MT4UML: Metamorphic Testing for Unsupervised Machine Learning

Faqeer ur Rehman
Gianforte School of Computing
Montana State University
Bozeman, MT, USA
faqeer.rehman@student.montana.edu

Dr. Clemente Izurieta
Gianforte School of Computing
Idaho National Laboratories
Montana State University
Bozeman, MT, USA
clemente.izurieta@montana.edu

***Abstract*—One of the advantages of using unsupervised machine learning algorithms is that they don't need labeled data; thus, ultimately saving higher labeling costs for an organization. However, the computational complexity and large input space put these algorithms into the category of non-testable programs, which also suffer from the oracle problem. One popular testing approach, borrowed from the Software Engineering (SE) domain is the Metamorphic Testing (MT) technique that has been proven to be an effective approach in alleviating the oracle problem in testing such non-testable programs. We take advantage of this MT approach to make some insightful contributions that include: i) proposing a broader set of 22 Metamorphic Relations (MRs) for assessing the behavior of the K-means clustering algorithm (a prototype-based approach) and the Agglomerative clustering algorithm (a hierarchy-based approach), provided by the leading scikit-learn Python library, ii) providing a detailed analysis/reasoning to show how the proposed MRs can be used to target both the verification and validation aspects of testing the clustering algorithms under investigation, and iii) showing that verification of MRs using multiple criteria is more beneficial than relying on using just a single criterion (i.e., clusters assigned). We further applied the proposed approach to test an open source customer segmentation application and the results obtained show that, i) 10 MRs have been violated for both the K-means and Agglomerative clustering algorithms, and ii) in comparison to K-means, the Agglomerative clustering algorithm is highly susceptible to small changes in inputs and may not offer a better alternative to scenarios captured by the violated MRs.**

***Index Terms*—Software Engineering, Machine Learning, Clustering, Unsupervised learning, Metamorphic Testing, Metamorphic Relations, Verification, Validation, Oracle problem**

I. Introduction

Machine Learning (ML) can broadly be classified into supervised and unsupervised machine learning algorithms. Supervised machine learning learns patterns from the labeled data, whereas, there is no class label available for the data points when the problem under investigation falls under the unsupervised machine learning category. Some of the widely used unsupervised machine learning algorithms include K-means clustering (a prototype-based approach) and Agglomerative clustering (a hierarchy-based approach). These algorithms can be used to address a wide range of real world problems i.e., customer segmentation, document clustering, clustering DNA patterns, recommendation systems, and anomaly detection. It is thus important that such applications are tested properly to ensure their quality before moving them to production environments. Further, one should also be aware of the possible changes in clustering results when the data itself undergoes changes in the future. However, similar to supervised ML algorithms, one of the challenges faced in testing unsupervised ML algorithms is their complexity and their exposure to the oracle problem. The oracle is a mechanism that a software tester uses to verify the output of the program under test. When the oracle is not available (or is available but infeasible to apply) we call a program suffering from the oracle problem.

Software Engineering for Machine Learning (SE4ML) is an emerging research area that focuses on applying SE best practices and methods for better development, testing, operation, and maintenance of ML-based systems [1] [2] [3] [4]. Our focus in this work is on the testing aspect of unsupervised ML algorithms and how a traditional software testing approach i.e., Metamorphic Testing (MT) can be utilized to perform better quality assurance from both the verification and validation perspective.

MT is considered an effective testing strategy in alleviating the oracle problem in testing both type of supervised and unsupervised ML algorithms. In MT, Metamorphic Relations (MRs) are proposed to test the program under test. Each MR is composed of a source test case and a follow-up test case. An MR is said to be violated if the result obtained for the source test case is different from the follow-up test case. The MRs can be used to target either i) the necessary characteristics (related to implementation), or ii) expected characteristics (related to user expectations) of the program under test. MT is different from using the classical evaluation methods i.e., residual sum-of-squares, silhouette coefficient, Davies–Bouldin index, etc. (frequently used to evaluate the clustering results of different algorithms) in the sense that MT is a testing technique, whereas, these evaluation methods aim to identify the algorithm more suitable for the problem under investigation. It is important to note that much of the research work has focused on utilizing the power of MT for testing supervised ML algorithms [5] [6] [7] [8] [9] but much less work has been done in using MT for testing unsupervised algorithms [10] [11]. Prior work (related to testing unsupervised ML

2022 9th Swiss Conference on Data Science (SDS) | 978-1-6654-6847-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/SDS54800.2022.00012

978-1-6654-6847-3/22/$31.00 ©2022 IEEE
DOI 10.1109/SDS54800.2022.00012

algorithms) uses MT to test clustering algorithms (provided by the WEKA tool [13]) only from the validation perspective. This motivates us to take MT one step further and utilize its power to test some widely used clustering algorithms (i.e., K-means clustering and Agglomerative clustering), provided by a popular and widely used Python library i.e., scikit-learn [12] from both the verification and validation perspective.

The following presents the main contributions made in this paper:

- We propose an MT based approach for verification and validation of two popular clustering algorithms, provided by the leading Python library known as scikit-learn. These includes K-means (a prototype-based approach) and Agglomerative clustering (a hierarchy-based approach) algorithms.
- We propose 22 MRs to assess the behaviour (from both the user's and developer's/implementation perspective) of the clustering algorithms under test.
- The proposed MRs are further analyzed, necessary reasoning is provided, and MRs are then categorized to show whether each of those MRs targets the verification or the validation aspect of testing the two algorithms under investigation.
- The effectiveness of the proposed approach is demonstrated by applying it to testing an open source customer segmentation application [1]. The results show that among the proposed MRs, 10 MRs are violated for both the K-means and the Agglomerative clustering algorithm.

The rest of the paper is organized as follows. Section II presents related work, whereas, Section III presents the Goal Question Metric (GQM) approach to frame the research work. Next, the proposed approach and the contributions made are presented in Section IV. Section V presents the results obtained by showing the effectiveness of the proposed approach. Lastly, in Section VI, conclusions are made along with potential future work.

II. Related Work

The MT technique has been shown to be an effective approach in alleviating the oracle problem in computationally complex machine learning based classifiers [5] [6] [7] [8] [9]. To the best of our knowledge we are able to find only two research papers in which MT has been utilized to test unsupervised clustering algorithms [10] [11], which is equally the motivator for this work and for making some beneficial research contributions. Yang et al. [10] proposed 7 MRs to test the K-means algorithm (in WEKA tool) that target the algorithm's correctness from a user perspective (validation) to check whether the user expectations from the algorithm are satisfied or not. Their results show that two of the MRs are violated but this does not necessarily mean that there is some implementation defect in the algorithm under test. Xie et al. [11] proposed 11 generic MRs that assess and validate the characteristics of different clustering algorithms from a user perspective. The authors conducted an experiment to test 6 clustering algorithms (provided by WEKA) and compared them using the proposed MRs. This research helps end-users (non-technical users) coming from diverse fields such as bioinformatics, finance, and electrical engineering to choose a specific type of algorithm from a large set of available algorithms that can best fit their needs. However, the following are the limitations we have found in their work:

- The proposed approaches only serve validation purposes, and only check whether the algorithms under test meet the user expectations.
- The proposed MRs only target the algorithms provided by the WEKA tool. It is equally worth exploring to test the behavior of other notable clustering algorithms provided by widely used Python libraries i.e., Scikit-learn. It will not only help the end-users in choosing the most suitable clustering algorithm but it will also help in selecting the right ML library for their problem under investigation.
- The proposed approaches use synthetic 2D data (i.e., not real data). It is worth exploring whether the proposed MRs are effective in testing the models that use multi-dimensional real-world data sets as well.

III. GQM

The GQM (Goal Question Metric) is a goal oriented approach [15] that we have used to frame the research work. As shown in Fig. 1, in the GQM approach, a set of **goal(s)** are identified, each of them is further refined using the **questions** to address the corresponding research goal, and then the metrics are outlined to answer the research questions in a quantifiable manner.

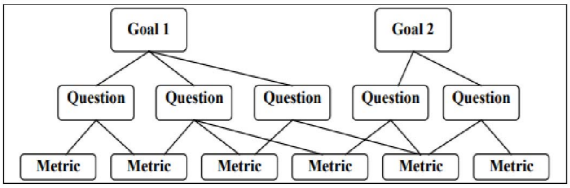

Fig. 1. GQM Approach [15]

Research Goal (RG): To investigate the MT technique for testing unsupervised algorithms *for the purpose of* improving their quality *from the perspective of* both the end user and a developer in *the context of* testing K-means and Agglomerative clustering algorithms.

Research Question 1 (RQ1): How effective are the proposed MRs in testing the clustering algorithms under test?

Research Question 2 (RQ2): Which algorithm is more stable for performing clustering-related tasks?

Research Metrics (RM): The following metrics are used to answer the above raised research questions.

RM1: *Number of violated MRs* - A count of the number of MRs that are violated by the algorithms under test.

[1]https://github.com/matifkhattak/MT4UML

RM2: *Violation Rate* - Percentage of instances for which the programs show inconsistent behaviour.

IV. Our Approach

Our approach for testing the K-means and Agglomerative clustering algorithms is based on the Metamorphic Testing (MT) technique [14]. In **K-means algorithm**, the following equation is used for calculation of new centroids [11], in which $c_i^{(t+1)}$ represents the i^{th} new centroid found, and x_j represents the j^{th} instance (where $j = 1, 2, ..., n$) belonging to the cluster C_i.

$$c_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{j=1}^{n} x_j \tag{1}$$

In **Agglomerative clustering**, the following equation represents the average linkage method used to merge two similar clusters [11]:

$$d(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x_r \in C_i} \sum_{x_s \in C_j} d(x_r, x_s) \tag{2}$$

where, $d(C_i, C_j)$ represents the distance between cluster C_i and cluster C_j, x_r represents the r^{th} data point (where r = 1,2,...,n) belonging to the cluster C_i, and x_s represents the s^{th} data point (where s = 1,2,...,n) belonging to the cluster C_j.

In the rest of this section, we define the proposed MRs along with the analysis/reasoning to show how the proposed MRs target both the *verification* and *validation* aspect of testing the clustering algorithms under investigation.

A. Proposed Metamorphic Relations for Unsupervised Algorithms

We provide a set of 22 MRs that can be broadly classified into 14 categories. Each of the proposed MRs either targets the *verification* or the *validation* aspect of testing the clustering algorithms under test. The MRs targeting the *verification* aspect aim to check whether the algorithms under test adhere to the necessary characteristics (from the implementation perspective) expected from the algorithms, whereas, the MRs targeting the *validation* aspect aim to check whether the algorithms under test meet the general user expectations or not. This large set of MRs are not just limited to the two algorithms under investigation (provided by the scikit-learn library), instead, they can also be used by naive users, developers and professionals to assess any clustering algorithm(s) they are interested in using for their respective problems. For better support and guidance, as shown in Tables I and II, we further provide the verification and validation analysis and reasoning for the proposed MRs. This analysis gives users a better idea in understanding the algorithms and in choosing the appropriate solution that best fits their needs.

MR1 - Duplication of Data Instance(s):

MR1.1 - Duplication of single instance: For a given source input *s*, with associated data instances assigned to clusters c_i (where $i = 1, 2, 3, ..., n$), we denote the output as O_s. If we duplicate a single instance in the follow-up input *f*, the output O_f should remain consistent i.e., $O_s = O_f$.

MR1.2 - Duplication of multiple instances: For a given source input *s*, with associated data instances assigned to clusters c_i (where $i = 1, 2, 3, ..., n$), we denote the output as O_s. If we duplicate multiple instances (i.e., each belonging to a different cluster) in the follow-up input, the output O_f should remain unchanged i.e., $O_s = O_f$.

MR1.3 - Duplication of cluster centroids: For a given source input *s*, we denote a set of centroids found as t_i (where $i = 1, 2, 3, ..., n$). If we duplicate these centroids in the follow-up input, the output O_f should remain unchanged i.e., $O_s = O_f$.

MR2 - Data Standardization: If the existing standardized data is once again standardized, the output for both the source and follow-up inputs should remain the same i.e., $O_s = O_f$.

MR3 - Duplication of Features: For a given source input *s*, we denote the output as O_s. For the follow-up input, if new features are added by duplicating existing features, the output O_f should remain unchanged i.e., $O_s = O_f$.

MR4 - Removal of Instance(s):

MR4.1 - Removal of instance from one cluster: For a given source input *s*, we denote the result as O_s. If an instance from a cluster c_i is removed for the follow-up input, it should not have any effect on changing the results for the remaining inputs, so the output O_f should remain the same i.e., $O_s = O_f$.

MR4.2 - Removal of instance from different clusters: For the follow-up input, if an instance from each of the clusters c_i (found during the source execution, where $i = 1, 2, 3, ..., n$) is removed, the output should remain consistent.

MR4.3 - Removal of multiple instances from a single cluster: For the follow-up input, if some random number of *n* instances are removed from a single cluster c_i, it should not have any effect on changing the results for the remaining inputs.

MR5 - Addition of Uninformative Attribute: For the follow-up input, if a new uninformative feature (i.e., a feature having the same value for all the instances) is added, the output should remain unchanged.

MR6 - Deterministic Output Across Multiple Runs: If a new data point is added, it should be assigned to the same cluster no matter how many times the algorithm under test is executed, i.e., the output for the execution at time $time_i$ (where i = $1, 2, 3, ..., n$) and $time_{i+1}$ should remain consistent for both the source and follow-up inputs.

MR7 - Shifting Features With constant *k*: For the follow-up input, if the feature(s) for all the instances are shifted with some constant *k*, the output should remain the same for both the source and follow-up inputs.

TABLE I
K-MEANS ALGORITHM: VERIFICATION (VR) AND VALIDATION (VD) ANALYSIS FOR THE PROPOSED MRS

#	VR?	VD?	Reasoning
MR1	×	✓	For the follow-up input, if we add a duplicate instance(s) to any of the clusters, this will result in different cluster centroid(s) (calculated using Equation 1) which may cause the original data points to get assigned to different clusters; thus, changing the output for the follow-up input. It is important to note that this is how (as shown in Equation 1) the centroid/mean calculation is implemented in the K-means algorithm under test, which ultimately means that the violation of this MR can not be characterized as violating the necessary characteristics (related to implementation) of the algorithm under test. Therefore, this MR can not be used for **verification** purposes (because its violation can not be characterized as the bug in the implementation) but can be used for **validation** purposes (because this is what the user's general expectation would be from this algorithm). ***Note**: Readers can use the same reasoning for the verification and validation aspect of testing the algorithms under test for rest of the proposed MRs.*
MR2	✓	✓	For the follow-up input, if we re-apply the standardization step, i) it will not change the mean and variance of the data points, and ii) it will maintain the same distance among the data points (similar to the source-input); thus, it should not not change the results.
MR3	×	✓	For the follow-up intput, if we add a duplicate attribute(s) to the original data points, they may have a strong influence on changing the distance between the data points and their existing centroids; thus, assigning the data points to different clusters. An example is provided (inside excel sheet available in GitHub repo), where, a violation of this MR can be seen.
MR4	×	✓	For the follow-up input, if we remove any instance(s), Equation 1 will result in the calculation of different centroids which ultimately may lead to changing the final output i.e., data points assigned to different clusters. Since this violation can not be characterized to the wrong implementation, it can only be used for validation purposes.
MR5	✓	✓	The addition of uninformative attributes (e.g., 0 or some other constant) will not change the existing relationship between the data points and will also maintain the same relationship between the data points and the initial centroids. Therefore, the output for the follow-up input should remain the same.
MR6	✓	✓	Running the algorithm at different times (keeping the initial centroids the same) should not have any effect on how the centroids are calculated. Further, as there is no change made in the data points, the output should remain the same. If the output changes, it means that there is some implementation bug in the algorithm under test.
MR7	✓	✓	If we shift all the features with some constant k, it will maintain the same distance between the data instances. So, Equation 1 will produce the same centroids as the one found during source execution; thus, not changing the final output. Let x be the centroid, y be the data point and the distance $d(x, y)$ found between them during source execution is z. Now, if we shift the features of both x and y with some constant (i.e., k) then the distance between them would be $d(x, y) = \sqrt{((x + c) - (y + c))^2} => \sqrt{(x - y)^2} => x - y => z$ (i.e., will remain the same). Therefore, the output for both the source and follow-up inputs should remain consistent.
MR8	✓	✓	If we scale all the features with some constant k, it will maintain the same relationship between the data points and the centroids. Let x and y be the two data points and during the source execution their relationship to the centroid c is $x - c < y - c$. During the follow-up execution, after scaling the features with constant k i.e., $k(x - c) < k(y - c) => x - c < y - c$, the relationship (i.e., greater than, less than, and equal) remains the same. Therefore, if the output for the follow-up input changes, that would suggest that there is some bug in the algorithm under test.
MR9	×	✓	If we replace any of the instances with some other instance (belonging to the same cluster), it may result in the calculation of different centroids (as per Equation 1); thus an instance x_i (where $i = 1, 2, .., n$) may get assigned to a different cluster.
MR10	✓	✓	For the follow-up input, if we change the location of features, it will not have any affect on the relationship between the data points and calculation of centroids (using Equation 1). So, the output should remain the same.
MR11	×	✓	For the follow-up input, if we add an informative attribute to each of the clusters instances, it may result in changing the centroids that can assign the instances to different clusters; thus, leading to a different final output.
MR12	✓	✓	For the follow-up input, if we change the order of rows/data points, it will not have any affect on existing relationship between the data point s and will lead to the calculation of same centroids (similar to the one found during source execution). Thus, the output for the source and follow-up inputs should remain consistent.
MR13	✓	✓	For the follow-input, if we apply reflection transformation, the distance between the data points will remain the same thus leading to the calculation of same centroids (using Equation 1). This should result in consistent output for both the source and follow-up inputs.
MR14	×	✓	If we add a new instance(s) to any of the clusters, this addition of new instance(s) may result in the change of centroids (different from the one found during source execution); thus, changing the final output.

MR8 - Scaling Features With Constant *k*: If the feature(s) for all the instances are scaled with some constant *k*, the output should remain unchanged for both the source and follow-up inputs.

MR9 - Replacement of Instance(s):

MR9.1 - Replacement of single instance: If a single instance belonging to a cluster c_i is replaced with some other instance x (belonging to the same cluster c_i), it should not have any impact on changing the clustering results i.e., the output O_f should remain the same for both the source and follow-up inputs.

MR9.2 - Replacement of multiple instances: If multiple instances belonging to a cluster c_i are replaced with some other instance x (belonging to the same cluster), the output O_f should remain consistent for both the source and follow-up inputs.

MR9.3 - Replacement of all instances: If all the instances belonging to a cluster c_i are replaced with some other instance x (belonging to the same cluster), the output O_f should remain consistent for both the source and follow-up inputs.

MR10 - Changing the Location of Features: If we change the order of features, the clustering result should remain unchanged for both the source and follow-up inputs.

MR11 - Adding an Informative Attribute: For a given source input *s*, we denote the identified set of clusters as C=$\{c_1, c_2, ..., c_n\}$. For the follow-up input, if a new attribute

TABLE II
AGGLOMERATIVE CLUSTERING ALGORITHM: VERIFICATION (VR) AND VALIDATION (VD) ANALYSIS FOR THE PROPOSED MRS

#	VR?	VD?	Reasoning
MR1	×	✓	For the follow-up input, if we add a duplicate instance(s) to any of the clusters, this may change the average distance between the two clusters (calculated using Equation 2), thus ending up with changing the clusters for the original data points. For understanding purposes, suppose in Fig. 2, the dendogram for the source input is cut to obtain the two clusters. The data points 7,6,6,10 will be assigned to one cluster, whereas, the data points 2,3 will be assigned to a second cluster. Now, if we duplicate the data point 3 and cut the dendogram to obtain the two clusters, it can be seen in Fig. 3 that the original data points 7,6,6,2,3 are now assigned to one cluster, whereas, the data point 10 is assigned to the second cluster; thus, changing the output for the follow-up inputs.
MR2	✓	✓	Same reasoning as provided for MR2 in Table I
MR3	×	✓	For the follow-up intput, if we add a duplicate attribute(s) to the original data points, they may have a strong influence on changing the average distance between the clusters; thus, changing the overall result. An example is provided (inside excel sheet available in GitHub repo), where, a violation of this MR can be seen.
MR4	×	✓	For the follow-up input, if we remove an instance(s), Equation 2 will result in changing the average distance between the clusters. As an example, in Fig. 2, if we remove the data point '2', the data point '3' (instead of data point '10') will get assigned to the cluster '7,6,6' . Now, if the dendogram is cut to obtain the two clusters, one cluster will have data points '7,6,6,3', whereas, the second one will have only '10', thus changing the clustering result for the follow-up inputs.
MR5	✓	✓	The addition of uninformative attribute (e.g., 0 or some other constant) will neither change the existing relationship between the data points nor the average distance between the clusters. Therefore, the output for the follow-up input should remain the same.
MR6	✓	✓	Running the algorithm at different times should not have any affect on how the average distance between the clusters is calculated. Apart from that, as there is no change made in the data points, so the output should remain consistent.
MR7	✓	✓	If we shift all the features with some constant k, it will maintain the same distance between all the data instances. So, Equation 2 will result in merging the same clusters that were merged during the source execution. As an example, let x and y be the two data-points merged together during the source execution. Now, if we shift the features of both the x and y with some constant 'c' then the distance between them i.e., $d(x,y) = \sqrt{((x+c)-(y+c))^2} => \sqrt{(x-y)^2}$, would remain similar to the one calculated during the source execution. Therefore, the output for both the source and follow-up inputs should remain the same.
MR8	✓	✓	If we scale all the features with some constant k, it will maintain the same relationship (i.e., greater than, less than, and equal) between the data points. As an example, let suppose that we have three data points x, y, and z that we are interested to group them into two clusters. During the source execution, let x and y are merged together to form one cluster, and z in another cluster. The current relationship between them is $x - y < z - y$. During the follow-up execution, after scaling the features with a constant k i.e., $k(x-y) < k(z-y) => x - y < y - z$, the relationship remains the same. Therefore, the output should remain consistent for both the source and follow-up executions.
MR9	×	✓	For the follow-up input, if we replace any of the instances with some other instance (that belongs to the same cluster), it may change the average distance between them (calculated using Equation 2) which can result in assigning the original input to different clusters. As an example, as shown in Fig. 2, if in cluster#2 (which contain data points 2 and 3), we replace the instance 2 with instance 3, it will assign them to the cluster#1 which contains the data points 7,6,6. Now if the dendogram is cut to obtain the two clusters, one cluster will have the data-points 7,6,6,3,3, whereas, the other will have only 10; thus, the original data-point (which is 3) has been assigned to the cluster#1 (instead of cluster#2). This will result in violation of this MR.
MR10	✓	✓	For the follow-up input, if we change the location of features, it will not have any affect on the relationship between the data points and calculation of average distance between the clusters (calculated using Equation 1). So, the output should remain unchanged for both the source and follow-up executions.
MR11	✓	✓	If we add an informative attribute such that it is strongly associated with each of the clusters, it will not change the existing relationship between the data points assigned to each clusters. As an example, let suppose that the dendogram for the source execution (as shown in Fig. 2) is cut to form two clusters, one cluster will have the data points 7,6,6,10, whereas, the other cluster will have the data points 2,3. Now, for the follow-up execution, if we add a new informative attribute which has the value 3 for all the instances in cluster#1 i.e., (7,3),(6,3),(6,3),(10,3) and value 4 for all the instances in cluster#2 i.e., (2,4),(3,4), it will not change the existing relationship between the clusters; thus, the output should remain the same.
MR12	✓	✓	For the follow-up input, if we change the order of rows/data instances, it will not have any affect on the way the calculation is made (using Equation 2) to merge the two clusters. Thus, the output for both the source and follow-up inputs should remain the same.
MR13	✓	✓	For the follow-input, if we apply reflection transformation to all data points, the distance between them will remain the same; thus, leading to the identification of the same clusters found during source execution.
MR14	×	✓	If we add a new instance(s) to any of the clusters, this addition of new instance(s) may result in change of average distance between the clusters; thus, changing the final output. An example is provided (inside excel sheet available in GitHub repo), where, a violation of this MR can be seen.

whose value is strongly associated with each of the clusters i.e., value x_1 with c_1, x_2 with c_2,..., and x_n with c_n, is added to the original data instances, the clustering result should remain the same for both the source and follow-up inputs.

MR12 - Rows Transformation:

MR12.1 - Reversing the order: If we reverse the order of data points/rows, the clustering result should remain consistent for both the source and follow-up inputs.

MR12.2 - Random shuffling: If we randomly shuffle the order of data points/rows, the clustering result should remain unchanged for both the source and follow-up inputs.

MR13 - Reflection Transformation: For a given source input *s*, we denote the output as O_s. For the follow-up input, if we multiply all the features with -1 (performing the data reflection), the output O_f should remain the same i.e., O_s = O_f.

MR14 - Addition of New Instance(s):

MR14.1 - Addition of instance with informative attributes: If we increase the cluster c_i density by adding a new data-point(s) in the middle of two existing data points

TABLE III
RESULTS OF TESTING K-MEANS AND AGGLOMERATIVE CLUSTERING ALGORITHMS

	K-Means				Agglomerative Clustering	
MR#	Same cluster assigned?	Centroids same?	Nearest point to centroid(s) same?	Violation? (Violation rate)	Same cluster assigned?	Violation? (Violation rate)
1.1	×	×	✓	✓(0.12%)	×	✓(0.05%)
1.2	×	×	✓	✓(0.10%)	×	✓(98.53%)
1.3	✓	✓	✓	×	N/A	N/A
2	✓	✓	✓	×	✓	×
3	✓	✓	✓	×	✓	×
4.1	×	×	✓	✓(0.14%)	×	✓(0.09%)
4.2	×	×	✓	✓(0.23%)	×	✓(0.23%)
4.3	×	×	×	✓(57.82%)	×	✓(93.40%)
5	✓	✓	✓	×	✓	×
6	✓	✓	✓	×	✓	×
7	✓	✓	✓	×	✓	×
8	✓	✓	✓	×	✓	×
9.1	✓	×	✓	✓	×	✓(2.14%)
9.2	✓	✓	✓	×	×	✓(98.07%)
9.3	×	×	✓	✓(0.02%)	×	✓(99.28%)
10	✓	✓	✓	×	✓	×
11	✓	×	×	✓	✓	×
12.1	✓	✓	✓	×	✓	×
12.2	✓	✓	✓	×	✓	×
13	✓	✓	✓	×	✓	×
14.1	×	×	✓	✓(0.05%)	×	✓(0.28)
14.2	✓	×	×	✓	×	✓(94.90%)

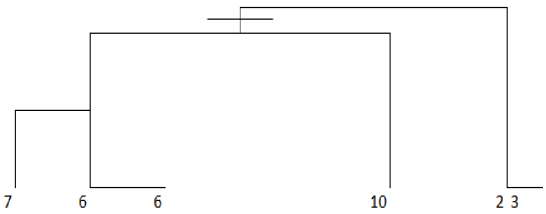

Fig. 2. Agglomerative Clustering Example

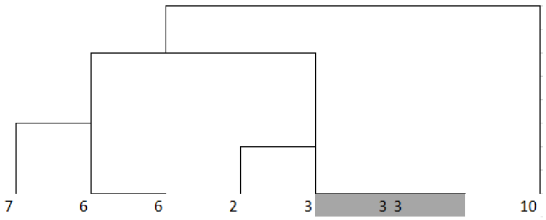

Fig. 3. MR1 for agglomerative clustering: Added 3 as a duplicate instance

(i.e., instance x and y belonging to the same cluster c_i), it should not have any effect on changing clustering results, i.e., the output should remain consistent for both the source and follow-up inputs i.e., $O_s = O_f$.

MR14.2 - Addition of instance with uninformative attributes: If we increase the cluster c_i density by adding a new data-point(s) such that all the features have 0 values in them, it should not have any effect on the clustering results, i.e., the output should remain consistent for both the source and follow-up inputs.

V. EXPERIMENTATION AND EVALUATION

To check the effectiveness of the proposed approach, we have selected an open source customer segmentation application that uses a real world multi-dimensional data-set [2]. The selected application uses K-means and Agglomerative clustering algorithms, provided by the leading Python library known as scikit-learn. It is worth mentioning that the proposed MRs are not just limited to this single application, instead, they can be used to test clustering algorithms in other domains as well (i.e., document clustering, clustering DNA patterns, and anomaly detection) in which the term 'data point/data instance' will represent either the document instance, DNA sequence, or the network traffic instance respectively.

In K-means, if the centroids are selected randomly, it will produce different results which can not be characterized as a violation of the MR. Therefore, we initialized the K-means algorithm with fixed centroids to make sure that it is converging to the same point for multiple iterations; thus, the focus is placed on testing the characteristics of the algorithms under test using the proposed MRs.

Table III summarizes the results obtained for both clustering algorithms under test. For testing the K-means algorithm, we present the results and verify the proposed MRs using multiple criteria e.g., whether the i) clusters, ii) centroids, and iii) nearest point to each centroid, are the same for both the source and follow-up inputs. This is beneficial, because if we are unlucky in identifying the violation(s) for MRs using the first criterion, we hope to uncover them using the other criteria. For example, in Table III, it can be seen that for some of the MRs e.g., MR9.1, MR11, and MR14.2, the clusters assigned

[2]https://github.com/matifkhattak/MT4UML

to data instances are the same for both the source and follow-up executions (thus, if only the first criterion is used, the MR is said to be satisfied) but those MRs were violated for the other criteria (e.g., verifying the centroids, and the nearest point to each of the centroids), thus showing the usefulness of using multiple criteria to verify the MRs. This also opens a new research direction for researchers to explore the type of different criteria that can be used for verification of MRs (for different clustering algorithms) instead of relying only on comparing the final output (i.e., clusters assigned), which may mislead the results.

The results presented in Table III show the violated MRs and their comparison for both algorithms under investigation. Each of the violated MRs either implies the implementation faults in the algorithm (verification), or its deviation from the user expectations (validation). For both the K-means and Agglomerative clustering alogrithms, 10 (out of 22) MRs (which quantifies RM1) have been violated. We also show the violation rate (i.e., RM2) for each of the violated MRs, and it can be seen that the K-means algorithm shows a higher violation rate for MR4.3, whereas, the Agglomerative clustering algorithm has a higher violation rate for MR1.2, MR4.3, MR9.2, MR9.3, and MR14.2. **This answers our first RQ that the proposed MRs are effective in testing the clustering algorithms under test**. To answer the second RQ, it can be seen that both of the algorithms under investigation show the violations for the same number of MRs. However, agglomerative clustering seems to be more sensitive to smaller changes because a small change is causing a higher violation rate among the violated MRs. **Therefore, we conclude that in comparison to agglomerative clustering, the K-means algorithm is more stable for the scenarios captured in the proposed MRs, which answers RQ2**.

VI. Conclusion And Future Work

Similar to supervised ML algorithms, one of the challenges faced in testing unsupervised algorithms is that they also suffer from the Oracle problem. Software Engineering for Machine Learning (SE4ML) is an emerging research area that focuses on applying SE best practices and methods for better development, testing, operation, and maintenance of ML-based systems. Our contribution in this work focuses on testing some popular unsupervised ML algorithms (i.e., K-means and Agglomerative clustering algorithms, provided by the leading Python library 'scikit-learn') and investigate how the traditional software testing approach i.e., Metamorphic Testing (MT) can be utilized to perform better quality assurance from both the verification and validation perspective. We propose a broader set of 22 MRs that both researchers and practitioners can take advantage of to assess the behaviour of the clustering algorithms under test from both the user's general expectation (validation) and from the implementation perspective (verification). For testing the K-means algorithm, we also propose multiple criteria that can be used for verification of the MRs. Our results show that both the algorithms under test exhibit violations (from the validation perspective) for 10 MRs, which implies that the behaviour of the algorithms deviates from the general user expectations. Further, in comparison to K-means, the Agglomerative clustering algorithm is highly susceptible to small changes in inputs and may not offer a better alternative to scenarios captured by the violated MRs.

In the future, and to improve the testing of agglomerative clustering based applications, we intend to develop new criteria that can be used to verify MRs, and which will ultimately help in building trust in using critical application algorithms. Second, to show the general applicability of the proposed MRs, we intend to utilize the proposed approach by testing a broad range of other clustering algorithms that are popular among both researchers and practitioners of the ML community.

References

[1] Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., ... & Zimmermann, T. (2019, May). Software engineering for machine learning: A case study. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 291-300). IEEE.

[2] Kumeno, F. (2019). Sofware engneering challenges for machine learning applications: A literature review. Intelligent Decision Technologies, 13(4), 463-476.

[3] Martínez-Fernández, S., Bogner, J., Franch, X., Oriol, M., Siebert, J., Trendowicz, A., ... & Wagner, S. (2021). Software Engineering for AI-Based Systems: A Survey. arXiv preprint arXiv:2105.01984.

[4] Masuda, S., Ono, K., Yasue, T., & Hosokawa, N. (2018, April). A survey of software quality for machine learning applications. In 2018 IEEE International conference on software testing, verification and validation workshops (ICSTW) (pp. 279-284). IEEE.

[5] ur Rehman, F., & Izurieta, C. (2021, September). A Hybridized Approach for Testing Neural Network Based Intrusion Detection Systems. In 2021 International Conference on Smart Applications, Communications and Networking (SmartNets) (pp. 1-8). IEEE.

[6] F. u. Rehman and C. Izurieta, "Statistical Metamorphic Testing of Neural Network Based Intrusion Detection Systems," 2021 IEEE International Conference on Cyber Security and Resilience (CSR), 2021, pp. 20-26, doi: 10.1109/CSR51186.2021.9527993.

[7] Pei, K., Cao, Y., Yang, J., & Jana, S. (2017, October). Deepxplore: Automated whitebox testing of deep learning systems. In proceedings of the 26th Symposium on Operating Systems Principles (pp. 1-18).

[8] Tian, Y., Pei, K., Jana, S., & Ray, B. (2018, May). Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In Proceedings of the 40th international conference on software engineering (pp. 303-314).

[9] Moreira, D., Furtado, A. P., & Nogueira, S. (2020, August). Testing acoustic scene classifiers using Metamorphic Relations. In 2020 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 47-54). IEEE.

[10] Yang, S., Towey, D., & Zhou, Z. Q. (2019, May). Metamorphic exploration of an unsupervised clustering program. In 2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET) (pp. 48-54). IEEE.

[11] Xie, X., Zhang, Z., Chen, T. Y., Liu, Y., Poon, P. L., & Xu, B. (2020). METTLE: a METamorphic testing approach to assessing and validating unsupervised machine LEarning systems. IEEE Transactions on Reliability, 69(4), 1293-1322.

[12] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

[13] Witten, I. H., & Frank, E. (2002). Data mining: practical machine learning tools and techniques with Java implementations. Acm Sigmod Record, 31(1), 76-77.

[14] Chen, T. Y., Cheung, S. C., & Yiu, S. M. (2020). Metamorphic testing: a new approach for generating next test cases. arXiv preprint arXiv:2002.12543.

[15] Basili, V. R. (1992). Software modeling and measurement: the Goal/Question/Metric paradigm.