Explainable Clustering with CREAM

Federico Sabbatini[1], **Roberta Calegari**[2]
[1]Department of Pure and Applied Sciences (DiSPeA), University of Urbino
[2]Department of Computer Science and Engineering (DISI), University of Bologna
f.sabbatini1@campus.uniurb.it, roberta.calegari@unibo.it

Abstract

This paper proposes CREAM, a new explainable clustering technique based on decision tree induction, providing human-interpretable clusters by performing hypercubic approximations of the input feature space. CREAM may also be applied to data sets describing classification and regression tasks, given that the algorithm discriminates amongst input and output features. We also present ORCHID, an automated tuning procedure to select the optimum CREAM parameters. Experiments demonstrating the effectiveness of CREAM in clustering, classification, and regression tasks are reported here, in comparison with other state-of-the-art techniques used as benchmarks.

1 Introduction

Supervised classification and regression as well as unsupervised clustering are amongst the core machine learning (ML) predictive techniques adopted in real-world, daily applications. Optimising these methods, and consequently increasing their algorithmic complexity, allows users to obtain impressive results, but at the expense of human interpretability and related explainability[1]. That is why ML models are often named *black boxes* (BBs) or *opaque models*, conversely to transparent models. The common trade-off between predictive performance and readability is often a hindering factor when BB models are required in critical applications involving, for instance, human health, wealth, freedom, and safety since it is crucial for human users to understand the rationale behind ML predictions. The increasing demand for transparency (European Commission, Directorate-General for Communications Networks, and Technology 2019; European Commission 2021) is reflected in a lack of trust for BB-based systems and in the consequent adoption of inspectable alternatives, possibly not capable of as much predictive power.

[1]It is worth emphasising that symbolic knowledge-extraction methods enable interpretability, which involves understanding an artificial intelligence model's internal workings. Explainability provides understandable justifications for the model's outputs, bridging complexity and human comprehension. Interpretability acts as a fundamental precursor to explainability by providing the necessary understanding to generate transparent explanations for the model's decisions and predictions

In the field of clustering several interpretable approaches have been proposed in substitution of classical, opaque techniques (Dasgupta et al. 2020; Moshkovitz et al. 2020; Manduchi et al. 2021; Chen and Güttel 2022, to mention some recent examples).

In this paper we present CREAM, a novel interpretable clustering technique based on tree induction and hypercubic approximation of clusters. It is inspired by ExACT (Sabbatini and Calegari 2023), since they share a common hierarchical nature and they both take advantage of Gaussian mixture models (GMMs) and DBSCAN (Ester et al. 1996), but CREAM is able to outperform its predecessor thanks to different splitting criteria when performing the hypercubic approximation of the input feature space. More in detail, CREAM is not bounded to an underlying unbalanced tree and therefore it produces more accurate output knowledge having refined rules. Our contribution is relevant for two different reasons. First, CREAM proved to be a valid clustering tool, for the benefit of the currently slim category of interpretable clustering techniques. Second, for its design, it is perfectly suitable to replace BB classifiers and regressors in order to possibly enhance the human readability of the predictions they provide. The paper also presents ORCHID, a tuning algorithm to select the best values to provide as CREAM parameters. Accordingly, in Section 2 background information is provided. In Section 3 the CREAM and ORCHID algorithms are presented and in Section 4 experiments to assess the effectiveness of CREAM are reported. Conclusions are drawn in Section 5.

2 Related Works

2.1 Traditional Clustering

Several clustering techniques are currently present in the literature, each one with its own peculiarities, usually bound to the clusters' properties, such as shape or density. As a consequence, there are no optimum algorithms showing better predictive performance over all the others in any possible applicative scenario. Amongst the most performing techniques there are GMMs (Murphy 2012), DBSCAN (Ester et al. 1996; Ling 1972) and DBSCAN++ (Jang and Jiang 2019), OPTICS (Ankerst et al. 1999), BIRCH (Zhang, Ramakrishnan, and Livny 1996), k-means (Lloyd 1982), Mean shift (Cheng 1995) and spectral clustering (Shi and Malik

2000; Yu and Shi 2003). In the following paragraphs, we analyse in detail only the traditional clustering techniques constituting the core of CREAM.

Gaussian Mixture Models GMMs proved to be flexible and powerful solutions to perform clustering. GMMs are based on the assumption that the data set points' generation follows the mixture of a finite number of Gaussian distributions. The goal of GMMs is to determine the parameters of these distributions. Given that these models are probabilistic and that each GMM prediction is associated with a probability score, also soft clustering is enabled. W.r.t. other algorithms, for instance, k-means, GMMs are not bounded to spherical clusters and thus they are preferred for general applications. The number of Gaussian components adopted during the GMM training is the most important hyper-parameter to be tuned and it heavily impacts the overall predictive performance of the clustering. Automated tuning of this parameter is possible through the Bayes information criterion (BIC), by training several GMM instances with different component amounts, calculating the corresponding BIC score for each instance, and finally picking the one associated with the lowest BIC score value.

DBSCAN A different approach is considered by DBSCAN (Density-Based Spatial Clustering of Applications with Noise), a procedure able to identify arbitrarily-shaped clusters of data through a density criterion. In this case, a user-defined parameter (ε) defining the maximum allowed intra-cluster distance should be provided. ε is strictly task-dependent. Being a pivotal parameter for the DBSCAN performance, automated tuning procedures have been proposed to determine ε without user efforts (Rahmah and Sitanggang 2016). DBSCAN is also a suitable tool to remove outliers from clusters identified by other algorithms.

2.2 Explainable Clustering

Explanation of clusters is acquiring relevance amongst researchers, especially if the clustering involves critical areas such as medical or financial (Manduchi et al. 2021, for instance). Several existing techniques induce an underlying decision tree (Basak and Krishnapuram 2005; Fraiman, Ghattas, and Svarc 2013; Bertsimas, Orfanoudaki, and Wiberg 2018; Dasgupta et al. 2020) according to 2 different approaches. In the top-down approach, more commonly adopted, first the tree root containing the whole data set is built, then it is recursively partitioned until some stopping criteria are satisfied. The bottom-up strategy, conversely, tries to group together individual data set samples to form the final clusters. All tree-based clustering techniques perform input space partitioning via cutting hyperplanes that are perpendicular to the most relevant input features, considering one feature per cut.

A different approach proposed by Chen et al. (2016) and based on a rectangular input space partitioning enables a higher human-interpretability extent given that each cluster is described in terms of 2 interval inclusions. However, the algorithm may merge existing input features into new combined features, thus hindering explainability for the sake of conciseness. In a recent work Chen and Güttel (2022) presents an interpretable density-based clustering technique named CLASSIX.

Our proposed technique differs from the existing clustering described above since human interpretability is achieved through the description of clusters by means of hypercubic regions obtained via the induction of a top-down decision tree, similarly to ExACT (Sabbatini and Calegari 2023). No composite features are created by merging the input ones, nor these features are discarded to limit the cluster description to 2-dimensional, rectangular regions. In the following paragraphs we report the details of the explainable clustering techniques used as benchmarks for the experiments reported in Section 4.

CLASSIX CLASSIX (name defined by the authors as a contrived acronym of "CLustering by Aggregation with Sorting-based Indexing" and the letter "X" for "eXplainability"; Chen and Güttel 2022) is a fast algorithm performing explainable clustering based on two phases. The first step is a greedy aggregation aimed at grouping together nearby training data points, previously sorted. Groups are then merged into clusters in the second phase. The performance of the technique depends on two user-defined parameters, one defining the minimum amount of training samples inside each cluster and the other representing the distance between instances to be grouped together during the first phase of the algorithm.

CLASSIX offers two ways of explaining clusters. The former, global, reports the coordinates of the starting point for each group identified at the end of the aggregation phase. The other, local, enables users to understand why a single instance belongs to a specific cluster or why a pair of instances belong to the same or different clusters. This explanation is achieved by describing the steps executed during the merging phase of the algorithm.

IMM Dasgupta et al. (2020) proposed the IMM algorithm (IMM stands for Iterative Mistake Minimisation) with the aim of developing an accurate and efficient interpretable clustering algorithm based on tree induction. The induced tree is binary and each internal node is associated with a partition of the training data involving a single feature. To identify k clusters the tree grows k leaves, keeping its dimension as small as possible. The tree induction is based on the minimisation of the cluster's fragmentation, intended as the splitting of instances belonging to the same cluster into more than one subtree.

Explanations for cluster assignments are obtained by describing each assignment via the path from the tree root to the leaf corresponding to the selected cluster. Given the structure of the induced tree, a clustering identifying k clusters requires at most $k-1$ constraints to describe any assignment, since $k-1$ is the tree depth in the worst case.

ExACT ExACT (Sabbatini and Calegari 2023) combines together the aggregation strategies proper of traditional clustering techniques and the cluster assignment via decision trees as done by other interpretable clustering procedures. For this reason with ExACT it is possible to obtain explainable clusters by inducing a top-down decision tree over the training data according to a strictly hierarchical strategy. Indeed, its clusters have the peculiarity of being concentric. The strategy adopted for the tree's internal nodes is to use hypercubic splits to separate whole clusters of data while avoiding the presence of instances from multiple clusters inside the same hypercubic region.

Explainability is obtained by approximating each identified cluster with a hypercube. The concentric nature of the ExACT's hierarchical approximations enables the creation of a global interpretable clustering in the form of a rule list, where each cluster is simply expressed through a rule having a single hypercube inclusion constraint, starting from the innermost cluster through the outermost. The same structure may be used to provide local explanations for single clustering assignments.

3 Explainable Clustering with CREAM

CREAM (Clustering-based Rule Extraction Advanced Method) is a recursive algorithm performing explainable clustering via binary decision tree induction. Trees are built top-down and each internal node corresponds to a hypercube-inclusion condition since each node is associated with a hypercubic approximation of the input space region corresponding to an identified cluster of data. To achieve this goal, CREAM adopts GMMs to identify relevant clusters within the training data and DBSCAN to remove outliers possibly included inside these clusters.[2] Hypercube approximation is then applied to explain the refined clusters in human-comprehensible terms. After having built the tree, each leaf is converted into a human-interpretable rule corresponding to a cluster. Approximated clusters may have the shape of hypercubes or *difference cubes*, i.e., regions obtained by subtracting one or more hypercubes from a wider, enclosing hypercube.

The design of CREAM ensures exhaustivity of the clustering, i.e., when used to carry out predictions, each input instance will be assigned to an identified hypercubic cluster. Rules obtained by applying CREAM are ordered, disjoint, and produced by recursively traversing the tree in reverse order, that is by examining the tree nodes' right subtrees first. All approximated clusters are associated with hypercube-inclusion constraints, since clusters having the shape of difference cubes may be expressed as inclusion in an outer hypercube and exclusion from one or more inner hypercubes. Given the human interpretability extent of hypercubic regions and the possibility to tune the algorithm parameters to limit the depth of the tree induced by CREAM, it is possible to achieve explainability (Blanco-Justicia and Domingo-Ferrer 2019).

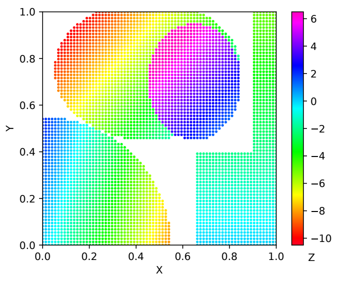
3.1 The Algorithm

CREAM is recursive and during its first iteration it considers the data set surrounding cube, that is the minimal cube

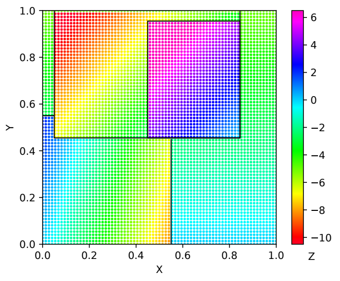
[2]Preliminary experiments proved GMMs and DBSCAN to achieve the best results in the general case. Furthermore, both can exploit fast and automated procedures to tune their hyper-parameters, enabling a flexible and versatile application in tasks that are not known a priori

Algorithm 1: CREAM pseudocode

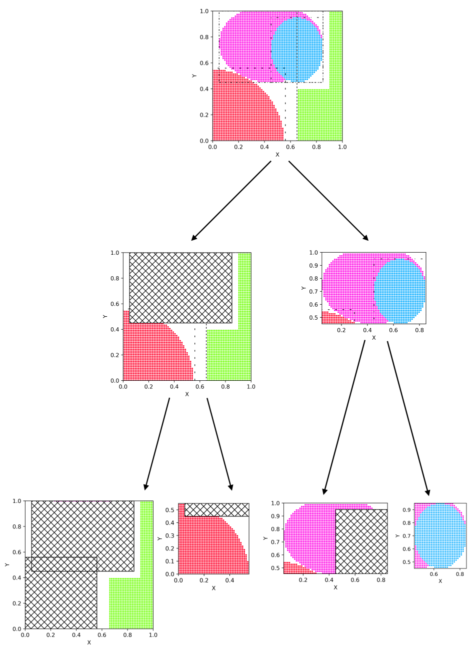
```
Require: maximum depth $\delta$
Require: predictive error threshold $\theta$
Require: maximum amount of clusters $\xi$
Ensure: root node of the induced tree
 1: function CREAM($D$)
 2:     $H_0 \leftarrow$ SURROUNDINGCUBE($D$)
 3:     $N_0 \leftarrow$ NEWNODE($H_0, D$)
 4:     $nodes \leftarrow \{(\infty, N_0, 1)\}$
 5:     while $nodes \neq \emptyset$ do
 6:         $nodes =$ SPLIT($nodes$)
 7:     return $N_0$
 8: function SURROUNDINGCUBE($D$)
 9:     return the minimal cube enclosing all the cluster $D$ points
10: function NEWNODE($H$, $D$)
11:     $node \leftarrow$ new Node()
12:     $node.cube \leftarrow H$
13:     $node.data \leftarrow D$
14:     $(node.right, node.left) \leftarrow (\emptyset, \emptyset)$
15:     return $node$
16: function SPLIT($nodes$)
17:     $(error, node, depth) \leftarrow$ POP($nodes$)
18:     $clusters \leftarrow$ CREATECLUSTERS($node.data$)
19:     $eligible \leftarrow$ CLUSTERSTOPAIRS($clusters, node$)
20:     if $eligible = \emptyset$ then
21:         return $nodes$
22:     $(error, node_r, node_l) \leftarrow \underset{(e,r,l)\in eligible}{\arg\min} \{e\}$
23:     $(node.right, node.left) \leftarrow (node_r, node_l)$
24:     for all $child \in \{node.right, node.left\}$ do
25:         $error \leftarrow$ PREDICTIVEERROR($child$)
26:         if $(error > \theta) \wedge (depth < \delta)$ then
27:             $nodes \leftarrow nodes \cup \{(error, child, depth + 1)\}$
28:     return $nodes$
29: function POP($nodes$)
30:     return $\underset{(e,n,d)\in nodes}{\arg\max} \{e\}$
31: function CREATECLUSTERS($D$)
32:     return at most $\xi$ clusters containing the data of $D$
33: function CLUSTERSTOPAIRS($C$, $N$)
34:     $pairs \leftarrow \emptyset$
35:     for all $cluster \in C$ do
36:         $data_i \leftarrow cluster \setminus \{ c \in cluster \mid c$ is an outlier $\}$
37:         $cube_i \leftarrow$ SURROUNDINGCUBE($data_i$)
38:         if $cube_i = N.cube$ then
39:             continue
40:         $data_o \leftarrow N.data \setminus data_i$
41:         $cube_o \leftarrow$ SURROUNDINGCUBE($data_o$)
42:         $node_i \leftarrow$ NEWNODE($cube_i, data_i$)
43:         $node_o \leftarrow$ NEWNODE($cube_o, data_o$)
44:         $error \leftarrow$ MEANERROR($\{node_i, node_o\}$)
45:         $pairs \leftarrow pairs \cup \{(error, node_i, node_o)\}$
46:     return $pairs$
47: function PREDICTIVEERROR($node$)
48:     return average predictive error for $node$
49: function MEANERROR($nodes$)
50:     return $\sum_{n \in nodes} \frac{\text{PREDICTIVEERROR}(n)}{|nodes|}$
```



(a) Data set.



(b) Approximation performed by CREAM.



(c) Binary tree induced by CREAM.

Figure 1: Example of CREAM partitioning performed on an artificial data set having 4 superposing clusters.

enclosing all the available training samples. This hypercube is associated with the CREAM tree root. Each tree node contains a set of training instances, representing a cluster, and an approximating region for that cluster, which may be a hypercube or a difference cube enclosing all the node's training instances. Only one node is refined during a single iteration. At each iteration CREAM partitions the node's hypercube into a smaller cube and a difference cube, which become the right and left children of the root, respectively. It is worth noticing that the difference cube represented in any left child of the tree is given by subtracting the right sibling from the common parent node. This implies that the rightmost leaf of the tree is always associated with a hypercube, whereas all the left child nodes represent difference cubes. No assumptions can be made about other right-child leaves.

The node splitting is repeated during the successive iterations with the same logic, leading to the binary tree creation. The strategy adopted to split nodes is pivotal for the presented procedure. CREAM adopts a greedy search algorithm, selecting for each node the best split amongst all the possible splits. The best split is the one resulting in two child nodes associated with the lowest predictive errors measured on the training data. The predictive error for a split is calculated by observing the difference between the expected output reported for the samples in the training set (i.e., the ground truth) and the output provided by CREAM by assuming that a particular split is definitive. There are no guarantees that this logic leads to the optimum partitioning at the end of the tree induction since it is based on the selection of local optima. However, during the experiments CREAM provided competitive results w.r.t. other techniques, demonstrating that the greedy approach is a good trade-off between search exhaustivity and performance.

To better understand the concept of predictive error, it is worthwhile to point out that CREAM distinguishes between input and output features and therefore it may be applied to classification and regression data sets. However, given the supervised nature of CREAM, when performing clustering the ground truth assignments should be provided to the algorithm. When the expected assignments are not known a priori, it is possible to adopt another clustering technique to obtain them. This allows CREAM to measure the clustering error performed at each step of the procedure. When applied to supervised classification and regression data sets it is sufficient to trivially provide CREAM with the output feature to enable predictive performance monitoring. To carry out predictions with CREAM also for classification and regression tasks we refer to the generalisation proposed by Sabbatini et al. (2022a). According to the generalisation CREAM assigns to the clusters as output values *(i)* the most common labels inside the clusters for classification tasks; *(ii)* the mean output values for the data inside the clusters for regression tasks with constant outputs; or *(iii)* a linear combination involving input variables for other regression tasks. For the corresponding predictive error estimation, we use the wrong assignment rate for clustering tasks, a metric inversely proportional to the accuracy score for classification tasks (e.g., $1 - accuracy$), and the mean absolute error for regression tasks.

In Figure 1 an example of partitioning provided by

CREAM on an artificial regression data set having 3 real-valued features is depicted. Figures 1a and 1b report the training data set and the approximation obtained with CREAM, respectively. The binary tree produced by CREAM is shown in Figure 1c.

The CREAM pseudocode and related details are reported in Algorithm 1. The steps performed during the execution of CREAM can be summarised as follows:

1. find the surrounding hypercube for the training data set to build the tree root;
2. set the root as current node;
3. apply GMM clustering to the current node data to automatically determine the number of contained clusters and the clusters themselves;
4. apply DBSCAN to the identified clusters for outlier removal, since sparse data would produce too big hypercubes in the next step;
5. find the minimal surrounding hypercubes for the data belonging to each cluster, obtaining hypercubic approximations of the clusters;
6. find the difference cube for each approximating hypercube, given by subtracting the approximating cube from the current node's cube, and create pairs of associated cubes;
7. calculate the average predictive error for each pair, select the one having the smallest average error and set it as the best pair;
8. assign the best approximating cube (and the contained training instances) to the current node's right child and the best difference cube (and corresponding data) to the current node's left child;
9. repeat from step 3 for every child of the current node having predictive error larger than the user-defined threshold θ until the maximum tree depth δ is reached.

3.2 Parameters Required by CREAM

Users adopting CREAM have to provide 3 parameters to it.

θ The error threshold θ calibrates the predictive performance of our explainable clustering technique, given that nodes associated with predictive errors larger than the threshold are further partitioned with additional iterations. The value of θ strictly depends on the peculiarities of the data at hand. According to the CREAM design, when applied to clustering data sets the rate of wrong assignments is adopted as predictive error and therefore θ represents an upper-bound. Analogously, if the algorithm is applied to a classification task θ has to be defined as a predictive error threshold. This means that cubes having an associated error larger than the threshold are further partitioned. Finally, for regression tasks, the θ parameter represents an upper-bound for the mean absolute error measured for a tree node.

δ The maximum depth δ regulates the overall human readability of the clustering provided by CREAM, since enabling the induction of deep trees may result in large amounts of clusters. Setting $\delta = d$ lets CREAM identifying at most 2^d clusters. There is a trade-off between a small number of clusters, corresponding to high human readability, and the capability of achieving good predictive performance. Larger amounts of smaller clusters tend to have better predictive performances than smaller amounts of larger clusters, that in turn are easier to be interpreted by humans.

ξ The third user-defined parameter required by CREAM (cf. step 3 of the algorithm) is ξ, representing the maximum amount of clusters identifiable via GMMs. ξ is an upper-bound value, given that the optimum amount of clusters is automatically selected during the algorithm execution via BIC score evaluations. We recommend that users provide ξ values greater or equal to the expected number of clusters, to avoid a wrong clustering. We point out here that the ε parameter required by DBSCAN (cf. step 4 of the algorithm) is automatically set according to methods found in the literature (Rahmah and Sitanggang 2016) and therefore it has not to be provided and tuned by users.

3.3 Automated Parameter Tuning

Tuning the parameters required by CREAM is a pivotal task heavily impacting the overall clustering performances, despite being challenging and tedious for human users. To avoid time-expensive manual tuning and to help users in the selection of the CREAM parameters, we developed the ORCHID (OptimiseR for Clustering via HIerarchical Decomposition) automated procedure. We stress here that the ξ parameter of CREAM has a negligible impact on the overall clustering performance if its value is kept larger than the actual amount of clusters to identify. For such a reason ORCHID only focuses on the tuning of θ and δ parameters.

The tuning procedure considers the design of CREAM to make some assumptions, following the same strategy adopted in the PEDRO optimiser (Sabbatini and Calegari 2022) for the GridEx and GridREx algorithms (Sabbatini, Ciatto, and Omicini 2021). Given that CREAM is a general-purpose algorithm applicable to clustering, classification and regression tasks, ORCHID accepts a training data set including both input features and expected outputs. The predictions are not compared with the expected outputs on the basis of classical clustering scores, since these cannot be applied to continuous values: ORCHID adopts typical classification and regression scoring metrics instead. ORCHID finds the optimum values for the δ and θ parameters by comparing several instances of CREAM only differing for a single parameter. Comparisons are based on ad-hoc metrics considering two indices. One is the predictive loss of CREAM applied to the provided training data set. The other is the human-readability loss associated with the same CREAM instance. The predictive loss, representing the error of the CREAM's predictions w.r.t. the expected outputs, may be expressed as the mean absolute error (for regression data sets) or as the percentage of wrong cluster assignments/classifications (otherwise). The readability loss may be expressed as the number of clusters identified by CREAM, assuming that human readability is hindered by large amounts of clusters. An ideal CREAM instance

Algorithm 2: ORCHID pseudocode

```
Require: maximum depth $\Delta$, default = 10
Require: predictive/readability loss trade-off $\Phi$, default 0.1
Require: maximum predictive loss increase $p_{max}$, default = 1.2
Require: minimum rule loss $r_{min}$, default = 0.9
Require: patience value $patience_0$, default = 5
Ensure: optimum values for the CREAM parameters
 1: function ORCHID($D$)
 2:     $\Pi \leftarrow \varnothing$                              ▷ set of all configurations
 3:     $\pi' \leftarrow$ undefined                               ▷ last configuration
 4:     $\delta \leftarrow 1$                                     ▷ depth parameter
 5:     while $\delta < \Delta$ do
 6:         $\Pi' \leftarrow$ THRESHOLDSEARCH($D, \delta$)
 7:         $\pi \leftarrow$ SELECTBEST($\Pi'$)
 8:         $\Pi \leftarrow \Pi \cup \Pi'$
 9:         $imp \leftarrow$ GAIN($\pi, \pi'$, "depth")
10:         if $(|\Pi| > 1) \wedge (imp < 1.2)$ then return $\Pi$
11:         $\delta \leftarrow \delta + 1$
12:         $\pi' \leftarrow \pi$
13:     return $\Pi$
14: function SELECTBEST($\Pi$)
15:     return the best configuration $\pi \in \Pi$, considering $\Phi$
16: function GAIN($\pi_1$, $\pi_2$, $criterion$)
17:     return an evaluation of $\pi_1$ w.r.t. $\pi_2$ according to $criterion$
18: function THRESHOLDSEARCH($D$, $\delta$)
19:     $step \leftarrow$ undefined                               ▷ step to update $\theta$
20:     $p_0 \leftarrow$ undefined                                ▷ initial predictive loss
21:     $\pi' \leftarrow$ undefined                               ▷ last configuration
22:     $\Pi \leftarrow \varnothing$                              ▷ set of all configurations
23:     $\theta \leftarrow 1.0$                                   ▷ threshold parameter
24:     $patience \leftarrow patience_0$                          ▷ residual patience
25:     while $patience > 0$ do
26:         $C \leftarrow$ CREAM$(D, \delta, \theta)$
27:         $\pi \leftarrow$ EVALUATE($C, D$)
28:         $\Pi \leftarrow \Pi \cup \{(\pi, \delta, \theta)\}$
29:         if $|\Pi| = 1$ then
30:             $\pi' \leftarrow \pi$
31:             $p_0 \leftarrow \pi.p$
32:             $\theta \leftarrow \frac{\pi.p}{20}$
33:             $step \leftarrow \frac{0.75\pi.p}{patience_0}$
34:             continue
35:         if $(\pi.r = 1) \vee (\pi.p = 0.0) \vee (\pi.p > p_0 \cdot p_{max})$ then
36:             return $\Pi$
37:         $imp \leftarrow$ GAIN($\pi, \pi'$, "threshold")
38:         if $(imp \leq 1) \vee (\pi.r > \lceil \pi'.r \cdot r_{min} \rceil)$ then
39:             $patience \leftarrow patience - 1$
40:         $\pi' \leftarrow \pi$
41:         $\theta \leftarrow \theta + step$
        return $\Pi$
42: function EVALUATE($C$, $D$)
43:     $\pi \leftarrow$ INIT()
44:     $\pi.p \leftarrow$ PREDICTIVELOSS($C, D$)
45:     $\pi.r \leftarrow$ amount of clusters identified by $C$
46:     return $\pi$
47: function PREDICTIVELOSS($C$, $D$)
48:     return the predictive loss of $C$ applied to the data in $D$
49: function INIT()
50:     return an object with $p$ and $r$ fields
```

should have both losses as small as possible. It is worthwhile to point out that the losses are intertwined. Given that CREAM is aware of the cluster labels, it is able to fragment clusters into disjoint input space hypercubic partitions without considering them as actually different clusters, for the sake of predictive performance. However, this is a hindering factor from the human-readability standpoint. In other words, spreading the samples of an expected cluster over multiple hypercubic regions may lead to better predictive accuracy, but surely hinders the readability of the overall clustering. Vice versa, limiting the number of clusters may induce larger predictive errors due to the hypercubic approximation strategy. Given all these considerations, large values for δ worsen the readability loss and enhance the predictive loss, as well as small values for θ.

The strategy adopted by ORCHID to highlight the best parameter values is to consider a small worsening in the predictive (resp. readability) loss balanced by a large enough enhancement in the readability (resp. predictive) loss. The trade-off between them is defined by the user and should be set according to the data set peculiarities.

Since the parameters analysed by ORCHID are real-valued, it is not possible to perform an exhaustive search inside the parameter space. Conversely, ORCHID explores it looking for local optima to highlight promising value configurations. At the end of the search, all the candidate configurations are compared according to 3 different criteria, providing to the user the parameter values associated with the CREAM instances achieving smaller predictive loss, smaller readability loss, and the best trade-off between the two. By representing with p and r the predictive and readability losses, respectively, and with Φ the user-defined trade-off between them, the following scoring function is adopted to evaluate an instance of CREAM:

$$score = p \cdot \lceil 2\, r\, \Phi \rceil \,. \qquad (1)$$

The best CREAM instance is the one having the lowest $score$.

In order to run ORCHID, a set of parameters should be provided by the users: *(i)* a maximum search depth Δ; *(ii)* a trade-off between readability and predictive losses Φ; *(iii)* a maximum predictive loss increase e_{max}; *(iv)* a minimum readability loss decrease r_{min}; *(v)* a patience value $patience_0$. The Δ parameter is the maximum depth used for CREAM during the execution of ORCHID. Φ is used to associate a numeric score with the CREAM instances, according to Equation (1). e_{max} is the maximum allowed worsening in the predictive loss, expressed as the percentage of the initial predictive loss, used to trigger an early interruption of the parameter exploration along not promising directions. Analogously, r_{min} is the minimum required enhancement in the readability loss. Finally, the patience value is adopted to enable ORCHID exiting local sub-optimum parameter space regions. Despite the number of parameters required by ORCHID, our experiments show that the procedure is able to give the optimum values for CREAM even with the default parameters, without any user effort.

Early interruptions are also triggered when ORCHID highlights absolutely optimum CREAM configurations, i.e.,

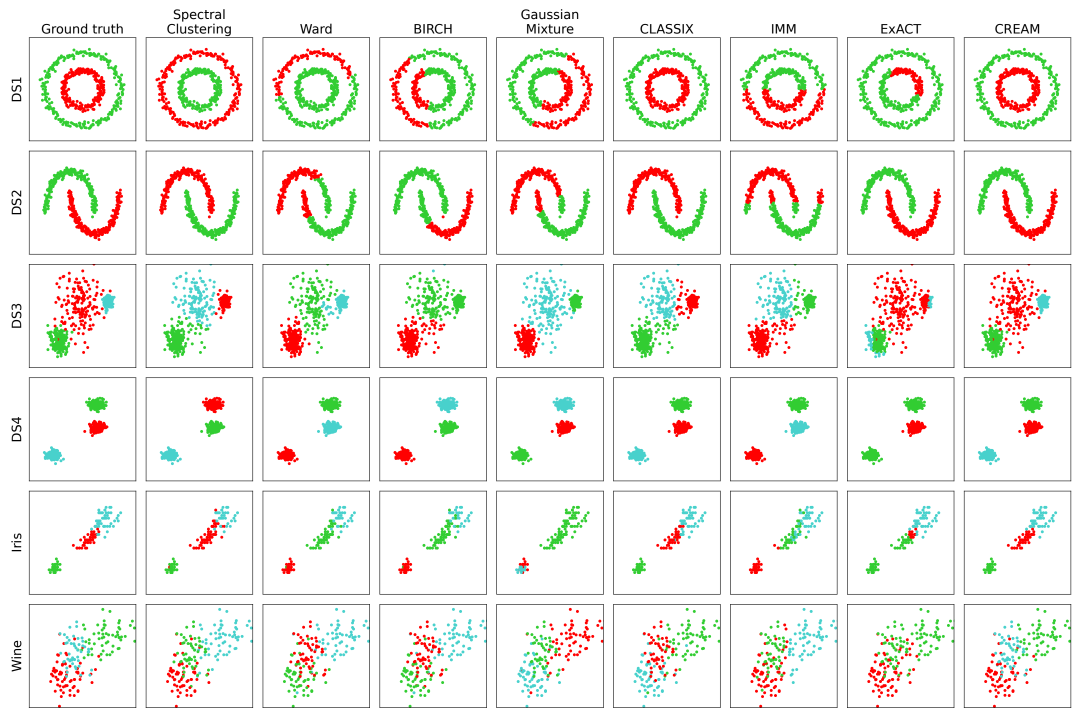


Figure 2: Comparison between CREAM and other state-of-the-art traditional and explainable clustering techniques applied to synthetic and real-world data sets.

with no predictive or readability losses, or when there is no gain between successive search iterations. Search iterations may refer to the depth or threshold parameters and corresponding improvements are calculated as follows:

$$\delta\text{-}gain(\pi_1, \pi_2) = \left(\left(1 - \frac{\pi_2^p}{\pi_1^p} \right)^{0.1} \cdot \frac{\lceil \pi_2^r \cdot \Phi \rceil}{\lceil \pi_1^r \cdot \Phi \rceil} \right)^{-1}, \quad (2)$$

$$\theta\text{-}gain(\pi_1, \pi_2) = 1 - \frac{\pi_2^r}{\pi_1^r} + \frac{\pi_1^p}{\pi_2^p}, \quad (3)$$

where π_1 and π_2 are compact representations of the p and r losses corresponding to a pair of CREAM instances.

The parameter space search performed by ORCHID to find the optimum parameters for CREAM is reported in Algorithm 2. The procedure starts by evaluating CREAM instances having maximum depth equal to 1. During the depth search ORCHID iteratively analyses instances with growing depth values, up to Δ or until an early interruption is triggered (i.e., no gain is detected by augmenting the depth; cf. Equation (2)). For each depth value, several thresholds are tested through a threshold search, starting with $\theta = 1.0$ and adapting it on the basis of the measured predictive loss. Since this loss may be reduced by selecting threshold values smaller than the loss itself, after the first iteration θ is set to a twentieth of the loss.[3] This value is then increased iteration after iteration. Excepting an early stop, the threshold search terminates when the patience expires, returning a set of candidate configurations. Patience is consumed only when a new threshold value is tested with no gain (cf. Equation (3)).

As a result, for each depth value, a set of candidate configurations is selected. The depth gain is calculated only by comparing the "local" best configurations associated with consecutive depth values. At the end of the procedure, the "global" best configuration is selected with Equation (1) amongst all the candidates.

[3]One-twentieth of the measured loss is a reasonable starting value since there is no need to test θ values larger than the measured loss. Given that θ is increased during the algorithm iterations, with this starting value it is possible to automatically perform a more fine-grained search of θ values smaller than the measured loss

4 Experiments

Experiments involving CREAM and other state-of-the-art techniques applied to clustering, classification and regression tasks are reported here. The adopted CREAM implementation is included in the PSyKE framework[4] (Sabbatini et al. 2021; Sabbatini et al. 2022b; Sabbatini, Ciatto, and Omicini 2022; Calegari and Sabbatini 2023).

4.1 Explainable Clustering

Being an explainable clustering technique, we first assess the performance of CREAM to cluster labelled data. CREAM has been applied to 6 different synthetic and real-world data sets. In particular, 4 out of 6 are synthetic clustering data sets taken from the Scikit-Learn Python library[5] (Pedregosa et al. 2011), whereas the remaining 2 are real-world classification data sets, i.e., Iris (Fisher 1936) and Wine (Forina et

[4]Code available at https://github.com/psykei/psyke-python
[5]https://scikit-learn.org/stable/modules/clustering.html

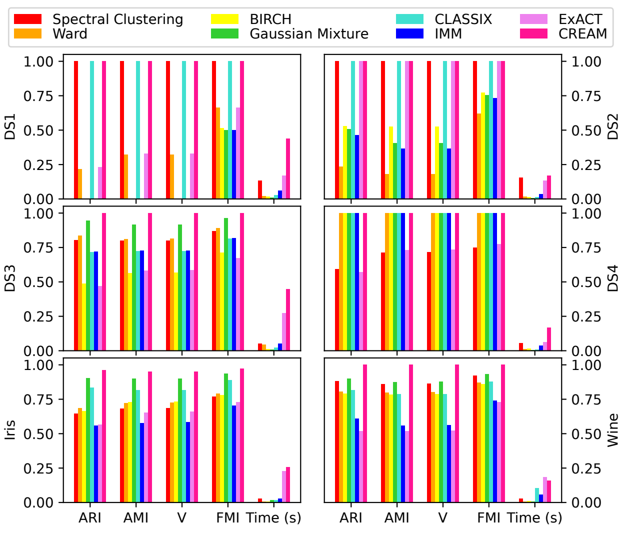


Figure 3: Performance assessments for the clustering algorithms and data sets shown in Figure 2.

Feature	SL	SW	PL	PW
Setosa	4.4 – 5.8	2.3 – 4.1	1.2 – 1.9	0.1 – 0.5
Versicolor	4.9 – 6.7	2.2 – 3.2	3.0 – 5.0	1.0 – 1.8
Virginica	4.4 – 7.9	2.2 – 4.1	1.2 – 6.9	0.1 – 2.5

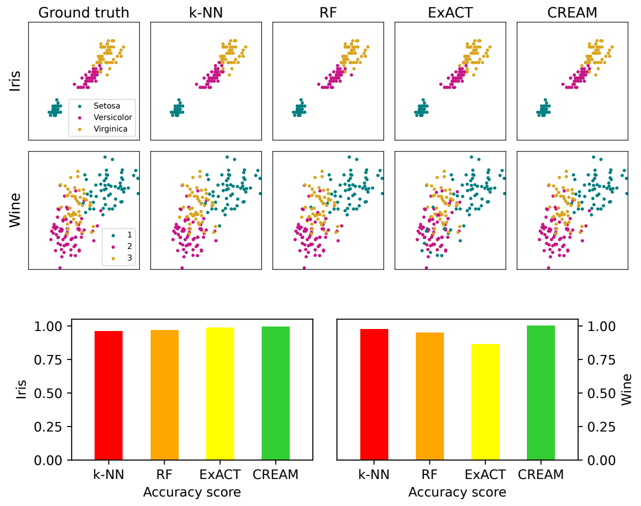
Table 1: Example of CREAM clustering for the Iris data set. Input features: sepal length and width (SL and SW, resp.), petal length and width (PL and PW, resp.). Values are expressed in cm.

al. 1988). All the adopted synthetic data sets are described by 2 continuous input features and they have 2 or 3 clusters, as shown in the first column of Figure 2. Both Iris and Wine data sets have 3 possible output values. Iris has 4 continuous input features, whereas Wine has 13 of them. In Figure 2 only petal length and width features are reported for the Iris data set. For the Wine data set the reported features are alcohol and proline.

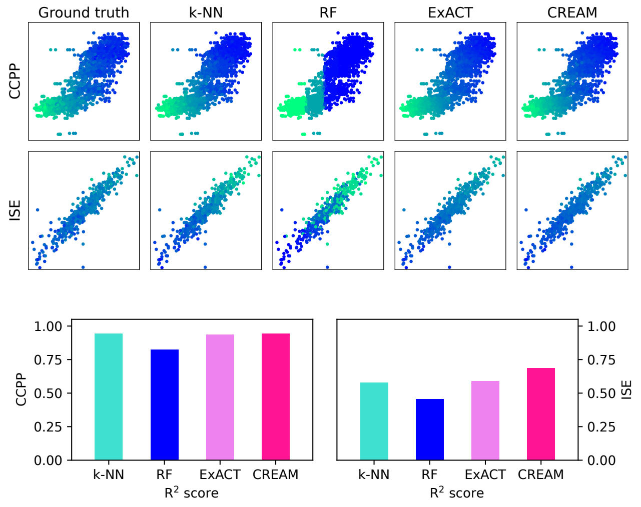
To assess the performance of CREAM in executing clustering 4 metrics have been exploited, namely: Fowlkes-Mallows index (FMI), adjusted rand index (ARI), V-measure (V), and adjusted mutual score (AMI) (Fowlkes and Mallows 1983; Hubert and Arabie 1985; Rosenberg and Hirschberg 2007; Nguyen, Epps, and Bailey 2010, respectively). The assessment is carried out by measuring these indices for all the aforementioned data sets and then by performing a comparison with several state-of-the-art traditional and explainable clustering techniques, evaluated via the same metrics. The set of traditional clustering algorithms is composed of spectral clustering, Ward, BIRCH and GMMs. CLASSIX, IMM and ExACT, on the other hand, are the explainable clustering techniques used as benchmarks. The output clusterings of these algorithms are reported in Figure 2. Experiments are completed by an execution time comparison amongst all the clustering techniques.

Figure 3 reports all the metrics measured for all combinations of clustering algorithms and data sets. Execution time is reported in seconds, averaged on 100 runs. We recall here that the ARI, AMI, V and FMI scores range in [0, 1], highlighting good-quality clusterings when values are close to 1, and that they are not susceptible to renaming and permutations of the predicted cluster labels. Therefore they are not suitable to evaluate the classification accuracy score of clustering techniques applied to perform classification tasks.

Qualitative and quantitative assessments of the algorithms' performance may be carried out by observing both Figures 2 and 3, where the superiority of CREAM, CLASSIX and spectral clustering is clearly highlighted. However, only CREAM is able to achieve a comparable or better performance w.r.t. all the other techniques in all data sets, with all the scores equal to 1 or slightly below. The main drawback of CREAM is its computational time since it appears



(a) Classification case study.



(b) Regression case study.

Figure 4: Comparison between CREAM and state-of-the-art models applied to real-world data sets.

Feature	AT	V	AP	RH	EP
Cluster 1	6.2 – 32.5	34.0 – 50.2	997.9 – 1026.4	35.6 – 100.1	502.5 - 2.2 AP - 0.3 AT - 0.1 V
Cluster 2	6.2 – 35.8	25.4 – 81.6	997.9 – 1026.5	25.6 – 100.1	234.7 - 1.4 AP - 0.3 AT + 0.3 RH - 0.1 V
Cluster 3	3.3 – 14.6	34.7 – 44.5	1011.3 – 1033.3	59.0 – 98.7	720.3 - 2.2 AP - 0.5 AT - 0.2 RH - 0.2 V
Cluster 4	2.3 – 35.8	25.4 – 81.6	992.9 – 1033.3	25.6 – 100.2	579.0 - 2.1 AP - 0.6 AT - 0.1 RH

Table 2: Example of CREAM clustering for the CCPP data set. Input features: ambient temperature and pressure (AT and AP, resp.), relative humidity (RH) and exhaust vacuum (V). Output feature: net hourly electrical energy output (EP).

to be the slowest algorithm in the pool. Nonetheless, it is able to complete its task in less than half a second.

4.2 Explainable Classification

CREAM is suitable to perform classification tasks since it is able to provide explainable predictions when queried with instances to be classified. In Figure 4a the results of CREAM, ExACT and other ML opaque classifiers applied to the Iris and Wine data sets are reported for a visual comparison. The BB classifiers are a k-nearest neighbours (k-NN) model and a random forest (RF). For each combination of data sets and classifiers, the predictive performance has been measured in terms of classification accuracy score, representing the percentage of correct predictions w.r.t. all the provided predictions. The measured accuracies are also reported in Figure 4a. CREAM is able to achieve higher accuracy than ML state-of-the-art models. Furthermore, its outputs are interpretable and the corresponding clusters may be expressed in a human-readable format.

As an example, we show in Table 1 the clusters obtained via CREAM for the Iris data set, also reporting the corresponding output class prediction. These results are obtained by setting the maximum depth $\delta = 2$, the error threshold $\theta = 0.1$ and the maximum amount of clusters $\xi = 3$. This value for the δ parameter enables a maximum amount of 4 final clusters. Being a classification data set, $\theta = 0.1$ means that hypercubic regions are further split if the corresponding accuracy score is smaller than $1 - \theta = 0.9$. The output clustering is clearly humanly interpretable, since for each one of the 3 possible Iris classes there is an associated hypercubic input space region described in terms of interval inclusion constraints over the 4 input variables.

4.3 Explainable Regression

We conclude the experiment section by reporting the results of CREAM applied to regression data sets, to show its versatility and potentialities. In particular, we adopted the Combined Cycle Power Plant (CCPP; Tüfekci 2014) and the Istanbul Stock Exchange (ISE; Akbilgic, Bozdogan, and Balaban 2014) data sets. The CCPP data set is composed of 5 continuous attributes, one of which is the output feature. The ISE data set describes a regression task with 7 continuous input features plus another input feature representing a timestamp. This latter has been neglected during the experiments reported here.

Figure 4b depicts the results of CREAM and other ML predictors applied to the CCPP and ISE data sets. For the CCPP data set only the ambient temperature and the exhaust vacuum input features are reported. For the ISE data set the represented input features are the stock market return index of the UK and the MSCI European index. The BB models are similar w.r.t. the classification experiment. Being regression tasks, the predictive performance of the models has been assessed via the R^2 score. Corresponding measurements are also reported in Figure 4b. Also in this case CREAM is able to outperform its interpretable and opaque counterparts since it achieves the highest scores.

To give a practical example of the CREAM capabilities in providing explainable predictions for regression tasks, the output of CREAM applied to the CCPP data set is resumed in Table 2. To obtain the reported results we adopted a CREAM instance with $\delta = 2$, $\theta = 0.1$ and $\xi = 4$. In this case $\theta = 0.1$ induces the refinement of hypercubic approximations having mean absolute predictive error greater than 0.1. For regression data sets like CCPP it is possible to obtain with CREAM also clusters having more directly interpretable constant output values instead of linear combinations. Nonetheless, in our experiments we privileged clusters with linear outputs since they proved to have better predictive performance, even if at the expense of human readability.

5 Conclusions

In this paper we present CREAM, a new explainable clustering technique applicable to data sets described by continuous input features. CREAM proved to be a very versatile procedure equivalent or superior to other clustering techniques, both traditional and interpretable. As a drawback, it requires more computational time to be executed. The user-defined parameters required by CREAM can be tuned by using the automated ORCHID procedure. CREAM can also be exploited for explainable classification and regression tasks, providing human-interpretability thanks to the hypercubic approximation of the identified clusters. Our experiments show its ability to provide predictions having higher predictive performance and wider extents of human-readability w.r.t. opaque ML state-of-the-art models. In the future, we plan to enrich CREAM with other splitting strategies, in order to achieve higher performance from both the predictive and computational time standpoints.

Acknowledgments

This work has been supported by the EU ICT-48 2020 project TAILOR (No. 952215).

References

Akbilgic, O.; Bozdogan, H.; and Balaban, M. E. 2014. A novel hybrid rbf neural networks model as a forecaster. *Statistics and Computing* 24:365–375.

Ankerst, M.; Breunig, M. M.; Kriegel, H.; and Sander, J. 1999. OPTICS: ordering points to identify the clustering structure. In Delis, A.; Faloutsos, C.; and Ghandeharizadeh, S., eds., *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, 49–60. ACM Press.

Basak, J., and Krishnapuram, R. 2005. Interpretable hierarchical clustering by constructing an unsupervised decision tree. *IEEE Trans. Knowl. Data Eng.* 17(1):121–132.

Bertsimas, D.; Orfanoudaki, A.; and Wiberg, H. M. 2018. Interpretable clustering via optimal trees. *CoRR* abs/1812.00539.

Blanco-Justicia, A., and Domingo-Ferrer, J. 2019. Machine learning explainability through comprehensible decision trees. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, 15–26. Springer.

Calegari, R., and Sabbatini, F. 2023. The PSyKE technology for trustworthy artificial intelligence. 13796:3–16. XXI International Conference of the Italian Association for Artificial Intelligence, AIxIA 2022, Udine, Italy, November 28 – December 2, 2022, Proceedings.

Chen, X., and Güttel, S. 2022. Fast and explainable clustering based on sorting. *CoRR* abs/2202.01456.

Chen, J.; Chang, Y.; Hobbs, B.; Castaldi, P. J.; Cho, M. H.; Silverman, E. K.; and Dy, J. G. 2016. Interpretable clustering via discriminative rectangle mixture model. In Bonchi, F.; Domingo-Ferrer, J.; Baeza-Yates, R.; Zhou, Z.; and Wu, X., eds., *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, 823–828. IEEE Computer Society.

Cheng, Y. 1995. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17(8):790–799.

Dasgupta, S.; Frost, N.; Moshkovitz, M.; and Rashtchian, C. 2020. Explainable k-means and k-medians clustering. *CoRR* abs/2002.12538.

Ester, M.; Kriegel, H.; Sander, J.; and Xu, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In Simoudis, E.; Han, J.; and Fayyad, U. M., eds., *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, 226–231. AAAI Press.

European Commission; Directorate-General for Communications Networks, C.; and Technology. 2019. *Ethics guidelines for trustworthy AI.* Publications Office.

European Commission. 2021. AI Act – Proposal for a regulation of the European Parliament and the Council laying down harmonised rules on Artificial Intelligence (Artificial Intelligence Act) and amending certain union legislative acts. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206.

Fisher, R. A. 1936. The use of multiple measurements in taxonomic problems. *Annals of Eugenics* 7(2):179–188.

Forina, M.; Leardi, R.; Armanino, C.; Lanteri, S.; Conti, P.; and Princi, P. 1988. Parvus: An extendable package of programs for data exploration, classification and correlation. *Journal of Chemometrics* 4(2):191–193.

Fowlkes, E. B., and Mallows, C. L. 1983. A method for comparing two hierarchical clusterings. *Journal of the American statistical association* 78(383):553–569.

Fraiman, R.; Ghattas, B.; and Svarc, M. 2013. Interpretable clustering using unsupervised binary trees. *Advances in Data Analysis and Classification* 7(2):125–145.

Hubert, L., and Arabie, P. 1985. Comparing partitions. *Journal of classification* 2:193–218.

Jang, J., and Jiang, H. 2019. DBSCAN++: towards fast and scalable density clustering. In Chaudhuri, K., and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 3019–3029. PMLR.

Ling, R. F. 1972. On the theory and construction of k-clusters. *The Computer Journal* 15(4):326–332.

Lloyd, S. P. 1982. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* 28(2):129–136.

Manduchi, L.; Hüser, M.; Faltys, M.; Vogt, J. E.; Rätsch, G.; and Fortuin, V. 2021. T-DPSOM: an interpretable clustering method for unsupervised learning of patient health states. In Ghassemi, M.; Naumann, T.; and Pierson, E., eds., *ACM CHIL '21: ACM Conference on Health, Inference, and Learning, Virtual Event, USA, April 8-9, 2021*, 236–245. ACM.

Moshkovitz, M.; Dasgupta, S.; Rashtchian, C.; and Frost, N. 2020. Explainable k-means and k-medians clustering. In *International conference on machine learning*, 7055–7065. PMLR.

Murphy, K. P. 2012. *Machine learning – A probabilistic perspective*. Adaptive computation and machine learning series. MIT Press.

Nguyen, X. V.; Epps, J.; and Bailey, J. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research* 11:2837–2854.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; VanderPlas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research (JMLR)* 12:2825–2830.

Rahmah, N., and Sitanggang, I. S. 2016. Determination of optimal epsilon (eps) value on DBSCAN algorithm to clustering data on peatland hotspots in Sumatra. In *IOP conference series: earth and environmental science*, volume 31, 012012. IOP Publishing.

Rosenberg, A., and Hirschberg, J. 2007. V-Measure: A conditional entropy-based external cluster evaluation measure. In Eisner, J., ed., *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, 410–420. ACL.

Sabbatini, F., and Calegari, R. 2022. Symbolic knowledge extraction from opaque machine learning predictors: GridREx & PEDRO. In Kern-Isberner, G.; Lakemeyer, G.; and Meyer, T., eds., *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel. July 31 - August 5, 2022*.

Sabbatini, F., and Calegari, R. 2023. Explainable black boxes via explainable clustering: ExACT and CReEPy. *Decision Support Systems (submitted to)*.

Sabbatini, F.; Ciatto, G.; Calegari, R.; and Omicini, A. 2021. On the design of PSyKE: A platform for symbolic knowledge extraction. In Calegari, R.; Ciatto, G.; Denti, E.; Omicini, A.; and Sartor, G., eds., *WOA 2021 – 22nd Workshop "From Objects to Agents"*, volume 2963 of *CEUR Workshop Proceedings*, 29–48. Sun SITE Central Europe, RWTH Aachen University. 22nd Workshop "From Objects to Agents" (WOA 2021), Bologna, Italy, 1–3 September 2021. Proceedings.

Sabbatini, F.; Ciatto, G.; Calegari, R.; and Omicini, A. 2022a. Hypercube-based methods for symbolic knowledge extraction: Towards a unified model. In Ferrando, A., and Mascardi, V., eds., *WOA 2022 – 23rd Workshop "From Objects to Agents"*, volume 3261 of *CEUR Workshop Proceedings*. Sun SITE Central Europe, RWTH Aachen University. 48–60.

Sabbatini, F.; Ciatto, G.; Calegari, R.; and Omicini, A. 2022b. Symbolic knowledge extraction from opaque ML predictors in PSyKE: Platform design & experiments. *Intelligenza Artificiale* 16(1):27–48.

Sabbatini, F.; Ciatto, G.; and Omicini, A. 2021. GridEx: An algorithm for knowledge extraction from black-box regressors. In Calvaresi, D.; Najjar, A.; Winikoff, M.; and Främling, K., eds., *Explainable and Transparent AI and Multi-Agent Systems. Third International Workshop, EXTRAAMAS 2021, Virtual Event, May 3–7, 2021, Revised Selected Papers*, volume 12688 of *LNCS*. Basel, Switzerland: Springer Nature. 18–38.

Sabbatini, F.; Ciatto, G.; and Omicini, A. 2022. Semantic Web-based interoperability for intelligent agents with PSyKE. In Calvaresi, D.; Najjar, A.; Winikoff, M.; and Främling, K., eds., *Explainable and Transparent AI and Multi-Agent Systems*, volume 13283 of *Lecture Notes in Computer Science*. Springer. chapter 8, 124–142.

Shi, J., and Malik, J. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(8):888–905.

Tüfekci, P. 2014. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems* 60:126–140.

Yu, S. X., and Shi, J. 2003. Multiclass spectral clustering. In *9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France*, 313–319. IEEE Computer Society.

Zhang, T.; Ramakrishnan, R.; and Livny, M. 1996. BIRCH: an efficient data clustering method for very large databases. In Jagadish, H. V., and Mumick, I. S., eds., *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, 103–114. ACM Press.