Clustering is an unsupervised learning technique used in data mining for finding groups with increased object similarity within but not between them. However, the absence of a-priori knowledge on the optimal clustering criterion, and the strong bias of traditional algorithms towards clusters with a specific shape, size, or density, raise the need for more flexible solutions to find the underlying structures of the data. As a solution, clustering has been modeled as an optimization problem using meta-heuristics for generating a search space to favor groups of any desired criterion. F1-ECAC is an evolutionary clustering algorithm with an objective function designed as a supervised learning problem, which evaluates the quality of a partition in terms of its generalization degree, or its capability to train an ensemble of classifiers. This algorithm is named after its previous version, ECAC (Evolutionary Clustering Algorithm Using Supervised Classifiers), considering its main point of difference, which is the inclusion of the F1-score instead of the Area Under the Curve metric in the objective function. F1-ECAC shows a significant increase in performance and efficiency to ECAC and is highly competitive to state-of-the-art clustering algorithms. The results demonstrate F1-ECAC’s benefits in usability in a wide variety of problems due to its innovative clustering criterion.

INDEX TERMS
Unsupervised learning, clustering, evolutionary clustering, optimization, classifier ensembles.

I. INTRODUCTION
Data are generated at unprecedented speeds and quantities across multiple industries and the processes within them [1]. Devices ranging from machinery in a plant to the intelligent devices in our pockets are now instrumented for data collection and transmission. This transition has reached our approach to personal computing as well as the operations, manufacturing, supply chain, and marketing sectors of productive systems. This amount of available data implies the inherent challenge of data science methods for extracting insights to transform data into knowledge and decisions [2].

Clustering is an unsupervised learning technique for exploratory data analysis [3], [4]. The main objective of clustering is to find patterns between data features for creating disjoint subsets of the data called clusters [5], [6]. Even though clustering generally seeks to form compact and isolated groups, the lack of a standard cluster definition has given rise to multiple methods over time [7], [8].

Clustering algorithms group unlabeled objects into clusters that are similar according to their own criterion [1], [9], [10]. Clustering criterion is the model that will represent the data’s structure and is associated with an algorithm [11]. For instance, partitional, hierarchical, and density-based methods aim to take care of possible cluster shape, size, and noise [1], [7], [12]. Currently, we have access to well-maintained libraries containing multiple clustering algorithms, such as Google’s TensorFlow [13], Facebook’s PyTorch [14], or Scikit-learn [15].

Selecting the most suitable clustering criterion is usually a task that relies on previous experience or domain-dependent knowledge [8]. Each algorithm will offer distinct representations of the data under their own bias, and they are sensitive to the hyper-parameters used [12]. In this context, we refer to clustering bias as the positive increase in performance when specific data conditions are met due to the tendency of an
algorithm to force a cluster structure to the data. This manual
criterion selection usually consists of a posterior evaluation
of the results to determine their usability, and there is no
clustering algorithm or criterion capable of capturing all the
underlying structures in a dataset [8]. Hence, an algorithm
that generates and evaluates adaptive solutions along the
clustering process is needed.

Clustering can be modeled as an optimization problem
to favor any desired similarity metric, and the selection
of the objective function to be maximized, or minimized,
will determine the characteristics of its resultant groups.
Nature-inspired meta-heuristics have been implemented for
generating a search space to optimize cluster compactness,
connectivity or spatial separation. Among them, evolutionary
algorithms are the most used for clustering [12].

In this paper, we present F1-ECAC, an enhanced single-
objective evolutionary clustering algorithm using classifiers
capable of flexibly representing the cluster structures and
adapting to the cluster structure of the data. Our proposal fol-
low the philosophy behind the Cluster Validity Index using
Classifiers (VIC) by Rodriguez et al. [1], which states that a
high-quality partition should induce a high-performing clas-

cifier. The main objective of this development is to tackle the
problem of clustering criterion selection and clustering bias.
This is achieved by introducing an ensemble of classifiers for
cluster assessment in the objective function of an evolutionary
clustering algorithm, which aims to evaluate a partition in
terms of its generalization capability. F1-ECAC returns high-
quality clusters that do not follow any geometrical shape,
and offers significant improvements in performance and effi-
ciency to its previous version, the Evolutionary Clustering
Algorithm using Supervised Classifiers (ECAC), presented in
[16].

The rest of the paper is organized as follows. Section II
describes multiple approaches to clustering, along with their
advantages and disadvantages. F1-ECAC is presented in
detail in Section III. In Section IV, the experimental frame-
work is described. Sections V and VI point out the results,
insights, and findings of this work, and finally, our conclu-
sions are presented in Section VII.

II. RELATED WORK

This section introduces state-of-the-art clustering algorithms
and their role in the literature. Their advantages and disad-
advantages are presented along with essential concepts related
to the problem to be solved.

A. TRADITIONAL CLUSTERING ALGORITHMS

Unlike common taxonomies of clustering algorithms found
in the literature [17], Handl and Knowles [12] propose a
categorization based on the clustering criterion optimized by
each algorithm. The three groups are:

1) Algorithms favoring compact clusters, which aim to
keep a small variation within clusters.

2) Algorithms favoring connectedness between objects
inside a group, clustering neighbors together.

3) Objectives based on spatial separation, which have the
disadvantage of easily returning trivial solutions, and
are often combined with other objectives.

In the following subsections, we will go through some of
the most representative clustering algorithms of the first two
mentioned families. The spatial separation objectives have
been implemented as cluster quality indexes [12], [18] but
are not a commonly found heuristic in clustering algorithms.
The following algorithms are also used in the performance
comparison of this work’s proposal.

1) k-MEANS

This partitional clustering method [19] is arguably the most
popular since its introduction in the 1960s and has seen
multiple variants in heuristics and implementations [20], [21].
The objective of the k-means algorithm is to minimize the
sum of squared distances of each point to its cluster center
by searching for a locally minimal solution in an iterative
scheme [20]. As k-means was designed to create compact
groups, it tends to favor clusters forming hyper-spheres and
higher k values (i.e. number of clusters). This algorithm is
unsuitable when dealing with complex cluster structures and
is sensitive towards the initial centroids.

2) SINGLE-LINKAGE AGGLOMERATIVE CLUSTERING

Hierarchical clustering methods generate a dendrogram for
determining nested clusters according to a similarity met-
ic [22]. A dendrogram structure can be broken at any desired
level for generating any number of groups [17]. Single-link-
age clustering uses the distance between the closest points
between clusters as the similarity metric, favoring their
connectedness [3].

3) DBSCAN

Clustering methods favoring connectedness can handle arbi-

tary cluster shapes but commonly struggle to detect spatial
separation between groups. Density-based Spatial Clustering
of Applications with Noise (DBSCAN) [23], [24], takes care of
this drawback and is arguably the most representative
in this category. In contrast to other traditional clustering
algorithms, DBSCAN does not require the number of groups
as a parameter as it generates clusters only if an object’s
neighborhood density complies with a minimum threshold
and a minimum number of objects. DBSCAN is an efficient
clustering algorithm scalable to large datasets and offers the
convenient approach of requiring only one parameter to be
established by the user [23].

B. SINGLE-OBJECTIVE CLUSTERING

The absence of expert knowledge for evaluating clustering
quality can be assessed with an internal criterion towards
distinct cluster definitions by modeling clustering as an opti-

mization problem (Ω, P) to find C∗ for which

\[ P(C^*) = \min_{C \in \Omega} P(C), \]

(1)
where $\Omega$ is the set of possible clusterings, $C$ is a partition of data $E$, and $P$ is an internal criterion usually based on some notion of similarity [12], [25]. Evolutionary clustering algorithms have been implemented to return solutions following innovative objective functions, operators, and representations [26]–[29].

1) HG-MEANS
Gribel and Vidal [30] presented a hybrid single-objective clustering algorithm that incorporates the exploration methodology of the genetic algorithm. HG-means applies variation operators to a random initial population such as selection, recombination, and mutation for minimizing the sum of squares cost. This objective function is the same as $k$-means, and it has been applied in multiple meta-heuristic approaches to clustering [26], [31]. The compactness of a partition is computed as

$$\text{Comp}(C) = \sum_{C_i \in C} \sum_{i \in C_i} \delta(i, \mu_k)^2,$$  \hspace{1cm} (2)

where $\delta(., .)$ represents the dissimilarity function to use (in this case, Euclidean metric), and $i$ is a point in data $E$, with centroids $\mu_k$ for each cluster $k$.

The algorithm’s crossover operator starts by selecting two parents through a binary tournament and solves a bipartite matching problem for finding pairs of clusters for randomly selecting between them with equal probability. Afterwards, the mutation operator randomly removes and relocates one of the centroids. Finally, the centroid set is used for a local search based on $k$-means at the end of each generation. HG-means introduces diversity management strategies for avoiding clones and eliminating the worst-performing individuals when exceeding the user-defined maximum population size parameter.

C. MULTI-OBJECTIVE CLUSTERING

Even though evolutionary clustering algorithms might avoid falling into locally minimal solutions, most existing clustering methods optimize just one criterion, either explicitly or implicitly. Hence an inadequate selection of clustering criterion selection would still lead to failure in displaying acceptable results. One way to overcome this problem is by considering the compromise of multiple complementary cluster characteristics, which is a more natural data condition. A multi-objective clustering problem ($\Omega$, $P_1$, $P_2$, $P_m$) searches for the optimal clustering $C^*$ for which

$$P_i(C^*) = \min_{C \in \Omega} P_i(C) \text{ for } t = 1, \ldots, j,$$  \hspace{1cm} (3)

where $\Omega$ is the set of reachable clusterings, $C$ is a clustering of data $E$, and $P_i$ are $j$ multiple clustering criteria [12], [25]. The framework of Pareto optimality is introduced to balance the fitness between separate objective functions. For instance, given solutions $C_1$, $C_2 \in \Omega$, solution $C_1$ dominates solution $C_2$ ($C_1 \prec C_2$) if and only if

$$\forall t : P_t(C_1) \leq P_t(C_2) \land \exists t : P_t(C_1) < P_t(C_2),$$  \hspace{1cm} (4)

and the Pareto-optimal solutions can be defined as

$$\Pi = \{C \in \Omega : \not\exists C' \in \Omega : C' \prec C\},$$  \hspace{1cm} (5)

which generates an objective space denominated as the Pareto front. It is expected that multi-objective algorithms find solutions that are at least equally good to single-objective algorithms in exchange for an increased computational cost [12], [25]. Multi-objective evolutionary clustering algorithms have been designed to optimize two objectives, mainly, and this novel clustering sub-discipline has witnessed the creation of algorithms varying in efficiency and performance in recent years [32]–[35]. In the following subsections, we will focus on two of the most representative algorithms in this category.

1) MOCLE
Faceli et al. [8], [36] proposed the Multi-objective Clustering Ensemble method (MOCLE), a Pareto-based evolutionary approach to clustering. MOCLE starts by creating an initial population with conceptually diverse clustering algorithms using varying hyper-parameters and cluster numbers in the range of $k$ to $2 \ast k$. This algorithm encodes each individual, or partition, as an array of sets, each containing the label of its members. MOCLE returns a set of solutions representing each region of the Pareto front obtained from the optimization of two complementary objective functions to be minimized:

1) Deviation also aims to minimize cluster compactness and is computed as

$$\text{Dev}(C) = \sum_{C_i \in C} \sum_{i \in C_i} \delta(i, \mu_k),$$  \hspace{1cm} (6)

similarly to (2), but without the squared distance value [12].

2) Connectivity calculates the frequency of neighbor data points to be clustered together and is computed as

$$\text{Connt}(C) = \sum_{i \in E} \sum_{l=1}^{L} x_{i,mn_l}$$

where $nn_l$ is the $l_{th}$ nearest neighbor of point $i$, and $L$ is the parameter for delimiting the number of neighbors, and $E$ represents the data to be clustered [12].

This algorithm differs from other methods mainly due to its crossover operator, which uses an ensemble technique for finding a consensus partition of two parents selected through a binary tournament. The number of clusters of the resulting child is randomly chosen from the cluster interval of the parents. The algorithm aims to favor high-quality partitions and gradually eliminate low-performing ones along the evolutionary process. The mutation operator is omitted in this algorithm to restrict search space to the selection and recombination of partitions, assuming that the relevant structures are contained inside the initial population.
2) Δ-MOCK
Garza-Fabre et al. [6] published a modified version of the Multi-objective Clustering with Automatic k-determination (MOCK) method [12], a state-of-the-art multi-objective evolutionary clustering algorithm. The updated algorithm Δ-MOCK includes changes to the representation and initialization schemes showing scalability improvements. Along the evolutionary process, mating selection through a binary tournament and genetic operators give rise to new offspring which compete against each other and their parents to survive. As stated by the authors, the algorithm optimizes the trade-off of two ‘fundamentally different but equally desirable’ objective functions to be minimized:

1) Connectivity within clusters, presented in (7).
2) Intra-cluster variance, which can be computed as

\[
\text{Var}(C) = \frac{1}{N} \sum_{C_i \in C} \sum_{i \in C_i} \delta(i, \mu_k)^2, \quad (8)
\]

where \(N\) represents the number of objects \(p_i\) of data \(E\) to be clustered.

Δ-MOCK uses NSGA-II [37] as its search engine and returns a Pareto front with balanced representation across its regions. The initialization process produces a population with multiple cluster numbers, delimited by the \(k_{max}\) hyper-parameter, which the authors suggest as \(k_{max} = 2 \times k^*\), where \(k^*\) is the estimated actual number of clusters. The creation of a Minimum-spanning Tree (MST) forms the basis for the genetic material to be explored in the evolutionary process. This algorithm employs a uniform crossover operator [38] for recombining two parents. Two offspring are started as an exact copy of their parents, and their genetic material is either preserved or interchanged conditioned by the hyper-parameter \(p_c\). This operator promotes the exploration of any combination of the selected parents. Δ-MOCK incorporates the neighborhood-biased operator that individually evaluates each gene, or link, with a mutation probability \(p_m\).

D. SUPERVISED AND UNSUPERVISED LEARNING
In the following subsections we will describe some approaches to classification and cluster quality assessment that take advantage of both supervised and unsupervised learning techniques.

1) OPTIMIZED ENSEMBLE CLASSIFIER WITH CLUSTER SIZE REDUCTION
Jan et al. [39] developed a classifier ensemble with similar components to this proposal, including clustering, an ensemble of classifiers and evolutionary optimization. The main focus of their work is targeted towards the implementation of incremental clustering for generating varied class-pure clusters to handle class imbalance. Their proposal uses partitions of the data for training and selecting a set of classifiers to get an optimized ensemble using an evolutionary searching schema and the accuracy metric. Even though Jan’s proposal aims to tackle a classification problem, whereas we are targeted towards a clustering problem, the benefits of using supervised and unsupervised learning together have been studied before and have induced the development of innovative proposals.

2) CLUSTER VALIDITY INDEX USING CLASSIFIERS
The foundation behind the clustering criterion in our algorithm’s objective function relies upon the Validity Index Using Classifiers (VIC) [1], which implies that the performance of a supervised classifier has a positive correlation with the quality of a partition. Therefore, this clustering index evaluates a solution in its capability to induce a good classification model. This index takes a partition as input to train and test the ensemble of classifiers in multiple folds. The returned numerical value is computed according to the classifiers’ maximum average Area Under the Curve (AUC). The authors suggest not using accuracy as a performance measure as it can be misleading when class imbalance is present. We did not implement this index directly in the current development due to its computational cost, but its concepts are essential for this work.

E. REMARKS
Even though the previously mentioned evolutionary clustering algorithms explore the search space with complementary objectives, they are still conditioned towards predefined cluster shapes, which is a limiting factor to be addressed by our proposal using the clustering criterion mentioned before. The difference in the algorithms’ performance will be revisited in Section VI, where their multiple data interpretations will be discussed.

III. F1-ECAC
In this section, we will go into detail about the framework followed by F1-ECAC. Our algorithm integrates the advantages of evolutionary clustering mentioned in Section II, with a particular focus on performance and efficiency. Even with more complex search strategies, solutions will not solve the difficulties caused by the bias of traditional algorithms if the objectives of an evolutionary approach benefit the same structures. Therefore, we designed F1-ECAC as a single-objective problem with an innovative objective function to overcome the absence of a-priori knowledge on the optimal clustering criterion. The fitness calculation of an individual must be capable of detecting the underlying structures of the data, which supervised classifiers are capable of doing. The benefits of using classifiers as cluster validity index were studied in [1], and their inclusion in the objective function in evolutionary clustering was proposed for the first time in our previous work [16]. A clustering algorithm should adapt to the data and not the other way around, which F1-ECAC accomplishes by optimizing the generalization capability of clustering solutions.

To summarize, F1-ECAC is a single-objective evolutionary clustering algorithm that uses the capability of a partition to train an ensemble of classifiers as a clustering
Algorithm 1 F1-ECAC Algorithm
1: function F1-ECAC(E, Π, g, k)
2:    $\triangleright$ $E$: data; $\Pi$: population size; $g$: maximum generations; $k$: number of clusters.
3:    $\Phi \leftarrow i_{1}, \ldots, i_{\Pi}$ $\triangleright$ Generation of initial population $\Phi$ with $k$ clusters.
4:    $X \leftarrow$ data $E$ features.
5:    Compute the fitness of each individual in $\Phi$ using OBJ($X$, $\Phi$). $\triangleright$ See Algorithm 2.
6:    for $u = 1, g$ do
7:        $selected \leftarrow \Pi$ individuals are selected from $\Phi$ through a binary tournament.
8:        $parents \leftarrow selected$ elements form $\Pi$ pairs.
9:        $C_1, \ldots, C_\Pi \leftarrow$ crossover($parents$).
10:       $C'_1, \ldots, C'_\Pi \leftarrow$ mutation($C_1, \ldots, C_\Pi$).
11:       $\Phi' \leftarrow C'_1, \ldots, C'_\Pi$.
12:       Compute $\Phi'$s fitness with OBJ($X$, $\Phi'$).
13:       $\Phi \leftarrow \Phi'$. $\triangleright$ New population.
14:       $b \leftarrow$ all-time best solution.
15:       if $b$'s fitness $=$ 1 then
16:           break
17:       end if
18:    end for
19:    return $b$

Algorithm 2 Objective Function Algorithm
1: function OBJ($X$, $i$)
2:    $\triangleright$ $X$: data $E$ features; $i$: an individual’s genotype.
3:    $X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}} \leftarrow$ train-test split of $X$ and $i$.
4:    $\Psi_i$ classifier training with $X_{\text{train}}$, using $i_{\text{train}}$ as target labels.
5:    $F_i \leftarrow$ $F_i$ score (macro-average) of $\Psi_i$’s predictions using $X_{\text{test}}$ against $i_{\text{test}}$.
6:    if $\Psi_i$ could not converge then
7:        $F_i \leftarrow 0$
8:    end if
9: end for
10:    $fitness \leftarrow \text{mean}(F_1, \ldots, F_j)$
11:    return $fitness$

FIGURE 1. Genotype of a partition with twelve objects clustered in three groups.

A. SOLUTION REPRESENTATION AND INITIAL POPULATION
F1-ECAC employs an integer encoding with a label-based representation for one individual or clustering solution [26]. A genotype is a vector with $N$ positions, where $N$ is the number of objects in dataset $E$. The $i$th position (or gene) of the structure represents the corresponding cluster of the $i$th object from $E$. Each gene contains a cluster number in the range $C_k \in \{1, \ldots, k\}$. This representation is widely used in the literature [27], [28], [30], even though the vector’s size increases as a function of $N$ and may cause repetition in the search space by the $k!$ possible genotypes for the same solution. This representation was selected hand in hand with the proposed fitness function, as the classifiers must receive as target labels one vector that fully represents a solution, making it the most suitable for this specific application.

Fig. 1 displays a genotype for a dataset containing 12 objects clustered in three groups. Genes $q_1, \ldots, q_N$ have values on the cluster $C_k$ to which each object is assigned. F1-ECAC is a partitional clustering method, so each object can only be assigned to one cluster, and the partitions are mutually exclusive. The initial population of our algorithm is generated randomly, using the uniform distribution for group assignment, conditioned by the user-specified fixed $k$ value.

B. OBJECTIVE FUNCTION
F1-ECAC’s main differentiating point is its objective function, which uses the principle of generalization as a clustering quality criterion by including an ensemble of classifiers instead of a distance metric, as commonly proposed. Fitness values are computed as presented in Algorithm 2. Fig. 2 depicts the pipeline for computing the fitness value of the genotype shown in Fig. 1. A single-objective approach was selected to balance performance and efficiency, and the results in Section V depict shorter computation times whereas increasing the algorithm’s clustering performance. F1-ECAC’s objective function uses data $E$’s normalized features $X$ and an individual’s genotype $i$ as target labels for training and testing. It is important to remark that the target labels to train and test the classifiers are the genotypes generated along the evolutionary process, which represent a solution, or partition.

Classifiers constitute the ensemble in the objective function from distinct families to reduce bias, as suggested in [1]. The selection of classifiers is highly influential for the algorithm’s performance and numerical and categorical data are supported. Our multi-expert approach to computing fitness uses three classifiers for training, and each of them generates predictions for testing. The classifiers’ performance is used to assess cluster quality, and their outputs are equally weighted.
The classifiers in F1-ECAC’s objective function are:

1) **Support Vector Machines** (SVMs) aim to maximize the margin in the hyper-plane determined by the support vectors for separating classes. SVMs compute the structure and parameters of the model simultaneously, returning a globally optimal solution by mapping the input space into a higher-dimensional feature space with the desired kernel function [40], [41].

2) **k-Nearest Neighbors** is a similarity-based classification method relying on a distance metric and a number of neighbors $k$ for classifying previously unseen data. This method performs best as more data is available and executes queries without actually building a model. It offers a straightforward parameter setup and generates non-linear decision boundaries [42].

3) **Decision Trees** are introduced for this work as the third classification algorithm. This predictive model determines the class of a new data point starting at the root node and descending through the interior nodes based on the feature vector until reaching a terminal node and making a decision according to some quality criterion. Decision trees are arguably the most intelligible machine learning model [2], [43].

The performance metric for evaluating each classifier’s trained model is the $F_1$ score (the harmonic mean of precision and recall). This metric will return a value for measuring generalization between 0 and 1. The $F_1$ score is computed as

$$
\text{precision} = \frac{tp}{tp + fp},
$$

$$
\text{recall} = \frac{tp}{tp + fn},
$$

$$
F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}},
$$

where $\text{precision}$ and $\text{recall}$ (or sensitivity) are employed to consider the influence of correctly classified positive examples ($tp$), misclassified negative examples ($fn$), and examples misclassified as positives ($fp$) [44]. The *macro* score is used for multi-class classification where $k > 2$, computing an unweighted average of each classifier’s $F_1$ score per class. The fitness value returned by the function is defined as the unweighted mean of the classifiers’ *macro* $F_1$ score.

### C. GENETIC OPERATORS

1) **STANDARD ONE-POINT CROSSOVER**

The standard one-point crossover operator proceeds according to a crossover probability $p_c$ for exchanging the genetic material of two parents delimited by a random crossover point. Two parents are selected through a binary tournament and paired according to their position in the data structure for the reproductive process. If $p_c$ determines that the operator will not proceed, $parent_1$ is returned immediately. In the opposite case, the crossover continues, and two new offspring are created from slicing their parents, as shown in Fig. 3, which illustrates the crossover operator using the genotype presented in Fig. 1 and an additional individual as the second parent. The process is repeated twice per pair of parents for merging them in the two possible combinations. As mentioned in Section II, this is an operator widely found in the literature, applied with variations in state-of-the-art evolutionary clustering algorithms.

This operator guarantees to keep at least one member per cluster after the recombination process for

1) avoiding invalid solutions, and
2) to handle the context insensitivity problems caused by the selected genotype representation and crossover operators mentioned by Hruschka [26]and Falkenauer [45].

2) **PROPORTIONAL MUTATION**

The proportional mutation operator is executed as indicated by the mutation probability $p_m$. If the function proceeds, the resulting child with the exchanged genetic information of the parents from the crossover operator continues to mutate
D. ECAC AND F1-ECAC

Table 1 demarcates the modifications made to F1-ECAC’s predecessor, ECAC, reported in [16]. The differentiation points are outlined next.

- Logistic Regression [2], [47] was substituted by Decision Trees due to the former’s computational cost, which is amplified by the multiple fitness calculations in a one-vs-rest schema.
- A multi-class schema was established to avoid training each classifier \( k \) times and only training them once. By doing this, we benefit from the efficient implementations offered in the selected library (more details in Section IV).
- Using the \( F_1 \) score for evaluating the classifiers is way more efficient, as it does not require the computation of binarized labels, prediction probabilities, and the AUC-ROC curve, thus reducing complexity whereas still exploiting the obtained confusion matrices. This is the main design change, and we decided to name F1-ECAC after it, following our previous algorithm’s naming convention.
- The mutation operator was extended to 5% of the genes in a genotype to keep an important effect in the search even with large datasets.

The main focus on F1-ECAC’s updates relies upon its objective function, as we ascribe most of the algorithm’s performance to its capability of discerning partition quality when computing fitness values. All of the non-mentioned phases of ECAC were kept the same, and we refer the reader to [16] for further detail on F1-ECAC’s previous version.

### IV. EXPERIMENTAL SETUP

This section specifies the inputs, hyper-parameters, and computational experiments. The algorithms described in Section II were tested and used for comparing the performance of F1-ECAC. The experiments were held in a dual-core Intel Core i5 2.7 GHz processor with 8 GB of RAM.

#### A. DATASETS

For experimentation, we selected 20 numerical datasets from two sources, the UCI Machine Learning Repository [48] and Fränti and Sieranoja’s Clustering Benchmark repository [49]–[53]. The class labels of all the datasets are only used as a ground truth reference for evaluating the resulting partitions and are not involved in the clustering process. Table 2 shows the composition information of the datasets, which range from 2 to 36 classes, 2 to 34 features, and 101 to 788 objects.

Fig. 5 displays the 2-D synthetic datasets used in the experiments and their ground truth clusters. Using this type of data helps us understand the behavior of clustering algorithms, but we are aware of how clustering algorithms process different real-world data. Therefore, we decided to use only five synthetic datasets for the analysis. All of the remaining visualizations in the document were computed using the Seaborn library [54] in the Python programming language [55].

#### B. HYPER-PARAMETER CALIBRATION

F1-ECAC was tested against seven contestant methods, including ECAC, using multiple performance indicators and hypothesis testing for assessing statistical significance in the difference between observations. Each algorithm was run 10 times per dataset to avoid their stochastic nature affecting the analysis. The number of clusters was established using each dataset’s specifications for the algorithms that have this requirement, and they were tested with equal parameters for condition equality and fair comparison. The obtained clusterings were compared against its ground truth using the

### TABLE 1. Improvements done to ECAC in the design of F1-ECAC.

| Modification                  | ECAC                                      | F1-ECAC                                |
|------------------------------|-------------------------------------------|----------------------------------------|
| Classifiers                  | Support Vector Machine                    | Support Vector Machine                 |
| \( k \)-Nearest Neighbors    | Logistic Regression                        | \( k \)-Nearest Neighbors               |
| \( k \)-Nearest Neighbors    | Decision Trees                             | Decision Trees                         |
| Training schema              | one-vs-rest                                | multi-class                             |
| Evaluation metric            | AUC                                        | \( F_1 \) score                         |
| Label binarization           | \( \checkmark \)                            | -                                      |
| Mutation operator            | 1-point                                    | Proportional: 5%                        |
### TABLE 2. Datasets used for experimentation, and their classes, features, and objects.

| Dataset            | Classes | Features | Objects |
|--------------------|---------|----------|---------|
| aggregation        | 7       | 2        | 788     |
| breast-cancer-wisconsin | 2      | 30       | 569     |
| breast-tissue      | 6       | 9        | 106     |
| dermatology        | 6       | 34       | 366     |
| ecoli              | 8       | 7        | 336     |
| forest             | 4       | 27       | 523     |
| glass              | 6       | 9        | 214     |
| iris               | 3       | 4        | 150     |
| Jain               | 2       | 2        | 373     |
| leaf               | 36      | 14       | 340     |
| liver              | 16      | 5        | 345     |
| parkinsons         | 2       | 22       | 195     |
| pathbased          | 3       | 2        | 300     |
| r15                | 15      | 2        | 600     |
| seeds              | 3       | 7        | 210     |
| segment            | 7       | 19       | 210     |
| spiral             | 3       | 2        | 312     |
| transfusion        | 2       | 4        | 748     |
| wine               | 3       | 13       | 178     |
| zoo                | 7       | 16       | 101     |

### FIGURE 5. Plots of the synthetic data used in our benchmark.

Adjusted RAND Index (ARI) [56]. Even though most real applications do not count on a reference partition, this index is considered an objective external measure of cluster quality to assess the similarity between two partitions [12].

1) **F1-ECAC**

To evaluate our decision to include the three classifiers in the objective function as an ensemble, we ran the evolutionary process using each classifier separately. Therefore, we generated solutions based only on the fitness returned by one classifier. This test lets us conclude whether our ensemble is the best alternative for the objective function or if one classifier could be enough.

2) **F1-ECAC AND ECAC**

F1-ECAC was run using 200 individuals per population for 200 generations. Hence 120 thousand classifiers are trained along the evolutionary process. The operator probabilities were established as $p_c = 0.95$ and $p_m = 0.98$, which implies a 95% probability of recombining two parents and a 98% probability of mutating 5% of the genes from a genotype. The classifiers in F1-ECAC’s objective function and the contest methods belonging to the traditional algorithm family were coded using the Scikit-learn library [15] along with the Pandas library [57], [58] for data manipulation. These were the parameters used in F1-ECAC’s classifier ensemble for fitness calculation.

- **Support Vector Machine**: linear kernel function, $L_2$ regularization parameter value of 1, and unlimited iterations.
- **k-Nearest Neighbors**: 5 nearest neighbors inversely weighted by their distance, and Euclidean metric.
- **Decision Tree**: entropy criterion with unlimited maximum depth.

The fitness computation uses 25% of the data for training and 75% for testing, considering the efficiency of the objective function computation. The crossover point and mutation gene are computed randomly using the uniform distribution. F1-ECAC is available in a Python implementation at https://github.com/benjaminsainz/f1-ecac.

ECAC’s initial population was established at 20 individuals that went through the evolutionary process for 2,000 generations, and the classifiers were set up with the following differences.

- **Support Vector Machine**: maximum iterations set to 5000.
- **k-Nearest Neighbors**: 2 nearest neighbors with uniform weights.
- **Logistic Regression**: maximum iterations set to 100, $L_2$ regularization parameter value of 1.

The rest of the hyper-parameters were set the same as F1-ECAC. We used the non-parametric Wilcoxon test [59] for hypothesis testing for comparing the ARI and runtime in seconds by F1-ECAC and ECAC, considering a significance level of $\alpha = 0.05$ in all the cases. Now that the experimental setup of our proposed method has been stated, we will present the configurations of the reference methods used for comparison.

3) **CONTESTANT METHODS**

In the following bullets, we present the setup specifications for the benchmark methods sorted by family.

- **k-means**: 250 maximum iterations with 10 random initializations.
- **Single-linkage**: Euclidean distance metric.
- **DBSCAN**: minimum samples per clusters set to 7 and $\epsilon$ value of 0.3.
- **HG-means**: 20 individuals per population for 2,000 generations using the author’s implementation [60].
- **MOCLE**: we implemented this algorithm as it is not publicly available as of the date of development of this study. We used the code of NSGA-II from [61]. The algorithm naturally generates variable population sizes in each run, and it was set to 50 maximum generations. We used a label-based representation for ease of implementation, and the initial population...
TABLE 3. Adjusted RAND Index of the solutions returned using one classifier in the objective function and F1-ECAC.

| Dataset           | Support Vector Machine | k-Nearest Neighbors | Decision Tree | F1-ECAC |
|-------------------|------------------------|---------------------|---------------|---------|
| aggregation       | 0.4484                 | 0.1007              | 0.1034        | 0.2685  |
| breast-cancer-wisconsin | 0.0226               | 0.0360              | 0.0019        | 0.0570  |
| breast-tissue     | 0.4248                 | 0.4094              | 0.4857        | 0.5046  |
| dermatology       | 0.0195                 | 0.0235              | 0.0172        | 0.0283  |
| ecoli             | 0.3704                 | 0.2993              | 0.2714        | 0.4506  |
| forest            | 0.0037                 | 0.0065              | 0.0020        | 0.0100  |
| glass             | 0.2699                 | 0.2091              | 0.2392        | 0.2880  |
| iris              | 0.8021                 | 0.7399              | 0.8294        | 0.9169  |
| jain              | 0.2936                 | 0.0829              | 0.0196        | 0.3513  |
| leaf              | 0.4515                 | 0.4030              | 0.4197        | 0.4993  |
| liver             | 0.0939                 | 0.0851              | 0.0857        | 0.0880  |
| parkinsons        | 0.0368                 | 0.0447              | 0.0597        | 0.0419  |
| pathbased         | 0.3657                 | 0.0940              | 0.1577        | 0.2900  |
| r15               | 0.4389                 | 0.4951              | 0.3770        | 0.5478  |
| seeds             | 0.7908                 | 0.6135              | 0.5937        | 0.7763  |
| segment           | 0.6311                 | 0.5386              | 0.5521        | 0.6777  |
| spiral            | -0.0013                | 0.0447              | 0.0575        | 0.0134  |
| transfusion       | 0.0505                 | 0.0252              | 0.0122        | 0.0542  |
| wine              | 0.6457                 | 0.6208              | 0.5935        | 0.7577  |
| zoo               | 0.0856                 | 0.0770              | 0.0846        | 0.1173  |

was constituted using $k$-means, Single and Average linkage Agglomerative Clustering (computing distance between clusters as centroids instead of the closest points between them) [15], and Shared Nearest Neighbors clustering method with diverse parameters [62], [63]. We used 5% of the data’s number of objects as $L$ for computing connectedness. The crossover operator for recombination was performed by Strehl and Ghosh’s Meta-clustering Algorithm [64] using Kultzak’s version [65], avoiding clones. Our implementation of this algorithm is available at https://github.com/benjaminsainz/mocle.

• Δ-MOCK: we used the authors’ implementation [66], population size set to 100 with 100 maximum generations, Δ-locus representation and a maximum number of clusters set to 50.

The data were standardized for all of the algorithms to enhance convergence. As multi-objective algorithms return a Pareto front with multiple partitions, we selected the one with the highest ARI to obtain one solution per run as done by the other method families. The non-parametric Friedman test was implemented to evaluate statistical significance between the ARI returned by the algorithms.

V. EXPERIMENTAL RESULTS

This section points out the results from the experiments held as described in Section IV. We will start by presenting the performance increase by implementing an ensemble method instead of using one classifier in the objective function. Then, we will continue with the difference in solution quality and runtime returned by F1-ECAC and ECAC, to finally move on to a general performance assessment between F1-ECAC and its benchmarking methods.

A. F1-ECAC

Table 3 presents the average Adjusted RAND Index obtained by replacing the classifier ensemble with only one classifier to compare it with F1-ECAC. Our method obtained a higher score than the separate classifiers in this metric in 14 out of the 20 datasets. Remarkably, using one classifier in the
objective function led to better results than using the classifier ensemble in 3 out of the 5 synthetic datasets. This could be attributed to the clear separation and lack of overlapping in clusters with this data type.

For visualizing the convergence of F1-ECAC’s evolutionary process, Fig. 6 displays the overall best solution and the average fitness per generation using the Iris dataset. The final solution obtained a fitness of 0.95 and an ARI of 0.96. Our algorithm reaches convergence around generation 100, however, the process continues for more generations to increase the search space within reasonable runtime. Both plots follow an upwards trend, with the average population’s fitness being lower than the best individuals, which is expected. Nevertheless, once reaching the convergence point, the mean populations’ quality almost matches the quality of the partitions with the highest fitness.

B. F1-ECAC VS. ECAC

Fig. 7 plots the average ARI per dataset obtained by F1-ECAC and ECAC using the logarithmic scale for ease of visualization. F1-ECAC achieved an ARI increase for all cases except for the Parkinson’s dataset with an ARI decrease of 0.023. According to the Wilcoxon test, the observations of both algorithms had a statistically significant difference
TABLE 4. Average ARI of the solutions generated by each algorithm using the synthetic datasets.

| Dataset        | k-means | Single-linkage | DBSCAN | HG-means | MOCLE | Δ-MOCK | F1-ECAC |
|----------------|---------|----------------|--------|----------|-------|--------|---------|
| aggregation    | 0.7272  | 0.8058         | 0.7338 | 0.9991   | 1     | 0.2686 |         |
| jain           | 0.5528  | 0.0099         | 0.9316 | 0        | 0.4893| 1       | 0.3513  |
| pathbased      | 0.4797  | 0.0009         | 0.2255 | 0.4851   | 0.7271| 0.2900  |         |
| r15            | 0.9928  | 0.5425         | 0.2637 | 0.0011   | 0.9928| 0.9928  | 0.5478  |
| spiral         | -0.0058 | 1              | 0.4573 | 0.0438   | 0.6203| 0.0134  |         |

TABLE 5. Performance comparison (average ARI) of the partitions produced by each algorithm from the UCI ML Repository datasets.

| Dataset                  | k-means | Single-linkage | DBSCAN | HG-means | MOCLE | Δ-MOCK | F1-ECAC |
|--------------------------|---------|----------------|--------|----------|-------|--------|---------|
| breast-cancer-wisconsin  | 0.6707  | 0.0048         | 0      | 0.4914   | 0.5415| 0.0048 | 0.0570  |
| breast-tissue            | 0.2781  | 0.0007         | 0      | 0.0053   | 0.1824| 0.0419 | 0.5046  |
| dermatology              | 0.7124  | 0.2759         | 0      | 0.0034   | 0.1717| 0.0060 | 0.0283  |
| ecoli                    | 0.5023  | 0.0399         | 0      | 0.4262   | 0.7446| 0.0283 | 0.4506  |
| forest                   | 0.4325  | 0.0017         | 0      | 0.4987   | 0.5392| 0.0076 | 0.0100  |
| glass                    | 0.1715  | 0.0143         | -0.0076| 0.2598   | 0.2762| 0.2709 | 0.2880  |
| iris                     | 0.6201  | 0.5584         | 0.0641 | 0.7302   | 0.8008| 0.0043 | 0.9169  |
| leaf                     | 0.3565  | 0.0221         | 0      | 0.2950   | 0.2880| 0.3182 | 0.4993  |
| liver                    | 0.0148  | 0.0083         | 0      | 0.0305   | 0.0640| 0.0039 | 0.0880  |
| parkinsons               | -0.0978 | -0.0134        | 0      | 0        | 0.1674| 0.0813 | 0.0419  |
| seeds                    | 0.7733  | 0              | 0      | 0.7166   | 0.7416| 0.7132 | 0.7763  |
| segment                  | 0.4687  | 0.0010         | 0      | 0.4063   | 0.4524| 0.0012 | 0.6777  |
| transfusion              | 0.0527  | -0.0036        | 0.0002 | 0.0795   | 0.0836| 0.0086 | 0.0542  |
| wine                     | 0.8975  | -0.0068        | 0      | 0.3711   | 0.3652| 0.4025 | 0.7577  |
| zoo                      | 0.7581  | 0.4425         | -0.0578| 0.7088   | 0.7561| 0.7670 | 0.1173  |

with P-values below 0.05. The efficiency boost of F1-ECAC is depicted in Fig. 8, plotted using the logarithmic scale. The runtime difference between the algorithms was present in every dataset, and it also was significantly different. Compared to its previous version, F1-ECAC showed an ARI improvement per dataset of 83% and was 7 times faster on average.

C. CLUSTERING PERFORMANCE

The performance analysis of the algorithms is presented divided into subsections for the two types of tested data described in Section IV: 1) Synthetic and 2) Real-world datasets.

1) SYNTHETIC DATA

Synthetic data is helpful to assess the application range of a clustering method and evaluate if it fits our data mining task. However, we often do not count on enough information to make this decision in real applications. Table 4 presents the average ARI of the solutions obtained by each algorithm for each of the synthetic datasets. The experiments where no solution could be generated were left as a blank space.

Δ-MOCK generated high-quality partitions when clustering synthetic data, and it got the highest average ARI in 4 of the 5 datasets. Also, k-means, MOCLE and Δ-MOCK were tied with the highest ARI in the R15 dataset with the same value.

2) REAL-WORLD DATA

We will now proceed to display the results using real-world datasets and focus on these tests to assess how the algorithms

Δ-MOCK generated high-quality partitions when clustering synthetic data, and it got the highest average ARI in 4 of the 5 datasets. Also, k-means, MOCLE and Δ-MOCK were tied with the highest ARI in the R15 dataset with the same value.

We will now proceed to display the results using real-world datasets and focus on these tests to assess how the algorithms

Δ-MOCK generated high-quality partitions when clustering synthetic data, and it got the highest average ARI in 4 of the 5 datasets. Also, k-means, MOCLE and Δ-MOCK were tied with the highest ARI in the R15 dataset with the same value.
perform with more natural data. The average ARI of the solutions returned by each method is summarized in Table 5.

In Fig. 9, we present a bar plot with the number of datasets in which each algorithm surpassed the rest. Only four methods obtained the highest performance against the other six algorithms at least once (i.e. Δ-MOCK, k-means, MOCLE, and F1-ECAC). F1-ECAC shows stable solutions across varying cluster structures with different numbers of features and objects. Our algorithm obtained a higher score than the rest in 7 out of the 15 datasets, followed by MOCLE with 4, k-means with 3, and Δ-MOCK with 1. We attribute Single-linkage absence in this list due to its strong bias towards connectedness. Also, DBSCAN failed to display acceptable cluster structures in most cases due to its hyperparameter setting sensitivity, which tends to be domain-dependent. HG-means was the only evolutionary clustering algorithm that did not score the highest on any dataset and was overcome by the other methods in its family.

The number of datasets is higher than two times the number of methods analyzed. Hence this assumption for the $1 \times N$ Friedman test is valid for the real-world data experiments. The $P$-value computed by the Friedman test was 0 for all experiments. Table 7 shows the unadjusted and adjusted $P$-values of the Friedman test using the Holm Post-Hoc method [59] for each algorithm with respect to F1-ECAC, since it was the highest-ranked treatment as shown in Table 6. Using an $\alpha$ of 0.05, we conclude that the difference between the results of F1-ECAC against DBSCAN, Single-linkage, and Δ-MOCK are statistically significant, according to the $P_{Holm}$ adjusted values. Therefore, HG-means, k-means, and MOCLE do not show a significant difference compared to the results of F1-ECAC.

We performed a correlation analysis to assess the statistical relationship between the number of clusters, instances, and features in the data and the ARI of the solutions returned by each algorithm using real-world data. Fig. 10 plots the Spearman correlation between 10 variables, and we selected a 0.5 (absolute value) threshold to determine high correlations.

Lastly, Fig. 11 displays one partition generated by each of the five algorithms that scored the highest ARI at least in one dataset and its ground truth (considering both synthetic and real-world datasets). We plotted the best-performing dataset per algorithm according to Tables 4 and 5 and selected its highest-scoring solution. We decided to include Ecoli instead of the Zoo dataset in Subfigure (e) to visualize better the partition returned by MOCLE. The outcomes of the experimental phase will be further analyzed in the following section.

VI. DISCUSSION

A. F1-ECAC

The design decisions we followed let us fulfill our goal of enhancing our algorithm’s previous version (i.e. ECAC) in clustering performance and efficiency by modifying its evolutionary process and objective function. The advantages of using a classifier ensemble rather than a single classifier are pointed using Table 3. The results suggest that using three classifiers together led the algorithm to generate better partitions according to their similarity to the reference partition. The multi-expert approach acts similarly to a voting system where we want to find if a partition can induce a well-trained classifier without forcing it to do so.

F1-ECAC is a clustering algorithm with boundaries set in generating a partition from an unlabeled dataset. A fitness of 1 might suggest an overfitting issue, which might induce classifier variance, but it is not relevant to this specific application. It is not inside the scope of this work to train a high-quality
classifying model as we are tackling a clustering problem and not a prediction task. Hence, classifier bias and variance are not an issue because our algorithm uses three classifiers in a multi-expert approach as a generalization metric for the evolutionary process.

1) BENEFITS
F1-ECAC’s top average ranking proves the algorithm’s competitive performance, further supported by the results raised by the Friedman test. The absence of correlation between F1-ECAC and the contestant methods accounts for it as an algorithm with solutions that differ from conventional clustering approaches. F1-ECAC was able to adapt its solutions to the intrinsic nature of the datasets and get partitions that match the reference labels with an ARI of up to 0.96. Moreover, the correlation between the ARI of F1-ECAC’s solutions and the number of clusters shown in Fig. 10 is low, which stands for our algorithm’s stability regardless of this parameter.

2) LIMITATIONS
Synthetic data processing is a limitation of F1-ECAC, as suggested by the results obtained in this set of tests from Table 4. Nonetheless, our proposal is targeted towards real applications and was not designed to favor these structures, as they are not commonly found in real-world data mining tasks. Algorithms designed to favor 2-D structures offer substantial visualization advantages but lack cross-domain applicability due to their inherent clustering bias.

B. COMPARISON WITH TRADITIONAL METHODS
Despite the absence of significant difference when comparing F1-ECAC against \( k \)-means, our algorithm avoids some of the issues of other methods. The main disadvantage of traditional and evolutionary clustering algorithms following dissimilarity metrics based on distance is their nature of imposing a cluster structure, thus inducing clustering bias. This problem makes their application limited to data complying with specific characteristics. \( k \)-means and Single-linkage make evident the strong bias of traditional clustering methods as they benefit from specific structures, in this case, the compact clouds of R15 and the connectedness in the Spiral dataset’s clusters as seen in Subfigures (b), and (d) from Fig. 11. Our algorithm can solve this issue by optimizing the class separability according to each classifier’s kernel, which leads to diverse solutions with no predefined structures.

C. COMPARISON WITH EVOLUTIONARY METHODS
The results put F1-ECAC as the highest-ranked, followed by MOCLE, \( k \)-means, and HG-means. Even with the absence of statistical difference with HG-means and MOCLE, our algorithm outperformed both algorithms under challenging benchmarks. For instance, F1-ECAC could almost perfectly cluster the Iris dataset into three groups, only missing two instances and placing them in the wrong cluster (with an ARI of up to 0.96). This is a relevant problem because of the increased difficulty in differentiating the overlapping clusters, evident in Subfigure (g) from Fig. 11, which even the more complex methods could not achieve. Our proposed algorithm was ranked higher than the other evolutionary approaches, whereas being more intelligible and direct by generating a search space with only one objective and avoiding selecting solutions according to a Pareto front.

Some evolutionary approaches (i.e. MOCLE, \( \Delta \)-MOCK, and F1-ECAC) could generate overlapping and
non-geometric clusters in some datasets by favoring one or more objectives as shown in Subfigures (f), (h), and (j) from Fig. 11. Nevertheless, the partitions returned by k-means, HG-means and MOCLE got a considerably high correlation caused by the inclusion of the former in the clustering process of the latter two. This is a clear example of how despite following evolutionary search techniques, an algorithm could return solutions biased towards a structure if it follows conventional clustering criteria based on distance functions.

D. REMARKS
The inclusion of classifiers had been studied before to assess the quality of a partition as an internal index [1], but in this development, we used that principle and escalated it to design an evolutionary clustering algorithm. The benefits of using classifiers for finding the separation between classes were integrated into a clustering algorithm with relatively low runtime. It is important to remark that this algorithm is not intended to be used for processing big data. Just as it happens with the other clustering algorithms mentioned in this document, the dissimilarity computation using these metrics becomes unfeasible as the volume of data increases. Nonetheless, F1-ECAC proved competitive results and cross-domain applicability according to the experimental framework established for this development.

VII. CONCLUSION
F1-ECAC is a clustering method that uses the benefits of supervised learning to solve unsupervised learning problems. Our proposed method is implemented in an algorithm that is ready to be deployed in real-world data mining tasks. The hyper-parameter setting is straightforward and does not require prior information on the cluster structure of the data. The user only needs to specify the number of clusters, population size and maximum generations, which does not affect the objective function’s performance but influences the computation time to reach convergence.

Unlike common methods across the literature, we proposed an evolutionary approach to clustering by following a very different clustering criterion for evaluating partition quality based on a partition’s generalization degree instead of using a distance dissimilarity metric between clusters, thus avoiding cluster shape bias. Our proposed method outperformed state-of-the-art clustering algorithms whereas keeping the efficiency benefits of single-objective optimization. Using an ensemble of classifiers in the objective function and incorporating an evolutionary strategy led F1-ECAC to surpass the performance of other multi-objective approaches. We will apply F1-ECAC to real-world clustering tasks and data mining research in future work, and we will experiment with other techniques for exploring the search space to obtain high-quality partitions.

REFERENCES
[1] J. Rodríguez, M. A. Medina-Pérez, A. E. Gutierrez-Rodriguez, R. Monroy, and H. Terashima-Marin. “Cluster validation using an ensemble of supervised classifiers,” Knowl.-Based Syst., vol. 145, pp. 134–144, Apr. 2018.
[2] F. Provost and T. Fawcett, Data Science for Business: What You Need to Know About Data Mining and Data-Analytic Thinking. Newton, MA, USA: O’Reilly Media, 2013.
[3] A. K. Jain and R. C. Dubes, Algorithms for Clustering Data. Upper Saddle River, NJ, USA: Prentice-Hall, 1988.
[4] L. Kaufman and P. J. Rousseeuw, Finding Groups Data: Introduction to Cluster Analysis, vol. 344. Hoboken, NJ, USA: Wiley, 2009.
[5] S. Khedairia and M. T. Khadir. “A multiple clustering combination approach based on iterative voting process,” J. King Saud Univ.-Comput. Inf. Sci., vol. 1, pp. 1–10, Sep. 2019.
[6] B. M. Sainz-Tinajero, A. E. Gutierrez-Rodriguez, H. G. Ceballos, and J. Mach. Learn. Res., vol. 12, pp. 2825–2830, Oct. 2011.
[7] 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.
[8] B. M. Sainz-Tinajero, A. E. Gutierrez-Rodriguez, H. G. Ceballos, and J. Mach. Learn. Res., vol. 12, pp. 2825–2830, Oct. 2011.
[9] B. M. Sainz-Tinajero, A. E. Gutierrez-Rodriguez, H. G. Ceballos, and J. Mach. Learn. Res., vol. 12, pp. 2825–2830, Oct. 2011.
[10] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. “An efficient k-means clustering algorithm: Analysis and implementation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 7, pp. 881–892, Jul. 2002.
[11] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. “An efficient k-means clustering algorithm: Analysis and implementation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 7, pp. 881–892, Jul. 2002.
[12] S. Khedairia and M. T. Khadir. “A multiple clustering combination approach based on iterative voting process,” J. King Saud Univ.-Comput. Inf. Sci., vol. 1, pp. 1–10, Sep. 2019.
HECTOR G. CEBALLOS received the master’s and Ph.D. degrees in intelligent systems from the Tecnologico de Monterrey. He was the Head of the Scientometrics Office, Vice-Rectory Research, Tecnologico de Monterrey, for 18 years, where he is currently the Director of the Living Lab and Data Hub, Institute for the Future of Education. He is a full-time Faculty Member of the Computer Science Graduate Program and he is ascribed to the Intelligent Systems Research Group. His main research interests include social network analysis, process mining and agent theory, applied to research analytics, and educational data mining. He is a member of the Mexican National System of Researcher and an Adherent Member of the Mexican Academy of Computing.

FRANCISCO J. CANTU-ORTIZ received the B.S. degree in computer systems engineering from the Tecnologico de Monterrey (ITESM), Mexico, the M.S. degree in computer science from North Dakota State University, USA, and the Ph.D. degree in artificial intelligence from The University of Edinburgh, U.K. He is currently a Computer Science and Artificial Intelligence Professor with the Tecnologico de Monterrey. His research interests include data science, AI analytics, science and technology management, and philosophy of science and religion. He has published more than 100 scientific documents. He is a Certified Researcher by the National Council for Science and Technology, Mexico. He is a member of the Advisory Board for QS-World University Rankings and an associate editor of various journals and conferences (http://semtech.mty.itesm.mx/fcantu/).