Rule Extraction in Unsupervised Anomaly Detection for Model Explainability: Application to OneClass SVM

Alberto Barbado[1,2], **Oscar Corcho**[2]
alberto.barbadogonzalez@telefonica.com, ocorcho@fi.upm.es
[1] Telefónica, 28050 Madrid, Spain
[2] Universidad Politécnica de Madrid, Departamento de Inteligencia Artificial, 28660 Boadilla del Monte, Spain

Abstract

OneClass SVM is a popular method for unsupervised anomaly detection. As many other methods, it suffers from the *black box* problem: it is difficult to justify, in an intuitive and simple manner, why the decision frontier is identifying data points as anomalous or non anomalous. Such type of problem is being widely addressed for supervised models. However, it is still an uncharted area for unsupervised learning. In this paper, we describe a method to infer rules that justify why a point is labelled as an anomaly, so as to obtain intuitive explanations for models created using the OneClass SVM algorithm. We evaluate our proposal with different datasets, including real-world data coming from industry. With this, our proposal contributes to extend Explainable AI techniques to unsupervised machine learning models.

Keywords XAI, OneClass SVM, unsupervised learning, rule extraction

Introduction

Responsible Artificial Intelligence [1] is an emerging discipline that is gaining relevance both in academic and industrial research. An increasing number of organisations are defining policies and criteria for the usage of data and the development of support decision systems using AI techniques.

For example, Telefónica has defined its own AI principles [2], which can be organised into the following categories:

- Detecting sensitive data in the datasets used to train Machine Learning (ML) models (for example, the use of gender information to support the decision of giving a credit score) or detecting whether there is a bias in the data, what may have an unwanted effect in decision making. Besides analysing directly the dataset, there are also methods to perform an evaluation on a trained model to check whether there is any bias on its decisions. This is done using a group of metrics known as fairness metrics [3, 4, 5].
- Explaining how an algorithm reaches a conclusion in a way that is clear and intuitive for a human being. This is crucial not only to avoid the fear of considering AI as black boxes that may suddenly take harmful decisions in the future, but also to contribute to the democratisation of AI and increase trust on these systems.

The first group of challenges is being widely addressed by researchers, with tools to audit datasets and trained models to detect risks, and at the same time provide solutions to mitigate those biases [6, 7]. The second group is addressed through the use of Explainable AI (XAI) techniques, which generate post-hoc explanations based on the information provided by the black box model. In the literature, there are many XAI proposals for supervised ML models. However, some of the most recent and thorough reviews on XAI [8, 9, 1, 10] do not mention the application of such techniques to unsupervised learning.

In this paper we focus on unsupervised ML models used for anomaly detection. Our main contribution is a novel algorithmic solution to generate explanations for the particular case of anomaly detection using OneClass SVM (OCSVM) algorithms. Such explanations will be obtained using rule extraction techniques.

We perform an empirical validation of the results of our proposal using open datasets as well as real data from Telefónica. Our evaluation consists in analysing whether the number of rules for the non-anomalous data points are inferior to the ones obtained for the anomalous data points. We also compare the rules extracted with our algorithm with those extracted with a surrogate Decision Tree (DT) trained over the features and outputs of the unsupervised anomaly detection model.

The rest of the paper is organized as follows. First, we describe some related work in the area of XAI and rule extraction applied to SVM. After identifying research opportunities derived from those works, the paper introduces the algorithm implementation for OCSVM. Following this, we present an empirical evaluation of our algorithm with several datasets. We then conclude, showing also potential future research lines of work.

Related Work

This section reviews unsupervised ML models used for anomaly detection, and reviews previous work on rule extraction in SVM that is relevant for our proposal.

Unsupervised ML for Anomaly Detection

Many algorithms for unsupervised anomaly detection exist. Examples are IsolationForest [11], Local anomaly Factor (LOF) [12] and OCSVM [13]. The latter has relevant advantages over the former ones, mainly in terms of computational performance. This is due to the fact that it creates a decision frontier using only the support vectors (like general supervised SVM) and that model training always leads to the same solution because the optimization problem is a convex one. However, SVM (hence OCSVM) algorithms are some of the most difficult ones to explain due to the mathematically complex method that obtains the decision frontier.

SVM for classification theoretically maps the data points available in the dataset to a higher dimensional space than the one determined by their features, so that the separation among classes may be done linearly. It uses a hyperplane obtained from data points from all of the classes. These data points, known as support vectors, are the ones that are closer to each other and the only ones needed to determine the decision frontier. However, it is not really necessary to map to a higher dimension due to the fact that the equation that appears in the optimization of the algorithm uses a dot product of those mapped points. Because of that, the only thing to be calculated is such dot product, something that can be accomplished with the well-known kernel trick. Hence instead of calculating explicitly the mapping to a higher dimension the equation is solved using a kernel function.

In OCSVM there are no labels. Hence all data points are considered to belong to a same class at the beginning. The decision frontier is computed trying to separate the region of the hyperspace with a higher number of data points close to each other from another that has small density, considering those points as anomalies. To do so the algorithm tries to define a decision frontier that maximizes the distance to the origin of the hyperspace and that at the same time separates from it the maximum number of data points. This compromise between those factors leads to the optimization of the algorithm and allows obtaining the optimal decision frontier. Those data points that are separated are labeled as non-anomalous (+1) and the others are labeled as anomalous (-1).

The optimization problem is reflected in the following equations:

$$min_{w,\xi_i,\rho} = \frac{1}{2}||w||^2 + \frac{1}{\nu n}\sum_{i=1}^{n}(\xi_i - \rho)$$
$$\text{subject to:} \tag{1}$$
$$(w, \phi(x_i)) \geq \rho - \xi_i \;\; for\, i = 1, ..., n$$
$$\xi_i \geq 0 \;\; for\, i = 1, ..., n$$

In that equation, ν is a hyper-parameter known as *rejection rate*, which needs to be selected by the user. It sets an upper bound on the fraction of anomalies that can be considered, and also defines a lower bound on the fraction of support vectors that can be considered. Using Lagrange techniques, the decision frontier obtained is the following one:

$$f(x) = sgn((w, \phi(x_i) - \rho) \Rightarrow$$
$$f(x) = sgn(\sum_{i=i}^{n}\alpha_i K(x_i, x) - \rho) \tag{2}$$

Hence the hyper-parameters that must be defined in this method are the rejection rate, ν, and the type of kernel used.

Rule Extraction in SVM

Several papers deal with the importance of XAI applied to models such as SVM. In particular [14] aims to resolve the black box problem in SVM for supervised classification tasks. It obtains a set of rules that explain in a simple manner the boundaries that contain the values of the different classes. Thanks to that, it is easier to understand what are the conditions that will identify a data point as belonging to one class or to another (in OCSVM it would be belonging to class +1 - normal data - or class -1 - anomaly -). The challenge consists in discovering a way to map the algorithm results to a particular set of rules. There are two general ways to do it. One consists in inferring the rules directly from the decision frontier, using a decompositional rule extraction technique. The other (simpler) one uses a method called pedagogical rule extraction technique that does not care about the decision frontier itself and considers the algorithm as a black box from where to extract those rules depending on which class it uses to classify different relevant data points used as an input.

The first method is clearly more transparent, as it deals directly with the inner structure of the model. However, it is generally more difficult to implement. One way to implement it is with a technique known as SVM+ Prototypes [15], which consists in finding hypercubes using the centroids (or prototypes) of data points of each class and using as vertices the data points from that hyperspace area farther away from that centroid (or use directly the support vectors themselves if they are available). It will then infer a rule from the values of the vertices of the hypercube that contain the limits of all the points inside it, creating one rule for each hypercube.

For example, a dataset that contains two numerical features X and Y will be defined in a 2-dimensional space. The algorithm will create a square that contains the data points on each of the classes, as shown in Figure 1. The rule that justifies that a data point belongs to class 2 is:

- Rule 1: CLASS 2 IF X≥X1 ∧ Y≥Y1 ∧ X≤X2 ∧ Y≤Y2

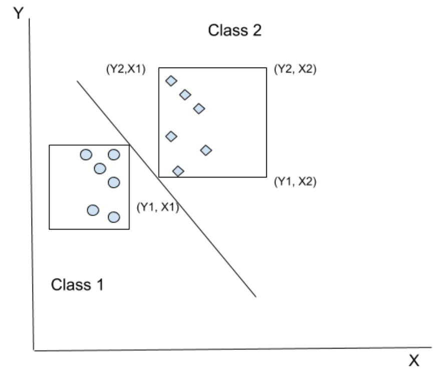


Figure 1: SVM with linear kernel classifying data points of two classes.

The generated hypercubes may wrongly include points from the other class when the decision frontier is not linear or spherical, as shown in Figure 2. In this case the algorithm considers an additional number of clusters trying to include the points into a smaller hypercube, as shown in Figure 3.

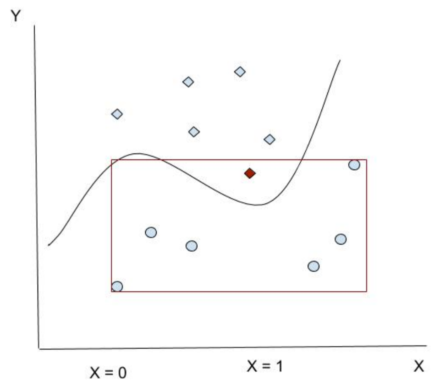


Figure 2: A hypercube generated using the farthest points leads to the wrong inclusion of data from the another class.

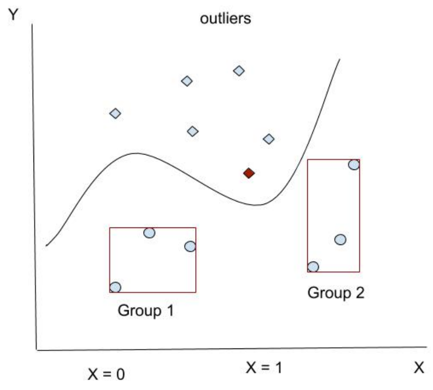


Figure 3: Using more hypercubes avoids the aforementioned problem. Now there is no wrong inclusion of data points from another class.

A rule will be generated for each hypercube, considering all those scenarios as independent, leading to this output:

- Group 1: CLASS 1 IF X...
- Group 2: CLASS 1 IF X...

There are some downsides of that method in supervised classification tasks, especially when the problem is not simply a binary classification or when the algorithm is performing a regression. For instance, the number of rules may grow immensely due to the fact that a set of rules will be generated for each category and each set may contain a huge number of rule groups, leading to an incomprehensible output.

However, in OCSVM these difficulties may be potentially mitigated due to two reasons. On the one hand, the explanations are reduced to rules that explain when a data point is not an anomaly (so there would be no need to define rules for the anomalies). On the other hand, the algorithm tries to group all non-anomalous points together, setting them apart from the outliers. Because of this, the chance to define a hypercube that does not contain a point from the another class may be higher than in a standard classification task. Both the unbalanced inherent nature of data points in anomaly detection (few anomalies vs. many more non-anomalous data points) and the fact that non-anomalous points tend to be closer to each other may help achieving good results with this method.

Method

We first describe the intuition behind our rule extraction approach from an OCSVM model for anomaly detection. Then, we describe in detail the algorithm implementation.

Algorithm Intuition

We propose using rule extraction techniques within OCSVM models for anomaly detection, by generating hypercubes that encapsulate the non-anomalous data points, and using their vertices as rules that explain when a data point is considered non-anomalous. Our method has the following characteristics, according to the taxonomy for XAI in [10]:

- Post-hoc: Explainability is achieved using external techniques.
- Global and individual: Explanations serve to explain how the whole model works, as well as why a specific data point is considered anomalous or non-anomalous.
- Model-agnostic: As with other techniques for global explanations [10], the only information needed to build the explanations are the input features and the outcomes of the system after fitting the model.
- Counterfactual: The explanations for why a data point is anomalous also include information on the changes that should take place in the feature values in order to consider that data point as non-anomalous.

Since explanations are based on hypercubes, the OCSVM kernel to be used is the Radial Basis Function (RBF), illustrated in Figure 4.

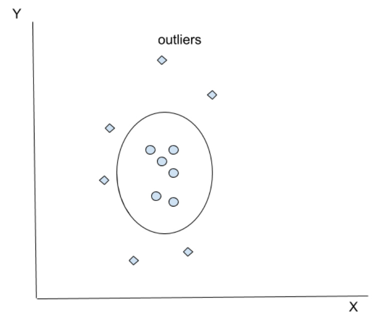


Figure 4: With an RBF Kernel the correct hypercube will be the one that encloses the points that are not anomalies, since the OCSVM algorithm will try to enclose most of the points inside the decision frontier and leave anomalies outside.

The dimensionality of the hyperspace will correspond to the different numerical features used for fitting the model. However, a caveat to be considered is when some features are categorical non ordinal variables. In that case the approximation would be to extract a rule for each of the possible combinations of categorical values among the data points that are not considered anomalous. Considering again the aforementioned 2-dimensional example, with variable X being binary categorical, a dataset may look like in Figure 5:

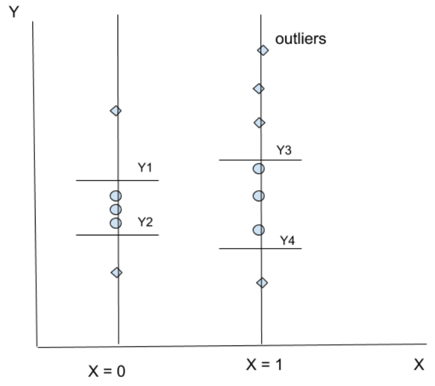


Figure 5: Rule extraction with a categorical variable.

In that case, two rules would be extracted, one for each of the possible states of X:

- Rule 1: NOT OUTLIER IF X = 0 $\wedge$ Y $\geq$ Y2 $\wedge$ Y $\leq$ Y1
- Rule 2: NOT OUTLIER IF X = 1 $\wedge$ Y $\geq$ Y4 $\wedge$ Y $\leq$ Y3

Generally speaking, the algorithm logic can be summarised as:

- Apply OCSVM to the dataset to create the model.
- Depending on the characteristics of variables, do:
  - Case 1. Numerical only: Iteratively create clusters in the non-anomalous data (starting with one cluster) and create a hypercube using the centroid and the points further away from it. Check whether the hypercube contains any data point from the anomalous group; if it does, repeat using one more cluster than before. End when no anomalies are contained in the generated hypercubes. If there are anomalies and the data points in a cluster are inferior to the number of vertices needed for the hypercube, complete the missing vertices with artificial datapoints and end when there are no anomalies or when the convergence criterion is reached.
  - Case 2. Categorical only: The rules will correspond directly to the different value states contained in the dataset of non-anomalous points.
  - Case 3. Both numerical and categorical. This case would be analogous to Case 1, but data points will be filtered for each of the combinations of the categorical variables states. For each combination, there will be a set of rules for the numerical features.
- Use these vertices to obtain the boundaries of that hypercube and directly extract rules from them.

Algorithm Description

Algorithm 1 contains the proposal for rule extraction for an OCSVM model that may be applied over a dataset with either categorical or numerical variables (or both). *ocsvm_rule_extractor* is the main function of the algorithm. Regarding input parameters, X is the input data frame with the features, l_n is a list with the numerical columns, l_c is a list with the categorical columns, d is a dictionary with the hyperparameters for OCSVM (since it is an RBF kernel, the hyperparameters are the values for the upper bound on the fraction of training errors and a lower bound of the fraction of support vectors, ν, and the kernel coefficient, γ). This function starts with the feature scaling of the numerical features (function *featureScaling*). After that, it fits an OCSVM model with all the data available and detects the anomalies within it, generating two datasets, X_y with the anomalous data points and X_n with the rest (function *filterAnomalies*). The next step is assessing that the number of data points within the decision frontier are enough to build the hypercubes. In order to check this, the dimension of the hypercube should be lower than the number of non-anomalous data points for that categorical state. The dimension of that hypercube is computed using the number of numerical features for each combination of categorical states; for each categorical state it will be $2^{len(l_n)}$, where $len(l_n)$ is the length of a list containing all the numerical columns names.

Next, the algorithm checks whether features are numerical, categorical or both. In case of only numerical columns, it calls function *getRules*, described in Algorithm 2. *getRules* receives as input a matrix with anomalous and non-anomalous data points, X_y and X_n respectively, and the number of vertices for each hypercube, n_v, based on the number of numerical features. It also receives the list of names of those numerical features, l_n. The main purpose of this function is to cluster non-anomalous data points in a set of hypercubes that do not contain any anomalous data points. To do that, it iteratively increases the number of clusters (hypercubes) until there are no anomalous points within any hypercube. The function *outPosition* checks whether the rules defined based on the vertices of the hypercube do not include any data point from the anomalous subset, X_y.

getRules calls function *getVertices* with a specific number of clusters, n_{cl}. This function then performs the clustering over the non-anomalous data points, X_n, using the function *getClusters* that returns the label of the cluster for each data point, as well as the centroid position for each cluster using the K-means++ algorithm [16]. Then, it iterates through each cluster and first obtains the subset of data points for that cluster X_{nc} with the function $insideCluster$. After that, if there are enough data points in that cluster (more data points than the vertices of the hypercube) it computes the distance of each of them to the centroid with *computeDistance* and uses the furthest n_v as vertices.

If there are not enough data points in that cluster (less than the number of vertices for the hypercube), all of them are used limits and the missing vertices are artificially generated to close the hypercube (leaving outside all anomalous data points).

Algorithm 1 Rule Extractor algorithm for OneClassSVM - Main

```
1:  procedure OCSVM_RULE_EXTRACTOR($X, l_n, l_c, d$)
2:      for $c \in l_c$ do
3:          $X[:, c] \leftarrow featureScaling(X[:, c])$
4:      end for
5:      $model \leftarrow OneClassSVM(d)$
6:      $model.fit(X)$
7:      $preds \leftarrow model.train(X)$
8:      $distances \leftarrow model.decisionFunction(X)$
9:      $X_y, X_n \leftarrow filterAnomalies(X, preds)$
10:     $n_v \leftarrow 2^{len(l_n)}$
11:     if $(len(l_c) + 1) \times n_v > X_n.rows$ then
12:         return $None$
13:     else
14:         continue
15:     end if
16:     if $len(l_1) = 0$ then
17:         $rules \leftarrow getRules(X_n, X_y, X, n_v, l_n)$
18:     else if $len(l_2) = 0$ then
19:         $rules \leftarrow getUnique(X_n, l_c)$
20:     else
21:         $categories \leftarrow getUnique(X_n, l_c)$
22:         $rules$ empty list
23:         for $c \in categories$ do
24:             $X_{nf}, X_{yf} \leftarrow filterCategory(X_n, X_y, c)$
25:             $rules.append(getRules(X_{nf}, X_{ny}, n_v, l_n))$
26:         end for
27:     end if
28:     $rules \leftarrow featureUnscaling(rules, l_n)$
29:     $rules \leftarrow pruneRules(rules, l_n, l_c)$
30:     return $rules$
31: end procedure
```

If all the features are categorical, then the rules for non-anomalous data points will simply be the unique combination of values for them. If there are both categorical and numerical features, the algorithm obtains the hypercubes (as mentioned for numerical features only) for the subset of data points associated to each combination of categorical values.

After obtaining the rules, function $featureUnscaling$ is used to express rules in their original values (not the scaled ones used for the ML models). And function $pruneRules$ checks whether there are rules that may be included inside others; that is, for each rule it checkes whether there is another with a bigger scope that will include it as a subset case.

There is a convergence criteria, applied in the scenario where a cluster has less data points than the number of vertices. In this case, if there are anomalies within the hypercube, the algorithm checks the number of data points within the cluster X_{nc} versus a threshold defined by the number of vertices n_{cl} multiplied by a reference value e. If there are less data points than this threshold, then that cluster is discarded and not considered for rule generation. Since the proposal aims to explain what makes data points non-anomalous and what should have happened to label a data point as non-anomalous, this approximation is feasible.

Algorithm 2 Rule Extractor algorithm for OneClassSVM - Additional functions

```
1:  procedure GETRULES($X_n, X_y, n_v, l_n$)
2:      $rules$ empty list
3:      $e$ reference value
4:      $check \leftarrow True$
5:      $n_{clusters} \leftarrow 0$
6:      while check do
7:          $n_{cl} \leftarrow n_{cl} + 1$
8:          $vInfo \leftarrow getVertices(X_n, X, n_v, l_n, n_{cl})$
9:          for $iterValue \in vInfo$ do
10:             $rules_{cluster} \leftarrow iterValue[0]$
11:             $X_{nc} \leftarrow iterValue[1]$
12:             $rules.append(rules_{cluster})$
13:             if $len(rules_{cluster}) = len(X_n)$ then
14:                 if $X_{nc} > e \times n_{cl}$ then
15:                     $check \leftarrow True$
16:                     $break$
17:                 else
18:                     $check \leftarrow False$
19:                 end if
20:             else
21:                 $l_y \leftarrow outPosition(rules_{cluster}, X_y)$
22:                 if $l_y = 0$ then
23:                     $check \leftarrow False$
24:                 else
25:                     $check \leftarrow True$
26:                     $break$
27:                 end if
28:             end if
29:         end for
30:     end while
31:     return $rules$
32: end procedure

33: procedure GETVERTICES($X_n, n_v, l_n, n_{cl}$)
34:     $d_{bounds}$ empty list
35:     $d_{points}$ empty list
36:     $labels, centroids \leftarrow getClusters(X_n, l_n, n_{cl})$
37:     for $c \in n_{cl}$ do
38:         $X_{nc} \leftarrow insideCluster(labels, X_n)$
39:         if $len(X_{nc}) > n_v$ then
40:             $vertices \leftarrow computeDist(X_{nc}, labels[c])$
41:         else
42:             $vertices \leftarrow X_n$
43:         end if
44:         $d_{bounds}.append(vertices)$
45:         $d_{points}.append(X_{nc})$
46:     end for
47:     return $d_{bounds}, d_{points}$
48: end procedure
```

Evaluation

We use our algorithm over different datasets (both public and from Telefonica's real data), to evaluate the following hypotheses:

- The OCSVM algorithm with an RBF decision frontier groups non-anomalous points within a hypersphere. Hence the number of rules extracted using non-anomalous data will be fewer than using anomalous ones.

- Our algorithm yields a number of rules similar to other proposals for global model-agnostic explanations, such as Surrogate Trees.

In our experiments we will count the number of rules extracted in different scenarios, and check whether there are significative differences. Several experiments are conducted over different datasets:

- Rule extraction over non-anomalous data points with our algorithm.
- Rule extraction over anomalous data points with our algorithm. In this scenario, there is no convergence criteria since the number of anomalous data points will be very low (many times lower than the number of vertices).
- Rule extraction over all the data points using a DT overfitted in the dataset, using the features as input and the anomalous/non-anomalous labels as target variable. This yields two types of rules, the ones that explain the anomalous data points, and the ones that explain the non-anomalous ones.

The used datasets belong to different domains, have different sizes and different number of features (both categorical and numerical). They are indicated in Table 1:

- Datasets 1 and 2 about seismic activity [17]. Dataset 1 is bi-dimensional with only numerical features ('gdenergy', 'gdpuls'). Dataset 2 has 2 categorical features ('hazard', 'shift') and 7 numerical ('seismoacoustic', 'shift', 'genergy', 'gplus', 'gdenergy', 'gdpuls', 'hazard', 'bumps', 'bumps2').
- Dataset 3 from a call center at Telefónica. It is real data that includes the total number of calls received in one of its services during every hour. Using these data, some features are extracted (weekday), and they are cyclically transformed, so that each time feature turns into two features for the sine and cosine components. The rules in this case are also transformed back into the original features in order to enhance rule comprehension.
- Dataset 4 about cardiovascular diseases [18]. There are 4 categorical features ('smoke', 'alco', 'active', 'is_man') and 7 numerical ('age', 'height', 'weight', 'ap_hi', 'ap_lo','cholesterol','gluc').
- Datasets 5 and 6 on the US census for year 1990 [19]. Dataset 5 has 2 categorical features ('dAncstry1_3', 'dAncstry1_4') and 4 numerical ones ('dAge', 'iYearsch', 'iYearkwrk', 'dYrsserv'). Dataset 6 has the same numerical features, but 18 categorical ones (dAncstry_i_j with i that ranges fro 1 to 2, and j that ranges from 3 to 11 if i equals 1, and 2 to 10 if i equals 2.)

We ran experiments with the following infrastructure: the implementations of the OCSVM algorithm, the K-Means++ clustering and the DT algorithms are based on Scikit-Learn [20]. The rest of the code described in Algorithms 1 and 2 were developed from scratch, and available in Github [21].

Dataset	Ref.	Nº Cat.	Nº Num.	Nº Rows
1	[17]	0	2	669
2	[17]	2	7	1705
3	Telefónica	0	5	2712
4	[18]	4	7	42000
5	[19]	2	4	100000
6	[19]	18	4	200000

Table 1: Description of each dataset, with their reference (Ref.), categorical features (Nº Cat.), numerical features (Nº Num) and number of rows.

OCSVM models use as hyperparameters: $\nu = 0.1, \gamma = 0.1, kernel = rbf$. K-means++ models use $max\_iter = 100, n\_init = 10, random\_state = 0$. The DT hyperparameters are dynamically set for each dataset, using the Gini criterion to find the best splits, and using $random_state = 42$ as seed. For each dataset, the DT is as follows:

- 1: Depth = 11, Nodes = 53, Leaf nodes = 30
- 2: Depth = 17, Nodes = 129, Leaf nodes = 69
- 3: Depth = 13, Nodes = 159, Leaf nodes = 91
- 4: Depth = 27, Nodes = 2265, Leaf nodes = 1184
- 5: Depth = 17, Nodes = 353, Leaf nodes = 162
- 6: Depth = 30, Nodes = 1843, Leaf nodes = 1051

Results are detailed in Table 2. It shows the number of rules extracted for all datasets with the different algorithms.

Dataset	Prop. [NA]	Prop. [A]	DT [NA]	DT [A]
1	89	25	11	14
2	33	20	30	93
3	151	37	36	72
4	4754	45	171	63
5	249	185	62	26
6	762	762	86	11

Table 2: Number of rules extracted using our algorithm (Prop.) or the surrogate DT over anomalous (A) and non-anomalous (NA) data points.

Hypothesis 1 is not validated. Table 2 shows that the number of rules for anomalous data points is always inferior to those for non-anomalous ones. This may be due to the fact that the pruning algorithm is not good enough yet for rule reduction. For instance, one rule is removed if another wider rule includes it. However, many times the rules are not included within only one wider rule, but within many wider rules all together. This can be visually seen using dataset 1. Figures 6 and7 show the rules obtained before and after applying the pruning algorithm. Although the number of rules is reduced, many rules may still be removed. This affects more the rules obtained with non-anomalous data points since they are closer than the ones obtained with anomalous ones, as shown in Figure 8 (here, points without a visible hypercube are due to the fact that all the vertices are the same; thus, it is the same point as the anomaly).

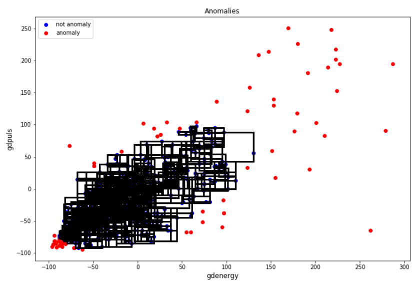


Figure 6: Hypercubes in a 2D space for non-anomalous data (before pruning).

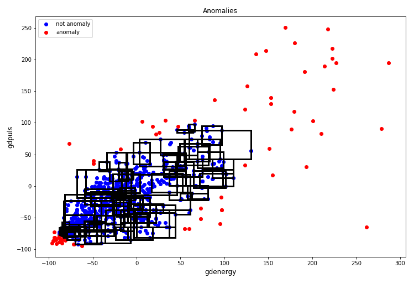


Figure 7: Hypercubes in a 2D space for non-anomalous data (after pruning).

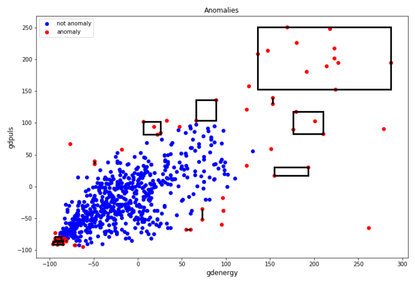


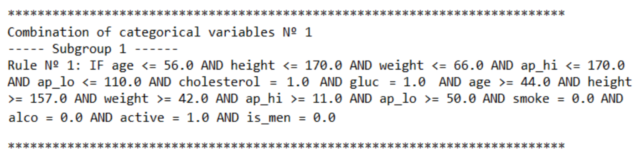
Figure 8: Hypercubes in a 2D space for anomalous data (after pruning).

Regarding hypothesis 2, in most of the cases DT generates less rules than our algorithm (with the exception of anomalous data points in dataset 4). However, results may be considered similar.

Figures 9 and 10 show a sample rule over the same dataset for our algorithm and for the DT surrogate method, respectively. The DT method uses symbols for equality and inequalities, while in our method all the extracted rules use only $\geq$ and $\leq$, and not $>$ or $<$.

Limitations of our Approach

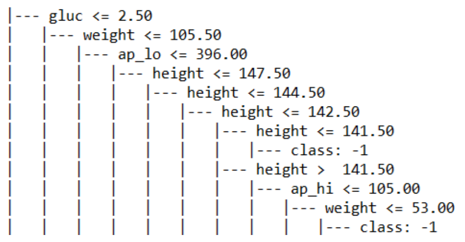
Our method can only be used if there is a minimum number of data points (at least one point per vertex for the hypercube). This limitation may be avoided by generating artificial vertices, as done for clusters with few data points.



```
*****************************************************************************
Combination of categorical variables Nº 1
----- Subgroup 1 ------
Rule Nº 1: IF age <= 56.0 AND height <= 170.0 AND weight <= 66.0 AND ap_hi <= 170.0
AND ap_lo <= 110.0 AND cholesterol = 1.0 AND gluc = 1.0  AND age >= 44.0 AND height
>= 157.0 AND weight >= 42.0 AND ap_hi >= 11.0 AND ap_lo >= 50.0 AND smoke = 0.0 AND
alco = 0.0 AND active = 1.0 AND is_men = 0.0

*****************************************************************************
```

Figure 9: A sample rule for non-anomalous data points for dataset 4 using our proposal



```
|--- gluc <= 2.50
|   |--- weight <= 105.50
|   |   |--- ap_lo <= 396.00
|   |   |   |--- height <= 147.50
|   |   |   |   |--- height <= 144.50
|   |   |   |   |   |--- height <= 142.50
|   |   |   |   |   |   |--- height <= 141.50
|   |   |   |   |   |   |   |--- class: -1
|   |   |   |   |   |   |--- height >  141.50
|   |   |   |   |   |   |   |--- ap_hi <= 105.00
|   |   |   |   |   |   |   |   |--- weight <= 53.00
|   |   |   |   |   |   |   |   |   |--- class: -1
```

Figure 10: A sample rule for non-anomalous data points for dataset 4 using the DT surrogate method

Conclusion

We described a novel implementation of a rule extraction technique for model-agnostic, both global and local, and counterfactual explanations for unsupervised learning in the scenario of anomaly detection, using the OCSVM algorithm.

We applied our algorithm to open and private datasets, with anomalous and non-anomalous data, with the initially unexpected result that using anomalous data points yields less rules than using the non-anomalous ones. We also compared the results of our algorithm with other surrogate methods such as using a DT model trained over the same features and using as target variable the anomalous or non-anomalous labels. In most scenarios both methods yielded similar results, thus showing that our approach is a feasible technique for XAI for anomaly detection.

Future Work

Our rule extraction method can be optimised by using different clustering algorithms, so as to analyse whether there is a reduction in the number or rules that are extracted. It will be interesting to compare cluster methods that consider together both categorical non ordinal as well as numerical features, instead of dividing the hyperspace according to the categorical value combinations, as we did in our work. One example of such algorithm is K-modes [22]. Besides, when there are clusters with anomalies, instead of increasing the number of clusters and apply it over the whole dataset, the algorithm may separate data points that are already inside a cluster without anomalies and apply a different clustering algorithm to partition the subspace that had anomalies. It will be interesting to compare the number of rules extracted with this approach versus the one in this paper.

Rule extraction should also be designed to consider all types of comparisons ($\geq$, $\leq$, $>$ and $<$), and more sophisticated pruning techniques should be applied to improve rule reduction, since a rule may be contained in two or more rules together.

References

[1] Alejandro Barredo Arrieta et al. *Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI*. 2019. https://arxiv.org/abs/1910.10045. arXiv: 1910.10045 [cs.AI].

[2] Richard Benjamins, Alberto Barbado, and Daniel Sierra. *Responsible AI by Design*. 2019. https://arxiv.org/abs/1909.12838. arXiv: 1909.12838 [cs.CY].

[3] Till Speicher et al. "A Unified Approach to Quantifying Algorithmic Unfairness". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '18* (2018). DOI: 10.1145/3219819.3220046. URL: http://dx.doi.org/10.1145/3219819.3220046.

[4] Moritz Hardt, Eric Price, and Nathan Srebro. "Equality of Opportunity in Supervised Learning". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 3323–3331. ISBN: 978-1-5108-3881-9. URL: http://dl.acm.org/citation.cfm?id=3157382.3157469.

[5] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. *Mitigating Unwanted Biases with Adversarial Learning*. 2018. https://arxiv.org/abs/1801.07593. arXiv: 1801.07593 [cs.LG].

[6] Rachel KE Bellamy et al. "AI fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias". In: *arXiv preprint arXiv:1810.01943* (2018).

[7] Pedro Saleiro et al. "Aequitas: A Bias and Fairness Audit Toolkit". In: *arXiv preprint arXiv:1811.05577* (2018).

[8] Leilani H Gilpin et al. "Explaining explanations: An overview of interpretability of machine learning". In: *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE. 2018, pp. 80–89.

[9] Shane T Mueller et al. "Explanation in Human-AI Systems: A Literature Meta-Review, Synopsis of Key Ideas and Publications, and Bibliography for Explainable AI". In: *arXiv preprint arXiv:1902.01876* (2019).

[10] Christoph Molnar. *Interpretable machine learning*. Lulu.com, 2019. https://christophm.github.io/interpretable-ml-book/.

[11] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest". In: *2008 Eighth IEEE International Conference on Data Mining*. IEEE. 2008, pp. 413–422.

[12] Markus M Breunig et al. "LOF: identifying density-based local outliers". In: *ACM sigmod record*. Vol. 29. 2. ACM. 2000, pp. 93–104.

[13] Bernhard Schölkopf et al. "Support vector method for novelty detection". In: *Advances in neural information processing systems*. 2000, pp. 582–588.

[14] David Martens et al. "Rule extraction from support vector machines: an overview of issues and application in credit scoring". In: *Rule extraction from support vector machines*. Springer, 2008, pp. 33–63.

[15] Haydemar Núñez, Cecilio Angulo, and Andreu Català. "Rule extraction from support vector machines". In: *Esann*. 2002, pp. 107–112.

[16] David Arthur and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2007, pp. 1027–1035.

[17] Saket Sathe and Charu Aggarwal. "LODES: Local density meets spectral outlier detection". In: *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM. 2016, pp. 171–179.

[18] Meghana Padmanabhan et al. "Physician-friendly machine learning: A case study with cardiovascular disease risk prediction". In: *Journal of clinical medicine* 8.7 (2019), p. 1050.

[19] Catherine Blake. "UCI repository of machine learning databases". In: *http://www. ics. uci. edu/~ mlearn/MLRepository. html* (1998).

[20] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[21] Alberto Barbado. *Rule extraction in unsupervised outlier detection for XAI*. https://github.com/AlbertoBarbado/unsupervised-outlier-transparency. 2019.

[22] Anil Chaturvedi, Paul E Green, and J Douglas Caroll. "K-modes clustering". In: *Journal of classification* 18.1 (2001), pp. 35–55.