IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 8, 2020, accepted March 3, 2020, date of publication March 13, 2020, date of current version March 25, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2980581

An Explainable Artificial Intelligence Model for Clustering Numerical Databases

OCTAVIO LOYOLA-GONZÁLEZ[1], **ANDRES EDUARDO GUTIERREZ-RODRÍGUEZ**[2], **MIGUEL ANGEL MEDINA-PÉREZ**[3], **RAÚL MONROY**[3], **JOSÉ FRANCISCO MARTÍNEZ-TRINIDAD**[4], **JESÚS ARIEL CARRASCO-OCHOA**[4], **AND MILTON GARCÍA-BORROTO**[5]

[1]Tecnologico de Monterrey, Puebla 72453, Mexico
[2]Tecnologico de Monterrey, San Antonio Buenavista 50110, Mexico
[3]Tecnologico de Monterrey, Estado de Mexico 52926, Mexico
[4]Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla 72840, Mexico
[5]Instituto Superior Politécnico José Antonio Echeverría, Habana 11901, Mexico

Corresponding author: Miguel Angel Medina-Pérez (migue@tec.mx)

ABSTRACT Nowadays, the international scientific community of machine learning has an enormous campaign in favor of creating understandable models instead of black-box models. The main reason is that experts in the application area are showing reluctance due to black-box models cannot be understood by them, and consequently, their results are difficult to be explained. In unsupervised problems, where experts have not labeled objects, obtaining an explanation of the results is necessary because specialists in the application area need to understand both the applied model as well as the obtained results for finding the rationale behind each obtained clustering from a practical point of view. Hence, in this paper, we introduce a clustering based on decision trees (eUD3.5), which builds several decision trees from numerical databases. Unlike previous solutions, our proposal takes into account both separation and compactness for evaluating a feature split without decreasing time efficiency and with no empirical parameter to control the depth of the trees. We tested eUD3.5 on 40 numerical databases of UCI Machine Learning Repository, showing that our proposal builds a set of high-quality unsupervised decision trees for clustering, allowing us to obtain the best average ranking compared with other popular state-of-the-art clustering solutions. Also, from the collection of unsupervised decision trees induced by our proposal, a set of high-quality patterns are extracted for showing the main feature-value pairs describing each cluster.

INDEX TERMS Explainable model, clustering, unsupervised decision trees, numerical databases.

I. INTRODUCTION

Pattern recognition is about classifying objects, i.e., assigning labels to objects [1]. If there is no previous knowledge about the labels of any object in the database, the process of grouping objects in clusters according to some predefined criterion is named clustering [2]. Clustering is one of the most important techniques in pattern recognition. It has been widely studied and applied in many areas [3], [4], such as computer vision [5], information retrieval [6]–[8], marketing [9], and bioinformatics [10].

In general, clustering algorithms differ from one another in the specific clustering criterion they use or how this criterion is measured. One of the most used clustering criteria assumes that objects of the same cluster should be more similar than objects from different clusters [1], [3]. The similarities among objects are determined using a comparison function [11], which is usually a distance function [12].

There is plenty of work about clustering in the literature including k-means [13], EM [14], DBSCAN [15], Pairwise Clustering [16], Kernel k-means [17], DENCLUE [18], x-means [19], Minimum-entropy Clustering [20], Normalized Cut [21], k-medoids [22], SVM Clustering [23], ABC Clustering [24], and PathXP [25]. A recent review of clustering algorithms appears on [26].

In different application areas, such as agriculture, bioinformatics, text processing, and web mining, users need some explanation about clustering results more than just a list

The associate editor coordinating the review of this manuscript and approving it for publication was Kashif Munir.

of objects for each cluster [27]–[30]. For example, on text processing, conventional text clustering techniques do not explain the resulting clusters in terms of the features (words), which is a piece of vital information to understand the topics associated with each cluster [29], [31]. On the other hand, in the context of web mining [32], applying traditional clustering algorithms becomes a laborious and time-consuming process, involving expertise in possibly different domains for getting an explanation of the results. For these applications, a clustering approach based on unsupervised decision trees helps in the interpretation of the results [33].

There are several clustering proposals not based on decision trees, such as k-means [13], EM [14], DBSCAN [15], k-medoids [22], and SVM Clustering [23]. However, clustering based on unsupervised decision trees has the following characteristics: a) no object distance measure is required; b) the object set in the internal nodes is split searching for highly separated and compact groups according to a split evaluation measure; and c) unsupervised trees allow describing the clustering results.

In this paper, we introduced a clustering proposal based on a collection of unsupervised decision trees. Our proposal does not require a parameter that controls the number of objects in the leaf nodes, and also it automatically stops expanding a branch if the new child nodes are evaluated worse than the best evaluation computed in that branch. Also, an important advantage of our proposal is that it provides patterns associated with each cluster, describing the whole database with a few patterns, which is useful in practical applications. In general, patterns are relations between objects' descriptions and their values. Moreover, our proposal uses a traditional clustering ensemble algorithm for grouping objects in all leaves of the generated trees. From our experimental results, we can conclude that our clustering proposal obtains the best position in the Friedman's ranking against other popular state-of-the-art clustering solution, by using 40 numerical databases taken from the UCI Machine Learning Repository [34].

The remainder of the paper is organized as follows. Section II provides preliminaries of decision tree induction and pattern-based machine learning. Section III reviews solutions proposed for clustering. Section IV introduces the new algorithm for pattern-based clustering. Section V presents our experimental setup as well as discusses the performance of our proposal against other popular state-of-the-art clustering solutions on 40 numerical databases. Finally, Section VI provides the conclusions that emerge from this research and our future work.

II. PRELIMINARIES

Since in this paper we propose a clustering algorithm based on building several decision trees, from which several patterns can be extracted, in this section, we briefly describe the procedure for inducing decision trees (Section II-A), some concepts and procedures related to the contrast pattern topic (Section II-B), and a brief introduction to data clustering (Section II-C).

A. DECISION TREE INDUCTION

Decision tree (DT) algorithms have been used in several practical applications, such as bioinformatics [35], blood pressure prediction [36], agriculture [37], and travel time prediction [38], among others. The main reasons are that DTs have shown accurate results, and they provide a model easy to be interpreted by experts in the application area [39].

For inducing a decision tree, there are two main approaches: top-down and bottom-up [40]. As the top-down is the most used [40], and it also will be used by our proposal, we will focus on describing it in a general way.

The top-down approach starts building a root node, which contains all objects of a training database D. After that, the root node is split into P partitions (usually named as children nodes) and repeats this procedure recursively over the children nodes until some stopping criterion is met [40], [41].

From a set of features and splitting criteria, the induction procedure can generate several split candidates S [40], [41], for numerical features the most used is:

- For each feature f_i containing a set of values V belonging to f_i, the following candidates splits can be generated:
  - Binary properties as possible by using $f_i \leq c_j$ and $f_i > c_j$, according to a collection of cut points C, which is generated by computing the midpoint between every two values appearing V, where the objects belonging to different classes [40], [41].

After generating many candidate splits at each node, a measure H for selecting the best candidate split is executed. This measure aims to reward with the highest value to the candidate split that better separates the problem's classes (usually called as pure nodes). Opposite, this measure should penalize those candidate splits producing impure nodes [40], [41].

At the end of the induction process, a pruning method U can be executed for obtaining a new decision tree, improving its performance. The main idea is to remove those nodes from the tree that dot not contribute to obtaining better performance and generating a decision tree less complicated and thus more comprehensible to experts in the application area [40], [41].

Finally, pseudocode for inducing a decision tree was stated in Algorithm 1. Also, more information about decision tree induction is stated in Section III and Section IV.

B. NUMERICAL PATTERNS

Nowadays, there is a great interest in the development of eXplanaible Artificial Intelligence (XAI) models [39]. The main idea is to provide both accurate and explainable models for experts in the application area. Pattern-based Machine Learning has proven to be an approach showing an accurate and explainable model in several practical applications, such as visitor analysis on websites [42], understanding the criminal behavior [43], bot detection [44], and detecting pneumatic failures on temporary immersion bioreactors [45]; among others.

Algorithm 1 BuildDT - Pseudocode for Inducing a Decision Tree

```
input  : D(-) a database.
output: DT(-) a decision tree.
DT ← the root node, containing all the objects in the database D;
if stop criterion == true then
    DT.leaf=true;
    return DT;
end
foreach $feature_i \in \{1 \cdots |feature|\}$ do
    Generate all possible candidate splits $S$ for the $feature_i$;
end
Compute the quality of all candidate splits $S$ by using a measure for assessing split candidate;
$H$ ← Select the candidate split, from $S$, with the highest quality value;
$DS$ ← Partitions of the database D based on the candidate split $H$;
DT.Children = BuildDT($DS$);
return DT
```

In this research, we follow the notion of pattern introduced in [39], [43], [44], [46], [47], a pattern is an expression defined in a certain language that describes a set of objects. Usually, a numerical pattern is represented by a conjunction of relational statements (a.k.a items), each one of the form $[f_i \# v_j]$, where v_j is a value in the domain of feature f_i and # is a relational operator from the set $\{\leq, >\}$. For example, a pattern describing a set of sick patients can be written in a logical form as: $[temperature \leq 39] \wedge [heart\_rate > 110]$. The support (or cover) of a pattern is the ratio of the number of objects described by the pattern in the dataset [44], [46], [47].

A good strategy to extract patterns is using decision trees due to the following reasons:

- The local discretization performed by a decision tree with numeric features has proved better results than using a priori global discretization [47], [48].
- Decision trees contain a small proportion of candidate features even in longer tree paths, which reduces the search space of potential patterns significantly and generate a small collection of high-quality patterns [47]–[49].
- Pattern mining algorithms based on decision trees can handle missing values by introducing a penalizing factor in the measure for evaluating candidate splits [47].

Therefore, in this research, we will obtain a set of good patterns from a collection of trees, very useful for clustering datasets of different contexts.

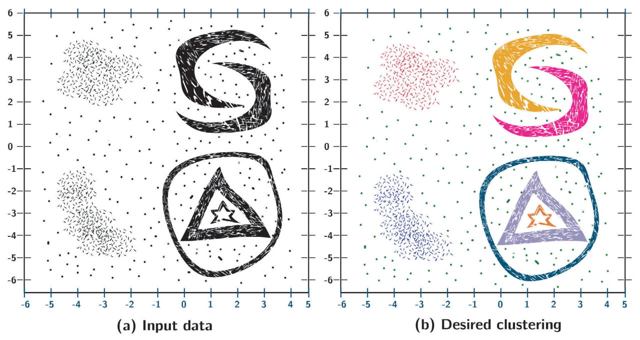


FIGURE 1. An example of input data for clustering is shown in (a). The seven clusters to be found ideally is shown in (b), which were denoted by seven different colors. Notice that clusters have differences regarding shape, size, and density. Unfortunately, none of the available clustering algorithms can detect all these clusters.

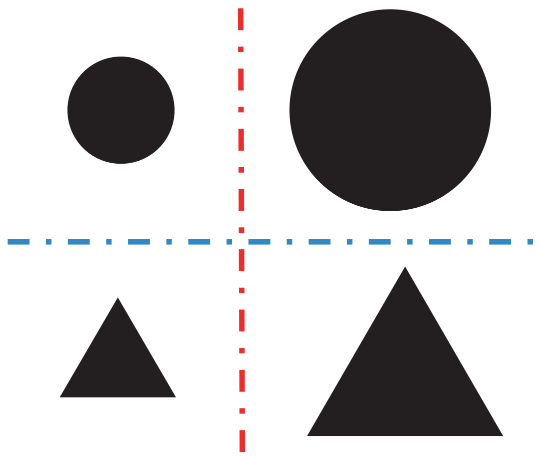


FIGURE 2. An example of clustering four figures into two groups by using the beholders' insight. Usually, there is no consensus on how the figures should be grouped by either their size (red-dashed horizontal line) or geometrical shape (blue-dashed vertical line).

C. DATA CLUSTERING

The goal of data clustering (a.k.a unsupervised classification) is to discover the natural grouping of a set of unlabeled objects, similar to experts can do [3]. However, data clustering is an arduous task due to clusters can differ in terms of their shape, size, and density; and the presence of noise in the data makes the detection of the clusters even more challenging (see Fig. 1). For example, Fig. 2 shows four figures, we have asked 38 people, separately, how can they cluster these figures into two groups by using a line? One-half of these people prefer to cluster the figures taking into account the size (red-dashed horizontal line) and the other half by the shape (blue-dashed vertical line). Notice that using data clustering algorithms can produce different and undesired results when knowledge of experts in the application area is bypassing.

An ideal data clustering can be defined as a set of objects where they are compact and isolated while more similar they are. A cluster is a subjective notion based on the eyes of the beholder and whose significance and interpretation require knowledge of the application area (Fig. 2). It is easy for humans to cluster objects represented in two and possibly three dimensions, but it is complicated for them to create clusters using objects represented by more than three dimensions. It is this challenge, along with the unknown number of clusters for the given data that has resulted in several approaches for data clustering [3].

Currently, there are thousands of clustering algorithms proposed, which can be divided into two groups: hierarchical and partitional [3].

- Hierarchical clustering algorithms aim to find nested clusters, either following an agglomerative approach or using a top-down approach.
  - The agglomerative approach starts a cluster for each object and merging the most similar pair of clusters successively to form a cluster hierarchy.
  - The top-down approach starts with all the objects in one cluster and recursively dividing each cluster into smaller clusters, similar to a decision tree induction procedure.
- Partitional clustering algorithms aim to find all the clusters simultaneously as a partition of the data and do not impose a hierarchical structure. They depend on a similarity matrix with size $n \times n$, where n is the number of input objects. One of the most prominent partitional clustering algorithms is k-means [50], which has shown good clustering results in several contexts [3].

Partitional clustering algorithms rely on a distance measure for grouping the input objects, which biases the clustering results. Ideally, a distance measure should be conformed from the interaction between experts in the application area and mathematics specialists due to experts have the known about the problem to solve and provide important information on how should be the comparison between objects. However, in practice, authors use different distance functions proposed for general purposes, and they select the best one according to the best clustering results obtained from several experiments. Also, it is known that partitional clustering algorithms obtain different results when the distance measure is changed [39].

A clustering approach showing good results is the one based on a collection of clustering algorithms, where each one represents a different partition of the input data [3]. There exist three main ways for building a collection of clustering algorithms, which combine their partitions for obtaining a final data clustering: (i) applying different clustering algorithms [51], (ii) applying the same clustering algorithm but using different values of parameters [52], and (iii) combining of different data representations and clustering algorithms [53]. As was stated by Jain [3], by using a collection of clustering algorithms, it is possible to obtain good data clustering results even when the clusters are not compact and well separated.

Commonly, compactness and separation are two measures to take into account for finding good data clustering. Usually, experts desire to obtain compact and well-separated clusters avoiding the data overlapping because the explanation of overlapped groups is an arduous task [50].

Based on those mentioned above, we hypothesized that building a collection of unsupervised decision trees (using different values of parameters), which takes into account the compactness and separation measures, then, we can obtain a clustering algorithm showing an explanation for each obtained group and reaching better results than other popular state-of-the-art clustering algorithms.

Throughout this section, we have stated a brief introduction to the decision tree induction procedure and pattern-based machine learning as well as some terms and concept for data clustering, which will be used throughout the document. In the next section, we present a set of papers related to our proposal.

III. PREVIOUS WORKS

As our paper introducing a clustering algorithm based on decision trees for numerical databases, we focus this section on review those clustering solutions based on decision trees for numerical data.

An unsupervised decision tree represents a hierarchical clustering. The internal nodes of these trees, including the root node, are decision nodes, which contain a feature-value item assigned to each child node, while leaf nodes represent clusters of objects. The primary purpose of an unsupervised decision tree algorithm is to induce a tree structure where each node represents a good cluster according to a specific criterion. Algorithms which induce supervised decision trees like C4.5 [41], can be modified to induce unsupervised trees, but the split evaluation criteria must be defined according to unsupervised quality measures, i.e., criteria that do not use class-label information. For example, after discretizing numerical features, the splits in a tree could be evaluated based on entropy [33] as follows:

$$H = -\sum_i n_i \log n_i, \qquad (1)$$

where n_i is the frequency of each bin in the discretized numerical feature. To induce an unsupervised decision tree, at every decision node, the data is split based on a single feature. In this case, the feature that reaches the maximum value in (1) is selected for splitting. Then, the objects in the node are partitioned into subsets based on the selected feature, and each subset is allocated into a child node. The induction process ends when, for every feature, the value of (1) is less than a user predefined threshold.

In 1987, Fisher proposed COBWEB, a hierarchical clustering based on a tree structure [54]. COBWEB is induced by merging nodes in hierarchical clustering. COBWEB follows a bottom-up approach by using a merging procedure. COBWEB takes two nodes and combines them if the resulting partition has better quality than they have separately;

consequently, a new node is created, and the probabilities of the feature-value pairs in that node are updated. The two original nodes are made children of the newly created node [54]. COBWEB was proposed for nominal data; consequently, Gennari *et al.* proposed CLASSIT, an extended version of COWEB, but for numerical data [55].

In 1998, Blockeel *et al.* proposed CLUS, a hierarchical clustering algorithm based on unsupervised decision trees [56]. CLUS uses a distance function p (any distance can be used) between objects for computing the prototype of a cluster K_i as $p(K_i)$. The authors proposed (2) for measuring the distance between two clusters K_i and K_j.

$$d(K_i, K_j) = d(p(K_i), p(K_j)) \tag{2}$$

CLUS induces an unsupervised decision tree by using (2) for splitting the nodes recursively until a stopping criterion is met. The number of leaf nodes of the induced decision tree corresponds to the total of clusters found [56].

In 2001, Breiman proposed to create a collection of different decision trees for improving the results obtained by one induced decision tree [57]. The most crucial property of classifier ensembles is diversity. The main reason is that a collection of nearly identical classifiers cannot outperform any of their components [57]. One of the most prominent approaches suggested by Breiman for creating a collection of different decision trees is Random Forest [57]. Random Forest builds 120 different decision trees by selecting a random subset of features in each induction; the size of the subset is $\log_2(|\ Features\ |)$. However, Random Forest was proposed for nominal and numerical databases on supervised problems.

In 2011, Shi and Horvath proposed a variant of Random Forest for building unsupervised decision trees by using the Euclidean distance for evaluating the candidate splits [58]. The authors used the unsupervised decision trees for clustering on tumor marker data. The authors proposed to weight each feature according to its relation and dependency with other features. Shi and Horvath generated a study for knowing the impact of building from 1 to 5,000 unsupervised decision trees. From this study, the authors stated that a collection of 250 decision trees presented the best performance.

In 2018, Kruber *et al.* proposed a modification of the Random Forest algorithm for clustering traffic situations [59]. The authors proposed to use a data-adaptive similarity measure for assessing the candidate splits. After building a collection of unsupervised decision trees, the authors proposed to use a hierarchical clustering approach for generating the final clusters. Although Kruber *et al.* showed good clustering results, the main drawback of this proposal is that it uses a proximity matrix, which depends on the dataset size, producing a high computational cost.

In 2019, Madhyastha *et al.* proposed URerF, a variant of Random Forest for building several unsupervised decision trees to approximately learn geodesic distances in linear and nonlinear manifolds with noise [60]. URerF operates on low-dimensional sparse linear combinations of features, rather than the full observed dimensionality. URerF was tested on both nominal and numeric databases showing good performance for clustering. The authors did not explore any theoretical claims associated with the tested clustering algorithms. Indeed, they did not even assess whether any of these algorithms approximate the precise geodesic.

In 2015, Gutierrez-Rodríguez *et al.* proposed a clustering algorithm (called PCN) for numerical databases based on an explainable artificial intelligence (XAI) model, which follows the rationale behind the Random Forest solution [46]. PCN introduced UD3, an algorithm for inducing a binary unsupervised decision tree recursively by computing the feature split, which maximizes the separation of the means of the nodes (clusters). PCN induces several unsupervised decision trees by using UD3, and it considers the induced trees as hierarchical clusters and combines them by using the CPC [61] algorithm to generate the final clusters.

An advantage of PCN is that it builds trees efficiently, its time complexity fulfills: $O\left(mn\log_2(n)\right) \leq C \leq O\left(mn^2\right)$ where n is the number of objects and m is the number of features (typically $m << n$). PCN has shown better results than obtained by other popular state-of-the-art clustering algorithms, such as COWEB, CLUS, and CPC; among others.

However, PCN has three significant drawbacks. First, UD3 evaluates a feature split as the separation of the means. Considering only the separation of the means may cause very different points distributions to produce the same mean as illustrated in Fig. 3. Second, UD3 requires a parameter that controls the number of objects in the leaf nodes, and the authors do not provide any automatic algorithm to compute this parameter. Third, as we shall show in the experiment section, there are some databases where PCN cannot compute the clusters.

Based on the reviewed literature, we hypothesized that UD3 [46] could be improved without sacrificing efficiency. The main idea is to create a new split evaluation criterion, taking into account both compactness and separation [3]. In this way, in the next section, we propose a new algorithm for inducing unsupervised decision trees differentiating among the three examples shown in Fig. 3. Unlike UD3, our proposal does not require a parameter that controls the number of objects in the leaf nodes, and it automatically stops expanding the branches.

IV. PROPOSED ALGORITHM: UD3.5

Some motivations for clustering data based on unsupervised decision trees are: (i) they are faster to build; (ii) the function for measuring splits can assess specific criteria of clustering allowing partition the data as desired; and (iii) the resulting clusters can be explained by the paths from the roots to the leaves of the trees [46].

A. UD3

As we stated in Section III, a recent research [46] proposes to build binary unsupervised decision trees (UD3) based on the

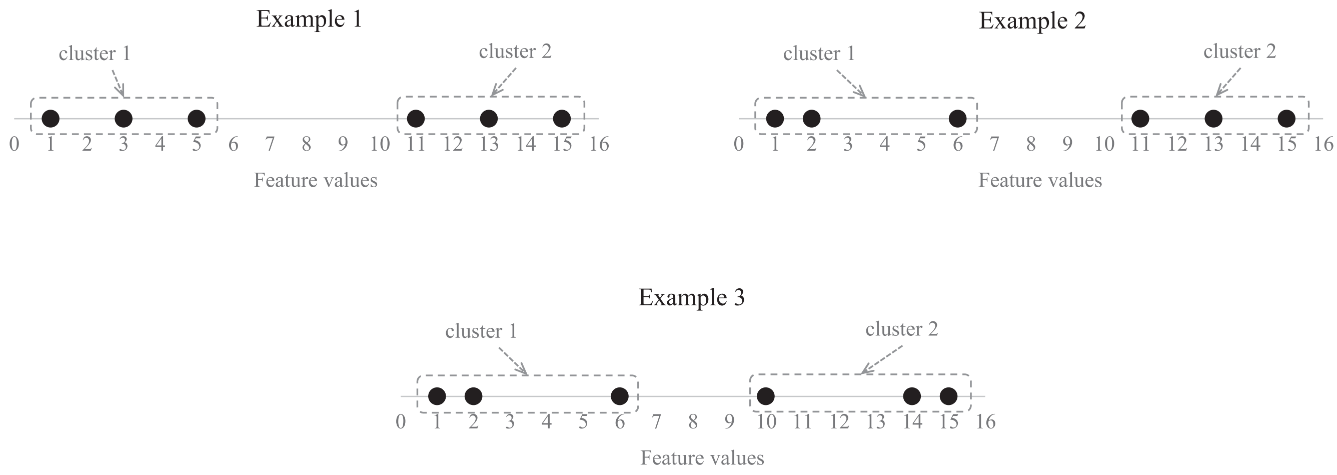


FIGURE 3. Three examples of one-dimensional points clustering where UD3 [46] determines that are equally good when the first example should receive a higher evaluation.

following split evaluation criterion for features:

$$Q(f_j, lm, rm) = \frac{|lm - rm|}{v_{n_j} - v_1}, \qquad (3)$$

where:

- $f_j = \{v_1, v_2, \ldots, v_{n_j}\}$ is a sorted set of n_j values of a one-dimensional feature. For example, in Fig. 3, for the Case 1 is represented as $f_j = \{v_1 = 1, v_2 = 3, v_3 = 5, v_4 = 11, v_5 = 13, v_6 = 15\}$.
- lm and rm are the mean values of the left and the right clusters respectively. For example, in Fig. 3, the values of lm and rm for the three cases are:
  - Case 1: $lm = (1 + 3 + 5)/3 = 3$ and $rm = (11 + 13 + 15)/3 = 13$.
  - Case 2: $lm = (1 + 2 + 6)/3 = 3$ and $rm = (11 + 13 + 15)/3 = 13$.
  - Case 3: $lm = (1 + 2 + 6)/3 = 3$ and $rm = (10 + 14 + 15)/3 = 13$.

A drawback of Q (3) is that it does not take into account the compactness of the clusters in the left and right nodes (i.e., the distance between objects in the same cluster). An example of the undesirable behavior of Q (3) is that it returns the same evaluation (0.714) for the three splits shown in Fig. 3. A good split evaluation criterion should evaluate better case 1, where clusters, in the left and right nodes, are more compact than those in cases 2 and 3. A more accurate alternative is the silhouette index [62], but its evaluation for all possible split values in a feature is costly: $O(n^3)$. However, the evaluation of Q (3) is very efficient since if the feature values are sorted, and the means of the clusters can be dynamically updated while iterating for every possible split value with a time complexity of $O(n \log(n))$ (see Algorithm 2 for an example of pseudocode).

B. UD3.5

In order to take into account the compactness and separation of the clusters, in the left and right child nodes, without sacrificing the efficiency of (3), we propose the following evaluation criterion:

$$Q_1(f_j, k, lm, rm) = \frac{(k-1) * (l1 + l2)/2 + (n_j - k + 1) * (r1 + r2)/2}{n_j}, \qquad (4)$$

where:

- $f_j = \{v_1, v_2, \ldots, v_{n_j}\}$ is a sorted set of n_j values of a one-dimensional feature.
- k is the index of a value in f_j which tags the beginning of the right cluster in a one-dimensional feature. In Fig. 3, the value of k is 4 for the three cases and the obtained clusters are:
  - Case 1: $\{\{v_1 = 1, v_2 = 3, v_3 = 5\}, \{v_4 = 11, v_5 = 13, v_6 = 15\}\}$.
  - Case 2: $\{\{v_1 = 1, v_2 = 2, v_3 = 6\}, \{v_4 = 11, v_5 = 13, v_6 = 15\}\}$.
  - Case 3: $\{\{v_1 = 1, v_2 = 2, v_3 = 6\}, \{v_4 = 10, v_5 = 14, v_6 = 15\}\}$.
- lm and rm are the mean values of the clusters in the left and right child nodes, respectively.
- $(l1 + l2)/2$ evaluates the compactness and separability of the cluster, in the left child node, where $l1 = Q_2(v_0, lm, rm)$ and $l2 = Q_2(v_{k-1}, lm, rm)$ (see (5)). Here the compactness of the left cluster is measured based on the closeness of its extreme values (v_0 and v_{k-1}) with respect to the mean (lm) of this cluster. The separation with respect to the right cluster is measured based on the distance of v_0 and v_{k-1} with respect to the mean (rm) of the right cluster. The function Q_2 is defined

as follows:

$$Q_2(v, m1, m2) = \frac{|v - m2| - |v - m1|}{\max\{|v - m2|, |v - m1|\}}, \qquad (5)$$

where v is a feature value, $m1$ is the mean of the cluster to which v belongs, and $m2$ is the mean of the cluster to which v does not belong. Q_2 returns a value in the interval $[-1, 1]$. A result closer to 1 indicates that v is closer to the mean of its cluster than its distance to the mean of the other cluster. A result closer to -1 indicates that v is closer to the mean of the cluster to which it does not belong. Q_2 is inspired on the silhouette index [62] but Q_2 has less computational complexity. Computing the silhouette index [62] of a single value v has a time complexity of $O(n)$ because v is compared with every element in every cluster. The time complexity of computing Q_2 for a single value v is $O(1)$ because it is constant no matter the number of the elements in the clusters.
The evaluation of $(l1 + l2)/2$ for the examples in Fig. 3 is as follows:

- Case 1: $(Q_2(1, 3, 13) + Q_2(5, 3, 13))/2 \approx 0.792$.
- Case 2: $(Q_2(1, 3, 13) + Q_2(6, 3, 13))/2 \approx 0.702$.
- Case 3: $(Q_2(1, 3, 13) + Q_2(6, 3, 13))/2 \approx 0.702$.

- $(r1 + r2)/2$ is the evaluation of the right cluster where $r1 = Q_2(v_k, rm, lm)$ and $r2 = Q_2\left(v_{n_j}, rm, lm\right)$ (see (5)). Here the compactness of the right cluster is measured based on the closeness of its extreme values (v_k and v_{n_j}) with respect to the mean (rm) of this cluster. The separation with respect to the left cluster is measured based on the distance of v_k and v_{n_j} with respect to the mean (lm) of the left cluster. The evaluation of $(r1 + r2)/2$ for the examples in Fig. 3 is as fallows:
  - Case 1: $(Q_2(11, 13, 3) + Q_2(15, 13, 3))/2 \approx 0.792$.
  - Case 2: $(Q_2(11, 13, 3) + Q_2(15, 13, 3))/2 \approx 0.792$.
  - Case 3: $(Q_2(10, 13, 3) + Q_2(15, 13, 3))/2 \approx 0.702$.
- $(k - 1)$ and $(n_j - k + 1)$ are the number of elements in the left and the right clusters respectively. Multiplying the clusters evaluations by the number of elements favors balanced clusters, i.e., clusters with approximately the same number of elements.

Unlike the split evaluation criterion proposed in [46], which produces the same value for the three cases in Fig. 3, our function Q_1 produces different values for the three cases (case 1: ~ 0.792, case 2: ~ 0.747, and case 3: ~ 0.702) boosting the cluster of the case 1 and penalizing the cluster of the case 3 which is in concordance with the separability and compactness concepts [1], [3].

In order to build UD3 [46], its authors require a minimum support threshold that controls the depth of the tree. This threshold depends on the database, and the authors do not provide any automatic procedure to compute it. Our proposal (UD3.5) does not require this parameter because it splits a node if the found clusters evaluate better than the best evaluation found in their branch according to (4). Algorithm 2 shows a pseudocode of UD3.5.

Algorithm 2 UD3.5

```
Data: $T$ - A database described by a set of features $F$.
      $branchEval$ - The best evaluation (initially 0) found in the branch.
Result: The root node of the unsupervised decision tree.
Build the current node $N$ of the tree with all the objects in $T$
if $N$ is the root node of the tree then
    foreach feature $f_j \in F$ do
        Sort the values $v_k \in f_j$ in ascendent order
Select randomly a subset $F' \subset F$ of $\log_2(|F|) + 1$ features
$bestGlobalEval = 0$, $bestFeature = null$, and $bestValue = 0$
foreach feature $f_j = \{v_1, v_2, \ldots, v_{n_j}\} \in F'$ do
    $leftSum = v_1$, $rightSum = v_2$
    foreach feature value $v_k \in f_j$, $k = 3..n_j$ do
        $rightSum = rightSum + v_k$
    $bestEval = 0$, $bestK = 0$, $bestLeftMean = 0$, and $bestRightMean = 0$
    foreach feature value $v_k \in f_j$, $k = 2..n_j$ do
        if $v_k \neq v_{k-1}$ then
            $leftMean = leftSum/(k - 1)$
            $rightMean = rightSum/(n_j - k + 1)$
            $eval = Q_1(f_j, k, leftMean, rightMean)$ (see (4))
            if $eval > bestEval$ then
                $bestEval = eval$, $bestK = k$,
                $bestLeftMean = leftMean$, and
                $bestRightMean = rightMean$
        $leftSum = leftSum + v_k$
        $rightSum = rightSum - v_k$
    $globalEval = 0$
    foreach feature value $v_k \in f_j$, $k = 1..bestK - 1$ do
        $globalEval = globalEval + Q_2(v_k, bestLeftMean, bestRightMean)$ (see (5))
    foreach feature value $v_k \in f_j$, $k = bestK..n_j$ do
        $globalEval = globalEval + Q_2(v_k, bestRightMean, bestLeftMean)$
    $globalEval = globalEval/n_j$
    if $globalEval > bestGlobalEval$ then
        $bestGlobalEval = globalEval$, $bestFeature = f_j$,
        and $bestValue = v_{bestK}$
if $bestGlobalEval > branchEval$ then
    $branchEval = bestGlobalEval$
    $N_{LeftChild} = UD3.5(\{o | o \in T \wedge bestFeature(o) < bestValue\}, branchEval)$.
    $N_{RightChild} = UD3.5(\{o | o \in T \wedge bestFeature(o) \geq bestValue\}, branchEval)$.
return $N$
```

C. COMPLEXITY ANALYSIS OF UD3.5

Before building the tree, UD3.5 sorts all the values for each feature in the database in ascending order. If we assume the worst-case scenario where every object has a different feature value, then the time complexity of sorting the features is $O(mn \log n)$, where m is the number of features, and n is the number of objects. Q_1 evaluates four times the function Q_2 which time complexity is $O(1)$. Hence, the time complexity of Q_1 for a single value of k is $O(1)$ because it is constant, no matter the number of the elements in the clusters. Computing Q_1 for all possible k in a feature of n values has a time complexity of $O(n)$ because since the feature values are sorted before building the tree, the means of the clusters can be dynamically updated while iterating for every possible split value. Once UD3.5 finds the best split value for a feature, the final evaluation of the feature, used for the stop criterion, is computed by accumulating the evaluation of Q_2 for every value in the feature ($O(n)$). UD3.5 evaluates $\log_2 m$ features at every level; therefore, the time complexity of selecting the best feature is $O(n \log_2 m)$ for the first level. Since the features can be repeated, the time complexity for building all levels of the tree is $O(ln \log_2 m)$, where l is the depth of the tree. If we do not consider the stopping criterion, there will be $l = n - 1$ levels in the worst case (skewed binary tree), and there will be $l = \log_2 n$ levels in the best case (balanced tree). Given that the time complexity of sorting the values of the m features is $O(mn \log n)$; if we assume that $n \geq m$, the overall time complexity C of UD3.5 for building a tree is $O\left(mn \log_2 n\right) \leq C \leq O\left(n^2 \log_2 m\right)$ which is lower than the time complexity of UD3 for the worst-case scenario (see [46] for further details about the time complexity for UD3).

Fig. 4 shows an example of four well-defined clusters of 2D points, which our proposal UD3.5 correctly identifies. The previous work [46] fails to identify the clusters because it relies only on the separation of the means of the clusters without considering the compactness of the clusters with respect to the means.

D. eUD3.5

Building just only one tree for UD3.5 may discard features, which could be useful for finding a good clustering. Therefore, we propose to build 100 different trees, as suggested in [57], by selecting the best feature to split from a random subset of features. We select a trade-off between the best tree (the one which selects the best splits from all features) and the generation of all possible trees, because the former is unique, and the latter is unfeasible in practice due to its time complexity. For each obtained tree, we consider its leaf nodes as clusters that we combine in order to obtain the number of clusters (k), specified by the user.

The previous work proposed in [46] uses CPC [61] to combine the clusters, but there are some databases that CPC cannot cluster due to some overlapping constraint among the seeds are not fulfilled. On the other hand, CPC has a high time complexity: $O(p^2 n)$ according to [61] where p is the number of clusters to be combined and n is the number of objects. Taking into account that the number of trees is constant, if we consider the worst-case scenario where every tree has a leaf for every object, then the time complexity of CPC is $O(n^3)$.

In this paper, in order to combine the clusters obtained from the trees, we propose to use a less complex but effective technique previously reported in the state-of-the-art literature [1], [3]. The cluster ensembles algorithm can be summarized as follows:

1) Every time a pair of objects (o_i, o_j) coincides in a cluster, we increment the value of a similarity matrix s at position (i, j). A high value of $s[i, j]$ indicates that the objects o_i and o_j are very likely to be in the same resulting cluster.
2) Once we fill the matrix s, we apply k-means [13] to cluster the objects using the matrix s as the measure of similarity between objects (the user specifies the number of clusters).

Let us assume the worst-case scenario for our algorithm, i.e., having only two clusters for every tree; then, since the number of generated trees is constant, the time complexity for filling the similarity matrix s is $O(n^2)$. Since the number of iterations of k-means is constant, and we do not compute any distance measure between objects (we use the similarity matrix s to compare the objects), the time complexity of k-means is $O(kn)$. Taking into account that $k \leq n$, we can conclude that the time complexity of combining the clusters obtained from the trees is $O(n^2)$, which is lower than the time complexity of CPC.

In summary, in this section, we have proposed a new algorithm (UD3.5) for building unsupervised decision trees. Based on our example in Fig. 4 and the complexity analysis section, we can conclude that UD3.5 improves the partitions created by UD3 [46] without sacrificing efficiency. Also, UD3.5 is based on a new split evaluation criterion that takes into account both compactness and separation [3] of the clusters; hence it can differentiate among the three examples in Fig. 3. Unlike UD3, our proposal does not require a parameter that controls the number of objects in the leaf nodes. Also, UD3.5 automatically stops expanding a branch if the new child nodes are evaluated worse than the best evaluation computed in that branch. Additionally, for creating diversity, we build 100 different unsupervised decision trees by using the same feature selection strategy of Random Forest [57] and our proposal, UD3.

In the next section, our proposal is assessed over 40 numerical databases taken from the UCI Machine Learning Repository [34].

V. EXPERIMENTAL FRAMEWORK

This section shows the experiments designed to evaluate the performance of the proposed algorithm. For our experiments, we used 40 well-known databases taken from the UCI Machine Learning Repository [34]. Table 1 shows, for each database, its name in the repository (Database), the number

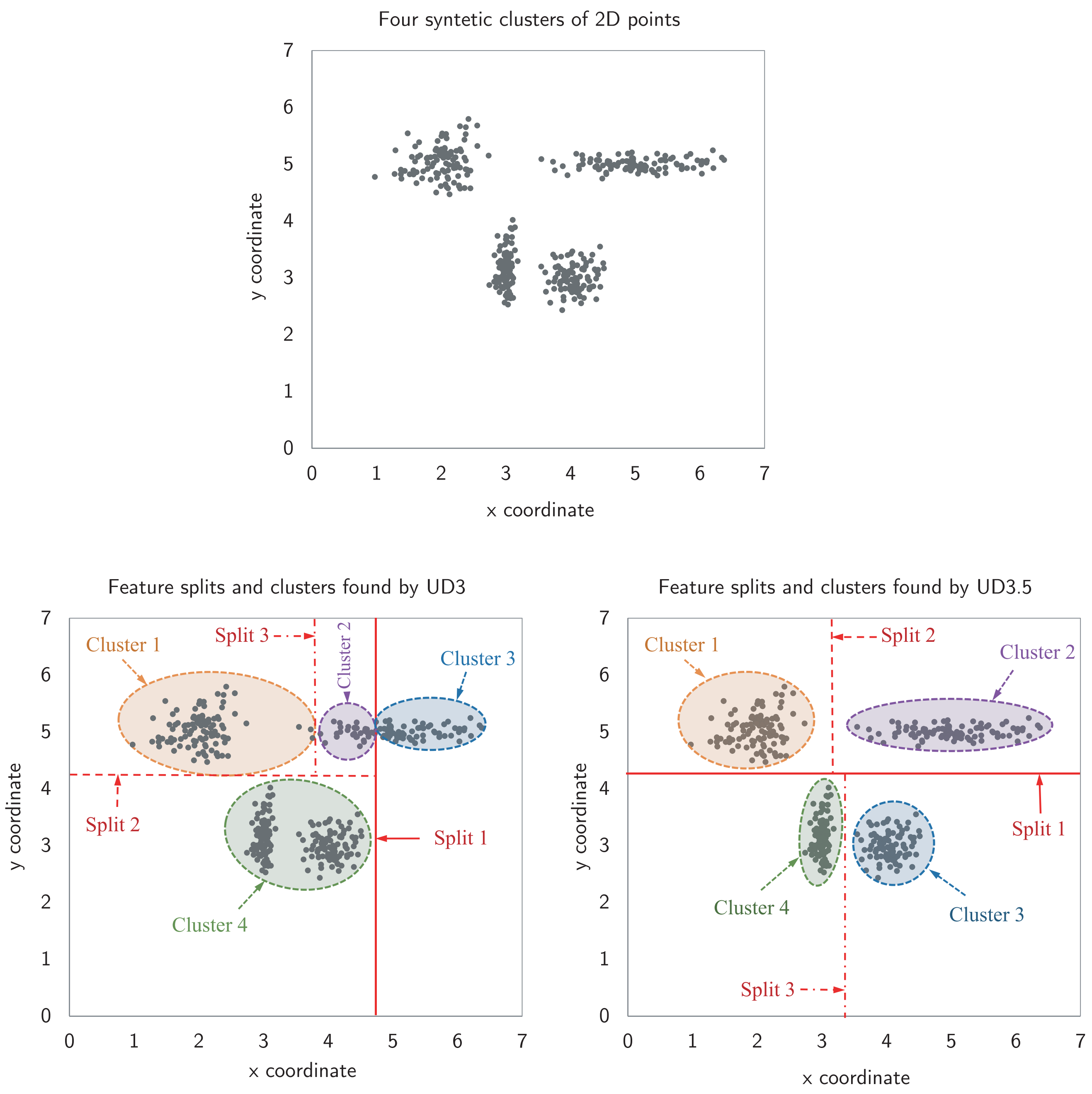


FIGURE 4. Synthetic 2D points with the clusters computed by our proposal (UD3.5) and by the algorithm (UD3) proposed in [46].

of objects (# objects), the number of features (# features), and the number of classes (# classes).

We evaluated the ensemble explained in Section IV-D, which is based on our UD3.5 algorithm (we named eUD3.5 to our ensemble). We also evaluated the clustering algorithm (PCN) based on UD3 [46] and CPC [61] because it has shown better results than other popular state-of-the-art clustering algorithms based on unsupervised decision trees, such as COWEB [54], CLASSIT [55], and CLUS [56]. Moreover, we evaluated URerF [60] because, as far as we know, it is the most recent clustering algorithm using a collection of decision trees. Additionally, we tested three of the most popular clustering algorithms not based on decision trees: k-means [13], EM [14], and DBSCAN [15], because they have shown good results on numerical databases regarding several clustering proposals.

To execute the clustering algorithms mentioned above, we leveraged the scikit-learn library [63] for executing DBSCAN, and the scikit-learn library jointly with the code published by [60] (at https://sporf.neurodata.io) for executing URerF. Also, we have used the Weka Data mining tool [64] for executing both k-means and EM. Additionally, we have

TABLE 1. Description of the databases used in our experiments.

Database	# objects	# features	# classes
banknote	1372	4	2
biodeg	1055	41	2
breast-tissue	106	9	6
breast-w	699	9	2
climate	540	20	2
cloud	108	4	4
ctg	2126	41	3
diabetes	768	8	2
ecoli	336	7	8
faults	1941	27	7
glass	214	10	6
ilpd	583	11	2
ionosphere	351	34	2
iris	150	4	3
knowledge	403	5	4
land-cover	675	147	9
liver-disorders	345	6	2
mammographic	961	6	2
movement	360	90	15
ozone	2534	72	2
page-blocks	5473	10	5
parkinsons	195	22	2
seeds	210	7	3
segment	2310	19	7
sensor-readings	5456	24	4
sonar	208	60	2
spambase	4601	57	2
spectf	267	44	2
spectrometer	531	100	4
transfusion	748	4	2
vehicle	846	18	4
vertebral-2	310	6	2
vertebral-3	310	6	3
vowel	990	10	11
waveform-1	5000	21	3
waveform-2	5000	40	3
wholesale	440	7	2
wine	178	13	3
winequality	6497	11	7
yeast	1484	8	10

used the implementation of UD3 and PCN provided by their authors [46]. Consequently, we have modified the source code provided by [46] for creating our implementations of UD3.5 and eUD3.5. All tested clustering algorithms were executed by using the parameter values recommended by their authors.

We followed the experimental protocol of [46], which used F-measure [65] to evaluate the clustering results. F-measure evaluates the dependence between C classes in the database and K clusters built by the clustering algorithms as follows:

$$F - measure(C, K) = \sum_{c_i \in C} \frac{\mid C_i \mid}{\mid T \mid} \max_{k_j \in K} \{F(c_i, k_j)\}, \qquad (6)$$

where

$$F(c_i, k_j) = \frac{2 * Recall(c_i, k_j) * Precision(c_i, k_j)}{Recall(c_i, k_j) * Precision(c_i, k_j)} \qquad (7)$$

and $\mid T \mid$ represents the number of objects in the database, C is the set of classes, K is the set of clusters, $Recall(c_i, k_j) = n_{ij} / \mid c_i \mid$, $Precision(c_i, k_j) = n_{ij} / \mid k_i \mid$, and n_{ij} is the number of objects of the class $c_i \in C$ belonging to cluster $k_j \in K$.

Finally, to determine the statistical significance of differences among the clustering results, we applied the Friedman test [66]. After, if the Friedman test outputs statistical differences, we will apply the Finner dynamic post-hoc, as was proposed by [67] for knowing which are these differences. We compared the results considering a level of significance $\alpha = 0.05$.

A. EXPERIMENTAL RESULTS

Table 2 shows the results obtained by the six tested clustering algorithms where the name of the database and the F-measure value associated with each tested clustering algorithm are shown. From this table, we can notice that DBSCAN achieves the best results (10 victories) against the remaining tested clustering algorithms. Notice that PCN achieves an F-measure value of 0.0 in 9 databases because the algorithm computes some seeds in which overlapping constraints are not fulfilled. However, it is essential to note that the number of victories does not determine if a clustering algorithm is better than the remaining ones; for knowing this,

TABLE 2. F-measure reached by the testing clustering algorithms in the databases shown in Table 1. The best result for each database appears in bold. Notice that for the iris database, both eUD3.5 and PCN shared the best result.

Database	eUD3.5	URerF	PCN	k-means	EM	DB SCAN
banknote	**0.5987**	0.5371	0.5985	0.5771	0.5782	0.5940
biodeg	0.6522	0.6316	**0.7021**	0.6433	0.6174	0.6984
breast-tissue	0.5404	0.3841	0.0000	0.4849	**0.5810**	0.2917
breast-w	0.9414	0.9646	**0.9702**	0.9569	0.9436	0.5744
climate	0.7329	0.6933	0.6399	0.7306	0.7161	**0.8875**
cloud	0.3454	0.3768	**0.4117**	0.3660	0.3379	0.4000
ctg	0.6915	0.5312	0.5246	**0.7704**	0.5176	0.7280
diabetes	0.6794	0.6579	0.6144	0.6656	0.6674	**0.6940**
ecoli	0.7059	0.7041	0.0000	0.6919	**0.7075**	0.4255
faults	**0.4696**	0.3463	0.4304	0.4069	0.4491	0.3198
glass	0.5276	0.5567	0.5666	**0.6998**	0.5390	0.4387
ilpd	0.6912	0.7205	0.6498	0.6498	0.6268	**0.7218**
ionosphere	0.7195	0.723	0.7292	0.7177	**0.7538**	0.6811
iris	**0.9600**	0.8912	**0.9600**	0.8853	0.9048	0.7053
knowledge	**0.7806**	0.606	0.7380	0.5014	0.4879	0.4256
land-cover	0.5312	0.2761	0.0000	**0.6765**	0.6394	0.2384
liver-disorders	0.6245	0.6403	0.5646	0.6243	0.6096	**0.6742**
mammographic	0.6499	0.6467	0.7703	0.7776	**0.7943**	0.6446
movement	0.4405	0.4914	0.0000	0.4787	**0.5435**	0.1951
ozone	0.6619	0.7600	0.7207	0.7112	0.6264	**0.9138**
page-blocks	0.8583	0.8048	0.6412	0.6561	0.4958	**0.8605**
parkinsons	0.7024	0.7286	0.7277	0.6500	0.6895	**0.7453**
seeds	0.8893	0.8897	0.8752	**0.8905**	0.8747	0.4945
segment	0.2537	0.5397	0.6446	**0.6555**	0.6290	0.2500
sensor-readings	**0.5206**	0.4158	0.4256	0.4134	0.4210	0.4741
sonar	0.5753	0.5825	0.5719	**0.7977**	0.5385	0.6368
spambase	0.6781	0.6567	0.0000	**0.7990**	0.7736	0.6651
spectf	0.7183	0.5775	0.6980	0.6909	0.6729	**0.7732**
spectrometer	0.4462	0.4184	0.0000	0.4521	0.4435	**0.4583**
transfusion	0.6438	**0.7095**	0.6669	0.6305	0.6000	0.6562
vehicle	**0.4754**	0.4382	0.4347	0.4256	0.4237	0.4003
vertebral-2	0.6536	0.6348	0.7066	0.6600	**0.7222**	0.7045
vertebral-3	**0.6637**	0.5074	0.6204	0.5720	0.6146	0.5357
vowel	0.2860	**0.3889**	0.0000	0.3505	0.3378	0.1752
waveform-1	0.5388	0.6332	**0.5668**	0.5320	0.5386	0.5000
waveform-2	0.5542	**0.6183**	0.5604	0.5295	0.5409	0.5000
wholesale	0.7019	0.5908	**0.7752**	0.6135	0.6123	0.7044
wine	0.9323	0.7069	0.8716	0.9436	**0.9720**	0.5070
winequality	0.4350	0.2871	0.0000	0.3485	0.3598	**0.4795**
yeast	0.3828	0.3973	0.0000	**0.4453**	0.4038	0.3646
Victories	**7**	**3**	**6**	**8**	**7**	**10**

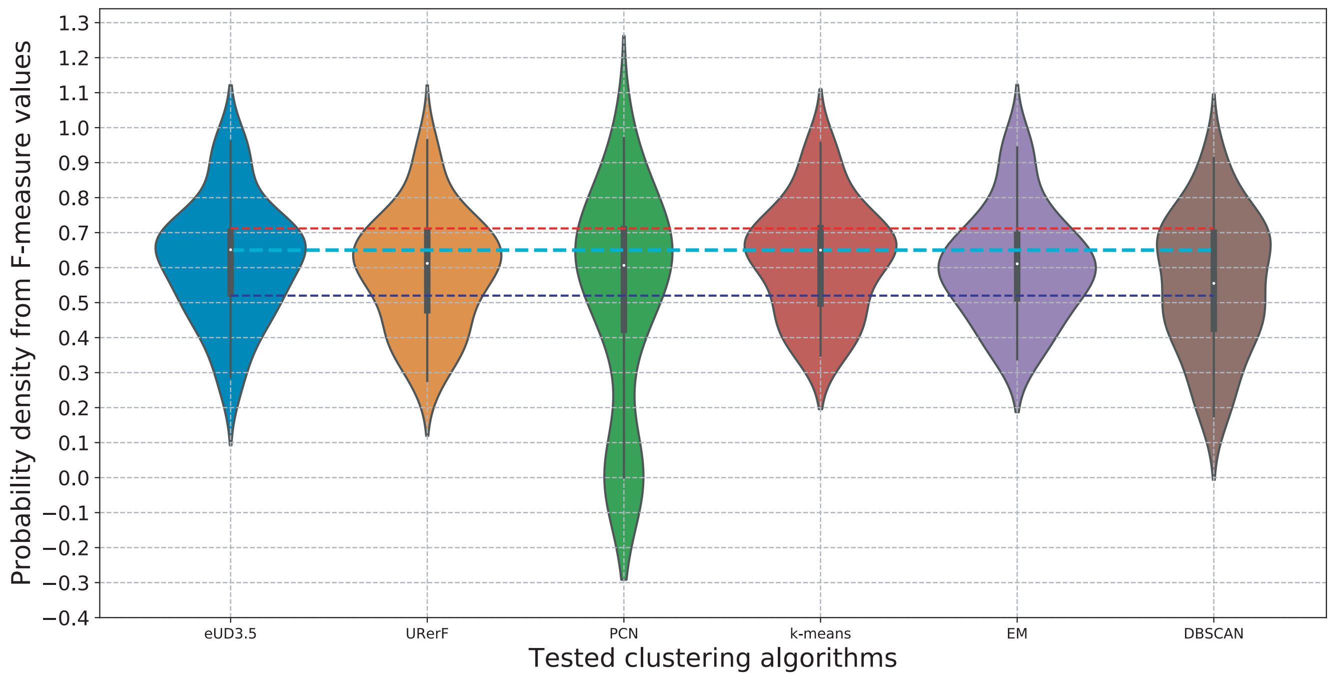


FIGURE 5. Violin plot showing all results obtained from all tested clustering algorithms.

a group of statistical methods and measures must be applied as shown in Table 4.

In Fig. 5, we show a violin plot from our obtained results for more details. In general, a violin plot is a combination of the box plot with a kernel density plot. In the violin plot, we can find more information than the box plot. Inside the violin figure, we have the minimum and maximum values, the median (the white dot inside the thin black line inside the violin), and the first and third quartiles (top and bottom side of the thick black line, respectively, inside the violin) for the F-measure. We can get an idea of how consistent the results are because quartiles closer to the median indicate lower variability in the results. F-measure values from the thin black line inside the violin to the point outside in the border of the violin are considered outliers. The best possible value for F-measure is 1.0, which corresponds to perfect clustering. A violin plot can show the statistics mentioned above, and it also shows the entire distribution of the data (the form of the violin figure).

From Fig. 5, we can see that our proposal and k-means obtain the best medians (see the dashed cyan line) regarding the remaining tested clustering algorithms. However, notice that our proposal obtains the lowest variability in the results because it has its quartiles closer to the median (see dashed red and blue lines). Also, it is important to highlight that our proposal, URerF, k-means, EM, and DBSCAN follow a normal distribution for their outputs. However, notice that PCN follows a multi-modal distribution for their outputs, i.e., a distribution with more than one peak. Those models outputting results that follow a multi-modal distribution are complicated to interpret because, naturally, by using a new set of databases, they would get a set of values where the mean and the median can change drastically [68].

TABLE 4. Statistical results for all the tested clustering algorithms, considering all the tested databases.

Clustering	Average F-measure	SD	Variance	Ranking	Unadjusted p-value
eUD3.5	0.6213	0.1668	0.0278	2.988	-
k-means	0.6268	0.1579	0.0249	3.363	0.370028
URerF	0.5916	0.1618	0.0262	3.600	0.143152
PCN	0.5094	0.3013	0.0908	3.650	0.113267
EM	0.6076	0.1557	0.0242	3.700	0.088531
DBSCAN	0.5534	0.1876	0.0352	3.700	0.088531

Table 4 shows the average F-measure, the standard deviation (SD), the average ranking according to the Friedman's test, and the unadjusted p-value of the Finner's procedure for all tested clustering algorithms, considering all the databases stated in Table 1. This table is ordered according to the average of Friedman's ranking value.

After applying the Friedman's test, it outputs a p-value of 0.479406, accepting the null hypothesis (all analyzed clustering algorithm have a similar performance from a statistical point of view). However, from Table 4, we can see that our proposal reached the best ranking maintaining the second-best average F-measure (slightly surpassed by k-means).

As we saw throughout this section, our proposal (eUD3.5) and k-means have shown similar performances. However, one advantage of eUD3.5 with respect to the other clustering algorithms is that it can describe the computed clusters by means of the patterns extracted from every decision tree. For example, Table 3 shows the more general patterns that

TABLE 3. **The patterns that eUD3.5 uses to describe the three clusters that it computes by using the iris database, taken from the UCI Machine Learning Repository [34].**

ID	Patterns	Percent coverage by cluster		
		Cluster 1	Cluster 2	Cluster 3
P_1	$[petallength \leq 2.45]$	100.00	0.00	0.00
P_2	$[petalwidth \leq 0.80]$	100.00	0.00	0.00
P_3	$[petalwidth \in [0.80, 1.65]]$	0.00	100.00	0.00
P_4	$[petallength \geq 2.45] \wedge [petalwidth \leq 1.65]$	0.00	100.00	0.00
P_5	$[petalwidth \geq 0.80] \wedge [petallength \geq 4.95] \wedge [sepalwidth \leq 2.35]$	0.00	1.92	0.00
P_6	$[petalwidth <= 1.70] \wedge [petallength \geq 4.95] \wedge [sepallength \in [6.95, 7.55]]$	0.00	1.92	0.00
P_7	$[petallength \in [2.45, 4.95]]$	0.00	90.38	14.58
P_8	$[petalwidth \geq 0.80] \wedge [petallength \leq 4.95]$	0.00	90.38	14.58
P_9	$[petalwidth \geq 0.80]$	0.00	100.00	100.00
P_{10}	$[petallength \geq 2.45]$	0.00	100.00	100.00
P_{11}	$[petalwidth \geq 1.65]$	0.00	0.00	100.00
P_{12}	$[petalwidth \geq 0.80] \wedge [petallength \geq 4.95] \wedge [sepalwidth \in [2.35, 2.55]]$	0.00	0.00	6.25
P_{13}	$[petallength \geq 4.95] \wedge [petalwidth \leq 2.05] \wedge [sepallength \geq 7.55]$	0.00	0.00	4.17
P_{14}	$[petalwidth \geq 0.80] \wedge [petallength \geq 4.95] \wedge [sepalwidth \geq 2.95]$	0.00	1.92	56.25
P_{15}	$[petallength \geq 4.95]$	0.00	9.62	85.42

eUD3.5 computes when clustering the iris database, see Section II for more details about pattern-based clustering. From Table 3, we can see that *cluster*1 can be explained by using P_1, *cluster*2 by using P_3, and *cluster*3 by using P_{11} because they are covering 100% of objects belonging to each cluster. Notice that these patterns are short and easy to be understood by an expert in the application area.

From these results, we can conclude that eUD3.5 improves the F-measure and efficiency of PCN, which is the clustering algorithm most related to our work. Also, eUD3.5 can describe all the obtained clusters through the patterns associated with each cluster. Another important conclusion is that our proposal can achieve an average F-measure result similar to k-means, and our proposal does not require any distance measure. Notice that those machine learning models containing distance measures are very hard to explain and to be understood by experts in practical domains [39].

VI. CONCLUSION AND FUTURE WORK

The clustering algorithms based on unsupervised decision trees provide an easy way for describing each cluster by following the path from the root to the leaves.

In this paper, we proposed eUD3.5, a new algorithm for inducing a collection of diverse unsupervised decision trees. eUD3.5 relies on a new split evaluation criterion, which improved the performance of PCN without sacrificing efficiency. Also, our proposal reached the best position in Friedman's ranking compared with other popular state-of-the-art clustering algorithms.

Unlike other state-of-the-art clustering algorithms based on decision trees, our proposal does not require a parameter that controls the number of objects in the leaf nodes, and also it automatically stops expanding a branch if the new child nodes are evaluated worse than the best evaluation computed in that branch.

An important advantage of our proposal is that it can provide patterns associated with each cluster, describing the whole database with a few patterns by keeping those which are more general. This kind of description is useful in some applications (e.g., medicine, biotechnology, psychology, and banking sector) where the specialists analyze the clusters for accepting or rejecting some hypothesize, for designing further studies, and ultimately explaining the phenomena.

As future work, we will focus on extending our proposal to deal with both categorical and numeric features, large databases, and dynamic data. We also plan to use our algorithm for discovering the structure of small classes and use this information within supervised classifiers in order to improve the classification accuracy on imbalanced databases.

REFERENCES

[1] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, 2nd ed. Hoboken, NJ, USA: Wiley, 2014.

[2] Y. Li, J. Cai, H. Yang, J. Zhang, and X. Zhao, "A novel algorithm for initial cluster center selection," *IEEE Access*, vol. 7, pp. 74683–74693, 2019.

[3] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, Jun. 2010.

[4] J. Rodríguez-Ruiz, R. Monroy, M. Medina-Pérez, O. Loyola-González, and B. Cervantes, "Cluster validation in clustering-based one-class classification," *Expert Syst.*, vol. 36, no. 6, 2019, Art. no. e12475. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.12475

[5] Y. Martínez-Díaz, N. Hernández, R. J. Biscay, L. Chang, H. Méndez-Vázquez, and L. E. Sucar, "On Fisher vector encoding of binary features for video face recognition," *J. Vis. Commun. Image Represent.*, vol. 51, pp. 155–161, Feb. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1047320318300245

[6] M. A. Álvarez-Carmona, M. Franco-Salvador, E. Villatoro-Tello, M. Montes-y-Gómez, P. Rosso, and L. Villaseñor-Pineda, "Semantically-informed distance and similarity measures for paraphrase plagiarism identification," *J. Intell. Fuzzy Syst.*, vol. 34, no. 5, pp. 2983–2990, May 2018.

[7] M. A. Álvarez-Carmona, L. Pellegrin, M. Montes-y-Gómez, F. Sánchez-Vega, H. J. Escalante, A. P. López-Monroy, L. Villaseñor-Pineda, and E. Villatoro-Tello, "A visual approach for age and gender identification on Twitter," *J. Intell. Fuzzy Syst.*, vol. 34, no. 5, pp. 3133–3145, May 2018.

[8] M. A. Álvarez-Carmona, E. Villatoro-Tello, M. Montes-Y-Gómez, and L. Villaseñor-Pineda, "A comparative analysis of distributional term representations for author profiling in social media," *J. Intell. Fuzzy Syst.*, vol. 36, no. 5, pp. 4857–4868, May 2019.

[9] A. Miklosik, M. Kuchta, N. Evans, and S. Zak, "Towards the adoption of machine learning-based analytical tools in digital marketing," *IEEE Access*, vol. 7, pp. 85705–85718, 2019.

[10] Y. Guo, F.-L. Chung, G. Li, and L. Zhang, "Multi-label bioinformatics data classification with ensemble embedded feature selection," *IEEE Access*, vol. 7, pp. 103863–103875, 2019.

[11] S. Bandyopadhyay and S. Saha, *Unsupervised Classification: Similarity Measures, Classical and Metaheuristic Approaches, and Applications*. Berlin, Germany: Springer-Verlag, 2013, pp. 59–73. [Online]. Available: https://link.springer.com/book/10.1007/978-3-642-32451-2#about

[12] C. Cheng and F. Xiao, "A new distance measure of belief function in evidence theory," *IEEE Access*, vol. 7, pp. 68607–68617, 2019.

[13] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, Oakland, CA, USA, 1967, vol. 1, no. 14, pp. 281–297.

[14] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Process. Mag.*, vol. 13, no. 6, pp. 47–60, Nov. 1996.

[15] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining (KDD)*, 1996, vol. 96, no. 34, pp. 226–231.

[16] T. Hofmann and J. M. Buhmann, "Pairwise data clustering by deterministic annealing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 1, pp. 1–14, Jan. 1997.

[17] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.

[18] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Proc. 4th. Int. Conf. Knowl. Discovery. Data Mining*, vol. 98, Aug. 1998, pp. 58–65.

[19] D. Pelleg and A. W. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *Proc. 17th Int. Conf. Mach. Learn. (ICML)*. San Francisco, CA, USA: Morgan Kaufmann, 2000, pp. 727–734.

[20] S. J. Roberts, C. Holmes, and D. Denison, "Minimum-entropy data clustering using reversible jump Markov chain Monte Carlo," in *Artificial Neural Networks—ICANN* (Lecture Notes in Computer Science), vol. 2130, G. Dorffner, H. Bischof, and K. Hornik, Eds. Berlin, Germany: Springer, 2001, pp. 103–110.

[21] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[22] A. S. Hadi, L. Kaufman, and P. J. Rousseeuw, "Finding groups in data: An introduction to cluster analysis," *Technometrics*, vol. 34, no. 1, p. 111, Feb. 1992.

[23] S. Winters-Hilt and S. Merat, "SVM clustering," *BMC Bioinf.*, vol. 8, no. S7, p. S18, Nov. 2007.

[24] D. Karaboga and C. Ozturk, "A novel clustering approach: Artificial bee colony (ABC) algorithm," *Appl. Soft Comput.*, vol. 11, no. 1, pp. 652–657, Jan. 2011.

[25] M. Piernik, D. Brzezinski, and T. Morzy, "Clustering XML documents by patterns," *Knowl. Inf. Syst.*, vol. 46, no. 1, pp. 185–212, Jan. 2016.

[26] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Ann. Data Sci.*, vol. 2, no. 2, pp. 165–193, Jun. 2015.

[27] R. S. Michalski, K. A. Kaufman, J. Pietrzykowski, J. Wojtusiak, S. Mitchell, and D. Seeman, "Natural induction and conceptual clustering: A review of applications," *Rep. Mach. Learn. Inference Lab.*, vol. 1051, pp. 3–06, 2006.

[28] B. B. Baridam and O. Owolabi, "Conceptual clustering of RNA sequences with codon usage model," *Global J. Comput. Sci. Technol.*, vol. 10, no. 8, pp. 41–45, 2010.

[29] A. Hotho, S. Staab, and G. Stumme, "Explaining text clustering results using semantic structures," in *Knowledge Discovery in Databases: PKDD*, N. Lavrač, D. Gamberger, L. Todorovski, and H. Blockeel, Eds. Berlin, Germany: Springer, 2003, pp. 217–228.

[30] I. Tiddi, M. d'Aquin, and E. Motta, "Dedalo: Looking for clusters explanations in a labyrinth of linked data," in *The Semantic Web: Trends and Challenges* (Lecture Notes in Computer Science), vol. 8465, V. Presutti, C. d'Amato, F. Gandon, M. d'Aquin, S. Staab, and A. Tordai, Eds. Springer, 2014, pp. 333–348.

[31] S. Yang, G. Huang, and B. Cai, "Discovering topic representative terms for short text clustering," *IEEE Access*, vol. 7, pp. 92037–92047, 2019.

[32] H. S. Maghdid, "Web news mining using new features: A comparative study," *IEEE Access*, vol. 7, pp. 5626–5641, 2019.

[33] J. Basak and R. Krishnapuram, "Interpretable hierarchical clustering by constructing an unsupervised decision tree," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 1, pp. 121–132, Jan. 2005.

[34] D. Dua and C. Graff. (2019). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[35] G. Stiglic, S. Kocbek, I. Pernek, and P. Kokol, "Comprehensive decision tree models in bioinformatics," *PLoS ONE*, vol. 7, no. 3, Mar. 2012, Art. no. e33812, doi: 10.1371/journal.pone.0033812.

[36] B. Zhang, J. Ren, Y. Cheng, B. Wang, and Z. Wei, "Health data driven on continuous blood pressure prediction based on gradient boosting decision tree algorithm," *IEEE Access*, vol. 7, pp. 32423–32433, 2019.

[37] C.-C. Yang, S. O. Prasher, P. Enright, C. Madramootoo, M. Burgess, P. K. Goel, and I. Callum, "Application of decision tree technology for image classification using remote sensing data," *Agricult. Syst.*, vol. 76, no. 3, pp. 1101–1117, Jun. 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0308521X02000513

[38] J. Cheng, G. Li, and X. Chen, "Research on travel time prediction model of freeway based on gradient boosting decision tree," *IEEE Access*, vol. 7, pp. 7466–7480, 2019.

[39] O. Loyola-González, "Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view," *IEEE Access*, vol. 7, pp. 154096–154113, 2019.

[40] L. Rokach and O. Maimon, *Data Mining With Decision Trees: Theory and Applications*, 2nd ed. River Edge, NJ, USA: World Scientific, 2014.

[41] J. R. Quinlan, *C4.5: Programs for Machine Learning*, vol. 1. San Mateo, CA, USA: Morgan Kaufmann, 1993.

[42] B. Cervantes, F. Gómez, R. Monroy, O. Loyola-González, M. A. Medina-Pérez, and J. Ramírez-Márquez, "Pattern-based and visual analytics for visitor analysis on Websites," *Appl. Sci.*, vol. 9, no. 18, p. 3840, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/18/3840

[43] O. Loyola-González, "Understanding the criminal behavior in Mexico City through an explainable artificial intelligence model," in *Advances in Soft Computing*, L. Martínez-Villaseñor, I. Batyrshin, and A. Marín-Hernández, Eds. Cham, Switzerland: Springer, 2019, pp. 136–149.

[44] O. Loyola-González, R. Monroy, J. Rodríguez, A. López-Cuevas, and J. I. Mata-Sánchez, "Contrast pattern-based classification for bot detection on Twitter," *IEEE Access*, vol. 7, pp. 45800–45817, 2019.

[45] O. Loyola-González, M. Medina-Pérez, D. Hernández-Tamayo, R. Monroy, J. Carrasco-Ochoa, and M. García-Borroto, "A pattern-based approach for detecting pneumatic failures on temporary immersion bioreactors," *Sensors*, vol. 19, no. 2, p. 414, 2019. [Online]. Available: http://www.mdpi.com/1424-8220/19/2/414

[46] A. E. Gutierrez-Rodríguez, J. F. Martínez-Trinidad, M. García-Borroto, and J. A. Carrasco-Ochoa, "Mining patterns for clustering on numerical datasets using unsupervised decision trees," *Knowl.-Based Syst.*, vol. 82, pp. 70–79, Jul. 2015.

[47] M. García-Borroto, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, M. A. Medina-Pérez, and J. Ruiz-Shulcloper, "LCMine: An efficient algorithm for mining discriminative regularities and its application in supervised classification," *Pattern Recognit.*, vol. 43, no. 9, pp. 3025–3034, Sep. 2010.

[48] M. García-Borroto, J. F. Martínez-Trinidad, and J. A. Carrasco-Ochoa, "Finding the best diversity generation procedures for mining contrast patterns," *Expert Syst. Appl.*, vol. 42, no. 11, pp. 4859–4866, Jul. 2015.

[49] J. R. Quinlan, *Learning Efficient Classification Procedures and Their Application to Chess and Games* (Machine Learning: An Artificial Intelligence Approach). Palo Alto, CA, USA: Springer-Verlag, 1983.

[50] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, 1988.

[51] H.-S. Yoon, S.-Y. Ahn, S.-H. Lee, S.-B. Cho, and J. H. Kim, "Heterogeneous clustering ensemble method for combining different cluster results," in *Data Mining for Biomedical Applications*, J. Li, Q. Yang, and A.-H. Tan, Eds. Berlin, Germany: Springer Berlin Heidelberg, 2006, pp. 82–92.

[52] A. L. N. Fred and A. K. Jain, "Data clustering using evidence accumulation," in *Proc. Object Recognit. Supported Interact. Service Robots*, vol. 4, Aug. 2002, pp. 276–280.

[53] Y. Yang and K. Chen, "Temporal data clustering via weighted clustering ensemble with different representations," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 2, pp. 307–320, Feb. 2011.

[54] D. H. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Mach. Learn.*, vol. 2, no. 2, pp. 139–172, Sep. 1987.

[55] J. H. Gennari, P. Langley, and D. Fisher, "Models of incremental concept formation," *Artif. Intell.*, vol. 40, nos. 1–3, pp. 11–61, Sep. 1989.

[56] R. J. H. Blockeel and R. L. De, "Top-down induction of clustering trees," in *Proc. 15th Int. Conf. Mach. Learn.*, 1998, pp. 55–63.

[57] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[58] T. Shi and S. Horvath, "Unsupervised learning with random forest predictors," *J. Comput. Graph. Statist.*, vol. 15, no. 1, pp. 118–138, Mar. 2006, doi: 10.1198/106186006X94072.

[59] F. Kruber, J. Wurst, and M. Botsch, "An unsupervised random forest clustering technique for automatic traffic scenario categorization," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2811–2818.

[60] M. Madhyastha, P. Li, J. Browne, V. Strnadova-Neeley, C. E. Priebe, R. Burns, and J. T. Vogelstein, "Geodesic learning via unsupervised decision forests," 2019, *arXiv:1907.02844*. [Online]. Available: https://arxiv.org/abs/1907.02844

[61] N. Fore and G. Dong, "CPC: A contrast pattern based clustering algorithm," in *Contrast Data Mining: Concepts, Algorithms, and Applications* (Data Mining and Knowledge Discovery Series), G. Dong and J. Bailey, Eds. Boca Raton, FL, USA: CRC Press, 2012, ch. 14, pp. 197–216.

[62] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987.

[63] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in PyThon," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[64] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.

[65] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Inf. Retr.*, vol. 12, no. 4, pp. 461–486, 2009.

[66] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[67] S. García and F. Herrera, "An extension on," statistical comparisons of classifiers over multiple data sets'' for all pairwise comparisons," *J. Mach. Learn. Res.*, vol. 9, pp. 2677–2694, Dec. 2008.

[68] M. W. Clark, "Some methods for statistical analysis of multimodal distributions and their application to grain-size data," *J. Int. Assoc. Math. Geol.*, vol. 8, no. 3, pp. 267–282, Jun. 1976, doi: 10.1007/BF01029273.

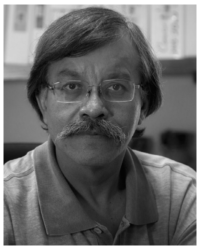
OCTAVIO LOYOLA-GONZÁLEZ received the B.Eng. degree in informatics engineering and the M.Sc. degree in applied informatics from the University of Ciego de Ávila, in 2010 and 2012, respectively, the Ph.D. degree in computer science from the National Institute for Astrophysics, Optics, and Electronics, Mexico, in 2017. He is currently a Researcher and a Professor with the Tecnologico de Monterrey, Campus Puebla, for undergraduate and graduate programs of computer sciences. He has been involved in many research projects about pattern recognition, which have been applied to cybersecurity, biotechnology, and dactyloscopy problems. He has published several books and articles on subjects related to contrast pattern-based classification, data mining, one-class classification, masquerader detection, fingerprint recognition, and cybersecurity. He is also a member of the Mexican Researchers System (Rank one). In 2018, he received the Best Thesis Award José Negrete for the Doctoral Thesis Category on Artificial Intelligence sponsored by the Mexican Society for Artificial Intelligence (SMIA). He was a Prizewinner of the XXXI National Contest of Computer Science Thesis (ANIEI 2018), Prizewinner to the Best Ph.D. Thesis in the Computer Science Coordination at the National Institute of Astrophysics, Optics, and Electronics, in 2018.

ANDRES EDUARDO GUTIERREZ-RODRÍGUEZ received the Ph.D. degree in computer sciences from the National Institute of Astrophysics Optics and Electronics (INAOE), Puebla, Mexico, in 2015. He is currently a Full Professor with the Tecnologico de Monterrey, Campus Toluca. His research areas are data mining, machine learning, clustering, optimization, and fingerprint recognition. He has published several research articles in international journals and conferences.

MIGUEL ANGEL MEDINA-PÉREZ received the Ph.D. degree in computer science from the National Institute of Astrophysics, Optics, and Electronics, Mexico, in 2014. He is currently a Research Professor with the Tecnologico de Monterrey, Campus Estado de Mexico, where he is also a member of the GIEE-ML (Machine Learning) Research Group. He has rank one in the Mexican Research Systems. He has published ten articles in referenced journals such as the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, *Pattern Recognition*, *Information Fusion*, *Knowledge-Based Systems*, *Information Sciences*, and *Neurocomputing*. He has an extensive experience developing software to solve pattern recognition problems. A successful example is a fingerprint and palm print recognition framework which has more than 1.2 million visits and 132 thousand downloads. His research interests include pattern recognition, data visualization, explainable artificial intelligence, fingerprint recognition, and palm print recognition.

RAÚL MONROY received the Ph.D. degree in artificial intelligence from Edinburgh University, in 1998, under the supervision of Prof. Alan Bundy. He is currently a Professor of computing with the Tecnologico de Monterrey. Since 1998, he has been a member of CONACyT's National Research System, currently rank three. He is also the Leader of the GIEE-ML (Machine Learning) Research Group, Tecnologico de Monterrey, and also the Head of the graduate programme in computing, at region CDMX. His research is concerned with the discovery and application of novel model machine learning models, which he often applies to cybersecurity problems.

JOSÉ FRANCISCO MARTÍNEZ-TRINIDAD received the B.S. degree in computer science from the Physics and Mathematics School, Autonomous University of Puebla (BUAP), Mexico, in 1995, the M.Sc. degree in computer science from the Faculty of Computers Science, Autonomous University of Puebla, Mexico, in 1997, and the Ph.D. degree from the Center for Computing Research, National Polytechnic Institute (CIC, IPN), Mexico, in 2000. He edited/authored seven books and more than 120 journal and conference papers on subjects related to *Pattern Recognition*.



JESÚS ARIEL CARRASCO-OCHOA received the Ph.D. degree in computer science from the Center for Computing Research, National Polytechnic Institute (CIC-IPN), Mexico, in 2001. He works as a full time Researcher with the Instituto Nacional de Astrofísica, Óptica y Electrónica. He has published more than 100 articles on topics related to *Pattern Recognition* and *Data Mining*, and co-edited seven books. His current research interests include logical combinatorial pattern recognition, data mining, testor theory, feature and prototype selection, text analysis, and fast nearest neighbor classifiers and clustering.

MILTON GARCÍA-BORROTO received the degree from Las Villas Central University, Cuba, in 2000, and the M.Sc. and Ph.D. degrees from the National Institute of Astrophysics, Optics and Electronics, Mexico, in 2007 and 2010, respectively. He is currently a Professor with the Instituto Superior Politécnico José Antonio Echeverría, La Habana, Cuba. His research interests are pattern recognition, intelligent systems, machine learning, and biometrics.