Fuzzy Outlier Detection by Applying the ECF-Means Algorithm

A clustering ensemble approach for mining large datasets

Gaetano Zazzaro, Angelo Martone
CIRA
Italian Aerospace Research Centre
Capua (CE), Italy
e-mail: {g.zazzaro, a.martone}@cira.it

***Abstract*—This paper focuses on how to mine large datasets by applying the ECF-means algorithm, in order to detect potential outliers. ECF-means is a clustering algorithm, which combines different clustering results in ensemble, achieved by different runs of a chosen algorithm, into a single final clustering configuration. Furthermore, ECF is also a manner to "fuzzify" a clustering algorithm, assigning a membership degree to each point for each obtained cluster. A new kind of outlier, called o-rank fuzzy outlier, is also introduced; this element does not strongly belong to any cluster, which needs to be observed more closely; moreover, a novel validation index, called $o.FOUI$, is defined too, based on this new kind of fuzzy outliers. The proposed method for fuzzification is applied to the k-means clustering algorithm by using its Weka implementation and an ad-hoc developed software application. Through the three exposed case studies, the experimental outcomes on real world datasets, and the comparison with the results of other outlier detection methods, the proposed algorithm seems to provide other types of deeper detections; the first case study concerns the famous Wine dataset from the UCI Machine Learning Repository; the second one involves the analysis and exploration of data in meteorological domain, where various results are explained; finally, the third case study explores the well-known Iris dataset which, traditionally, has no outliers, while new information is discovered by the ECF-means algorithm and exposed here with many results.**

Keywords-ECF-means; Fuzzy Outlier Detection; Data Mining; Ensemble Clustering; k-means; Weka.

I. INTRODUCTION

The Ensemble Clustering Fuzzification (ECF) means [1] is an algorithm aimed at combining multiple clustering models to produce a better result than that of the individual clustering components. The proposed ensemble approach is carried on using the well-known k-means algorithm, its Weka implementation, and an ad-hoc developed software application. Compared to the version described in [1], we made some updates obtaining a new version of the algorithm. First of all, the most important variation to the algorithm consists in removing any equal partitions determined by two different runs of k-means algorithm. Therefore, all the ensemble results and the evaluation indexes are calculated on the number of different obtained partitions and not on the total number of performed iterations. Moreover, we also define a new validation index, called o-rank fuzzy outlier index ($o.FOUI$) by calculating the percentage of o-rank fuzzy outliers discovered by ECF.

The clusters achieved by the algorithm can be read in a "soft" way, in order to better explore and understand the results, and discover potential outliers in the dataset.

An outlier, or an anomaly, is an observation that is numerically distant from the rest of the data. What "distant" means depends on the context and on the domain, on the type of data and on the objective of analysis that must be achieved. Additionally, outlier detection is the process, or a technique, to find patterns in data that do not conform to estimated behavior. It plays a primary role in both statistical and data mining tasks, so much that identifying, understanding, and predicting anomalies from data is one of the key pillars of modern and advanced data analysis.

Nowadays methods and algorithms for data analysis increasingly involve huge amounts of data which are certainly rich in valuable information and useful knowledge, but also full of noise and impurities. The main challenges of outlier detection with this increasing complexity, variety, and volume of datasets, are how to discover similar outliers in one fell swoop, as a group [2]. Thus, an advantageous activity of data analysis must strictly have a data preparation step that includes data cleaning and anomalies removal activities.

Several outlier detection techniques have been proposed in literature [3]. Roughly, the approaches to outlier detection can be divided into two main categories: statistical and non-statistical methods. Within the non-statistical methods, the Machine Learning techniques for Data Mining are very popular, studied, and even among the most applied. Statistical methods are typically model-driven while Data Mining methods are typically data-driven. Mainly the Data Mining approaches for anomaly detection are divided into Proximity-based, Density-based, and Clustering-based techniques. This paper focuses on clustering-based outlier detection algorithms that look for outliers by applying one of the clustering algorithms and retrieve the anomalies subset. Moreover, this paper presents our idea on how apply the ECF-means algorithm for outlier detection task, and therefore, we explore datasets by applying this clustering-based technique.

Whilst usually it is hard to attach an outlier score to objects by using most of the clustering algorithms [4], the ECF-means algorithm defines a new class of outliers, consisting of elements that in [1] are named o-rank fuzzy outliers.

Therefore, ECF-means is able to assign a score to each element of the dataset depending on the vector constituted by its memberships to every obtained clusters. An element whose score is lower than a fixed threshold level is an outlier, because it belongs to two or more different clusters, without a clear and unambiguous membership. In this way, it is possible to have different kinds of outliers by fixing different thresholds: an element with level 1 is not a true outlier, whilst an element of level 0 is a full anomaly. These elements can be analyzed separately in order to understand if they are records that have undergone measurement errors and need to be deleted, or elements that are, from some point of view, special elements, confirming the idea that outliers are sometimes more interesting than the majority of the data.

Finally, the detected outliers are something new compared to those discovered by the crisp or traditional techniques for outlier analysis, and the two ways rarely discover the same anomalies.

In this paper, we present some applications of ECF-means algorithm, including datasets explorations, clustering results, outliers detection and classification. For the sake of clarity, this algorithm has many purposes and this paper focuses on an its application aimed at detecting outliers in various real-world benchmark datasets, testing its results and comparing the discovered anomalies with those detected by classical statistical methods. Our examples will demonstrate the efficiency of the ECF-means approach, comparing our results with those retrieved by other methods.

A. Structure of the paper

In Section II, we present some outlier detection generalities, including different approaches to discover anomalies in large datasets, and application fields. Particular attention is given to the clustering-based algorithms for anomaly detection and to properties of the algorithms for outliers detection.

In Section III, we provide cluster analysis general outlines, including main definitions, its scope and its role in Data Mining. Furthermore, some concepts regarding Ensemble Clustering, soft and hard clustering are mentioned.

In Section IV, the original k -means algorithm is synthesized, exposing its pros and cons.

In Section V, the ECF-means is presented, including some main definitions; in particular, we introduce the o-rank fuzzy outlier definition.

In Section VI, we present some validation measures for cluster analysis and for fuzzy clustering, including Silhouette, Partition Entropy Coefficient, the Threshold Index, and a novel index called o -rank fuzzy outlier index.

In Section VII, the ECF-means SW application is explained, underlining the updates of the new version.

In Sections VIII, IX, and X we show how the new version of the implemented software tool has been used in three different applications, underlining how it helped us to explore datasets, to discover new knowledge, to detect potential outliers, and to group objects in order to train custom models.

Finally, in Section XI, we show our general considerations in order to motivate future works and researches.

II. Outlier Detection

In outlier detection, the main goal is to discover objects that are different than the most other objects in the dataset. In many applications outliers contain important information and their correct identification is crucial.

A. Different Approaches and Application Fields

Outlier detection, or interchangeably anomaly detection, is the process of finding data objects with behaviors that are very different from expectation. Precisely, such objects are called outliers, anomalies, abnormalities, discordants or deviants. Outlier detection is generally considered a problem of machine learning or data mining, in the same way as classification and clustering.

A very common definition of an outlier is provided in [5] and it states:

"An Outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism."

In Data Mining and Statistics, outliers are particularly important aspects of the data. It may be difficult to evaluate the amount of noise in the data set or the number of outliers. More than that, what is noise or an outlier to one person may be interesting to another person. So, a unique mechanism that can identify outliers in datasets is impossible to define. Furthermore, the resolution of potential conflicts in detection is often possible thanks only to domain knowledge, which cannot always come to the aid of data analysts, especially when they are dealing with big data.

Table I shows several examples with categories of anomalies and it confirms why an uniform definition of anomaly is very hard to achieve. It is also evident that there is no clear separation between these types of outliers.

TABLE I. Examples of Outliers

	Example	Category
1	Outlier with respect to rest of the data points	Point Outlier, Contextual Outliers
2	Outlier with respect to local neighborhood	Point Outlier, Contextual Outliers
3	Outlier with respect to the data distribution	Point Outlier, Contextual Outliers
4	Outlier with respect to local dense regions	Point Outlier, Collective Outliers
5	Outlier tight cluster in a sparse region	Point Outlier, Collective Outliers
6	Outlier sparse cluster in a dense region	Collective Outliers

Outliers can be classified into three main categories that intersect with the examples in Table I:

1. point outliers, where a single instance of data is anomalous if it is too far off from the rest;
2. contextual outliers, if its value significantly deviates from the rest the data points in the same context; note that this means same value may not be considered an outlier if it occurred in a different context; these outliers are common in time-series data;
3. collective outliers, where a set of data instances collectively helps in detecting anomalies.

The technique chosen for anomaly detection can be based on the type of data you are dealing with. Many of these approaches have been exactly developed for certain application domains, while others are more generic and reusable. One of the main criteria used to select the outlier detection technique is based on the nature of the outlier to discover.

Approaches for outlier detection are reported in Table II [6] [7] [8].

TABLE II. METHODS FOR OUTLIER DETECTION

Method	Description
Nearest neighbor-based	lazily analyze the nearest neighborhood of a test point to assign it an outlier score relative to their neighborhood; in particular, these methods are based on the idea that normal instances lie in dense neighborhoods whilst anomalous points lie in sparse neighborhoods
Clustering-based	the anomaly score assigned to a test point is based on its relationship with its nearest cluster; by using these algorithms, outliers do not belong to a cluster or are far away from the nearest cluster representative
Classification-based	operate under the hypothesis that a classifier can be trained from a given feature space that can discriminate between normal and outlier classes
Statistical	are based on building a probability distribution model and considering how likely objects are under that model; outliers appear in the low probability regions of the distribution and have high anomaly score
Spectral decomposition-based	are unsupervised learning techniques that find an approximation of the data using a combination of attributes that reveal low dimensional structure in dimensional data; points that are different from others in the lower approximation are detected as outliers
Information theoretic	by which an anomalous point is an instance that has irregularities in the information content by using different information theoretic measures such as entropy or complexity

Other authors report [4] that the Machine Learning methods for outlier detection can be classified as proximity-based techniques, mainly founded on Nearest Neighbor concept, density-based techniques, where outliers are objects that are in regions of low density, and clustering-based techniques, where outliers can form small groupings or where an outlier is an object does not strongly belong to any cluster.

TABLE III. APPLICATIONS FIELDS OF OUTLIER DETECTION

	Field	Goal
1	Medical Informatics	Healthcare analysis, diagnostics, in a wide variety of medical specializations
2	Intrusion Detection	Break-ins, penetrations, and other form of computer abuse detection
3	Fraud Detection	Credit card, mobile phone
4	Fault / Damage Detection	Industry damage, multi-failures in Avionics systems, and others
5	Crime Investigation / Counter Terror Op. Planning	Fake news, misinformation, security, surveillance, social network monitoring
6	Other Domains	Speech Recognition, Traffic Monitoring, Detecting Faults in Web Applications, Detectiong Outliers in Astronomical Data, in CRM, in Census Data, in Biological Data, Novelty Detection in Robot Behaviour, Click Through Protection, and others

The well-known DBSCAN clustering algorithm [9] can be considered, for example, both in the density-based category and in the clustering-based one, because DBSCAN assumes that regions or points with density lower than a global threshold are noise or outliers.

It is very important to underline that outlier detection can represent both the objective of the analysis and a data cleaning method in the Data Preparation step of the CRISP-DM [10].

Outlier detection is central for a wide variety of applications and activities [11] [12] such as reported in Table III.

B. Clustering-based Outlier Detection

Outlier detection finds objects that are not strongly related to other objects, while on the contrary cluster analysis finds groups of strongly related objects in the dataset. One implication is that clustering can be successfully used for outlier analysis.

This approach is based on the idea that anomalous points do not belong to a cluster or to a single cluster without ambiguity. Another possibility occurs when the anomalous point is far away from the nearest normal cluster representative (such as the centroid, the medoide, etc.). The use of an unsupervised outlier detection approach based on a clustering algorithm also has the advantage of avoiding the bias introduced by training an algorithm with anomalous instances, labeled wrongly as normal data, producing many and unmanageable misclassifications.

Applying a clustering method to detect anomalies can discover, such as all the kinds of outlier detection approaches, single, spare, and rare elements, or also clusters of outliers, forming a collection, which is why in this case anomalies are called “collective outliers”. For the sake of clarity, many times the collective outliers aggregate far from the rest of the elements of the dataset, and then they could be modeled by a rule, forming a real new cluster and not an actual set of anomalies. So, an almost always applicable rule by which we can understand if a potential anomaly is a real anomaly, is to recognize if the anomaly belongs to a pattern. As already mentioned, the subset of the detected anomalous points may also not form a cluster, for example, because they are not cohesive with each other or they are not well separated from the rest of the clusters.

The main idea to detect anomalies by using a clustering approach is based on the property that “normal” data belong to large and dense clusters, whereas outliers belong to small or sparse clusters, or they are singletons, not belonging to any cluster. Consequently, all clusters smaller than a minimum size can be discarded, reassessed, or reported.

The clustering approach to anomaly detection can be used with any clustering algorithm, but requires thresholds for the minimum cluster size and the distance between a small cluster and other clusters. The idea of a clustering-based algorithm for outlier detection is captured by the definition [4]:

“An object is a cluster-based outlier if the object does not strongly belong to any cluster.”

In this context, “strongly” also means “clearly” and “without ambiguity”, and in the next sections the meaning of these words will be more explicit.

The binding of a point to its cluster can be measured by a score. A simple way of defining this score is to first cluster the whole dataset and then use the distance of the point to its closest cluster centroid. The points that have the highest scores are potential outliers.

After objects of a set are clustered and outliers are discovered and removed, the set with the rest of objects can be clustered again, analyzing if outliers affect the clustering. This is a very thorny problem, especially if k-means is chosen as the clustering algorithm, the results of which depend on the choice of the initial centroids and on the number k of clusters to be determined.

Another problem to be highlighted concerns the size of the clusters. If a cluster has very few points, these are potential outliers only if the other clusters have much larger sizes, otherwise nothing can be said. In a nutshell, outlier detection is strongly conditioned by the selected threshold.

Furthermore, many clustering techniques, such as k-means, do not automatically determine the number of clusters. This is a problem when using clustering in outlier detection, since whether an object is considered an outlier or not may depend on the number of obtained clusters.

All these aspects make us understand how the problem of outlier detection is strictly dependent on the set of data, the chosen method, the parameters of the algorithm, and the analysis goals. As consequence we have that we are not always sure that a discovered point is really anomalous.

The cluster-based outlier definition inherently contains a fuzzy reinterpretation of cluster analysis and outlier detection. For this reason, the ECF-means algorithm is a good candidate to detect potential outliers and to explore the dataset and clustering results.

C. Fuzzy Clustering-based Approach to Outlier Detection

In order to detect anomalies, fuzzy clustering is an alternative approach to classical clustering-based methods that are centered on the concept of crisp membership. Recently fuzzy clustering-based methods are applied in many fields, analyzing various types of data, and also getting very good results [13] [14] [15] [16]. In most of these works, however, authors use the fuzzy c-means clustering (FCM) algorithm, or a variant of it, for detecting outliers. Moreover, many crisp clustering techniques have difficulties in handling extreme outliers but fuzzy clustering algorithms tend to give them very small membership degree in surrounding clusters.

A systematic approach is to first cluster all objects and then assess the degree to which an object belongs to any cluster. Generally, the rules of assignment of the degree of membership characterizes the fuzzy clustering results. For example, the distance of an object to its cluster center can be used to measure the degree to which the object belongs to a specific cluster. If the elimination of an object results in a substantial improvement in the objective, then we would classify the object as an outlier. In a nutshell, clustering creates a model of the data and anomalies distort that model.

Another way to state if an object is an anomalous point is to calculate a level of "undecidability" of the point; for example, if the point belongs to two clusters with the same degree, then it has a high possibility of being an outlier.

D. Ensemble Outlier Detection

Mainly, an ensemble classical approach to outlier detection combines the outputs of individual outlier detection components by a weighted majority voting rule in a complete unsupervised context. The construction of ensembles is proposed as a solution to increase the individual capacity of each algorithm component and to improve the "anomalousness" of a potential outlier. However, no gain will be obtained by using components whose results are equal. So the discovered outliers sets must have some different anomalous points.

Whilst unsupervised outlier detection algorithms often suffer from high false positive detection rates, an ensemble approaches can be used to reduce these rates and many applications have shown good results to achieve more accurate and reliable anomalies [17].

The ensemble components can be built by applying each approach in Table II. However, the classification-based methods are more common than clustering-based ones [18].

The approach presented here is not a classical clustering-based method, it is a rather unique approach between ensemble methods for anomaly detection, due to its hybrid nature. In fact, on the one hand, it is the consequence of an aggregation clustering method, on the other hand, it exploits its fuzzy implication to assign scores to all points of the dataset, and on the basis of these scores it attributes a level of "outlierness" to the points.

E. Properties of the Algorithms for Outlier Detection

An anomaly detection method can enjoy some properties.

The idea presented here is to apply an algorithm for anomaly detection to a dataset, to remove the discovered anomalous points, and then to apply to the remaining data the same algorithm again.

Let S be a set of points, F an outlier detection method, and A the set of outliers of S discovered by F. In this case we can write $F(S) = A$. If $F(S - A) = \emptyset$, then F is an ***invariant*** algorithm for S (or S is invariant respect to F). In this case, F finds all the outliers of S in one fell swoop. If F is invariant for each set, simply F is invariant.

The mentioned DBSCAN algorithm [4] [9] can be considered as a clustering-based outlier detection method; in a nutshell, the clustering results depend on the radius ε of the epsilon-range-queries (which are hyperspheres) and on m parameter that is the minimum number of data objects required in an epsilon-range-query. For DBSCAN the outliers are noise, i.e., those points belonging to those clusters that have less than m elements, and therefore, they are not reached by any epsilon-range-query. Fixing ε and m, DBSCAN is invariant for each set.

The ECF-means algorithm is invariant for some datasets and not for others, as will be shown in the case studies in the next sections.

For the sake of clarity, the invariance property is not a property of the anomaly detection method, but of the set analyzed by the method.

If F enjoys the invariance property for each set, then on one hand, F is a very "robust" method for outliers detection,

but on the other hand, the method is a very strong property, or it can be very hard to prove it. It can be relaxed by a more general property explained in Figure 1.

Invariance Property of Order n

$S = A_0$ set of points
F outlier detection method
A_1 the set of outliers of S discovered by F:
$A_1 = F(S) = F(A_0)$
A_2 the set of outliers of $A_0 - A_1$:
$A_2 = F\big(S - F(S)\big) = F(A_0 - A_1)$
A_3 the set of outliers of $A_0 - (A_1 \cup A_2)$:
$A_3 = F\left(S - \left(F(S) \cup F\big(S - F(S)\big)\right)\right) = F(A_0 - (A_1 \cup A_2))$
… …
$A_n = F(A_0 - (A_1 \cup A_2 \cup \ldots \cup A_{n-1}))$
$A_{n+1} = F(A_0 - (A_1 \cup A_2 \cup \ldots \cup A_n))$

F is n-invariant for S if and only if $A_{n+1} = \emptyset$ and $A_n \neq \emptyset$

Figure 1. Generic Invariance Property of F.

In the generic invariance property of F, $A_{n+1} \cap A_n = \emptyset$ $(n > 0)$.

III. CLUSTER ANALYSIS

In order to detect potential outliers and to explore the dataset, cluster analysis is widely used in data mining. In this section, some general clustering considerations are shown.

A. Introduction

Clustering, or cluster analysis, belongs to intersection of Statistics, Machine Learning, and Pattern Recognition. It is a very useful unsupervised method for discovery pattern in large amount of data. It is a technique to group a set of objects into subsets or clusters. It is widely used [19] for Data Mining tasks, because it can be easily applied to understand, explore, prepare, and model data. It plays an outstanding role in many applications, such as scientific data exploration, information retrieval and text mining, web analysis, bioinformatics, and many others.

It can be applied at various steps of the Knowledge Discovery in Database (KDD) process. KDD can be carried out according to the Cross Industry Standard Process for Data Mining (CRISP-DM) [10]. Table IV shows the six steps of CRISP and where the cluster analysis can be applied. Some cluster analysis tasks are also reported.

TABLE IV. CRISP-DM STEPS AND CLUSTER ANALYSIS

	CRISP-DM Step Name	Cluster Analysis Tasks
I	Business Understanding	--------
II	Data Understanding	Data Exploration and Description
III	Data Preparation	Data Selection, Cleaning and Reduction, Features Selection, Gain and Raising
IV	Modeling	Generate Test Design, Data Modeling (Segmentation, Associative Rules, …), Model Customization
V	Evaluation	--------
VI	Deployment	--------

In the literature, there are many categories of algorithms for clustering: Heuristic-based, Model-based, and Density-based [20]. Their common goal is to create clusters so that objects in the same cluster should be as similar as possible, whereas objects in one cluster should be as dissimilar as possible from objects in the other clusters. Usually, it is not easy to choose the most useful algorithmic approach, the most satisfying result, and therefore, the most usable configuration. In fact, the different models for clustering may produce configurations that are very different from one another. Anyone applying a clustering algorithm immediately realizes how difficult it is to choose the final cluster configuration. We may have different results because we choose different algorithms, or different parameters of the fixed algorithm. Furthermore, the numerous available evaluation metrics often do not facilitate this choice because they lead to very discordant results.

In spite of the availability of a large number of validation criteria, the ability to truly test the quality of a final configuration remains vague and hard to achieve. Specific domain knowledge is not an aid because it is often hard to translate it into operating rules, neither the domain expert has a real target class for evaluating and comparing the results. So, why do not consider all the obtained configurations? That is, why do not find a method that summarizes all the results of clusterings? Meta-learning ensemble methods may be an answer. The idea is that no single model or criterion truly captures the optimal clustering, but a cooperation of models could provide a more robust solution. Cluster Ensemble, or Aggregation Clustering, or Multiview Clustering, aims to find a single clustering from multi-source basic clusterings on the same group of data objects [21]. However, these ensemble methods, such as voting-based clustering [22], consensus clustering [23], or clustering aggregation [24] do not assign a level of membership to every point in clusters.

In order to overcome the limits mentioned above, the ECF-means algorithm can be included within ensemble procedures. It is also an *a posteriori* criterion for optimization of the obtained groupings. This procedure takes in input any partitioning clustering algorithm for which it is possible to initially choose the k number of clusters to be determined and a seed for the random choice of the initial k centroids.

The k-means algorithm is one of the clustering algorithms that checks all the conditions listed. So, it is considered as the reference clustering algorithm. In Weka implementation of k-means [25] [26], the name of the algorithm is *SimpleKMeans*; in this version the seed parameter is s, which is the initialization value for the random number generator. Using the same seed value will always result in the same initial centroids then. Exploiting this seed parameter, many different configurations are evaluated and compared, and also used in our meta-algorithm for ensemble final configuration.

ECF-means can lead also to a "soft" interpretation of the clusters, in order to better explore and understand the results, and to find possible outliers in the dataset.

B. Definitions and Scope

A Clustering algorithm produces a partition on an unlabeled data set, such that no cluster is empty, no two

clusters intersect, and the union of all clusters is the data set itself.

The goal is to create clusters that are coherent internally, but substantially different from each other. In a nutshell, objects in the same cluster should be as similar as possible, whereas objects in one cluster should be as dissimilar as possible from objects in the other clusters.

Similarity between objects that belong to a cluster is usually measured by a metrics d. Two objects x and y are similar if the value of $d(x, y)$ is small; what "small" means depends on the context of the problem. d is defined by some distance measure. Typically, the Euclidean Distance (or simply the squared Euclidean Distance) is widely used in many applications (it is also used in the ECF-means) for the computation of similarities:

$$ED^2(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2$$

It is important to underline that, also depending on the type of data, other many metrics are possible.

Numerous clustering algorithms are available in the literature and there are several points of view for examining clustering techniques; a very good landscape of Clustering algorithms can be retrieved in [20], and an in-depth and complete study of clustering techniques, algorithms and applications can be retrieved in [27].

C. Ensemble Clustering

Different clustering approaches or different views of the data can lead to different solutions to the clustering problem. Indeed, also initial settings of a fixed algorithm may produce clusters that are very different from one another. This evidence is closely related to the theory of Ensemble Clustering (or Multiview Clustering), which studies this issue from a broader perspective [21] [28].

Therefore, instead of running the risk of picking an unsuitable clustering algorithm, a cluster ensemble can be used in order to get a "better" clustering configuration. The idea is that no single model or criterion truly captures the optimal clustering, but a collective of models will provide a more robust final solution.

Most ensemble models use the following three steps to discover the final clusters configuration:

1. Generate N different clusterings, by using different approaches, or different data selection, different settings of the same algorithm, or different clusterings provided by different runs of the same algorithm. These represent the ensemble components.
2. Combine the results into a single and more robust clustering, by using a meta-rule or a set of rules.
3. Evaluate the ensemble clustering result and compare it with the results of the N components.

As already mentioned, the ensemble components can be selected in a wide variety of ways.

Some strategies for building clustering ensemble components follow:

1. By using different subsets of features. Each clustering configuration is found by means of overlapping or disjoint subsets of the original features set.
2. By selecting different subsets of the data, via random sampling.
3. The different components can be selected combining a variety of models and algorithms such as partitioning, hierarchical or density-based methods, random or deterministic algorithms, and so on.
4. The different components can correspond to different settings of the same algorithm.
5. The different components could be obtained from a single algorithm, randomizing the initial choice of the clusters centroids. Of course, an example is k-means; thus, the ensemble can be formed as the result of N different runs of the algorithm.

After the individual components have been obtained, it is often a challenge to find a meta-rule able to combine the results from these different solutions in order to create a unified ensemble clustering.

D. Hard and Soft Clustering

Clustering algorithms can also be classified into hard and soft algorithms. A hard clustering algorithm leads to a partition of crisp sets. In a crisp set, an element is either a member of the set or not. On the other hand, a soft clustering algorithm leads to fuzzy clusters. Fuzzy sets allow elements to be partially in a set. Each element is given a degree of membership in a set.

One of the most famous fuzzy clustering algorithms is fuzzy c-means [29] (FC-means), which allows an object to belong to two or more clusters with a membership degree between zero (not an element of the set) and one (a member of the set). It has been widely used in many real-world application domains where well-separated clusters are typically not available.

The ECF-means algorithm leads to a fuzzy partitioning of the dataset, by repeatedly applying the results of the k-means algorithm, as reported in next sections.

IV. THE k-MEANS ALGORITHM

k-means is a simple clustering algorithm whose main goal is to find k non-overlapping clusters. Each final cluster is represented by its centroid that is typically the mean of the points in that cluster.

A. Introduction, scope and procedure

k-means is one of the oldest and still widely used algorithms for cluster analysis. Without any doubt, it represents the archetype of the clustering partitioning algorithms. Because of its mathematical simplicity, it is also the most studied unsupervised learning technique [30], and over the years, many of its variations and extensions have been implemented (for High-Dimensional Data, for Data Streams, Time Series, for Data with noise, and so on).

k-means is also a simple prototype-based clustering algorithm that uses the centroid of the objects in a cluster as the prototype of the cluster.

Its basic algorithmic structure is shown in Figure 2.

k-means Clustering Algorithm

Input: S set of instances; k number of clusters
Output: set of k clusters with k centroids

1. Randomly initialize k cluster centers (centroids)
2. **While** termination condition is not satisfied {
3. Assign instances to the closest cluster center
4. Update cluster centers using the instances assignment
5. }

Figure 2. k-means Algorithm.

The condition of termination of the process is satisfied when no point changes clusters.

B. Pros and Cons

The algorithm has been very successful thanks to its simplicity and also for its linear time complexity $O(knl)$, where n is the number of objects to be clustered and l is the number of iterations that the algorithm is performing.

Like most partitioning clustering algorithms, k-means has some disadvantages:

1. It is very sensitive to outliers and noise.
2. The number of clusters need to be specified by the user and often it is not simple to choose it.
3. It is not able to discover concave-shaped clusters.
4. Since the initial choice of k centroids is random, different selections can also lead to very different final partitions, especially for large datasets with many features.

The k-means algorithm always terminates, but it does not necessarily find the "best" set of clusters.

C. Fuzzy c-means

The fuzzy c-means (FCM) algorithm has got many versions. The code of Figure 3 does not use incremental updates of cluster centroids.

Basic Fuzzy c-means Algorithm

1. Select an initial fuzzy pseudo-partition, i.e. assign values to all the w_{ij}
2. **repeat**
3. Compute the centroid of each cluster using the fuzzy pseudo-partition
4. Recompute the fuzzy pseudo-partition. i.e., the w_{ij}
5. **until** The centroids do not change.
(Alternative stopping conditions are "if the change in the error is below a specified threshold" or "if the absolute change in any w_{ij} is below a given threshold."

Figure 3. Basic Fuzzy c-means Algorithm.

k-means can be regarded as a special case of fuzzy c-means [4] and the behavior of the two algorithms is quite similar.

V. Ensemble Clustering Fuzzification Means

The initial selection of centroids can significantly affect the result of the k-means algorithm. To overcome this, the algorithm can be run several times for a fixed value of k, each time with a different choice of the initial k centroids.

In many software implementations of k-means, for example, in its Weka version, it is possible to choose a seed parameter (s), useful for the random selection of the first initial centroids (s is the random number seed to be used). Using this parameter, it is possible to realize, as will be described in the following sections, a procedure able to optimize and reinforce the obtained partition.

A. Introduction and Definitions

Let $S \subseteq \mathbb{R}^m$ be a set of points. Let k be the desired number of clusters to be determined. Changing the seed (s) from 0 to $N-1$, N partitions of S can be generated by applying the k-means algorithm. Some of these partitions are exactly the same, considering or not the order of groupings. Others, however, differ for very few records, and others for many.

In the following $N \times k$ matrix, called Clustering Matrix C of S, each row is a partition of k clusters of S.

$$C = \begin{pmatrix} C_{1,1} & C_{1,2} & \dots & C_{1,k} \\ C_{2,1} & C_{2,2} & \dots & C_{2,k} \\ \dots & \dots & C_{i,j} & \dots \\ C_{N,1} & C_{N,2} & \dots & C_{N,k} \end{pmatrix}$$

$C_{i,j}$ is the j-th cluster obtained at the i-th iteration of the clustering algorithm, with $i = 1, \dots, N$ and $j = 1, \dots, k$.

It is possible to associate a new $N \times k$ matrix to C, called MU matrix, which is the matrix of the centroids of the clusters:

$$C \to \begin{pmatrix} \mu(C_{1,1}) & \mu(C_{1,2}) & \dots & \mu(C_{1,k}) \\ \mu(C_{2,1}) & \mu(C_{2,2}) & \dots & \mu(C_{2,k}) \\ \dots & \dots & \mu(C_{i,j}) & \dots \\ \mu(C_{N,1}) & \mu(C_{N,2}) & \dots & \mu(C_{N,k}) \end{pmatrix} = MU$$

$\mu(C_{i,j})$ is the arithmetic mean of the j-th cluster of the i-th iteration of the algorithm, with $i = 1, \dots, N$ and $j = 1, \dots, k$.

B. Clusters Sort Algorithm

The algorithm in Figure 4 is useful for sorting the clusters partitions of C matrix. This step is essential because k-means can produce different orders of clusters in different runs, even if the partitioning results can be the same.

Please note it is possible that the average of some elements of the second row C_2 in Algorithm 1 has a minimum distance from two or more averages of elements of the first row C_1. In this case, the minimum value of the minimum values is chosen.

C. The ECF-means Algorithm

Let C be a Clustering Matrix of S, sorted by using the Algorithm 1.

We define $\underline{C_j}$ as ***floor of*** $\boldsymbol{C_j}$: $\underline{C_j} = \bigcap_{i=1}^{N} C_{i,j}$, with $j = 1, \dots, k$. It is possible that $\underline{C_j} = \emptyset$ ($j = 1, \dots, k$). Moreover, $\underline{\boldsymbol{S}} = \bigcup_{j=1}^{k} \underline{C_j}$ is defined as the ***floor of*** $\boldsymbol{S}$.

Algorithm 1: Clusters Sort Algorithm

Input: two different rows of C:
$C_1 = (C_{1,1}, C_{1,2}, \dots, C_{1,k})$ and $C_2 = (C_{2,1}, C_{2,2}, \dots, C_{2,k})$
Output: a new order of the second row:
$(C'_{2,1}, C'_{2,2}, \dots, C'_{2,k}) = C'_2$

C_1 represents the reference row of the current sorting procedure (e.g., obtained by fixing $s = 0$ in the Weka k-means algorithm).

1. Calculate the $2 \times k$ matrix of clusters centroids:
$$MU = \begin{pmatrix} \mu(C_{1,1}), \mu(C_{1,2}), \dots, \mu(C_{1,k}) \\ \mu(C_{2,1}), \mu(C_{2,2}), \dots, \mu(C_{2,k}) \end{pmatrix}$$
2. Compute the Euclidean Distances (ED) in MU. The following $k \times k$ matrix is the Δ matrix of the EDs:
$$\Delta = \begin{pmatrix} d_{1,1} & d_{1,2} & \dots & d_{1,k} \\ d_{2,1} & d_{2,2} & \dots & d_{2,k} \\ \dots & \dots & \dots & \dots \\ d_{k,1} & d_{k,2} & \dots & d_{k,k} \end{pmatrix}$$
Where:
$d_{i,j} = ED\left(\mu(C_{1,i}), \mu(C_{2,j})\right)$, with $i, j = 1, \dots, k$.
3. Calculate the minimum value of each row of Δ.
$\min\{d_{1,1}, d_{1,2}, \dots, d_{1,k}\} = d_{1,\overline{j_1}} = min_1$
$\min\{d_{2,1}, d_{2,2}, \dots, d_{2,k}\} = d_{2,\overline{j_2}} = min_2$
… … … … … …
$\min\{d_{k,1}, d_{k,2}, \dots, d_{k,k}\} = d_{k,\overline{j_k}} = min_k$
4. The second row C'_2 is:
$(C'_{2,1}, C'_{2,2}, \dots, C'_{2,k}) = (C_{2,\overline{j_1}}, C_{2,\overline{j_2}}, \dots, C_{2,\overline{j_k}})$
Where:
$C'_{2,1} = C_{2,\overline{j_1}}$ is the cluster (in C_2) that has the centroid with the minimum distance from the centroid of the first element of C_1.
… … …
$C'_{2,k} = C_{2,\overline{j_k}}$ is the cluster (in C_2) that has the centroid with the minimum distance from the centroid of the k-th element of C_1.

Figure 4. Clusters Sort Algorithm.

Let x be an element of S; we can count the number of clusters of the first column of C where x is, the number of clusters of the second column of C where x is, and so on. In this way, we can associate a new numerical vector to x, called ***attitude of x*** ($att(x)$):

$$att(x) = \left(att_1(x), att_2(x), \dots, att_k(x)\right),$$

where $att_j(x)$ is the number of clusters in the j-th column of C where x is located. $att_j(x) = N \Leftrightarrow x \in \underline{C_j}$ and $\sum_{j=1}^{k} att_j(x) = N$. In this manner, we are defining a function att_j ($j = 1, \dots, k$ and $I = \{1,2, \dots, N\}$):

$$att_j : x \in \bigcup_I C_{i,j} \rightarrow att_j(x) = \left|\{i \in I : x \in C_{i,j}\}\right|$$

where, as usual, $|A|$ is the number of the elements of the set A.

Finally, we can define the ***probability vector of x***, as:

$$p(x) = \left(\frac{att_1(x)}{N}, \frac{att_2(x)}{N}, \dots, \frac{att_k(x)}{N}\right)$$

Thanks to the simple mathematical notions of the current section, we are able to "soften" the "hard" k-means algorithm and we can have a new Fuzzy Clustering Algorithm. According to this approach, each element of the dataset belongs to each cluster with a different degree of membership, and the sum of these probabilities is equal to one.

Furthermore, the method can also be interpreted in a different way. Indeed, this "fuzzification" procedure can be used not only with k-means algorithm, but also for others partitional clustering algorithms for which it is possible to choose the number of clusters to be determined. In this way, the algorithm is part of the Ensemble algorithms. For these reasons, ECF-means is also a meta-algorithm because we reach a fuzzy partition of the dataset by using a multiple clustering algorithm schema.

The Algorithm 2 of Figure 5 is able to assign a probability membership to each point of the dataset and to "slice" the data in clusters.

Algorithm 2: ECF-means (Fuzzification of k-means)

Input: $S \subseteq \mathbb{R}^m$; number k of clusters to be determined; membership threshold t ($0 \leq t \leq 1$); number N of k-means iterations
Output: set of k clusters of level t; probability vector of each element x of S

1. **Apply** the k-means algorithm to S, fixing the random seed $s = 0$, obtaining the clusters $C_{0,1}, \dots, C_{0,k}$ ($C(0)$-configuration)
2. **foreach** $s = 1, \dots, N-1$
3. **Apply** the k-means algorithm to S, obtaining the clusters $C'_{s,1}, \dots, C'_{s,k}$ ($C'(s)$-configuration)
4. **Apply** the Clusters Sort Algorithm to $C'(s)$, considering $C(0)$ as reference, obtaining the clusters $C_{s,1}, \dots, C_{s,k}$ ($C(s)$-configuration)
5. **end**
6. **foreach** $j = 1, \dots, k$
7. **foreach** $x \in S$
8. **Calculate** $p_j(x) = att_j(x)/N$
9. **Fix** the cluster $C_j^t = \{x \in S |\, p_j(x) \geq t\}$
10. **end**
11. **end**

Figure 5. ECF-means Algorithm.

The membership threshold t in the Algorithm 2 is fixed by the user and it is very useful to change the "level" to clusters final configuration. If $t = 1$, then $C_j^1 = \{x \in S | p_j(x) = 1\} = \underline{C_j}$. Additionally, $\underline{S} = \bigcup_{j=1}^{k} C_j^1$. If $t = 0$, then $C_j^0 = \{x \in S | p_j(x) \geq 0\}$ and $\bigcup_{j=1}^{k} C_j^0 = C_j^0 = S$.

Let $p(x)$ be the probability vector of x and let $M = \max att(x) = \max\{att_1(x), att_2(x), \dots, att_k(x)\}$ be the maximum of $att(x)$, if this exists. We can define the position of M in $att(x)$ as $PMA(x)$, if this exists.

D. o-rank Fuzzy Outlier

As has been defined, to each point of the dataset it is possible to associate a probability vector, which is as matter of fact a vector of degrees of memberships. What happens if we cannot unambiguously identify the cluster to which the point belongs? What happens if the two highest components

of the vector are equal (or almost equal)? In this case, two interpretations are possible; the "ambiguity":

1. depends on the clustering procedure, for example, on the choice of k or on the selection of the number N of iterations,
2. is intrinsic to the point because the point is an anomaly.

The first interpretation leads to the definition of a novelty validation measure called Threshold Index TI, which is reported in the next section. Thanks to the second interpretation it is possible to define a new particular type of outlier, called o-rank fuzzy outlier.

An element $x \in S$ is an ***o*-rank fuzzy outlier of S** if $p_j(x) - p_l(x) \leq o$ $(0 \leq o \leq 1)$, where $p_j(x)$ and $p_l(x)$ are the first two highest value components of $p(x)$. The o parameter is also named "outlier threshold".

The definition of o-rank fuzzy outlier helps us to treat these points as special points, which need to be observed more closely, because they belong at least to two different clusters, with a degree of "ambiguity".

The o parameter allows to create a hierarchy of outliers; that is, o represents a score to be assigned to each point of the dataset. For example, if $k = 3$ and $p(x_1) = (1, 0, 0)$, $p(x_2) = (0.1, 0.45, 0.45)$, $p(x_3) = (0.75, 0.05, 0.2)$, and $p(x_4) = (0.5, 0.4, 0.1)$, then:

- x_1, x_2, x_3, x_4 are 1-rank outliers
- x_2, x_3, x_4 are 0.55-rank outliers
- x_2, x_4 are 0.1-rank outliers
- x_2 is a 0-rank outlier

The set of 1-rank fuzzy outliers of S is S; the set of 1-rank fuzzy outliers of S that aren't o-rank fuzzy outlier, with $o < 1$, is the floor of S. From the "o-rank fuzzy outlier" perspective, each point in the dataset is an outlier.

The o-rank fuzzy outliers, where o is close to 0, are the most interesting points of the dataset, which need to be analyzed separately from the rest of the other points. Their fuzzy nature pushes us to a deepening, also to understand if they are "polluting" elements of the dataset, or they are wrongly selected by the algorithm and, therefore, they are ambiguous points for the algorithm but not anomalous in the dataset, or they constitute the main objective of detection.

If ECF is considered a method for outliers detection, and if o.FOU is the set of o-rank fuzzy outliers of S, it could be very interesting to find the highest o for which ECF-means is invariant for S, naturally if a such o exists:

$$\max\{o: ECF(S - o.FOU) = \emptyset\}$$

For this selected o, the set of o-rank fuzzy outliers of S that make ECF-means an invariant method for outliers detection is a special subset that has to be detailed.

E. Updates to the New Version of the Algorithm

As pointed in the previous sections, changing the seed (s) from 0 to $N - 1$, N partitions of S can be generated by applying the k-means algorithm. Some of these partitions are exactly the same, considering or not the order of groupings. In this new version of the algorithm, after applying the clusters sorting algorithm, all the identical configurations are deleted, leaving only all the different partitions. Obviously, two partitions are different if they have at least one element that belongs to two different clusters of the two partitions.

This change effects on the whole designed method. As consequence of this choice, all the ensemble results and the evaluation indexes are calculated on the number of different obtained partitions and not on the total number of performed iterations.

Two different seeds could lead to the same partition because through them the same initial centroids are selected, or because of some topological reason, which is for the geometric distributions of the elements in the dataset. It might be interesting to find out some rules by which different selections of the initial centroids lead to the same final configuration and to discover if a pattern exists for these elements. This change was made above all to avoid that the Weka algorithm based on the seed (s) too often chooses the same initial centroids and, therefore, that the corresponding partitions are too privileged and impact on the final result.

Even the case studies presented have clearer results thanks to the changes that have been made, and the new validation index facilitates the interpretation of the achieved results.

Note that, especially for small datasets, the possible different final configurations can be very few; in this case, an ensemble approach can be superfluous or even useless.

VI. Cluster Validity Assessment

Clustering validation has long been recognized as one of the critical issues essential to success of clustering applications [27].

A. Introduction

One of the most important issues in clusters analysis is the evaluation of the clustering results. In order to compare the outputs of different clustering algorithms, or the different partitions retrieved by the same clustering algorithm by using different parameters, it is necessary to develop some validity criteria. Moreover, if the number of clusters is not given by the clustering algorithm, many cluster validity methods have been developed in the literature; indeed, to find the optimal number of clusters in the data set is a very central task in data analysis. These methods lead to many different indices, specialized for the various categories and approaches of clustering algorithms; moreover, the methods are usually divided in supervized and un-supervized methods, or in internal and external validation criteria [27].

Most validation indices take into account the concepts of cohesion and separation [31]; therefore, the index is a measure that "optimizes":

- Cohesion, also called compactness or tightness: patterns in one cluster should be as similar to each other as possible. The fitness variance of the patterns in a cluster is an indication of the cluster's cohesion.
- Separation: clusters should be well separated. Distance among the representatives of the clusters provides an indication of cluster separation.

B. Silhouette Index

One of the most widespread and useful indices is the Silhouette [32] [33]. The Silhouette method is an unsupervized method (it does not need a class attribute to calculate it) for evaluation of clusterings. It is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).

The Silhouette index is calculated strarting from the definition of the silhouette of each point of the dataset. The silhouette $s(i)$ of a point i is calculated by the following formula:

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

Where:

- $s(i)$ is the silhouette of i
- i is a generic point in the dataset;
- $a(i)$ is the average distance between i and all other data within the same cluster.
- $b(i)$ is the smallest average distance of i to all points in any other cluster, of which i is not a member.

From the above definition it is clear that $-1 \leq s(i) \leq 1$.

The Silhouette index is the aritmetic mean of the silhouettes of each point in the dataset.

A value of the Silhouette index far from zero expresses a good result of the clustering algorithm.

C. Validation Measures for Fuzzy Clustering

Let $U = (u_{li})\ (1 \leq l \leq k, 1 \leq i \leq n)$ be the membership's matrix of a fuzzy partition of a dataset S with n records, and k is the number of clusters.

The first validity index for fuzzy clustering is the Partition Coefficient Index (PC) [34]. PC is based on U and it is defined as:

$$PC = \frac{1}{n}\sum_{l=1}^{k}\sum_{i=1}^{n} u_{li}^2$$

$PC \in [1/k, 1]$. Furthermore, a PC value close to $1/k$ indicates that clustering is "very fuzzy"; the value $1/k$ is obtained when $u_{li} = 1/k$, for each l, i.

Another index is the Partition Entropy Coefficient (PE):

$$PE = -\frac{1}{n}\sum_{l=1}^{k}\sum_{i=1}^{n} u_{li} log_a(u_{li})$$

$PE \in [0, log_a k]$. Furthermore, a low PE value indicates that clustering is "not very fuzzy". PE values close to the upper limit indicate an absence of any clustering structure within the dataset or the inability of the algorithm to extract it.

The main disadvantage of PC and PE is their monotonic evolution tendency with respect to k. To avoid this, a modification of the PC index can reduce the monotonic tendency and was defined by:

$$MPC = 1 - \frac{k}{k\ -\ 1}\ (1\ -\ PC)$$

where $0 \leq MPC \leq 1$.

Finally, let we define a novel validity index, which we call the ***Threshold Index TI***, by the following formula:

$$TI = \frac{|\underline{S}|}{|S|}$$

TI provides a measure of the quantity of elements in the dataset that are fixed in every partition (they belong to the floor set) with respect to the size of the whole dataset.

Finally, if o.FOU is the set of o-rank fuzzy outliers, then let we define the ***o-rank fuzzy outlier index*** (***o.FOUI***), by the following formula:

$$o.FOUI = \frac{|o.FOU|}{|S|}$$

$o.FOUI$ tells how many elements in the dataset are o-rank fuzzy outliers with respect to the size of the whole dataset.

Thanks to these validation measures, it is possible to have a rough idea of the fuzzy nature of the whole dataset and, thanks to $o.FOUI$, if too many points are outliers, then it is necessary to modify the parameters of the algorithm, such as k or N. For this reason, they are useful to better select these parameters.

VII. ECF-MEANS TOOL V2.0

With the purpose of testing the ECF-means algorithm, a software application has been designed and developed. It has been carried on using a Client/Server architectural pattern, where the Server part consists of the algorithm and other support utilities, while the Client part is made by a browser-based application, responsible of the ECF-means result visualization.

The algorithm had an important update and it leads to a new version of the software tool, and that is why we have renamed the tool updating it to the version 2.0. However, the tool presents an option to choose whether to keep the old version of the algorithm, leaving even the same configurations, or go to version 2.0, deleting the same partitions.

A. Software Implementation

The ECF-means web application is built up of two main modules: the first one wraps the ECF-means Algorithm that has been implemented in Java programming language, and it makes use of the Weka k-means algorithm (*SimpleKMeans*) [25] [35] as clustering algorithm implementation.

The second module consists of the web application client part, which has been implemented by using JavaScript libraries, such as D3.js, as visualization library, and jQuery for Ajax asynchronous data communication and Document Object Model (DOM) manipulation tasks.

B. GUI & Data Visualization

The implemented tool provides a user-friendly GUI, by which it is very easy to load datasets, fix the ECF-means parameters, and understand the algorithm results visually.

The GUI can be divided into three functional blocks, as highlighted by red numbered circle in Figure 6.

Through the first functional block, user can upload a dataset from a local file system, in csv or arff formats; after that he/she can specify the number of clusters k (default is set to two), the initial seed number (default 0), and the number of iterations N to perform (default 100).

The o Value spin box controls the o-rank fuzzy outlier value and hence the $o.FOUI$ index. Furthermore, its value changes cluster points shape and color: if a point has a difference between the two highest values of its probability membership vector less than this value, the point is displayed as a grey square.

Two checkboxes control computation of, respectively:

1. Silhouette Indexes.
2. Only different cluster configuration, as stated in Section V.E.

Lastly, a set of buttons allow the following operations:

1. Run: runs the ECF-means algorithm and displays the results (clustering graphical visualization and validation measures output).
2. Save Results: saves results to an output csv file.
3. Stop: ends the current computation.

The second block is where clustering visualization takes shape under scatter plot form: dataset points are displayed as circle with the color of the belonging cluster (resulting from the highest value of the probability membership vector) and with an opacity due to the degree of membership to the same cluster (stronger opacity means higher membership).

If the attributes of the dataset are two, Voronoi lines (computed considering initial seed, default $s = 0$) are also displayed.

In the top of the block, some input controls are used to affect data visualization. In particular, two combo boxes are used to allow the choosing of dataset's attributes that has to be displayed. Below this, a slider allows to set the degrees of membership above which a point is displayed (Membership Threshold t).

Instead, rightmost input fields, in order from top to bottom, control the visualization of:

1. The Initial Seed Centroid points (each with a "cross" symbol).
2. The Mean Seed Centroid points (each with a "plus" symbol).

Figure 6. Software GUI and clustering visualization.

The last third block is divided into three box panels: in the first one the validation measures are displayed, as described in Section VI.C, such as PC, PE, MPC, TI, and $o.FOUI$. In addition, Sum of Squared Errors (SSE) and Silhouette (S) measures have also been included; they are calculated considering Initial Seed (IS) and Mean Seed (MS), where MS is the mean value of the measure over all the N iterations, which lead to the definition of IS-SSE, MS-SSE, IS-S, and MS-S. Below this box, a second box displays the number of different retrieved partitions.

Lastly, in the bottom part of the block, a popup panel is displayed when a user clicks on a cluster point, where information about this clicked point are reported, such as point features, cluster memberships vector, best cluster assignment, etc.

C. Output Results

The ECF application exports results in csv format, where each row of the output file represents a point x of the dataset. The application appends ECF-means algorithm results as additional columns to the attributes columns of the point x.

TABLE V. COLUMN NAMES MEANING

Column Names	Description
ISCDistance$_i$, with $i = 1, \dots, k$	Vector of Euclidean Distances between point x and Initial Seed Centroids
ISCMembership	Cluster membership derived from the position of the smallest value in ISCDistance vector
MSCDistance$_i$, with $i = 1, \dots, k$	Vector of Euclidean Distances between point x and Mean Seed Centroids
MSCMembership	Cluster membership derived from the position of the smallest value in MSCDistance vector
Membership$_i$, with $i = 1, \dots, k$	Probability vector of point x, $p(x)$
ECFMembership	Cluster membership derived from the $PMA(x)$
o-rank fuzzy outlier	Y, if the point x is an *o*-rank fuzzy outlier, where o is fixed through the o Value input box by the user N, otherwise

Table V shows these additional column names meaning, where Mean Seed Centroid (MSC) is the arithmetic mean value of all computed centroids in N iterations.

VIII. CASE STUDY 1: THE WINE DATASET

The Wine dataset from the UCI Machine Learning Repository [36] is widely mined both by applying classification algorithms and clustering techniques. Chemical indicators are used in order to analyze the wine dataset. This case study is useful to underline the differences between classical methods for outlier detection and ECF-means.

A. Dataset Exploration

The Wine dataset has got 178 instances described by 13 attributes, with no missing values, and divided into three classes {0, 1, 2}, with the distribution [59, 71, 48]. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

TABLE VI. LIST OF WINE DATASET VARIABLES (FEATURES)

#	Name	#	Name
1	Alcohol	8	Nonflavanoid phenols
2	Malic acid	9	Proanthocyanins
3	Ash	10	Color intensity
4	Alcalinity of ash	11	Hue
5	Magnesium	12	OD280/OD315 of diluted wines
6	Total phenols	13	Proline
7	Flavanoids		

The attributes (variables) are listed in Table VI.

B. Outliers Detection with RapidMiner Tool

The RapidMiner tool [37] provides a series of case studies for data analysis by using machine learning techniques, and one of these concerns outlier detection in Wine dataset. The goal of these case study is to select anomalies in data resulting from a chemical analysis of wines by finding the data clusters and, then, identifying the anomalies based on local outlier factors. The clusters are achieved by applying the X-means algorithm [38] which determines the correct number of centroids based on a heuristic. Briefly, X -means is an extended version of k-means. It begins with a minimum set of centroids and then iteratively exploits if using more centroids makes sense according to the data. If a cluster is split into two sub-clusters is determined by the Bayesian Information Criteria (BIC), balancing the trade-off between precision and model complexity.

TABLE VII. X-MEANS RESULTS

Cluster	Number of Items
cluster_0	51
cluster_1	62
cluster_2	29
cluster_3	36
Tot	178

TABLE VIII. LIST OF OUTLIERS

#	Id	Cluster	Thershold
1	26	cluster_1	1.605
2	60	cluster_0	1.744
3	70	cluster_0	1.867
4	72	cluster_0	1.538
5	74	cluster_1	1.963
6	79	cluster_0	1.611
7	96	cluster_1	1.880
8	97	cluster_0	1.534
9	111	cluster_0	1.655
10	122	cluster_1	1.943
11	159	cluster_3	1.684

The outliers are detected by find the "outlier scoring" using the LOF (local outlier factor) mechanism [39]. The examples are filtered to get one data set with the outliers and another with the rest (non-anomalous points), using "outlier=1.5" as a threshold. Then, the outputs of the

RapidMiner analysis flow are the clustered data and two example sets with the outliers/non-outliers records.

The X-means algorithm provides 4 clusters, as reported in Table VII.

The outliers set, which we call OUT, provided by the RapidMiner flow, consists of 11 anomalies reported in Table VIII.

The set OUT of anomalies will be compared with the set of outliers obtained by the ECF-means algorithm, in order to discover if there is an o-rank threshold such that the two sets have some elements in common ($OUT \cap o.FOU \neq \emptyset$).

C. DBSCAN Algorithm Application

The mentioned DBSCAN algorithm can be applied in order to detect noise in the Wine dataset. Fixing $\varepsilon = 0.6$ and $m = 5$ in the Weka algorithm version, DBSCAN retrieves one cluster and provides 11 unclustered instances (noises), which are listed in Table IX, and they form the set NOI.

TABLE IX. LIST OF NOISES

#	Id	OS
1	60	YES
2	70	YES
3	74	YES
4	79	YES
5	96	YES
6	97	YES
7	111	YES
8	116	NO
9	122	YES
10	125	NO
11	159	YES

In the third column of Table IX, the "YES" value says if the unclustered instance is also in OUT set.

D. ECF-means Application

The ECF-means tool provides some different results. First of all, by changing k from 2 to 5, the best clustering outcomes are obtained by selecting $k = 3$. In Table X the validation measures are listed and they are calculated by fixing I = 500, which is the total number of performed iterations. #DP indicates the number of different partitions obtained during the 500 performed iterations.

TABLE X. VALIDATION MEASURES (I=500)

k	MPC	MS-S	TI	#DP
2	0.21	0.4	0.01	8
3	0.89	0.48	0.62	21
4	0.65	0.42	0.27	246
5	0.57	0.41	0	466

In spite of the fuzzy nature of the analyzed dataset (MPC=0.89), setting $k = 3$ the values of the Silhouette (MS-S = 0.48) and of the Threshold Index (TI = 0.62) are the highest of the list.

The floor $\underline{S}$ of Wine dataset has got 110 instances; these elements do not have a fuzzy nature (or a degree of "ambiguity"), and the first cluster has got only one element, the second one has got 58 elements, and the third one has got 51 elements. Set $S - \underline{S}$ contains potential outliers, and by changing the o parameter, different levels of o-rank fuzzy outliers are obtained, as Table XI shows.

TABLE XI. LIST OF o-RANK FUZZY OUTLIERS

o	o-$FOUI$	Id of o-rank fuzzy outliers
0.05	0.01	69
0.1	0.01	69
0.15	0.01	69, 74
0.25	0.02	69, 72, 74
0.35	0.03	61, 69, 71, 72, 74, 75

$p(69) = (0.5238, 0, 0.4762)$ is the probability vector of the element with Id = 69, whilst the probability vector of 74 is $p(74) = (0.4286, 0.5714, 0)$. The element 72, which is a 0.25-rank fuzzy outlier is not a real outlier because it is not an ambiguous element, considering its probability vector $p(72) = (0.619, 0.381, 0)$.

The detected 0.15-rank fuzzy outliers, discovered by the ECF-means algorithm, listed in Table XI, form the 0.15-$FOU = \{69, 74\}$ set.

The intersection set $OUT \cap NOI \cap 0.15.FOU$ is $\{74\}$ and this point does not represent anything new.

However, the new detected point, marked with id = 69, is really halfway between the first cluster and the third one and its discovery makes us understand how the proposed ECF-means method is able to enrich our knowledge on the analyzed dataset. Moreover, this point is found also by the RapidMiner outlier detection process, using "outlier=1.3" as threshold of the selecting filter, instead of 1.5.

IX. CASE STUDY 2 IN METEOROLOGICAL DOMAIN

The application of the ECF-means algorithm to the meteorological dataset [40] has been extensively presented in [1], where numerous results have been described. To deepen the nature of the dataset and to have an example of data analysis, you can consult [41].

This case study is useful for explaining how the algorithm is able to explore even datasets with numerous records, how the number of discovered outliers increases with the increase of the o parameter, and how the "ambiguous" points also belong to overlapping geographical areas.

A. Summary of the Previous Results

Briefly, in this section a summary of the results of the ECF-means application is presented.

An historical dataset made up of 9200 meteorological observations has been collected. Data have been retrieved from ECMWF MARS Archive containing the surface Synoptic observations (SYNOP) provided by 4 geographical sites: Charles De Gaulle (CDG) airport in Paris and Grazzanise, Milan, and Pantelleria airports in Italy.

SYNOP observations are recorded every hour and the list of the meteorological variables [41] used for applying the ECF-means algorithm is reported in Table XII. Each airport site has got 2300 records and the SITE attribute has 4 values.

Considering the Silhouette measure, the best clustering partition is obtained by selecting $k = 3$ for k -means

algorithm (Silhouette=0.49). Fixing $k = 3$ and considering SITE attribute as Class attribute, CDG and Milan are inserted into the same cluster (Cluster 0) by the algorithm (4 sites in 3 clusters): it seems that the two sites have a lot in common! Thus, we try to merge these two sets, obtaining a new set called CDG+MIL.

TABLE XII. LIST OF METEOROLOGICAL VARIABLES (FEATURES)

#	Name	#	Name
1	Pressure	6	cloud cover
2	three-hour pressure change	7	height of base of cloud
3	wind direction	8	Dewpoint Temperature
4	wind speed	9	Drybulb Temperature
5	Visibility	10	SITE

The incorrectly clustered instances are 3710 and represent 40.32% of the original dataset. k-means does not provide homogeneous clusters with respect to SITE attribute. From an intuitive point of view, the 3 sites (Grazzanise, Pantelleria and CDG+MIL) have an ambiguous meteorological nature and the 3710 unclustered instances are on the border between two or more sites. In other words, the datasets have overlapping areas, with "similar" meteorological conditions, and perhaps the sites are not so different, and they are not well-separated from each other.

The ECF-means algorithm tries to overcome the problem in which the k-means algorithm falls in this meteorological case study, and in part it succeeds, if only because it provides much more information on datasets, clusters, and on clustering results, thanks to which the data analyst can make more informed and useful choices.

By fixing $k = 3$, thanks to the ECF-means application, we are able to select the floor $\underline{S}$ of whole dataset. It has got 5363 records. The incorrectly clustered instances are 704 and represent 13.12% of $\underline{S}$.

The elements belonging to $\underline{S}$ never fluctuate from one cluster to another (considering the 350 iterations) and constitute approximately 58.3% of the initial dataset.

The obtained results lead to a clear improvement of the clustering: the clusters seem much more separate, if the contingency matrices are calculated starting from the floor set. ECF-means manages to break down the percentage of instances that are incorrectly clustered from 40.32% to 13.12%.

B. Outlier Detection by Applyng the InterquartileRange

The Weka tool provides a filter for detecting outliers and extreme values based on interquartile ranges. It can be found among the filters in the preprocess tab. The filter complies with the following schema:

A point x is an outlier ⇔

$$\begin{cases} Q_3 + OF * IQR < x < Q_3 + EVF * IQR \\ or \\ Q_1 - EVF * IQR \leq x < Q_1 - OF * IQR \end{cases}$$

A point x is an extreme value ⇔

$$\begin{cases} x > Q_3 + EVF * IQR \\ or \\ x < Q_1 - EVF * IQR \end{cases}$$

Where:

- $Q_1 = 25\%$ quartile,
- $Q_3 = 75\%$ quartile,
- IQR = Interquartile Range, difference between Q_1 and Q_3,
- OF = Outlier Factor,
- EVF = Extreme Value Factor.

The filter adds two new columns to the dataset: *Outlier* and *ExtremeValue*, which assume the values "yes" or "no", depending on the previous filter schema.

The dataset is divided by the three sites: Grazzanise, Pantelleria and CDG+MIL, and then, fixing $OF = 3.0$ and $EVF = 6.0$, the filter is applied to the three geographical sites separately.

Table XIII shows the number and the percentages of the outliers and of the extreme values discovered by the Weka filter.

TABLE XIII. SYNOP OUTLIERS AND EXTREME VALUES

	Grazzanise	Pantelleria	CDG+MIL
Outlier	144 (6.3%)	267 (11.6%)	319 (6.9%)
Extreme Values	65 (2.8%)	141 (6.1%)	157 (3.4%)

These values are statistical outliers and they depend on the distributions of all the meteorological variables.

By removing these points, putting together the three subsets, the residual dataset has got 8365 instances (some elements are both outliers and extreme values), and the three sites have the distribution [2145, 2012, 4208].

C. Outlier Detection by Applyng ECF-means

It is very useful to make a classification of the o-rank fuzzy outliers of the SYNOP dataset. As we already said, from the "o-rank fuzzy outlier" perspective, each point in the dataset is an outlier. The elements that belong to the $\underline{S}$, for example, are 1-rank fuzzy outliers but aren't o-rank fuzzy outlier, with $o < 1$.

First of all, the intersection between the set OUT of 835 detected outliers by applying the InterquartileRange filter and the set of instances that are in S and not in $\underline{S}$ has got 748 elements:

$$\left|OUT \cap (S - \underline{S})\right| = 748$$

This makes us realize how the anomalies discovered by the ECF method have much in common with the statistical outliers that are in the OUT set.

The elements belong to $o.FOU$ are o-rank fuzzy outliers and they have a fuzzy nature. However, o-rank fuzzy outliers with small o ($o < 1.5$) are the most interesting ones, and we will compare them with the OUT elements in more detail. So, also in this case study we will determine the set $OUT \cap o.FOU$ ($|OUT \cap o.FOU| \leq 748$, for small values of o).

By changing the o parameter (and fixing $k = 3$), we have the results of Table XIV, where the fuzzy outliers are divided with respect to the SITE attribute.

TABLE XIV. SYNOP o-RANK FUZZY OUTLIERS

o	Grazzanise	Pantelleria	CDG+MIL	Tot
0.05	7	22	25	54 (0.6%)
0.1	41	27	63	131 (1.4%)
0.15	145	204	248	597 (6.5%)
0.2	328	454	587	1369 (14.9%)

We found that 597 points (about 6.5% of the initial dataset) have an Outlier Threshold (difference between the two highest values of the probability membership vector), less than 0.15. These ambiguous instances belong also to OUT set:

$$|OUT \cap 0.15.FOU| = 597$$

The remaining $748 - 597 = 151$ points, which are in OUT but do not belong to $0.15.FOU$ set, are all 0.2-rank fuzzy outliers.

D. Invariance Property

Is the ECF-means method for outlier detection invariant for the SYNOP dataset? Or better, is there an o value for which the set of o -rank fuzzy outliers of $(SYNOP - o.FOU) = \emptyset$?

The tests we performed, by changing the o parameter, showed us that ECF-means is not an invariant algorithm for SYNOP set. But fixing $o = 0.15$, the set of the 0.15-rank fuzzy outliers of $(SYNOP - 0.15.FOU)$ is very small. Indeed, this set has got only 59 instances, which are new points but, deepening this set, we have discovered that these elements belong to the $0.2.FOU$ set.

For the sake of study, we repeated the procedure by setting $o = 0.2$, and then discovering the 0.2-rank fuzzy outliers of $(SYNOP - 0.2.FOU)$. Also, in this case, the resulting set is very small and it has got 40 instances.

On the other hand, if we calculate the o-rank fuzzy outliers of the floor $\underline{S}$ of the SYNOP set (we are applying the ECF-means algorithm to $\underline{S}$ that has got 5363 records) we find out that $0.15.FOU$ is empty and $0.1.FOU$ has got only 11 elements, whilst $0.05.FOU$ has got 27 records (0.5% of $\underline{S}$), and this result is really amazing.

E. Experimental Results

The results obtained in this case study encourage us to reflect a lot. Unlike the previous case study, here we do not find any new points compared to those discovered by the traditional method based on statistics. But surely the most interesting result is the ability that the proposed method has to assign a score to each set of outliers.

Moreover, this case study gave us the opportunity to investigate the invariance property of the ECF-means method and to research the connection between the floor set of SYNOP and the o-rank fuzzy outliers. Although we did not find a precise analytical relationship between floor of SYNOP and o (and we do not believe it exists because of the many algorithm parameters involved), we discovered a method to reduce the number of o-rank fuzzy outliers and to determine a "stable" subset of the original dataset.

SYNOP is a very difficult set to analyze, not only because it has many statistical outliers (about 9.1% of the original dataset), but also because, as repeatedly pointed out by the meteorologist and as also discovered in [1] and in [41], there are no net separations between airport areas, from a weather point of view. For example, the elements in $0.15.FOU$ are very fuzzy points and they belong to more than one cluster, and probably to more than one airport site (to overlapping areas).

In [41] the dataset is mined in order to find short-range temporal models for fog prediction; a next step would be to find out if prediction errors belong to $o.FOU$, for some o; or, if the generic forecast error decreases by removing the detected outliers from the dataset, or analyzing only the floor $\underline{S}$ of SYNOP dataset.

X. CASE STUDY 3: THE IRIS DATASET

The famous Iris dataset is a multivariate dataset that contains 3 classes of 50 instances each, where each class refers to a species of iris plant (Iris-setosa, Iris-virginica, and Iris-versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

The use of this dataset is very common in classification and clustering tasks, where numerous results have been obtained. Moreover, historically this dataset has been one the driving force of many theoretical studies, above all to discover and to deepen many non-linear classification methods.

From a statistical point of view, the Iris dataset has neither outliers nor extreme values; this makes the dataset a good example for exploration by the ECF-means algorithm, in order to discover new kind of outliers.

A. Summary of k-means Results

The data set only contains two clusters with rather obvious separation: one of the clusters contains Iris-setosa, while the other cluster contains both Iris-virginica and Iris-versicolor.

Fixing $k = 3$, the incorrectly clustered instances are 18, as reported in the contingency matrix of Table XV, and represent 12% of the original Iris dataset, by applying the simple k-means algorithm (or by applying the ECF-means algorithm, fixing $s = 0$ and $I = 1$).

TABLE XV. CLASSES TO CLUSTERS ($s = 0$ AND $I = 1$)

0	1	2	Assigned to cluster
0	0	50	Iris-setosa → Cluster 2
40	10	0	Iris-versicolor → Cluster 0
8	42	0	Iris-virginica → Cluster 1
32%	35%	33%	

All the records of the Iris dataset belonging to the Iris setosa class come together in a single and homogeneous cluster. No point in this cluster shows an ambiguous or fuzzy nature, so none of these points is an outlier. The fuzzy outliers of the dataset are, therefore, to be searched among the records labeled by versicolor tag or virginica one.

B. ECF-means Algorithm Application

Fixing $k = 3$ and choosing the number of iterations $I = 7500$, the performed iterations generate the validity indexes and the statistics of Table XVI, where the o parameter is set to 0.1.

TABLE XVI. VALIDITY INDEXES (7500 ITERATIONS)

Index	Value	Descriptive note
MPC	0.73	If the Normalized Partition Coefficient Index is 1, then the clustering is not fuzzy
Silhouette Index	0.57	An everage Silhouette greater than 0.5 indicates reasonable partitioning of data
TI	0.5	50% of the instances belong to the floor of the Iris dataset; then, the other 50% of the instances have a fuzzy nature
No. of $0.1.FOU$	13	13 elements have $o \leq 0.1$ and they are 0.1-rank fuzzy outliers. From the results of the next tables $o \leq 0.057$
$0.1.FOUI$	0.0867	13/150=0.0867
No. of different partitions	6	The 7500 performed Iterations generate 6 different partitions. The first seeds (s) generating these partitions are: 0, 1, 31, 98, 1794, 5894

EFC-means provides the results in the contingency matrix of Table XVII, where the incorrectly clustered instances are 17 and represent 11.33% of the original Iris dataset.

TABLE XVII. CLASSES TO CLUSTERS ($k = 3,\ I = 7500$)

0	1	2	Assigned to cluster
0	0	50	Iris-setosa→ Cluster 2
47	3	0	Iris-versicolor → Cluster 0
14	36	0	Iris-virginica → Cluster 1
41%	26%	33%	

Thanks to the ECF-means application, we are able to select the floor $\underline{S}$ of whole Iris dataset S. $\underline{S}$ has got 75 elements (TI = 0.5) that have the distributions in Table XVIII.

TABLE XVIII. CLASSES TO CLUSTERS ($\underline{S}$)

0	1	2	Assigned to cluster
0	0	32	Iris-setosa→ Cluster 2
4	3	0	Iris-versicolor → Cluster 0
0	36	0	Iris-virginica → Cluster 1
5.33%	52%	42.67%	

$\underline{S}$ has got 3 incorrected clustered instances that represent 4% of $\underline{S}$.

In conclusion, if $t = 1$, then $|C_0^1| = 4$, $|C_1^1| = 39$, and $|C_2^1| = 32$.

C. Outlier Detection by Applyng ECF-means

The 50 elements in Table XVII tagged by Iris-setosa class are divided in 4 subsets, depending on their probability vectors, which are in Table XIX. The last column of the table contains the difference between the two highest probabilities of the vector (the o parameter).

The 32 elements in the A1 subset belong to the floor of Cluster_2 (Table XVIII) and, therefore, to the $\underline{S}$ set. Although the elements in B1, C1, and D1 do not belong to $\underline{S}$, their probability of belonging to the Cluster_2 is very high (around 79.6%).

TABLE XIX. IRIS-SETOSA PROBABILITY VECTORS (7500 ITERATIONS)

	N. of elements	Clus_0	Clus_1	Clus_2	o
A1	32	0	0	1	1
B1	13	0.204	0	0.796	0.592
C1	4	0.2035	0	0.7965	0.593
D1	1	0.2037	0	0.7963	0.5926

In conclusion, the elements tagged by Iris-setosa class label form the Cluster_2. This result, as we know, does not surprise us, and confirms the goodness of the ECF-means algorithm.

The results obtained by analyzing the other floral classes seem to be much more interesting.

The 50 Iris-versicolor elements have the probability vectors of Table XX. These elements fluctuate between the Cluster_0 and the Cluster_1, and no element is in Cluster_2.

TABLE XX. IRIS-VERSICOLOR PROBABILITY VECTORS (7500 ITERATIONS)

	N. of elements	Clus_0	Clus_1	Clus_2	o
A2	34	0.796	0.204	0	0.592
B2	6	0.5231	0.4769	0	0.0462
C2	4	1	0	0	1
D2	3	0	1	0	1
E2	2	0.7965	0.2035	0	0.593
F2	1	0.5285	0.4715	0	0.057

The elements of A2, C2, and E2 belongs to Cluster_0, without any doubt. In particular, the 4 elements of C2 are in the floor of Cluster_0.

Surely, by analyzing the table, the instances marked by the Iris-versicolor class label are mainly found in Cluster_0, and the floor of this cluster has got only 4 elements, and then they belong to the $\underline{S}$ set.

The 3 elements in D2 are put in the floor of the Cluster_1, but they have the Iris-versicolor label class, and then they are unclustered instances.

The 7 elements belonging to B2∪F2 are 0.057-rank fuzzy outliers, even if their probabilities of belonging to Cluster_0 are greater than the probabilities of belonging to Cluster_1.

The 50 Iris-virginica elements have the probability vectors of Table XXI. Also, these elements fluctuate between the Cluster_0 and the Cluster_1.

In this case, the floor of the Cluster_1 has got 36 instances (to which are added the 3 instances tagged by Iris-versicolor label, retrieved in the Iris-versicolor case, Table XX).

TABLE XXI. IRIS-VIRGINICA PROBABILITY VECTORS (7500 ITERATIONS)

	N. of elements	Clus_0	Clus_1	Clus_2	o
A3	36	0	1	0	1
B3	8	0.796	0.204	0	0.592
C3	6	0.5231	0.4769	0	0.0462

The 8 points of B3 seem to belong to Cluster_0 and then they are unclustered instances.

The 6 elements in C3 are, without any doubt, 0.0462-rank fuzzy outliers.

TABLE XXII. 0.057-RANK FUZZY OUTLIERS OF IRIS DATASET

	El. Num.	Sepal length	Sepal width	Petal length	Petal width
1	52	6.4	3.2	4.5	1.5
2	57	6.3	3.3	4.7	1.6
3	66	6.7	3.1	4.4	1.4
4	71	5.9	3.2	4.8	1.8
5	77	6.8	2.8	4.8	1.4
6	86	6	3.4	4.5	1.6
7	87	6.7	3.1	4.7	1.5
8	124	6.3	2.7	4.9	1.8
9	127	6.2	2.8	4.8	1.8
10	128	6.1	3	4.9	1.8
11	139	6	3	4.8	1.8
12	147	6.3	2.5	5	1.9
13	150	5.9	3	5.1	1.8

Choosing the maximum o in the previous Table IX, Table X, and Table XI, for which the elements are fuzzy outliers, the Iris dataset has got 13 elements that are 0.057-rank fuzzy outliers (B2∪F2∪C3), and they are presented in Table XXII.

D. Invariance Property

Also for Iris dataset we study the property of invariance. First of all, in summary, from the previous application of ECF-means algorithm, we have:

- $|0.057.FOU| = 13$,
- $|Iris - 0.057.FOU| = 137$,
- $|\underline{Iris}| = 75$.

The ECF-means method is invariant for the Iris dataset. That is to say, the set of 0.057-rank fuzzy outliers of $(Iris - 0.057.FOU) = \emptyset$. Not only that, the smallest o value for which we have o-rank fuzzy outliers is $o = 0.552$.

Furthermore, if we consider $\underline{Iris}$, the set of o-rank fuzzy outliers of $\underline{Iris}$ is empty for any value of o such that $o \leq 0.246$. Then, for this case study, $\underline{\underline{Iris}} \neq \underline{Iris}$, but $|\underline{\underline{Cluster\_0}}| = 2$, $\underline{\underline{Cluster\_1}} = \emptyset$, and $\underline{\underline{Cluster\_2}} = \underline{Cluster\_2}$, which confirms the elements labeled by the Iris-setosa class are still preserved by the algorithm.

E. Experimental Results

The 13 0.057-rank fuzzy outliers are labeled by the Iris-versicolor class or by the Iris-virginica one, and not by Iris-setosa class, as reported in the scatter plot chart of sepalwidth, petalwidth space in Figure 7 (left part), where they are marked by boxes in grey color. In particular, 7 fuzzy outliers have the Iris-versicolor class and 6 the Iris-virginica class. They belong to $S - \underline{S}$ and can be analyzed separately in order to understand their fuzzy nature.

Moreover, there are 11 unclustered instances (D2∪C3), but the elements in D2 belong to the floor of the Cluster_1. Finally, the floor of Iris dataset has got 75 points (A1∪C2∪D2∪A3), as reported in Figure 7 (right part); the floor of a set can be easily achieved by scrolling the "Membership Threshold" slider up to 100% in the graphical user interface of the tool, because the floor has got only the elements with the probability vectors equal to $(1, 0, 0)$, or to $(0, 1, 0)$, or to $(0, 0, 1)$.

The charts also show the centroids of the three clusters. The "cross" symbol indicates the Initial Seed Centroid points, whilst the "plus" symbol are the Mean Seed Centroid points. In each cluster, the farther away are the "cross" and the "plus" symbols, the more points are fuzzy (and, therefore, fewer points belong to the floor of the considered cluster).

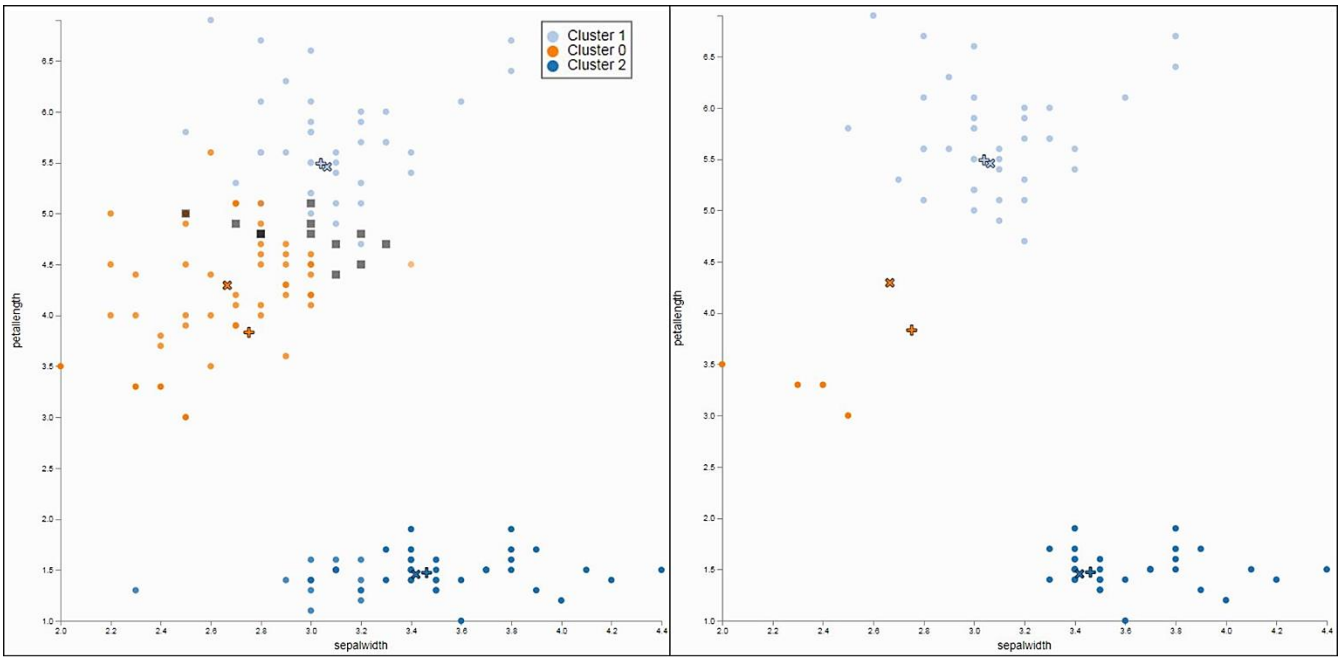

Figure 7. Scatter plot chart of S with 0.057-rank fuzzy outliers in grey (on the left) and $\underline{S}$ scatter plot chart (on the right).

Thanks to the obtained results, we can easily understand how the algorithm is able to optimize the partitioning of the data space with respect to the class that expresses the floral typology. The algorithm is able to find this partitioning in one fell swoop. By applying the simple k-means we may not be able to get the same partition. However, the most interesting result is that the algorithm is able to preserve the cluster with Iris-setosa label and to find the floating elements that are at the limits of the floral types.

These "disturbing" elements can be analyzed separately in order to understand if they are, from some point of view, outliers or records that have undergone measurement errors.

The arithmetic mean of the 13 discovered fuzzy outliers is, for example, $\mu_{13} = (6.277, 3.008, 4.761, 1.67)$ and the arithmetic mean of the distances of the 13 points from μ_{13} is 0.441. Moreover, if you subdivide the 13 points into two subsets, taking into account the two values of the target class, the 7 records with Iris-versicolor value have $\mu_7 = (6.4, 3.157, 4.628, 1.543)$ and the arithmetic mean of the distances of the 7 points from μ_7 is 0.389; the 6 records with Iris-virginica value have $\mu_6 = (6.133, 2.833, 4.917, 1.817)$ and the arithmetic mean of the distances of these 6 points from μ_6 is 0.25.

These measures give us an idea of how the outliers are close to each other, and how they have a low dispersion around the average.

XI. Conclusion and Future Works

In this work we have presented a procedure for detecting outliers or anomalies in large datasets, using the ECF-means algorithm, which is an ensemble method useful for optimizing and "fuzzifing" cluster analysis results. After describing the various types of outliers and the main approaches of data mining for their discovery, in this paper we defined a new category of outliers, called o-rank fuzzy outliers, based on the different degrees of membership of the point to the various obtained clusters.

By using an ad hoc implemented software application and the Weka version of the well-known k-means algorithm, three case studies have been proposed, aimed at showing the strengths of the ECF-means ensemble approach.

In particular, with the tool it is possible:

1. optimize the choice of algorithm parameters, such as the number k of the clusters to be determined, by calculating validation metrics such as the Silhouette, the partition entropy coefficient, and two new metrics that are the threshold index (TI) and the o-rank fuzzy outlier index (o.FOUI), which have been defined for the first time in this work. These new indexes give us a rough estimate of the amount of outliers of the dataset;
2. calculate a degree of membership of each point to each cluster;
3. select the o-rank fuzzy outliers of the dataset as those points that have a fixed level of ambiguity, and also choose the level;
4. view the obtained results using a simple and interactive graphical interface.

The presented case studies showed that the ECF-means algorithm is able to detect more and different anomalies than those discovered by the usual outlier detection methods: these new points are elements that have a high level of ambiguity and which must be subjected to a more in-depth analysis. Being able to assign a level or score of ambiguity to each point is certainly one of the strengths of the proposed method, which is a method for exploring data.

However, we had that all the most exciting results can be obtained by the active interaction with the software tool interface, thanks to which, by scrolling the sliders, changing parameters, and visualizing groupings, numerous properties of the dataset can be discovered.

In future works, we are going to evaluate our method on other several datasets, for example, on others from UCI ML Repository.

Future investigations about the algorithm:

1. to find the relationship between the clusters centroids calculated considering the initial seed ($s = 0$) (the "cross" symbols in the GUI) and those obtained considering the means of the centroids computed over all the N iterations (the "plus" symbols in the GUI), as it was stated in the case study of the Iris dataset;
2. to understand the relationship between the distance among the cross and the plus centroids and the floor of each cluster;
3. to discover a criterion for the o parameter optimization in relation to the other parameters, such as k and N.

Additionally, in order to understand the final configuration, the floor of the analyzed dataset, and the discovered fuzzy outliers, why not train a model by using a machine learning technique, able to predict the probability vector of a point in a test set? The model could be a multivariate and regressive classifier with $k - 1$ output variables, useful to know the level of "ambiguity" of a new and not yet clustered observation.

For the current study, we have chosen the simple k-means as the reference clustering algorithm. Furthermore, we can consider other algorithms in substitution or in addition to it, and this will surely be one of the next improvement of the tool.

Another information that can be very useful to the data analyst is the frequencies of the partitions that are obtained by varying the seed s. This feature, and other statistical computations, will also be added to the next tool update.

Acknowledgment

The authors would mention the TECVOL II and the Big Data Facility projects, both funded by the Italian PRORA, in which the meteorological database has been realized and the tool has been designed and developed.

References

[1] G. Zazzaro and A. Martone, "ECF-means – Ensemble Clustering Fuzzification Means," Proc. of the Eighth International Conference on Advances in Information Mining and Management (IMMM), pp. 20-27, 2018.

[2] S. Vijendra and P. Shivani, "Robust Outlier Detection Technique in Data Mining: A Univariate Approach," arXiv Preprint arXiv:1406.5074, [Online] available from: http://arxiv.org/abs/1406.5074, 2019.01.25.

[3] A. Zimeck and P. Filzmoser, "There and back again: Outlier detection between statistical reasoning and data mining algorithms," WIREs Data Mining Knowl Discov. 8:e1280, Wiley Periodicals, Inc., 2018.

[4] P. Tan, M. Steinbach, and V. Kumar, "Introduction to Data Mining", Pearson Addison Wesley, 2005.

[5] D. Hawkins, "Identification of Outliers," Chap. and Hall, 1980.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection," in Encyclopedia of Machine Learning and Data Mining. Second Edition, Springer, 2017.

[7] S. Agrawal and J. Agrawal, "Survey on Anomaly Detection using Data Mining Techniques," Proc. of 19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems, Elsevier, 2015.

[8] C. C. Aggarwal, "Outlier Analysis. 2nd Ed.," Springer, 2017.

[9] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Dicovering Clusters in Large Spatial Databases with Noise," Proc. Of the 2nd Intl. Conf. on Knowledge Discovery and Data Mining, AAAI Press, pp. 226-231, August 1996.

[10] P. Chapman et al., "CRISP-DM 1.0. Data Mining guide," 2000.

[11] K. Singh and S. Upadhyaya, "Outlier Detection: Applications And Techniques," IJCSI International Journal of Computer Science Issues, vol. 9, Issue 1, no. 3, pp. 307-323, January 2012.

[12] Z. Niu, S. Shi, J. Sun, and X. He, "A Survey of Outlier Detection Methodologies and Their Applications," AICI 2011, Part I, LNAI 7002, pp. 380-387, 2011.

[13] H. Izakian and W. Pedrycz, "Anomaly Detection in Time Series Data using a Fuzzy C-Means Clustering," in IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint.

[14] B. S. Harish and S. V. Aruna Kumar, "Anomaly based Intrusion Detection using Modified Fuzzy Clustering," International Journal of Interactive Multimedia amd Artificial Intelligence, vol. 4, no. 6, 2017.

[15] L. Xie, Y. Wang, L. Chen, and G. Yue, "An Anomaly Detection Method Based on Fuzzy C-means Clustering Algorithm," Proceedings of the Second International Symposion on Networking and Network Security (ISNNS'10), 2010.

[16] R. Sampat and S. Sonowani, "A Survey of Fuzzy Clustering Techniques for Intrusion Detection System," International Journal of Engineering Research & Technology (IJERT), vol. 3 Issue 1, 2014.

[17] Z. Zhao, C. K. Mohan, and K. G. Mehrotra, "Adaptive Sampling and Learning for Unsupervised Outlier Detection," Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference, 2016.

[18] B. Micenková, B. McWilliams, and I. Assent, "Learning Outlier Ensembles: The Best of Both Worlds – Supervised and Unsupervised," Proc. of the ODD^2 workshop organized together with KDD'14, 2014.

[19] P. Berkhin, "Survey of clustering data mining techniques," Technical report, Accrue Software, San Jose, CA, 2002.

[20] A. K. Jain, A. Topchy, M. H. C. Law, and J. M. Buhmann, "Landscape of Clustering Algorithms," IAPR International Conference on Pattern Recognition, vol. 1, pp. 260-263, Cambridge, UK, 2004.

[21] L. I. Kuncheva, "Combining Pattern Classifiers. Methods and Algorithms," Wiley-Interscience, A John Wiley & Sons, Inc., Publication, 2004.

[22] A. Strehl and J. Ghosh, "Cluster Ensembles – A Knowledge Reuse Framework for Combining Multiple Partitions," Journal of Machine Learning Research 3 (2002), pp. 583-617.

[23] S. Monti, P. Tamayo, J. Meisirov, and T. Golub, "Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data," Machine Learning, 52, pp. 91–118, 2003, Kluwer Academic Publishers.

[24] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering Aggregation," on 21st International Conference on Data Engineering (ICDE), pp. 341-352, 2005.

[25] R. R. Bouckaert et al., "WEKA Manual for Version 3-9-2," the University of Waikato, Hamilton, New Zealand, December, 2017.

[26] E. Frank, M. A. Hall, and I. H. Witten, "Data Mining: Practical Machine Learning Tools and Techniques," Morgan Kaufmann, 2016.

[27] G. Gan, C. Ma, and J. Wu, "Data Clustering. Theory, Algorithms, and Applications," ASA-SIAM Series on Statistics and Applied Probability, SIAM, Philadelphia, ASA, Alexandria, VA, 2007.

[28] C. C. Aggarwal, "Data Mining. The Textbook," Springer International Publishing, 2015.

[29] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The Fuzzy c-Means Clustering Algorithm," Computers & Geosciences Vol. 10, no. 2-3, pp. 191-203, 1984.

[30] J. Wu, "Advances in k-means Clustering. A Data Mining Thinking," Springer, 2012.

[31] S. Das, A. Abraham, and A. Konar, "Metaheuristic Clustering," Studies in Computational Intelligence, vol. 178, Springer, 2009.

[32] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," Journal of Computational and Applied Mathematics 20, 53-65, 1987.

[33] C. Subbalakshmi, G. R. Krishna, S. K. M. Rao, and P. V. Rao, "A Method to Find Optimum Number of Clusters Based on Fuzzy Silhouette on Dynamic Data Set," International Conference on Information and Communication Technologies (ICICT 2014), Procedia Computer Science 46, 346-353, Elsevier, 2015.

[34] R. N. Dave, "Validating fuzzy partition obtained through c-shells clustering," Pattern Recog. Lett. 17, pp. 613–623, 1996.

[35] M. Hall et al., "The WEKA Data Mining Software: An Update," SIGKDD Explorations, vol. 11, Issue 1, 2009.

[36] D. Dua and E. K. Taniskidou, UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science, 2017.

[37] M. Hofmann and R. Klinkenberg, "RapidMiner: Data Mining Use Cases and Business Analytics Applications," Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, CRC Press, October 25, 2013.

[38] D. Pelleg and A. W. Moore, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters," in Seventeenth International Conference on Machine Learning, 727-734, 2000.

[39] M. M. Breuning, H-P Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," in Proc. Of the 2000 ACM SIGMOD, pp. 93-104, ACM, USA, 2000.

[40] ECMWF, Mars User Guide. User Support. Operations Dep. 2013.

[41] G. Zazzaro, G. Romano, and P. Mercogliano, "Data Mining to Forecasting Fog Events and Comparing Geographical Sites. Designing a novel method for predictive models portability," Int. Journal on Advances in Nets and Services, vol. 10, no. 3 & 4, pp. 160-171, 2017.