Machine Learning (2021) 110:89–138
https://doi.org/10.1007/s10994-020-05896-2

# Interpretable clustering: an optimization approach

Dimitris Bertsimas[1] · Agni Orfanoudaki[1] · Holly Wiberg[1]

Received: 9 February 2019 / Revised: 30 June 2020 / Accepted: 7 July 2020 / Published online: 16 August 2020

**Abstract**
State-of-the-art clustering algorithms provide little insight into the rationale for cluster membership, limiting their interpretability. In complex real-world applications, the latter poses a barrier to machine learning adoption when experts are asked to provide detailed explanations of their algorithms' recommendations. We present a new unsupervised learning method that leverages Mixed Integer Optimization techniques to generate interpretable tree-based clustering models. Utilizing a flexible optimization-driven framework, our algorithm approximates the globally optimal solution leading to high quality partitions of the feature space. We propose a novel method which can optimize for various clustering internal validation metrics and naturally determines the optimal number of clusters. It successfully addresses the challenge of mixed numerical and categorical data and achieves comparable or superior performance to other clustering methods on both synthetic and real-world datasets while offering significantly higher interpretability.

**Keywords** Clustering · Interpretability · Unsupervised learning · Mixed integer optimization

## 1 Introduction

Clustering is the unsupervised classification of patterns, observations, data items, or feature vectors, into groups. The clustering problem has been addressed in many machine learning contexts where there is no clear outcome of interest, such as data mining, document retrieval, image segmentation, and pattern classification; this reflects its broad appeal and usefulness in exploratory data analysis (Hastie et al. 2009). In many such problems, there is little prior information available about the data, and the decision-maker must make

Editor: Hendrik Blockeel.

Dimitris Bertsimas
dbertsim@mit.edu

Agni Orfanoudaki
agniorf@mit.edu

Holly Wiberg
hwiberg@mit.edu

1 Operations Research Center, E40-111, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

as few assumptions about the data as possible. It is under these restrictions that clustering methodology is particularly appropriate for the exploration of relationships between observations to make an assessment, perhaps preliminary, of their structure.

Unlike supervised classification, there are no class labels and thus no natural measure of accuracy. Instead, the goal is to group objects into clusters based only on their observable features, such that each cluster contains objects with similar properties and different clusters have distinct features. There have been numerous approaches to generating these clusters. Partitional methods such as $K$-means (MacQueen 1967) provide a single partition of the data into a fixed number of clusters; these methods have been improved by new initialization methods in recent decades (Arthur and Vassilvitskii 2007). Hierarchical methods produce a nested series of partitions (Sneath et al. 1973) based on a distance metric. Other more sophisticated methods include model-based clustering (Hastie et al. 2009) and density-based clustering (Ester et al. 1996) which are better able to capture clusters of irregular shape or varied density.

The end product of a clustering algorithm is a partition of the dataset. In some cases, this final cluster assignment is sufficient for the machine learning purpose, such as when one wants to simply assess the separability of the data points into distinct clusters or use it as a preprocessing step in certain prediction tasks. However, in many other decision-making applications, there is a need to interpret the resulting clusters and characterize their distinctive features in a compact form (Forgy 1965). For example, consider a medical setting in which we seek to group similar patients together to understand subgroups within a patient base. In this application, it is critical to understand how the resulting clusters differ, whether by demographics, diagnoses, or other factors.

While the importance of cluster interpretability is well-understood, there has been limited success in addressing the issue (Doshi-Velez and Kim 2017). None of the clustering algorithms described above were constructed with a goal of interpretability in the original feature space. They therefore require a post-processing step to synthesize the cluster meanings. The notion of cluster representation was introduced by Duran and Odell (1974) and was subsequently studied by Diday and Simon (1976) and Stepp and Michalski (1986). The representation of a cluster of points by its centroid has been popular across various applications (Radev et al. 2004). This works well when the clusters are compact or isotropic, but fails when the clusters are elongated or non-isotropic (Jain et al. 1999). These clusters can be better characterized computing additional metrics, such as the variance in each dimension. However, this increases the number of summary statistics used for each cluster and creates a high burden in interpretation, especially when the number of features grows large. Another common approach is the visualization of clusters on a two-dimensional graph using Principal Component Analysis (PCA) projections (Jolliffe 2011; Rao 1964). However, in reducing the dimensionality of the feature space, PCA obscures the relationship between the clusters and the original variables.

Tree-based supervised learning methods, such as CART (Breiman et al. 1984), are a natural fit for problems that prioritize interpretability, since their feature splits and decision paths offer insight into the differentiating features between members in each leaf. Most recursive partitioning algorithms generate trees in a top-down, greedy manner, which means that each split is selected in isolation without considering its effect on subsequent splits in the tree. Bertsimas and Dunn (2017, 2019) have proposed a new algorithm which leverages modern mixed-integer optimization (MIO) techniques to form the entire decision tree in a single step, allowing each split to be determined with full knowledge of all other splits. The Optimal Classification Trees (OCT) algorithm enables the construction of decision trees for classification and regression that have performance comparable with state-of-the-art methods such as random forests and gradient boosted trees without sacrificing the interpretability offered by a single tree.

A general hybrid approach can leverage such methods by first running a partitional or hierarchical clustering method and using the resulting assignments as class labels. The data can then be fit using a classification tree, in which each leaf is given a cluster label based on the most common assignment of observations in that leaf, and the decision paths leading to each cluster's leaves give insight into the differentiating features (Jain et al. 1999). Hancock et al. (2003) use decision trees to interpret and refine hierarchical clustering results for global sea surface temperatures. While these trees give an explicit delineation of cluster attributes, the methods involve a two-step process of first building the clusters and subsequently identifying their differentiating features. Thus, the main clustering mechanism utilizes a different architecture compared to the decision tree which might be hard to capture with univariate feature splits.

Several algorithms have been proposed to build interpretable clusters, where interpretability is a consideration during cluster creation rather than considered as a later analysis step. Chavent et al. (1999) presented a method that constructs binary clustering trees characterized by a novel transformation of the feature space. Further efforts focused on alternative measures for feature selection in the transformation function as well as new algorithmic implementation schemes (Basak and Krishnapuram 2005). In both of these cases, the feature space transformation involved in these methods takes a toll on interpretability. Other researchers have proposed methods to construct decision trees in the original feature space, which more closely matches our objective. Liu et al. (2000) introduced the idea of translating a clustering problem to a supervised problem that is amenable to decision tree construction. A modified purity criterion is used to evaluate splits in a way that identifies dense regions as well as sparse regions. However, this method requires additional pre-processing through the introduction of synthetic data in order to create a binary classification setting. Blockeel et al. (2000) also proposed a general top-down tree induction framework with applicability to clustering ("Predictive Clustering Trees") as well as other supervised learning tasks. Fraiman et al. (2013) developed another clustering algorithm, "Clustering using unsupervised binary trees" (CUBT), which forms greedy splits to optimize a cluster heterogeneity measure. Though these algorithms make progress towards the goal of constructing clusters directly using trees, they both employ a greedy splitting approach and do not offer flexibility in the choice of cluster validation criterion.

The need for accurate and interpretable machine learning methods is undoubtedly present, being voiced even from regulatory organizations such as the European Union (Goodman and Flaxman 2016). Even though tree-based methods have been introduced, no existing interpretable unsupervised learning algorithm can accurately partition the feature space both for numerical and categorical data.

## 1.1 Contributions

Motivated by the limitations of existing solutions to interpretable clustering, we develop a novel tree-based unsupervised learning method that leverages traditional optimization and machine learning techniques to obtain interpretable clusters with comparable or superior performance when compared to existing algorithms. Our contributions are as follows:

1. We provide an MIO formulation of the unsupervised learning problem that leads to the creation of globally optimal clustering trees, motivating our new algorithm *Interpretable*

*Clustering via Optimal Trees* (ICOT). Our method builds upon the OCT algorithm and extends it to the unsupervised setting. In ICOT, interpretability is taken into consideration during cluster creation rather than considered as a later analysis step.

2. We provide an implementation of our method with an iterative coordinate-descent approach that scales to larger problems, well-approximating the globally optimal solution. We use widely two established validation criteria, the Silhouette Metric (Rousseeuw 1987) and the Dunn Index (Dunn 1974), as the algorithm's objective function. We propose additional techniques that leverage the geometric principles of cluster creation to improve the algorithm's efficiency. Furthermore, we introduce sampling heuristics that recover fast, high-quality solutions in our empirical experiments and provide a complexity analysis of the local search procedure for one iteration of the algorithm.
3. We develop our algorithm in a way such that tuning of the tree's complexity is redundant. This is enabled by the fact that our loss functions take into account both intra-cluster density as well as inter-cluster separation. The user can optionally tune the algorithm by selecting the maximum depth of the tree and the minimum number of observations in each cluster.
4. We propose a solution to the incorporation of both mixed numerical and categorical data. Our re-weighted distance measure prevents a single variable type from dominating the distance calculation and allows users to optionally tune the balance the two types of covariates.
5. We evaluate the performance of our method against various clustering approaches across synthetic datasets from the Fundamental Clustering Problems Suite (FCPS) (Ultsch 2005) which offer different levels of variance and compactness. We demonstrate ICOT's superior performance against a two-step supervised learning method across both the Silhouette Metric and Dunn Index, offering a 27.8% and 352.7% score improvement respectively. We also compare ICOT against several state-of-the-art methods that represent various clustering approaches, namely partitional, hierarchical, model-based, and density-based clustering. We find that ICOT is competitive against these methods across multiple internal validation criteria.
6. We provide examples of how the algorithm can be used in real-world settings. We perform clustering on patients at risk of cardiovascular disease from the Framingham Heart Study (FHS) dataset (Daniel Levy 2006; Feinleib et al. 1975) to identify similar patient profiles and group economic profiles of European countries during the Cold War (Krim and Hamza 2015). Through these experiments, we illustrate the effect of varying key parameters in the ICOT algorithm. We also compare ICOT to other state-of-the-art algorithms in the FHS experiment and to CUBT in the economic profile experiment. We discuss the interpretability of the methods as well as their performance on the internal validation criteria.
7. Finally, we test the capability of the algorithm to scale to large problem instances using both the FCPS as well as real-world data from a Boston-based bike sharing program. We demonstrate that our suggested heuristic techniques do not significantly impact the quality of the recovered solutions. In addition, our experiments illustrate that ICOT can efficiently handle datasets of sizes up to hundreds of thousands of observations.

The structure of the paper is as follows. In Sect. 2, we formulate the problem of optimal tree creation within an MIO framework. Sect. 3 provides a comprehensive description of the algorithm implementation. In Sects. 4 and 5, we conduct a range of experiments using synthetic and real-world datasets to evaluate the performance and interpretability of our method compared to other state-of-the-art algorithms. In Sect. 6, we investigate the effect of our scaling methods on runtime and solution quality. In Sect. 7, we discuss the key findings from our work and in Sect. 8 we include our concluding remarks.

## 2 MIO formulation

In this section, we present an MIO approach which allows us to construct globally optimal tree-based models in an unsupervised learning setting. In Sect. 2.1, we provide an overview of the MIO framework introduced by Bertsimas and Dunn (2017, 2019). Section 2.2 introduces the validation criteria that are used as objective functions in the optimization problem. In Sect. 2.3, we outline the complete ICOT formulation for one of the loss functions considered.

### 2.1 The OCT framework

The OCT algorithm formulates tree construction using MIO which allows us to define a single problem, as opposed to the traditional recursive, top-down methods that must consider each of the tree decisions in isolation. It allows us to consider the full impact of the decisions being made at the top of the tree, rather than simply making a series of locally optimal decisions, avoiding the need for pruning and impurity measures.

We are given the training data $(\mathbf{X}, \mathbf{Y})$, containing $n$ observations $(\mathbf{x_i}, y_i)$, $i = 1, \dots, n$, each with $p$ features and a class label $y_i \in \{1, \dots, K\}$ as an indicator of which of the $K$ potential labels is assigned to point $i$. We assume without loss of generality that the values of each training vector are normalized such that $\mathbf{x_i} \in [0, 1]^p$. A decision tree recursively partitions the feature space to identify a set of distinct, hierarchical regions that form a classification tree. The final tree $\mathcal{T}$ is comprised of nodes that can be categorized in:

- Branch Nodes: Nodes $t \in \mathcal{T_B}$ apply a split with parameters $\mathbf{a}$ and $b$. For observation $i$, if the corresponding vector $\mathbf{x_i}$ satisfies the relation $\mathbf{a}^T\mathbf{x_i} < b$, the point will follow the left branch from the node. Otherwise it takes the right branch.
- Leaf Nodes: Nodes $t \in \mathcal{T_L}$ assign a class to all the points that fall into them. Each leaf node is characterized by one class which is generally determined by the most frequently occurring class among the observations that belong to it.

First, we formally define the constraints that construct the decision tree. We use the notation $p(t)$ to refer to the parent node of node $t$, and $A(t)$ to denote the set of ancestors of node $t$. We define the split applied at node $t \in \mathcal{T_B}$ with variables $\mathbf{a}_t \in \mathbb{R}^p$ and $b_t \in \mathbb{R}$. The vector $\mathbf{a}_t$ indicates which variable is chosen for the split, meaning that $a_{jt} = 1$ for the variable $j$ used at node $t$. $b_t$ gives the threshold for the split, which is between $[0, 1]$ after normalization of the feature vector. If a branch node does not apply a split, then we model this by setting $\mathbf{a}_t = \mathbf{0}$ and $b_t = 0$. Together, these form the constraint $\mathbf{a}_t^T x < b_t$. The indicator variables $d_t$ are set to 1 for branch nodes and 0 for leaf nodes. Using the above variables, we introduce the following constraints that allows us to model the tree structure (for a detailed analysis of the constraints, see Bertsimas and Dunn (2017)):

$$\sum_{j=1}^{p} a_{jt} = d_t, \quad \forall t \in \mathcal{T}_{\mathcal{B}}, \tag{1}$$

$$0 \le b_t \le d_t, \quad \forall t \in \mathcal{T}_{\mathcal{B}}, \tag{2}$$

$$a_{jt} \in \{0, 1\}, \quad j = 1, \dots, p, \quad \forall t \in \mathcal{T}_{\mathcal{B}} \tag{3}$$

We next enforce the hierarchical structure of the tree. Branch nodes are allowed to apply a split only if their parent nodes apply a split:

$$d_t \le d_{p(t)}, \quad \forall t \in \mathcal{T}_{\mathcal{B}} \backslash \{1\} \tag{4}$$

Next we present the corresponding constraints that track the allocation of points to leaves. For this purpose, we introduce the indicator variables $z_{it} = \mathbb{1}\{x_i \text{ is in node } t\}$ and $l_t = \mathbb{1}\{\text{leaf } t \text{ contains any points}\}$. We let $N_{min}$ be a constant that defines the minimum number of observations required in each leaf. We apply the following constraints:

$$z_{it} \le l_t, \quad \forall t \in \mathcal{T}_{\mathcal{L}}, \tag{5}$$

$$\sum_{i=1}^{n} z_{it} \ge N_{min} l_t, \quad \forall t \in \mathcal{T}_{\mathcal{L}} \tag{6}$$

We also enforce each point to belong to exactly one leaf:

$$\sum_{t \in \mathcal{T}_{\mathcal{L}}} z_{it} = 1, \ i = 1, \dots, n \tag{7}$$

Finally, we introduce constraints that force the assignments of observations to leaves to obey the structure of the tree given by the branch nodes. We want to apply a strict inequality for points going to the lower leaf. To accomplish this, we define the vector $\epsilon \in \mathbb{R}^p$ as the smallest separation between two observations in each dimension $p$, and $\epsilon_{max}$ as the maximum over this vector.

$$a_m^{\mathsf{T}} x_i \ge b_t - (1 - z_{it}), \ i = 1, \dots, n, \quad \forall t \in \mathcal{T}_{\mathcal{B}}, \quad \forall m \in A_R(t) \tag{8}$$

$$a_m^{\mathsf{T}}(x_i + \epsilon) \le b_t + (1 + \epsilon_{max})(1 - z_{it}), \ i = 1, \dots, n, \quad \forall t \in \mathcal{T}_{\mathcal{B}}, \quad \forall m \in A_L(t) \tag{9}$$

In the classification setting the objective function of MIP formulation is comprised of two components, prediction accuracy and tree complexity. The tradeoff between those two parameters is controlled by the complexity parameter $\alpha$. Given the training data $(\mathbf{x_i}, y_i)$, $i = 1 \dots n$, a general formulation of the objective function is the following:

$$\underset{T}{\text{minimize}} \quad R_{xy}(T) + \alpha |T|$$

where $R_{xy}(t)$ is a loss function assessed on training data and $|T|$ is the number of branch nodes in the tree $T$.

The above model can be used as an input for an MIO solver. Empirical results suggest that such a model leads to optimal solutions in minutes when the maximum depth of the tree is small (approximately 4). Effectively, the rate of finding solutions is directly dependent to the number of binary variables $z_{it}$ and therefore a faster implementation was needed for more complex problems. For this reason, the authors introduced the idea of warm starts as the initial starting point of the method. Using a high-quality integer feasible solution as a warm start increases the speed of the algorithm and provides a strong initial upper bound on the final solution. In addition, heuristics, like local search, allow a further speed up as shown in Bertsimas and Dunn (2017, 2019) that leads to a good approximation of the optimal solution.

## 2.2 Loss functions for cluster quality

Clustering validation, the evaluation of the quality of a clustering partition (Maulik and Bandyopadhyay 2002), has long been recognized as one of the vital issues essential to the success of a clustering application (Liu et al. 2010). External clustering validation and internal clustering validation are the two main categories of clustering quality metrics. The main difference lies in whether or not external labels are used to assess the clusters; internal measures evaluate the goodness of a clustering structure without respect to ground-truth labels (Larose and Larose 2014). An example of external validation measure is entropy, which evaluates the "purity" of clusters based on the given class labels (Wu et al. 2009). True class labels are not present in real-world datasets, and thus these cases necessitate the use of internal validation measures for cluster validation.

We will consider two internal validation measures as loss functions for our MIO formulation of our problem. The chosen loss functions consider the global assignment of observations to clusters. The score of a clustering assignment depends on both the compactness of the observations within a single cluster, as well as its separation from observations in other clusters. Compactness measures how closely related the objects in a cluster are. Separation measures how distinct a cluster is from other clusters. Several internal validation metrics have been proposed to balance these two objectives (Liu et al. 2010). Two common criteria, the Silhouette Metric and Dunn Index, are outlined below.

*Silhouette Metric* The Silhouette Metric introduced by Rousseeuw (1987) compares the distance from an observation to other observations in its cluster relative to the distance from the observation to other observations in the second closest cluster. The Silhouette Metric for observation $i$ is computed as follows:

$$s(i) = \frac{b(i) - a(i)}{\max(b(i), a(i))}, \tag{10}$$

where $a(i)$ is the average distance from observation $i$ to the other points in its cluster, and $b(i)$ is the average distance from observation $i$ to the points in the second closest cluster. In other words, $b(i) = \min_k b(i, k)$ where $b(i, k)$ is the average distance of $i$ to points in cluster $k$, minimized over all clusters $k$ other than the cluster that point $i$ is assigned to. From this formula it follows that $-1 \leq s(i) \leq 1$.

When $s(i)$ is close to 1, one may infer that the $i$th sample has been "well-clustered", i.e. it was assigned to an appropriate cluster. If observation $i$ has score close to 0, it suggests that it could also be assigned to the nearest neighboring cluster with similar quality. If $s(i)$ is close to -1, one may argue that such a sample has been assigned to the wrong partition. These individual scores can be averaged to reflect the quality of the global assignment.

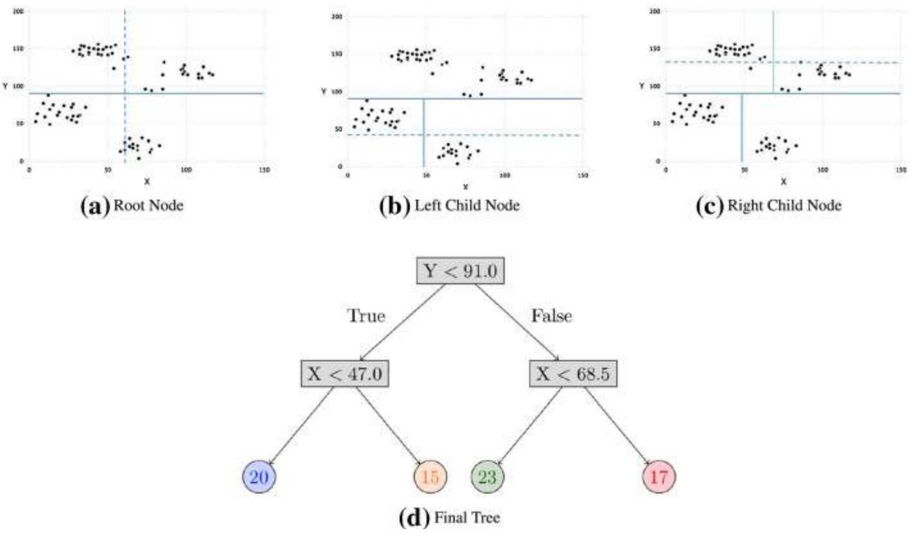

**Fig. 1** An example of a clustering tree built on the Ruspini dataset

$$SM = \frac{1}{n}\sum_{i=1}^{n} s(i), \tag{11}$$

*Dunn Index* The Dunn Index (Dunn 1974) characterizes compactness as the maximum distance between observations in the same cluster, and separation as the minimum distance between two observations in different clusters. The metric is computed as the ratio of the minimum inter-cluster separation to the maximum intra-cluster distance.

$$DI = \frac{\min\limits_{1\le i<j\le m} \delta(C_i, C_j)}{\max\limits_{1\le k\le m} \Delta_k}, \tag{12}$$

where we let the maximum distance of cluster $C$ be denoted by $\Delta_C$ and the distance between clusters $i$ and $j$ be denoted by $\delta(C_i, C_j)$. If the dataset contains compact and well-separated clusters, the distance between the clusters is expected to be large and the diameter of the clusters is expected to be small. Thus, large values of the metric correspond to better partitions and signify that the distance between clusters is large relative to the distance between points within a cluster.

We provide an example to illustrate how an internal validation criterion can be used to geometrically partition the space through a decision tree. In Fig. 1, we cluster observations from the Ruspini dataset (Ruspini 1970) using the Silhouette Metric. In Fig. 1a, the algorithm identifies the best candidate splits on both features, $x_1$ and $x_2$, at the root node, and then compares their resultant cluster scores, as measured by the Silhouette Metric. The $x_2$ split provides a better cluster assignment, so this split is chosen as denoted by the solid line. After the first data partition, splits are considered for each of the child nodes, which corresponds to further separating the lower and upper halves of the graph. Upon identification of candidate $x_1$ and $x_2$ splits on the left child node, the $x_1$ split is chosen based on the Silhouette Metric of the global cluster assignment, as shown in Fig. 1b. The process is then completed for the right child node, and an $x_1$ split is also chosen here in Fig. 1c. Now, each of the four leaves is evaluated, which corresponds to exploring splits in the four quadrants defined by the solid blue lines. There are no splits within any of these four leaves that improve the overall score of the clustering assignment, so the tree construction is complete. The final tree is shown in Fig. 1d. The resultant tree provides a final partition which clearly elucidates the distinguishing features of each group. We note that this example demonstrates a *greedy* tree construction. In the ICOT algorithm, all splits would be subsequently reoptimized with respect to the overall tree. However, in this case the greedy tree is able to provide the optimal partition.

Note that both of our considered criteria require the definition of at least two clusters since they both involve a pairwise distance computation between clusters to measure separation. As a result, calculations for the null-case are not considered. The determination of the best internal validation criterion for a given dataset remains an open question in the field of unsupervised learning theory (Liu et al. 2010). As stated in Halkidi et al. (2001), the Dunn Index is more computationally expensive and more sensitive to noisy data compared to the Silhouette Metric. It is also less robust to outliers compared to the Silhouette Metric which averages an observation-based score for the global assignment. However, empirical results suggest that the Dunn Index has superior performance in returning intuitive partitions of the data when they are well-separated.

## 2.3 The ICOT formulation

The OCT framework needs to be modified to address an unsupervised learning task. We present changes in the original MIO formulation of OCT to be able to partition the data space into distinct clusters following the same structure and notation as in Sect. 2.1. We outline in detail the model for the Silhouette Metric loss function. The Dunn Index formulation follows closely and is thus omitted. There are two primary modifications in the ICOT formulation compared to the OCT:

1. The objective function is comprised solely by the chosen cluster quality criterion, such as the Silhouette Metric, and does not include any penalty for the tree complexity. The separation component of the validation criterion naturally controls the complexity of the tree and thus for the ICOT formulation the complexity parameter is rendered redundant.
2. Each leaf of the tree is equivalent to a cluster. Observations in different leaves are not allowed to belong to the same cluster.

The objective of the new formulation is to maximize the Silhouette Metric (*SM*) of the overall partition. The Silhouette Metric quantifies the difference in separation between a point and points in its cluster, versus the separation between that point and points in the second closest cluster.

Let $d_{ij}$ be the distance (i.e. Euclidean) of observation $i$ from observation $j$. We define $K_t$ to be number of points assigned assigned to cluster $t$.

$$K_t = \sum_{i=1}^{n} z_{it}, \quad \forall t \in \mathcal{T}_{\mathcal{L}} \tag{13}$$

We define $c_{it}$ to be the average distance of observation $i$ from cluster $t$:

$$c_{it} = \frac{1}{K_t} \sum_{j=1}^{n} d_{ij} z_{jt}, \quad \forall i = 1, \dots, n, \ t \in \mathcal{T}_{\mathcal{L}}. \tag{14}$$

We define $r_i$ to be the average distance of observation $i$ from all the points assigned in the same cluster:

$$r_i = \sum_{t \in \mathcal{T}_{\mathcal{L}}} c_{it} z_{it}, \quad \forall i = 1, \dots, n. \tag{15}$$

We then let $q_i$ denote the minimum average distance of observation $i$ to the observations from the next closest cluster. We define auxiliary variables $\gamma_{it}$ to enforce this constraint, such that $\gamma_{it}$ an indicator of whether $t$ is the second closest cluster for observation $i$.

$$q_i \geq \sum_{t \in \mathcal{T}_{\mathcal{L}}} \gamma_{it} c_{it}, \ i = 1, \dots, n. \tag{16}$$

$$\sum_{t \in \mathcal{T}_{\mathcal{L}}} \gamma_{it} = 1, \ i = 1, \dots, n. \tag{17}$$

$$\gamma_{it} \leq M(1 - z_{it}), \ i = 1, \dots, n, \quad \forall t \in \mathcal{T}_{\mathcal{L}}. \tag{18}$$

Finally, to define the Silhouette Metric of observation $i$, we will need the maximum value between $r_i$ and $q_i$ which normalizes the metric.

$$m_i \geq r_i, \ i = 1, \dots, n. \tag{19}$$

$$m_i \geq q_i, \ i = 1, \dots, n. \tag{20}$$

The score for the Silhouette Metric for each observation is computed as $s(i)$ and the overall score for the clustering assignment is then the average overall all the Silhouette Metric scores from the training population:

$$s_i = \frac{q_i - r_i}{m_i}, \ i = 1, \dots, n. \tag{21}$$

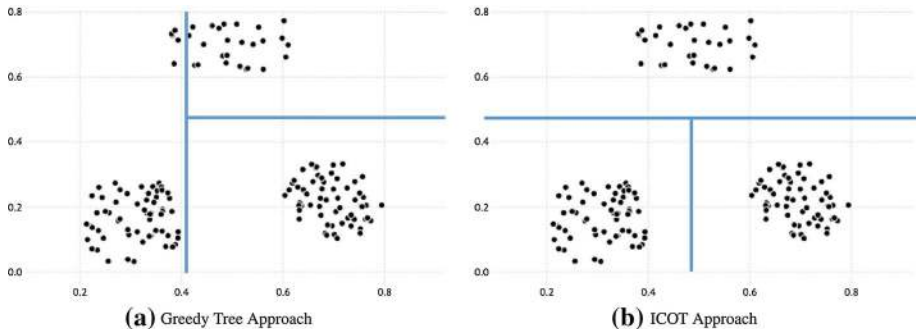

**Fig. 2** An illustration in a synthetic example of a local optimum that might be identified by a greedy unsupervised learning algorithm

$$SM = \frac{1}{n}\sum_{i=1}^{n} s_i. \tag{22}$$

Putting all of this together gives the following MIO formulation for the ICOT model:

$$\begin{aligned}
\underset{x}{\text{minimize}} \quad & -\frac{1}{n}\sum_{i=1}^{n} s_i \\
\text{subject to} \quad & s_i = \frac{q_i - r_i}{m_i}, \quad i = 1, \dots, n, \\
& m_i \geq q_i, \quad i = 1, \dots, n, \\
& m_i \geq r_i, \quad i = 1, \dots, n, \\
& q_i \geq \sum_{t \in \mathcal{T}_\mathcal{L}} \gamma_{it} c_{it}, \quad i = 1, \dots, n, \\
& \sum_{t \in \mathcal{T}_\mathcal{L}} \gamma_{it} = 1, \quad i = 1, \dots, n, \\
& \gamma_{it} \leq M(1 - z_{it}), \quad i = 1, \dots, n, \quad \forall t \in \mathcal{T}_\mathcal{L}, \\
& r_i = \sum_{\forall t \in \mathcal{T}_\mathcal{L}} c_{it} z_{it}, \quad i = 1, \dots, n, \\
& c_{it} = \frac{1}{K_t}\sum_{j=1}^{n} d_{ij} z_{jt}, \quad i = 1, \dots, n, \forall t \in \mathcal{T}_\mathcal{L}, \\
& K_t = \sum_{i=1}^{n} z_{it} \quad \forall t \in \mathcal{T}_\mathcal{L}, \\
& \sum_{j=1}^{p} a_{jt} = d_t, \quad \forall t \in \mathcal{T}_\mathcal{B}, \\
& 0 \leq b_t \leq d_t, \quad \forall t \in \mathcal{T}_\mathcal{B}, \\
& d_t \leq d_{p(t)}, \quad \forall t \in \mathcal{T}_\mathcal{B} \backslash \{1\}, \\
& z_{it} \leq l_t, \quad \forall t \in \mathcal{T}_\mathcal{L}, \\
& \sum_{i=1}^{n} z_{it} \geq N_{min} l_t, \quad \forall t \in \mathcal{T}_\mathcal{L}, \\
& \sum_{t \in \mathcal{T}_\mathcal{L}} z_{it} = 1, \quad i = 1, \dots, n, \\
& a_m^\intercal x_i \geq b_t - (1 - z_{it}), \quad i = 1, \dots, n, \quad \forall t \in \mathcal{T}_\mathcal{B}, \ m \in A_R(t), \\
& a_m^\intercal (x_i + \varepsilon) \leq b_t + (1 + \varepsilon_{max})(1 - z_{it}), \quad i = 1, \dots, n, \quad \forall t \in \mathcal{T}_\mathcal{B}, \in A_L(t), \\
& a_{jt}, d_t \in \{0, 1\}, \quad j = 1, \dots, p, \quad \forall t \in \mathcal{T}_\mathcal{B}, \\
& z_{it}, l_t \in \{0, 1\}, \quad i = 1, \dots, p, \quad \forall t \in \mathcal{T}_\mathcal{L}, \\
& \gamma_{it} \in \{0, 1\}, \quad i = 1, \dots, n, \quad \forall t \in \mathcal{T}_\mathcal{L}.
\end{aligned}$$

Figure 2 illustrates the benefit of an optimization framework over greedy tree construction. The synthetic dataset seen in the figure has two dense lower regions and one less dense upper region. In a greedy approach, the first split separates the lower clusters and cuts through the upper cluster. While it is clearly better to split horizontally first (since it does not split a region), a greedy algorithm chooses the split without consideration of the possibility of future splits. Therefore, if the tree can only make one split, it is better to separate the lower clusters since they have such high density. ICOT's optimization approach considers the global tree structure, avoiding such pitfalls and identifying the true optimal partition. It starts by making a horizontal split and subsequently separates the high-density lower regions without cutting through the upper cluster. A globally optimal partition has Silhouette Metric score equal to 0.758 whereas the greedy tree yields only 0.688.

## 3 Algorithm overview

In this section, we outline the practical details of the algorithm implementation. Section 3.1 describes ICOT's coordinate-descent algorithm that approximates the globally optimal solution in an efficient and intuitive manner. Section 3.2 addresses the challenge of computing distance scores in the presence of mixed numerical and categorical variables and introduces a solution for appropriately handling distance in this setting. Finally, in Sect. 3.3 we propose heuristics in our algorithm implementation which leverage the underlying structure of the data to more quickly traverse the search space and identify high-quality solutions.

### 3.1 Coordinate-descent implementation

The MIO formulation provides the optimization framework for our problem solving approach. In practice, the algorithm is implemented using a coordinate-descent procedure which allows it to scale to much higher dimensions than directly solving the optimization problem. The implementation provides a good approximation of the optimal solution while still abiding by the same core principles of the original formulation.

ICOT initializes a greedy tree and subsequently runs a local search procedure until the objective value, a cluster quality measure, converges. This process is repeated from many different starting greedy trees, generating many candidate clustering trees. The final tree is chosen as the one with the highest cluster quality score across all candidate trees. This single tree is returned as the output of the algorithm.

The initial greedy tree is constructed from a single root node. A split is made on a randomly chosen feature by scanning over all potential thresholds for splitting observations into the lower and upper leaves. At each candidate split, we compute the global score for the potential assignment. We choose the split threshold that gives the highest score and update the node to add the split if this score improves upon the global score of the current assignment. We perform the same search for each leaf that gets added to the tree, continuing until either the maximum tree depth is reached or no further improvement in our objective value is achieved through further splitting on a leaf.

Following the creation of the greedy tree, a local search procedure is performed to optimize the clustering assignment. Tree nodes are visited in a randomly chosen order, and various modifications are considered. A branch node has two options; it can be deleted, in which case it is replaced with either its lower or upper subtree, or a new split can be made at the node using a different feature and threshold. A leaf node can be further split into two leaves. At each considered node, the algorithm finds the best possible change and updates the tree structure only if it improves the objective from its current value. All nodes get added back to the list of nodes to search once an improvement has been found. The algorithm terminates when the objective value converges. The algorithm is explained further in Algorithm 1.

**Algorithm 1** ICOT Algorithm.

```
Input: Feature vectors $\mathbf{x}^1, \ldots, \mathbf{x}^n$
Output: Cluster assignments $y^1, \ldots, y^n$
 1: Initialize a greedy tree, with clusters $c_1, \ldots, c_K$ and loss $l_0$.
 2: Indices to search: $S = \{1, \ldots, K\}$; Loss: $l = l_0$.
 3: while $S$ not empty do
 4:     for all $k \in S$ do
 5:         if $C_k$ is leaf node then
 6:             Find best possible new split with loss $\hat{l}$.
 7:         else
 8:             Find best possible node modification, either through a different split or split deletion, with loss $\hat{l}$.
 9:         end if
10:         if $\hat{l} < l$ then
11:             Update tree and add all leaves to $S$. $l \leftarrow \hat{l}$.
12:         else
13:             Remove $k$ from $S$.
14:         end if
15:     end for
16: end while
```

The user can specify to optimize either the Silhouette Metric or Dunn Index described in Sect. 2.2. These metrics penalize low separation, which naturally limits the depth of the tree. In traditional tree-based algorithms such as CART or OCT, the loss function improves with successive tree splits. Thus, these methods require a pruning step or additional parameter, such as a complexity penalty of maximum depth, to control the tree size. ICOT does not require the explicit control of tree size due to this natural balance between separation and compactness in the cluster quality metrics. This eliminates the need for setting an explicit $K$ parameter, which is typically required in both partitional and hierarchical clustering methods. The tree continues to split until further splits no longer improve the quality of the overall assignment, and so the final number of leaves represents the optimal number of clusters.

The user can enforce further structure on the tree through setting the optional minimum bucket parameter, $N_C$. This controls the minimum number of observations that are required in each leaf and effectively in each cluster. Note that there is not a monotonic relationship between the magnitude of $N_C$ and the number of leaves (clusters) generated by the algorithm. Smaller minimum buckets may lead to smaller cluster counts due to the positive effect of isolated outlier clusters on the metrics; overfitting is difficult to quantify in an unsupervised learning setting because there is no ground truth to compare against, and thus the metrics do not naturally penalize single outliers. Thoughtful choice of the minimum bucket parameter allows ICOT to avoid creating clusters of single or small sets of outliers, which often lack meaning and generalizability in grouping tasks. Traditional methods, such as $K$-means, deal with outliers by increasing the $K$ parameter and forcing the algorithm to provide with a higher number of clusters. $N_C$ can significantly affect the clustering solution and should be cross-validated or experimented on in order to get accurate and intuitive results from ICOT. The maximum depth can be used to impose an upper bound on the number of clusters if desired, although this parameter does not address potential outlier issues.

The ICOT algorithm is implemented in Julia (Bezanson et al. 2012) and is available to academic researchers under a free academic license.[1]

## 3.2 Mixed-variable handling

Both the Silhouette Metric and Dunn Index assess the quality of a given cluster assignment using the pairwise distance matrix of the observations. Distance is quantified differently for numerical and categorical variables and thus must be adjusted appropriately in the presence of mixed variable types. In the case of continuous features, the data are first normalized to be in the [0, 1] range. The pairwise numerical distance matrix $d^N$ is computed using the Euclidean distance between each pair of normalized variables. In the case of categorical features, distance is defined based on whether the observations take on different values. For example, if one observation takes on category $A$ and another observation takes on category $B$ on a given feature, the distance on this feature will be 1. The distance is zero if the observations take on the same value. For each pair of observations, these indicators are summed over all categories to define the categorical feature distance matrix $d^C$.

When the feature space includes both numerical and categorical variables, special consideration must be given to avoid over-weighting the categorical variables. In particular, categorical variables are often one-hot encoded (i.e. converted to binary 0/1 columns) to allow them to be treated as numerical in machine learning methods. This adjustment is insufficient in our case as it will result in placing too high of an importance on the categorical distance.

We handle this issue by taking a linear combination of the two separate distance matrices for numerical and categorical variables. We first compute separate distance matrices for the numerical and categorical features. We let $S^N$ denote the set of indices for the numerical features, and $S^C$ denote the categorical indices. The computations for $d^N$ and $d^C$ are explicitly defined in Eqs. 23 and 24.

$$d_{ij}^N = \sqrt{\sum_{k \in S^N} (x_k^i - x_k^j)^2} \tag{23}$$

$$d_{ij}^C = \sum_{k \in S^C} \mathbb{1}\{x_k^i \neq x_k^j\} \tag{24}$$

We then compute the final distance matrix by taking a linear combination of these two matrices, given in Eq. 25.

$$d_{ij} = \alpha d_{ij}^N + (1 - \alpha) d_{ij}^C \tag{25}$$

By default, the two distances are weighted according to their proportion of all covariates, so $\alpha = \frac{|S^N|}{|S^N| + |S^C|}$. The user can also specify an alternative $\alpha$ parameter. At $\alpha = 1$, the distance matrix only accounts for numerical covariates, whereas $\alpha = 0$ only considers disagreements in categorical variables.

[1] Please email icot@mit.edu to request an academic license for the ICOT package.

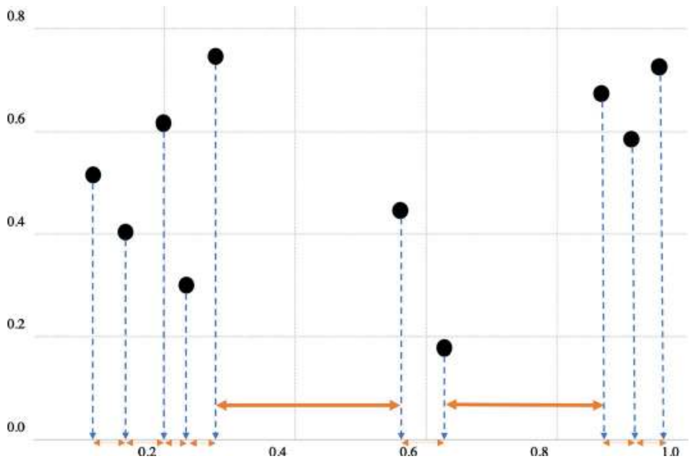

**Fig. 3** An example of the restricted geometric search function

## 3.3 Scaling methods

Our coordinate-descent procedure is more computationally intensive than the original OCT algorithm due to unique characteristics of clustering. In particular, we must compute a global clustering quality score at each split threshold evaluation, unlike classification tasks in which the loss change for a potential split can be assessed locally at the node. This global score assessment involves higher computational effort per split evaluation and thus motivates the development of more efficient search procedures. We introduce two scaling methods to take advantage of the geometric intuition behind cluster creation as well as existing clustering methods. We furthermore propose a subsampling approach to allow the algorithm to scale to much larger problems.

### 3.3.1 Restricted geometric search space

ICOT leverages the geometric structure of the feature space by restricting the set of candidate splits to those with sufficient separation. An exhaustive search of candidate splits on a given numerical feature requires $n_k - 1$ threshold evaluations, where $n_k$ is the number of observations in a given node. This is due to the fact that there are exactly $n_k - 1$ different possible partitions of the data on the given feature at node $k$ (less if multiple observations have the same value on this feature).

To improve the efficiency of our algorithm, we only consider a subset of these thresholds. For any feature, we refer to a threshold's gap as the separation between the observations directly below and above it. Since the quality of a clustering assignment is directly tied to the distance separating distinct clusters, the cluster quality will be superior when considering thresholds with large gaps. We take advantage of this intuition by skipping over thresholds with small gaps.

We control the extent of search space restriction through the parameter $T$. When considering a numerical feature split at node $k$, all threshold gaps for observations in the node are sorted ($n_k - 1$ values). Only thresholds above the $T$th percentile of gap size are considered.

For example, if $T = .9$ and $n_k = 100$, only the thresholds with the 10 largest gaps are considered, reducing the number of computations per node by 90%.

Figure 3 provides an illustration of how the Restricted Geometric Search would be applied in a simple example. When $T = 0.7$, ICOT will investigate only the top 30% of the gaps between observations. Thus only the larger, bold, gaps would be potential splits for a branch node that considers the covariate corresponding to the horizontal axis.

### 3.3.2 *K*-means warm start

We also employ warm starts to more efficiently identify high-quality clustering trees. We leverage the *K*-means algorithm to partition the data into clusters and use OCT to generate a tree that reasonably separates these clusters. This becomes the starting point of ICOT's coordinate descent algorithm. The algorithm first runs *K*-means on the original data across various *K* parameters and selects the assignment that optimizes our chosen cluster quality criterion. The resulting assignments are used as class labels for the construction of a supervised classification tree using OCT. ICOT's coordinate-descent procedure then begins from the resultant OCT tree rather than a greedy tree. Each leaf from the OCT tree becomes a separate cluster when initializing the ICOT algorithm, even though the predicted class labels may match between multiple leaves. Overall, the *K*-means warm start expedites tree initialization and improves the efficiency of the search procedure.

### 3.3.3 Bootstrapping

We introduce bootstrapping on the number of input observations, $N$. Our goal is to make the algorithm amenable to solve problems of larger sample size. This procedure involves subsampling a reduced population of size $N_r$ and solving smaller problems $N_{\text{rep}}$ times. This allows the algorithm to scale linearly with respect to the number of repetitions. It can be easily parallelized as it contains multiple independent sub-problems. Each iteration samples $N_r$ observations without replacement and runs ICOT, returning a tree model which is then evaluated on a validation population. Upon completion of all $N_{\text{rep}}$ iterations, the algorithm selects the best performing tree model on the validation criterion. Beyond improving the speed of the algorithm, bootstrapping provides a lot of flexibility to the user. The choice of $N_r$ and $N_{\text{rep}}$ may vary depending on the time constraints and the required quality of the final solution. We explore the latter in greater detail in Sects. 6.2.3–6.2.4.

### 3.3.4 Complexity analysis

We provide a brief analysis of the worst-case complexity for each iteration of the coordinate-descent implementation of the algorithm. The argument is an extension of the complexity analysis for Optimal Classification Trees (Dunn 2018). First, we consider the complexity of calculating our cluster quality criteria.

An initial step for the computation of any score is the construction of a distance matrix that contains all the distances between each point $i, j \in [N]$, the training population. The matrix creation involves $\frac{n(n-1)}{2}$ calculations, which has complexity $\mathcal{O}(n^2)$.

*Silhouette metric (SM)* For each observation $i$, we must compute the average distance between $i$ and the members of each cluster. If we have $T$ nodes, and each cluster contains at most $n$ points, this has complexity $\mathcal{O}(nT)$. We need to find the distance to the next-closest cluster for which $i$ is not a member. As we iterate through each of the clusters, we track the closest distance found so far and update if it improves. We note that the number of clusters is $\mathcal{O}(T)$ and is upper bounded by the total number of nodes. This computation is repeated for all $n$ observations. Thus, the complexity of computing the Silhouette Metric is

$$cp_{SM} = \mathcal{O}(n(nT)) = \mathcal{O}(n^2T)$$

*Dunn index (DI)* For each cluster, we must find the largest distance between any two points within the cluster and the smallest distance between a point in the cluster and outside of the cluster. This involves sorting at worst all pre-computed pairwise distances of which there are $\frac{n(n-1)}{2}$, giving complexity $\mathcal{O}(Tn^2 \log(n))$. As we iterate through the sorted values, we track the highest intra-cluster and lowest inter-cluster distances and update if we find a value that improves either metric. In total, this yields complexity

$$cp_{DI} = \mathcal{O}(Tn^2 \log(n)) = \mathcal{O}(Tn^2 \log(n)).$$

We now move on to the calculation of the algorithm's complexity in each iteration. Once an initial tree is constructed, each inner iteration of ICOT's local search consists of identifying the best potential split change at a given node. For each of the $p$ features, there are at most $n-1$ potential split thresholds (if all observations are in this node). At each of these thresholds, we must (1) find the assignment of all points to clusters (i.e. tree leaves), which has complexity $\mathcal{O}(nT)$, where $T$ is the total number of nodes in the tree and (2) calculate the cluster quality criterion $cp$, either $cp_{SM}$ or $cp_{DI}$. Thus, the inner iteration has complexity $\mathcal{O}(np(nT + cp))$. We must repeat this for each leaf, which adds a factor of $T$.

Ultimately, one iteration of ICOT when trained on the Silhouette Metric has worst-case complexity:

$$\mathcal{O}(npT(nT + n^2T)) = \mathcal{O}(n^2pT^2 + n^3pT^2) = \mathcal{O}(n^2pT^2 + n^3pT^2)$$

When optimizing the Dunn Index, ICOT's complexity is:

$$\mathcal{O}(npT(nT + n^2T \log(n))) = \mathcal{O}(n^2pT^2 + n^3pT^2 \log(n))$$

Both of these results demonstrate that each iteration of ICOT is highly sensitive to scaling with respect to $n$, with a higher cost when training on the Dunn Index (by a factor of $\log(n)$). Through the geometric search in Sect. 3.3.1, we are able to reduce the number of splits considered by a constant factor; with a threshold of 0.99, rather than considering $np$ splits, we only consider $0.01 * np$ splits. Additionally, the warm-starts explained in Sect. 3.3.2 provide higher quality starting solutions which reduces the number of iterations required to reach convergence and thus reduces runtime. This is demonstrated empirically in Sect 6. Finally, the sub-sampling method introduced in Sect. 3.3.3 allows us to leverage ICOT for arbitrarily large problems; Sect. 6 also shows empirical evidence that the resultant trees still generalize well to the larger datasets despite only being trained on a subset.

## 4 Experiments based on synthetic datasets

In this section, we present results of ICOT across various synthetic datasets. We use these experiments to assess the quality of the algorithm's solution on both validation criteria. Sect. 4.2 compares ICOT to other popular clustering alternatives in terms of their ability to recover high-quality clustering assignments when training on both the Silhouette Metric and Dunn Index. We also examine the tradeoff between the two metric scores when training on one and evaluating on the other.

### 4.1 Experimental setup

We evaluated ICOT on the Fundamental Clustering Problems Suite datasets (FCPS) (Ultsch 2005), a standard set of synthetic datasets for unsupervised learning evaluation. These datasets have ground truth cluster labels, which allow for an objective comparison of cluster quality. Our experiments only consider nine of the 10 FCPS datasets, as the tenth contains no true clusters and thus does not offer insight into clustering algorithms.

The ICOT experiments use the "fully scaled" version of the algorithm, with a $K$-means warm start and a geometric threshold of 0.99. We left the minimum bucket size at its default value (1 observation) and restricted the maximum depth of the tree to depth 3. We left the $\alpha$ parameter at its default value. We ran 100 random restarts of the algorithm in each experiment.

We consider six alternative clustering algorithms which span a range of methodological approaches and interpretations. The following methods are compared:

1. Optimal Classification Trees Hybrid Method (`OCT`) A two-step $K$-means and OCT hybrid approach, in which $K$-means clusters serve as class labels for a supervised multiclass classification problem. Each observation is assigned a label based on the predicted class of its leaf. `OCT` is implemented using the InterpretableAI package in Julia (Bertsimas and Dunn 2017, 2019).
2. $K$-`means++` We run $K$-means with a $K$-means++ initialization, which was introduced by Arthur and Vassilvitskii (2007) and has been shown to improve upon a standard $K$-means implementation. $K$-`means++` has been incorporated in the `ClusterR` R package (Mouselimis 2019). We run the method with 100 random restarts and a maximum of 100 clustering iterations.
3. Hierarchical Clustering (`Hclust`) Hierarchical clustering is the most popular agglomerative clustering method. It combines individual points into clusters using a linkage measure until all points end up in a single cluster, returning a single dendogram that exhaustively links all individual points (Hastie et al. 2009). While this is a tree-based method, it does not have binary splits and cannot be explicitly represented as a function of the features. `Hclust` is implemented in R using average linkage.
4. Gaussian Mixture Models (`GMM`): `GMM` assigns observations to clusters characterized by Gaussian distributions. The algorithm uses expectation-maximization (EM) to find the parameters for each of $K$ Gaussian distributions, each representing a cluster (Hastie et al. 2009). This approach has a key advantage of accounting for cluster variance in assignment, which is a deficiency of traditional methods such as $K$-means. For each observation, this method returns a soft-assignment, which gives a probability of belonging to each cluster. To make this assignment amenable to our quantitative comparison which requires an explicit assignment, we assign observations to their most likely cluster. `GMM`

**Table 1** Comparison of methods across the FCPS datasets, when trained and evaluated on the Silhouette Metric

| Data | (N,P) | ICOT | OCT | *K*-means++ | Hclust | GMM | DBSCAN | PCT | Truth |
|---|---|---|---|---|---|---|---|---|---|
| Atom | (800.2) | 0.503 | 0.433 | 0.611* | 0.593 | 0.565 | 0.540 | 0.516 | 0.311 |
| Chainlink | (1000.2) | 0.396 | 0.28 | 0.479 | 0.496* | 0.409 | 0.357 | 0.312 | 0.158 |
| EngyTime | (4096.2) | 0.573* | 0.4 | 0.439 | 0.379 | 0.433 | 0.450 | 0.377 | 0.398 |
| Hepta | (212.3) | 0.453 | 0.332 | 0.702* | 0.702* | 0.608 | 0.702* | 0.368 | 0.702* |
| Lsun | (400.2) | 0.549 | 0.534 | 0.569* | 0.554 | 0.537 | 0.439 | 0.564 | 0.439 |
| Target | (770.2) | 0.629* | 0.409 | 0.593 | 0.619 | 0.578 | 0.533 | 0.516 | 0.295 |
| Tetra | (400.3) | 0.504* | 0.266 | 0.504* | 0.504* | 0.504* | 0.504* | 0.307 | 0.504* |
| TwoDiamonds | (800.2) | 0.486* | 0.486* | 0.486* | 0.485 | 0.412 | 0.266 | 0.486* | 0.486* |
| WingNut | (1070.2) | 0.422 | 0.393 | 0.426* | 0.418 | 0.407 | 0.384 | 0.422 | 0.384 |
| Count best/tie | | 4 | 1 | 6 | 3 | 1 | 2 | 1 | 3 |
| Average score | | 0.502 | 0.393 | 0.534 | 0.528 | 0.495 | 0.464 | 0.430 | 0.409 |
| SD score | | 0.074 | 0.089 | 0.091 | 0.101 | 0.081 | 0.126 | 0.095 | 0.153 |

The asterisks indicate the best score across all algorithms for each criterion

is implemented in the `ClusterR` R package (Mouselimis 2019). We run the method with 20 EM and *K*-means iterations and confirmed that the results stabilize by this point. We compute observation distances using Euclidean distance.

5. Density-based Spatial Clustering of Applications with Noise (`DBSCAN`) `DBSCAN` is a popular method that constructs clusters based on the highest density regions of a dataset (Ester et al. 1996). `DBSCAN` does not return a complete assignment; outliers in low-density areas are left out of any clusters. While this exclusion approach makes the method robust to outliers, it complicates quantitative evaluation. To allow for a fair comparison on the internal validation metrics, we assign each outlier point to the most common cluster of its five nearest neighbors. If all neighbors are also unassigned, we assign the point to its own cluster. This method is implemented in the `DBSCAN` package in R (Hahsler et al. 2019), with additional post-processing to complete the outlier assignment.
6. Predictive Clustering Trees (`PCT`): Predictive clustering trees build recursive binary decision trees for clustering tasks (Blockeel et al. 2000). The methodology is implemented in Java through the `Clus` package. We adopt the default “VarianceReduction” splitting heuristic.

We are unable to present synthetic comparisons to other recent work in interpretable clustering, such as CUBT, as there are no available implementations of the algorithms. We present results of ICOT against the CUBT experiments presented by Fraiman et al. (2013) in Sect. 5.3.

We run all of the comparison methods on normalized data. ICOT normalizes the distance matrix within the algorithm, and we input a normalized dataset into the other comparison method functions. For each of the comparison methods, we tune key parameters to optimize the Silhouette Metric (or Dunn Index). In *K*-`means++`, `Hclust`, and `GMM`, we tune the number of clusters $K \in [2, 10]$. `DBSCAN` does not have an explicit $K$ parameter, but the $\varepsilon$ parameter informs the neighborhood size when constructing clusters; larger $\varepsilon$ values generally translate to larger clusters (and lower $K$). We tune $\varepsilon \in [0.1, 0.11, 0.12 \dots, 1.0]$.

**Table 2** Comparison of methods across the FCPS datasets, when trained and evaluated on the Dunn Index

| Data | (N,P) | ICOT | OCT | *K*-means++ | Hclust | GMM | DBSCAN | PCT | Truth |
|---|---|---|---|---|---|---|---|---|---|
| Atom | (800,2) | 0.137 | 0.035 | 0.052 | 0.097 | 0.048 | 0.371* | 0.064 | 0.371* |
| Chainlink | (1000,2) | 0.028 | 0.013 | 0.038 | 0.037 | 0.016 | 0.265* | 0.018 | 0.265* |
| EngyTime | (4096,2) | 0.064* | 0.002 | 0.005 | 0.014 | 0.004 | 0.029 | 0.002 | 0.000 |
| Hepta | (212,3) | 0.357 | 0.162 | 1.080* | 1.080* | 0.482 | 1.080* | 0.293 | 1.080* |
| Lsun | (400,2) | 0.077 | 0.027 | 0.056 | 0.071 | 0.117* | 0.117* | 0.026 | 0.117* |
| Target | (770,2) | 0.550* | 0.011 | 0.029 | 0.550* | 0.113 | 0.117 | 0.013 | 0.253 |
| Tetra | (400,3) | 0.200* | 0.044 | 0.200* | 0.200* | 0.200* | 0.200* | 0.046 | 0.200* |
| TwoDiamonds | (800,2) | 0.044 | 0.022 | 0.031 | 0.049* | 0.021 | 0.030 | 0.022 | 0.022 |
| WingNut | (1070,2) | 0.063* | 0.020 | 0.026 | 0.036 | 0.016 | 0.063* | 0.063* | 0.063* |
| Count best/tie | | 4 | 0 | 2 | 4 | 2 | 6 | 1 | 6 |
| Average score | | 0.169 | 0.037 | 0.169 | 0.237 | 0.113 | 0.253 | 0.061 | 0.264 |
| StdD score | | 0.176 | 0.048 | 0.347 | 0.358 | 0.153 | 0.330 | 0.090 | 0.330 |

The asterisks indicate the best score across all algorithms for each criterion

**Table 3** Comparison of internal validation scores by choice of training criterion in the ICOT algorithm

| Training criterion | Silhouette metric | Dunn index |
|---|---|---|
| Silhouette metric | 0.475 | 0.149 |
| Dunn index | 0.416 | 0.177 |

Finally, PCT matches our methodology most closely and does not require an explicit cluster number ($K$) or density threshold ($\varepsilon$); for this algorithm, we simply tune the maximum depth from 1 to 3. In all cases, we select the parameter value that yields the best internal validation score on the metric of interest.

In the following experiments, all results are averaged over five experiments per algorithm and parameter combination. All experiments were conducted on two CPUs of type 2 socket Intel E5-2690 v4 2.6 GHz/35M Cache; 16GB of NUMA enabled memory were used per CPU.

## 4.2 Solution quality

In these experiments, we look to assess various clustering methods in terms of their recovery of high-quality solutions, as measured by both the Silhouette Metric and the Dunn Index. We additionally investigate the performance of the "true" cluster labels on both of these criteria.

Tables 1 and 2 show the results of these methods along with the true FCPS labels, evaluated with both the Silhouette Metric and Dunn Index. The runtimes for all methods are included in the "Appendix".

ICOT dominates the two-step supervised learning method in all cases for both metrics, offering an average Silhouette Metric improvement of 27.8% and Dunn Index improvement of 352.7% over OCT. This demonstrates the advantage of building clusters directly through a tree-based approach rather than using a hybrid supervised learning method that applies a tree to cluster labels *a posteriori*.

ICOT matches or outperforms the best alternative clustering method in 4/9 cases with both the Silhouette Metric and with the Dunn Index. ICOT ties or beats $K$-`means++` in 7/9 cases on the Dunn Index and 4/9 on the Silhouette Metric, attesting to its competitiveness against the most widely-used clustering technique. We also note that when measured against our most interpretable alternative, `PCT`, ICOT ties or wins in all cases on the Dunn Index and 7/9 on the Silhouette Metric.

When considering performance by the ranked wins/ties of each method, $K$-`means++` is the best method for the Silhouette Metric and `DBSCAN` is the best method for the Dunn Index. No method dominates ICOT in the win/tie ranking; namely, there is no method that performs better on both the Silhouette Metric and Dunn Index. When looking at the average score across all nine datasets, `Hclust` is the only method to dominate ICOT on both training metrics. However, we note that `Hclust` also has a significantly higher standard deviation on both metrics, indicating a lack of consistency in solution recovery quality.

Our method is weakest when the underlying clusters are non-separable with parallel splits, since ICOT places hard constraints on an observation's cluster membership based on splits in feature values. In these cases, such as with the Hepta dataset, ICOT is unable to recover the true structure. The flexibility offered by alternative methods is advantageous in these cases. Overall, our results demonstrate that despite the highly constrained setting that we impose on the solution structure, we are still able to perform competitively with far less constrained (and less interpretable) methods.

Cluster quality evaluation is highly dependent on the chosen metric; the ground truth assignment is only the "best" method in 3/9 cases with the Silhouette Metric and 6/9 cases with the Dunn Index. ICOT identifies strictly "better" clusters than the ground truth in 6/9 cases for the Silhouette Metric and 3/9 cases for the Dunn Index, as measured by their scores on the respective metrics. This phenomenon raises the broader question of how to assess cluster quality, as recovering known labels in synthetic data does not necessarily translate to meaningful cluster assignments.

### 4.2.1 Sensitivity to training criterion choice

Table 3 shows the ICOT scores on the FCPS datasets as measured by each validation criterion, broken down by training loss function. The values refer to the average score across all nine datasets. As expected, both metrics have their best performance when they are used as the training criterion to optimize for ICOT. The choice to train on the Silhouette Metric results in a 12.4% loss in Dunn Index score as compared to when training on the Dunn Index. Similarly, training originally on the Dunn Index results in a loss of 15.8% in the Silhouette Metric. This quantifies the sensitivity to the choice of training criterion. Both metrics incur a cost in terms of performance loss on other internal validation criteria, with a slightly lower loss on the Dunn Index.

## 5 Experiments based on real-world datasets

In this section, we present results for two real-world examples. We address two important questions often encountered in practice and demonstrate the value of clustering in their analysis; interpretability and performance on internal validation criteria. We illustrate models produced by ICOT, `OCT`, $K$ `-means++`, `Hclust`, `GMM`, `DBSCAN`, `PCT`, and the CUBT algorithm. We also consider the impact of tuning key user-defined parameters on the ICOT model. Section 5.2 outlines a patient similarity case study utilizing data from the well-known Framingham Heart Study (FHS). In these models we consider results across several minimum bucket sizes which offer different levels of granularity in the final output. We also experiment with various $\alpha$ parameters, allowing us to control the weight of numerical vs. categorical features in the distance matrix. Section 5.3 focuses on grouping economic profiles of European countries during the Cold War using only tree-based unsupervised learning techniques.

## 5.1 Experimental setup

We adopted a similar experimental setup to the one described in Sect. 4.1 for the synthetic experiments. In particular, the ICOT experiments use the "fully scaled" version of the algorithm, with a $K$-means warm start and a geometric threshold of 0.99. We ran 100 random restarts of the algorithm in each experiment. The $\alpha$ and minimum bucket parameters are varied as part of the experiments. We ran all of the experiments on normalized data, which is particularly relevant in this setting where features vary greatly in magnitude.

We consider the same six alternative clustering algorithms: `OCT`, $K$-`means++`, `Hclust`, `GMM`, `DBSCAN`, and `PCT`. The latter four methods cannot integrate both categorical and numerical features, so we updated the feature space to one-hot encode the categorical variables as binary features. We used the same fixed algorithm parameters for all methods as outlined in Sect. 4.1. We tuned the $K$ parameter over the range of 2 to 10 clusters for all methods other than `DBSCAN`. We tuned $\epsilon \in [1, 5]$ for `DBSCAN`. All experiments were conducted on two CPUs of type 2 socket Intel E5-2690 v4 2.6 GHz/35M Cache; 16 GB of NUMA enabled memory were used per CPU.

## 5.2 Patient similarity for the Framingham Heart Study

Patient similarity is the concept of identifying groups of individuals with comparable health profiles from their electronic medical records, often with the goal of assessing treatment receptivity and outcomes. The goal is to cluster patients in compact groups without any particular outcome of interest and to study the health progression for those individuals over time. Clustering methods have been particularly popular in this application as they do not require an independent covariate in model creation.

We provide an illustration of our method using data from the Offspring Cohort from the FHS, a large-scale longitudinal clinical study. It started in 1948 with the goal of observing a large population of health adults over time to better understand cardiovascular disease risk factors. Over 80 variables were collected for 5209 people over the course of more than 40 years. The FHS is arguably one of the most influential longitudinal studies in the field of cardiovascular and cerebrovascular research. This data has now been used in more than 2400 studies and is considered one of the top 10 cardiology advances of the twentieth century alongside the electrocardiogram and open-heart surgery (Daniel Levy 2006).

Our dataset consists of 1,200 observations from distinct participants of the Offspring Cohort and 11 covariates (age, gender, presence of diabetes, levels of HDL, BMI status, Blood Pressure (BP) status, blood glucose levels, hematocrit levels, history of myocardial infarction, history of stroke, and current smoking habits) (Daniel Levy 2006; Feinleib et al. 1975). We explore how the ICOT model is impacted as we vary the $\alpha$ parameter and the minimum bucket parameter, $N_C$ (Sects. 5.2.1, 5.2.2). Subsequently, we compare the results

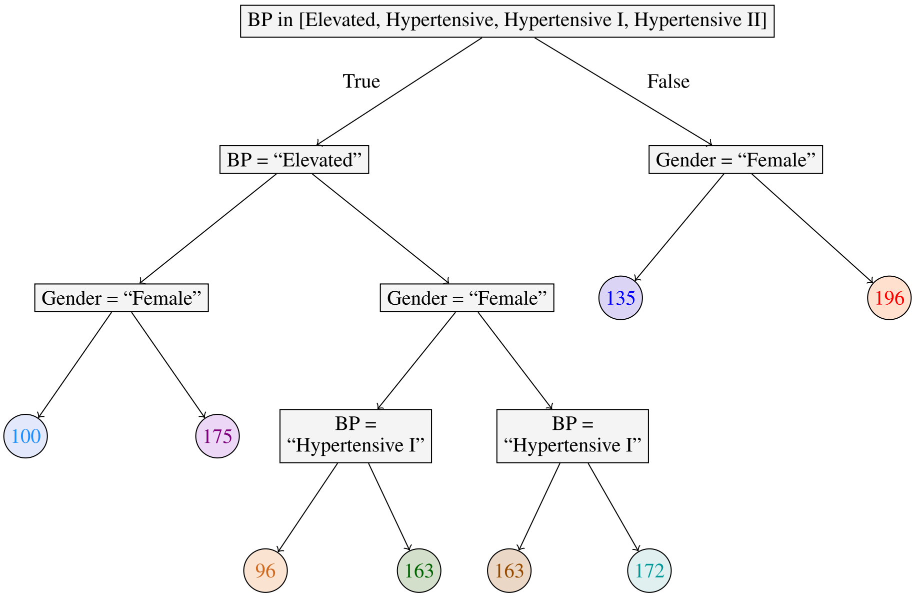

**Fig. 4** ICOT tree for minimum bucket = 50 and $\alpha = 0.3$

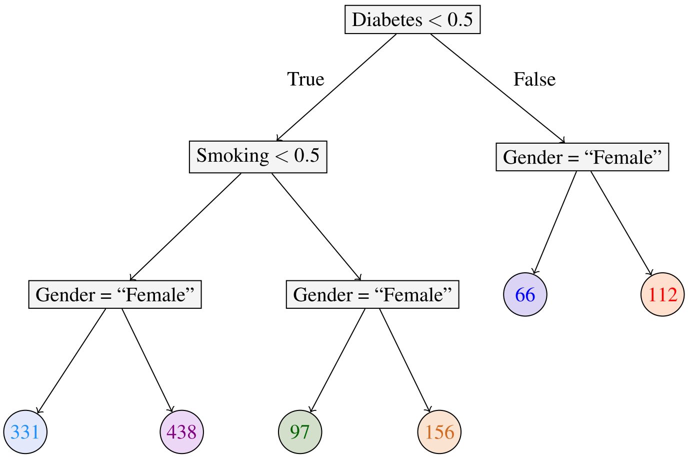

**Fig. 5** ICOT tree for minimum bucket = 50 and $\alpha = 0.6$

of ICOT with other clustering methods in terms of interpretability and quantitative performance on the validation criteria (Sects. 5.2.3–5.2.4).

### 5.2.1 The effect of the $\alpha$ parameter

In this set of experiments, we focus on the impact of the $\alpha$ parameter on the creation of the ICOT model. The FHS dataset contains mixed numerical and categorical attributes and

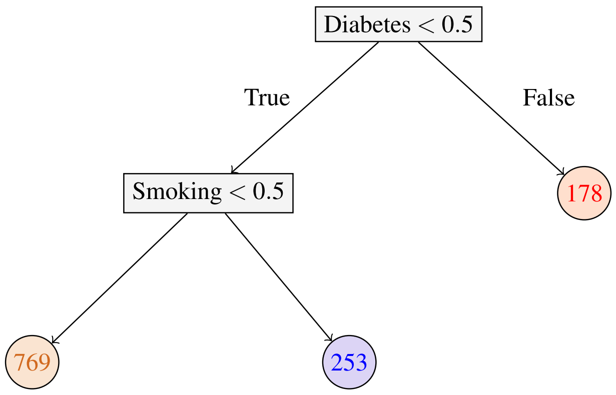

**Fig. 6** ICOT tree for minimum bucket = 50 and $\alpha = 0.9$

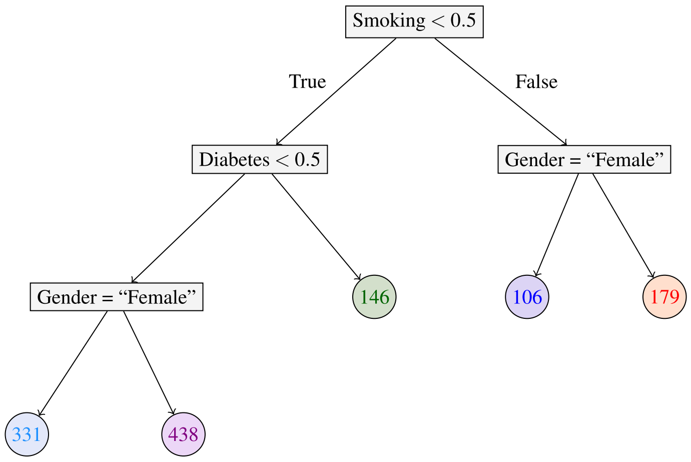

**Fig. 7** ICOT tree for minimum bucket = 100 and $\alpha = 0.6$

thus the determination of this parameter clearly affects the feature selection process during tree construction as well as the final number of clusters. We fix the minimum bucket parameter, $N_C = 50$, requiring at least 50 patients in each cluster to ensure that groups are not skewed by outliers in the data.

Figure 4 shows the model output when $\alpha = 0.3$. The number of observations in each group is indicated by the numbers in the leaves. When the distance matrix places 70% weight on categorical features, the algorithm partitions the feature space based only on those. As a result, only BP status and gender appear as splits in the tree. ICOT identifies eight groups of patients: (1) 100 women with Elevated BP; (2) 175 men with Elevated BP; (3) 96 women with Hypertensive Status I; (4) 163 women with Hypertensive Status II; (5) 163 men with Hypertensive Status I; (6) 172 men with Hypertensive Status II; (7) 135 women with normal BP; (8) 196 men with normal BP.

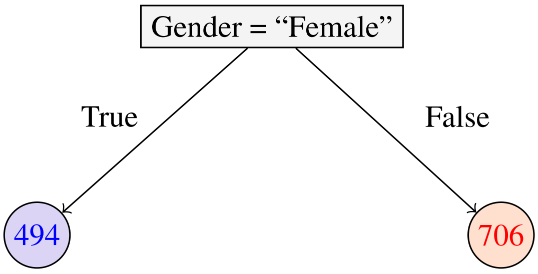

**Fig. 8** ICOT tree for minimum bucket = 200 and $\alpha = 0.6$

When $\alpha = 0.6$ the output model contains variables from both types of data, balancing better the numerical and categorical feature space. Due to the distance metric re-weighting, the new model is now able to incorporate both numerical and categorical features, yielding intuitive groups of participants by cardiovascular risk. Figure 5 illustrates the final tree with five split nodes and six clusters. Given these parameters, ICOT distinguishes between female and male participants in the presence or absence of diabetes. Moreover, it highlights the importance of smoking solely for the diabetic subgroup.

Finally, when $\alpha = 0.9$, ICOT only distinguishes the FHS population based on numeric features such as smoking and diabetes. These results highlight the importance of the algorithm tuning process when leveraging data with mixed features. In the absence of a ground truth, the decision maker is called to select the most appropriate model depending on the application or a potential downstream predictive task. The ability to directly parametrize the distance matrix provides the user with higher flexibility and clarity during the model development process. We discuss the implications of categorical features in the quantitative performance evaluation in Sect. 5.2.4.

### 5.2.2 The effect of the minimum bucket parameter

In these experiments, we set $\alpha = 0.6$ to balance the distance between numerical and categorical features and we vary the minimum number of observations required to form a distinct cluster. Figures 5, 7, and 8 show the models produced by the algorithm for different values of the minimum bucket, $N_C$, when training on the Silhouette Metric. Note that varying this constraint directly affects the end model, changing the structure of the final tree. Even though our empirical results may suggest that there is a monotonic relation between the size of the minimum bucket and the number of clusters identified, this assumption is not necessarily a general rule.

Comparing between Figs. 5 and 7, we see that the output is stable given the minimum bucket restrictions. Both models share the same features in the splits. In the latter model, splits that already had at least 100 members in both leaves (the leftmost two clusters) remained intact and new ones were created in order to closely match the tree with $N_C = 50$. When we increase the minimum sample size to 200 participants, the resulting model only separates the population by gender.

Notice that across all the experiments presented, three variables appear to bear the highest importance in the clustering task: smoking habits, diabetic status, and gender. The results appeared to be stable in the feature selection process, confirming the intuition behind the effect of both the minimum bucket and $\alpha$. ICOT's interpretable structure allowed us to specify the key differentiating characteristics between the participants and contextualize them in the medical setting.

**Table 4** The centroid mean, standard deviation values, and number of observations for all identified clusters from the *K*-`means++` algorithm on the one-hot encoded dataset

| Variable names | Cluster 1 | | Cluster 2 | | Cluster 3 | |
|---|---|---|---|---|---|---|
| | Mean | SD | Mean | SD | Mean | SD |
| Gender: female | 0.367 | 0.485 | 0.376 | 0.485 | 0.487 | 0.5 |
| Gender: male | 0.633 | 0.485 | 0.624 | 0.485 | 0.513 | 0.5 |
| Diabetes | 0.922 | 0.269 | 0.054 | 0.227 | 0.142 | 0.35 |
| Smoking | 0.2 | 0.402 | 0.249 | 0.433 | 0.226 | 0.419 |
| Age | 64 | 7.114 | 61.102 | 9.976 | 65.335 | 9.156 |
| HDL | 39.497 | 12.679 | 46.681 | 14.592 | 46.547 | 14.663 |
| Blood glucose levels | 198.901 | 39.916 | 98.792 | 10.908 | 103.898 | 15.428 |
| Myocardial infarction | 0.333 | 0.519 | 0.337 | 0.632 | 0.239 | 0.518 |
| Hematocrit levels | 44.929 | 3.163 | 43.942 | 3.866 | 43.409 | 3.634 |
| Blood pressure status: elevated | 0.211 | 0.41 | 0.358 | 0.48 | 0 | 0 |
| Blood pressure status: hypertensive crisis | 0.044 | 0.207 | 0 | 0 | 0.066 | 0.249 |
| Blood pressure status: hypertensive status 1 | 0.256 | 0.439 | 0.239 | 0.427 | 0.165 | 0.372 |
| Blood pressure status: hypertensive status 2 | 0.356 | 0.481 | 0 | 0 | 0.769 | 0.422 |
| Blood pressure status: normal | 0.133 | 0.342 | 0.404 | 0.491 | 0 | 0 |
| BMI category: normal | 0.1 | 0.302 | 0.263 | 0.44 | 0.246 | 0.431 |
| BMI category: obese | 0.489 | 0.503 | 0.296 | 0.457 | 0.305 | 0.461 |
| BMI category: overweight | 0.411 | 0.495 | 0.44 | 0.497 | 0.447 | 0.498 |
| BMI category: underweight | 0 | 0 | 0.001 | 0.037 | 0.003 | 0.05 |
| Number of observations | 90 | | 716 | | 395 | |

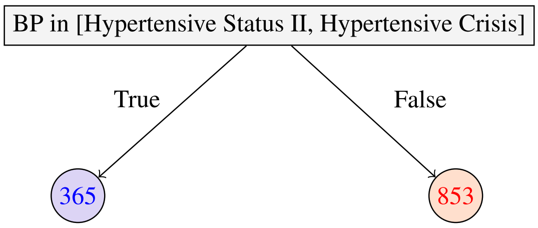

**Fig. 9** Two-step OCT tree, optimized with respect to the Silhouette Metric

### 5.2.3 Results on interpretability

In this section, we compare the interpretability of partitions from different clustering algorithms. For tree based approaches, such as the two step OCT method and `PCT`, we present the final model. For the rest of the algorithms, we outline the centroids of each cluster. Since these methods also do not allow us to directly control the minimum number of observations per cluster, we present the results of each algorithm for the number of clusters that maximizes the Silhouette Metric. Here, we present detailed results for the *K*-`means++`. The reader can find the corresponding information for the other methods in the "Appendix".

Figures 4, 5, 6, 7 and 8 demonstrate different ICOT models when we vary the algorithm's hyperparameters. Note that the trees provide meaningful categorizations that clinicians frequently use and think about in stratifying patient risk. Elevated BP measurements,

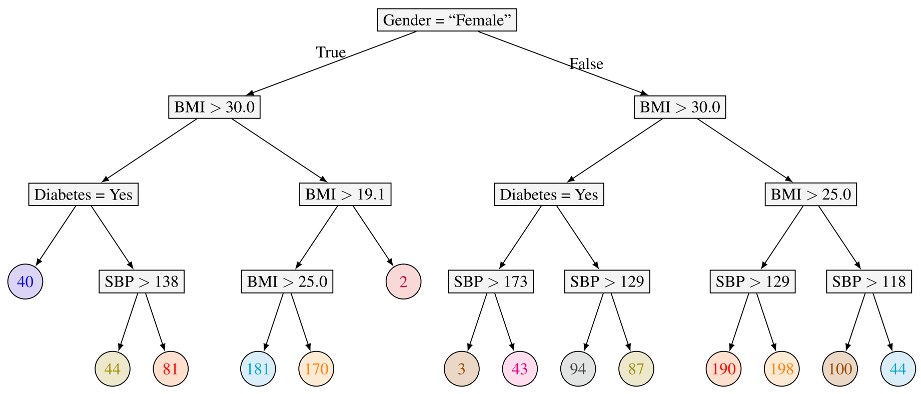

**Fig. 10** PCT Tree for FHS patients

**Table 5** The validation criteria results for ICOT, $K$-`means++`, `Hclust`, `GMM`, `DBSCAN`, `PCT` and the two-step hybrid OCT method when trained on each metric

| Metric | ICOT | OCT | $K$-means++ | Hclust | GMM | DBSCAN | PCT |
|---|---|---|---|---|---|---|---|
| Silhouette metric | 0.296 | 0.131 | 0.264 | 0.270 | 0.224 | 0.511 | 0.249 |
| Dunn index | 0.561 | 0.256 | 0.150 | 0.469 | 0.503 | 0.448 | 0.503 |

gender, smoking are all commonly used categories that determine future health trajectories, such as the risk of cardiovascular events or potential interventions for managing chronic diseases (i.e., blood pressure). The role of these variables has been widely recognized in medical literature (Kannel 1996; Wolf et al. 1988; Offner et al. 1999; Everhart and Wright 1995).

Table 4 shows the covariate values of the cluster centroids created by the $K$-`means++` algorithm. Notice that there is no clear distinction of features that characterize each cluster. For the categorical ones, the centroid value depends on the relative frequency of the classes in the particular covariate and not only on its predominance in the cluster. For example, the fact that the Smoking value for Centroid 1 is equal to 0.2 does not provide deep insights in the smoking habits of the participants in that group. There is a similar proportion of smokers in this cluster compared to Clusters 2 and 3. It is difficult to provide intuitive labels for the groups with clinical implications by only studying Table 4. Furthermore, analyzing the centroid means and standard deviations to gain intuition into the distinctive attributes and spread of each cluster becomes increasingly harder as the number of features increases. Relative ranking of the centroid values could be used in the FHS case, where $p = 18$ (after one-hot encoding) and the number of clusters is small. In a high dimensional dataset, delving into such a table would be practically impossible.

Figure 9 shows the result of the hybrid OCT tree. The model contains just one split, resulting in two clusters providing limited insights regarding the data. In this setting, changing the minimum bucket did not affect the final solution. Figure 10 shows the final PCT tree. This method proposes a deeper tree involving four features: Gender, Diabetes

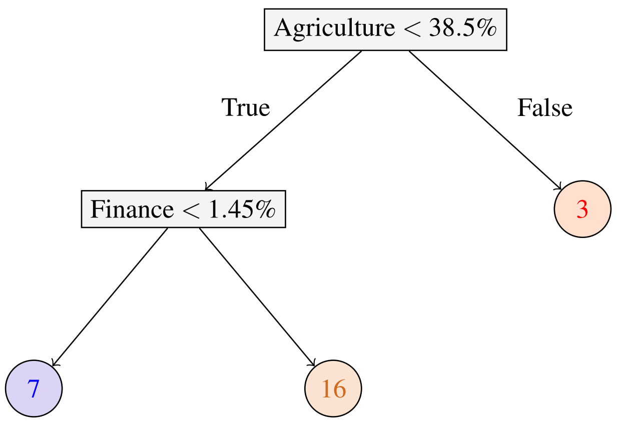

**Fig. 11** Visualization of the ICOT tree for the European Jobs dataset

**Table 6** European country clusters from the ICOT algorithm

| Cluster 1 | Cluster 2 | | Cluster 3 |
|---|---|---|---|
| Bulgaria | Austria | Belgium | Greece |
| Czechoslovakia | Denmark | Finland | Turkey |
| E. Germany | France | Ireland | Yugoslavia |
| Hungary | Italy | Luxembourg | |
| Poland | Netherlands | Norway | |
| Romania | Portugal | Spain | |
| USSR | Sweden | Switzerland | |
| | United Kingdom | W. Germany | |

status, BMI, and Systolic Blood Pressure. It suggests that diabetes status is a differentiator only in obese patients (BMI above 30). It also suggests that the relevant Systolic Blood Pressure threshold is higher for “less healthy” patients, namely those who are diabetic or have higher BMI.

### 5.2.4 Results on quantitative performance

Although interpretability is our primary objective in cluster development, we also want to ensure that our resultant groupings are reasonable from the perspective of the internal validation criteria which provide a quantitative evaluation. Table 5 shows the metric scores obtained for both the Silhouette Metric and the Dunn Index. For each method, we use the Silhouette Metric to cross-validate and find the optimal number of clusters. We then report the score on both metrics for the entire population.

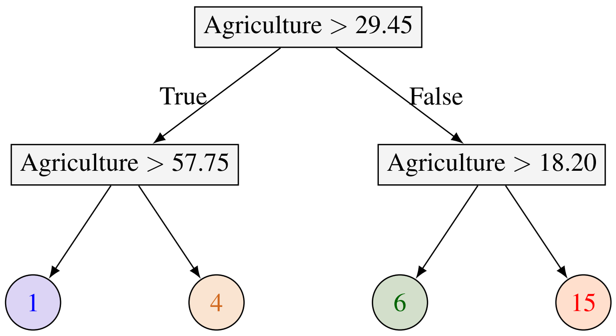

**Fig. 12** CUBT tree with four clusters

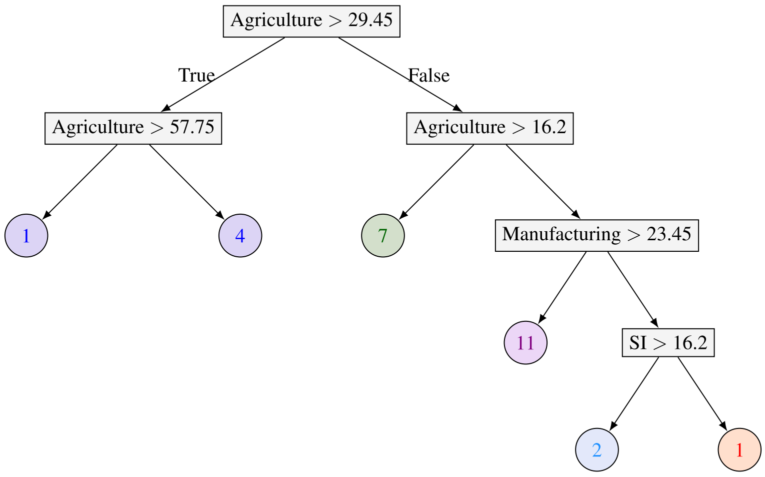

**Fig. 13** CUBT tree with five clusters

**Table 7** European country clusters from the CUBT algorithm, with $K = 4$

| Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | |
|---|---|---|---|---|
| Turkey | Greece | Bulgaria | Austria | Belgium |
| | Poland | Hungary | Czechoslovakia | Denmark |
| | Romania | Ireland | E. Germany | Finland |
| | Yugoslavia | Portugal | France | Italy |
| | | Spain | Luxembourg | Netherlands |
| | | USSR | Norway | Sweden |
| | | | Switzerland | United Kingdom |
| | | | W. Germany | |

ICOT dominates all competing algorithms in the Dunn Index (0.509) and has the second to best performance in the Silhouette Metric (0.296) after `DBSCAN` (0.511). In particular, we note that it has an advantage over `PCT` in both metrics, consistent with our findings in

**Table 8** European country clusters from the CUBT algorithm, with $K = 5$

| Cluster 1 | Cluster 2 | Cluster 3 | | Cluster 4 | Cluster 5 |
|---|---|---|---|---|---|
| Greece | Bulgaria | Austria | Belgium | Netherlands | Denmark |
| Poland | Czechoslovakia | E. Germany | Finland | Norway | |
| Romania | Hungary | France | Italy | | |
| Turkey | Ireland | Luxembourg | Sweden | | |
| Yugoslavia | Portugal | Switzerland | United Kingdom | | |
| | Spain | W. Germany | | | |
| | USSR | | | | |

**Table 9** Comparison of ICOT (trained on the Silhouette Metric) and the CUBT algorithm on the internal validation criterion

| Metric | ICOT | CUBT ($K = 4$) | CUBT ($K = 5$) |
|---|---|---|---|
| Silhouette Metric | 0.344 | 0.140 | 0.044 |
| Dunn Index | 0.346 | 0.262 | 0.259 |

the synthetic experiments. Overall these results suggest that ICOT's advantage in interpretability does not come at the expense of identifying well-separated and compact clusters. The gains over OCT also attest to the value of ICOT's ability to train directly on the cluster quality criterion over simply applying a two-step method where $K$-means clusters are used as class labels for a supervised problem.

## 5.3 Economic profiles of European countries

In this section we consider European countries by their employment statistics during the Cold War to develop groupings of similar economic profiles. We present this example to offer a comparison to the CUBT algorithm (Fraiman et al. 2013) as this is the primary real-world experiment offered in their work.

Our dataset (Krim and Hamza 2015) provides the breakdown of where citizens were employed in 1979 across major industry sectors: agriculture (Agr), mining (Min), manufacturing (Man), power supplies services (PS), construction (Con), service industries (SI), finance (Fin), social and personal services (SPS), and transportation and communication (TC). Thus our feature space includes nine covariates ($p = 9$) observed for 26 distinct European countries ($n = 26$).

### 5.3.1 Results on interpretability: ICOT

We trained a clustering tree using the Silhouette Metric, the default $\alpha$ parameter, and a minimum bucket size of 3 to prevent individual outlier countries from dominating the tree in a single split. The final tree is shown in Fig. 11, and the resulting groupings are shown in Table 6.

ICOT's chosen partition is highly intuitive given the economic and political climate of the Cold War. With the exception of Yugoslavia, all Eastern Bloc countries are placed in Cluster 1 due to their particularly low percentage of workers in the financial sector. This split reflects the broader political setting for those countries that were under a Communist regime. Greece, Turkey and Yugoslavia are grouped together due to their notably high agricultural sector employment. They are also located in the same geographical region and thus their economy similarity is justified. The rest of the countries form Cluster 2, which is composed of all the Western European countries.

### 5.3.2 Results on interpretability: CUBT

Fraiman et al. (2013) provide two alternative clustering partitions using their proposed CUBT algorithm, one with four clusters and the other with five clusters. The resultant tree for $K = 4$ is shown in Fig. 12 with the groupings listed in Table 7. The corresponding results for $K = 5$ are shown in Fig. 13 and Table 8, respectively. Due to inconsistencies between the trees and country groups listed in the paper (Fraiman et al. 2013), we report results based on the tree models presented. It is possible to select a minimum bucket size in the CUBT algorithm, but the authors chose to omit it in these experiments, resulting in isolated clusters with single outlier countries. While this provides insight on its own, we chose to enforce a sufficiently large leaf size to make our results more generalizable and insightful for the full set of European countries.

The tree with four clusters splits only on agriculture sector employment through a series of recursive splits, providing less insight into the differentiating characteristics of the countries. The tree with five clusters splits on high agriculture employment first to separate out the first two clusters, but then further differentiates the low agriculture countries on both manufacturing and service industry employment. The bulk of the countries fall into the third cluster, which is characterized by a manufacturing-heavy workforce. Note that CUBT allows for cluster re-joining in the algorithm, which results in multiple leaves being assigned to the same cluster (indicated by a single color). Overall, while the CUBT algorithm provides high interpretability as with ICOT, a qualitative analysis of the resulting clusters suggests that there is a slight loss in meaningful cluster separation.

### 5.3.3 Results on the validation criteria

The quantitative performance of these models on our two key internal validation criteria are shown in Table 9. ICOT obtains significantly better clusters as quantified by both the Dunn Index and Silhouette Metric. We note that ICOT has an advantage in the Silhouette Metric due to the fact that it was trained to optimize this criterion, whereas the CUBT results were trained via a different method. However, the Dunn Index provides a neutral evaluation criterion and shows a preference towards ICOT's results as well.

## 6 Scaling experiments

In this section, we present results regarding the effect of scaling techniques on ICOT with respect to both the quality of the final solutions as well as the degree to which the algorithm is able to scale. In Sect. 6.1, we discuss the impact of algorithm heuristics, such as the $K$-means warm start and the geometric threshold, using the FCPS suite. We use real-world data from Hubway for testing the scalability and quantitative performance of bootstrapping in Sect. 6.2.

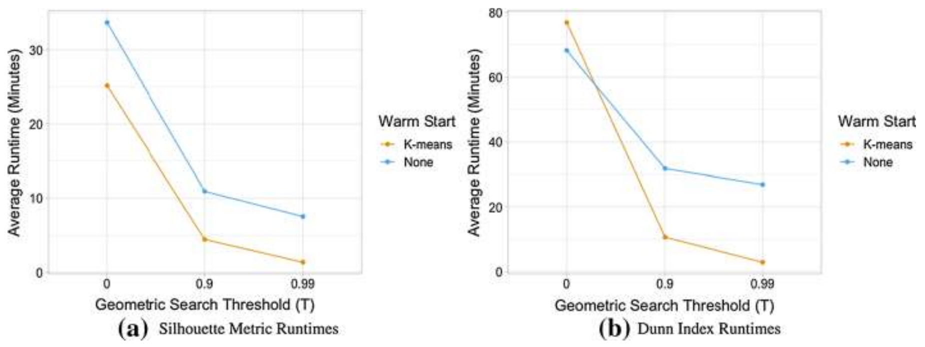

**Fig. 14** Average runtimes across FCPS datasets with varied scaling parameters for the geometric search threshold (T) and choice to use a warm start

**Table 10** Comparison of cluster quality scores with the original versus fully scaled ICOT versions

| Dataset | Silhouette metric | | | Dunn index | | |
|---|---|---|---|---|---|---|
| | Baseline | Fully scaled | % Change | Baseline | Fully scaled | % Change |
| Atom | 0.521 | 0.503 | −3.45% | 0.137 | 0.137 | 0.00% |
| Chainlink | 0.391 | 0.396 | 1.28% | 0.032 | 0.028 | −12.62% |
| Hepta | 0.455 | 0.453 | −0.44% | 0.357 | 0.357 | 0.00% |
| Lsun | 0.567 | 0.549 | −3.17% | 0.117 | 0.077 | −34.10% |
| Target | 0.629 | 0.629 | 0.00% | 0.362 | 0.550 | 51.93% |
| Tetra | 0.504 | 0.504 | 0.00% | 0.200 | 0.200 | 0.00% |
| TwoDiamonds | 0.486 | 0.486 | 0.00% | 0.044 | 0.044 | 0.00% |
| WingNut | 0.406 | 0.422 | 3.94% | 0.063 | 0.063 | 0.00% |
| Average Score | 0.495 | 0.493 | -0.23% | 0.164 | 0.182 | 0.65% |

## 6.1 Scaling via algorithm heuristics

In this section, we evaluate the impact of implementing the scaling methods described in Sect. 3.3. We first Sect. 6.1.2 examines the runtime reductions that we obtain as we vary the scaling parameters. We then consider how the heuristics affect solution recovery in Sect. 6.1.3.

### 6.1.1 Experimental setup

We evaluated the impact of our scaling methods on algorithm speed through a comparison of the average runtime across eight datasets in the FCPS suite with various parameters. The ninth dataset (EngyTime) was omitted as the experiment size was intractable on the unscaled method. We ran experiments over restricted geometric search thresholds of $T = 0$ (scan all thresholds), $T = 0.9$ and $T = 0.99$. We also repeated the experiments with and without the $K$-means warm start. The parameter pair ($T = 0$, no warm start) represents the original "baseline" method, and the pair ($T = 0.99$, $K$-means warm start) represents the fully scaled method. We ran each dataset and parameter combination across five seeds and present the averaged results.

All experiments were conducted on two CPUs of type 2 socket Intel E5-2690 v4 2.6 GHz/35M Cache; 16 GB of NUMA enabled memory were used per CPU.

### 6.1.2 Scaling runtimes

The runtimes for the Silhouette Metric and Dunn Index are shown in Fig. 14. The geometric search alone reduces the runtime by 77.6% (60.6%) at the $T = 0.99$ threshold for the Silhouette Metric (Dunn Index). When combining the geometric search ($T = .99$) with the $K$-means warm start, our fully scaled method offers a 96.0% (95.7%) reduction in algorithm runtime for Silhouette (Dunn). We observe that the baseline method actually has a slight runtime advantage over the $K$-means warm start when there is no restriction on the search space ($T = 0$). The apparent shorter runtime with the baseline method at $T = 0$ can be explained by the possibility of getting caught in a locally optimal solution with a naive start, which can lead the algorithm to terminate faster.

Due to the speedups from these two scaling techniques, ICOT is able to scale to handle datasets with a number of observations ($N$) in the thousands and the number of covariates ($p$) in the hundreds. The scaled algorithm solves within several hours for problems of this magnitude.

### 6.1.3 High quality solution recovery

The scores of the baseline model and our fully scaled version are shown in Table 10. The scaled method yields an average loss of $-0.28\%$ over the baseline when trained on the Silhouette Metric, and gives an average improvement of 0.64% with the Dunn Index. Of the eight datasets considered using the Silhouette Metric (Dunn Index), three (five) have identical cluster recovery in both the original and fully scaled experiments; three (two) have a slight loss when using scaling heuristics, and two (one) actually improve with the scaling methods. These results suggest that the scaled ICOT algorithm still yields high quality results.

The differences in the score between the baseline and scaled versions are largely attributable to the warm start rather than the choice of geometric threshold. The score improves in the scaled version when the baseline algorithm was caught in a local optimum, but the $K$-means warm start enabled it to avoid this. This score improvement offered by the $K$-means warm starts further supports the use of this heuristic beyond runtime improvements.

## 6.2 Scaling via bootstrapping

In Sect. 6.2.1, we introduce the Hubway dataset, a real-world collection of user ride data from a Boston-based bike sharing program. Section 6.2.2 outlines the experimental setup, providing details on the parameters of the method. Sections 6.2.3 and 6.2.4 explore the effect of the bootstrapping methodology on the quality of the final solution and the algorithm runtime respectively.

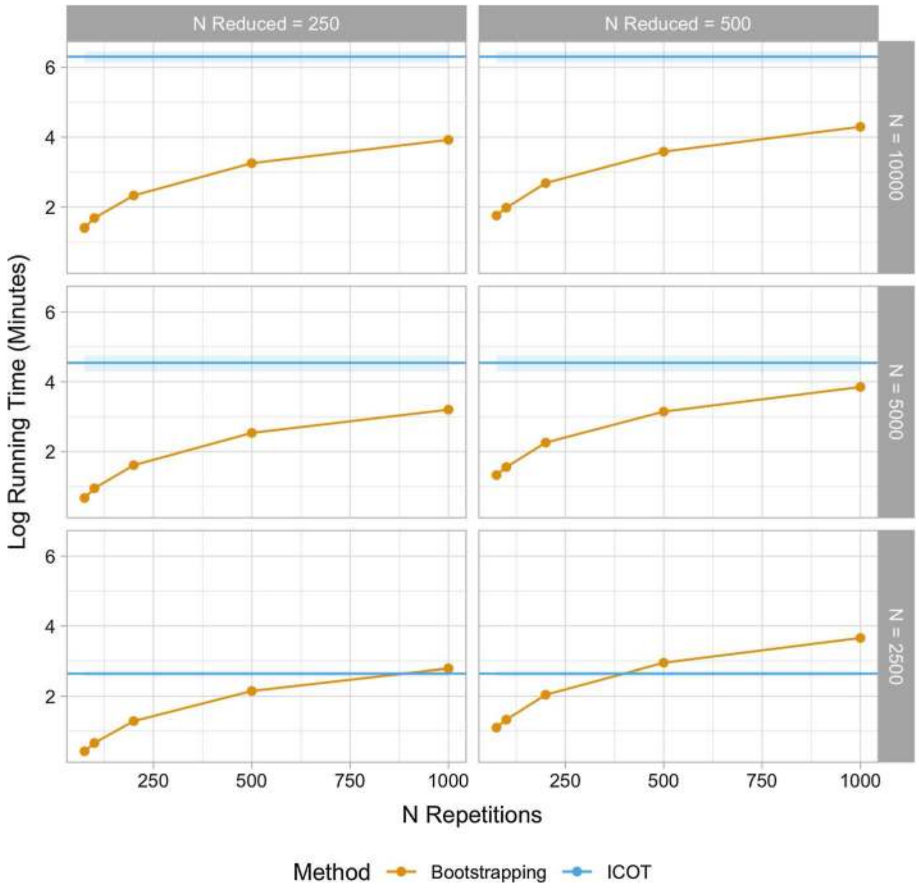

**Fig. 15** Results regarding the impact of bootstrapping on the runtime (Log of Minutes) as the number of repetitions ($N_{\mathrm{rep}}$), sub-sample size ($N_r$), and sample size ($N$) change. Both methods were trained on the Silhouette Metric. The error bars express the standard deviation of the metric

### 6.2.1 The Hubway dataset

In this setting, our goal is to identify similar groups of registered users of the Hubway bike-sharing program (Bertsimas et al. 2016). This Boston-based company allows citizens to rent bicycles from any of their 140 stations and ride to any other station in the city. The platform has emerged as a popular form of transportation for daily commuters and leisure riders alike. Our dataset includes 194,301 observations from Hubway trips taken from June 2012 through September 2012. The dataset contains nine mixed numerical and categorical attributes, including the duration of the trip, the age and the gender of the rider, the time period of the ride and whether it took place during the week or the weekend.

This experiment illustrates an application of clustering for market segmentation. This is a strategy that divides a broad target market into smaller groups of similar customers. It can then be used to tailor marketing strategies to individual groups through means such as promotions or differentiated pricing. Unsupervised learning is often employed for this task since it naturally identifies similar groups within a given dataset.

### 6.2.2 Experimental setup

In these experiments, we aim to quantify the benefit of using bootstrapping as a wrapper function over the ICOT algorithm. We explore the effect of three key parameters that might affect both the quality and runtime of the solutions.

1. Sample Size ($N$): The number of observations included in the training set. Since the Hubway dataset contains 194,301 data points, we sub-sample randomly without replacement to create a sample of size $N$. We follow the same process to create a different testing set that is used for the evaluation of the validation criterion. We restrict $N$ to numbers that can be efficiently solved by ICOT, $N \in [2500, 5000, 10000]$, to allow us to compare to the algorithm's solutions on the full input data.
2. Size of reduced data ($N_r$): The number of observations included in each iteration of the bootstrap algorithm. Each sub-sample is randomly created from the training set without replacement, but the iteration samples are constructed independently. Thus, different iterations can contain the same observation. We let $N_{\text{rep}} \in [250, 500]$.
3. Number of repetitions ($N_{\text{rep}}$): The number of iterations of the bootstrapping method. We test the quality and runtime of the final model by letting $N_{\text{rep}} \in [25, 50, 75, 100, 200, 500, 1000]$.

All results presented for ICOT use a version of the algorithm that includes the $K$-means warm start and a geometric threshold of 0.99. The minimum bucket size is set to one and the maximum depth of the tree to depth four. We assigned to the $\alpha$ parameter its default value. Similarly to the FCPS experiments, we ran 100 random restarts of the algorithm in each round. Results summarize the outcomes of five randomized repetitions of each experiment.

In the following experiments, all results are averaged over 50 experiments per algorithm and parameter combination. All experiments were conducted on two CPUs of type 2 socket Intel E5-2690 v4 2.6 GHz/35M Cache; 30GB of NUMA enabled memory were used per CPU.

### 6.2.3 Scaling performance

The purpose of introducing bootstrapping into the ICOT framework is to extend its application to problems of larger size that the fully scaled version was not able to efficiently manage. Bootstrapping provides a lot of flexibility to the user and thus can be easily adapted to the speed requirements of a specific case study. In this section, our aim is to demonstrate how choices regarding the parameters affect the overall running time and compare the outcomes with and without bootstrapping. Figure 15 provides an overview of the results when the algorithm was trained on the Silhouette Metric. The corresponding results for the Dunn Index are summarized in the "Appendix". We report the *log*(time) to render the $y$-scale more comprehensible to the reader, especially for higher instances of $N$. The average runtime scales linearly with respect to $N_{\text{rep}}$ and exponentially to $N_r$. As we include additional repetitions, the method sequentially runs more iterations of the same "reduced" experiment. However, as we increase the $N_r$, the runtime scales at the same rate as the original ICOT method. When $N_{\text{rep}} > 500$, bootstrapping starts improving on the original algorithm only for instances of $N > 2500$. Nevertheless, in cases of larger sample size ($N = 10,000$), bootstrapping can achieve the same solution quality ($N_r = 250, N_{\text{rep}} = 500$)

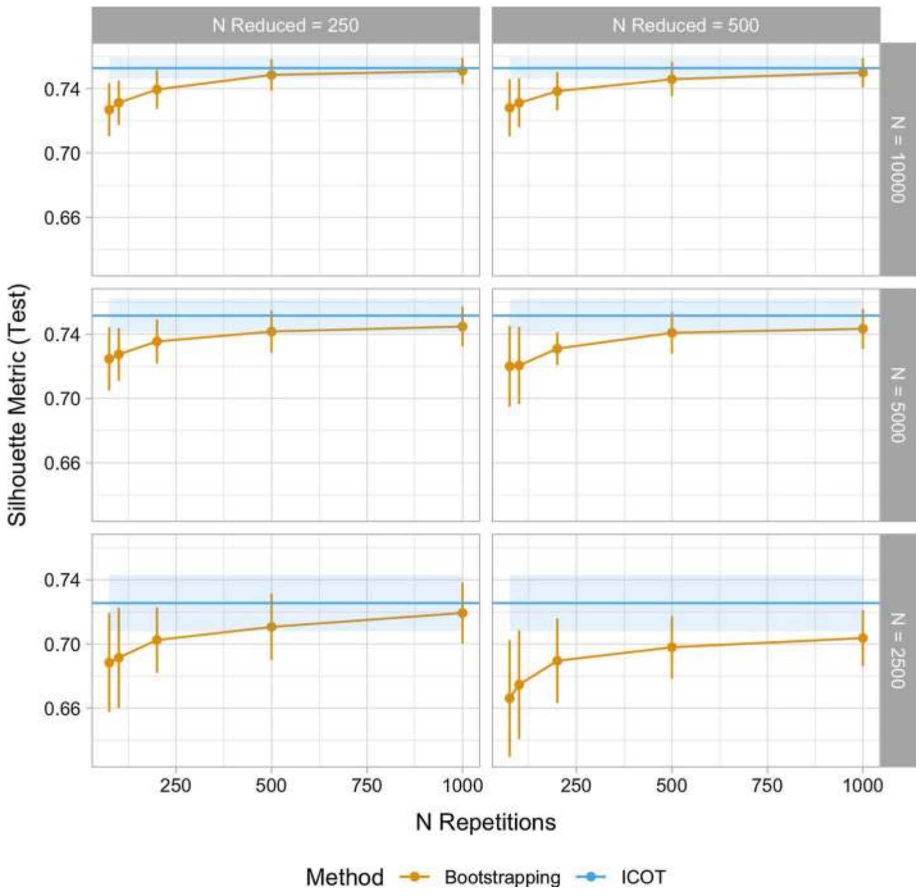

**Fig. 16** Results regarding the impact of bootstrapping on the Silhouette Metric as the number of repetitions ($N_{\text{rep}}$), sub-sample size ($N_r$), and sample size ($N$) change. The error bars express the standard deviation of the metric

in 27.65 minutes instead of 554.693. When $N = 5,000$, the discrepancy is not as high but still considerable, 13.095 and 96.529 minutes respectively.

These results indicate the value of adding bootstrapping into the ICOT framework, as it solves in reasonable time problems of much larger size that otherwise would have been out of the algorithm's scope.

### 6.2.4 High quality solution recovery

The bootstrapping approach constructs trees on a sub-group of the overall population and thus does not access the full input data. We sought to ensure that the speed-up in runtime would not come at a high toll with respect to solution quality. Thus, we performed a direct comparison of the two methods over the validation criteria for different ranges of the parameters described above. Figure 16 provides a results summary for the Silhouette Metric. A similar graph for the Dunn Index can be found in the "Appendix". The shaded region around ICOT indicates the standard deviation of the metric. Similarly, the error bars illustrate the same measure for each combination of the tuning parameters. As expected, larger sample sizes are positively correlated with the validation score. The graphs show that increasing the number of repetitions can significantly improve the quality of the solution. We notice that for $N_{\text{rep}} > 500$, bootstrapping can achieve equivalent performance to ICOT, with minor losses in some cases. The effect of the $N_r$ parameter is less evident, though, as the results indicate minor discrepancies between $N_r = 250$ and $N_r = 500$. In conclusion, these experiments provide evidence that bootstrapping does not result in a high toll on the quality of suggested feature partitions.

## 7 Discussion

ICOT builds trees that provide explicit separations of the data on the original feature set, creating interpretable models with real-world applicability to a wide range of settings. From healthcare to revenue management to macroeconomics, our algorithm can significantly benefit practitioners that may find value in unsupervised learning techniques in their work.

Our empirical results on the FCPS dataset offer insight into ICOT's performance against existing methods, including traditional approaches such as $K$-means, density-based, and hierarchical algorithms. We also report results with respect to other interpretable methods, including the Predictive Clustering Trees framework and the hybrid two-step supervised approach. Overall, our proposed method is superior to the majority of the algorithms for both validation criteria. Specifically, in Sect. 4, we show that when assessing clusters with the Silhouette Metric, ICOT is the second best method after $K$-`means++` while on the Dunn Index ICOT is only outperformed by `DBSCAN`. Essentially, our experiments demonstrate that our newly proposed framework is able to achieve comparable performance to the state-of-the-art clustering algorithms while enabling the explicit characterization of cluster membership. We thus accept a slight decrease in the validation criteria for the gain in interpretability, which is critical in many settings.

We also observe significant improvements in ICOT over other interpretable approaches. The relatively poor performance of the two-step OCT approach validates the utility of a method that simultaneously builds clusters and identifies a tree-based structure rather than simply employing existing tree-based methods on clustered data *a posteriori*. Additionally, ICOT offers a considerable advantage over PCT and CUBT, suggesting that our algorithmic approach improves upon on existing interpretable clustering work and offers a novel contribution to the space.

Most clustering methods, including ICOT, identified data partitions with higher cluster quality scores than the true FCPS data labels, highlighting the subjectivity of what constitutes good clusters. We leave the choice of cluster quality metric to the user, since both criterion have their respective merits and perform well in different data contexts. In general, the Dunn Index excels on well-separated datasets but is not robust to outliers. In contrast, the Silhouette Metric is often better at accounting for mixed densities and identifying meaningful separation in less structured data settings.

The additional scaling experiments on the FCPS dataset demonstrate substantial runtime reductions offered by both the restricted geometric search space and $K$-means warm start. Overall these empirical results suggest that the scaling methods are successful at significantly decreasing runtime while maintaining high-quality cluster identification. The geometric search heuristic is particularly useful for problems with a high number of observations as it lowers the computational load per node evaluation by a factor of $T$. We note that despite the efficiency gains offered by our scaling methods, our current implementation of ICOT does not scale beyond 1000s of observations and 100s of covariates. However, using the Hubway dataset we were able to demonstrate that the ICOT algorithm coupled with bootstrapping is able to scale to even hundreds of thousands of observations at a reasonable time without a considerable toll on the solution quality. This functionality broadens the method's applicability to even high-dimensional settings; for example, bootstrapping might be particularly useful when clustering a large company's customer transaction records ($n$ in the millions). This is a case where we would recommend the subsampling approach. A similar technique could be applied for cases where the number of features is very high ($p$ in the 10000s), such as when using genomic profiles for patients. Additionally, variables could be preprocessed to restrict to the most significant subset, either using traditional statistical tests or the variable importance ranking provided in the $K$-means algorithm output.

Therefore, we believe that ICOT is the best performing alternative for interpretable clustering although computationally more intensive. PCTs are more efficient but in many cases lead to lower quality solutions. Our method has an edge over K-means++ and DBSCAN due to the transparency it offers, although these alternatives sometimes show a slight edge on the Silhouette Metric and the Dunn Index. ICOT is most appropriate in applications where the user values both interpretation of the cluster labels and high performance on clustering metrics, and the efficiency of the algorithm is not a bottleneck. These conditions are generally true in the exploratory analysis contexts where clustering is most often applied.

Our work's handling of numerical and categorical features offers a contribution beyond the realm of clustering. The issue of mixed-type attributes is considered among specialists as one of the most important challenges in machine learning (Piatetsky-Shapiro et al. 2006; Yang and Wu 2006). The overwhelming majority of state-of-the-art clustering algorithms are restricted to numerical objects, like vectors or metric objects, which does not correspond to datasets usually found in practice. This problem extends more broadly to algorithms that rely on distance computations, such as $k$-Nearest Neighbors. In contrast, our solution gives a comprehensive answer to this problem by introducing a novel distance metric for the algorithm.

We note that the algorithm's single-variable splits are unable to represent all possible cluster shapes and could potentially cut through clusters. This structure allows us to maintain the direct interpretation of a tree leaf representing a single cluster. In many applications, a simple interpretation of the tree partition is highly valued, which was a key motivation behind this method's development. In order to capture more complex structures, one could consider the possibility of "rejoining" leaves, namely allowing multiple leaves to be considered as a single cluster. Rejoining can occur between two adjacent leaves coming from a single parent node through the local search's consideration of split deletions. However, we do not consider the possibility of joining other leaves. While ICOT does not natively support this, it could easily be incorporated as a post-processing step. After obtaining the final ICOT tree, one can consider the effect of merging different node combinations on the chosen metric.

We finally observe that despite the tree structure of our algorithm output, our model does not obey a hierarchical structure. Namely, truncating the tree to a lower depth does not necessarily represent the optimal clustering solution at this depth. Our coordinate-descent algorithm allows for nodes to be re-optimized with knowledge of deeper nodes. In contrast, a hierarchical interpretation only holds in cases where the tree grows greedily since the shallow truncated tree cannot be affected by deeper levels.

The application of ICOT to real-world datasets reveals the significant benefit on both interpretability and performance in the unsupervised learning field. The combination of the OCT mechanism, the employment of established internal validation criteria as well as the systematic handling of mixed numerical and categorical attributes allow ICOT to provide complete partitions of the feature space with actionable insights to practitioners. Moreover, the flexibility of the method to user specific constraints with respect to the minimum bucket size, the maximum depth of the tree and the $\alpha$ parameter render the algorithm particularly amenable to a wide range of applications from various fields.

## 8 Conclusion

In this paper, we have introduced a new methodology of cluster construction that addresses the issue of cluster interpretability. We propose a novel unsupervised learning tree-based algorithm that yields high-quality solutions via an optimization approach. Through computational experiments with benchmark and real-world datasets, we show that ICOT offers significant gains in interpretability over state-of-the-art clustering methods while achieving comparable or even better performance as measured by well-established internal validation criteria. This makes ICOT an ideal tool for exploratory data analysis as it reveals natural separations of the data with intuitive reasoning.

**Acknowledgements** We would like to thank the anonymous reviewers from the Machine Learning For Healthcare Workshop, NeurIPS 2018 for their insightful comments on an earlier version of this paper. We would also like to thank the reviewers from Machine Learning, whose comments significantly improved the paper. We would also like to acknowledge the contribution of Charlene Ong, MD, MPHS, who provided us with medical interpretations of our results on the Framingham Heart Study dataset.

## Appendix

### Synthetic experiments

See Tables 11 and 12.

**Table 11** Comparison of runtimes (in minutes) across the FCPS datasets, when trained and evaluated on the Silhouette Metric

| Data | (N,P) | ICOT | OCT | *K*-means++ | Hclust | GMM | DBSCAN | PCT |
|---|---|---|---|---|---|---|---|---|
| Hepta | (212,3) | 0.196 | 0.001 | 0.041 | 0.009 | 0.003 | 0.022 | 0.061 |
| Lsun | (400,2) | 0.358 | 0.002 | 0.049 | 0.015 | 0.003 | 0.034 | 0.079 |
| Tetra | (400,3) | 0.399 | 0.002 | 0.053 | 0.019 | 0.004 | 0.035 | 0.064 |
| Target | (770,2) | 1.240 | 0.002 | 0.068 | 0.031 | 0.004 | 0.053 | 0.081 |
| TwoDiamonds | (800,2) | 0.394 | 0.002 | 0.072 | 0.036 | 0.004 | 0.052 | 0.079 |
| Atom | (800,2) | 2.200 | 0.006 | 0.073 | 0.036 | 0.004 | 0.058 | 0.097 |
| Chainlink | (1000,2) | 4.580 | 0.003 | 0.084 | 0.045 | 0.004 | 0.062 | 0.096 |
| WingNut | (1070,2) | 1.430 | 0.003 | 0.090 | 0.047 | 0.004 | 0.065 | 0.096 |
| EngyTime | (4096,2) | 64.700 | 0.021 | 0.326 | 0.252 | 0.016 | 0.240 | 0.448 |
| Average Runtime | | 8.389 | 0.005 | 0.095 | 0.054 | 0.005 | 0.069 | 0.123 |
| Std. Dev. Runtime | | 21.162 | 0.006 | 0.088 | 0.075 | 0.004 | 0.066 | 0.123 |

The datasets are ordered by ascending problem size

**Table 12** Comparison of runtimes (in minutes) across the FCPS datasets, when trained and evaluated on the Dunn Index

| Data | (N,P) | ICOT | OCT | *K*-means++ | Hclust | GMM | DBSCAN | PCT |
|---|---|---|---|---|---|---|---|---|
| Hepta | (212,3) | 0.264 | 0.001 | 0.043 | 0.009 | 0.003 | 0.026 | 0.062 |
| Lsun | (400,2) | 1.840 | 0.002 | 0.050 | 0.016 | 0.003 | 0.034 | 0.063 |
| Tetra | (400,3) | 0.644 | 0.002 | 0.049 | 0.018 | 0.003 | 0.035 | 0.065 |
| Target | (770,2) | 2.030 | 0.002 | 0.069 | 0.031 | 0.004 | 0.056 | 0.085 |
| TwoDiamonds | (800,2) | 0.595 | 0.003 | 0.069 | 0.039 | 0.004 | 0.051 | 0.084 |
| Atom | (800,2) | 4.410 | 0.006 | 0.078 | 0.039 | 0.004 | 0.079 | 0.096 |
| Chainlink | (1000,2) | 8.980 | 0.003 | 0.087 | 0.047 | 0.005 | 0.069 | 0.104 |
| WingNut | (1070,2) | 4.430 | 0.003 | 0.086 | 0.046 | 0.005 | 0.062 | 0.105 |
| EngyTime | (4096,2) | 1234.0 | 0.021 | 0.351 | 0.271 | 0.030 | 0.268 | 1.125 |
| Average Runtime | | 139.688 | 0.005 | 0.098 | 0.057 | 0.007 | 0.076 | 0.199 |
| Std. Dev. Runtime | | 410.376 | 0.006 | 0.096 | 0.081 | 0.009 | 0.074 | 0.348 |

The datasets are ordered by ascending problem size

## Real-world experiments: results on interpretability

See Tables 13, 14, 15, 16, 17 and 18.

**Table 13** The centroid mean values and number of observations for all identified clusters from the `Hclust` algorithm on the one-hot encoded dataset

| Variable | Cl. 1 | Cl. 2 | Cl. 3 | Cl. 4 | Cl. 5 | Cl. 6 | Cl. 7 | Cl. 8 | Cl. 9 | Cl. 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Gender: female | 0 | 0 | 0 | 0 | 1 | 1 | 0.992 | 1 | 0 | 1 |
| Gender: male | 1 | 1 | 1 | 1 | 0 | 0 | 0.008 | 0 | 1 | 0 |
| Diabetes | 0.156 | 0.189 | 0.065 | 0.232 | 0.272 | 0.141 | 0.025 | 0.087 | 0.556 | 0 |
| Smoking | 0.277 | 0.367 | 0.21 | 0.201 | 0.256 | 0.17 | 0.252 | 0.183 | 0.222 | 1 |
| Age | 62 | 64.156 | 61.419 | 60.567 | 63.76 | 66.437 | 63.824 | 61.478 | 63.889 | 69 |
| HDL | 40.772 | 45.04 | 41.295 | 38.321 | 48.205 | 52.622 | 56.792 | 57.376 | 37.778 | 35 |
| Blood glucose levels | 112.026 | 108.047 | 103.009 | 115.311 | 119.034 | 105 | 95.172 | 100.594 | 118 | 88 |
| Myocardial infarction | 0.328 | 0.478 | 0.511 | 0.348 | 0.152 | 0.193 | 0.151 | 0.174 | 0.333 | 0 |
| Hematocrit levels | 45.953 | 46.133 | 45.917 | 45.796 | 41.623 | 40.645 | 40.359 | 40.7 | 47.211 | 43.8 |
| Blood pressure status: elevated | 0.344 | 0.4 | 0 | 0.311 | 0.2 | 0.274 | 0.319 | 0 | 0 | 0 |
| Blood pressure status: hypertensive crisis | 0 | 0 | 0 | 0 | 0.056 | 0.067 | 0.042 | 0 | 1 | 0 |
| Blood pressure status: hypertensive status 1 | 0.297 | 0.289 | 0 | 0.372 | 0.32 | 0.23 | 0.202 | 0 | 0 | 1 |
| Blood pressure status: hypertensive status 2 | 0.359 | 0.311 | 0 | 0.317 | 0.424 | 0.43 | 0.437 | 0 | 0 | 0 |
| Blood pressure status: normal | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| BMI category: normal | 0 | 1 | 0.231 | 0 | 0 | 0 | 1 | 0.365 | 0 | 0 |
| BMI category: obese | 0 | 0 | 0.242 | 1 | 1 | 0 | 0 | 0.322 | 0.556 | 0 |
| BMI category: overweight | 1 | 0 | 0.527 | 0 | 0 | 1 | 0 | 0.304 | 0.444 | 0 |
| BMI category: underweight | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.009 | 0 | 1 |
| Number of observations | 256 | 90 | 186 | 164 | 125 | 135 | 119 | 115 | 9 | 1 |

**Table 14** The centroid standard deviation values and number of observations for all identified clusters from the `Hclust` algorithm on the one-hot encoded dataset

| Variable | Cl. 1 | Cl. 2 | Cl. 3 | Cl. 4 | Cl. 5 | Cl. 6 | Cl. 7 | Cl. 8 | Cl. 9 | Cl. 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Gender: female | 0 | 0 | 0 | 0 | 0 | 0 | 0.092 | 0 | 0 | NA |
| Gender: male | 0 | 0 | 0 | 0 | 0 | 0 | 0.092 | 0 | 0 | NA |
| Diabetes | 0.364 | 0.394 | 0.246 | 0.423 | 0.447 | 0.349 | 0.157 | 0.283 | 0.527 | NA |
| Smoking | 0.449 | 0.485 | 0.408 | 0.402 | 0.438 | 0.377 | 0.436 | 0.388 | 0.441 | NA |
| Age | 9.704 | 9.405 | 9.356 | 8.922 | 9.894 | 9.684 | 9.591 | 10.446 | 7.897 | NA |
| HDL | 11.032 | 12.776 | 10.821 | 10.053 | 14.228 | 14.796 | 15.869 | 15.946 | 14.158 | NA |
| Blood glucose levels | 35.778 | 30.445 | 20.232 | 33.916 | 41.933 | 21.675 | 15.165 | 26.716 | 32.852 | NA |
| Myocardial infarction | 0.634 | 0.768 | 0.73 | 0.538 | 0.422 | 0.465 | 0.404 | 0.464 | 0.5 | NA |
| Hematocrit levels | 2.812 | 2.902 | 2.667 | 2.582 | 2.945 | 2.91 | 2.77 | 2.839 | 2.078 | NA |
| Blood pressure status: elevated | 0.476 | 0.493 | 0 | 0.464 | 0.402 | 0.448 | 0.468 | 0 | 0 | NA |
| Blood pressure status: hypertensive crisis | 0 | 0 | 0 | 0 | 0.231 | 0.25 | 0.201 | 0 | 0 | NA |
| Blood pressure status: hypertensive status 1 | 0.458 | 0.456 | 0 | 0.485 | 0.468 | 0.422 | 0.403 | 0 | 0 | NA |
| Blood pressure status: hypertensive status 2 | 0.481 | 0.466 | 0 | 0.467 | 0.496 | 0.497 | 0.498 | 0 | 0 | NA |
| Blood pressure status: normal | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | NA |
| BMI category: normal | 0 | 0 | 0.423 | 0 | 0 | 0 | 0 | 0.484 | 0 | NA |
| BMI category: obese | 0 | 0 | 0.429 | 0 | 0 | 0 | 0 | 0.469 | 0.527 | NA |
| BMI category: overweight | 0 | 0 | 0.501 | 0 | 0 | 0 | 0 | 0.462 | 0.527 | NA |
| BMI category: underweight | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.093 | 0 | NA |
| Number of observations | 256 | 90 | 186 | 164 | 125 | 135 | 119 | 115 | 9 | 1 |

**Table 15** The centroid mean values and number of observations for all identified clusters from the GMM algorithm on the one-hot encoded dataset

| Variable | Cl. 1 | Cl. 2 | Cl. 3 | Cl. 4 | Cl. 5 | Cl. 6 | Cl. 7 | Cl. 8 | Cl. 9 | Cl. 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Gender: female | 0.433 | 0.37 | 0.478 | 0.667 | 0 | 0.396 | 0.622 | 0 | 1 | 0.707 |
| Gender: male | 0.567 | 0.63 | 0.522 | 0.333 | 1 | 0.604 | 0.378 | 1 | 0 | 0.293 |
| Diabetes | 1 | 0 | 0.157 | 0.267 | 0.126 | 0.218 | 1 | 0.059 | 0.027 | 0 |
| Smoking | 0.2 | 0.289 | 0.235 | 0.233 | 0.257 | 0.238 | 0.178 | 0.211 | 0.219 | 0.252 |
| Age | 67.9 | 61.6 | 65.283 | 66.1 | 60.943 | 61.564 | 63.111 | 61.4 | 62.582 | 62.447 |
| HDL | 42.788 | 46.723 | 47.475 | 46.65 | 40.99 | 44.369 | 41.937 | 41.346 | 57.304 | 47.566 |
| Blood glucose levels | 166.442 | 97.997 | 107.035 | 110.763 | 109.272 | 113.065 | 164.927 | 102.371 | 94.912 | 102.83 |
| Myocardial infarction | 0.133 | 0.296 | 0.261 | 0.233 | 0.451 | 0.238 | 0.267 | 0.508 | 0.116 | 0.228 |
| Hematocrit levels | 43.674 | 43.958 | 43.591 | 43.22 | 45.982 | 43.953 | 43.65 | 45.934 | 40.166 | 42.517 |
| Blood pressure status: elevated | 0.233 | 0 | 0 | 0 | 1 | 0 | 0.178 | 0 | 0.466 | 0.138 |
| Blood pressure status: hypertensive crisis | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Blood pressure status: hypertensive status 1 | 0.767 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Blood pressure status: hypertensive status 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0.667 | 0 | 0 | 0.61 |
| Blood pressure status: normal | 0 | 0 | 0 | 0 | 0 | 0 | 0.156 | 1 | 0.534 | 0.252 |
| BMI category: normal | 0.367 | 0.289 | 0.348 | 0.167 | 0.206 | 0 | 0 | 0.232 | 0.548 | 0 |
| BMI category: obese | 0 | 0 | 0 | 0.4 | 0.291 | 1 | 1 | 0.238 | 0 | 1 |
| BMI category: overweight | 0.633 | 0.704 | 0.652 | 0.433 | 0.503 | 0 | 0 | 0.53 | 0.445 | 0 |
| BMI category: underweight | 0 | 0.007 | 0 | 0 | 0 | 0 | 0 | 0 | 0.007 | 0 |
| Number of observations | 30 | 135 | 230 | 30 | 175 | 101 | 45 | 185 | 146 | 123 |

**Table 16** The centroid standard deviation values and number of observations for all identified clusters from the GMM algorithm on the one-hot encoded dataset

| Variable | Cl. 1 | Cl. 2 | Cl. 3 | Cl. 4 | Cl. 5 | Cl. 6 | Cl. 7 | Cl. 8 | Cl. 9 | Cl. 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Gender: female | 0.504 | 0.485 | 0.501 | 0.479 | 0 | 0.492 | 0.49 | 0 | 0 | 0.457 |
| Gender: male | 0.504 | 0.485 | 0.501 | 0.479 | 0 | 0.492 | 0.49 | 0 | 0 | 0.457 |
| Diabetes | 0 | 0 | 0.364 | 0.45 | 0.332 | 0.415 | 0 | 0.237 | 0.164 | 0 |
| Smoking | 0.407 | 0.455 | 0.425 | 0.43 | 0.438 | 0.428 | 0.387 | 0.409 | 0.415 | 0.436 |
| Age | 8.47 | 10.276 | 8.674 | 7.087 | 9.896 | 9.877 | 7.86 | 9.377 | 11.038 | 9.576 |
| HDL | 10.841 | 14.2 | 15.702 | 14.453 | 11.239 | 13.111 | 11.725 | 10.829 | 15.987 | 15.481 |
| Blood glucose levels | 54.182 | 10.026 | 27.025 | 30.544 | 32.953 | 36.836 | 49.842 | 18.316 | 11.488 | 14.155 |
| Myocardial infarction | 0.346 | 0.574 | 0.554 | 0.504 | 0.708 | 0.568 | 0.447 | 0.731 | 0.362 | 0.476 |
| Hematocrit levels | 3.686 | 4.218 | 3.822 | 3.73 | 2.701 | 3.896 | 2.959 | 2.665 | 2.714 | 2.932 |
| Blood pressure status: elevated | 0.43 | 0 | 0 | 0 | 0 | 0 | 0.387 | 0 | 0.501 | 0.347 |
| Blood pressure status: hypertensive crisis | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Blood pressure status: hypertensive status 1 | 0.43 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Blood pressure status: hypertensive status 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0.477 | 0 | 0 | 0.49 |
| Blood pressure status: normal | 0 | 0 | 0 | 0 | 0 | 0 | 0.367 | 0 | 0.501 | 0.436 |
| BMI category: normal | 0.49 | 0.455 | 0.477 | 0.379 | 0.405 | 0 | 0 | 0.424 | 0.499 | 0 |
| BMI category: obese | 0 | 0 | 0 | 0.498 | 0.456 | 0 | 0 | 0.427 | 0 | 0 |
| BMI category: overweight | 0.49 | 0.458 | 0.477 | 0.504 | 0.501 | 0 | 0 | 0.5 | 0.499 | 0 |
| BMI category: underweight | 0 | 0.086 | 0 | 0 | 0 | 0 | 0 | 0 | 0.083 | 0 |
| Number of observations | 30 | 135 | 230 | 30 | 175 | 101 | 45 | 185 | 146 | 123 |

**Table 17** The centroid mean values and number of observations for 10 identified clusters from the `DBSCAN` algorithm on the one-hot encoded dataset

| Variable | Cl. 1 | Cl. 2 | Cl. 3 | Cl. 4 | Cl. 5 | Cl. 6 | Cl. 7 | Cl. 8 | Cl. 9 | Cl. 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Gender: female | 0.41 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Gender: male | 0.59 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| Diabetes | 0.065 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| Smoking | 0.241 | 0.222 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| Age | 62.742 | 50.667 | 73 | 64 | 68 | 60 | 65 | 63 | 56 | 61 |
| HDL | 46.25 | 85.433 | 39 | 29 | 26 | 26 | 45 | 42 | 47 | 36.857 |
| Blood glucose levels | 99.678 | 88.167 | 135 | 135 | 159 | 131.973 | 221 | 152 | 177 | 170.385 |
| Myocardial infarction | 0.304 | 0.222 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| Hematocrit levels | 43.755 | 40.644 | 47.2 | 46.2 | 46.2 | 46.2 | 47 | 39 | 46.7 | 46.7 |
| Blood pressure status: elevated | 0.231 | 0.389 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| Blood pressure status: hypertensive crisis | 0.024 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Blood pressure status: hypertensive status 1 | 0.213 | 0.056 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Blood pressure status: hypertensive status 2 | 0.271 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| Blood pressure status: normal | 0.262 | 0.556 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BMI category: normal | 0.253 | 0.667 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BMI category: obese | 0.298 | 0.111 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| BMI category: overweight | 0.448 | 0.167 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| BMI category: underweight | 0.001 | 0.056 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Number of observations | 1063 | 18 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

We omit the rest of the cluster centroids as they only comprise of one observations. Clusters 3–10 follow this paradigm

**Table 18** The centroid standard deviation values and number of observations for 10 identified clusters from the `DBSCAN` algorithm on the one-hot encoded dataset

| Variable | Cl. 1 | Cl. 2 | Cl. 3 | Cl. 4 | Cl. 5 | Cl. 6 | Cl. 7 | Cl. 8 | Cl. 9 | Cl. 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Gender: female | 0.492 | 0 | NA | NA | NA | NA | NA | NA | NA | NA |
| Gender: male | 0.492 | 0 | NA | NA | NA | NA | NA | NA | NA | NA |
| Diabetes | 0.246 | 0 | NA | NA | NA | NA | NA | NA | NA | NA |
| Smoking | 0.428 | 0.428 | NA | NA | NA | NA | NA | NA | NA | NA |
| Age | 9.886 | 4.863 | NA | NA | NA | NA | NA | NA | NA | NA |
| HDL | 13.799 | 7.712 | NA | NA | NA | NA | NA | NA | NA | NA |
| Blood glucose levels | 11.051 | 6.214 | NA | NA | NA | NA | NA | NA | NA | NA |
| Myocardial infarction | 0.602 | 0.428 | NA | NA | NA | NA | NA | NA | NA | NA |
| Hematocrit levels | 3.801 | 1.576 | NA | NA | NA | NA | NA | NA | NA | NA |
| Blood pressure status: elevated | 0.422 | 0.502 | NA | NA | NA | NA | NA | NA | NA | NA |
| Blood pressure status: hypertensive crisis | 0.152 | 0 | NA | NA | NA | NA | NA | NA | NA | NA |
| Blood pressure status: hypertensive status 1 | 0.409 | 0.236 | NA | NA | NA | NA | NA | NA | NA | NA |
| Blood pressure status: hypertensive status 2 | 0.445 | 0 | NA | NA | NA | NA | NA | NA | NA | NA |
| Blood pressure status: normal | 0.44 | 0.511 | NA | NA | NA | NA | NA | NA | NA | NA |
| BMI category: normal | 0.435 | 0.485 | NA | NA | NA | NA | NA | NA | NA | NA |
| BMI category: obese | 0.458 | 0.323 | NA | NA | NA | NA | NA | NA | NA | NA |
| BMI category: overweight | 0.498 | 0.383 | NA | NA | NA | NA | NA | NA | NA | NA |
| BMI category: underweight | 0.031 | 0.236 | NA | NA | NA | NA | NA | NA | NA | NA |
| Number of observations | 1063 | 18 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

We omit the rest of the cluster centroids as they only comprise of one observations. Clusters 3–10 follow this paradigm

## Scaling experiments

See Fig. 17, 18.

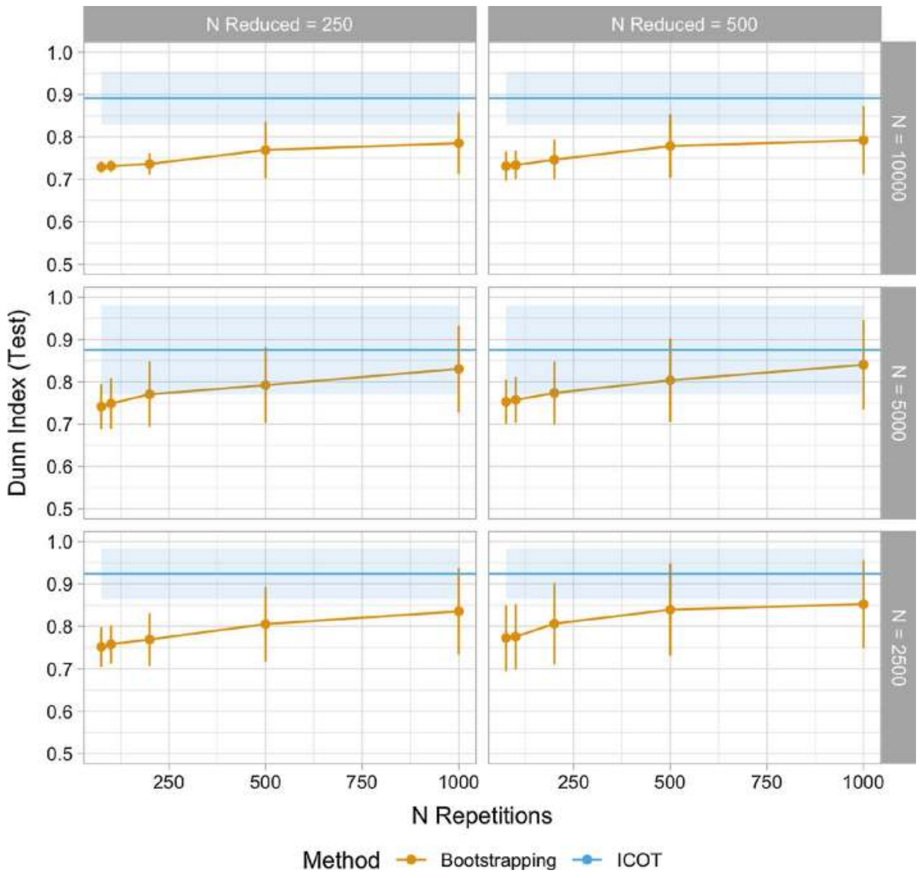

**Fig. 17** Results regarding the impact of bootstrapping on the Dunn Index as the number of repetitions ($N_{\text{rep}}$), sub-sample size ($N_r$), and sample size ($N$) change in the Hubway Dataset. The error bars express the standard deviation of the metric

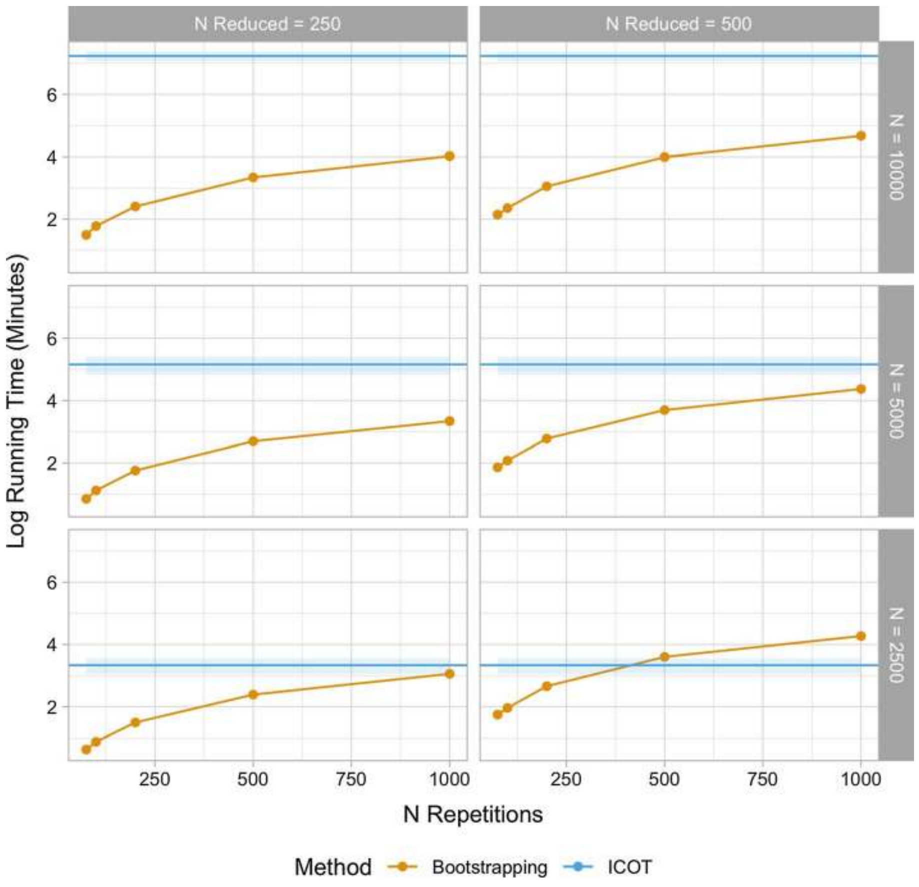

**Fig. 18** Results regarding the impact of bootstrapping on the runtime (Log of Minutes) as the number of repetitions ($N_{\text{rep}}$), sub-sample size ($N_r$), and sample size ($N$) change. Both methods were trained on the Dunn Index. The error bars express the standard deviation of the metric

## References

Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the 18th annual ACM-SIAM symposium on Discrete algorithms* (pp 1027–1035). Society for Industrial and Applied Mathematics.

Basak, J., & Krishnapuram, R. (2005). Interpretable hierarchical clustering by constructing an unsupervised decision tree. *IEEE Transactions on Knowledge and Data Engineering*, *17*, 121–132. https://doi.org/10.1109/TKDE.2005.11.

Bertsimas, D., & Dunn, J. (2017). Optimal classification trees. *Machine Learning*, *106*(7), 1039–1082.

Bertsimas, D., & Dunn, J. (2019). *Machine Learning under a Modern Optimization Lens*. Waltham: Dynamic Ideas Press.

Bertsimas, D., O'Hair, A. K., & Pulleybank, W. R. (2016). *The Analytics Edge*. Waltham: Dynamic Ideas Press.

Bezanson, J., Karpinski, S., Shah, V.B., & Edelman, A. (2012). Julia: A fast dynamic language for technical computing. *arXiv preprint*. arXiv:1209.5145.

Blockeel, H., De Raedt, L., & Ramon, J. (2000) Top-down induction of clustering trees. *arXiv preprint*. arXiv:cs/0011032.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. Boca Raton: CRC Press.

Chavent, M., Guinot, C., Lechevallier, Y., & Tenenhaus, M. (1999). Méthodes divisives de classification et segmentation non supervisée : recherche d'une typologie de la peau humaine saine. *Revue de Statistique Appliquée*, *47*(4), 87–99.

Daniel Levy, S. B. (2006). *A change of heart: Unraveling the mysteries of cardiovascular disease*. New York: Vintage.

Diday, E., & Simon, J. C. (1976). *Clustering analysis* (pp. 47–94). Berlin: Springer. https://doi.org/10.1007/978-3-642-96303-2_3.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint*. arXiv:1702.08608.

Dunn, J. C. (1974). Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, *4*(1), 95–104. https://doi.org/10.1080/01969727408546059org/10.1080/01969727408546059.

Dunn, J. W. (2018). *Optimal trees for prediction and prescription*. PhD thesis, Massachusetts Institute of Technology.

Duran, B., & Odell, P. (1974). *Cluster analysis* (1st ed., p. 100). Berlin: Springer.

Ester, M., Kriegel, H. P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd*, *96*, 226–231.

Everhart, J., & Wright, D. (1995). Diabetes mellitus as a risk factor for pancreatic cancer: A meta-analysis. *JAMA*, *273*(20), 1605–1609.

Feinleib, M., Kannel, W., Garrison, R., McNamara, P., & Castelli, W. (1975). The framingham offspring study. Design and preliminary data. *Preventive Medicine*, *4*(4), 518–525. https://doi.org/10.1016/0091-7435(75)90037-7.

Forgy, E. W. (1965). Cluster analysis of multivariate data: Efficiency versus interpretability of classifications. *Biometrics*, *21*, 768–769.

Fraiman, R., Ghattas, B., & Svarc, M. (2013). Interpretable clustering using unsupervised binary trees. *Advances in Data Analysis and Classification*, *7*(2), 125–145. https://doi.org/10.1007/s11634-013-0129-3. arXiv:1103.5339.

Goodman, B., & Flaxman, S. (2016). European union regulations on algorithmic decision-making and a "right to explanation". *arXiv preprint*. arXiv:1606.08813.

Hahsler, M., Piekenbrock, M., & Doran, D. (2019). dbscan: Fast density-based clustering with r. *Journal of Statistical Software*, *Articles 91*(1), 1–30. https://doi.org/10.18637/jss.v091.i01.

Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, *17*(2), 107–145. https://doi.org/10.1023/A:1012801612483.

Hancock, T. P., Coomans, D. H., & Everingham, Y. L. (2003). Supervised Hierarchical Clustering Using CART. In *Proceedings of MODSIM 2003 International Congress on Modelling and Simulation, Townsville, QLD, Australia* (pp. 1880–1885).

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Unsupervised learning. In *The elements of statistical learning* (pp 485–585). Springer.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, *31*(3), 264–323. https://doi.org/10.1145/331499.331504.

Jolliffe, I. (2011). Principal component analysis. In *International encyclopedia of statistical science* (pp. 1094–1096). Springer.

Kannel, W. B. (1996). Blood pressure as a cardiovascular risk factor: Prevention and treatment. *JAMA*, *275*(20), 1571–1576.

Krim, H., & Hamza, A. B. (2015). *Geometric methods in signal and image analysis*. Cambridge: Cambridge University Press.

Larose, D. T., & Larose, C. D. (2014). *Discovering knowledge in data: An introduction to data mining*. Hoboken: Wiley.

Liu, B., Xia, Y., & Yu, P. S. (2000). Clustering through decision tree construction. In *Proceedings of the ninth international conference on Information and knowledge management - CIKM '00, McLean, VA* (pp. 20–29). https://doi.org/10.1145/354756.354775. arXiv:1011.1669v3.

Liu, Y., Li, Z., Xiong, H., Gao, X., & Wu, J. (2010). Understanding of internal clustering validation measures. In *IEEE 10th International Conference on Data Mining (ICDM), 2010* (pp 911–916). IEEE.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley symposium on mathematical statistics and probability, volume 1: Statistics* (pp. 281–297). Berkeley, CA: University of California Press.

Maulik, U., & Bandyopadhyay, S. (2002). Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(12), 1650–1654.

Mouselimis, L. (2019). ClusterR: Gaussian Mixture Models, K-Means, Mini-Batch-Kmeans, K-Medoids and Affinity Propagation Clustering. *R package version*, *1*(2).

Offner, P. J., Moore, E. E., & Biffl, W. L. (1999). Male gender is a risk factor for major infections after surgery. *Archives of Surgery*, *134*(9), 935–940.

Piatetsky-Shapiro, G., Djeraba, C., Getoor, L., Grossman, R., Feldman, R., & Zaki, M. (2006). What are the grand challenges for data mining?: Kdd-2006 panel report. *ACM SIGKDD Explorations Newsletter*, *8*(2), 70–77.

Radev, D. R., Jing, H., Styś, M., & Tam, D. (2004). Centroid-based summarization of multiple documents. *Information Processing and Management*, *40*(6), 919–938. https://doi.org/10.1016/j.ipm.2003.10.006.

Rao, C. R. (1964). The use and interpretation of principal component analysis in applied research. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, *26*(4), 329–358.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*, 53–65.

Ruspini, E. H. (1970). Numerical methods for fuzzy clustering. *Information Sciences*, *2*(3), 319–350.

Sneath, P. H., Sokal, R. R., et al. (1973). Numerical taxonomy. *The Principles and Practice of Numerical Classification*.

Stepp, R. E., & Michalski, R. S. (1986). Conceptual clustering of structured objects: A goal-oriented approach. *Artificial Intelligence*, *28*(1), 43–69.

Ultsch, A. (2005). Fundamental clustering problems suite (FCPS). Technical report, University of Marburg.

Wolf, P. A., D'Agostino, R. B., Kannel, W. B., Bonita, R., & Belanger, A. J. (1988). Cigarette smoking as a risk factor for stroke: The framingham study. *JAMA*, *259*(7), 1025–1029.

Wu, J., Xiong, H., & Chen, J. (2009). Adapting the right measures for k-means clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 877–886). ACM.

Yang, Q., & Wu, X. (2006). 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, *5*(04), 597–604.