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

Alberto Barbado\textsuperscript{1,2}, Oscar Corcho\textsuperscript{2}, Richard Benjamins\textsuperscript{1}

\texttt{alberto.barbadogonzalez@telefonica.com, ocorcho@fi.upm.es, richard.benjamins@telefonica.com}

\textsuperscript{1} Telefónica, 28050 Madrid, Spain
\textsuperscript{2} Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, 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 evaluate several rule extraction techniques over OneClass SVM models, as well as present alternative designs for some of those algorithms. Together with that, we propose algorithms to compute metrics related with eXplainable Artificial Intelligence (XAI) regarding the "comprehensibility", "representativeness", "stability" and "diversity" of the extracted rules. We evaluate our proposals with different datasets, including real-world data coming from industry. With this, our proposal contributes to extend XAI techniques to unsupervised machine learning models.

Keywords XAI, OneClass SVM, unsupervised learning, rule extraction, anomaly detection, metrics

1. Introduction

Responsible Artificial Intelligence (RAI) is defined as the different AI principles that should be considered when developing and deploying real applications based on AI\textsuperscript{[1]}. RAI serves as a methodological framework to both identify core aspects (or AI principles) that should be considered when developing AI solutions while also proposing how to implement them. These AI principles include aspects such as Fairness, Explainability, Security, Privacy and Human-centric design\textsuperscript{[2]}. The AI principle of Explainability is addressed through the use of Explainable AI (XAI) techniques, which can be applied to black-box models in order to obtain post-hoc explanations based on the information that they provide. In the literature, there are many XAI proposals for supervised ML models. However, some of the most recent and thorough reviews on XAI\textsuperscript{[3,4,5]} do not mention many applications of those techniques to unsupervised learning.

Outlier detection is one of the tasks where unsupervised learning is applied. It is defined as the process of detecting anomalous observations within a dataset, and sometimes remove it as a first step within data-mining applications\textsuperscript{[6]}. There is often no prior information about outliers in a dataset, hence unsupervised ML algorithms offer the chance to infer patterns and detect anomalies. However, not only is it important to detect outliers, but also to explain them. Explanations can help to understand why a particular datapoint has been labelled anomalous (and what changes in the feature values would lead to classify it as an inlier), and how the model behaves globally (for instance, what features influence more for classifying a datapoint as an outlier).

The output of an unsupervised ML model for anomaly detection can be seen as binary (an observation may be an "outlier" or an "inlier"). Thus, surrogate post-hoc XAI techniques can yield explanations similarly to a supervised binary classifier where the two possible outputs are imbalanced. Hence, the explanations for the model can be obtained by using XAI techniques already designed for supervised ML binary classifiers. This is already addressed in the literature, particularly by using feature-relevance XAI techniques\textsuperscript{[7,8]}.

Among the different model-agnostic post-hoc XAI techniques that can be applied, rule extraction offers the possibility to provide both global and local explanations, as indicated by the recent literature\textsuperscript{[1]}. This is achieved by using an "IF...THEN" schema that explains both the output of a particular datapoint as well as the global behaviour of the original model. In the case of outlier detection, they can explain both a particular outlier and also how the features of the whole model contribute to identify points as outliers or inliers. Even though there are some examples of this in the literature, particularly for the case of OCSVM\textsuperscript{[9]}, there are not many studies covering it to the best of our knowledge.

There is a particularity of the usage of rule extraction for explaining anomalies. An outlier detection system that uses rules as explanations may have more interest in explaining faithfully why a datapoint is an outlier, and what should have happened in order for it to be an inlier, rather than being able to cover all possible scenarios with explanations that may be wrong. This means that the extracted rules need to have a 100% precision (P@1); rules that classify datapoints...
from one class (i.e. "outliers") without including datapoints from the other one. Considering the example of rules extracted that cover inliers, this is important because the counterfactual explanation for how to turn an outlier into an inlier should lead to a scenario where the model will always classify it as an inlier.

This is linked to another aspect regarding XAI. Even though there are many model-agnostic post-hoc rule extraction techniques that can be used for explaining a ML model in general, and an unsupervised one for anomaly detection in particular, there is still one question present: which technique provides the best explanations?. This leads to an open issue within the XAI literature: how to evaluate the quality of explanations?. Here, the literature suggests some concepts to consider while designing new metrics and algorithms. The metrics need to consider the type of explanations provided (rule based in this case) and the type of data used. For instance, in our case we work on anomaly detection without a prior ground truth. Hence, some XAI metrics (like those related to accuracy measurement of the rule predictions over a test set) are not applicable. Together with that, other particularities of the problems addressed may influence in what metrics are more important. In our case, since we are using P@1 rules, measuring the fidelity of the explanations is not necessary (since the comparison will only be possible against the model output). However, other metrics gain more relevance, such as stability. With that, some relevant aspects to measure for this case are:

• "Comprehensibility": Are explanations easy enough to understand?
• "Representativeness": Are explanations relevant? Do they explain all possible cases?
• "Stability": Do explanations match the predictions of the model? Or are there inconsistencies?
• "Diversity": Are explanations sufficiently different among them? Or are they redundant?

There are only a few implementations of these concepts as metrics to measure and compare rule extraction techniques (and even though, mainly for rule extraction techniques over supervised ML).

Following this, the main contributions of our work are:

• Applying different model-agnostic rule extraction techniques to explain the anomalies detected by an unsupervised OCSVM ML model through P@1 rules and for different types of kernels (RBF and Linear). We use existing rule extraction techniques and propose some variations over the one described in [10].
• Quantifying the quality of the generated P@1 rule explanations for unsupervised anomaly detection with OCSVM using XAI metrics that measure comprehensibility, representativeness, stability and diversity. Particularly, we propose novel ways to implement stability and diversity.
• Evaluating how the comprehensibility aspect regarding the number of P@1 rules generated significantly varies depending on the kernel considered (Linear vs RBF) and depending on whether the explanations are for outliers or for inliers.

The empirical evaluations carried out use both open datasets as well as real data from Telefónica. Our evaluation consists in analysing the results for the aforementioned metrics using different rule extraction algorithms over OCSVM models with two kernel configurations, Linear and RBF.

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. This chapter will also introduce the rule extraction techniques considered, as well as literature related to XAI metrics and the psychology of explanations. After identifying research opportunities derived from these works, the paper introduces some alternatives over some of the rule extraction techniques, as well as algorithms to compute the metrics described before. Following this, we present an empirical evaluation of our algorithm with several datasets. We then conclude, showing also potential future research lines of work.

2. 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.

2.1. Unsupervised ML for Anomaly Detection

The review of [1] provides an extensive analysis of the SOTA of ML models for anomaly detection, including unsupervised ones. Unsupervised ML models for anomaly detection can be differentiated according to their feature map, or according to the type of model used (in terms of how the decision frontier is obtained). Regarding the feature map, there are two possible types. First, Shallow models (i.e. Minimum Volume Ellipsoid) versus Deep ones (i.e. Generative Adversarial Networks). Regarding the type of model, four types are mentioned: classification (i.e. OCSVM), probabilistic (i.e. Kernel Density Estimation), reconstruction (i.e. Principal Component Analysis, Deep AutoEncoders) and distance-based (i.e. IsolationForest, Local Outlier Factor). OCSVM is a type of Kernel-based One-Class Classification anomaly detection model that is well-suited for multimodal, nonlinear and nonconvex datasets. OCSVM is also an algorithm that, since its original formulation [11], has been developed with many variations.

OCSVM has advantages in terms of computational performance [12]. One of the reasons is that it creates a decision frontier using only the support vectors (like general supervised SVM). Another advantage is that model training always leads to the same solution because the optimization problem is a convex one. However, SVM (hence OCSVM) algorithms are difficult to explain due to the mathematically complex method that obtains the decision frontier [1].

From a theoretical point of view, SVM for classification 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:

\[
\begin{align*}
\min_{w, \xi, \rho} & = \frac{1}{2}||w||^2 + \frac{1}{\nu n} \sum_{i=1}^{n} (\xi_i - \rho) \\
\text{subject to:} & \\
(w, \phi(x_i)) & \geq \rho - \xi_i \quad \text{for } i = 1, \ldots, n \\
\xi_i & \geq 0 \quad \text{for } i = 1, \ldots, n
\end{align*}
\]

In that equation, \(\nu\) 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:

\[
f(x) = sgn((w, \phi(x) - \rho) \Rightarrow \\
f(x) = sgn(\sum_{i=1}^{n} \alpha_i K(x_i, x) - \rho)
\]

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

2.2. Rule Extraction techniques in XAI

Rule extraction belongs to the group of post-hoc XAI techniques \[1\]. This group of techniques are applied over an already trained ML model (generally a blackbox one) in order to explain the decision frontier inferred by using the input features to obtain the predictions. Rule extraction techniques are further differentiated into two subgroups: model specific and model-agnostic. Model specific techniques generate the rules based on specific information from the trained model, while model-agnostic ones only use the input and output information from the trained model, hence they can be applied to any other model. Post-hoc XAI techniques in general are then differentiated depending on whether they provide local explanations (explanations for a particular data point) or global ones (explanations for the whole model). Most rule extraction techniques have the advantage of providing explanations for both cases at the same time.

2.2.1 Model Specific Rule Extraction techniques in XAI for SVM

\[13\] offers a review of rule extraction techniques for SVM, including the ones that are model specific. Here, regarding model specific techniques, they highlight three different types of algorithms. The first of them are rule extraction algorithms that use the support vectors from the original model as an input source for generating the rules. This is the case of SQRex-SVM \[14\] where the authors propose the usage of a subset of the support vectors for inferring the rules with the usage of a modified sequential covering algorithm. The second type of algorithms use both information from the support vectors together with information from the separating hyper-plane. This is the case of RulExSVM \[15\], where the authors propose a technique applicable for SVM with a RBF kernel. The algorithm uses the support vectors in order to build hyper-rectangles that intersect with the separating hyper-plane. Finally, the last type of techniques use the support vectors, the separating hyper-plane, and the training data. The training data is used to define the regions in the hyperspace, and the support vectors and the hyper-plane define the size of those regions. Within this category appears the proposal of \[10\].

The authors of \[10\] propose a technique called SVM+Prototypes that can be considered model-agnostic or model specific depending on how is implemented. The general intuition consists in finding hypercubes (or hyperspheres) using the centroids (or prototypes) of data points of each class. Then, it can use as vertices either the support vectors from the SVM model, or the data points from that hyperspace area farther away from that centroid. For the first alternative, the proposal is model specific, since it focuses on a specific component of the model itself (the support vectors). The second one is model-agnostic, since it does not use any information that is specific only for SVM models. After this, it infers 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 \(X \geq X_1 \land Y \geq Y_1 \land X \leq X_2 \land Y \leq Y_2\)

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 \[3\].

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 sim-
spatially 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.

2.2.2 Model-agnostic Rule Extraction techniques in XAI

Many rule extraction proposals contribute to XAI without the need to use any specific information from a particular type of model \([1]\). The only information necessary for building the rules is the input features and the model outputs. Some techniques use all the training data, while others need only a few input instances, or they can even generate artificial datapoints to infer the decision frontier. Even though the techniques were initially conceived for supervised ML, they can be extended for unsupervised ML for anomaly detection, since there output is analogous to a binary classifier where the classes are heavily imbalanced, as discussed in Section 1.

A general way to approximate any blackbox model globally is by using a surrogate supervised decision model trained over the same dataset, but instead of using the real labels (the ones used for the blackbox model), it is trained over the predictions of that blackbox model \([5]\). This may be accomplished with any ML model, but it is useful to do it with a whitebox model that can be directly interpreted. Among these whitebox models, some of them may be used for rule extraction. An example is a Decision Tree (DT) model. DT allows explaining the classification logic of the blackbox model through the usage of rules, which can be used even for classifying new instances. The advantages of using a DT as a surrogate global model is its flexibility (it can be applied over any model in an agnostic way) and simplicity (it is a solution that is easy to explain). However, this approximation at the end leads to explain a proxy model, and not the actual data, since the surrogate model never sees the true target values.

Anchors \([16]\) is a model-agnostic XAI technique that extracts rule explanations for individual data points. The purpose of Anchors is finding a decision rule that approximates the decision function of the blackbox model around that individual data point. This rule “anchors” the prediction of that data point, so that any perturbation of the features of that point that are still inside the rule will always return the same output from the blackbox model. The approach is as follows. First, the algorithm generates candidate rules that may explain the data point. Then, it evaluates those candidate rules. In order to do that, Anchors generates permutations around the data point (similar data points to the
original one) that yield the same result. The result is evaluated by calling the blackbox model (the oracle) and obtaining the classification for that data point. In order to optimize the exploration-exploitation of generating and evaluating data points, it uses a reinforcement learning approach with a Multi-Armed Bandit (MAB) approximation. In this MAB, each arm of the Bandit problem is a candidate rule, and the data points generated, after obtaining their classification result from the blackbox model, are used to compute a precision metric used to evaluate the candidate rule’s payoff. This reinforcement learning approach helps minimizing the number of calls to the model in order to reduce the computational cost of the algorithm. Among all the candidate rules, the algorithm then checks if the best one of them matches a predefined convergence criteria. To do that, it filters rules according to a precision threshold, and selects form the remaining ones the one with highest coverage. That rule is used to explain that original data point. If there are no rules that match the convergence criteria, then the algorithm keeps iterating (using a beam search approach) using the $B$ best rules from the previous step in order to generate new candidate rules for the following one. In those following steps, Anchors keep extending the rules with more features (in the first step, it only uses one feature per candidate rule). Thus, Anchors offers a model-agnostic approach that generate IF-THEN rules, easy to interpret, that are generated in an efficient way thanks to the usage of reinforcement learning (MAB) that can be parallelised. However, Anchors is very sensitive to its initial configuration, like many permutation approach algorithms, such as LIME \cite{LIME}. Another important consideration of Anchors is that, while it keeps the calls to the oracle to a minimum (thanks to MAB), it still requires a lot of calls, and that can affect the runtime of the algorithm.

RuleFit \cite{RuleFit} is a model-agnostic surrogate model that learns a linear regression model (Lasso) that uses as features both the original features of the model, as well as new generated features that represent decision rules. In order to accomplish that, first, a tree model is trained over the output and the input features, and the decision paths between the tree levels are turned into decision rules, except for the ones that lead to the leaf nodes, which are not considered. These rules are used as additional features, along with the original ones, on the Lasso surrogate model. Thanks to this, RuleFit yields both rules as well as their contribution, measured through the coefficients of the Lasso model. In summary, RuleFit generates a white-box model that includes rules as features, that can be interpreted as a standard linear regression one. The only caveat is that, for the original coefficients, the predicted outcome changes by $|\beta_j|$ if feature $x_j$ changes by one unit if the other features remain unchanged, while for a feature-rule $r_k$ it is different; if all the conditions of the feature $r_k$ are met, the predicted outcome changes by $\alpha_k$ (the weight associated to that rule-coefficient) for regression. Similarly, for classification tasks, when the conditions of $r_k$ are met, the odds for event vs. no-event changes by a factor of $\alpha_k$.

Similarly to RuleFit, SkopeRules \cite{SkopeRules} is another way to generate rules from tree ensembling techniques. They differ, however, in how they obtain the rules. First, SkopeRules generates the rules using surrogate tree ensembles trained using the input features and the target variable. Then, it applies a filtering step in which, using a threshold for Precision and Recall, some rules are removed and some are kept. This step allows to select only high-performing rules, and removing the ones that do not yield good results. The last step is known as "semantic rule duplication". This step eliminates duplicate rules (rules that are the same or very similar to other ones). It also eliminates again low-performing rules based on their results for a F1-metric. This allows to obtain high-performing as well as heterogeneous rules. The final set of rules is the output of SkopeRules, differing from RuleFit because it does not use a Lasso model to aggregate all rules.

Falling Rule Lists (FRL) \cite{FRL} are classification models that generate a sorted list of IF-THEN rules, thus, they can serve as a model-agnostic global post-hoc rule extraction technique. The rules are binary, and are looked one after the other, in order to see if a particular datapoint can be classified into one of the classes. The rules are sorted according to the probability of classifying a datapoint into that class using that rule. Due to that, FRL offers a list of IF-ELSE IF rules associated to a particular class with a decreasing probability score. This is inspired in the concept of healthcare triage: patients are classified within risk level groups, and the highest-risk ones should be considered first. The particular algorithm cited and used in this paper uses an approach for learning based on a Bayesian framework, instead of a greedy decision tree learning method, named Bayesian Falling Rule Lists (BFRL).

Boolean Decision Rules via Column Generation (BRCG) \cite{BRCG} also provides a binary classifier by using disjuntive normal form (DNF, OR-of-ANDs) or conjunctive normal form (CNF, AND-of-ORS) through interpretable rules. In case of DNF (the one used in this paper), they provide an unordered set of decision rules that classify a datapoint into the positive category if at least one of the rules is satisfied. This is different than other methods already mentioned, such as BFRL where the rules are ordered in an IF-THEN schema, or the surrogate DT model, that provides the rules in a tree structure schema. In this article, we use the BRCG-light approximation from \cite{BRCG-light}, that replaces the integer programing solver used in the original paper by a heuristic beam search one.

Generalized Linear Rule Models (GLRM) \cite{GLRM} generate decision rules and combine within a linear model (generalized additive model, GAM). Thus, they provide both a non-linear modelling, thanks to the decision rules, while keeping the interpretability by using a linear model that ensembles them. However, as \cite{GLRM} notice, while it is feasible to interpret linear combinations of rules, if the number of rules increases too much, there is a risk of losing the interpretability of the model. The authors of the original paper highlight that in order to reduce the rules generated and not lose interpretability, they use a rule selection technique based on column generation (CG). CG searches the spaces of rules and generates them only when they are needed, and then fits again the GLM model. This allows analysing again old rules, re-weight them, and discard the ones that are not needed anymore. This is different to other methods used in the lit-
erature, mainly pre-selecting a subset of candidate rules using optimization techniques, or a greedy optimization approach by adding rules one by one using sequential covering or boosting techniques.

2.3. XAI for Anomaly Detection

XAI is useful for both explaining an anomaly detection model from a global perspective, or for explaining the identification of particular instances as outliers. From the global explanation level, \cite{24} use two anomaly detection ML algorithms (Decision tree and DeepLog) to detect outliers over log data. Together with that, they use Shapely values in order to generate model-agnostic feature relevance explanations that help to see which features contribute more for predicting outliers by seeing the individual contribution of each feature to the general outlier probability.

XAI \cite{8} has also been used for anomaly detection for predictive maintenance. The authors highlight that even when an anomaly detection model is very accurate, the operators that will get the model prediction may not trust it if it remains a black box that does not provide any insights about its decisions. Because of that, they propose an anomaly detection system where the explanations are generated thanks to the usage of a whitebox model (ElasticNet Logistic Regression). So, they provide explanations in terms of feature relevance, focusing on explaining what contributes to an anomalous state. With that, they highlight that explanations for anomaly detection can be generated in a similar way to those of a supervised ML model for binary classification (even though anomaly detection models provide an output heavily imbalanced).

Shapely values for explaining anomalies also appear at \cite{25}, where the SHAP algorithm is used to generate feature relevance explanations in order to explain what contributes specimen mix-up. For the anomaly detection, they use a Gradient Boosting Tree in order to be able to learn efficiently from highly unbalanced data while yielding good predictions. The authors highlight the importance of having a highly accurate model that is able to predict correctly the specimen mix-up, because this is a crucial problem that may lead to an incorrect diagnostic or an inappropriate therapy.

An additional recent reference is \cite{7}. Here, the authors also cover within the review of the SOTA of anomaly detection the importance of using XAI in order to have a deeper understanding of the model. Their focus on explanations is mainly for unsupervised deep learning (DL) models, where the explanations can be produced by model-agnostic post-hoc techniques for feature relevance (LIME) or by using model specific algorithms (LRP). One of the usages of XAI that they describe is the improvement of the model based on the explanations provided. They show an example for anomaly detection based on images, where XAI helps to see the cases where the pixels used for making the decision are actually the correct ones.

The analysis of the literature highlights how detecting anomalies is critical within some domains, and because of that, their detection needs to be very precise. However, being able to detect anomalies is not enough, and explanations are needed for both understanding the model better (and seeing if it can be trusted or improved), as well as for explaining the model for other audiences in order to see if they can also rely on the predictions or not (something connected to the explanation generation for different user profiles \cite{1}). A model may perform apparently very well and explanations may help to see that the model is taking its decision by using features that are not relevant \cite{5}, so in that case, the model may not be finally trusted. This shows that XAI can complement the classical evaluation of models based only on their performance.

However, after the assessment of a model and seeing that it behaves correctly (from both the XAI and the performance point of view), before providing explanations to some user profiles, it is important to ensure that they are aligned to what the model predicts, and are not showing any contradictory information. A way to do it within the rule extraction scenario is by using P@1 rules with respect to the model output. With that, even though they may not be explaining the whole model or are not able to explain every instance, their explanations will be completely aligned with the model.

For feature relevance explanations, the literature shows that they help to see how they contribute to the positive class (outliers in anomaly detection). For rule extraction explanations, they can help to explain outliers with respect to what will turn that outlier into an infier. Considering this, the explanations will target the infier class, so the outliers can be explained in a counterfactual approach with respect to the non-anomalous subspace (for local explanations). For global explanations that help to see what feature values are normally associated to outlier situations, the explanations would still target the outlier class.

2.4. XAI for OCSVM

The terms "XAI", "explainable" or "interpretable", together with "OC-SVM" or "OCSVM" only provide 4 results searching them within titles, abstract and/or keywords within Scopus®. One of them is the work of \cite{26}. Here, the authors propose a model-specific method based on the concept that OCSVM models can be rewritten as pooling neural networks. Due to the asymmetry between inliers and outliers, they model with a min-pooling over distances for outliers, and a max-pooling over similarities for inliers. Thanks to turning OCSVM models to a neural network, they apply a deep Taylor decomposition (DTD) to obtain explanations in terms of input features. DTD serves as a framework to apply layer-wise retropropagation (LRP) in order to obtain the feature contribution of the input features to a predicted output. The authors extend the explanations generated to include using both input features or support vectors. In \cite{27} the authors benchmark different unsupervised ML algorithms for anomaly detection (IsolationForests, OCSVM, Cluster Support Vector Data Description and One-Class decision Tree, OC-Tree), and analyse them over data from the medical domain. They indicate that OC-Tree provides the best results. OC-Tree has the advantage of being a hybrid method that combines the first kernel density estimation for anomaly detection with a decision tree that automatically provides rules that explains the first model. The benchmark of the models is performed in terms of predictive performance, mentioning
that OC-Tree is then better for that use case since it directly provides explanations. In [23] the authors use OCSVM and Variational Autoencoders for detecting engine faults within 2.4L diesel engines. The faults, which may belong to two types, are precisely the anomalies. For that they use 130 feature parameters. Together with that, they include a post-hoc explainability layer by using LIME (thus, explaining the models in terms of feature relevance).

[9] also shows the combination of OCSVM with XAI. For the XAI part, they use the algorithm Ripper for rule induction. For this algorithm, they use the information from the support vectors from the OCSVM. At the evaluations, they use three different datasets and measure the performance of the rules extracted in terms of Precision, Recall and F1 metrics over the ground truth of the real anomalies. They also train OCSVM models with a RBF kernel.

The previous analysis of the literature shows that even though there are some works regarding XAI and OCSVM, they are either focused in a particular data field, or they do not compare many rule extraction methods (for the previous literature that uses OCSVM with rule extraction). Due to that, there is still an open area regarding the benchmark of rule extraction techniques over OCSVM models for anomaly detection.

2.5. Metrics for XAI

Beyond indicating the importance of both detecting outliers and being able to explain how the decision took place, it is also crucial to quantify the quality of those explanations. There are some recent reviews in the literature that deal with the challenge of providing metrics in XAI, such as [29]. In that article, the authors analyse the literature and define a taxonomy of properties that should be considered in the individual explanations generated by XAI techniques. Even though the paper deals with quantifying the quality of the explanations for an individual datapoint, some of them are also applicable for global explanations.

- **Accuracy**: It is related to the usage of the explanations to predict the output using unseen data by the model.
- **Fidelity**: It refers to how well the explanations approximate the underlying model. The explanations will have high fidelity if their predictions are constantly similar to the ones obtained by the blackbox model. The authors mention how accuracy and fidelity are intertwined: If the explanations have high fidelity (thus, approximate the model well) and the model has high accuracy, the explanations will also have high accuracy. However, the explanations may have high accuracy (because they predict very well over unseen data) while having low fidelity (because they do not approximate well the original model).
- **Consistency**: It refers to the similarity of the explanations obtained over two different models trained over the same input dataset. High consistency appears when the explanations obtained from the two models are similar. However, a low consistency may not be a bad result since the models may be extracting different valid patterns from the same dataset due to the "Rashomon Effect" (seemingly contradictory information is factual telling the same from different perspectives).
- **Stability**: It measures how similar the explanations obtained are for similar datapoints. Opposed to consistency, stability measures the similarity of explanations using the same underlying model.
- **Comprehensibility**: This metric is related to how well a human will understand the explanation. Due to this, it is a very difficult metric to define mathematically, since it is affected by many subjective elements related to human’s perception (such as context, background, prior knowledge, etc.). However, there are some objective elements that can be considered in order to measure "comprehensibility", such as whether the explanations are based on the original features (or based on synthetic ones generated after them), the length of the explanations (how many features they include), or the number of explanations generated (i.e. in the case of global explanations). In general terms, using the original features, while keeping the number of explanations generated and the features used to a minimum, will increase comprehensibility.
- **Certainty**: It refers to whether the explanations include the certainty of the model about the prediction or not (i.e. a metric score).
- **Importance**: Some XAI methods that use features for their explanations include a weight associated with the relative importance of each of those features.
- **Novelty**: Some explanations may include whether the datapoint to be explained comes from a region of the feature space that is far away from the distribution of the training data. This is something important to consider in many cases, since the explanation may not be reliable due to the fact that the datapoint to be explained is very different from the ones used to generate the explanations.
- **Representativeness**: It measures how many instances are covered by the explanation. Explanations can go from explaining a whole model (i.e. weights in linear regression) to only be able to explain one datapoint.

Considering the case of rule extraction techniques, the outputs (rules) for the whole dataset can be analyzed from the perspective of global explanations. In this context, one additional aspect to consider is diversity, a metric that indicates whether the explanations are redundant or repetitive and can already be mostly covered by another explanation, or if they provide insights that are not deductible from the other explanations available.

From among all these metrics, [13] already commented on the importance of comprehensibility, accuracy and fidelity for rule extraction techniques that explain a SVM model. The metrics are defined as:

- **Accuracy** = No. instances classified correctly by the rules / Length test set
- **Fidelity** = No. instances where the rule predictions match the model predictions / Length test set

For "consistency", No. of rules and No. of antecedents (analogous to rule size).

[30] analyses XAI metrics for Random Forests by using an Interpretability Matrix that shows the relationship be-
between Rule Coverage - Rule Certainty - Feature Relevance. [31] shows a model-agnostic comparative for rule extraction algorithms using C4.5Rule-PANE, REFNE, RxREN and TREPAN. For that, they use 8 datasets of up to 8124 total instances and 40 features. As blackbox models they use Neural Networks models for classification (with different configurations). Finally, they propose several metrics for measuring the quality of the explanations.

- **Completeness**: Percentage of input instances covered by rules over total input instances. Analogous to "Representativeness".
- **Correctness**: Percentage of input instances correctly classified by rules over total input instances. Analogous to "Accuracy".
- **Fidelity**: Percentage of input instances on which the predictions of model and rules agree over total instances.
- **Robustness**: Applying small perturbations over the datapoints that do not change the prediction of the model, the sum of differences between the original prediction and the new prediction, divided by the number of instances analyzed. It is analogous to the concept of "Stability".

\[
\text{Robustness} = \frac{\sum_{n=1}^{N} f(x_n) - f(x_n + \delta)}{N}
\]  

(3)

Robustness is further analysed in [32], where the authors evaluate it for feature relevance model-agnostic post-hoc XAI techniques (LIME and SHAP).

- **Number of rules** and **Average rule length**, similar to [13].

They apply these metrics and see, using the Friedman’s test, that C45-Pane has significantly superior results over all of the datasets considering all of the metrics, followed by TREPAN. The remaining papers yielded by the query aforementioned either do not deal with metrics for rule extraction techniques, or only focus in the "Accuracy" aspect.

For our research regarding rule extraction, we will focus in analysing the degree of "comprehensibility" of the rules, the coverage of those rules of the datapoints available ("representativeness"), if the rules approximate the underlying model ("stability"), and if they have overlaps among them and are redundant ("diversity"). The advantage of these metrics for unsupervised ML is that they do not need any ground truth information about the "correct" output that the model should have, as opposed to other metrics like "accuracy". This is interesting because many times that ground truth is not available. This is applicable to our use case, since we want to detect anomalies over real industry datasets belonging to Telefónica where there is no prior information about the anomalies.

The challenge here is defining how to quantify the metrics of stability and diversity (since comprehensibility and representativeness for rule extraction is already defined in the previous SOTA. Stability was also defined previously, but it is important to minimize the number of data points to consider in order to generalize the metric for large datasets (where generating perturbation for every input datapoint would be computationally expensive). This is specially important when working with unsupervised ML models, where many times there is not a reduced test set available, and the evaluations needs to be done using only the training data. Also, when working with P@1 rules, where the fidelity to the original model will always be perfect, it is important to measure the stability in order to see how the explanations behave with unseen data (moreover when there is no test set available). Due to that, we will propose and use algorithms to quantify them in a rule extraction scenario applied for unsupervised ML for anomaly detection.

### 2.6. The psychology of explanations

In ML, explanations are the “key” to open blackbox algorithms, and are therefore likely to play an important role in possible future AI regulations. It is expected that in certain domains or applications, whenever an algorithm takes an autonomous decision or provides a recommendation that has a significant impact on people’s lives, some kind of explanation is required [33]. In this paper, we have quantified several quality parameters of explanations generated by different methods for non-supervised learning algorithms, including the comprehensibility of the generated explanations. But for explanations to be comprehensible and effective for people in different situations, we also need to consider the consumer of the explanations: the people.

How do people understand explanations? In psychology, explanations are seen as crucial for human knowledge and learning, as they are considered proofs of understanding. According to Wilkinson [34], there are three kinds of views of explanations: the formal-logical view (an explanation is like a deductive proof given some propositions), the ontological view (events -state of affairs- explain other events), and the pragmatic view (an explanation needs to be understandable by the “demander”). Explanations that are sound from a formal-logical or ontological view, but leave the demander in the dark, are not considered good explanations. For example, a very long chain of logical steps or events (e.g. hundreds) without any additional structure can hardly be considered a good explanation for a person, simply because he or she will lose track.

Wilkinson introduces two more concepts to define the adequacy of explanations for demanders. The level of explanation refers to whether the explanation is given at a high-level or more detailed level. The right level depends on the knowledge and the need of the demander: he or she may be satisfied with some parts of the explanation happening at the higher level, while other parts need to be at a more detailed level. The kind of explanation refers to notions like causal explanations and mechanistic explanations. Causal explanations provide the causal relationship between events but without explaining how they come about (a kind of "why" question). For instance, smoking causes cancer. A mechanistic explanation would explain the mechanism whereby smoking causes cancer (a kind of "how" question). Causal explanations can be further divided into common-cause explanations (a single cause has several consequences), common-effect explanations (several causes
converge to one consequence), and simple linear chain explanations (one causes leads to one consequence) [35].

As said, a satisfactory explanation does not exist by itself, but depends on the demander’s need. In the context of ML algorithms, we can distinguish between several typical demanders of explainable algorithms [1]:

- Domain experts: those are the “professional” users of the model, such as medical doctors who have a need to understand the workings of the model before they can accept and use the model.

- Regulators, external and internal auditors: like the domain experts, those demanders need to understand the workings of the model in order to certify its compliance with company policies or existing laws and regulations.

- Practitioners: professionals that use the model in the field where they take users’ input and apply the model, and subsequently communicate the result to the users’ situations, such as loan applications.

- Redress authorities: the designated competent authority to verify that an algorithmic decision for a specific case is compliant with the existing laws and regulations.

- Users: people to whom the algorithms are applied and that need an explanation of the result.

- Data scientists, developers: technical people who develop or reuse the models and need to understand the inner workings in detail.

In summary, for explainable AI to be effective, the final consumers (people) of the explanations need to be duly considered when designing XAI systems.

3. Our Proposal

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.

3.1. 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. As already mentioned in the introduction, [10] proposes an algorithm to extract rules from a SVM model by performing clustering over the datapoints that belong to one of the classes. The clustered datapoints will be used to obtain a geometric surface that encloses the rest of the datapoints inside. There are two ways to accomplish it: building hypercubes or building hyperspheres. This paper will focus the analysis over the first approach: building hypercubes. The paper will also focus in the model-agnostic variant, where the algorithm obtains the furthestmost datapoints from inside the cluster as vertices for the hypercube, so they enclose the rest of datapoints of that category inside (the model specific alternative uses the support vectors). In case that the hypercube generated encloses points from the other category, then the number of clusters will be increased, aiming to obtain smaller cubes that could fit the data without including points from the other class. This is done iteratively until no points from the other class are inside the hypercubes, or a maximum number of predefined iterations is reached. During the process, if a hypercube does not contain points from the other class, then that hypercube is translated into a rule, and those datapoints are removed from the following iteration steps.

Images [4] and [5] shows an example application of this algorithm for a 2D space. In Image [4], appears the initial scenario, where the first step in the iteration process consists in applying one cluster over the dataset for datapoints of one of the classes (blue ones). However, with one cluster, the 2D square that enclose the datapoints contains points from outside the other class, so more clusters need to be applied. As [5] shows, iteration 3 (with 3 clusters) is the first one with squares without red points, so those subspaces are turned into rules and the points inside them removed from the iteration process, that starts again with one cluster for the remaining datapoints. Iteration 6 will be the last one, and 5 rules have been extracted up to that point.

![Figure 4: Clustering over a 2D space. With one cluster over datapoints from one class (blue), there are still others from the other class (red) inside the square.](image4)

![Figure 5: Applying the proposal of [10], the number of clusters keeps increasing until no points from the other class are inside, an then that hypercube is translated into a rule.](image5)

The approximation proposed before is not the only one that can be applied in order to extract the rules. Image [6] shows one of our alternative proposals over [10] method. Instead of removing datapoints that are inside a rule without points from the other class, the process always keeps all datapoints in every iteration since there could be clustering...
patters that could only be found if all points are together. In this approach, the number of clusters is constantly increased until no datapoints from the other class are inside the hypercubes, or the maximum number of iterations is reached. We will further address this method as "keep" in the remaining of the paper. In contrast, the references to [10] method will be addressed as "keep_reset".

Figure 6: Keeping all datapoints in every iteration could lead to a reduced number of clusters since there may be data patterns that could only be found in this scenario.

Another proposal that we include in this paper over [10] is splitting the subspaces in a binary partition scheme. This is an alternative over the original proposal, that constantly increases the number of clusters until one rule has only datapoints from the same class, and then restarting the clustering process from the beginning for the remaining ones. We will address this method as "split" for the remaining of the paper. Image 7 shows how the same 2D example using this approach.

Figure 7: Splitting subspaces with a binary partition scheme until no red points are inside the rule.

According to the taxonomy for XAI in [5], our method has the following characteristics:

• 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 [5], 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 the explanation algorithm is model-agnostic, it can work for any blackbox model. The only information needed is the train dataset and the outputs from the model. To illustrate it, this paper will show evaluations over OCSVM models with different kernels: radial basis function (RBF) and linear kernel.

Regarding the clustering technique itself, potentially any algorithm could be used, both for [10] or for any of out two proposals over it from this paper. However, there is a caveat that should be considered. The clustering algorithm needs to take into account if the features are only numerical, categorical (non ordinal), or both.

One algorithm that will be used in this paper for extracting the hypercubes is K-Means ++ [36]. However, the standard version of this clustering algorithm is designed for numerical features, and categorical ones should be treated differently. 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 8.

Figure 8: 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 ∧ Y ≥ Y2 ∧ Y ≤ Y1
• Rule 2: NOT OUTLIER IF X = 1 ∧ Y ≥ Y4 ∧ Y ≤ 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.

Besides K-Means++, there are other clustering algorithms that could be applied. In this paper we will analyse also the rules obtained by applying K-Prototypes \(^{[37]}\). The advantage of using K-Prototypes is that it can work directly with both categorical and numerical features.

### 3.2. 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_extract` is the main function of the algorithm.

### Algorithm 1 Main pipeline

```plaintext
1: procedure OC-SVM_RULE_EXTRACT(X, d_f, d_p, m, t)
2: \( l_n \leftarrow d_f[l_n] \)
3: \( l_c \leftarrow d_f[l_c] \)
4: \( X[l_n] \leftarrow \text{featureScaling}(X[l_n]) \)
5: \( X \leftarrow \text{featureEncoding}(X[l_c]) \)
6: \( \text{model} \leftarrow \text{OneClassSVM}(d_p) \)
7: \( \text{model.fit}(X) \)
8: \( \text{pred} \leftarrow \text{model.train}(X) \)
9: \( \text{distances} \leftarrow \text{model.decisionFunction}(X) \)
10: \( X_{y}, X_n \leftarrow \text{filterAnomalies}(X, \text{preds}) \)
11: if \( \text{len}(l_n) = 0 \) then
12: \( \text{rules} \leftarrow \text{getR}(X_{n}, X_y, X, d_f, m, t) \)
13: else if \( \text{len}(l_n) = 0 \) then
14: \( \text{rules} \leftarrow \text{rulesEmptyList}(X_n, l_c) \)
15: else
16: \( \text{cat} \leftarrow \text{getUnique}(X_n, l_c) \)
17: \( \text{rules empty list} \)
18: \( \text{for } c \in \text{cat do} \)
19: \( X_{nf}, X_{nf} \leftarrow \text{filterCat}(X_{n}, X_y, c) \)
20: \( \text{rules.append}(\text{getR}(X_{nf}, X_{nf}, d_f, m, t)) \)
21: end for
22: end if
23: \( \text{rules} \leftarrow \text{featureUnscaling}(\text{rules}, l_n) \)
24: \( \text{rules} \leftarrow \text{pruneRules}(\text{rules}, d_f) \)
25: return \( \text{rules} \)
26: end procedure
```

3.2.1. Parameters

- `X`: Data points, a dictionary with two lists (\( l_n \) a list with the numerical columns and \( l_c \) a list with the categorical columns), \( d_p \) is a list of the hyperparameters for OCSVM (kernel type, upper bound on the fraction of training errors and a lower bound of the fraction of support vectors, \( \nu \), and the kernel coefficient, \( \gamma \)).
- `d_f`: Function `featureScaling`.
- `m`: Function `featureEncoding`.

A function `getUnique` returns a dictionary with two lists (`\( n_f \)` a list with the categorical columns and `\( n_c \)` a list with the numerical columns).

The next step is checking the type of features available. 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.

Function `getR()` calls different subfunctions depending on the `t` parameter value, but in any of the cases, the approach is similar: clustering non-anomalous data points in a set of hypercubes that do not contain any anomalous data points.

The "keep" approach, described in algorithm 2, 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, \( X_y \). `getRulesKeep` then calls function `getVertex` (described in algorithm 3) with a specific number of clusters, \( n_{cl} \). This function 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 specified cluster algorithm.

If the algorithm is K-Prototypes, then if considers both categorical and numerical features (using `getKP` function). If is K-Means++, then it applies the clustering over numerical features only (using `getKM` function).

Then, it iterates through each cluster and 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 `getDist` and uses the furthest \( n_v \) as datapoints for obtaining the vertices that enclose the cluster using the `getVertex` function. \( n_v \) is a value that represents the hyperspace dimensionality, and is obtained with `hyperDimension` function. In case there are less datapoints than the number of vertices that a hypercube of that dimen-
sionality has, then all of them are used for obtaining the vertices. This last scenario does not stop the iterations, since a hypercube in this situation could still include outliers, needing further splitting. As long as there are no outliers inside the rules, they are stored in rules list. However, as soon as there is one rule with outliers inside, then the whole process is repeated again with one more cluster. This keeps taking place until no outliers are inside the rules or the maximum number of iterations is reached.

Algorithm 2 Rule Extraction - Keeping all datapoints

1: procedure GETRULESKEEP($X_n, X_y, m, d_f$
2: $l_n \leftarrow d_f[l_n]$
3: $l_c \leftarrow d_f[l_c]$
4: max_iter reference value
5: check $\leftarrow$ True
6: n_clusters $\leftarrow$ 0
7: while check do
8: rules empty list
9: if n_clusters $> max_{\text{iter}}$ then
10: check $\leftarrow$ False
11: else
12: ncl $\leftarrow$ ncl + 1
13: vInfo $\leftarrow$ getVertex($X_n, X, d_f, m, ncl$
14: for iterValue $\in$ vInfo do
15: rulescluster $\leftarrow$ iterValue[0]
16: Xnc $\leftarrow$ iterValue[1]
17: l_y $\leftarrow$ outPosition(rulescluster, $X_y$)
18: if len(l_y) $= 0$ then
19: rules.append(rulescluster)
20: check $\leftarrow$ False
21: else
22: check $\leftarrow$ True
23: end if
24: end for
25: end if
26: return rules
27: end procedure

The "split" approach is defined in algorithm 3. This function has some similarities with 2 with the following differences. Instead increasing the number of clusters in every iteration, n_cl is always 2. Also, l_sub receives the data after every split. Initially, l_sub contains only one dataset, the inliers $X_n$. However, after another iteration, its value is set to the data from the clusters in which the rules did contain some outlier.

In any of the three methods, 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 checks whether there is another with a bigger scope that will include it as a subset case.

Algorithm 3 Rule Extraction - Binary partition approach

1: procedure GETRULESSPLIT($X_n, X_y, m, d_f$
2: $l_n \leftarrow d_f[l_n]$
3: $l_c \leftarrow d_f[l_c]$
4: max_iter reference value
5: check $\leftarrow$ True
6: l_sub $\leftarrow$ [X_n]
7: rules empty list
8: while check do
9: if len(l_sub) $= 0$ or j $> max_{\text{iter}}$ then
10: break
11: end if
12: l_original $\leftarrow$ l_sub
13: l_sub $\leftarrow$ []
14: for d in l_original do
15: ncl $\leftarrow$ 2
16: vInfo $\leftarrow$ getVertex($X_n, X, d_f, m, ncl$
17: for iterValue $\in$ vInfo do
18: rulescluster $\leftarrow$ iterValue[0]
19: Xnc $\leftarrow$ iterValue[1]
20: l_y $\leftarrow$ outPosition(rulescluster, $X_y$)
21: if len(l_y) $= 0$ then
22: rules.append(rulescluster)
23: check $\leftarrow$ False
24: else
25: check $\leftarrow$ True
26: l_sub $\leftarrow$ l_sub.append(Xnc)
27: end if
28: end for
29: end for
30: end while
31: return rules
32: end procedure

3.3. Influence of the kernel

As mentioned before, OCSVM models are configured using mainly three hyperparameters: $\nu$, $\gamma$ and the kernel type. Depending on the kernel type, the construction of the decision frontier to differentiate between outliers and inliers changes. In particular, Radial Basis Function (RBF) kernel will find hyperspheres (one or more) that enclose the inliers, leaving outliers outside.

The diverse density of outliers versus inliers highlights that there may be differences in the rules depending on which class they enclose. Mainly, since the decision function is a hypersphere, the intuition is that it will be easier to find rules that enclose all those points. Figure 9 illustrates this idea.

3.4. Algorithms for metrics

As mentioned before, the metrics considered in this paper are divided into four subsets: comprehensibility, representativeness, stability and diversity. Since, to the best of our knowledge, some of these metrics are not implemented within the main XAI frameworks, we propose within these paper a set of algorithms to compute them in a rule extraction scenario. We apply the metrics for the case of unsu-
Algorithms 4 Additional functions

1: procedure GET_VERTEX\(X_n, d_f, m, n_{cl}\)
2: \(l_n \leftarrow d_f[l_n]\)
3: \(l_c \leftarrow d_f[l_c]\)
4: \(n_v \leftarrow \text{hyperDimension}(X_n, d_f)\)
5: \(d_{bounds}\) empty list
6: \(d_{points}\) empty list
7: if \(m = k\) then
8: \(\text{labels, centroids} \leftarrow \text{getKP}(X_n, l_n, l_c, n_{cl})\)
9: else
10: \(\text{labels, centroids} \leftarrow \text{getKM}(X_n, l_n, n_{cl})\)
11: end if
12: for \(c \in n_{cl}\) do
13: \(X_{nc} \leftarrow \text{insideCluster}(\text{labels}, X_n)\)
14: if \(\text{len}(X_{nc}) > n_v\) then
15: \(p_{\text{chosen}} \leftarrow \text{getDist}(X_{nc}, \text{labels}[c])\)
16: else
17: \(p_{\text{chosen}} \leftarrow X_n\)
18: end if
19: \(\text{vertices} \leftarrow \text{getVertices}(p_{\text{chosen}})\)
20: \(d_{bounds}.\text{append}(\text{vertices})\)
21: \(d_{points}.\text{append}(X_{nc})\)
22: end for
23: return \(d_{bounds}, d_{points}\)
24: end procedure

Proposition 1 implements rule-extraction techniques to build an inlier model for those artificial samples and checks if the predictions using the rules are the same. The steps for these metric are described below, and the detailed pseudocode appears in Algorithm 5.

**Comprehensibility**: The metrics for "comprehensibility" are directly analyzed from the rules themselves; \(n_{rules}\) is computed counting the number of rules generated, and \(size_{rules}\) is computed checking the elements that define the rule (i.e., \(X > 3\) AND \(X < 7\) AND \(Y > 1\) have a \(size_{rules} = 3\) while \(X > 3\) have a \(size_{rules} = 1\)). This proposal already appears in [13].

**Representativeness**: The metric \(\text{per}_{.p1}\) for "representativeness" simply checks the percentage of datapoints for the target class explained with \(P@1\) rules. The other metric in this group is \(\text{pl}_{coverage}\). It checks the median performance of the rules themselves: it computes the median percentage of coverage for the target class by each rule. This

Figure 9: 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.

**Modelling agreement**: The metric \(\text{precision}_{vs}\_\text{model}\) computes the "stability" metric of the hypercubes. The first step is obtaining the prototypes from the dataset and generate random samples near them. Then, obtain the prediction of the original model for those artificial samples and checks if the predictions using the rules are the same.

The steps for these metric are described below, and the detailed pseudocode appears in Algorithm 5.

**Proposition 1** receives the dataset \(X\) of inliers/outliers (depending on whether the rules are computed for inliers or outliers), the rules \(X_r\) and the OCSVM fitted and trained model \(clf\). Then obtains the prototypes with \(\text{ProtodashExplaner()}\) function and generates the random samples \(X_s\) near them with \(\text{randomNear()}\), where an upper and lower limits \((th_s, th_l)\) can be defined for how close are those points to the

**Stability**: The metric \(\text{precision}_{vs}\_\text{model}\) computes the "stability" metric of the hipercubes. The first step is obtaining the prototypes from the dataset and generate random samples near them. Then, obtain the prediction of the original model for those artificial samples and checks if the predictions using the rules are the same.
prototypes. Then, it checks which rules enclose that datapoint with checkInR(), and if at least one of them encloses the datapoint, it is considered that it can be classified using the rules. The metric precision vs model is specified in n_precision variable, that checks the percentage of agreement between the classifications using the rules and the ones with the model, through checkInModel() function.

Algorithm 5 Stability
1: procedure GETAGREEMENT(X, X_r, clf)
2: X_p = ProtodashExplainer(X)
3: S ← []
4: for p ∈ X_p do
5:     S ← S.append(randomNear(p, th1, th_s))
6: end for
7: n_precision ← 0
8: l_rules ← []
9: for d ∈ X_s do
10:     l_iter ← []
11: for r ∈ X_d do
12:     l_iter ← l_iter.append(checkInR(d, r))
13: end for
14: r_rules ← max(l_iter)
15: r_model ← checkInModel(d, clf)
16: if r_rules = r_model then
17:     n_precision ← n_precision + 1
18: end if
19: end for
20: n_precision ← n_precision/len(X_s)
21: return n_precision
22: end procedure

Diversity: The metric to measure "diversity" is score_intersect, and it analyses if the rules are different with few overlapping concepts. This is computed checking the area of the hypercubes of the rules that overlap with another one. The way to check this is by seeing the 2D planes of each hypercube (by keeping two degrees of freedom for the features in the hyperplane coordinates; n-2 features are maintained and the other two are changed between their max/min values in order to obtain the vertices of that 2D plane). Then, it obtains the area of the 2D planes for the rules that overlaps, and each of those 2D areas is turned into a score between 0 and 1 by using the Jaccard similarity index and dividing the area of intersection of the 2D planes by their area of union.

The pseudocode for this metric appears in algorithm 6. Algorithm 6 receives the dataset X of inliers/outliers (depending if the rules are computed for inliers or outliers), the rules X_r, the list of columns for numerical features l_n and the one for categorical l_c. The first step is obtaining all the two tuples combinations of numerical features, using combinations() function. After that, it obtains the combination of categorical values with function unique(). The algorithm then analyses separately the rules that belong to each categorical combination values. For each of those subsets of rules X_r-i, if there are at least two rules, then it defines the tuples of possible rule combinations, combR.

Then, it iterates per each combination of two numerical features. These two features will correspond to the features that will be changed, leaving the rest of the l_fix features fixed, in order to extract 2D planes from the hypercubes with get2D(), and storing those planes in polys variable. Those planes are used for obtaining the Jaccard similarity index with scorePolys() function. If there is an iteration where one of the two dimensions has the same value, it is skipped since the area will be 0. (checkEqual(pair_f)).

Image 10 describes the process for an example in a 3D space. Since all the rules translate into a hypercube, we can choose two features at a time (leaving the rest fixed) and obtain the coordinates for those 2D planes (using their vertices values). Then, for two rules, we can see the area of overlapping between those 2D hyperplanes, as well as their area of union. With that areas, we obtain the Jaccard similarity index. Since the Jaccard similarity index (score_i) yields a value between 0 and 1 (0 when there is no overlapping, and 1 when the area of intersection is the same as the area of union in a total overlapping), we can turn it into a metric in order to express a score value by doing 1 − score_i, so a perfect score will be the one corresponding to no overlap between the rules. This is repeated for all 2D planes of the hypercubes, and we compute the mean of all the individual scores in order to have one final metric (final_score) that is still between 0 and 1, with 1 the perfect score and 0 the worst.

Figure 10: The overlapping between rules (hypercubes) approximated using their 2D planes' area of intersection.

All the algorithm that we have proposed for computing XAI metrics are XAI-specific metrics: metrics that are specific for a particular type of XAI technique (in this case, rule extraction).

3.5. Pruning rules
Many of the rules obtained with all the methods described above are suboptimal, since they can be enclosed into another bigger rule. In order to reduce the number of rules, and remove redundancies, we apply a simple pruning technique prior to the computing and evaluation of metrics. We check every hypercube generated and see if their limits are inside any other rule. If they are, we eliminate that rule from the set of rules. We check this for every rule against every other rule in the dataset, and we keep checking it in a loop until no rules are eliminated.
late all metrics. In general, it is better to have a lower value for comprehensibility metrics (less rules, less rule size) since that contributes to an enhancement of comprehensibility. Regarding the rest of the metrics, higher values are better. Thus, a simple way to compute this is adding the results for representativeness, stability and diversity (adjusting their relative importance by a set of weights), and subtracting comprehensibility results. Since the values for the metric of comprehensibility are the only ones that are not in a range of 0 to 1, we scale them before computing this metric in order to have all values in the same range by dividing them with respect to the number of inliers or outliers (number of rules) or by a value based on the number of features (rule size).

With this, a higher final value will be better. This is expressed in Equation 4.

$$C = \alpha_1 * (1 - n\text{-rules}) + \alpha_2 * (1 - size\text{-rules}) + R = \beta_1 * per\_p1 + \beta_2 * p1\_cov \quad S = \gamma * p\_vs\_m \quad D = \theta * score\_int$$

$$final\_metric = \frac{(R + S + D + C)}{N}$$

N is equal to 5 in this case since we are considering 5 metrics. The different $\alpha$, $\beta$, $\gamma$ and $\theta$ parameters could be adjusted in order to weight the different metrics in case one of them are more important than others. Our proposed methods to compute a general metric is a very naive way to approach it, and more sophisticated ways could be explored. However, it important to highlight the need to be able to analyse everything together for some use cases since there are many XAI aspects to measure and it may difficult the comparison between XAI techniques.

### 4. Evaluation

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

- **Hypothesis 1 (H1):** The rule extraction method of [10] and our proposed variations applied over OCSVM for anomaly detection using a RBF kernel yield significantly less P@1 rules when applied for explaining inliers than over outliers or when using a linear kernel.

- **Hypothesis 2 (H2):** Our proposed variations over [10] yield similar results for P@1 rules that explains the inliers of an OCSVM anomaly detection model when compared to [10] in terms of explainability regardless of the kernel (considering Linear and RBF).

- **Hypothesis 3 (H3):** The rule extraction method of [10] and our proposed variations yield better results for P@1 rules that explains the inliers of an OCSVM anomaly detection model in terms of explainability than other rule extraction techniques and regardless of the kernel (considering Linear and RBF).

As mentioned in Section 3, explanations in terms of rule extraction for anomaly detection may help to see with a counterfactual view what would make an outlier turn into

---

**Algorithm 6 Diversity**

```plaintext
1: procedure GETINTERSCORE(X, X_r, l_n, l_c)
2: \( l\_free \leftarrow \text{combinations}(l\_n, 2) \)
3: \( X\_c \leftarrow \text{unique}(X[l\_c]) \)
4: \( \text{score} \leftarrow [\] \)
5: \( n\_inter \leftarrow 0 \)
6: for \( \text{cat} \in \text{rows}(X\_c) \) do
7: \( X\_r\_i \leftarrow X\_r[\text{cat}] \)
8: if \( \text{len}(X\_r\_i) > 2 \) then
9: \( \text{combR} \leftarrow \text{combinations}(X\_r\_i, 2) \)
10: for \( \text{pair}_f \in \text{combR} \) do
11: if checkEqual(pair_f) then
12: continue
13: end if
14: \( l\_fix \leftarrow l\_n[! = \text{pair}_f] \)
15: \( \text{polys} \leftarrow \text{get2D}(\text{combR}, l\_fix, \text{pair}_f) \)
16: \( \text{score}_i, n\_i \leftarrow \text{scorePolys}(\text{polys}) \)
17: \( \text{score} \leftarrow \text{score}\_appends(1 - \text{score}_i) \)
18: \( n\_inter \leftarrow n\_inter + n\_i \)
19: end for
20: end if
21: end for
22: final\_score \leftarrow \text{mean}(\text{score})
23: return final\_score
24: end procedure
```
an inlier by explaining the inlier class (for local explanations). For explaining what feature values are normally associated with outlier datapoints (global explanations) these explanations will target the outlier class. This is why hypothesis 1 checks the contribution of RBF kernel for grouping outliers or inliers. Thus, we will have N datasets x 2 types of datapoints. This serves for performing an hypothesis contrast based on the Wilcoxon signed-rank test [39], since it has been proved useful for comparing different ML model metrics results over several datasets for both classification [40] and regression tasks [41].

4.1. Datasets

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

- Datasets 1 and 2 are about seismic activity [42]. Dataset 1 is bi-dimensional with only numerical features (‘genergy’, ‘gdpuls’). Dataset 2 has 2 categorical features (‘hazard’, ‘shift’) and 7 numerical (‘seismoaoustic’, ‘shift’, ‘genergy’, ‘gplus’, ‘genergy’, ‘gdpuls’, ‘hazard’, ‘bumps’, ‘bumps2’).
- Dataset 3 is about cardiovascular diseases [43]. There are 4 categorical features (‘smoke’, ‘alco’, ‘active’, ‘is_man’) and 7 numerical (‘age’, ‘height’, ‘weight’, ‘ap_hi’, ‘ap_lo’, ‘cholesterol’, ‘gluc’).
- Dataset 4 is from a call center at Telefónica (TEF Comms). 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 5 contains Telefónica’s data about IoT devices attached to cars for vehicle tracking. The data is aggregated in daily windows for each vehicle, representing features that model the daily behaviour of that vehicle. It contains 49 numerical features (such as the number of events with high RPM or the maximum temperature of the coolant), and 12 categorical ones (binary variables that indicate the model and make of that car, among others).
- Dataset 6 refers to US census for year 1990 [44]. It has 2 categorical features (‘dAnxtry1_3’, ‘dAnxtry1_4’) and 7 numerical ones (‘dAge’, ‘iYearSch’, ‘iFertil’, ‘iImmigr’, ‘iYearwrk’, ‘dTravtime’, ‘dRearning’).

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 [45]. The rest of the code described in Algorithms 1 and 2 were developed from scratch, and available in Github [46], together with the algorithms used for measuring the "stability" and the "diversity" of the rules.

4.2. Model Configuration

OCSVM models use as hyperparameters: \( \nu = 0.1 \), \( \gamma = 0.1 \), \( \text{kernel} = \text{rbf} \) or \( \nu = 0.1 \), \( \gamma = 0.1 \), \( \text{kernel} = \text{linear} \) for linear kernel. K-means++ models use \( \text{max} = 100 \), \( n_{\text{init}} = 10 \), \text{randomState} = 0. K-Prototypes uses \( \text{init} = 'Huang' \), \( n_{\text{iter}} = 5 \), \( n_{\text{init}} = 5 \). DT uses default parameters, with \text{randomState} = 42 and Gini criterion to find the best splits. All Protodash applications use \( \text{kernelType} = 'Gaussian' \), \( \text{sigma} = 2 \), with \( m = 1000 \) for the samples used in the Anchors rule extraction step, and \( m = \text{len}(\text{rules}) \) for the computation of metrics, having \( m \) at least a value of 20. RuleFit uses \( \text{tree}_\text{size} = \text{len}(\text{feature_cols}) * 2 \), \( \text{rfmode='classify'} \) with \( \text{len}(\text{feature_cols}) \) the number of features that appear in each dataset. RuleFit also considers only rules with a non zero coefficient, and with an importance > 0. For SkopeRules, since we want only P@1 rules, we use \( \text{random_state} = 42 \), \( \text{precision}_\text{min} = 1.0 \), \( \text{recall}_\text{min} = 0.0 \). FRL and Anchors use both their default library parameters. BRLG uses \( \text{lambda} = 1e - 3 \), \( \text{lambda}_1 = 1e - 3 \), \( \text{CNF} = False \). LOGRR uses \( \text{lambda}_0 = 0.005 \), \( \text{lambda}_1 = 0.001 \), \( \text{useOrd} = True \). GRLM uses \( \text{maxSolverIter} = 2000 \) considering only coefficients with value > 0.

4.3. Results

In this Section we check our hypotheses. We will refer to K-Means approach as KM, and K-Prototypes as KP. Thus, for instance, K-Means with the "split" method will be identified as KM_split.

Figure 14 and Table 1 provide the results associated to hypothesis 1. Here, we want to check if there are significantly less P@1 rules for inliers using a RBF kernel, compared to using a linear kernel for inliers, or the same RBF kernel for outliers. For the Wilcoxon signed-rank tests we will compare only combinations of method-kernel-inliers/outliers for datasets that have at least 1 P@1 rule (since, as Figure 13 shows, not every method in every dataset is able to yield P@1 rules). Since the comparisons involve few datapoints in some cases, we check against a minimum p-value of 0.1. Considering this, only KM_split and KM_keep have signif-

| Dataset | Ref. | Nº Cat. | Nº Num. | Nº Rows |
|---------|------|---------|---------|---------|
| 1       | [42] | 0       | 2       | 669     |
| 2       | [42] | 2       | 7       | 1705    |
| 3       | [43] | 4       | 7       | 42000   |
| 4       | TEF Comms | 0     | 5       | 2712    |
| 5       | TEF Fleet | 12    | 49      | 59844   |
| 6       | [44] | 2       | 7       | 106819  |

Table 1: Description of each dataset, with their reference (Ref.), categorical features (Nº Cat.), numerical features (Nº Num) and number of rows.
significant differences in the number of rules. In those cases, H1 is actually rejected: RBF for inliers yields more rules than either RBF for outliers, or linear for inliers. Regarding the other methods, there are no statistically strong results to conclude anything. This results are shown in Table 2. Figure 13 shows how at datasets D3 and/or D5 (the ones with more features) the number of rules abruptly increase for some of the methods (“split” ones and KM_keep_reset). With that, even though it is not assured for every method, these rule extraction methods when applied to inliers and when using an RBF kernel tend to generate more rules than in the other cases.

After comparing those rule extraction methods in terms of the number of rules in order to see significant differences depending on the type of datapoints (inliers/outliers) and the type of kernel (rbf/linear), we proceed to check H2. Here, we compare the methods considering all the XAI metrics proposed previously. This is done by checking every metric over every combination of dataset, kernel and type of data (inliers/outliers), and performing a Wilcoxon signed-rank test in order to see if there are no significant differences between the methods for each of the metrics. Since the datapoints in this case are superior than those present at H1, we check against a minimum p-value of 0.05. For H2 there is no need to check the size of the rules since they will be the same for all the methods using K-Means and for all the methods using K-Prototypes. We only compare between datasets-kernel-type of data that exists in both methods considered. Thus, the means for the KM methods may vary depending on whether they are compared between them or they are compared against KP ones (and vice versa).

At Table 3 we see the methods and metrics that have significant differences according to Wilcoxon signed-rank test. There are some cases where the metrics do differ significantly, as some methods yield better results. This is the case of KM_split. This method outperforms every other one regarding the percentage of datapoints covered by its P@1 rules (per_p1). It does so in exchange of yielding a greater number of rules than some of the other methods. Thus, it increases "representativeness" by losing in terms of "comprehensibility". In general, KM methods cover more datapoints with P@1 rules that their counterparts with KP. Considering the other metric from "representativeness", p1_coverage, there are no significant differences between KM_split and KM_keep_reset, but both methods yield better results than KM_keep. Thus, usually the P@1 rules that they yield are able to cover more datapoints. This is logical, since the algorithm that yields the rules in the case of KM_keep tends to generate smaller hypercubes. An example of this can be seen in Figure 11 for dataset D1. We can see how KM keep indeed yields rules that are smaller than the ones from the other methods.

Regarding "representativeness" (diversity_score), there are no significant differences between KM methods, but all of them outperform all the KP ones. In terms of "stability", we see no significant difference between any of the methods. Finally, the general metric (final_metric), shows that actually KM_keep outperform KM_split, and KM_keep_reset. Thus, even though KM_keep had worse results in terms of "representativeness" than the other KM methods, it is compensated by the other metrics. With this analysis, we see that KM methods appear to be better than KP ones for P@1 rules and for explaining anomalies over a OCSVM model. However, KM methods are more contested; they seem to have similar results in some metrics (KM_keep_reset and KM_discard are very similar between them), while being different in others (mainly compared to KM_keep in terms of "representativeness"). Thus, H2 is partially supported.

A visual analysis for the metrics is provided at Figure 14. It shows the XAI metrics aforementioned for the clustering-based rule extraction methods and for every combination of dataset, kernel and type of data considered in this paper.

Finally, we check H3. Since the techniques compared for H2 yield similar results, we will focus only in KM_split and KM_keep, and benchmark them against the remaining rule extraction techniques covered in this paper. Figure 12 shows visualizations for some of these methods over D1 (when using a RBF kernel).
The results appear in Table 5. Here, we see how KM_split is generally better for every metric except for the ones related to "comprehensibility". In particular, KM_split is able to significantly cover more data points from the target class with P@1 rules (per_p1) than any of the other methods, and also yields rules that have better coverage (p1_coverage) than FRL and brlg. However, the mean coverage per rule compared to the other methods is not significantly different. Regarding "stability", KM_split outperforms brlg, but brlg has a better result in terms of "diversity". KM_split improves FRL and Anchors in "stability", and DT in "diversity". Finally, considering the general metric, KM_split has significantly better results than any of the other methods, with the exception of DT, where it does not show significant differences. Considering KM_keep, the results are similar, as shown in Table 5. One difference is that since KM_keep has a better general metric than KM_split, it is also able to significantly outperform DT in that aspect. Also, KM_keep is not outperformed in terms of "diversity" by brlg, as opposed to KM_split.

As a conclusion, we see that both the solution of [10] with K-Means++ (and with the modification for generating rules for categorical features), together with the variations considered in this paper (for also K-Means++ as clustering method) yield similar results in terms of most of the XAI metrics considered in this paper for explaining the results of a OCSVM anomaly detection model using P@1 rules. The results in terms of comprehensibility (number of rules) are influenced depending on the type of kernel used, and whether they are explaining inliers or outliers. Finally, comparing these techniques with other rule extraction methods, we saw a trade-off between comprehensibility and the remaining XAI metrics. The clustering-based rule extraction techniques used in this paper are able to explain better with P@1 rules the results of a OCSVM model (considering the datasets and kernels of this paper) in terms of "representativeness", "stability" and "diversity", but in exchange of "comprehensibility", which is penalized.

### 4.4. Software Used

The main libraries used for the work done in this paper are the following:

- OCSVM, DT [47]
- Anchors [48]
- Protodash, GRLM, BRLG [21]
- RuleFit [49]
- SkopeRules [50]
- FRL [51]

As we mentioned before, we also provide a repository with all the developed algorithms [46].

### 5. Conclusion

In this paper we have analysed the application of XAI techniques over unsupervised outlier detection models through the usage of rule extraction methods applied to OCSVM models. Among the rule extraction techniques, we used both algorithms from the literature, as well as new alternatives that we propose and evaluate together with them. Our first aim was analysing the quality of the rules extracted from a XAI perspective. We have done this by defining metrics for different aspects related to XAI: comprehensibility, representativeness, stability and diversity, as well as proposing a function to aggregate all those metrics together. We evaluated those metrics over different datasets, both from public sources as well as from Telefónica’s, using communications and IoT generated data for that purpose. The results for the metrics show that clustering-based techniques yield results that are similar to each other (when using K-Means++ clustering). When comparing these methods with other rule extraction techniques over different datasets and different kernels, we saw that when working with P@1 rules, the clustering-based methods yielded better results in terms of "representativeness", "stability" and "diversity", in exchange of "comprehensibility", which is penalized by using more rules with more size than other methods.

Our evaluation considered model-agnostic techniques that can be applied over any black-box model. In order to check this empirically, we have used OCSVM models with different types of kernel configurations (linear and RBF). We saw how, indeed, all rule extraction techniques provide similar results regardless of the kernel used.

### 5.1. Limitations of our Approach

Regarding the XAI metrics and evaluations, the first limitation to consider is that the only model used is OCSVM (with two types of kernel). Even though the algorithms used and the metric definitions are model-agnostic and may be potentially applied over other outlier detection models, the results may differ if other unsupervised models or kernels are used. Together with this, our suggestion of a function that aggregates every metric is a simple baseline that can be further improved.

Also, the analyses carried out are focused in P@1 rules. Thus, the conclusions may be different if all the rules extracted are used, regardless of their precision value.

Finally, all the rules (for cluster-based methods) and all the checking to see if a data point is inside a hypercube (for all methods) are defined with inequalities ($\leq$, $\geq$). Because of that, the results may be different if we allow values from the other class to be at the limit of the hypercube.

### 5.2. Future Work

There are several research lines that can be pursued following the work presented at this paper. The first one to consider is benchmarking these results against other rule extraction techniques that are not covered in this paper. An example is G-Rex algorithms [52]. Another research line that can be followed is analysing the metrics of the rule extraction techniques over other unsupervised anomaly detection models, such as IsolationForests [53] or LOF [54], as well as using other kernel configurations in OCSVM (such as a polynomial one). Also, while we have proposed a vanilla function to incorporate the metrics belonging to different XAI areas, there is much room of improvement over it in order to find an
optimal function that weights appropriately every term. Finally, rule extraction should also be designed to consider all types of comparisons (\(\geq\), \(\leq\), \(>\) and \(<\)), and this is something that could also be considered in the cluster-based methods developed.

References

[1] Alejandro Barredo Arrieta, Natalia Diáz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI”. In: Information Fusion 58 (2020), pp. 82–115.

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

[3] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. “Explanation 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.

[4] Shane T Mueller, Robert R Hoffman, William Clancey, Abigail Emrey, and Gary Klein. “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).

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

[6] Victoria Hodge and Jim Austin. “A survey of outlier detection methodologies”. In: Artificial intelligence review 22.2 (2004), pp. 85–126.

[7] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. “A unifying review of deep and shallow anomaly detection”. In: Proceedings of the IEEE (2021).

[8] Rocco Langone, Alfredo Cuzzocrea, and Nikolaos Skantzos. “Interpretable Anomaly Prediction: Predicting anomalous behavior in industry 4.0 settings via regularized logistic regression tools”. In: Data & Knowledge Engineering 130 (2020), p. 101850.

[9] T Maruthi Padma and P Jhanji Lakshmi. “A hybrid rule extraction method for one-class support vector machines”. In: 2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO). IEEE. 2015, pp. 1–4.

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

[11] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. “Support vector method for novelty detection”. In: Advances in neural information processing systems. 2000, pp. 582–588.

[12] Yanxin Wang, Johnny Wong, and Andrew Minar. “Anomaly intrusion detection using one class SVM”. In: Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004. IEEE. 2004, pp. 358–364.

[13] Nahla Barakat and Andrew P Bradley. “Rule extraction from support vector machines: a review”. In: Neurocomputing 74.1-3 (2010), pp. 178–190.

[14] Nahla H Barakat and Andrew P Bradley. “Rule extraction from support vector machines: A sequential covering approach”. In: IEEE Transactions on Knowledge and Data Engineering 19.6 (2007), pp. 729–741.

[15] Xiuju Fu, ChongJin Ong, Sathiya Keerthi, Gih Guang Hung, and Liping Goh. “Extracting the knowledge embedded in support vector machines”. In: 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541). Vol. 1. IEEE. 2004, pp. 291–296.

[16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “Anchors: High-precision model-agnostic explanations”. In: Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[17] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “Why should i trust you?” Explaining the predictions of any classifier”. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016, pp. 1135–1144.

[18] Jerome H Friedman, Bogdan E Popescu, et al. “Predictive learning via rule ensembles”. In: The Annals of Applied Statistics 2.3 (2008), pp. 916–954.

[19] Fulton Wang and Cynthia Rudin. “Falling rule lists”. In: Artificial Intelligence and Statistics. 2015, pp. 1013–1022.

[20] Sanjeeb Dash, Oktay Gunluk, and Dennis Wei. “Boolean Decision Rules via Column Generation”. In: Advances in Neural Information Processing Systems 31. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 4655–4665. URL: http://papers.nips.cc/paper/7716-boolean-decision-rules-via-column-generation.pdf

[21] Vijay Arya, Rachel K. E. Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Q. Vera Liao, Ronny Luss, Aleksandra Mojsilović, Sami Mourad, Pablo Pedemonte, Ramya Raghavendra, John Richards, Prasanna Sattigeri, Karthikkeyan Shanmugam, Moninder Singh, Kush R. Varshney, Dennis Wei, and Yunfeng Zhang. One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Tech-
[22] Dennis Wei, Sanjeeb Dash, Tian Gao, and Oktay Gunluk. “Generalized Linear Rule Models”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalkica Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, Sept. 2019, pp. 6687–6696. URL: [http://proceedings.mlr.press/v97/wei19a.html](http://proceedings.mlr.press/v97/wei19a.html)

[23] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. “One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques”. In: *arXiv preprint arXiv:1909.03012* (2019).

[24] AJ Tallón-Ballesteros and C Chen. “Explainable AI: Using Shapley Value to Explain Complex Anomaly Detection ML-Based Systems”. In: *Machine Learning and Artificial Intelligence: Proceedings of MLIS 2020* 332 (2020), p. 152.

[25] Tomohiro Mitani, Shunsuke Doi, Shinichiroh Yokota, Takeshi Imai, and Kazuhiko Ohe. “Highly accurate and explainable detection of specimen mix-up using a machine learning model”. In: *Clinical Chemistry and Laboratory Medicine (CCLM)* 58.3 (2020), pp. 375–383.

[26] Jacob Kauffmann, Klaus-Robert Müller, and Grégoire Montavon. “Towards explaining anomalies: a deep Taylor decomposition of one-class models”. In: *Pattern Recognition* (2020), p. 107198.

[27] Sarah Itani, Fabien Lecron, and Philippe Fortemp. “A one-class classification decision tree based on kernel density estimation”. In: *Applied Soft Computing* 91 (2020), p. 106250.

[28] Gyeong Jang and Sung-bae Cho. “The Anomaly Detection of 2.4 L Diesel Engine Using One-Class SVM with Variational Autoencoder”. In: *Annual Conference of the PHM Society*. Vol. 11. 1. 2019.

[29] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. “Machine Learning Interpretability: A Survey on Methods and Metrics”. In: *Electronics* 8.8 (2019), p. 832.

[30] Mario Popolin Neto and Fernando V Paulovich. “Explainable Matrix—Visualization for Global and Local Interpretability of Random Forest Classification Ensembles”. In: *IEEE Transactions on Visualization and Computer Graphics* (2020).

[31] Luca Longo. “A comparative analysis of rule-based, model-agnostic methods for explainable artificial intelligence”. In: ()

[32] David Alvarez-Melis and Tommi S Jaakkola. “On the robustness of interpretability methods”. In: *arXiv preprint arXiv:1806.08049* (2018).

[33] The European Commission. *White Paper On Artificial Intelligence - A European approach to excellence and trust*. The European Commission, Feb. 2014. URL: [https://ec.europa.eu/info/sites/info/files/commission-white-paper-artificial-intelligence-feb2020_en.pdf](https://ec.europa.eu/info/sites/info/files/commission-white-paper-artificial-intelligence-feb2020_en.pdf)

[34] Sam Wilkinson. “Levels and kinds of explanation: lessons from neuropsychiatry”. In: *Frontiers in psychology* 5 (2014), p. 373.

[35] Frank C Keil. “Explanation and understanding”. In: *Annu. Rev. Psychol.* 57 (2006), pp. 227–254.

[36] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Tech. rep. Stanford, 2006.

[37] Jinchao Ji, Tian Bai, Chenguang Zhou, Chao Ma, and Zhe Wang. “An improved k-prototypes clustering algorithm for mixed numeric and categorical data”. In: *Neurocomputing* 120 (2013), pp. 590–596.

[38] Karthik S Gurumoorthy, Amit Dhurandhar, Guílmero Cecchi, and Charu Aggarwal. “Efficient Data Representation by Selecting Prototypes with Importance Weights”. In: 2019 *IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 260–269.

[39] William Jay Conover. *Practical nonparametric statistics*. Vol. 350. John Wiley & Sons, 1998.

[40] Janez Demšar. “Statistical comparisons of classifiers over multiple data sets”. In: *The Journal of Machine Learning Research* 7 (2006), pp. 1–30.

[41] Bogdan Trawiński, Magdalena Smętek, Zbigniew Telec, and Tadeusz Lasota. “Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms”. In: *International Journal of Applied Mathematics and Computer Science* 22.4 (2012), pp. 867–881.

[42] 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.

[43] Meghana Padmanabhan, Pengyu Yuan, Govind Chada, and Hien Van Nguyen. “Physician-friendly machine learning: A case study with cardiovascular disease risk prediction”. In: *Journal of clinical medicine* 8.7 (2019), p. 1050.

[44] Catherine Blake. “UCI repository of machine learning databases”. In: [http://www.ics.uci.edu/mlrepository.html](http://www.ics.uci.edu/mlrepository.html) (1998).

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
Alberto Barbado. Rule extraction in unsupervised outlier detection for XAI. [https://github.com/AlbertoBarbado/unsupervised-outlier-transparency]. 2019.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. “API design for machine learning software: experiences from the scikit-learn project”. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning. 2013, pp. 108–122.

Janis Klaise, Arnaud Van Looveren, Giovanni Vacanti, and Alexandru Coca. Alibi: Algorithms for monitoring and explaining machine learning models. Version 0.3.2. Feb. 17, 2020. URL: https://github.com/SeldonIO/alibi

Christoph Molnar. RuleFit. Version 0.2. Nov. 24, 2017. URL: https://github.com/christophM/rulefit

SkopeRules. URL: https://github.com/scikit-learn-contrib/skope-rules

Tong Wang Zhen Li. Bayesian Rule Set Mining. URL: https://pypi.org/project/ruleset/

Rikard Konig, Ulf Johansson, and Lars Niklasson. “G-REX: A versatile framework for evolutionary data mining”. In: 2008 IEEE International Conference on Data Mining Workshops. IEEE, 2008, pp. 971–974.

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.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. “LOF: identifying density-based local outliers”. In: ACM sigmod record. Vol. 29. 2. ACM. 2000, pp. 93–104.

David Martens, Johan Huysmans, Rudy Setiono, Jan Vanthienen, and Bart Baesens. “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.

Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martin, Sameep Mehta, Aleksandra Mojsilovic, et al. “AI fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias”. In: arXiv preprint arXiv:1810.01943 (2018).

Pedro Saleiro, Benedict Kuester, Abby Stevens, Ari Anisfeld, Loren Hinkson, Jesse London, and Rayid Ghani. “Aequitas: A Bias and Fairness Audit Toolkit”. In: arXiv preprint arXiv:1811.05577 (2018).

Till Speicher, Hoda Heidari, Nina Grgic-Hlaca, Krishna P. Gummadi, Adish Singla, Adrian Weller, and Muhammad Bilal Zafar. “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

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. 3323–3333. ISBN: 978-1-5108-3881-9. URL: http://dl.acm.org/citation.cfm?id=3157382.3157469

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

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.

Anil Chaturvedi, Paul E Green, and J Douglas Carroll. “K-modes clustering”. In: Journal of classification 18.1 (2001), pp. 35–55.

Arthur Zimek and Peter Filzmoser. “There and back again: Outlier detection between statistical reasoning and data mining algorithms”. In: Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8.6 (2018), e1280.

Vic Barnett. Outliers in statistical data. Tech. rep. 1978.

Richard J Beckman and R Dennis Cook. “Outlier . . . . . . . s”. In: Technometrics 25.2 (1983), pp. 119–149.

Satyananda Kashyap, Alexandros Karargyris, Joy Wu, Yaniv Gur, Arjun Sharma, Ken CL Wong, Mehdi Moradi, and Tanveer Syeda-Mahmood. “Looking in the Right Place for Anomalies: Explainable AI Through Automatic Location Learning”. In: 2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI). IEEE. 2020, pp. 1125–1129.

Irad Ben-Gal. “Outlier detection”. In: Data mining and knowledge discovery handbook. Springer, 2005, pp. 131–146.
6. Annex

Figure 13: Number of P@1 rules per clustering-based method and per dataset. K-prototypes methods are not used for D1 and D5 since those datasets do not have categorical columns (there would not be significant differences between the methods).

| Method | RBF - Inliers mean | RBF - Outliers mean | Linear - Inliers mean |
|--------|--------------------|---------------------|-----------------------|
| KM_split | 396.2              | 242.3               | 221.2 (0.09)          |
| KM_keep_rest | 166.7              | 222.7               | 155.8 (0.84)          |
| KM_keep | 154.3              | 67.5 (0.03)         | 67.3 (0.03)           |
| KP_split | 548.3              | 20                  | 73.5 (0.625)          |
| KP_keep_rest | 9.5                | 7                   | 13 (0.5)              |
| KP_keep | 14                 | 9                   | 19 (0.75)             |

Table 2: Comparison of the number of rules generated by the different clustering-based rule extraction methods between RBF kernel for inliers, and RBF kernel for outliers or linear kernel for inliers.
Figure 14: XAI metrics for the clustering-based rule extraction methods described in this paper. The metrics include representativeness, stability and diversity, as well as a global metric.
| method 1   | method 2        | metric      | mean 1 | mean 2 | p-value |
|-----------|----------------|-------------|--------|--------|---------|
| KP_split  | KP_keep        | n_rules     | 213.92 | 9.75   | 0.0034  |
| KM_split  | KM_keep_reset  | n_rules     | 286.56 | 181.72 | 0.0342  |
| KM_keep   | KP_keep_reset  | n_rules     | 259.75 | 5.42   | 0.0005  |
| KM_split  | KP_keep_reset  | n_rules     | 407.08 | 9.75   | 0.0001  |
| KM_keep   | KP_keep        | n_rules     | 259.75 | 9.75   | 0.0024  |
| KM_split  | KM_keep        | per_p1      | 0.83   | 0.54   | 0.0004  |
| KM_keep   | KP_keep        | per_p1      | 0.58   | 0.28   | 0.021   |
| KM_split  | KP_keep        | per_p1      | 0.69   | 0.18   | 0.0015  |
| KM_keep   | KP_keep_reset  | per_p1      | 0.69   | 0.28   | 0.0161  |
| KM_split  | KM_keep        | per_p1      | 0.54   | 0.76   | 0.0312  |
| KM_split  | KP_keep        | per_p1      | 0.75   | 0.18   | 0.001   |
| KM_split  | KM_keep_reset  | per_p1      | 0.75   | 0.28   | 0.0093  |
| KM_split  | KM_keep        | per_p1      | 0.83   | 0.76   | 0.0231  |
| KM_split  | KM_keep        | pl_coverage | 0.06   | 0.17   | 0.0023  |
| KM_split  | KM_keep        | pl_coverage | 0.15   | 0.06   | 0.0104  |
| KM_split  | KM_keep        | diversity_score | 0.89   | 0.75   | 0.0329  |
| KM_split  | KM_keep        | diversity_score | 0.85   | 0.67   | 0.0005  |
| KM_split  | KM_keep        | diversity_score | 0.88   | 0.67   | 0.0005  |
| KM_split  | KM_keep        | final_metric | 0.54   | 0.61   | 0.0     |
| KM_split  | KM_keep        | final_metric | 0.61   | 0.55   | 0.0001  |

Table 3: Wilcoxon signed-rank hypothesis contrast for the methods and metrics where there are significant differences.
| method 1   | method 2   | metric       | mean 1 | mean 2 | p-values |
|-----------|-----------|--------------|--------|--------|----------|
| KM_split  | DT        | n_rules      | 286.56 | 7.72   | 0.0      |
| KM_split  | FRL       | n_rules      | 286.56 | 2.17   | 0.0      |
| KM_split  | SkopeRules| n_rules      | 286.56 | 2.83   | 0.0      |
| KM_split  | RuleFit   | n_rules      | 286.56 | 21.44  | 0.0028   |
| KM_split  | Anchors   | n_rules      | 286.56 | 20.44  | 0.0      |
| KM_split  | brlg      | n_rules      | 286.56 | 0.22   | 0.0      |
| KM_split  | brlg      | size_rules   | 29.0   | 0.22   | 0.0      |
| KM_split  | RuleFit   | size_rules   | 29.0   | 3.33   | 0.0      |
| KM_split  | DT        | size_rules   | 29.0   | 5.72   | 0.0003   |
| KM_split  | Anchors   | size_rules   | 29.0   | 3.39   | 0.0      |
| KM_split  | SkopeRules| size_rules   | 29.0   | 1.69   | 0.0      |
| KM_split  | FRL       | size_rules   | 29.0   | 2.17   | 0.0      |
| KM_split  | DT        | per_p1       | 0.83   | 0.28   | 0.0008   |
| KM_split  | Anchors   | per_p1       | 0.83   | 0.14   | 0.0      |
| KM_split  | SkopeRules| per_p1       | 0.83   | 0.22   | 0.0001   |
| KM_split  | RuleFit   | per_p1       | 0.83   | 0.45   | 0.0031   |
| KM_split  | FRL       | per_p1       | 0.83   | 0.12   | 0.0      |
| KM_split  | brlg      | per_p1       | 0.83   | 0.02   | 0.0      |
| KM_split  | FRL       | pl_coverage  | 0.15   | 0.08   | 0.0442   |
| KM_split  | brlg      | pl_coverage  | 0.15   | 0.02   | 0.0007   |
| KM_split  | FRL       | precision_vs_model | 0.58 | 0.26 | 0.004 |
| KM_split  | brlg      | precision_vs_model | 0.58 | 0.04 | 0.0 |
| KM_split  | Anchors   | precision_vs_model | 0.58 | 0.35 | 0.0268 |
| KM_split  | DT        | diversity_score | 0.88 | 0.59 | 0.0432 |
| KM_split  | brlg      | diversity_score | 0.88 | 0.89 | 0.0268 |
| KM_split  | RuleFit   | final_metric | 0.54 | 0.27 | 0.0005 |
| KM_split  | SkopeRules| final_metric | 0.54 | 0.33 | 0.0003 |
| KM_split  | FRL       | final_metric | 0.54 | 0.21 | 0.0003 |
| KM_split  | Anchors   | final_metric | 0.54 | 0.26 | 0.0007 |
| KM_split  | brlg      | final_metric | 0.54 | 0.04 | 0.0 |

Table 4: Wilcoxon signed-rank hypothesis contrast for the methods and metrics where there are significant differences, comparing KM_split method against the remaining rule-extraction techniques covered in this paper.
| method 1      | method 2      | metric         | mean 1 | mean 2 | p-value |
|--------------|---------------|----------------|--------|--------|---------|
| KM_keep Anchors | n_rules       | 122.25        | 29.25  | 0.0005 |
| KM_keep RuleFit | n_rules       | 122.25        | 18.08  | 0.0015 |
| KM_keep brlg   | n_rules       | 122.25        | 0.22   | 0.0005 |
| KM_keep SkopeRules | n_rules     | 122.25        | 3.42   | 0.0005 |
| KM_keep DT     | n_rules       | 122.25        | 3.0    | 0.0005 |
| KM_keep FRL    | n_rules       | 122.25        | 2.08   | 0.0005 |
| KM_keep RuleFit | size_rules    | 40.0          | 2.75   | 0.0005 |
| KM_keep Anchors | size_rules    | 40.0          | 3.92   | 0.0005 |
| KM_keep FRL    | size_rules    | 40.0          | 1.67   | 0.0005 |
| KM_keep DT     | size_rules    | 40.0          | 5.25   | 0.0005 |
| KM_keep SkopeRules | size_rules   | 40.0          | 2.12   | 0.0005 |
| KM_keep brlg   | size_rules    | 40.0          | 0.22   | 0.0005 |
| KM_keep Anchors | per_p1       | 0.48          | 0.03   | 0.0005 |
| KM_keep DT     | per_p1       | 0.48          | 0.03   | 0.001  |
| KM_keep brlg   | per_p1       | 0.48          | 0.02   | 0.0005 |
| KM_keep FRL    | per_p1       | 0.48          | 0.1    | 0.0005 |
| KM_keep SkopeRules | per_p1      | 0.48          | 0.19   | 0.021  |
| KM_keep brlg   | p1_coverage  | 0.04          | 0.02   | 0.0005 |
| KM_keep Anchors | p1_coverage  | 0.04          | 0.0    | 0.0033 |
| KM_keep FRL    | precision_vs_model | 0.5  | 0.22  | 0.0425 |
| KM_keep brlg   | precision_vs_model | 0.5  | 0.04  | 0.0005 |
| KM_keep DT     | diversity_score | 0.89 | 0.43  | 0.0068 |
| KM_keep brlg   | diversity_score | 0.89 | 1.0   | 0.0051 |
| KM_keep FRL    | final_metric  | 0.61          | 0.18   | 0.001  |
| KM_keep SkopeRules | final_metric | 0.61          | 0.43   | 0.0015 |
| KM_keep RuleFit | final_metric  | 0.61          | 0.16   | 0.0005 |
| KM_keep DT     | final_metric  | 0.61          | 0.35   | 0.0269 |
| KM_keep Anchors | final_metric  | 0.61          | 0.21   | 0.001  |
| KM_keep brlg   | final_metric  | 0.61          | 0.0    | 0.0005 |

Table 5: Wilcoxon signed-rank hypothesis contrast for the methods and metrics where there are significant differences, comparing KM_keep method against the remaining rule-extraction techniques covered in this paper.