New convex-based metamorphic relations and large-scale machine learning model evaluation

Jessy Colonval[1] and Fabrice Bouquet[1]

Université Marie et Louis Pasteur, CNRS, institut FEMTO-ST (UMR 6174), F-25000 Besançon, France
{jessy.colonval,fabrice.bouquet}@femto-st.fr

Abstract. Machine learning (ML) models are victims of the oracle problem, i.e. it is not possible to know with absolute confidence an output for a given input. This prevents them from being evaluated using conventional software engineering techniques. However, there is an approach called "metamorphic relation" that reduces the oracle problem and helps to evaluate ML models. Unlike conventional tests, a metamorphic relation does not check if an input produces a specific output, but checks if a relationship between inputs and outputs is respected. Naturally, metamorphic relations have already been proposed in the literature, either to evaluate the behavior of a specific ML model, or to evaluate the general behavior of any ML model. The purpose of this paper is to propose new metamorphic relations to complement those of the literature, in order to propose a more complete methodology for evaluating ML models. So, in order to challenge this methodology, all these metamorphic relations are used to evaluate 21 different machine learning algorithms.

Keywords: Metamorphic relations · Outliers · Machine learning · Convexes · Software engineering.

1 Introduction

This paper is in the field of software engineering, with a particular focus on quality assurance, i.e. the set of actions required to provide satisfactory confidence in the quality of a software product in accordance with pre-established requirements and expectations. [1]. It can be divided into four cyclical phases known as the **Deming** or the **PDCA** cycle:

- **Plan**: plan and establish process-related objectives necessary for software quality.
- **Do**: develop and test processes.
- **Check**: monitor and check that processes meet planned objectives.
- **Act**: implement actions to improve processes.

This implies the implementation of tests and the necessity to be able knowing the output for every input provide, which is not easy for machine learning algorithms.

Martin D. Davis and Elaine J. Weyuker [2], define the mechanism that verifies the correctness of the results obtained according to those expected as an **oracle**. And the belief that a tester is systematically able to determine if a test output is correct, like the oracle's hypothesis. But, this assumption is wrong and there are circumstances where this is not possible:

(1) there is no **oracle** ;
(2) it's theoretically possible to determine an **oracle**, but in practice it's too difficult to implement with current tools and knowledge.

Machine learning algorithms correspond to the scenario **(2)**. However, the use of "metamorphic relations" helps to reduce the **oracle** problem [3]. Unlike conventional tests, metamorphic relations do not check the correctness of outputs produced by a program according to given inputs. They verify that necessary relations among inputs and outputs of multiple executions of the target function.

Machine learning models are capable of interpolating unknown data based on training with an already known dataset. For this purpose, an algorithm called the **training algorithm** handles the training phase, which ultimately produces a **model**. Thus, there are two distinct aspects worth evaluating, each of which highlights different types of error:

(1) if a **training algorithm** is defective, then this translates into the presence of bugs in its algorithm and all future trainings will likely be poor quality;
(2) if a **model** is defective, then this translates into its inability to behave correctly according to previously defined expectations.

The literature already proposes metamorphic relations for evaluating in specific contexts AI/ML **models**, and metamorphic relations for evaluating the overall effectiveness of **training algorithms**. Xie et al. propose various metamorphic relations that can be used to evaluate the behavior of different supervised classifiers [4]. And Saha and Kanewala are using and implementing some of these relations to conduct a larger-scale study of their effectiveness in mutant detection of kNN classifier [5]. The metamorphic relations highlighted in these works are designed to evaluate the behavior of **training algorithms**. While, Zhou and Sun propose metamorphic relations designed to evaluate predictive capacity of an obstacle perception subsystem of an autonomous car [6]. In other words, it evaluates the **model** in a specific context.

This paper introduces 8 new metamorphic relations in order to evaluate the prediction capacities of any machine learning **model** (see section 4). Part of these relations will be based on the use of convexes of which certain concepts are presented in section 3. However, these relations will be useful only for the evaluation of the **model** part. In order to have a complete evaluation schedule and evaluate the correctness of a **training algorithm**, 5 other metamorphic relations from the state-of-art will be chosen (see section 2). Thus, these 13 relations will be used to evaluate a large amount of **model** produces by 21 **training algorithms** from the *scikit-learn*[1] libraries (see section 5).

[1] `https://scikit-learn.org/stable/`

2 Context & State-of-the-art

2.1 Dataset concepts

A dataset can be visualized like a table with rows and columns. Each row represents an individual, often unique, while the columns represent the characteristics of that individual. Several terms exist to designate them, but here, the rows will be called **points** and the columns will be called **attributes**. In general, there are two types of attributes [7, Chap 11]:

- The **behavioral attributes**, A_B, are attributes of interest. They are defined for all points, many of which have only one. Since they characterise a certain quality of the point, these attributes have **categorical** or **discrete numerical** values. For example: the type of glass, the presence of a cardiac irregularity, or the description of an image. The values of a **behavioral attribute** often refer to as a class, but here we'll call them **behavioral values**, V_B, to avoid confusion.
- The **contextual attributes**, A_C, is an attribute expressing the characteristics of a point, defined over a discrete or continuous domain of values, also known as spatial attributes. One point often has several. For example: the composition of a pane of glass, the number of heartbeats per minute or the RGB code of a pixel.

Several **behavioral attributes** may be present, but they can be reduced to a single attribute by merging all the **behavioral values**.

There are several kinds of datasets, depending on the type of data and the number of attributes they contain. But in this paper used, only datasets **multivariate** (i.e. more than 2 **contextual attributes**) with mixed values, **numeric** and **categorical**. Thus, it is possible to define a domain specific to a dataset.

Definition 1 (Dataset domain). *The smallest real value interval that contains all the* ***contextual values*** *of all the* ***contextual attributes*** *of the dataset.*

All datasets used in this paper will be created synthetically by an algorithm from literature [8]. This algorithm builds datasets using several clusters and a hypercube (or a random polytope) of n_{A_C} dimensions, where n_{A_C} is the number of **contextual attributes**. Each cluster is composed of points of the same behavioral value, and its centroid of the hypercube's vertices. So, depending on the number of clusters per behavioral value and the size of the hypercube, quite different datasets can be obtained. A Python version has been implemented and integrated into the library *scikit-learn*: *make_classification*[2].

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html`

2.2 State-of-the-art metamorphic relations

As mentioned in the introduction, we have selected 5 relation metamorphic from the state-of-art in order to complete evaluation schedule. Their goal is to evaluate the correctness of **training algorithms** before the evaluation of the **model** part. They evaluate the ability of a **training algorithm** to tolerate minor modifications in the training set and produce **models** equivalent to those produced without these modifications. Note that these relations do not evaluate a **model**'s ability to make good predictions, which means that **training algorithms** that validate these relations may produce **models** with poor-quality predictions.

MR 1 (Identity)**.** *If no changes have been applied to the training set and the meta-parameters of the* ***training algorithm*** *are the same, then the predictions on the test set should remain unchanged [5].*

MR 2 (Points shuffle)**.** *If* ***points*** *are swapped and the meta-parameters of the* ***training algorithm*** *are the same, then the predictions on the test set should remain unchanged [9].*

MR 3 (Attributes shuffle)**.** *If* ***contextual attributes*** *are swapped and the meta-parameters of the* ***training algorithm*** *are the same, then the predictions on the test set should remain unchanged [4, 5].*

MR 4 (Transformation)**.** *If an affine transformation function* $f(x) = kx + b, (k \neq 0)$ *is applied to every value* x *in a subset of contextual attribute of the training set and the meta-parameters of the* ***training algorithm*** *are the same, then the predictions on the test set should remain unchanged [4, 5].*

MR 5 (Behavioral value permutation)**.** *If the behavioral values of all points are interchanged and the meta-parameters of the* ***training algorithm*** *are the same, then the predictions on the test set must apply this permutation [4, 5].*

Note that some **training algorithms** have random aspects that can be controlled using a random seed. In order to avoid influencing the evaluation process, it is crucial that this seed, like the other meta-parameters, stay constant throughout.

3 Convexes concept presentation

Some proposed metamorphic relation uses the convexes as a way to separate the influence area of each behavioral value. Thus, it is necessary to introduce the notion of convex, and point creation in a convex algorithms.

3.1 What is a convex? How do you create them?

Definition 2 (Convex). *A part* H *of* $\mathbb{R}^n$ *said to be convex if, for all pairs* (x, y) *of elements of* H*, the segment* $[x, y]$ *is entirely contained within* H*. In other words,* H *is convex when* $\forall x, y \in H$ *and* $\forall \lambda \in [0; 1], \lambda x + (1 - \lambda)y \in H$ *[10].*

This definition is adapted to be applied to any datasets in order to separate points by behavioral value such that all points with the same value are contained in a single convex.

Definition 3 (Convex per behavioral value). *Thus, for a dataset D composed of n_C contextual attributes and 1 behavioral attribute containing n behavioral values such as $\{b_0, \ldots, b_n\}$. Then D is separated into convexes containing all the points of a behavioral value such that $H_{b_o} \cup \ldots \cup H_{b_n} = D$.*

Several algorithms to build convexes exists in the state-of-art. For the rest, we will use the Python implementation[3] of the algorithm QuickHull in order to build convexes in any dimensions [11]. Thus, according to the definition 3, the separation of a dataset in different convexes for each of its behavioral values gives something like the figure 1.

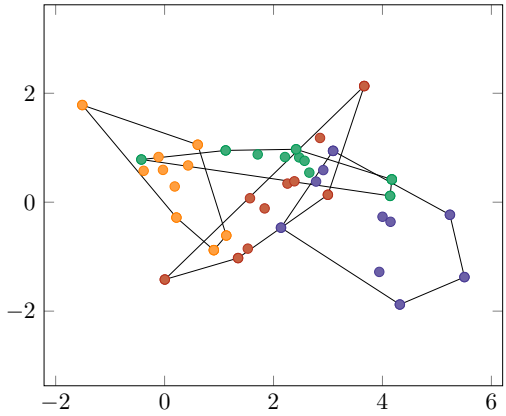


Fig. 1: A dataset divides in 4 convexes, 1 per behavioral value

3.2 How to create a point in only one convex?

Some convex-based metamorphic relations need to create points inside exactly one convex or inside several convexes simultaneously. These points could be randomly created until they are in the desired convex(es), but this approach is too time-consuming, especially when convex dimensions are large. That's why we're proposing two algorithms for creating a point directly in a convex based on its properties.

▼ **Algorithm 3.1 : Create a point p in a single convex H**

Step 1 → Randomly select 2 facets of H: f_0 and f_1;
Step 2 → Randomly place a point p_0 on f_0 and a point p_1 on f_1;

[3] `http://www.qhull.org/`

Step 3 → Randomly place a point p on the segment $[p_0; p_1]$;
Step 4 → Repeats the steps 1 to 3 until p isn't inside an another convex.

The algorithm 3.1 works because the definition 2 guarantees that all segments drawn from two points on the facets of a convex are fully included in the same convex. Thus, all points on this segment are also in the convex.

3.3 How to create a point in several convexes?

There's already a method in the literature for creating points in an intersection of several convexes, but it only works for 3D polyhedral convexes. So, on the basis of the Sutherland-Hodgman algorithm [12] and algorithm 3.1, we have implemented an algorithm that allows this creation for any convex shape.

▼ **Algorithm 3.2 : Create a point p in the convexes H_i and H_j, where $H_i \neq H_j$**

Step 1 → Let $V_{i,j} = \emptyset$ be the set of points;
Step 2 → Check for all vertices s_j of H_j if they are in H_i, if yes then:
  (2.1) → add it to $V_{i,j}$;
  (2.2) → if one of these adjacent neighbors s'_j is not in H_j, then add the intersection between the segment $[s_j, s'_j]$ and a facet of H_j to $V_{i,j}$.
Step 3 → If $V_{i,j}$ is not empty, then the intersection between H_i and H_j is the convex $H_{i,j}$ whose vertices are $V_{i,j}$.
Step 4 → Use the algorithm 3.1 on $H_{i,j}$ to create p.

Algorithm 3.2 can be used to create points in an intersection of more than 2 convexes. In this case, the convex H_i must be replaced by a set of several convexes. Plus, it should be noted that several convexes can superpose without any of their respective vertices being included in an another convex. Thus, algorithm 3.2 won't create any points.

4 Metamorphic relations for model validation

This section introduces new metamorphic relations that aim to evaluate the capacity of any ML model to make consistent predictions regarding the training corpus used.

4.1 Convex-based metamorphic relations

The datasets are consistent in the distribution of their behavioral values, and this consistency is induced by the models based on contextual attributes. After training, these models create "zones of influence" specific to each behavioral

value, which make it possible to predict which behavioral value the points to be predicted belong to. Thus, being able to know this zone of influence before training would make it possible to generate test cases and evaluate the relevance of these predictions.

However, it is impossible to know it exactly without training the models first. But it is possible to obtain an approximate and simplified version that will be different from the one used by the model, but they are likely to share common properties. This is even more true for the most trivial parts of this distribution consistency, e.g., an area in which only points with the same behavioral value are present. The reason is that they will come from the same dataset and there is no reason for the models to interpret these parts differently. This is why convexes are used here to construct these approximate areas of influence.

Convex-based metamorphic relations can be separated into two categories according to point placement: **(1)** when they are inside one or several convexes; **(2)** when they are outside all convex. This separation serves as the basis for the creation of our various metamorphic relations.

By category **(1)**, we can infer 2 relations according to the location of the point to be evaluated. The RM 6 relation when the point is inside a single convex, and the RM 7 relation when it is inside several convexes (i.e. at least 2). However, in both cases, the **models** must predict the point as having one of the behavioral values represented by the convexes they contain it.

MR 6 (Membership)**.** *Verifies that all points in exactly one convex H_c, where c is the represented behavioral value, must be predominantly predicted by the* ***models*** *as c.*

MR 7 (Superposition)**.** *Verifies that all points present in m convex $\{H_{c_0}, \ldots, H_{c_m}\}$, where $c_0, \ldots, c_m$ are the m behavioral values represented, must be predominantly predicted by the* ***models*** *as one of the m behavioral values.*

In the same way as category **(1)**, category **(2)** deduces 4 metamorphic relations according to the location of the point to be evaluated relative to the domain of the training corpus. The MR 8 relation when the point is inside the domain, the MR 9 when the point is outside the domain and the MR 10 and MR 11 relations when the point is on the domain boundaries.

MR 8 (Attachment)**.** *Verifies that all points present in zero convexes must be predominantly predicted by the* ***models*** *as having a behavioral value equal to the closest convex(es).*

MR 9 (Robustness)**.** *Verifies that all points outside the domain of the training corpus must not cause error when a* ***model*** *tries to predict its behavioral value.*

MR 10 (Boundary Robustness)**.** *Verifies that all points at the boundaries of the training corpus domain must not cause error when a* ***model*** *tries to predict its behavioral value.*

MR 11 (Boundary Attachment)**.** *Verifies that all points at the boundaries of the training corpus domain must be predominantly predicted by the* ***models*** *as having a behavioral value equal to the closest convex(es).*

4.2 How to attach a point to a specific convex?

Relations MR 8 and MR 11 expect the models to predict the point evaluated as having the behavioral value of the nearest convex. However, this notion of proximity is unclear and needs to be defined in concrete terms. For that, it is necessary to determine the reference element of the convex on which the calculations will be based, e.g. **vertices** or **edges**. Plus, convexes contain irregularly distributed points, so the reference element can be more or less isolated from the points. Thus, we decided to take this isolation into consideration during the proximity calculation.

Definition 4 (Isolation score of a vertex). *Let d be a function of distance between two points, P_H the set of points on the convex H and u a vertex of the convex such that $u \in P_H$. Then the **isolation score** $\omega \in [0;1]$ of u is:*

$$\omega_H(u) = \frac{1}{card(P_H) - 1} * \sum_{p \neq u}^{p \in P_H} \frac{d(p,u)}{max(\{d(q,u) \,|\, \forall q \in P_H\})} \tag{1}$$

Definition 5 (Isolation of an edge). *Let d be a distance function between a point and a segment, $[u,v]$ an edge of the convex H and P_H the set of points of this same convex. Then the **isolation score** $\omega \in [0;1]$ of $[u,v]$, where u and v are the two points of the edge, is:*

$$\omega_H([u,v]) = \frac{1}{card(P_H) - 2} * \sum_{p \neq u, p \neq v}^{p \in P_H} \frac{d(p,[u,v])}{max(\{d(q,[u,v]) \,|\, \forall q \in P_H\})} \tag{2}$$

Definition 6 (Attachment score). *This score, τ, is equal to the minimum product of an isolation score ω and the distance between the reference element of a convex H and a point p_o outside H. So, when the reference element is a **vertex**, then:*

$$\tau_H(p_o) = min(\{\omega_H(u) * d(p_o,u) \,|\, u \in S_H\}) \tag{3}$$

where S_H is the set of vertices of H.
*And, when the reference element is an **edge**, then:*

$$\tau_H(p_o) = min(\{\omega_H([u,v]) * d(p_o,[u,v]) \,|\, [u,v] \in A_H\}) \tag{4}$$

where A_H is the set of edges of H.

Thus, according to these definitions, the closer the points in the convex H are to the reference element chosen (a **vertex** or an **edge**), the more ω tends to 0. Inversely, it tends to 1 the further away they are. However, several convexes may have relatively close attachment scores to p_o, to the degree that there may be some doubt as to which one should be attached. This means that p_o is close to several different behavioral value influence zones, we then say that p_o is **ambiguous**.

If a point is **ambiguous** here, then there's a high chance that it will also be **ambiguous** for the **models** when they try to predict it. In this case, the models will have good reason to be wrong, without this necessarily reflecting poor

training quality. Thus, **ambiguous** points will be attached to several convexes simultaneously and the **models** will be able to predict one of the behavioral values represented by these convexes without causing a failure of the metamorphic relations.

Definition 7 (Set of convexes attached). *Let E_H be the set of convexes and $\epsilon \in [0; 1]$ be the percentage of **interval of attachment** desired. Then the relation $R_{E_H}(p_o)$ which associates with p_o the set of convexes in E_H for which their **connection scores** are included in the nearest convex's connection interval, as follows:*

$$R_{E_H}(p_o) = \{H \in E_H \,|\, (1-\epsilon)\tau_H(p_o) \leq (1+\epsilon)\tau_{min}(p_o)\} \tag{5}$$

where $\tau_{min}(p_o) = min(\{\tau_H(p_o) \,|\, \forall H \in E_H\})$.

Equation 5 adds to the set of convexes attached to p_o all those with **attachment scores** equal, $\pm\epsilon$, to the smallest **attachment score** obtained. In this paper, the ϵ will be fixed at 0.01 and the reference elements used will be the vertices. Because, the results obtained using the vertex-based approach were significantly better than those obtained using edges.

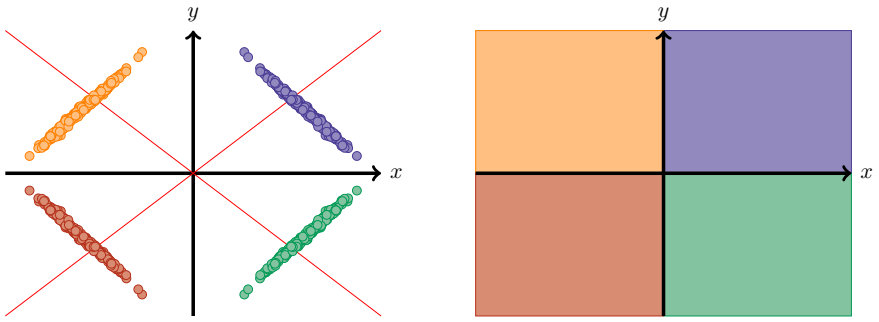
4.3 Non-convex metamorphic relations

Models have other behaviors that can be evaluated, but which are not translatable into convex-based **metamorphic relations**. Indeed, the latter makes it possible to evaluate the predictive capacity of a **model**, but they do not make it possible to evaluate: **(1)** the accuracy of their predictions; **(2)** and their aberration management.

The aspect **(1)** focuses on the ability of **models** to be accurate in their predictions and their ability to correctly manage the switch from one behavioral value to another when the precision levels of the contextual values are very low. This means measuring the accuracy of predictions made at points that lie on the boundary between several different behavioral value influence zones to within a few hundredths. Indeed, it's difficult to have confidence in a **model** that lacks precision.

MR 12 (Precision)**.** *Let's consider a dataset of n_c contextual attributes, whose values are defined in the D domain, and* 1 *behavioral attributes with minimum* 2 *behavioral value. Placed symmetrically to the axes, a maximum of 2^{n_c} influence zones each representing a single behavioral value. Verifies that all predictions made in D result in the behavioral value of the nearest behavioral value influence zones.*

Figure 2 illustrates this metamorphic relation. Let's consider a dataset with 2 contextual attributes in the domains $[-7, 7]$ in x and $[-4, 4]$ in y and 1 behavioral attributes with 4 behavioral values. Figure 2a is obtained after symmetrically placing 4 clusters, 1 per value. Thus, all predictions made on this domain must take the form of Figure 2b.



(a) A well-formed dataset (b) Expected predictions

Fig. 2: Example of MR 12

The aspect **(2)** concerns the ability of **models** to correctly interpret outliers during the prediction process. Indeed, it is difficult to have confidence in **models** that are unable to handle mistakes correctly. The aim is to voluntarily ask a **model** to predict outliers in order to determine whether it is able to interpret them correctly by always giving predictions different from their associated behavioral values.

MR 13 (Outlier)**.** *Verifies that outliers are predicted differently than their associated behavioral values by the* ***models*** *[9].*

An outlier can be detected when it was already present in the dataset [13–17], or it can be created. In this paper, those used have no pre-existing outliers, so they will be created by an algorithm.

5 Models and training algorithms evaluation

The metamorphic relations presented will be used in order to evaluate several machine learning **training algorithms** from the *scikit-learn* library in Python and **models** produced.

5.1 Training algorithms evaluated & datasets used

In order to evaluate different algorithms and observe how the metamorphic relations defined, five different kinds of **training algorithms** that can be encountered have been identified according their paradigm that defines them:

(1) decision trees [18–25];
(2) support vector machines [15, 26];
(3) overall distribution of behavioural values;

(4) neural networks [27, 28];
(5) neighborhoods [29–31].

However, this separation is not strict, and a **training algorithm** can belong to several paradigms. Thus, the algorithms evaluated and their paradigms are those shown in Table 1.

	Paradigm				
	(1)	**(2)**	**(3)**	**(4)**	**(5)**
Bernoulli NB	■			■	
Categorical NB	■			■	
Complement NB	■			■	
Decision Tree Classifier	■				
Extra Tree Classifier	■				
Dummy Classifier			■		
Gaussian Process Classifier		■			
Gradient Boosting Classifier	■				
K Neighbors Classifier					■
Linear Discriminant Analysis		■			
MLP Classifier				■	
Multinomial NB	■			■	
Nearest Centroid					■
Passive Aggressive Classifier		■			
Perceptron				■	
Quadratic Distriminant Analysis		■			
Radius Neighbors Classifier					■
Random Forest Classifier	■				
Ridge Classifier		■			
SGD Classifier		■			
SVC		■			

Table 1: **Training algorithms** evaluated and their paradigms

Training algorithms use training sets to build a trained **model**. Thus, datasets from table 2 have different shapes (according to 6 key characteristics) to evaluate different **models** without being biased by a particular dataset choice. Thus, MR 1 to 5, MR 9, 10 and MR 13 will evaluate **models** trained with all these datasets. While other relations will evaluate **models** trained with the first dataset of table 2. However, MR 7 and 12 will be evaluated with other datasets that follow a very specific forms: 5 with different kind of convexes intersection for MR 7 and the one from figure 2a for MR 12.

In addition, several combinations of meta-parameters will be used for each training algorithm, always with the goal to build different models without being biased by a particular choice of meta-parameter. These meta-parameters are

Points	$\mathbf{A_C}$	$\mathbf{V_B}$	**Noise**	**Distribution**	$\frac{\mathbf{n_p * n_{A_C}}}{\mathbf{n_{V_B}}}$
250	10	2	**no**	**homogeneous**	$\geq \mathbf{2}$
250	1 000	2	**yes** - 10%	**homogeneous**	$< \mathbf{2}$
25 000	10	2	**yes** - 60%	**heterogeneous**	$\geq \mathbf{2}$
25 000	1 000	2	**no**	**heterogeneous**	$\geq \mathbf{2}$
125 000	100	2	**yes** - 60%	**homogeneous**	$\geq \mathbf{2}$
125 000	100	2	**no**	**homogeneous**	$\geq \mathbf{2}$
250	100	5	**yes** - 60%	**heterogeneous**	$< \mathbf{2}$
25 000	100	5	**yes** - 10%	**homogeneous**	$\geq \mathbf{2}$
125 000	10	5	**yes** - 10%	**heterogeneous**	$\geq \mathbf{2}$

Table 2: Training sets where n_p, n_{A_C} and n_{V_B} are the numbers of points, **contextual attributes** and **behavioral values**

presented in a GitHub directory of one of the authors[4] with the complete results obtained.

5.2 Results obtained

To summarize, for each **training algorithm**, several **models** are trained from different datasets and different meta-parameter combinations. However, among these **models**, we observed that some of them gave identical predictions regardless of the point. This was due to an incorrect choice of meta-parameter for a specific dataset. These **models** were considered to be "poorly trained" and were removed from the study. Thus, each **model** remains is evaluated with the 13 metamorphic relations presented.

Metamorphic relations are evaluated with various points in order to reinforce the results obtained: MR 1 to 5, MR 9 and 10 will use $10,000$ random points; MR 6, 7, 8 and 11 will use $1\,000$ random points according the restrictions of each relations; MR 13 will use 50 outliers; and MR 12 all points every 0.01 in the intervals $x \in]-7, -0.07[\cup]0.07, 7[$ and $y \in]-4, -0.04[\cup]0.04, 4[$[5].

In addition, each relation has a success rate that must be reached in order to validate it. Indeed, even the high-quality **models** cannot have perfect predictive capacity. It therefore seems appropriate to allow models a degree of error when they are evaluated. Thus, MR 1 to 5, MR 9 and 10 need 100% success; MR 13 need 96% success; MR 12 and 7 need 95% success; MR 6 need 90% success; MR 8 and 11 need 75% success. All these success rates have been decided according to the importance of and confidence placed in these relations.

In the end, Table 3 shows the results obtained after evaluating a total of $70,276$ **models**, all **training algorithms** included. Each of its cells contains the average percent of **models** per dataset, produced by a specific **training algorithm**, that validate one of the metamorphic relations according to the success

[4] `https://github.com/JessyColonval`

[5] Note that the intervals are slightly different from those given in section 4.3. Points close to the axes are ignored to tolerate model errors when the distance between two clusters is too close.

rate, defined earlier. A standard deviation score is shown when all datasets have been used.

Overall, these results are promising. Even if none of the algorithms evaluated fully validates all the metamorphic relations, most of them succeed to produce for each relation at least one **model** able to validate it. Thus, with the appropriate choice of meta-parameters for a specific training set, it seems possible to obtain a **model** that validates all these relations. However, few relations that evaluate **training algorithms** in particular seem to show the presence of bugs in some of them when particular meta-parameters are used:

– 2 of 21 **training algorithms** produce **models** that fail MR 1. This means that even with the same conditions, these algorithms can provide two differ-

		Training algorithm					Model							
		1	2	3	4	5	6	8	11	9	10	7	12	13
(1)	*Decision Tree Classifier*		98.89% ±2.90	1.34% ±0.59		54.54% ±4.39	21.00%	0.30%	1.35%			98.10%	44.80%	45.86% ±10.57
	Extra Tree Classifier		99.01% ±1.02	1.42% ±0.96		54.13% ±5.52	20.44%	0.20%	1.24%			98.38%	45.05%	49.80 ±10.01
	Gradient Boosting Classifier		44.33% ±13.45	0.26% ±0.36	90.05% ±14.07	25.33% ±7.69	20.44%	0.20%	1.24%			84.64%	43.08%	72.42 ±6.36
	Random Forest Classifier	51.18% ±6.19	41.41% ±20.00	0.10% ±0.14	51.18% ±6.19	26.24% ±10.03	34.15%	8.54%	10.98%			71.88%	90.40%	75.02 ±11.81
(1) (4)	*Bernoulli NB*					61.11% ±9.62						84.44%	50.00%	
	Categorical NB													
	Complement NB					66.67% ±0.00		50.00%	50.00%			50.00%		94.44% ±9.62
	Multinomial NB					61.1% ±9.62						20.00%		
	Gaussian Process Classifier					66.67% ±0.00						95.00%		
	Linear Discriminant Analysis		96.77% ±5.77		96.77% ±5.77	66.11% ±0.96		28.57%	28.57%					90.32% ±9.53
	Passive Aggressive Classifier		1.08% ±1.47		71.51% ±22.35	40.19% ±28.09	88.31%	47.40%	19.27%			55.18%	63.40%	90.50% ±11.05
(2)	*Quadratic Discriminant Analysis*		80.30% ±34.12		80.30% ±34.12	93.94% ±10.50		66.67%						35.10% ±33.47
	Ridge Classifier	72.77% ±0.70	55.51% ±1.15		98.76 ±0.01	48.03% ±0.55		36.70%	23.60%			33.02%		90.01 ±16.57%
	SGD Classifier		1.15% ±1.63		47.87% ±8.70	37.57% ±22.18	59.50%	36.70%	23.60%			54.09%	18.69%	65.26% ±6.14
	SVC		6.85% ±11.41		98.24% ±3.05	6.86% ±9.68	78.00%	3.60%				94.57%	82.53%	78.76% ±16.95
(3)	*Dummy Classifier*					14.29% 24.74						66.67%		31.32% ±9.91
(4)	*MLP Classifier*		23.33% ±10.36	0.75% ±0.96	64.59% ±15.80	0.22% ±0.38	63.44%	38.39%	34.26%			79.40%	52.15%	84.58% ±6.72
	Perceptron				60.53% ±5.21	22.34% ±0.78	43.07%	53.81%	39.87%			35.04%	1.10%	90.17% ±8.84
(5)	*K Neighbors Classifier*				97.27% ±0.00	61.82% ±0.00			97.30%			77.50%	80.20%	95.91% ±0.00
	Nearest Centroid					60.71% ±2.91	59.38%	28.12%	46.88%			60.00%	62.50%	90.27% ±4.64
	Radius Neighbors Classifier					53.85% ±0.00	81.82%	81.82%	81.82%			77.30%	32.19%	89.74% ±0.00

Legend: 100% | [75%; 100%[| [50%; 75%[| [25%; 50%[|]0%; 25%[| 0% | no result

Table 3: Results of the evaluation of **training algorithms** and their **models**

ent **models**. An explanation could be the presence of a random aspect that is not properly managed with a seed.
– These algorithms produce models that regularly fail the MR 2 and 3. Some of these errors can be explained by the existence of an order or selection of rows and/or columns. However, according to the documentation and the meta-parameters used, some models fail even when they use all the rows and/or columns without changing their order. One explanation might be that there is a specific selection process, contrary to what is planned.

A possible follow-up to this work could be to propose a correction of these few algorithms so that they can validate these relations.

6 Conclusion

This paper presented 8 new metamorphic relations, 6 of them based on the use of convexes which approximate the zones of influence of each behavioral value, in order to approach as closely as possible the zones built by the **models** after their training. This particular construction allows a **model**'s predictive capacity to be evaluated. While the 2 others are used to evaluate their accuracy and behavior in the presence of outliers. With the help of 5 other relations of the state of the art and that focus on the evaluation of a **training algorithm**, 21 machine learning algorithms based on different paradigms and from the *scikit-learn* Python library were evaluated. By using several training sets and several combinations of meta-parameters, a total of $70,276$ models have been trained and evaluated. Finally, the metamorphic relations that evaluate **training algorithms** have revealed the probable existence of implementation bugs in some of them. While the new relations have shown their relevance in the evaluation of trained **models**.

This work demonstrates the relevance of using convexes in **model** evaluation. However, only **models** trained on synthetic datasets, in the form of clusters, were evaluated. Future works should extend this type of evaluation to **models** trained on real-world clustered datasets, but also to datasets of any shape.

References

1. A. April and C. Y. Laporte, *L'assurance Qualité Logicielle 1.* Lavoisier, 2011.
2. M. D. Davis and E. J. Weyuker, "Pseudo-oracles for non-testable programs," in *Proceedings of the ACM '81 Conference*, pp. 254–257, ACM, Jan. 1981.
3. T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic Testing: A New Approach for Generating Next Test Cases," 1998.
4. X. Xie, J. W. K. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," *Journal of Systems and Software*, vol. 84, pp. 544–558, Apr. 2011.
5. P. Saha and U. Kanewala, "Fault Detection Effectiveness of Metamorphic Relations Developed for Testing Supervised Classifiers," in *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, pp. 157–164, Apr. 2019.

6. Z. Q. Zhou and L. Sun, "Metamorphic Testing of Driverless Cars," *Commun. ACM*, vol. 62, pp. 61–67, Feb. 2019.
7. C. C. Aggarwal, *Outlier Analysis*. Cham: Springer International Publishing, 2 ed., 2017.
8. I. Guyon, "Design of experiments for the NIPS 2003 variable selection benchmark," in *Undefined*, 2003.
9. A. Dwarakanath, M. Ahuja, S. Sikand, R. M. Rao, R. P. J. C. Bose, N. Dubash, and S. Podder, "Identifying implementation bugs in machine learning based image classifiers using metamorphic testing," in *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2018, (New York, NY, USA), pp. 118–128, Association for Computing Machinery, July 2018.
10. V. Klee, "What Is a Convex Set?," *The American Mathematical Monthly*, vol. 78, pp. 616–631, June 1971.
11. C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software*, vol. 22, pp. 469–483, Dec. 1996.
12. I. E. Sutherland and G. W. Hodgman, "Reentrant polygon clipping," *Commun. ACM*, vol. 17, pp. 32–42, Jan. 1974.
13. J. Colonval and F. Bouquet, "Multidimensional Adaptative kNN over Tracking Outliers (Makoto)," in *Advanced Data Mining and Applications* (X. Yang, H. Suhartanto, G. Wang, B. Wang, J. Jiang, B. Li, H. Zhu, and N. Cui, eds.), (Cham), pp. 535–550, Springer Nature Switzerland, 2023.
14. J. Colonval and F. Bouquet, "Hunting inside n-quantiles of outliers (hino)," in *Data Science: Foundations and Applications* (X. Wu, M. Spiliopoulou, C. Wang, V. Kumar, L. Cao, X. Zhou, G. Pang, and J. Gama, eds.), (Singapore), pp. 420–431, Springer Nature Singapore, 2025.
15. K.-L. Li, H.-K. Huang, S.-F. Tian, and W. Xu, "Improving one-class SVM for anomaly detection," in *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, vol. 5, pp. 3077–3081 Vol.5, Nov. 2003.
16. M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers.," in *ACM Sigmod Record*, vol. 29, pp. 93–104, June 2000.
17. H. Liu, S. Shah, and W. Jiang, "On-line outlier detection and data cleaning," *Computers & Chemical Engineering*, vol. 28, pp. 1635–1647, Aug. 2004.
18. J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, Mar. 1986.
19. J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., elsevier science ed., 1993.
20. S. L. Salzberg, "C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993," *Machine Learning*, vol. 16, pp. 235–240, Sept. 1994.
21. J. R. Quinlan, "Improved Use of Continuous Attributes in C4.5," *Journal of Artificial Intelligence Research*, vol. 4, pp. 77–90, Mar. 1996.
22. A. D. Gordon, L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and Regression Trees.," in *Biometrics*, vol. 40, p. 874, Sept. 1984.
23. P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, pp. 3–42, Apr. 2006.
24. L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001.
25. J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, pp. 367–378, Feb. 2002.

26. M. Amer, M. Goldstein, and S. Abdennadher, "Enhancing one-class support vector machines for unsupervised anomaly detection," in *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, ODD '13, (New York, NY, USA), pp. 8–15, Association for Computing Machinery, Aug. 2013.
27. F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386–408, Nov. 1958.
28. S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, Oct. 2016.
29. T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, pp. 21–27, Jan. 1967.
30. D. Wettschereck and T. G. Dietterich, "Locally adaptive nearest neighbor algorithms," in *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, (San Francisco, CA, USA), pp. 184–191, Morgan Kaufmann Publishers Inc., Nov. 1993.
31. R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu, "Diagnosis of multiple cancer types by shrunken centroids of gene expression," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, pp. 6567–6572, May 2002.