Explainable Distance-based Outlier Detection in Data Streams

THEODOROS TOLIOPOULOS\textsuperscript{1} AND ANASTASIOS GOUNARIS\textsuperscript{2} (Senior Member, IEEE)

\textsuperscript{1}Department of Informatics, Aristotle University of Thessaloniki, Greece (e-mail: tatoliop@csd.auth.gr)
\textsuperscript{2}Department of Informatics, Aristotle University of Thessaloniki, Greece (e-mail: gounaria@csd.auth.gr)

Corresponding author: Theodoros Toliopoulos (e-mail: tatoliop@csd.auth.gr).

This research work has been supported by the European Commission under the Horizon 2020 Programme, through funding of the RAINBOW project (Grant 871403).

I. INTRODUCTION

Real-time analytics is increasingly important in supporting every-day activities in modern organizations and companies, given also the ever-growing data streams stemming from numerous and diverse sources, such as social networks, smartphone apps and usage logs. Outlier (or anomaly) detection, which aims at identifying noise as well as anomalies and events of interest, is a key element in such analytics tasks [1]. Among the broad variety of the existing proposals, distance-based outlier detection algorithms [2] are commonly used in practice [3]. This category of proposals relies on counting the number of neighbors within a range $R$ of each point and employs a cut-off threshold $k$ to distinguish between inliers and outliers. Their broad applicability is mainly due to two key advantages, namely the totally unsupervised nature, that is there is no need for model training, and the amenability to streaming applications, as evidenced also by the persistence of the research interest in this topic, e.g., [4]–[7].

Nowadays, emphasis is also placed on explainable analytics, which, in the case of outlier detection, is closely related to the exact type of output of the various categories of techniques. Many unsupervised outlier detection techniques, such as density-based [8], [9], angle-based [10] and isolation-based [11] ones, output a score representing the degree of outlierness of each returned anomalous point. Further to that, LOCI [9] uses the resulting score to create more comprehensive plots that assist in classifying outliers as part of a micro-cluster or a cluster or as an isolated (outstanding) point. Also, many algorithms are proposed that detect the subspaces that contribute the most to the outlier score [12]–[15]. For example, the algorithms proposed in [12], [13], return the low-dimensionality subspaces (2d or 3d) that are the most important for each outlier. In addition, the algorithms in [14], [15], instead on focusing on each outlier data point independently, try to explain the whole set of outliers by presenting the subspaces that are deemed as responsible for most of them.

On the other hand, distance-based algorithms, especially those targeting a stream, transform the outlier detection problem into a binary process where a data point is either an outlier or an inlier [4]–[7], [16]–[20]. This does not provide any insights into the reason behind the outlierness of a data point besides its "isolation" from the rest of the points in a possibly high-dimensional dataset and this lack of explainability is aggravated by the known sensitivity of the results to the setting of the $R$ and $k$ parameters. In the broader field of machine learning, many model-agnostic explainability techniques, that emphasize on the notion of interpretable local fidelity, have been developed to tackle such cases. A key representative of such an explainer is LIME [21], which provides results that indicate possible features of importance. For example, Figure 1 presents the output of LIME for 4 outliers in a 2-dimensional dataset. As shown, the algorithm outputs different features of importance for differ-
Ideally, explainable distance-based outlier detection should reduce the costs of the factory. Any points, e.g., for the first outlier only the first dimension is important whilst, for the second one both dimensions play a role to some extent. However, in a distance-based setting, these explanations do not reveal the interpretable relationship between a point and its neighbors.

Understanding the relationship of a data point with the rest of the points in its vicinity can help in identifying different explanations on the reason behind its outlierness. For example, in a smart industry setting, every stage of a product’s pipeline has automated quality checks that identify different types of faults as outliers. Assume that a product’s size is different, meaning that, in a distance-based setting, it will not have enough neighbors and will be identified as faulty. In this case, the product is discarded and the production will continue as normal. But if the faulty size is part of or close to a cluster of similar values, this will eventually result in more outliers and pauses of the production line. Identifying as early as possible, in real-time, that other products (in the vicinity of the outlier) have potential faults can contribute to reduce the costs of the factory.

In Figure 2, we depict 6 classes, based on the vicinity of a data point, to which the point may belong: (1) Isolated point; (2) Point near a cluster; (3) Dense cluster; (4) Sparse cluster; (5) Dense micro-cluster; and (6) Sparse micro-cluster. Ideally, explainable distance-based outlier detection should be capable of annotating the results for each data point with one of these classes, although the focus is on labeling the reported outliers solely. This is exactly our contribution, which goes beyond simply identifying relevant subspaces and furthermore, can be applied in a streaming setting. The classes are aligned to and extend the understanding of the main cases with regards to outliers, as reported in many works, e.g., [9]. In more detail, in this work we make the following contributions:

1) We present a rule-based solution that provides a more comprehensible explanation for distance-based outliers taking into account the vicinity of their neighborhood by transforming the single-query (unsupervised) distance-based outlier detection job to a multi-query one. Each outlier point can be annotated into a different class as shown in Figure 2 based on the cardinality and density of its neighborhood, while no training is required. Two novel streaming distance-based outlier detection techniques that incorporate the solution are developed.

2) In order to tackle the curse of dimensionality, as well as to provide a more interpretable explanation, we implemented a technique that explores every lower dimensional subspace (either 2d or 3d) and leverages parallelism to allow our solution to operate with near real-time efficiency on intense streams.

3) We apply our methodology in a big-data streaming engine. More specifically, we extend the publicly available PROUD distributed framework [5] by incorporating the solutions as distinct modules in order to provide a fully working and interpretable big-data enabled distance-based outlier detection system. PROUD is built on top of Apache Flink and is a scalable and distributed framework for outlier detection with most of the state-of-the-art distance-based algorithms incorporated along with a self-adaptive partitioning technique.

4) We provide thorough evaluation with qualitative and quantitative results for every part of the solution as well as the complete framework in a streaming setting. We also compare our subspace exploration techniques against the REPEN state-of-the-art technique [22] that projects the full-dimensional dataset into a lower-dimensional representation while maintaining data irregularities.

As a final introductory note, our solution meets all four desired characteristics for explainers, as identified in [21], namely to be interpretable, to take a local fidelity approach, to be model-agnostic; and have a global perspective. It provides an explanatory labeling process along with subspace exploration for identifying the vicinity of an outlier and providing an interpretable outcome in line with the classes of Figure 2. Since the explanation process depends on the neighborhood

---

1All our code and datasets are available at https://github.com/tatoliop/PROUD-PaRallel-OUtlier-Detection-for-streams/tree/explainability

2https://flink.apache.org/
of a data point, its results are locally faithful, achieving the local fidelity characteristic. We also achieve the global perspective, through providing explanations for all outliers reported. Finally, our solution is technique-agnostic, since any distance-based outlier detection technique can be used in the detection process.

The remainder of this work is structured as follows. Section II provides the necessary information for distance-based outlier detection in data streams. Sections III and IV explain the outlier labeling process and the subspace exploration, respectively. Section V provides information about the PROUD framework and the complete parallel and streaming solution with the implemented explanation technique, while Section VI presents the experiments that cover all aspects of the solution. The final two sections discuss the related work and the conclusions, respectively.

II. BACKGROUND

The distance-based outlier detection problem is defined as follows:

Definition II.1. Given a distance function \( \text{dist} \) and a radius \( R \), two data points \( p_i \) and \( p_j \), with \( i \neq j \), are neighbors if \( \text{dist}(p_i, p_j) \leq R \).

Definition II.2. Given a set of data points \( P \) and the parameters \( R \) and \( k \), every point \( p_i \in P \), with less than \( k \) neighbors in a radius \( R \) around it, is reported as an outlier.

In a streaming environment, the dataset is an infinite sequence of data points that are continuously produced by one or more sources. In order to process such a stream, in this work, we split it into sliding windows. A window contains a finite set of data points with size \( W \) and evolves upon each slide \( S \), where new data points arrive and older ones expire. We consider windows with fixed size. The size of both the window and the slide can either indicate the number of data points (count-based windows) or the time period that the points were produced (time-based windows).

When the distance-based outlier detection problem is applied in a streaming setting, the definition is extended as follows:

Definition II.3. Given a window with size \( W \) and slide size \( S \), a data stream \( O \) and the parameters \( R \) and \( k \), for each window slide containing a set of objects \( P \subset O \), every data point \( p_i \in P \), with less than \( k \) neighbors in a radius \( R \) around it, is reported as an outlier.

The main difference between the two definitions is that, in a streaming setting, the outlier reporting needs to take place after every window slide while meeting latency constraints. This implies that every non-expired data point in the window may need to update its neighbors’ list each time new data points arrive into the stream and old ones expire and have its status re-assessed.

The state-of-the-art algorithms proposed to solve the streaming distance-based outlier detection problem include [4], [6], [16], [18]. All these algorithms try to reduce the distance computations needed in each window slide, which is the most intensive processing part, in different ways. MCOD and CPOD [16], [20] form clusters of data points that are inliers without the need to further process them, while Threash_LEAP [18] uses the time-slicing notion to further split the windows into distinct slices and speed-up processing. Finally, NETS [4] reduces the range queries by replacing the expired data points with the new ones. Some of the techniques operate in any metric space, e.g., [16], [20], while others assume euclidean space only, e.g., [4].

All the algorithms above use a single-query approach, meaning that the user provides only a single combination of \((W, S, R, k)\) values. Since distance-based outlier detection is sensitive to the parameters chosen, identifying the correct values can be tedious and long-running. Furthermore, supporting multiple queries allows the concurrent detection of different types of outliers using a single job. The proposed state-of-the-art multi-query streaming distance-based outlier detection algorithms include AMCOD [16], SOP [23], pMC-Sky [24] and MDUAL [7]. AMCOD is a variation of MCOD and uses the micro-clusters to group inliers. SOP transforms the queries into a single skyline computation while MDUAL uses the net-effect from NETS to quickly replace the expired data points with the new ones and eliminate unnecessary computations. pMC_Sky combines features from SOP and AMCOD, while targeting a distributed setting. In our work, we leverage the multi-query rationale to explain single-query results, as detailed in the next sections.

III. EXPLANATORY LABELING

In this section, we describe the process of labeling a distance-based outlier with regards to the given parameters \( R \) and \( k \) as one of the classes introduced in Section I, namely (1) Isolated point; (2) Point near a cluster; (3) Dense cluster; (4) Sparse cluster; (5) Dense micro-cluster; and (6) Sparse micro-cluster (see also Figure 2). We start by formally defining the classes that correspond to the different types of neighborhoods and then, we explain how we incorporate labeling into (streaming) distance-based outlier detection. We assume a euclidean vector space and a Minkowski distance function. Our solution is based on two key ideas.

- Firstly, to transform the single-query job into a multi-query one of the \( R \) and \( k \) parameters, i.e., to examine multiple values for the radius \( R \) and threshold \( k \). The outlierness is computed based on the initial parameters exclusively, however, the explanations are enabled through various values being tested simultaneously.
- Secondly, our explanatory labeling is derived by analyzing low dimensional datasets that come with more comprehensible visual representations, for which it is easier to reach a consensus regarding the classes, i.e., labels are not that subjective. Increasing the number of dimensions creates poor discrimination of the vicinity [25], which makes the clusters more vague and/or subjective.
A. CLASS SPECIFICATION

Our intention is to annotate outliers reported by any distance-based anomaly detection technique, but, to this end, we need to specify all 6 classes mentioned despite the fact that only 2 of them clearly correspond to outliers, namely isolated points and points near a cluster. The former class describes points that do not have neighbors in a broader range of radii than the initially set $R$ parameter. The latter refers to points that, for larger radii, start to acquire neighbors.

However, the notion and the flavors of a cluster need to be defined. Intuitively, a cluster is a set of points that are close to each other in relation to the distances between all points in the dataset. Nevertheless, having only a single cluster class for describing grouped data points can induce incorrect or meaningless explanations of the outlier detection results. Driven by this observation, we further separate the cluster class into sub-classes based on the cluster size and density.

More specifically, in our work we define two types of clusters based on their sizes, namely normal clusters and micro-clusters, and two types of clusters based on their density, namely dense and sparse ones. I.e., in total, there are four combinations. A micro-cluster has at most $\alpha_{size}$ times as many data points as a normal cluster. We set $\alpha_{size} = \frac{1}{10}$ which essentially defines a micro-cluster to be a cluster that differs in size from normal clusters by an order of magnitude but in general, $\alpha_{size}$ is configurable. Similarly, a sparse cluster has density at most $\alpha_{density}$ times the density of dense clusters, where in our work $\alpha_{density}$ is set to $\frac{1}{2}$. The density is defined as the average distance between all pairs of data points in a neighborhood.

Apparently, both size and density need not be characterized by binary classes. However, allowing more than two values for describing these cluster attributes at a finer level of granularity is orthogonal to our rationale, which can be extended accordingly. Similarly, as already mentioned, $\alpha_{size}$ and $\alpha_{density}$ can be set differently but fine-tuning them, which is important, is left outside the scope of our work.

B. DATA PROCESSING & LABELING

For every distance-based outlier detection algorithm, the basis of processing is the distance computation between the data points either explicitly or implicitly, according to the specified range $R$ and number of neighbors $k$ parameters. Motivated also by the fact that we aim to incorporate our explanation solution in a streaming setting and to be as less intrusive as possible, the outlier labeling is based on processing such distance computations exclusively.

However, we expand the search space for neighbors not only in the user-specified range $R$ but also in different radii while using different values of $k$ for the status assessment. Combining the multiple values of $R$ and $k$ allows us to discover the outlierness status of a data point in both the query setting and in bigger/smaller neighborhoods around it. This is enabled through transforming the single query distance-based outlier detection job (i.e., the single combination of $R$ and $k$ values) into a multi-query one. More specifically, we search outliers for a range of $R$ values: $2^{-p}R, 2^{-p+1}R, \ldots, 2^{-1}R, R, 2R, \ldots, 2^{p-1}R, 2^pR$. Similarly, we explore a range of $k$ values: $2^{-p}k, 2^{-p+1}k, \ldots, 2^{-1}k, 2k, \ldots, 2^{p-1}k, 2^p k$. $p$ is a small positive integer, which in our solution is set to $2$. Overall, there are $2p+1$ and $k$ values, which produce $(2p+1)^2$ combinations of parameters, i.e., $(2p+1)^2$ outlier detection queries instead of 1; for $p = 2$, multi-query outlier detection simultaneously checks 25 configurations.

A multi-query distance-based outlier detection algorithm runs and assesses the status of each data point for each $R-k$ combination of each slide. The algorithm can be any of the ones mentioned in Section II. Every exact outlier detection algorithm computes the number of neighbors for the original $R$ neighborhood for each data point. In the case of outliers, the algorithm needs to find the exact number of neighbors, whilst in the case of inliers it suffices to guarantee that this number is above the $k$ threshold; this can save several distance computations and thus reduces the processing runtime. Note that only the outliers from the default $R, k$ values are outputted, while the rest are only used to gather more metadata for the outlier explanation.

Going into more detail, we distinguish between 3 types of input data for the labeling process of the outlier data points, that derive from the detector’s distance computations. The first one, called $NN$ (number of neighbors), is the data point’s number of neighbors for each $R_i$, $i = -p, \ldots, p$ value divided by the default value of the $k$ parameter. The second type, called $Status table$, is a 2-dimensional table that reflects the binary status (i.e., inlier vs. outlier) of the data point for each $R_i - k_i$ combination. The third and final input type is the combination of the aforementioned two types. Tables 1 and 2 present examples for the first two input types, respectively. All these input types are produced by the detection algorithm.

The labeling model used in our implementation is a rule-based technique. In order to extract the rules, one annotated synthetic 2-dimensional dataset was created, which is available along with the source code. Each data point in this dataset belongs to a specified class and clusters do not

| $R$ | $NN$ |
|-----|------|
| $R/4$ | $0/k$ |
| $R/2$ | $0/k$ |
| $R$ | $0/k$ |
| $2R$ | $5/k$ |
| $4R$ | $15/k$ |

| $R-k$ | $k/4$ | $k/2$ | $k$ | $2k$ | $4k$ |
|-------|------|------|-----|-----|-----|
| $R/4$ | false | true | false | false | false |
| $R/2$ | false | false | false | false | false |
| $R$ | false | false | false | false | false |
| $2R$ | true | true | true | false | false |
| $4R$ | true | true | true | true | false |
overlap. Based on this dataset and the class characteristics specified in the previous section, the rules for each input were created in order to correctly label a data point. Since the solution is meant to be used as an unsupervised streaming distance-based technique, it does not require training at runtime. As the experiments of section VI will present, the rule-based technique can classify the outliers with high accuracy even when the classes are not easily distinguishable. Below, we explain these rules in detail through their corresponding algorithms, for the case where \( p = 2 \).

As far as the NN input type is concerned, we first compute the normalized value of the number of neighbors for each range as shown in Table 1. This procedure yields an array \( NN \) of 5 values (for \( p = 2 \)), where \( NN[1] \) holds the normalized number of neighbors for the smallest radius (\( \frac{R_1}{R} \)) and \( NN[5] \) corresponds to the biggest range (\( 4R \)). Finally, the procedure starts checking the array values in descending order in order to derive the label. The rules are presented in Table 3. For example, assume that \( k = 5 \) and Table 1 has the following values: \( NN[1] = NN[2] = NN[3] = 0; NN[4] = 1; NN[5] = 3 \). Since \( NN[5] > NN[4] > 0 \) but \( NN[3] = 0 \) we can output that the data point is an outlier near a cluster.

On the other hand, for the status input type, the procedure takes as input a 2-dimensional array of 5 rows and columns representing the \( R \) and \( k \) values in ascending ordering, e.g., the first row and column correspond to the smallest range and \( k \) values, respectively, as exemplified in Table 2. Each cell contains a boolean value, which represents the status of the data point, i.e., outlier or inlier. In such an array, we can state that, if a data point is an outlier for some row \( x \) and column \( y \) then every cell within the same row \( (x) \) and column values of \( y > y \) as well as every cell within the same column \( y \) and row values of \( x < x \) is also an outlier. Proofs of these are omitted but they can be found in multiquery distance-based outlier detection proposals, such as [7], [24]. By using this monotonicity knowledge, the algorithm further processes the array and stores the first column for each row that the outlier status is found, creating a smaller 1-dimensional array of 5 values (\( outlier_{pos} \)). Each position in this array corresponds to a different radius. Finally, with a similar process to the first input type, the procedure starts by checking the first outlier position for the biggest range going down to the smallest one, as presented in the rules of Table 4. For example, Table 2 is transformed to the array \( outlier_{pos} = [1, 1, 1, 2, 3] \). Then, according to the rules of Table 4, since \( 1 < outlier_{pos}[5] \leq 4, outlier_{pos}[4] > 1 \) and \( outlier_{pos}[3] = 1 \), the data point is labeled as part of a sparse micro-cluster. Finally, we can combine the two aforementioned rule sets to derive an ensemble approach. In such an approach, both algorithms run and, if the results differ, the strictest label is chosen. To this end, we use the strictness ranking operator denoted by \( \succ \), and the ranking is \( Dense \ cluster \succ Sparse \ cluster \succ Dense \ micro-cluster \succ Sparse \ micro-cluster \succ Point \ near \ cluster \succ Isolated \ point \), derived from the size and density of the neighborhood. In the above examples, the \( NN \) input indicates a data point near a cluster while the \( Status \ table \) input indicates that it belongs to a sparse micro-cluster. Based on the strictness ranking, the algorithm outputs the data point as an outlier that belongs to a sparse micro-cluster.

IV. SUBSPACE EXPLORATION

The previous section provides a way to label a data point into an interpretable class that can also explain the reason behind the outlierness of the point. However, this labeling provides better insights through visualization in low dimensional spaces. As the number of dimensions goes above the 3-dimensional space, the visual representation and comprehension of a cluster diminishes quickly. Moreover, the distance-based clustering and outlier detection algorithms themselves suffer from the curse of dimensionality. Increasing the dimensions reduces the variation of distances between the data points [25] rendering outliers more difficult to detect since most of the points have the same average distance to their neighbors [26].

In order to tackle the high-dimensionality problem in outlier detection, several techniques have been proposed in the literature with the most common ones being subspace representation (e.g., [22]) and subspace exploration (e.g., [27]). In the former case, a supervised machine learning model is used to transform the high-dimensional dataset into a lower-dimensional representation, while in the latter one all \( n \)-dimensional subspaces are investigated for outliers and one of them is chosen for visualization. In our case, where an intense and high volume stream is the input, the subspace representation cannot be used. This is due to the fact that a data stream is continuously evolving with time and, therefore, its data distribution is volatile. A trained model cannot cope with the sudden changes and the representation will not be as effective, thus raising the need for continuous training. This, in turn, creates an extra overhead in the whole process and is particularly challenging to be applied in an online setting.

On the other hand, investigating every subspace can help in identifying important subspaces, where most outliers are detected but also increases the workload. In our solution we have opted to explore and label outliers in every lower-dimensional subspace, but we discuss how we can prune some subspaces at the end of this work. The overhead is mitigated by the fact that each subspace can be explored in parallel, which allows us to capitalize on the capabilities that modern massively parallel streaming data processing platforms, such as Apache Spark and Flink, provide. Details about the engine and the implementation of the solution are part of the next section.

Detecting outliers in subspaces does not yield false positives, as the following theorem states:

**Theorem IV.1.** Let \( d = |D| \) be the size of the set of dimensions \( D \) of the points in the data stream. A data point that is assessed as an outlier in an \( n \)-dimensional space,

\[ \text{Proof:} \]

\[ \text{(omitted)} \]
where \( n = |N| \) and \( N \subseteq D \) is also an outlier in any \( x \)-dimensional space, \( x = |X| \), with \( \subseteq X \subseteq D \).

**Proof.** In a vector space, all parameterizations of a Minkowski distance function add a non-negative value for each additional dimension considered. Therefore the distance between two points monotonically increases with considering additional dimensions. Consequently, the more the dimensions, the less the neighbors of a data point within a fixed distance \( R \).

However, not detecting outliers in lower-dimensional spaces does not tell us anything regarding outlierness in the full space due to the presence of false negatives:

**Lemma IV.2.** A data point that is assessed as an outlier in a \( x \)-dimensional space, \( x = |X| \), \( X \subseteq D \) is not bound to be an outlier in an \( n \)-dimensional space, \( n = |N| \) with \( \subseteq X \subseteq D \), where \( D \) is the complete set of dimensions.

**Proof.** It is trivial to present a counterexample, where a large set of points coincide in all but the last dimension, and in the last dimension, all their pairwise distances exceed the radius \( R \). In such a case, considering all but the last dimension detects no outliers, while, in the full-dimensional spaces, all points are outliers. However, false negatives do not always appear; e.g., in our counterexample, considering only the last dimension is adequate to detect all outliers.

Based on Theorem IV.1 and the easiness in representing and understanding clusters in 2d and 3d subspaces, in our solution, we identify and explore every 2d and 3d subspace in the dataset. For every subspace, the respective outliers are detected and interpreted based on the labeling process in Section III. The resulting outlier set from the subspaces includes every data point that is identified as an outlier in at least one subspace and is a subset of the complete outlier set of the full-dimensional dataset, as defined in Lemma IV.2.

The total number of 2d subspaces of a \( d \)-dimensional dataset is \( \binom{d}{2} = O(d^2) \) while the set of 3d subspaces of a \( d \)-dimensional dataset are \( \binom{d}{3} = O(d^3) \). E.g., in a 4-dimensional dataset, our solution explores 6 2-dimensional subspaces. If each subspace is represented by \( (i, j) \) where \( i \) and \( j \) are the index numbers of each dimension, the technique explores the \((1,2), (1,3), (1,4), (2,3), (2,4), (3,4)\) subspaces. For the rest of the section, for the sake of simplicity, the examples refer to 2d subspaces but in the complete solution and the evaluation, the effects of choosing 3d subspaces are also investigated.

For each data point that is assessed as an outlier in the user provided \( R \) and \( k \) input parameters, the output is a vector of classes stemming from the labeling process in every subspace. Transforming triangular matrices to vectors is trivial (e.g., see [28]). Continuing the example of the 4-dimensional dataset, an output data point with a vector: \([\text{dense micro}, \text{cluster}, \text{sparse micro}, \text{cluster}, \text{inlier}, \text{inlier}, \text{inlier}, \text{dense micro}, \text{cluster}]\) means that the point is an inlier for the 3rd, 4th and 5th subspaces and an outlier for the 1st, 2nd and 6th subspaces; more specifically, it belongs to a dense micro-cluster in the 1st and 6th subspaces and to a sparse micro-cluster in the 2nd one.

Such a vector also helps to identify the true class of the point in the full-dimensional space. Increasing the dimensionality is followed by an increase in the distances. This in

|     | NN[5]    | NN[4]    | NN[3]    | NN[2]    | NN[1]    | label   |
|-----|---------|---------|---------|---------|---------|---------|
| 0   | any value | any value | any value | any value | any value | Isolated |
| not 0 | 0       | any value | any value | any value | Near cluster |
| not 0 | any value | 0       | any value | any value | Near cluster |
| 0 < NN[5] < 2.5 | not 0 | 0 < NN[3] < 1 | any value | any value | Sparse microcluster |
| 0 < NN[5] < 2.5 | not 0 | not 0 | 0       | any value | Sparse microcluster |
| NN[5] ≥ 2.5    | not 0 | not 0 | not 0 | 0 any value | Sparse cluster |
| not 0 | 0 < NN[4] ≤ 2.5 | NN[3] ≥ 1 | not 0 | any value | Dense microcluster |
| not 0 | NN[4] > 2.5 | NN[3] ≥ 1 | not 0 | any value | Dense cluster |

**TABLE 3.** NN-based labeling

|     | outlier_pos[5] | outlier_pos[4] | outlier_pos[3] | outlier_pos[2] | outlier_pos[1] | label   |
|-----|---------------|---------------|---------------|---------------|---------------|---------|
| 1   | any value     | any value     | any value     | any value     | any value     | Isolated |
| not 1 | 1       | any value     | any value     | any value     | Near cluster |
| 1 < outlier_pos[5] ≤ 4 | not 1 | 1               | any value     | any value     | Sparse microcluster |
| outlier_pos[5] > 4 | not 1 | not 1           | any value     | any value     | Sparse cluster |
| not 1 | not 1     | outlier_pos[3] ≥ 3 | 1      | any value     | Dense microcluster |
| 1 < outlier_pos[5] ≤ 4 | not 1 | 1 < outlier_pos[3] < 3 | 1      | any value     | Sparse microcluster |
| outlier_pos[5] > 4 | not 1 | 1 < outlier_pos[3] < 3 | 1      | any value     | Sparse cluster |
| 1 < outlier_pos[5] < 6 | not 1 | not 1           | not 1        | 1            | Dense microcluster |
| outlier_pos[5] ≥ 6 | not 1 | not 1           | not 1        | not 1        | Dense cluster |
| not 1 | not 1     | not 1           | not 1        | not 1        | Dense cluster |

**TABLE 4.** Status-based labeling
turn means that clusters formed in lower dimensions degrade and the mean distance of the cluster points is increased. From this, we can assume that the class of the vector that corresponds to a less strict class better describes the data point’s true class in the full-dimensional space. Therefore, to identify the true class from the vector, the strictness ranking of the classes from Section III is employed. Based on the ranking for each class we remark that:

1) Every isolated data point will stay isolated in the full-dimensional space.
2) The data points near clusters will most likely become isolated data points.
3) Every point in a dense cluster or dense micro-cluster will most likely become part of a sparse cluster/micro-cluster respectively.
4) Every point in a sparse cluster or sparse micro-cluster will either stay the same or start moving outside these sparse micro-clusters.

Based on the above, we can label each outlier with the most probable class that it might belong to, based on the labeling vector. Continuing from the previous example, the most probable class that the data point with example vector might belong to is the sparse micro-cluster class.

Another useful output that stems from this process is identifying the most important subspaces. An important subspace for a slide is the one that contains most of the slide’s outliers and indicates the dimensions where the distances between the data points are more distinguishable. This means that on the long run, as the stream continues producing data, we can identify how the dimensions of interest, i.e., the most important subspaces, change through the course of time. This aspect is not further explored in this work, though, but constitutes an inherent capability of our solution.

V. EXTENSION OF THE PROUD FRAMEWORK
PROUD (PaRallel OUtlier Detection for streams) [5] is a distributed framework built on top of Apache Flink that is used for outlier detection jobs on high-volume, intense and continuous streams of data. It is fully modular and encapsulates most of the state-of-the-art single and multi query distance-based outlier detection techniques; furthermore, any new proximity-based technique can easily be incorporated as well. Any type of source and sink types can also be implemented with a wide variety of pre-built libraries being available through Flink, e.g., to ingest data from Kafka.

The pipeline of a detection job in PROUD follows a 3-step process. Firstly, the incoming data points are transformed and pass through the partitioning phase during which they are split into overlapping cells and sent to the set of worker nodes. Secondly, each node runs an outlier detection technique on its group of data outputting a local and independent set of outliers, and finally, the complete set is created from the combination of the local ones and written to the selected data sink. PROUD incorporates two ways of partitioning the incoming data using a (a) grid-based and a (b) tree-based technique. The first one splits the euclidean space into overlapping cells whilst the second one uses a VP-tree [29] with its leaves representing a cell and can be used for any metric space. Techniques for adaptive self-balancing of the workload have been also implemented [30] in order to distribute the load of each group by increasing/decreasing their boundaries when necessary.

The techniques from Sections III and IV are implemented in the different steps of the framework to provide the complete solution as an independent distance-based outlier detection engine that can tackle both interpretability and high-dimensionality problems. The subspace exploration algorithm is implemented as an independent optional module before the partitioning phase. It splits the dataset into 2d or 3d subspaces where each one is handled as a distinct outlier detection job with its own partitioning structure.

As mentioned in Section IV, using every combination of 2 or 3 dimensional subspaces and running a different outlier detection process for each of them can impend a big overhead in a streaming job. Using a traditional centralized approach would slow down the process and the results would not be available in real-time as is the goal for stream processing scenarios. By using a distributed framework, each subspace and subsequently the workload can easily be distributed to different worker nodes in order to have the necessary high throughput required by the application. The PROUD framework, due to the distributed nature of Flink, can easily scale out with more nodes added if necessary. Furthermore the adaptive partitioning techniques can increase/decrease the number of data points in each group to, furthermore, balance the workload of each node.

Figure 3 presents the complete pipeline of the framework using the subspace exploration technique for a 3-dimensional input dataset. After each data point ingestion, their features are split into the three corresponding 2d subspaces using the exploration technique. Each subspace uses its own partitioning scheme to further split the data points into different task slots for parallel processing. The results from each outlier detection process per subspace are combined, as explained in Section IV to form the global output. Merging the local outputs is shown as a single step in the figure but need not necessarily run in a single task; i.e., it can be parallelized as well.

The explanatory labeling is implemented through two new outlier detection techniques that transform the single query into a multi-query process and label each detected outlier. Note that both these techniques encapsulate the labeling rules from Tables 3 and 4, as well as their combination, but differ in the manner they use them. The first technique is called Explain and is based on the pMCSky multi-query algorithm. During its processing, the technique starts by computing the distances of the data points for the biggest radius of the list of R values. Afterwards it assesses the outliersness of each data point for each combination of R and k values. If a data point is an outlier for the default detection parameters, then it passes through the labeling rules based on the chosen input type. Finally, the outlier along with its class are outputted.
The second technique is called ExplainNet and is implemented in order to avoid re-labeling the same data point in consecutive slides. The technique uses a data structure to store the number of neighbor changes that happen in the outlier data points compared to the previous slide. For example, assume a data point $i$ is outputted as an outlier with the Isolated label in slide $x$. Afterwards, during the labeling phase of slide $x + 1$, if the point’s total number of neighbors has not changed then it is outputted as an outlier with the label of the previous slide. The rest of the technique is similar to the Explain one. A drawback of ExplainNet is that only the net change of the total number of neighbors is stored and no proximity information of new and old neighbors, e.g. the expired neighbor could be in the $\frac{R}{2}$ radius and the new one in the $4R$ radius. In other words, ExplainNet trades result quality for less labeling-related operations.

**VI. EVALUATION**

In this section, the results from the experiments regarding all aspects of the solution are presented. The first set of experiments show the explanation process’ qualitative results using a real-world dataset as well as 3 custom annotated ones to compare with popular supervised classification algorithms. Next, the difference between the detected outliers of 3 multidimensional datasets against the resulting outlier set of the subspace techniques are investigated, including a comparison against [22], along with the runtime measurements in a continuous data stream. Finally, the performance of the whole framework is investigated in the third set of experiments. When evaluating the quality of the results, no streaming aspects are considered.

**A. EXPLANATORY LABELING PERFORMANCE**

This set of experiments is divided into two parts. In the former, we provide evidence about the effectiveness of our labeling, and in the latter, we evaluate our adopted rules, as captured by Tables 3 and 4 and their combination, against ML supervised classifiers. All data and code are available at the github repository, and the experiments are fully repeatable.

1) **Quality of labeling results**

In order to assess the quality of the explanatory process, a real world dataset called Glass$^3$ has been employed. Since the dataset is 10-dimensional and we need to visualize the data in a manner that is easily comprehensible, the first and third dimensions were chosen.

Figure 4 presents the results of the explanatory process using all 3 input types discussed in Section III (i.e., $NN$, status, and both). Each plot represents the dataset’s inliers and outliers in a 2d euclidean space. Each outlier is also represented based on the class it belongs to according to the result of the explanation algorithms. Values of the $R$ and $k$ parameters for the outlier detection are the same in all 3 cases and set as follows: $R = 0.15$, $k = 5$. As such, the same set of outliers and inliers are present in every plot.

As the figure presents, every input type has similar but not exactly the same labelings. Nevertheless, every plot depicts sensible outputs. Starting with isolated points, the data points that belong to this class are further away from the rest of the dataset’s points. The isolated points differ between the input types due to the nature of each type. For the $NN$ input, the number of neighbors around a data point constitutes the key factor, while for the Status table input, we only consider the outlierness of the data point. This differentiates the classes of the same data points, i.e. points on $(11,0.7)$ and $(13,3)$, which have a close neighbor and are not labeled as isolated ones when the NN-based rules are applied.

Similar small differences are also observed regarding other outlier classes, such as points near clusters and sparse clusters/micro-clusters. When combining the inputs and the algorithms by taking into account the strictness ranking of the classes, we essentially assign more weight on the identification of bigger clusters and the results are at the bottom plot of the figure. Another difference between the rules of Tables 3 and 4 is the labeling of data points as dense micro-clusters based on the Status table input type, while, when processing the $NN$ input, it is derived that they belong to a sparse cluster (blue cross points in the middle plot). This difference is also attributed to the type of the inputs’ metrics and the combination (bottom plot) solves this by deciding in

---

$^3$https://archive.ics.uci.edu/ml/datasets/glass+identification
favor of the bigger cluster, which is a stricter label (sparse cluster instead of micro-cluster). In both cases, the technique can explain different settings. Based on the NN input and subsequently the combinatory one, the reported outliers are labeled as part of a sparse cluster, which implies that they are probably fridge points or part of a region where fewer data points exist. On the other hand, the label of dense micro-cluster based on the Status table declares that these points belong to a micro-cluster that does not comprise enough points in relation to the $k$ parameter. Finally, some data points are labeled as parts of a dense cluster based on the Status table input, which is also transferred to the combination plot. These points correspond to the fridge points of such a cluster.

Since the explanatory process is dependent on the outlier detection parameters ($R$ and $k$), Figure 5 presents the visualized results of the process in the same dataset using the combination of the rule sets when the $R$ parameter is increased to $R = 0.25$. Increasing the $R$ value yields less outliers since the detection can find neighbors in a bigger region around the data points. This also affects every input type of the explanatory labeling. Comparing the bottom plot of Figure 4 with Figure 5, the behavior is as expected. The isolated data points are greatly reduced since more neighbors are found when using a bigger value for the radius $R$. In Figure 5, the only isolated point is on the lower right side, which is further away from every other point in the dataset while the previously labeled isolated points are now identified as points near a cluster. This also affects the other classes with the resulting output favoring bigger clusters, since the range value is almost doubled.

Overall, this experiment allows us to claim that our solution is effective but it is unclear which of the three flavors (NN-based, Status-based or the strictest from both) should be chosen. The next experiment aims to provide an answer to this issue.

2) Comparison with classification algorithms

To assess the correctness of the rule-based labeling process, we compare the explanatory results of our solution with the results from two popular classification algorithms, namely Random Forest and XGboost. The same synthetic dataset, which was used for the rules extraction was also used for the training of the supervised classifiers. The rules are im-
implemented along with the rest of the framework in Scala. For the supervised algorithms, we chose the RandomForest and XGBoost implementations from the scikit-learn Python library\(^4\).

For the testing procedure, 3 synthetic datasets are used. The first one, called custom_big, includes every class with a total of 4404 data points and is created based on the cluster definitions of Section III, e.g. a micro-cluster contains 200 data points whilst a cluster contains 2000. The second dataset, called custom_medium, excludes the near-cluster data point class and has a total of 524 data points. The third dataset, called custom_small, excludes the sparse micro-cluster class and has a total of 28 data points and is used to provide insights on not strictly defined clusters. All 3 datasets are visualized in Figure 6.

For each dataset, only the outlier set of points from the outlier detection process is used instead of the whole set. This yields a total of 2523 outliers for custom_big, 228 for custom_medium and 17 for the custom_small datasets. Note here that the points forming a dense micro-cluster in the custom_small dataset are not part of the outlier set, meaning that the category is also missing from the explanatory process of the rules and the classification algorithms.

Table 5 presents the weighted F1 score for each algorithm along with our rules solution for every input type on the 3 aforementioned dataset. The weighted F1 score is chosen as the metric since it is considered a credible metric for multi-class classification as well as for class-imbalanced datasets, such as the ones employed, i.e., there are only 2 points for the isolated class and more than 2000 points for the sparse cluster on the custom_big dataset. As the table presents, our solution is at least as good as the two other popular classification methods. From the table we can also extract the information that the Status-based procedure is, in most datasets, the less efficient variant while the combinatory variant using both input types always yields better results.

Finally, in order to delve deeper into the explanatory process, Table 6 presents the confusion matrix for each input type using our solution. The custom_small dataset is chosen due to the fact that it is the most challenging one from the 3 synthetic datasets of the experiments and yields the lower weighted F1 score at a single input type on table 5. From the matrix, we can safely extract the conclusion that the Combined input type that builds upon both NN and Status table types yields better results. E.g., the sparse cluster class is incorrectly labeled based on the Status table input as a dense micro-cluster, while combining the two algorithms does not suffer from such a limitation. The same holds for the dense micro-cluster class, where, based on the NN input, the procedure incorrectly labels its data points as a sparse micro-cluster.

### B. SUBSPACE EXPLORATION PERFORMANCE

In this set of experiments, we investigate the subspace exploration technique and its efficiency. We have used 3 real-world datasets consisting of 3, 6 and 10 dimensions. The 3-dimensional dataset is called TAO and is a commonly used dataset for streaming outlier detection experiments (e.g., [6]) while the 6-dimensional one is called Mammography\(^5\). Finally, the 10-dimensional dataset is part of the Forest Cover dataset, which is available at the UCI KDD archive\(^6\). We have chosen the first 10 dimensions that have a bigger impact on the distance-based outlier detection process.

---

\(^4\)https://scikit-learn.org/

\(^5\)http://odds.cs.stonybrook.edu/mammography-dataset/

\(^6\)http://kdd.ics.uci.edu
TABLE 6. Confusion matrix for custom_small dataset on every input type using the custom rules

| Truth          | Isolated | Dense MC | Sparse MC | Dense Cluster | Sparse Cluster | Near cluster |
|----------------|----------|----------|-----------|---------------|----------------|--------------|
| Isolated       | 2        | 0        | 0         | 0             | 0              | 0            |
| Dense MC       | 0        | 2        | 0         | 0             | 0              | 0            |
| Sparse MC      | 0        | 0        | 2         | 0             | 0              | 0            |
| Dense Cluster  | 0        | 0        | 0         | 12            | 0              | 0            |
| Sparse Cluster | 0        | 0        | 0         | 0             | 12             | 0            |
| Near cluster   | 0        | 0        | 0         | 0             | 0              | 1            |

TABLE 7. Comparison of detected outliers using the subspace selection technique and the REPEN technique

| Technique   | Full-dimensional | 2d Subspaces | 2d REPEN | 3d REPEN |
|-------------|------------------|--------------|----------|----------|
| Outliers    | 145              | 119          | 36       | 60       |

1) Quality of results

The first experiment deals with the extent to which false negatives appear when the subspace technique is used. In 2d and 3d subspaces, the total set of outliers is the union of the data points that are considered as outliers in each subspace after deduplication. This process is expected to find less outliers than the outlier process on the full-dimensional dataset. Nevertheless, the output set is always part of the whole set of outliers with zero false negatives.

Figure 7 presents the findings from the experiments in all 3 test datasets. As expected, the information loss is more significant as the dataset has more dimensions. On the other hand using 3d subspaces improves the recall, significantly, which is more evident in the 10-dimensional dataset. In this case using 2d subspaces misses 73% of the outliers, while 3d subspaces reduce this percentage at only half the outliers. Also, it is evident that the effectiveness of subspace-only reporting heavily relies on the type of the dataset and, in the figure, the loss ratio ranges from very low to a high percentage.

In addition, we have experimented with another technique that is proposed for distance-based outlier detection pipelines, in order to project the multi-dimensional dataset into a lower dimensional representation in a manner that maintains data irregularities and thus allows for detecting the real outliers when processing the projected data. This technique is called REPEN and is presented in [22]. In this experiment, we have used the 6d Mammography dataset and we split it into 2 sets for training and testing the technique, respectively. REPEN was trained using the training subset yielding different 2d and 3d representations for the dataset. From the total set of representations, we have chosen the ones with the best AUC values of the training set for 2 and 3 dimensions, respectively. Finally, using the chosen representations, we have tested the algorithm using the testing subset. We have tested simultaneously the same subset using our subspace exploration technique with 2d subspaces. Table 7 presents the outliers found using each technique against the total outliers from the full-dimensional testing subset.

VOLUME 4, 2016
From the table values, our exploration technique using 2d subspaces outperforms the REPEN technique even when it uses a 3d representation.

2) Efficiency

The next experiment presents the throughput of the exploration technique using 2d and 3d subspaces against the full-dimensional dataset. For completeness we have also used both explanatory techniques, Explain and ExplainNet, and the pMCOD detector from the PROUD framework for comparison. Note here that the exploration technique can be used independently of the outlier detector and its explanation ability. The throughput metric presented in the next experiments, represents the processed data points per second for the outlier detection process. This process is split in different task slots, i.e. workers as explained in Section V, and the outputted metric is the median throughput of all task slots. In the experiment, we have used a cluster of 25 VMs with each one having 16 virtual cores and 16GB of RAM.

Figure 8 presents the results from the experiments. The 2d subspaces always has a better throughput than the 3d or the full-dimensional dataset. On the other hand, using 3d subspaces yields in most of the cases a closer throughput to the full-dimensional dataset. This means that using the explanatory subspace-based techniques yields at least as good a throughput as the full-dimensional explanatory outlier detection process, which creates a single logical detection job.

C. FRAMEWORK PERFORMANCE

The final set of experiments involves the complete framework for continuous distance-based outlier detection in combination with the explanatory labeling and subspace exploration techniques. The datasets used are the same ones as the experiments in the previous section, namely TAO, Mammography and FC along with a 16-dimensional dataset called Pendigits from the outlier detection dataset repository\(^7\).

Figure 9 presents the results from the first experiment showing the differences between the two implemented explainable outlier detection techniques, namely Explain and ExplainNet. The plot depicts the number of times that the explanatory labeling technique is used during the outlier detection process. This immediately translates to a small time overhead that the labeling imposes on the processing time of each slide. As expected, ExplainNet decreases the number of calls needed for the explanations by reusing metadata from the previous slide. However, as mentioned in Section V, it comes at the expense of an approximate labeling for the affected data points in some cases. On the other hand, the Explain technique labels each outlier in every slide through executing an increased number of function calls, which yields results of higher quality.

Figure 10 presents the throughput of the framework for all datasets using the 2 explainable outlier detection techniques in combination with 2d subspace exploration compared to the single query pMCOD and the multi-query pMCSky techniques running on the full-dimensional datasets. As expected, pMCOD is the fastest one yielding the best throughput on each dataset. This stems from the fact that pMCOD is distinctively designed to cut down range queries and processes a new slide as fast as possible. On the other hand, pMCSky has the smallest throughput in all cases. The pMCSky technique is used as a reference to the multiple values of the parameters \(R\) and \(k\) that the explanatory techniques need to process. This detector is used in multi-query situations where there is a need to find outliers for different sets of parameters on a single job. In our experiments we have set the parameters of pMCSky to the same values as the explanatory values, i.e., the radius (resp. threshold) ranges from \(R/4\) (resp. \(k/4\)) to \(4R\) (resp. \(4k\)).

Both explanatory techniques have similar performance that fluctuates between the other two detectors. This means that both of them are better than their counterpart on the multi-query space while splitting the dataset into 2d subspaces and processing each one independently. Moreover, their overhead in terms of latency is fully hidden by the slower pMCSky technique running in parallel. In other words, the performance of our solution as perceived by the user is the same as the one of state-of-the-art multiple query techniques when explanations based on inputs over the full dataset are provided and significantly faster when explanations are based on processing only subspaces in a multi-query manner.

The final experiment presents the scalability of the framework and more specifically the subspace exploration technique. The number of partitions directly affects the runtime of the framework when the subspace exploration is used. This stems from the fact that each subspace has its own partitioning structure and splitting the data balances the workload in a better fashion. Figure 11 presents the results for different values of partitions for the two explainable outlier detection techniques. From the figure we draw the observation that the ExplainNet technique scales better and is more robust than the Explain one. This is attributed to the fact that Explain needs more function calls with more of them occurring on the same partition while ExplainNet eliminates such overheads by reducing their number.

Based on all our experiments, we note that there are many ways that the techniques can be combined. First, if the user wants a fast explanatory outlier detection job without the need for the full set of outliers, the 2d subspace exploratory technique can be used to speed up the process. On the other hand, if the quality of the outlier set is more important, the 3d subspace technique is preferable. A final remark is that the full dimensional single-query job can also run concurrently with an exploratory technique (either 2d or 3d) in order to also get the full set of outliers for further processing. In all these ways, we avoid running our explanation-oriented extensions to pMCSky over the full dataset, which is significantly slower.

\(^7\)http://odds.cs.stonybrook.edu/pendigits-dataset/
VII. RELATED WORK

There are several works in the literature that propose solutions for either the explanatory process in data mining and/or tackling high-dimensionality in datasets. For example, in [31] the authors build upon the LIME framework with a way to generate synthetic neighbors for better quality of results.

Regarding explanation in anomaly detection, [32] and [33] propose outlier detection algorithms that present interpretable insights for the data points in question. The first one proposes the Local Outlier Probabilities algorithm, which outputs a percentage of outlierness for each data point. In contrast to most techniques, the score is derived using a statistical probability distance in the scale of \([0, 1]\) regardless of the dataset’s distribution. The latter algorithm uses an empirical copula to find the probability of observing a data point at least as far from the dataset’s normal distribution as the data point in question.

Another approach to interpretable outliers is presented in [34]. The authors’ solution builds human-interpretable rules for the detected anomalies. The solution is used on time series data by creating annotations for the data points and building a decision tree based on the annotations. Finally, the rules are extracted from the nodes of the tree. The work in [35] uses the Human-in-the-loop approach to get feedback from a domain expert about true outliers. It starts by clustering the possible outliers and creates a minimal set of questions from the inliers and the clusters that are presented to the user for answering. This helps the solution get better insights on the true outliers of the dataset. In [36], a system that measures the non-conformance of a tuple in a dataset by scoring its features is proposed. In [37] the authors use data flows to distinguish outliers into anomalies and furthermore explain anomalies by finding events of interest. Finally the authors of [38] build a novel AutoML pipeline with the goal of creating a classifier that can approximate an anomaly detector’s results and also choose the features that fully explain the

FIGURE 8. Outlier detection throughput comparison using both subspace choices versus the full-dimensional dataset

FIGURE 9. Total number of function calls for each explanatory technique on every dataset
Regarding tackling high-dimensionality in outlier detection, the proposal in [39] describes an approximate algorithm using k-NN detectors and, in order to reduce the dimensionality of the dataset, space-filling curves are used. Another solution is [40], which uses subspace selection and hyperplanes to deal with high-dimensional data in static settings. This technique chooses a reference set of points and determines the hyperplane for the outlier detection process. Additionally, the authors of [41] propose an outlier detection algorithm for high-dimensional data that uses a random hashing technique to output the outlier score for each data point. The algorithm, based on a sample of the dataset, chooses a random subset of dimensions and creates a random set of hash functions that are later applied to every data point in the dataset. The pipeline is repeated multiple times with different random hash functions and the final outlier score for each data point is outputted.

None of the techniques described above approaches explanations in distance-based outlier detection as we do; our solution leverages subspace exploration but goes one step further and is based on labels of descriptive classes while being applicable to a streaming setting. Finally, a benchmarking tool for semi-supervised outlier detection algorithms along with explainability techniques was created in [42]. The authors use Apache Spark with different custom stream datasets to measure different metrics on the detection and explanation phases of the algorithms in question. For the evaluation part, 3 deep learning algorithms are used. Our work is orthogonal to this effort and may provide another type of evaluation metric based on the proposed labels.
VIII. CONCLUSIONS & FUTURE WORK

In this work, we advocate extending distance-based outlier detection, which is an unsupervised process, with an intuitive post-processing explanation phase that goes beyond existing explainable AI solutions. Instead of solely focusing on identifying the most relevant dimensions for outliers, we additionally label outliers in a manner that is intuitive for the users, requires no runtime training and can be easily visualized. In addition, we have incorporated our solution into a fully working framework based on Apache Flink that applies continuous distance-based outlier detection over streams.

Our work may be deemed as specific to distance-based outlier detection, but it aims to complement rather than replace other initiatives, such as Exathlon [42]. Nevertheless, there are many directions to extend our solution. Firstly, devising a systematic methodology to tune our parameters, such as $\alpha_{size}$, $\alpha_{density}$ and $p$ values and possibly extend the set of labels considered is an open issue. Secondly, we aim to leverage our explanatory procedure to fine-tune the user defined $R$ and $k$ input parameters, which is a notoriously difficult issue. I.e., we intend to explore the application of our proposal to perform fine-tuning in addition to providing explanations in a single stream processing pipeline. More holistic solutions can also encapsulate concept drift along with outlier detection while maintaining explainability. Thirdly, transferring our solution to generic metric spaces, where outliers in subspaces are not necessarily outliers in the full dataset, needs to be investigated. Finally, we acknowledge the fact that in datasets of 50, 100 or more dimensions, exploring all pairs and triples is inefficient even for a massively parallel solution. To mitigate this problem, we aim to explore transferring subspace selection techniques, such as [15] that can identify which pairs and triples to focus on. Such a process, can run periodically as a side-job and its integration in our prototype system along with its evaluation is an interesting extension.

ACKNOWLEDGMENT

This research work has been supported by the European Commission under the Horizon 2020 Programme, through funding of the RAINBOW project (Grant 871403).

REFERENCES

[1] Charu C. Aggarwal. Outlier Analysis, 2nd Ed. Springer, 2017.
[2] Edwin M. Knorr, Raymond T. Ng, and Vladimir Tucakov. Distance-based outliers: Algorithms and applications. The VLDB Journal, 8(3-4), 2000.
[3] Sharmila Subramaniam, Themis Palpanas, Dimitris Papadopoulos, Vana Kalogeraki, and Dimitrios Gunopulos. Online outlier detection in sensor data using non-parametric models. In VLDB, pages 187–198, 2006.
[4] Susik Yoon, Jae-Gil Lee, and Byung Suk Lee. NETS: extremely fast distance-based outlier detection for windows over streaming data. In EDBT, pages 529–540, 2009.
[5] Lei Cao, Di Yang, Qinyang Wang, Yanwei Yu, Jiayuan Wang, and Elke A Rundensteiner. Scalable distance-based outlier detection over high-volume data streams. In ICDE, pages 76–87, 2014.
[6] F. Angiulli and F. Fassetti. Detecting distance-based outliers in streams of data. In CIKM, pages 811–820, 2007.
[7] Luan Tran, Liyue Fan, and Cyrus Shahabi. Distance-based outlier detection in data streams. Proceedings of the VLDB Endowment, 9(12):1089–1100, 2016.
[8] D. Yang, E.A. Rundensteiner, and M.O. Ward. Neighbor-based pattern detection for windows over streaming data. In EDBT, pages 529–540, 2009.
[9] F. Angiulli and F. Fassetti. Detecting distance-based outliers in streams of data. In CIKM, pages 811–820, 2007.
[10] Luan Tran, Liyue Fan, and Cyrus Shahabi. Distance-based outlier detection in data streams. Proceedings of the VLDB Endowment, 9(12):1089–1100, 2016.
[11] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 1135–1144, 2016.
[12] Guansong Pang, Longbing Cao, Bing Chen, and Huan Liu. Learning representations of ultra high-dimensional data for random distance-based outlier detection. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pages 2041–2050, 2018.
[13] Lei Cao, Jiayuan Wang, and Elke A Rundensteiner. Sharing-aware outlier analysis over high-volume data streams. In ICDM, pages 527–540. ACM, 2016.
[14] Theodoros Tolopoulos and Anastasios Gounaris. Multi-parameter streaming outlier detection. In WI, pages 208–216, 2019.
[15] Michael E Houle, Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Can shared-neighbor distances defeat the curse of dimensionality? In International conference on scientific and statistical database management, pages 482–500. Springer, 2010.
[16] Milos Radovanovic, Alexandros Nanopoulos, and Mirjana Ivanovic. Hics: High contrast subspace selection techniques, such as [15] that can identify which pairs and triples to focus on. Such a process, can run periodically as a side-job and its integration in our prototype system along with its evaluation is an interesting extension.

and queries. In Proceedings of the 2021 International Conference on Management of Data, pages 2063–2075, 2021.
[8] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. LOF: identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD international conference on Management of Data, pages 93–100, 2004.
[9] Spiros Papadimitriou, Hirooyuki Kitagawa, Phillip B Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. In Proceedings 19th international conference on data engineering (Cat. No. 03CH37405), pages 315–326. IEEE, 2003.
[10] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. Angle-based outlier detection in high-dimensional data. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 444–452, 2008.
[11] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In 2008 eighth ieee international conference on data mining, pages 413–422. IEEE, 2008.
[12] Fabian Keller, Emmanuel Müller, Andreas Wixler, and Klemens Böhm. Flexible and adaptive subspace search for outlier analysis. In Proceedings of the 22nd ACM international conference on Information & Knowledge Management, pages 1381–1390, 2013.
[13] Nguyen Xuan Vinh, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Jian Pei. Discovering outliers: Algorithms and applications. The VLDB Journal, 8(3-4), 2000.
[14] Fabian Keller, Emmanuel Müller, and Klemens Böhm. Hics: High contrast subspace selection techniques, such as [15] that can identify which pairs and triples to focus on. Such a process, can run periodically as a side-job and its integration in our prototype system along with its evaluation is an interesting extension.

ACKNOWLEDGMENT

This research work has been supported by the European Commission under the Horizon 2020 Programme, through funding of the RAINBOW project (Grant 871403).
[29] Peter N Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In SODA, volume 93, pages 311–321, 1993.

[30] Theodoros Toliopoulos and Anastasios Gounaris. Adaptive distributed partitioning in apache flink. In 36th IEEE International Conference on Data Engineering Workshops, ICDE Workshops 2020, Dallas, TX, USA, April 20-24, 2020, pages 127–132. IEEE, 2020.

[31] Yunzhe Jia, James Bailey, Kotagiri Ramamohanarao, Christopher Leckie, and Michael E Houle. Improving the quality of explanations with local embedding perturbations. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 875–884, 2019.

[32] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Loop: local outlier probabilities. In Proceedings of the 18th ACM conference on Information and knowledge management, pages 1649–1652, 2009.

[33] Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. Copod: copula-based outlier detection. arXiv preprint arXiv:2009.09463, 2020.

[34] Ines Ben Kraiem, Faiza Ghuzzi, Andrè P’Eninou, Geoffrey Roman-Jimenez, and Olivier Teste. Human-interpretable rules for anomaly detection in time-series. In INTERNATIONAL CONFERENCE ON EXTENDING DATABASE TECHNOLOGY, pages 457–462. OpenProceedings. org, 2021.

[35] Chengliang Chai, Lei Cao, Guoliang Li, Jian Li, Yuyu Luo, and Samuel Madden. Human-in-the-loop outlier detection. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pages 19–33, 2020.

[36] Anna Fariha, Ashish Tiwari, Arjun Radhakrishna, and Sumit Gulwani. Eustune: Explaining tuple non-conformance. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pages 2741–2744, 2020.

[37] Ted E Senator, Henry G Goldberg, and Alex Memory. Distinguishing the unexplainable from the merely unusual: adding explanations to outliers to discover and detect significant complex rare events. In Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, pages 40–45, 2013.

[38] Nikolaos Myrtakis, Ioannis Tsamardinos, and Vassilis Christophides. Proteus: Predictive explanation of anomalies. In 2021 IEEE 37th International Conference on Data Engineering (ICDE), pages 1967–1972. IEEE, 2021.

[39] Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. Fast and scalable outlier detection with approximate nearest neighbor ensembles. In International Conference on Database Systems for Advanced Applications, pages 19–36. Springer, 2015.

[40] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Outlier detection in axis-parallel subspaces of high dimensional data. In Pacific-asia conference on knowledge discovery and data mining, pages 831–838. Springer, 2009.

[41] Saket Sathe and Charu C Aggarwal. Subspace outlier detection in linear time with randomized hashing. In 2016 IEEE 16th International Conference on Data Mining (ICDM), pages 459–468. IEEE, 2016.

[42] Vincent Jacob, Fei Song, Arnaud Stiegler, Bijan Rad, Yanlei Dao, and Nesime Tatbul. Exathlon: A benchmark for explainable anomaly detection over time series. Proc. VLDB Endow., 14(11):2613–2626, 2021.

THEODOROS TOLIOPOULOS is a PhD candidate at Aristotle University of Thessaloniki working in the field of big data management and more specifically with big data streams. He graduated from the Dept. of Informatics of Aristotle University of Thessaloniki, Greece in 2013 (BSc) and 2018 (MSc). He has participated as a research assistant in several projects. His main research interests are in the areas of outlier detection and continuous stream analysis.

ANASTASIOS GOUNARIS is an associate professor at the Aristotle University of Thessaloniki, Greece. A. Gounaris received his PhD from the University of Manchester (UK) in 2005. His research interests are in the areas of large-scale data management, massive parallelism, workflow and business process optimization, big data analytics and data mining. More details can be found at http://datalab.csd.auth.gr/gounaris/.