation study, the participants were individually provided with a short, half-an-hour tutorial on MT and METTLE. A threat might exist that the participants were unable to grasp the core concepts such as MR. However, even with this potential

threat, the experimental and evaluation results were very encouraging and promising, demonstrating the effectiveness, practicality, and usefulness of METTLE from the user's perspective.

C. External Validity

In METTLE, we leveraged the concept of MT and developed a list of generic MRs for validation. Because these generic MRs do not cover all possible properties of clustering systems, this issue is, therefore, a potential threat to the external validity of this article. However, as a novel assessment and validation framework based on the user's perspective, METTLE allows users to specify their expected characteristics of a clustering system in their own contexts. In METTLE, as confirmed by our evaluation study, users could simply adopt some of all of the 11 generic MRs we developed, and, then, supplemented by more specific, user-defined MRs according to their own application scenarios. Thus, this threat is considered low.

As an application of MT, METTLE also has limitations in some areas of cluster analysis, for example, identifying the optimal number of clusters. MT was proposed to alleviate (not to completely solve) the oracle problem in software testing. Also, by its very nature, an MR can only reveal the absence of an expected characterization from the system, rather than computing the correctness of individual outputs.

Another external threat is the generality of our approach. In this regard, it is well known that in the field of cluster validation, there does not exist a single validation approach, which can effectively handle all dataset types [61]. METTLE is no exception. Our experiment only involved those datasets with well-formed clusters so that all the six subject clustering systems could properly handle these clusters. A similar approach to generating synthetic datasets for experiments has also been adopted in some other studies (e.g., [62]). Although the datasets used for assessment may vary case by case, the high-level properties of clustering systems to be assessed and evaluated by METTLE are rather general. Thus, we argue that the effect of different datasets on the effectiveness of METTLE should not be large. Our argument was supported by the results of the user evaluation study that participants from different application domains were able to use METTLE to validate and assess candidate clustering systems with domain-specific datasets.

Our user evaluation involved only 15 participants. It would certainly be better if more participants were involved in the study. However, it was not easy to find a large group of participants who were willing to participate in our study. Although the number of participants was not large, they are experienced and knowledgeable users of clustering systems from seven different application domains. Thus, to some extent, these participants could be considered as a group of "representative" users of clustering systems.

X. Related Work

MT has been successfully applied in many applications since its introduction by Chen *et al.* [21]. We refer the readers to recent surveys on MT [15], [63] to gain further insight into this technique. In this section, we highlight some recent work on MT by both academia and industry researchers.

Zhou *et al.* [26] proposed a user-oriented testing approach for the quality assessment of major online search engines (including, for example, Google and Bing) using the concept of MT. Their empirical results not only guide developers to identify the weaknesses of these search engines, but also help users choose a proper online search engine in a specific scenario. Segura *et al.* [64] applied MT to web application programming interfaces (APIs) for automatic fault detection. They first constructed MRs with an output-driven approach, and, then, applied their method to APIs of Spotify and YouTube. This application successfully detected 11 real-life problems, indicating the effectiveness of MT. A recent work [65] has been reported, which is related to using MT for software verification of machine-learning-based image classifiers. The effectiveness of MRs was tested by mutation testing, where 71% implementation faults were successfully caught.

Adding to the successful applications of MT to quality assessment as well as software verification and validation, MT has also been applied to detecting performance bugs [66]. In this work, a set of performance MRs was defined for the automatic analysis of feature models. A proof-of-concept experiment was conducted to confirm the feasibility of using a metamorphic approach to detecting performance faults.

In recent years, we have witnessed the advances in deep learning. Applying MT to AI-driven systems has grown rapidly. In [28], MT was used to validate the classification accuracy of deep learning frameworks. Also, DeepTest, a testing tool for Deep-Neural-Network-driven autonomous vehicles, was developed to leverage MRs to create a test oracle [30]. DeepTest automatically generates synthetic test cases for different real-world conditions, and is able to detect thousands of erroneous behaviors in autonomous driving systems. Furthermore, a framework called DeepRoad [31] was proposed for testing autonomous driving system, with a view to detecting inconsistent behaviors across various synthesized driving scenes based on MRs.

More recently, an internationally renowned IT consultancy and service firm, Accenture, has applied MT to test machine learning systems, providing a new vision for quality engineering [67]. In addition, GraphicsFuzz, a commercial spin-off firm from the Department of Computing, Imperial College London, has pioneered the combination of fuzzing and MT for testing graphics drivers [68]. GraphicsFuzz toolset has been successful at exploring defects in a large number of graphics driver across different platforms, for example, an Shield TV box with an NVIDIA GPU and Samsung Galaxy S9 with an ARM GPU. GraphicsFuzz was later acquired by Google in August 2018 [69].

XI. Conclusion

In this article, we proposed an MT-based approach (METTLE) to assessing and validating clustering systems by considering the various dynamic data perspectives for different application scenarios. We defined 11 generic MRs for six common types of data transformation. We used these generic MRs, together with six subject clustering systems, to conduct an experiment

for verifying the viability and effectiveness of METTLE. Our experiment demonstrated that METTLE is a vivid, flexible, and practical approach toward validating and assessing clustering systems. Furthermore, we conducted a user evaluation study on METTLE from the user's perspective. All the participants involved in the study unanimously showed positive feedbacks on METTLE.

In general, METTLE has the following merits with respect to validation and assessment.

1) *Validation*
   a) It is generic and can be easily applied to any clustering systems.
   b) It provides an elegant and tailor-made mechanism for end users to define their specific expectations and requirements (in terms of MRs) when validating clustering systems.
   c) It is further supported by a set of 11 generic MRs, which can be mostly applied to various clustering scenarios.
2) *Assessment*
   a) It provides an innovative approach to unveil the characteristics of unsupervised machine learning systems.
   b) It helps categorize clustering systems in terms of their strengths and weaknesses with respect to a set of MRs (corresponding to different types of data transformation). This is particularly helpful for those end users who are not knowledgeable about the logic and mechanisms of clustering systems.
   c) It allows end users to devise their own quality assessment schemes for evaluating a set of candidate clustering systems (with respect to the user-defined MRs and their corresponding weights).
   d) It demonstrates a systematic and practical framework for end users to assess and select appropriate clustering system for use.

The promising and encouraging work described in this article can be extended into three aspects. First, it would be worthwhile to conduct another experiment involving high-dimensional data samples (the experiment described in this article only involved datasets in 2-D space for easy visualization of the clustering results). Second, it would be fruitful to investigate the issue of how to define good and representative MRs (in addition to the 11 generic ones) that are applicable to a wide range of application scenarios. Third, the correlation between a violation to an MR and a particular error pattern certainly represents an interesting research topic that warrants further investigation.

REFERENCES

[1] G. Punj and D. W. Stewart, "Cluster analysis in marketing research: Review and suggestions for application," *J. Marketing Res.*, vol. 20, no. 2, pp. 134–148, 1983.

[2] K. Chaudhry, J. Yadav, and B. Mallick, "A review of fraud detection techniques: Credit card," *Int. J. Comput. Appl.*, vol. 45, no. 1, pp. 39–44, 2012.

[3] D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 11, pp. 1370–1386, Nov. 2004.

[4] M. Steinbach, P.-N. Tan, V. Kumar, S. Klooster, and C. Potter, "Discovery of climate indices using clustering," in *Proc. 9th ACM Int. Conf. Knowl. Discovery Data Mining*, 2003, pp. 446–455.

[5] A. Hotho, S. Staab, and G. Stumme, "Ontologies improve text document clustering," in *Proc. 3rd IEEE Int. Conf. Data Mining*, 2003, pp. 541–544.

[6] W. Liu, S. Liu, Q. Gu, J. Chen, X. Chen, and D. Chen, "Empirical studies of a two-stage data preprocessing approach for software fault prediction," *IEEE Trans. Rel.*, vol. 65, no. 1, pp. 38–53, Mar. 2016.

[7] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, and A. Y. Zomaya, "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 3, pp. 267–279, Sep. 2014.

[8] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn.*, 2016, pp. 478–487.

[9] A. Saxena *et al.*, "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, no. C, pp. 664–681, 2017.

[10] R. S. Olson, W. L. Cava, Z. Mustahsan, A. Varik, and J. H. Moore, "Data-driven advice for applying machine learning to bioinformatics problems," in *Proc. Pacific Symp. Biocomput.*, 2018, pp. 192–203.

[11] F. Di Maio, P. Secchi, S. Vantini, and E. Xio, "Fuzzy c-means clustering of signal functional principal components for post-processing dynamic scenarios of a nuclear power plant digital instrumentation and control system," *IEEE Trans. Rel.*, vol. 60, no. 2, pp. 415–425, Jun. 2011.

[12] A. Banerjee and J. Langford, "An objective evaluation criterion for clustering," in *Proc. 10th Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 515–520.

[13] A. Williams, "What is clustering and why is it hard?" Sep. 11, 2015. [Online]. Available: http://alexhwilliams.info/itsneuronalblog/2015/09/11/clustering1/. Accessed on: Dec. 29, 2018.

[14] U. von Luxburg, R. C. Williamson, and I. Guyon, "Clustering: Science or art?" in *Proc. Int. Conf. Unsupervised Transfer Learn. Workshop*, 2011, pp. 65–79.

[15] T. Y. Chen *et al.*, "Metamorphic testing: A review of challenges and opportunities," *ACM Comput. Surveys*, vol. 51, no. 1, pp. 4:1–4:27, 2018.

[16] E. Rendón, I. Abundez, A. Arizmendi, and E. M. Quiroz, "Internal versus external cluster validation indexes," *Int. J. Comput. Commun.*, vol. 5, no. 1, pp. 27–34, 2011.

[17] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Inf. Retrieval*, vol. 12, no. 4, pp. 461–486, 2009.

[18] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *Proc. 10th Int. Conf. Data Mining*, 2010, pp. 911–916.

[19] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona, "An extensive comparative study of cluster validity indices," *Pattern Recognit.*, vol. 46, no. 1, pp. 243–256, 2013.

[20] C. Hennig, M. Meila, F. Murtagh, and R. Rocci, Eds., *Handbook of Cluster Analysis*. Boca Raton, FL, USA: CRC Press, 2015.

[21] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: A new approach for generating next test cases," Tech. Rep. HKUST-CS98-01, Dept. Comput. Sci., Hong Kong Univ. Sci. Technol., Clear Water Bay, Hong Kong, 1998.

[22] T. Y. Chen, P.-L. Poon, and X. Xie, "METRIC: METamorphic relation identification based on the category-choice framework," *J. Syst. Softw.*, vol. 116, pp. 177–190, 2016.

[23] C.-A. Sun, A. Fu, P.-L. Poon, X. Xie, H. Liu, and T. Y. Chen, "METRIC$^+$: A metamorphic relation identification technique based on input plus output domains," *IEEE Trans. Softw. Eng.*, to be published, doi: 10.1109/TSE.2019.2934848.

[24] C. Murphy, G. E. Kaiser, L. Hu, and L. Wu, "Properties of machine learning applications for use in metamorphic testing," in *Proc. 20th Int. Conf. Softw. Eng. Knowl. Eng.*, 2008, pp. 867–872.

[25] X. Xie, J. W. 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, 2011.

[26] Z. Q. Zhou, S. Xiang, and T. Y. Chen, "Metamorphic testing for software quality assessment: A study of search engines," *IEEE Trans. Softw. Eng.*, vol. 42, no. 3, pp. 264–284, Mar. 2016.

[27] M. Olsen and M. Raunak, "Increasing validity of simulation models through metamorphic testing," *IEEE Trans. Rel.*, vol. 68, no. 1, pp. 91–108, Mar. 2019.

[28] J. Ding, X. Kang, and X. Hu, "Validating a deep learning framework by metamorphic testing," in *Proc. 2nd Int. Workshop Metamorphic Testing*, 2017, pp. 28–34.

[29] Z. Q. Zhou and L. Sun, "Metamorphic testing of driverless cars," *Commun. ACM*, vol. 62, no. 2, pp. 61–69, 2019.

[30] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proc. 40th Int. Conf. Softw. Eng.*, 2018, pp. 303–314.

[31] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems," in *Proc. 33rd ACM/IEEE Int. Conf. Automated Softw. Eng.*, 2018, pp. 132–142.

[32] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques* The Morgan Kaufmann Series in Data Management Systems, 3rd ed. Amsterdam, The Netherlands: Elsevier, 2011.

[33] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in *Proc. KDD Workshop Text Mining*, 2000, pp. 109–110.

[34] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: an Introduction to Cluster Analysis* Wiley Series in Probability and Statistics. Hoboken, NJ, USA: Wiley, 2009.

[35] T. Lange, V. Roth, M. L. Braun, and J. M. Buhmann, "Stability-based validation of clustering solutions," *Neural Comput.*, vol. 16, no. 6, pp. 1299–1323, 2004.

[36] C. Hennig, "Cluster-wise assessment of cluster stability," *Comput. Statist. Data Anal.*, vol. 52, no. 1, pp. 258–271, 2007.

[37] U. Möller, "Resampling methods for unsupervised learning from sample data," in *Machine Learning*, A. Mellouk and A. Chebira, Eds., London, U.K.: IntechOpen, 2009, pp. 289–304.

[38] I. M. G. Dresen, T. Boes, J. Huesing, M. Neuhaeuser, and K.-H. Joeckel, "New resampling method for evaluating stability of clusters," *BMC Bioinformatics*, vol. 9, no. 1, 2008, Art. no. 42.

[39] A. K. Jain and J. Moreau, "Bootstrap technique in cluster analysis," *Pattern Recognit.*, vol. 20, no. 5, pp. 547–568, 1987.

[40] J. Umbrich, M. Hausenblas, A. Hogan, A. Polleres, and S. Decker, "Towards dataset dynamics: Change frequency of linked open data sources," in *Proceedings of 3rd International Workshop on Linked Data on the Web*. [Online]. Available: https://aran.library.nuigalway.ie/bitstream/handle/10379/1120/dynamics_ldow2010.pdf?sequence=1&isAllowed=y. Accessed on: Dec. 29, 2018.

[41] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *J. Roy. Statist. Soc. Ser. C (Appl. Statist.)*, vol. 28, no. 1, pp. 100–108, 1979.

[42] J. Segal, "Two principles of end-user software engineering research," *ACM SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–5, 2005.

[43] M. Burnett, "What is end-user software engineering and why does it matter?" in *Proc. Int. Symp. End User Develop.*, 2009, pp. 15–28.

[44] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln, *Experimentation in Software Engineering*. Heidelberg, Germany: Springer, 2012.

[45] M. G. Omran, A. P. Engelbrecht, and A. Salman, "An overview of clustering methods," *Intell. Data Anal.*, vol. 11, no. 6, pp. 583–605, 2007.

[46] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 226–231.

[47] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. Amsterdam, The Netherlands: Elsevier, 2017.

[48] D. Pelleg and A. W. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *Proc. 17th Int. Conf. Mach. Learn.*, 2000, vol. 1, pp. 727–734.

[49] S. Konishi and G. Kitagawa, *Information Criteria and Statistical Modeling* Springer Series in Statistics. New York, NY, USA: Springer, 2008.

[50] R. A. Redner and H. F. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM Rev.*, vol. 26, no. 2, pp. 195–239, 1984.

[51] S. Dasgupta and P. M. Long, "Performance guarantees for hierarchical clustering," *J. Comput. Syst. Sci.*, vol. 70, no. 4, pp. 555–569, 2005.

[52] F. Pedregosa *et al.*, "Scikit-Learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[53] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 161–168.

[54] M. Fernandez-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3133–3181, 2014.

[55] H. Zhu, P. A. V. Hall, and J. H. R. May, "Software unit test coverage and adequacy," *ACM Comput. Surv.*, vol. 29, no. 4, pp. 366–427, 1997.

[56] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," Jun. 2019. [Online]. Available: https://arxiv.org/abs/1906.10742

[57] L. Ma *et al.*, "DeepGauge: Multi-granularity testing criteria for deep learning systems," in *Proc. 33rd ACM/IEEE Int. Conf. Autom. Softw. Eng.*, 2018, pp. 120–131.

[58] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," in *Proc. 41st Int. Conf. Softw. Eng.*, 2019, pp. 1039–1049.

[59] X. Xie, J. Li, C. Wang, and T. Y. Chen, "Looking for an MR? Try METWiki today," in *Proc. 1st Int. Workshop Metamorphic Testing*, 2016, pp. 1–4.

[60] Z. Q. Zhou, L. Sun, T. Y. Chen, and D. Towey, "Metamorphic relations for enhancing system understanding and use," *IEEE Trans. Softw. Eng.*, to be published, doi: 10.1109/TSE.2018.2876433.

[61] N. R. Pal and J. C. Bezdek, "On cluster validity for the fuzzy c-means model," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 1, pp. 152–153, Feb. 1997.

[62] Z. Huang, D. W. Cheung, and M. K. Ng, "An empirical study on the visual cluster validation method with Fastmap," in *Proc. 7th Int. Conf. Database Syst. Adv. Appl.*, 2001, pp. 84–91.

[63] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés, "A survey on metamorphic testing," *IEEE Trans. Softw. Eng.*, vol. 42, no. 9, pp. 805–824, Sep. 2016.

[64] S. Segura, J. A. Parejo, J. Troya, and A. Ruiz-Cortés, "Metamorphic testing of RESTful web APIs," *IEEE Trans. Softw. Eng.*, vol. 44, no. 11, pp. 1083–1099, Nov. 2018.

[65] A. Dwarakanath *et al.*, "Identifying implementation bugs in machine learning based image classifiers using metamorphic testing," in *Proc. 27th ACM SIGSOFT Int. Symp. Softw. Testing Anal.*, 2018, pp. 118–128.

[66] S. Segura, J. Troya, A. Durán, and A. Ruiz-Cortés, "Performance metamorphic testing: A proof of concept," *Inf. Softw. Technol.*, vol. 98, pp. 1–4, 2018.

[67] Accenture, "Quality engineering in the new: A vision and R&D update from Accenture labs and Accenture testing services," Jun. 27, 2018. [Online]. Available: https://www.accenture.com/t20180627T065422Z__w__/cz-en/_acnmedia/PDF-81/Accenture-Quality-Engineering-POV.pdf. Accessed on: Dec. 30, 2018.

[68] Imperial Innovations, "GraphicsFuzz launches testing solution for graphics drivers," *Imperial Innovations*, Apr. 26, 2018. [Online]. Available: https://www.imperialinnovations.co.uk/news-events/news/2018/apr/26/graphicsfuzz-launches-testing-solution-graphics-dr/. Accessed on: Dec. 31, 2018.

[69] F. Lardinois, "Google acquires GraphicsFuzz, a service that tests android graphics drivers," *techcrunch.com*, Aug. 6, 2018. [Online]. Available: https://techcrunch.com/2018/08/06/google-acquires-graphicsfuzz-a-service-that-tests-android-graphics-drivers/?via=indexdotco. Accessed on: Dec. 31, 2018.