The VLDB Journal (2022) 31:977–1008
https://doi.org/10.1007/s00778-021-00721-1

SPECIAL ISSUE PAPER

A survey on outlier explanations

Egawati Panjei[1] · Le Gruenwald[1] · Eleazar Leal[2] · Christopher Nguyen[1] · Shejuti Silvia[1]

Received: 21 February 2021 / Accepted: 3 December 2021 / Published online: 26 January 2022

Abstract
While many techniques for outlier detection have been proposed in the literature, the interpretation of detected outliers is often left to users. As a result, it is difficult for users to promptly take appropriate actions concerning the detected outliers. To lessen this difficulty, when outliers are identified, they should be presented together with their explanations. There are survey papers on outlier detection, but none exists for outlier explanations. To fill this gap, in this paper, we present a survey on outlier explanations in which meaningful knowledge is mined from anomalous data to explain them. We define different types of outlier explanations and discuss the challenges in generating each type. We review the existing outlier explanation techniques and discuss how they address the challenges. We also discuss the applications of outlier explanations and review the existing methods used to evaluate outlier explanations. Furthermore, we discuss possible future research directions.

Keywords Outlier explanation · Outlier interpretation · Outlier description · Outlier detection · Anomaly analysis

1 Introduction

Hawkins [40] defines an outlier as "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism." Outliers are also called anomalies, abnormalities, aberrations, contaminants, deviants, discordant observations, exceptions, peculiarities, or surprises in some applications [4,20]. Outlier detection plays an important role in many applications. Identified outliers reveal meaningful information about abnormal behavior in a system. For example, using outlier detection algorithms, medical and public health researchers can identify unusual patient symptoms that can be indicative of medical errors or unusual outcomes [96]. Outlier detection also finds applications in environmental monitoring [42,56], structural monitoring [14,28], network intrusion [12,99], and fraud detection [46,98].

✉ Egawati Panjei
egawati.panjei@ou.edu

Le Gruenwald
ggruenwald@ou.edu

Eleazar Leal
eleal@d.umn.edu

[1] School of Computer Science, The University of Oklahoma, Norman, OK, USA

[2] Department of Computer Science, University of Minnesota Duluth, Duluth, MN, USA

For applications to benefit more from the results of the outlier detection process, the results should be explainable. To this end, the process should include two tasks: *outlier detection* and *outlier explanation*. The Merriam-Webster dictionary defines explanation as "the act or process of explaining." To explain is "to make known" or "to give the reason for or cause of" or "to make something plain or understandable." In [70], Miller argues that "explainable artificial intelligence can benefit from existing models of how people define, generate, select, present, and evaluate explanation." In the context of outlier detection, the outlier explanation task provides guidance for users in investigating detected outliers.

Explanations will enhance the users' understanding of outliers and can be used to improve the outlier detection task further. As a result, explanations can assist outlier mitigation, which is a process for deciding what to do with the identified outliers and how to utilize them to improve predictive models, such as future web traffic for the network location [95], finance and asset pricing [2], and waterborne applications [80]. However, a discussion about outlier mitigation techniques is beyond the scope of this survey paper.

While there are surveys on outlier detection techniques [8,17,20,43,74,105,107], they mainly focus on the outlier detection task. To fill this gap, in this paper, we focus on the outlier explanation task. Specifically, we define different kinds of outlier explanations (Sect. 2), discuss challenges that an outlier explanation technique needs to address (Sect. 3),

and discuss the applications of outlier explanations (Sect. 4). We then review existing techniques that provide such explanations and how they address the challenges described in Sect. 3 and summarize the properties of the surveyed techniques (Sects. 5–8). We also review methods that evaluate outlier explanations (Sect. 9). Finally, we discuss possible future research directions (Sect. 10).

2 Types of outlier explanations

Different aspects of what constitutes a "good" explanation from a human's perspective and their implications for interpretable machine learning have been discussed in the literature [70,71]. As summarized in [71], there are seven aspects of human-friendly explanations: (1) *explanations are contrastive*, meaning they highlight the most significant difference between the object of interest and other objects; (2) *explanations are selected*, meaning that even if a real-world event has many causes, people seek only one or two of these as explanations; (3) *explanations focus on the abnormal*, meaning humans tend to focus on the uncommon causes that had low probability yet happened; (4) *explanations are social*, meaning the content and nature of the explanations depend on the target audience; (5) *explanations are truthful*, meaning the explanations are valid in reality; (6) *good explanations are consistent with prior beliefs of the explainee* since humans incline to disregard information that is irrelevant to their prior beliefs; and (7) *good explanations are general and probable*, meaning explanations are good when they can explain many events.

In the context of the outlier explanation task, we can group the above aspects of human-friendly explanations into two groups: (i) non-evaluative aspects and (ii) evaluative aspects. The first group, consisting of Aspects (1) to (3), refers to those aspects associated with the explanations generated by an outlier explanation algorithm regardless of how the explanations are evaluated, while the second group, consisting of Aspects (4) to (7), refers to the criteria used to evaluate the generated explanations.

For example, Aspect (1), *explanations are contrastive*, is a non-evaluative aspect because it is possible to ascertain that a given outlier explanation algorithm generates explanations that highlight the differences between outliers and inliers before the explanations are evaluated. A similar thing can be said about the other two non-evaluative aspects (2) and (3): it can be shown that an outlier explanation algorithm produces selected explanations and focuses on the abnormal before the explanations are evaluated. Using the evaluative aspects, the explanations can then be evaluated based on whether they meet the target audience (Aspect (4)), whether they are accurate (Aspect (5)), whether they are consistent with the prior's belief of the users (Aspect (6)), and how many outliers are covered by the explanations (Aspect (7)).

In this survey, we classify outlier explanations based on the first group, non-evaluative aspects, into three types: (a) *importance levels of outliers*, (b) *causal interactions among outliers*, and (c) *outlying attributes*. We leave the discussion of the second group, evaluative aspects, to Sect. 9, "Methods to Evaluate Outlier Explanations." We now explain why we have these outlier explanation types and how they connect to the non-evaluative Aspects (1)–(3).

Importance levels of outliers In real-world applications [87,97], users often need to examine multiple outliers within a time constraint; therefore, users need to prioritize effort on investigating the outliers. The priority is set based on the importance levels or ranking of the outliers. Without the ranking, users are left with no guidance of where to begin the investigation. Hence, even though the ranking does not tell why some objects are deemed anomalous, it tells users which outliers to investigate first, second, and so on. In other words, it reveals the position or standing of each object within the set of outliers by contrasting the priority level among the outliers. Thus, it is a *contrastive explanation* (Aspect (1)).

Causal interactions among outliers When examining outliers, knowing which outliers cause other outliers can help users better understand the outliers. Should they need to prevent similar outliers from happening in the future, they can act on the outliers that cause those outliers to happen. However, it is possible that the detected outliers are not the only forces that cause other outliers. Hence, when a causal interaction among outliers is used to explain outliers, it is a *selected explanation* (Aspect (2)). It also fits Aspect (3), *explanations focus on the abnormal,* as it focuses only on finding outliers' causes from the set of anomalous data. For example, given an outlier x that caused an outlier y, users can ask, "Would y have been a normal object if x had been eliminated?"

Outlying attributes Siddiqui et al. [86] state that the amount of effort that users need to investigate an outlier is roughly related to the number of attributes or features associated with the outlier. In practice, users may have to deal with hundreds or thousands of attributes. Hence, knowing the outlying attributes, which are the attributes that are responsible for the abnormality of outliers, can reduce the effort spent. Outlying attributes relate to Aspect (1), *explanations are contrastive*, because they highlight features that make outliers significantly different than inliers. Furthermore, they also fit Aspect (3), *explanations focus on the abnormal*, because they show which features have abnormal values such that the outlier detection identifies some objects as anomalies.

In the following subsections, we formally define each type of outlier explanation.

2.1 Importance levels of outliers

Two different strategies can be used to convey the level of importance of outliers: *numerical ranking* and *categorical ranking of outliers*, which we now describe.

2.1.1 Numerical ranking of outliers

A numerical outlier ranking consists in ordering the data objects based on their outlier scores. The outlier score of an object is a real-number value generated by an outlier scoring function. The more an object is deviated from other objects in the datasets, the higher its outlier score. We formally define the outlier scoring function in Definition 1.

Definition 1 *(Outlier Score)* Given a dataset $X = \{x_1, \ldots, x_N\}$ and a deviation function $h : X \to \mathbb{R}$ that computes the deviation of x_i to all the instances in X, the outlier score of an instance in X is computed by $f : X \to \mathbb{R}$ such that for any instances $x_i, x_j \in X$, $h(x_i) > h(x_j)$ if and only if $f(x_i) > f(x_j)$.

Adequate numerical ranking systems for outliers should prioritize more deviant data objects over less deviant ones. Therefore, overall, the strongest outliers are ranked first, and the normal data is ranked last [83]. We formally describe the numerical ranking of outliers in Definition 2.

Definition 2 *(Numerical Ranking of Outliers)* Given a set of n outliers $O = \{o_1, \ldots, o_n\}$ and an outlier score function $f : O \to \mathbb{R}$ that gives an outlier score to each outlier, *a numerical ranking of outliers* is a bijective function $g : O \to \{1, 2, \ldots, n\}$ such that for any outliers $o_i, o_j \in O$, $f(o_i) > f(o_j)$ if and only if $g(o_i) > g(o_j)$.

For example, in Fig. 1, x_1 is deviated from other objects more than x_2; hence, x_1's outlier score is higher than x_2's ($f(x_1) > f(x_2)$). As a result, the outliers' numerical rankings will rank x_1 higher than x_2 ($g(x_1) > g(x_2)$).

We can see the importance of a numerical ranking of outliers in many applications. For example, a numerical ranking of outliers can help system administrators prioritize actions when an alarm for network intrusion detection is raised. Viswanathan et al. [97] rank outliers found in a data center to minimize the burden of system administrators needing to handle abnormal behaviors across servers. The scale and complexity of a data center are much larger than the number of system administrators, whose time is limited. Often, the system administrators have to deal with false alarms but need a way to deal with them quickly to move on to the true positive. Outlier ranking is a method that helps speed up the process.

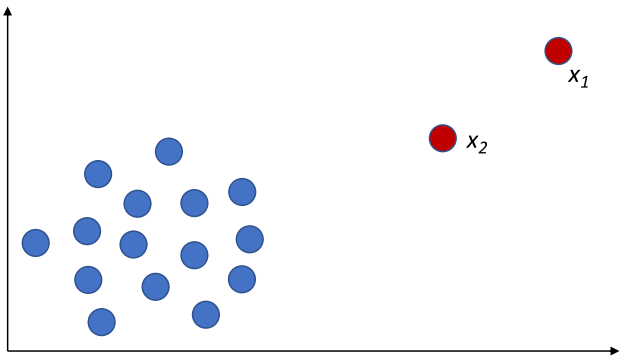

Fig. 1 Example of two outliers in a dataset

2.1.2 Categorical ranking of outliers

The importance level of outliers is usually described numerically; however, it can also be described categorically. Outliers can be grouped into different categories based on some importance-level criteria. The categorization tells users which outliers should be prioritized, and thus, it represents a categorical ranking of outliers. We define the categorical ranking of outliers in Definition 3.

Definition 3 *(Categorical Ranking of Outliers)* Given a dataset $X = \{x_1, x_2, \ldots, x_N\}$, a set of n outliers $O = \{o_1, \ldots, o_n\} \subseteq X$, a deviation function $h : X \to \mathbb{R}$ that computes the deviation of x_i to all the instances in X, and a finite set of m categories C with a total order $\preceq_C$ that corresponds to the levels of importance of the categories, a *categorical ranking of outliers* is a function $g : O \to C$ such that for any outliers $o_i, o_j \in O$, $g(o_i) \preceq_C g(o_j)$ if and only if $h(o_i) \leq h(o_j)$.

For example, Knorr and Ng [50] define the outlier categories $C = \{$"trivial outlier," "weak outlier," "strongest outlier"$\}$ to help gain better insights about the nature of outliers. They define an anomalous data point o as the "strongest outlier" in a subspace A if it meets two criteria: (i) o is not an outlier in any subspace $B \subset A$, and (ii) no outlier exists in any subspace $B \subset A$. If o does not satisfy the criteria in (i), then it is a "weak outlier." If it does not fit the two criteria, then it is a "trivial outlier." The terms "trivial outlier," "weak outlier," and "strongest outlier" are used to separate noise from meaningful abnormal data [4]. Noise can be ignored and needs to be removed from the data so that it does not affect the result of data mining, while further analysis should be conducted on meaningful outliers. For example, in personalized medicine, this process is essential as data can be very noisy due to changes in the laboratory environment or incomplete objectives of diagnostic decisions [73].

2.2 Causal interactions among outliers

A relationship among outliers can be conveyed as a causal interaction in which an outlier can lead to the occurrence of

other outliers. We describe this type of outlier explanation in Definition 4 based on the outlier causal relationship definition of Liu et al. [62] and the database causality definition of Meliou et al. [67].

Definition 4 *(Outlier Causal Interactions)* Given a set of n outliers $O = \{o_1, \ldots, o_n\}$ and its corresponding timestamp set $T = \{t_1, \ldots .t_n\}$, an outlier o_i is said to have caused an outlier o_j to occur if and only if:

- o_i's timestamp is older than o_j's timestamp ($t_i < t_j$), that is, o_i precedes o_j.
- a removal of o_i from O also removes o_j from O.

An example of causal interactions among outliers can be found in traffic applications, where after examining the traffic anomalies, one can uncover that some anomalies lead to others [62,100]. For example, studying a set of traffic anomalies in Oklahoma City on Saturday afternoon reveals that a traffic jam on South East 15th Street at 1:10 pm had a causal interaction with a traffic jam on East Grand Boulevard at 1:15 pm. It was because the traffic flow heading from East Grand Boulevard to South East 15th Street was delayed due to the traffic congestion in South East 15th Street.

2.3 Outlying attributes of outliers

Outlying attributes of outliers refer to the feature subspace where outliers are highly deviated from the normal data or, in other words, they refer to the attributes that contribute the most to the abnormality of the outliers. This type of outlier explanation can be the *outlying attributes of an individual outlier* or *of a group of outliers*.

2.3.1 Outlying attributes of an individual outlier

Outlying attributes of an individual outlier refer to the feature subspace or the subset of attributes responsible for the abnormality of the outlier. There are two major interpretations of this outlier explanation. Some algorithms [47,86] interpret it as the smallest subspace where the outlier score is greater than a threshold (Definition 5), while others [61,68,90] interpret it as the subset of attributes where each member has a contribution score higher than a threshold (Definition 6).

Definition 5 *(Outlying Attributes of an Outlier—Interpretation 1)* Given an outlier o, a set of d dimensions $D = \{A_1, A_2, \ldots, A_d\}$ where $o \in A_1 \times A_2 \times \cdots \times A_d$, an outlier score function f, and an outlier score threshold τ, *the outlying attributes of* o is a subspace $S \subseteq D$ such that the projection of o onto S, denoted as $\pi_S(o)$, has $f(\pi_S(o)) > \tau$ and $\nexists S' \subseteq D \mid (|S'| < |S|) \wedge (f(\pi'_S(o)) > \tau)$.

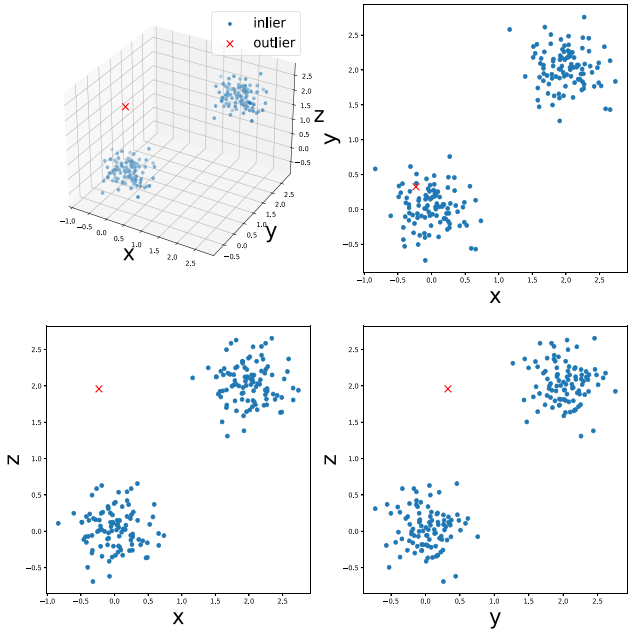

Fig. 2 An outlier detected in the 3D space *(x, y, z)* and projected on the 2D subspace

Definition 6 *(Outlying Attributes of an Outlier—Interpretation 2)* Given an outlier o, a set of d dimensions $D = \{A_1, A_2, \ldots, A_d\}$ where $o \in A_1 \times A_2 \times \cdots \times A_d$, an attribute's contribution score function $h : D \rightarrow \mathbb{R}$ that generates a real-value quantifying the contribution of each attribute to the abnormality of o, and a contribution score threshold $\gamma \geq 0$, *the outlying attributes of* o is a subspace $S \subseteq D$ such that $\forall A_i \in S, h(A_i) > \gamma$.

An object can be detected as an outlier in the whole dimension space D; however, the outlier can be well discriminated from the inliers in the succinct subset S of the original attributes [24,68]. For example, in Fig. 2, the red point is an outlier in the 3-dimensional space $D = \{x, y, z\}$, but the 2-dimensional projection shows that $S = \{x, z\}$ or $S = \{y, z\}$ could be the outlying attributes of the red point based on Definition 5. However, suppose the red point has the following contribution scores for its attributes $\{x = .1, y = .1, z = .8\}$ and the contribution score threshold is .2, the red point outlying attributes is $\{z\}$ based on Definition 6.

Now imagine human analysts have to deal with hundreds or thousands of attributes as in the case of e-commerce or healthcare systems. Whenever a data point is flagged as abnormal by an outlier detector, analysts need to manually go through the feature space to identify the subset of attributes responsible for the detection to verify whether the data object is a true outlier. Knowing the subspace where an outlier stands out can reduce the amount of work/time analysts need to judge the status of the outlier. In addition, since less time

is required to examine an outlier, analysts can review more flagged points during a time period.

2.3.2 Outlying attributes of a group of outliers

Outlying attributes of a group of outliers refer to the subset of attributes responsible for the abnormality of every outlier in the group. In other words, it is the feature subspace where these outliers deviate from normal data in the dataset. We formally define this concept in Definition 7.

Definition 7 *(Outlying Attributes of a Group of Outliers)* Given a set of n outliers $O = \{o_i | 1 \leq i \leq n\}$, a set of d dimensions $D = \{A_1, A_2, \ldots, A_d\}$ where $o_i \in A_1 \times A_2 \times \cdots \times A_d$, an outlier score function f, an outlier score threshold τ, and a function that maps O into k clusters $\mathscr{G}$, where each $G \in \mathscr{G}$ represents a group of outliers, *the outlying attributes of* G is $S \subseteq D$ such that $\forall o_j \in G$, $f(\pi_S(o_j)) > \tau$, where $\pi_S(o_j)$ is a projection of o_j onto S.

Similar to the outlying attributes of an individual outlier, this explanation is practical, especially for high-dimensional datasets, as examining every subset of their attributes is inefficient and even infeasible. Knowing which attributes are responsible for a group of outliers can help analysts. Instead of verifying every single outlier, analysts can verify the outlying attributes of each group of outliers at once. Furthermore, this explanation helps analysts identify potentially critical, repeating outliers.

3 Challenges in generating outlier explanations

We now discuss the challenges in generating each type of outlier explanation described in Sect. 2.

3.1 Challenges in generating numerical rankings of outliers

We identify the following challenges when it comes to generating a numerical ranking of outliers:

(a) *Unifying various outlier scores*
Outlier detection algorithms generate outlier scores that vary widely in their scale and range [54,61]. Some outlier detectors only provide binary values indicating whether an object is an outlier or an inlier. Some generate outlier scores in continuous values with various ranges. Furthermore, the same algorithm can produce different ranges of outlier scores because of the change in data distribution in the same application [54]. To provide a numerical ranking of outliers, the scores must be in the same scale and range. Hence, the explanation task needs to unify the scores before generating the ranking. Furthermore, the unified scores should have a clear contrast that differentiates outliers from normal data.

(b) *Incorporating user's feedback to reduce false-positive rate*
The numerical ranking provided by the outlier explanation task helps analysts prioritize their actions. When analysts are presented with a ranked list of outliers, they will investigate those outliers according to the given order [87]. After examining those outliers, analysts give feedback on whether each outlier is of interest or is a false positive outlier. Hence, the challenge is to incorporate the feedback into the detection and explanation tasks to reduce the false-positive rate and thus provide a more accurate ranking system. However, to answer this challenge, the algorithm used in the detection and explanation tasks must be an online version such that it can be updated without having the entire input available from the start.

3.2 Challenges in generating categorical rankings of outliers

We elucidate the challenges in generating a categorical ranking of outliers as follows:

(a) *Defining the levels of importance criteria*
The categorical ranking of outliers can vary from one application to another application. It can also depend on the type of outlier detection algorithm used to identify the outliers. One should define the levels of importance criteria by taking into account these two aspects. The criteria determine the number of categories to provide.

(b) *Incorporating user's feedback to reduce misclassified outliers*
Similar to the numerical ranking of outliers, incorporating users' feedback is also necessary to reduce false positives or misclassified outliers. This challenge requires an online model so that the feedback can be included without having to redo the computation from the beginning.

(c) *Selecting an efficient function to map the detected outliers into each category*
Once the importance level criteria are determined, the next challenge is to define a function to label each outlier into a category. Knorr & Ng's technique [50] generates this explanation by examining the subspaces from the lower cardinality to the higher one. It requires multiple passes on the dataset, and thus, it requires a lot of I/Os. An efficient function should minimize the number of I/Os required when categorizing a set of outliers.

3.3 Challenges in generating causal interactions among outliers

We describe the challenges in finding the causal interactions among outliers as follows:

(a) *Availability of outliers' timestamps*
As described in Definition 3, to tell which outlier causes which outlier to occur, the timestamp information indicating when an outlier occurs must be available. If a dataset does not have this information, we cannot generate this outlier explanation.

(b) *Defining a data structure to capture the causal interactions among outliers*
Even though when the timestamp information is available, finding the causal interactions among outliers in a dataset is not a trivial task. One needs to define a data structure that can capture the causal interactions in such a way that they can be processed efficiently. Liu et al. [62] use trees, while Xing et al. [100] apply directed acyclic graphs (DAGs).

3.4 Challenges in generating outlying attributes of an individual outlier

We identify the following challenges in finding the outlying attributes of an outlier:

(a) *Limiting subspace search*
To find a subset of attributes responsible for the outlierness of an anomalous object, one can examine all the possible combinations of attributes and compute the object's outlier score in every attribute space; however, this approach is infeasible when the number of attributes is large. Some techniques limit the search space by using a heuristic approach; however, the approach does not guarantee to find the optimum subspace, which is the subspace where the outlier is the most abnormal [47,50]. Instead of searching the subspace, some techniques [24,68] depend on the local neighborhood of each outlier to extract its outlying attributes, and some other techniques [89,90] use interpretable models to measure the contribution of each feature/attribute to the abnormality of the object.

(b) *Generating readily interpretable output*
In layman's terms, Occam's Razor principle [16] can be stated as "the simplest explanation is almost always the best." This principle implies that a technique that generates outlying attributes should minimize the number of features or attributes included in its output. Some existing techniques [47,86] generate a list of outlying attributes, while some other techniques [61,90] list all the original attributes and their corresponding contribution scores. The latter requires more effort from users to set a contribution threshold to determine the outlying attributes. Furthermore, even though some techniques [47,86] produce a set of outlying attributes, users can further benefit if the explanation output also includes the outlying attributes' values. Of course, the output's description should be presented in a human-interpretable way.

(c) *Incorporating user's prior knowledge about the attributes*
Sometimes, analysts already have some knowledge about which attributes are relevant to anomalous behavior. Hence, we should take into consideration this prior knowledge when generating outlying attributes. The question is how we can quantify such knowledge so that the outlier explanation algorithm can process it. For example, if analysts suspect that an attribute A_i is responsible for the abnormality in a dataset, should they use a real value to weigh A_i? What is the weight value range? Should they categorize the attributes based on prior knowledge?

3.5 Challenges in generating outlying attributes of a group of outliers

The challenges in generating outlying attributes of each individual outlier are also applied to a group of outliers as discussed in Sect. 3.4. In the following, we describe an additional challenge in generating outlying attributes of a group of outliers.

(d) *Finding discriminative outlying attributes*
In order to form groups from a set of outliers, one needs to discover the similarity among them. An outlier that shares the same properties with another outlier should fall into the same group. However, it is possible that an outlier shares different properties with different outliers. For example, outlier o_i shares the same outlying attributes S_1 with outlier o_j, but it also shares the outlying attributes S_2 with outlier o_k. Should the outlier explanation technique group o_i with o_j or o_k? Can all three objects be put into one group? If these outliers are grouped together, can the outlier explanation technique ensure that the outlying attributes of the group are discriminative and separate outliers from inliers sufficiently?

4 Applications of outlier explanations

The outlier explanation task is a part of outlier analysis. Therefore, it is relevant in real-world problems where the out-

lier detection task is applied. However, most published papers on outlier explanation techniques are generic [10,21,24,25,44,47,50,51,54,61,64,68,86,87,90]; only a few are designed for domain-specific applications [62,88,89,97,100,104]. In this section, we discuss some applications of outlier explanations.

4.1 Intrusion detection

Milenkoski et al. [69] state that "Intrusion detection is a common cyber security mechanism whose task is to detect malicious activities in the host or network environments." From a computer security perspective, a system/environment is considered secure if it has the properties of confidentiality, integrity, and availability of its data and services. Any attempts to violate these security properties are considered as attacks or intrusion. Outlier detection is applicable in the intrusion detection systems (IDS) because intrusions are different from the expected behaviors of the system [20].

Typically, an IDS needs to deal with a massive volume of data that arrive in streaming fashion. To stop ongoing attacks, the detection of malicious activities requires a timely reaction. However, the number of alarms raised can make analysis overwhelming for users. Therefore, an outlier explanation such as *a numerical or categorical ranking of outliers* is necessary to tell analysts which alerts they need to focus on first. Notice that an IDS can employ multiple outlier detectors across servers. Viswanathan et al.'s technique [97] described in Sect. 5.1.2 provides a way to rank alarms raised by various detectors in a data center.

An IDS also deals with multidimensional data. For example, Avritzer et al. [12] monitor so-called performance signatures to trigger alerts of five types of security attacks: denial of service (DOS), SQL injection, man-in-the-middle (MITM), buffer overflow, and stack overflow. The performance signatures employ system usage-based attributes such as: (i) CPU percentage, (ii) number of active threads, (iii) interface received bytes per sec, (iv) swap percentage, (v) number of TCP connections established, (vi) number of TCP resets, (vii) interface transmitted bytes per second, (viii) virtual memory usage, (ix) working set in bytes, and (x) memory percentage. Avritzer et al. assume that "the performance of the well-behaved system can be measured such that performance signatures of several types of attack can be identified." This assumption implies that when an alert has an explanation of the *outlying attributes* type that aligns with a known attack, the IDS can inform users of the type of the attack. However, when the *outlying attributes of an alert* do not fit any known attacks, analysts can use the information to either flag the alert as a false alarm or investigate it further to define a new type of attack. Furthermore, a collection of the unknown attacks can be grouped using XPACS [64] described in Sect. 7.2.4. The grouping will speed up the analysts' works on identifying new types of attacks.

4.2 Fake news detection

Fake news is "news stories that are false: the story itself is fabricated, with no verifiable facts, sources or quotes" [29]. It is part of the larger environment of misinformation and disinformation. According to the Merriam-Webster dictionary, misinformation is "incorrect or misleading information," while disinformation is "false information deliberately and often covertly spread (as by the planting of rumors) in order to influence public opinion or obscure the truth." In recent years, fake news articles have increased through social media. It is very concerning, especially during the COVID-19 pandemic, because fake news can cost lives [34].

The spreading of fake news can be regarded as an anomalous behavior in social networks [58,101]. Fake news tends to have poor grammar, contain bad language, and refer to vague or untraceable sources. In addition, it is often posted by bogus accounts that use misleading names, images, or bogus web addresses [34]. These are some factors that social media users can use to spot fake news. They can also be used as input features for a fake news detection algorithm. For example, in [58], Li et al. categorize the factors into four feature types: (i) *text content features*, such as number of positive sentiment words, whether the news contains question marks, etc.; (ii) *propagation features*, such as number of comments, number of likes, and number of retweets; (iii) *image feature* indicating whether the image in the article is tampered; and (iv) *user features*, which are features related to the users who publish the news, such as account age, follower–friend ratio, number of tweets, etc. These features are combined into a features vector used as input for an autoencoder. An autoencoder is an unsupervised neural network that has been used for anomaly detection in other applications [32,63]. The autoencoder will determine whether a news article is fake or real. Shareholders (i.e., Twitter watch group) can use this information to warn people. However, the autoencoder is a black-box model [37]. In order to build trust, it is crucial to explain why a news article is flagged fake. Thus, we can use a decision tree-based explainer (part of EXAD [89]) described in Sect. 7.1.3 to explain the black-box model. This technique generates an explanation of the *outlying attributes* type that tells which relevant features the autoencoder uses to flag a fake news article.

4.3 Fraud detection

Fraud detection uncovers malicious or criminal activities in organizations such as credit card companies, banks, insurance companies, online auctions, telecommunication companies, etc. Outlier detection algorithms have been used

widely for fraud detection, and some survey papers have a lengthy discussion about them [1,8,20], but can a fraud detection application benefit from the outlier explanation task?

Fraud detection, for example, credit card fraud detection, deals with millions of transactions every day, and most are legitimate transactions. However, it was reported by Javelin Institute [77] that in 2014, one in six legitimate cardholders experienced at least one decline because of suspected fraud. This false-positive detection impacted merchants because following the decline, the customers reduced their patronage and even stopped shopping with the merchants where their cards were declined. To mitigate this situation, most companies have human analysts monitoring all transaction alerts 24/7 [98]. An outlier explanation in the form of *rankings of the alerts* can help human analysts prioritize their effort in this costly operation.

A transaction is represented by a number of attributes, such as merchant-related attributes (unique id, bank of the merchant, type of the merchant, country), transaction-related attributes (amount, timestamp, currency, presence of a customer), terminal-related attributes (device type, how data is input into the terminal, service or not), etc. [98]. Fraud detection flags a transaction based on those attributes. Therefore, if an explanation about *the outlying attributes of a suspected fraudulent transaction* is made available, the analyst can make the decision faster. Notice that the transaction's attributes are a combination of categorical and numerical attributes. Thus, the outlier explanation techniques, such as SFE [86] (described in Sect. 7.1.1) and Explainer [52] (described in Sect. 7.1.3), can be used to provide such an explanation.

4.4 Medical data analysis

Outlier detection plays a vital role in the prediction and diagnosis tasks in medical data analysis [33]. It can uncover important information about patients based on the given physiological data. The data can be biomedical images such as X-ray radiography, computed tomography scan (CT), and magnetic resonance imaging (MRI). These data are used for detecting bone fractures, certain types of tumors, or tissue damage. The other kind of data is electrical biomedical signals, such as the electrocardiogram (ECG) used to monitor heart's activities, electroencephalogram (EEG) used to study brain damage from a head injury, and magnetoencephalography (MEG) used to identify abnormal brain conditions.

Explaining outliers detected in medical data is critical to ensure that practitioners and patients trust the system to make accurate predictions or diagnoses. For example, ECG signals are often mangled by artifacts that have no relation to the heart functions [59]. The artifacts can be caused by device errors or by motion. Hence, when an outlier detection is applied on ECG signals to diagnose a cardiac condition, the *categorical ranking of outliers* is necessary to help analysts filter the artifacts (noise) from real outliers.

Furthermore, ECG signals are extracted into morphological and derived features (Li and Boulanger [59] provide a comprehensive list of these features). These features are used as inputs to determine whether the heartbeat is regular or irregular. Therefore, knowing the *outlying attributes* of each irregular heart rate will help physicians decide whether the flag is true or false. Regardless of the type of outlier detection algorithms used, one can generate *outlying attributes* using, for example, Lookout [39] (Sect. 7.1.1), LODI [25] (Sect. 7.1.2), and COIN [61] (Sect. 7.1.2). Moreover, as deep learning is becoming popular in medical anomaly detection [33], the techniques proposed by Song et al. [89] (Sect. 7.1.3) and Amarasinghe et al. [10] (Sect. 7.1.4) can be helpful to provide *outlying attributes* to explain the neural network decision.

4.5 Structural health monitoring

Structural Health Monitoring (SHM) is "the process of implementing a damage detection strategy for aerospace, civil or mechanical engineering infrastructure" [31]. Early detection of damages, such as cracks and corrosion, can reduce maintenance costs and prevent catastrophic events [14]. SHM combines sensor technologies and digital twins, i.e., virtual representations, to enable continuous observations of the structures of interest. For example, SHM for civil infrastructure (e.g., buildings, bridges, dams) employs hundreds of sensors monitoring environmental conditions, such as temperature, humidity, and wind speed. It also includes sensors monitoring structural response, such as acceleration, deflection, and strain [66].

Outlier detection algorithms have been applied in SHM [14,66]. For example, Bigoni and Hesthaven [14] employ a one-class classifier outlier detection algorithm for each sensor so that it is possible to locate the damage on the structure of interest. They extract damage-sensitive engineering-based features from the raw signals generated by each sensor and use them as inputs for the outlier detection algorithm. The detector will tell whether any subsequent data object belongs to a group of what is considered as healthy signals (inlier), or it is an outlier. Should the engineers be provided with the information of which features are responsible for the detection of an outlier (*the outlying attributes*), they can verify the finding faster and take action to fix the damage.

However, not all outliers indicate a structural fault. For example, data anomalies can be caused by sensor system malfunctions where the sensors can record data, but the recorded values are inaccurate/incorrect [66]. These data anomalies, e.g., data loss, spikes, drift, and excessive noise, pose challenges for data analysis and can render the SHM activity futile. Hence, *grouping the detected outliers based on the*

outlying attributes that they have in common is necessary. By doing so, the analyst can examine and identify the root cause of each subset of outliers at once. Grouping outliers can be done using an algorithm such as the XPACS [64] algorithm described in Sect. 7.2.4.

5 Techniques to find the importance levels of outliers

The importance levels of outliers can be defined using the *numerical* or *categorical ranking of outliers*. In this section, we survey the techniques used to obtain those types of explanations.

5.1 Techniques to find numerical rankings of outliers

Some techniques generate a numerical ranking of outliers only for a specific outlier detection algorithm, while some techniques do not depend on any outlier detection algorithms. These two groups of techniques, which we call *outlier detection model-specific ranking* techniques and *outlier detection model-agnostic ranking* techniques, respectively, are described in Sects. 5.1.1–5.1.2. Then, in Sect. 5.1.3, we discuss the advantages and disadvantages of the techniques based on the challenges explained in Sect. 3.1.

5.1.1 Outlier detection model-specific ranking techniques

Some of the existing outlier detection algorithms output binary values that flag whether an instance is an outlier or an inlier, while others output outlier scores for each instance [4]. The numerical ranking of outliers is obtained by ordering the instances in the dataset by their outlier scores. In their survey of outlier detection methods, Chandola et al. [20] discuss how the existing outlier detection algorithms compute outlier scores. SVM-based outlier detection algorithms associate the outlier score of an instance with the probabilistic prediction score obtained from a classifier. In neural network-based outlier detection, the reconstruction error (the average of the sum squared errors between the target and neural network output of an instance) is the outlier score. Rule-based outlier detection algorithms use the inverse of the confidence associated with the best rule as the outlier score. Nearest neighbor-based outlier detection algorithms utilize the inverse of the number of k nearest neighbors of an instance or the inverse of the standard deviation of the local densities of the nearest neighbors of an instance as the outlier score. Clustering-based outlier detection algorithms apply the distance of an instance to its closest cluster centroid as the outlier score. Parametric statistics-based outlier detection algorithms adopt several ways to define the outlier score of an instance, such as the inverse of the PDF (probability density function) and the distance of an instance to the estimated mean. They also use the magnitude of the residual in the regression model in which, after fitting data into the model, the residual for each test instance is utilized to determine its outlier score. Histogram-based methods use the height (frequency) of the bin in which an object falls as the outlier score.

5.1.2 Outlier detection model-agnostic ranking techniques

Different outlier detection algorithms can generate different scoring results. Several techniques are proposed to provide unified outlier scores or improve outlier detection algorithms' outlier scores. Those techniques use either (i) *regularization and normalization*, (ii) *false-positive rate*, (iii) *nearest normal neighbors-based scoring*, or (iv) *user's feedback adaptive scoring*, which we now describe.

Regularization and normalization-based scoring Kriegel et al. [54] propose a framework to unify outlier scores through regularization and normalization. Their framework aims to establish enough contrast between outlier and inlier scores and obtain a rough probability value [0, 1] that defines an instance's outlierness. Regularization can be achieved in three ways: baseline regularization, linear inversion, and logarithmic inversion. Baseline regularization is applied for outlier detection methods of which the expected inliers score (base) is not 0, such as the LOF [18] and LDOF [106] methods. Linear inversion is used for outlier models that generate high scores for inliers. Logarithmic inversion is similar to linear inversion, yet it is for the outlier models that produce shallow contrast scores between outliers and inliers.

After applying regularization on the outlier scores, the next step is to apply normalization. The simplest way to do this is to use simple linear normalization; however, simple linear normalization does not add any contrast to the distribution of scores. Kriegel et al. [54] propose statistical scaling to normalize outlier scores. They suggest applying customized Gaussian scaling when working on high-dimensional data or Gamma scaling for low-dimensional datasets. An outlier score is transformed into a probability value using the cumulative distribution function and Gaussian error function of outlier scores. The authors define a formula called Gaussian scaling to estimate the mean and standard deviation, which are the parameters for the Gaussian error function. The same thing applies to Gamma scaling. They define a formula to estimate the Gamma distribution parameters and use these parameters to transform the outlier score to a probability value.

False-positive rate-based scoring Viswanathan et al. [97] define the rankings of outliers in a data center as the rating of outliers based on the false-positive rate across servers and metrics. The false-positive rate is the probability that an identified outlier is not an anomaly. The higher the probability, the lower the ranking of a particular outlier. They define the

metrics examined in the data stream and the time-based window size in which outliers are detected and evaluated during the detection task. The metrics in a data center can be the percentage of CPU idle time, I/O transfers per second, blocks read per second, blocks written per second, packets received per second, and packets sent per second. Each metric is associated with an outlier detector referred to as a local detector. For each time-based window data stream, the local outlier detectors identify the outliers of each metric.

The authors propose three statistics-based methods: *Gaussian approximation, Bernoulli approximation,* and *Extension of ranking to correlated metrics* to obtain the false-positive rate. *Gaussian approximation* assumes that each metric is independent, and the local detector for each time window calculates the probability. The Z-score of each metric denoted as Z_k is computed as $Z_k = \frac{M_k(t)-\mu_k}{\sigma_k}$, where $M_k(t)$ is the value of metric k at time t, μ_k is the mean of metric k, and σ_k is the standard deviation of metric k. During a time window of size W, the local detector collects the $Z_k(t-W+1), Z_k(t-W+2), \ldots, Z_k(t)$. Each local detector calculates the probability of the time window W, $P_k = c \times e^{-1/2 \sum_{j=t-W+1}^{t} Z_j^2}$ where c is a constant.

During the outlier explanation task, the local outlier detectors send the probability values to the central node to order them. The lower the probability, the higher the rank given. *Bernoulli approximation* also assumes that each metric is independent. Each local outlier detector calculates the number of samples (W_k) violating a given threshold (T_k) in a particular metric k during a time window. T_k represents the accepted value of an object to be considered as a normal object. At a time t, the central node receives the probability of the windows corresponding to $t-W+1$ to t (denoted as $t-W+1:t$) from all the local detectors. It then calculates the probability of observed events using the equation $P_k = p_k^{W_k(t-W+1:t)}(1-p_k)^{(W-W_k(t-W+1:t)}$ where p_k is the estimated probability that a particular metric k violates a given threshold. *Extension of ranking to correlated metrics* assumes that the metrics, for example, traffic metric and CPU metric, are not independent. It first computes a matrix of correlations between metrics $p(M_i, M_j)$ from the training data. For each metric M_k, it identifies the set of nearest neighbor metrics $N_k = j : p(M_k, M_j) > p*$ where $p*$ is a threshold for the correlation value which defines the minimum value required to define two metrics as neighbors. If the correlation value between two metrics is close to 1, then they are highly correlated. It then applies linear regression where M_k is a dependent variable and the other metrics belonging to N_k are independent variables. It calculates the predicted value of M_k using the linear regression equation and computes the residual R_k using the equation: R_k = *actual value of* M_k - *the predicted value of* M_k. Finally, it applies Gaussian or Bernoulli approximation on R_k to get the ranking.

Nearest normal neighbors-based scoring Liu et al. [61] propose the COIN algorithm, which takes into account prior knowledge about attributes contributing to the abnormality degree and local context of each outlier o_i. Local context (C_i) refers to the nearest normal neighbors of o_i based on the L_2 norm. COIN expands o_i into a hypothetical outlier class using the synthetic sampling method to balance the outlier data with normal data C_i. COIN then segments C_i into different clusters $\{C_{i,1}, C_{i,3}, \ldots, C_{i,L}\}$ using K-means or hierarchical clustering. After finding the local cluster $C_{i,l}$ of each outlier in the dataset, COIN uses linear classifiers to find hyperplanes that separate the outlier class and each local cluster $C_{i,l}$. It then computes the score of outlier o_i with the formula $d(o_i) = \frac{\sum_l |C_{i,l}| d_l(o_i)/\gamma_{i,l}}{|C_{i,l}|}$, where $d_l(o_i) = |w_{i,l}^T o_i|/||w_{i,l}||_2$, $w_{i,l}$ is the weight of the hyperplane associated with local cluster $C_{i,l}$, and $\gamma_{i,l}$ is the average distance of each instance to its closest neighbor in $C_{i,l}$.

Two vectors β and p are introduced to incorporate prior knowledge of attributes. β_m implies the relative degree of significance assigned to attribute a_m, while p_m is the prior knowledge on the contribution of attribute a_m in determining an instance as an outlier. $p_m = -1$ means attribute a_m tends to have a small contribution and $p_m = 1$ means the opposite, whereas $p_m = 0$ means there is no preference. Incorporating the prior knowledge, the outlier score of o_i with respect to $C_{i,l}$ is enhanced as $d_l(o_i) = ||\frac{|w_{i,l}^T o_i|}{\gamma_{i,l}||w_{i,l}||} \frac{w'_{i,l}}{||w_{i,l}||} \circ \beta||$ where $\circ$ represents the element-wise multiplication, $w'_m = \min(0, w[m])$ if $p_m = 1$ and $w'_m = \max(0, w[m])$ if $p_m = -1$. The outlier score of o_i when prior knowledge is included is $d(o_i) = \frac{\sum_l |C_{i,l}| d_l(o_i)}{|C_{i,l}|}$.

User's feedback adaptive scoring In [87], the authors propose the generalized linear anomaly detectors (GLADs) method in which outlier score is defined as a linear function

$$SCORE(x; w) = -w \cdot \phi(x)$$

where ϕ is a feature function that maps instance x to the n-dimensional weight vector w. One example of the GLADs family is the isolation forest (IF) [60]. Given a set of vector-valued instances, IF analyzes the dataset to construct a forest of randomized decision trees. Trees are constructed recursively, and each node in a tree represents a random feature. An instance that is different from other instances usually belongs to a leaf with low depth. IF assigns an outlier score to an instance x based on its average isolation depth across the randomized forest. The isolation depth of an instance in a tree is the depth of the leaf to which it belongs. The outlier score is the inverse of the average depth. When IF is represented as GLAD, each node in the trees except the root is weighted. The outlier score of an instance in a tree is the negative of the total sum of the weight multiplied by a binary indicator function. The value of the indicator function is 1

if the instance belongs to the node; otherwise, it is 0. The weight w is adjusted anytime the detector receives feedback.

GLADs uses online convex optimization (OCO) [85] to find the optimum weight (w) that can minimize the regret/loss function for a given period. The algorithm initializes the regularization parameter η and the prior weight vector w_0 that is also used to initialize θ_1. At each time t, the algorithm selects a new weight w_t, which is the member of the convex set that has the minimum Euclidian distance with θ_t. It then selects an instance x_t that has the maximum outlier score based on the weight w_t. OCO receives feedback y_t from the user; $y_t = 1$ if the user marks instance x_t as an outlier and $y_t = -1$ if x_t is marked as an inlier. x_t is then removed from the dataset and y_t is used to define the loss function f_t. The new value of θ_{t+1} is then computed by subtracting the regularized-weighted gradient descent of the loss function from θ_t, $(\theta_{t+1} = \eta\delta f_t(w_t))$. The authors suggest two types of loss function: linear loss and log-likelihood loss function. Linear loss function is defined as $f_r(w_t) = -y_t SCORE(x_t; w_t) = y_t w_t \phi(x_t)$. The log-likelihood loss function is defined as $f_r(w_t) = -y_t w_t \log(\exp(SCORE(x_t, w))/Z)$, where Z is a normalizing constant.

5.1.3 Discussion of the surveyed techniques on a numerical ranking of outliers

We now discuss the aspects of the techniques surveyed in Sect. 5.1.1 and Sect. 5.1.2 based on the challenges described in Sect. 3.1.

- Challenge (3.1.a): *Unifying various outlier scores*
  Three outlier detection model-agnostic ranking techniques address this challenge. The technique proposed by Kriegel et al. [54] and Liu et al. (COIN) [61] unifies the outlier scores by transforming them into values ranging from 0 to 1 that represent the probability of an instance of being an outlier. When an object's score is closer to 1, then it is considered an outlier. Contrary, Viswanathan et al.'s [97] unified score represents the probability of an instance of being an inlier. However, these techniques still depend on the user's expertise to determine the threshold when the score's gap between the outliers and the inliers is narrow. This threshold is essential to avoid false-positive outliers included in the ranking. The technique proposed by Kriegel et al. [54] increases the score contrast between outlier and inlier objects.
- Challenge (3.1.b): *Incorporating user's feedback*
  GLADs [87] is the only technique that can update the outlier scores based on the user's feedback; hence, it allows users to be proactive in determining the numerical ranking of outliers. However, it only quantifies the user's feedback of whether an object is a true outlier or not. It does not accept the input indicating whether an outlier is more abnormal than other outliers.

In Table 1, we summarize the techniques for generating a numerical ranking of outliers based on the model of the outlier detection algorithm, how they create the outlier scores, and the challenges they address.

5.2 Techniques to find categorical rankings of outliers

To the best of our knowledge, the technique proposed in [50] is the first and the only technique providing categorical outlier rankings. We first describe the technique and then discuss how it addresses the challenges explained in Sect. 3.2.

To explain why an object is an outlier in a multidimensional dataset, Knorr and Ng [50] propose to find the attribute spaces in which there is an outlier and label the outlier as strongest, weak, or trivial outlier. Figure 3 shows an illustration of *strongest, weak,* and *trivial* outliers in the 3D space {A, B, C}. P1 and P5 are non-trivial outliers in the subspace AB because they are not outliers in subspace A or subspace B. They are also the strongest outliers in AB because there is no other anomalous point in the subspace A or B. P20 is a weak outlier in the subspace AC because there is another outlier point P11 in the subspace C. P11 is a trivial outlier in the subspace AC because it is also an outlier in the subspace C.

The authors use two distance-based outlier detection algorithms, CELL and NL, reported in [49]. CELL is optimum to find outliers if the number of dimensions is four or less; otherwise, they suggest using NL. To explain the strongest and weak outliers, they use four different techniques: Up Lattice (naïve algorithm), Jump Lattice with Drilldown, Jump Lattice with Path Relationship, and Jump Lattice with Semi-Lattice.

Both Up Lattice and Jump Lattice with Drilldown insert any subset of the attribute set into a queue in ascending order. For example, when the dataset has three attributes (A, B, and C), then the subspaces A, B, C, AB, AC, BC, and ABC will enter the queue in the corresponding order. Using a bottom-up strategy, Up Lattice starts to examine outliers from the minimum cardinality of the attribute subset. However, there is a slight possibility to find outliers in this minimum subdimension space. Jump Lattice with Drilldown is designed to enable jumping to the intermediate subspace. The jumping lessens the amount of effort wasted. If there is no information about the best number of attribute subsets (k) to call the Drilldown procedure, then k is set to 3.

The Up Lattice and Jump Lattice with Drilldown algorithms require examining each node one at a time and take a lot of I/O operations. Therefore, Knorr and Ng design Jump Lattice with Path Relationship to enable grouping multiple

Table 1 Summary of the surveyed techniques that generate a numerical ranking of outliers

Name	References	Outlier detection model	How outlier scores are generated	Challenges addressed
–	Chandola et al. [20]	Neural network	Reconstruction error	None
		Rule-based	Inverse of the confidence associated with the best rule	None
		Clustering	Inverse of the confidence associated with the best rule	None
		Gaussian model	The distance of each instance from the estimated mean	None
		Regression	Residual's magnitude	None
		Density-based	The inverse of the probability density value	None
		Histogram-based	Bin's height where each instance falls into	None
		SVM	Probabilistic prediction value of each instance	None
–	Kriegel et al. [54]	Model-agnostic	Regularization and normalization –based scoring	Challenge 3.1.a
–	Viswanathan et al. [97]	Model-agnostic	False positive rate-based scoring	Challenge 3.1.a
GLADs	Siddiqui et al. [87]	Model-agnostic	User's feedback adaptive scoring	Challenge 3.1.b
COIN	Liu et al. [61]	Model-agnostic	Nearest-normal neighbors –based scoring	Challenge 3.1.a

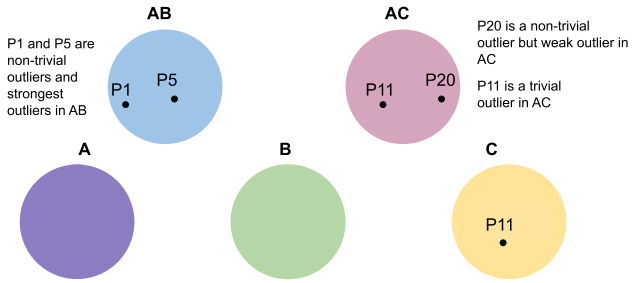

Fig. 3 Illustration of strong, weak, and trivial outliers when examining 3 attributes, A, B, and C

nodes. For example, the nodes {A, B, C}, {B, C}, and {C} are considered having a path relationship, and thus, the algorithm can scan them at the same time. Moreover, Jump Lattice with Semi-Lattice Relationship processes all the subspaces simultaneously. For example, the nodes {A, B, C}, {A, B}, {A, C}, {B, C}, {A}, {B}, and {C} satisfy a Semi-Lattice Relationship, and thus, the algorithm can examine all of these seven spaces at the same time. The experimental evaluation shows that Jump Lattice with Path Relationship and Semi-Lattice have significantly lower runtime than the other approaches.

We now discuss the aspects of the Knorr and Ng technique based on the challenges discussed in Sect. 3.2.

- Challenge (3.2.a): *Defining the levels of importance criteria*
  Knorr and Ng apply subspace-based criteria to determine the levels of importance of outliers and categorize each outlier as "trivial," "weak," or "strongest" one. The criteria suit their illustration of the categorical ranking of outliers for 1995–1996 National Hockey League players' statistics, which is the only dataset evaluated in the paper [50]. Even though they use only one example, this approach can be used in different applications. However, one should note that the levels of importance criteria can be different depending on the application.
- Challenge (3.2.b): *Incorporating user's feedback*
  Knorr and Ng's technique does not address this challenge.
- Challenge (3.2.c): *Selecting an efficient function to map the detected outliers into each category*
  To map an outlier into one of the three categories, Knorr and Ng's technique examines subsets of attributes (subspaces) using the bottom-up approach. Although their Jump Lattice with Path Relationship algorithm can scan multiple subspaces simultaneously, it still requires multiple passes on data. They recommend examining only the subspace of 1, 2, and 3 attributes, yet it still gives $O(n^3)$ subspaces to examine, where n is the total number of attributes.

6 Techniques to find causal interactions among outliers

Two techniques discover causal interactions among outliers: the outlier tree and directed acyclic graph (DAG) approaches. These two techniques are both for outlier detection model-specific outlier explanations, meaning they are only applied for a specific outlier detection algorithm. They are also domain-specific to spatial-temporal traffic data. We describe the outlier tree approach in Sect. 6.1 and the DAG approach in Sect. 6.2. In Sect. 6.3, we discuss the aspects of these techniques based on the challenges explained in Sect. 3.3.

6.1 Outlier tree

Liu et al. [62] propose the STOTree algorithm, depicted in Fig. 4, to find the causal interactions of outliers in spatial-temporal traffic data. They model a city as a directed graph using a Connected Component Labeling method, in which each node in the graph represents a region. There is a link connecting one node to another node if there is at least one moving object from the origin node heading to the destination node. They define an outlier as "a link whose non-spatial and non-temporal attributes are very different from the values of its spatial-temporal neighbor." The non-spatial and non-temporal attributes consist of (1) $\#obj$: the number of moving objects using that link at a given time bin, (2) Pct_o: the ratio of the number of objects going out from the origin node using that link over the total number of objects going out from the origin node, and (3) Pct_d: the ratio of the number of objects coming into the destination node using that link over the total number of objects coming into the destination node.

A time frame is divided into q time intervals (time bins). At every given time bin, each link is related to an attribute vector of three properties $\overrightarrow{f_{ij}} < \#obj, Pcto, Pctd >$. Every combination of $< \#obj, Pct_o, Pct_d >$ is treated as a data point. A data point that represents a link is a *spatial outlier* when its Mahalanobis distance [65] is relatively extreme compared to other data points. Time frames are neighbors when they take place at the same time on consecutive days or at the same days in consecutive weeks. The minimum Euclidean distance between a given time frame and its neighbors' time frames is considered as a non-spatial-temporal attribute called *minDistort*. Links that have the extreme *minDistort* are then considered as *temporal outliers*.

A link detected as a temporal and spatial outlier is called a *spatial-temporal outlier* (STO). Causality trees are constructed to reveal the causal relationships between the STOs. The root of a causality tree contains the first origin node. Each non-leaf child node represents both the destination region of a link taken from the previous time frame and the origin region of some links for the next consecutive time frame. Each level of the tree denotes the next consecutive time frame; therefore, the deeper the causality tree, the further we can see the relationship between an STO and other STOs.

We can discover frequent outlier sub-trees by examining the causality trees. Frequent outlier sub-trees reveal repeated anomalies and give an insight into existing problems in a traffic network. Finding the frequent outlier sub-trees is inspired by the idea of finding frequent item-sets [5,6,78]. The algorithm finds all single nodes from the causality trees whose support surpasses a predefined threshold. The selected single nodes are then used to form candidates of frequent sub-trees. The algorithm increases the frequency of a candidate by one every time it matches with a causality tree.

6.2 Directed acyclic graph (DAG) approach

Xing et al. [100] propose a framework that uses anomalous DAG to find causal interactions among outliers in spatial-temporal traffic data. The framework first builds a region-based traffic network using a grid-based approach [79]. It then computes the spatial-temporal feature of each TOD (a connection from an origin region to a destination region in a time interval of a day). The spatial-temporal feature of TOD is similar to $\overrightarrow{f_{ij}} < \#obj, Pcto, Pctd >$ in the STOTree technique described in Sect. 6.1. The framework then obtains the spatial-temporal density of each TOD based on its k-nearest neighbors. An anomalous TOD (ATOD) is a TOD that has a small density value. The framework inputs the set of ATODs into a depth-first search-based algorithm to build anomalous DAGs. When an ATOD has been visited, the pruning step is executed. This algorithm finds outlier causal relationships by extracting the connected anomalous DAGs using so-called multi-cause analysis.

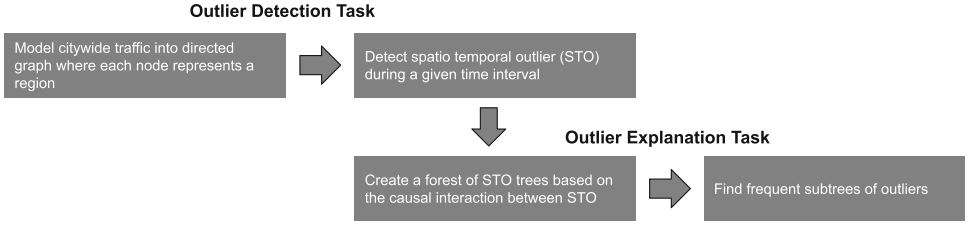

Fig. 4 Steps to find the causal interactions of outliers in spatial temporal traffic data

6.3 Discussion of the surveyed techniques on causal interactions among outliers

We now discuss the aspects of the Outlier Tree technique 6.1 and the DAG technique 6.2 based on the challenges explained in Sect. 3.3.

- Challenge (3.3.a): *Availability of outliers' timestamps*
  Both techniques are specific for spatial-temporal data where timestamps of the outliers are available.
- Challenge (3.3.b): *Defining a data structure to capture the causal interactions among outliers*
  Both techniques rely on so-called spatial-temporal features to define what objects are considered as outliers. Even though these techniques are used for trajectory traffic data, it is possible to use them for other spatial-temporal data as long as we can define similar spatial-temporal features. The Outlier Tree approach (STOTree) captures the causal interactions among outliers using a tree data structure, while the DAG approach builds graphs to obtain the explanations. The time complexity of STOTree is quadratic in the number of outliers in each time frame, while DAG's time complexity is lower than that of STOTree because of its pruning step [100].

7 Techniques to find outlying attributes of outliers

The outlying attributes of outliers can be a *subset of attributes responsible for the abnormality of an individual outlier* or *a group of outliers*. We describe the techniques providing these types of explanations in this section.

7.1 Outlying attributes of an individual outlier

Some of the techniques for finding the feature subspace contributing most to the abnormality of an individual outlier are outlier detection model-agnostic, and some are specific to the outlier detection model. These techniques fall into the following categories: (i) subspace search-based, (ii) local neighborhood-based, (iii) decision tree-based, (iv) layer-wise relevance propagation (LRP), (v) entropy-based reward, and (vi) game theory-based. We describe the techniques in Sects. 7.1.1–7.1.6. The discussion on how the techniques address the challenges explained in Sect. 3.4 is given in Sect. 7.1.7.

7.1.1 Subspace search-based techniques

The techniques in the subspace search-based category find the outlying attributes of each individual outlier by searching subspaces. To avoid examining all the possible subsets of attributes/features, two methodologies are introduced:

- *heuristic subspace search*
  This methodology limits the subspace search using a heuristic process that is not guaranteed to find an optimal subspace yet sufficient enough. It includes a bottom-up beam search that finds the outlying attributes by first examining a single attribute and then proceeding to the higher-dimension subspace.
- *branch-and-bound subspace search*
  This methodology finds the optimal outlying attributes by building a tree whose nodes represent the subset of attributes. In the worst case, it requires exploring all possible permutations of the subspace.

We now describe some techniques for the aforementioned methodologies.

Heuristic Subspace Search Given a multidimensional dataset, an object identified as an outlier in a subspace can be an inlier in another subspace. The first technique proposed to find the subset of attributes contributing the most to the abnormality of an object is the same as the technique that generates the categorical ranking of outliers described in Sect. 5.2. The proposed algorithm works by detecting outliers in the subsets of attributes with lower cardinality and then examining higher cardinality. The process to find the outlying subspace of an outlier follows the heuristic process of finding where the outlier stands out as the strongest outlier. If the algorithm finds an object O as an outlier in the 3D subspace (A, B, C) and O is not an outlier in any of the subspaces (A, B, C, AB, AC, and AB), then O is called the strongest outlier in ABC. Here, we find ABC as the outlying attributes of O. Thus, identifying the strongest outlier also means finding the outlying attributes of the outlier. This technique can be applied only to distance-based outliers.

Refout [47] randomly draws subspaces of dimensionality d_1 without replacement and adds them into the subspace pool P_1 until $\|P_1\|$ reaches a threshold. It then applies an outlier detection model to all the subspaces in P_1. Refout normalizes the outlier scores so that they are comparable among different subspaces. It saves the normalized scores of each object in every subspace. It ranks all objects according to their maximum outlier scores over all the subspaces in P_1. The top m ranked objects are then considered for subspace refinement, where Refout extracts a set of outlier scores from them. The algorithm uses the scores as inputs for a function that obtains a refined subspace for each object. The function performs a bottom-up beam search to find a subspace of length d_2.

The search starts from one-dimensional candidates. In each iteration, it computes the quality of the subspace candidate, i.e., the p-values expressing how well the subspace candidate separates the outlier score population and ranks

it. It keeps a list of all candidates ranked by their qualities but only uses the high-quality candidates to construct higher-dimensional candidates. The search stops when it is not possible to form a higher-dimensional candidate. If the number of features in the final candidate S' is less than d_2, the attribute of the candidates in the list will be added to the final candidate until $\|S'\| = d_2$. S' is then added to the refined subspace pool P_2. Finally, it applies the outlier detection model to all the subspaces in P_2 on all objects in the dataset. It outputs the outlier score for each object in the dataset and the subspace in P_2 where the object has the highest outlier score.

Lookout [39] provides outlying attributes of each outlier as a set of so-called focus plots. A focus plot is a 2-dimensional scatter plot of all data points where the x-axis and y-axis represent a pair of attributes. Lookout needs the complete dataset consisting of inliers and outliers and the set of all possible pairs of attributes P as inputs. The algorithm constructs an isolation forest (iforest) model [60] for each pair of attributes p in P using the complete dataset. It uses the iforest models to recompute the outlier scores of every outlier. Lookout records the outlier scores of each outlier on p. It then runs lazy greedy heuristics [57] to find the set of n focus plots that maximize the total maximum outlier scores of outliers represented through the n plots. The outlying attributes of each individual outlier are the focus plots where it has the highest outlier score.

Branch and Bound Subspace Search Siddiqui et al. [86] study the problem of computing and evaluating sequential feature explanations (SFE) for density-based outliers. An SFE is generated for every outlier by estimating the probability density function $f(x)$ (PDF). Given a set of N data points $x_1, x_2, \ldots, x_N$ in d dimensions, the probability density-based anomaly detector ranks the data points according to their joint PDF values. A data point having the lowest joint PDF value is given the highest outlier score by the detector. Note that PDF-based detectors use a threshold to determine an object as an outlier. The threshold is derived from a function parameterized by a percentile value α. Given a dataset projection on a subset of attributes E, the threshold function $\tau(E, \alpha)$ can generate different threshold values when it applies to different E.

The SFE objective function is defined as the minimum number of attributes that must be disclosed to the analysts so that they can confidently judge a detected point as a true anomaly. Formally, given an SFE E, the smallest prefix SP is defined as the smallest number of attributes in E such that a detected data point x having the PDF value less than a threshold generated by the function $\tau(E, \alpha)$. This definition is written as: $SP(x, E, \alpha) = \min\{k : f(x_{E_k}) < \tau(E, \alpha)\}$. The SFE objective is then defined as finding E that minimizes the expected value of SP denoted by ESP with respect to a prior $p(\alpha)$. Since the true value of the percentile parameter α is unknown, the authors use a discrete distribution over α that assigns $p(\alpha)$ to realistically small values. Mathematically, $ESP(x, E) = \sum_\alpha SP(x, E, \alpha)p(\alpha)$, and the objective of SFE is $\arg\min_E ESP(x, E)$.

The authors propose four greedy-based approaches to find the outlying attributes (the reader can refer to the original paper for these approaches). They also propose an algorithm based on a branch and bound search tree to optimize ESP. This algorithm starts by creating an empty root with d children nodes, representing every attribute of a data point x. It stores those nodes in a priority queue. Each node has an upper bound value computed from the joint PDF of x's projection on the subset of attributes represented by the node. A node in the priority queue is expanded in each iteration when it has the smallest upper bound value. Each child node is added to the priority queue if its lower bound value is smaller than the current smallest upper bound; otherwise, it is pruned. The lower bound is computed as $\sum_\alpha \hat{t}_\alpha p(\alpha)$ where $\hat{t}_\alpha = \min(j : j \leq i, f(x_{E_j}) < \tau_j(E, \alpha) \cup i)$, $f(x_{E_j})$ is the joint PDF value of a data point x in the SFE E of length j, and $\tau_j(E, \alpha)$ is the threshold value of a given percentile value α. The iteration stops when the priority queue is empty, or the number of nodes expanded reaches the maximum threshold.

7.1.2 Local neighborhood-based techniques

Several techniques depend on the local neighborhood of the outlier to derive its outlying attributes. LODI [25] is an algorithm that finds an optimal 1-dimensional subspace to rank and interpret local outliers in multidimensional datasets. For each instance in the dataset, LODI heuristically selects the optimal set of nearest neighbors (referred to as the neighboring set) with the maximum information potential using quadratic entropy. Each instance in the dataset is a vector in a D-dimensional space, where each dimension represents an attribute f_i. After selecting the neighboring set for each instance, LODI trains a binary classifier to find an optimal 1-dimensional subspace w that maximally separates an instance from its neighboring set.

LODI ranks all instances based on the relative difference between the statistical distance of each instance and that of its neighboring set along the direction of its optimal subspace. The algorithm selects the top M instances with the highest ranking as outliers. For each outlier, the projection of its neighboring set over w is the linear combination of its original attributes f_1 to f_D. The absolute coefficients within the eigenvector w correspond to the weights of the original attributes. A user-defined parameter $\lambda(0, 1)$ is used to select the top d absolute coefficients in w for each outlier. Note that $d << D$ and each outlier is associated with a small set of attributes f_1 to f_d. The attributes with large corresponding weights in w are the most important ones to identify outliers. LODI provides explanations for entropy-based outliers.

Similar to LODI, Micenková et al. [68] propose a local neighborhood-based outlier explanation. For a detected outlier x in the dataset, the technique first simulates n artificial instances, oversampled from the normal distribution centered on x. These instances are the outlier class. It then creates an inlier class by subsampling n instances from the inliers in the dataset. It trains a binary classifier to separate the outlier class from the inlier class. Then, it applies a standard attribute selection technique to determine the so-called *explanatory subspace* for the corresponding outlier. The *explanatory subspace* of an outlier is an attribute subset where the outlier has the highest deviation from other points, and at the same time, the dimensionality in the subspace is low. We can use this technique for outliers detected by an arbitrary outlier detection algorithm (outlier detection model-agnostic).

LGOP [24] focuses on identifying and interpreting local outliers using a graph-based model to capture the local geometry. LOGP builds a global graph by finding k nearest neighbors of every object in the dataset. Two objects x_i and x_j have a nonnegative weighted edge if x_i is among the k nearest neighbors of x_j and vice versa. The weight reflects how similar x_i and x_j are. If the points are identical, the weight is close to 1, while if they are very dissimilar, the weight is close to 0.

For each data instance x_i, LGOP extracts from the global graph a neighboring subgraph X^i (a $D \times k$ matrix), which comprises the vertices corresponding to the k nearest neighbors of x_i. LGOP then maps data points in every neighboring subgraph to a lower-dimensional space and, at the same time, preserves the local geometrical structure. It applies a singular value decomposition (SVD) of X^i for the mapping. LGOP uses the matrix resulted from the SVD computation for eigendecomposition. Since eigenvalues and eigenvectors are going in pairs, the first eigenvector corresponding to the largest eigenvalue is used to find the discriminative features by ordering the absolute values of the leading eigenvector decreasingly. The difference in coefficients between relevant and irrelevant features is expected to be larger, at least by a factor of two, than most of the differences between two ordered relevant features.

COIN [61] also finds outlying attributes of each outlier based on its local neighborhood. COIN is agnostic to outlier detection models. Given a dataset X consisting of inliers and outliers identified by an arbitrary outlier detector, COIN first identifies the local context C_i of each outlier o_i. The local context C_i refers to the nearest normal neighbors of o_i based on the L_2 norm. It also expands o_i into a hypothetical outlier class O_i using the synthetic sampling method to balance the outlier data with the inlier C_i data.

COIN segments C_i into different clusters $\{C_{i,1}, C_{i,2}, \ldots, C_{i,L}\}$ using K-means clustering or hierarchical clustering. It adopts prediction strength to determine the number of clusters of C_i. Any cluster $C_{i,l}$ having the number of members less than 3% of the total number of C_i members, is ignored in the training phase, in which outlier and inlier classes are trained using support vector machine (SVM) [23] to find the optimum weights $w_{i,l}$ that maximize the margin between the outlier class and the local context $C_{i,l}$. The length of the vector weight $w_{i,l}$ is equal to the number of attributes of the dataset such that every element in the $w_{i,l}$ corresponds to an attribute a_m. The significance of attribute a_m for each $C_{i,l}$ is described as $s_{i,l}(a_m) = |w_{i,l}[m]|/\gamma^m_{i,l}$, where $|w_{i,l}[m]|$ is the absolute value of the element in vector $w_{i,l}$ at index m and $\gamma^m_{i,l}$ is the average distance along the m axis between an instance in $C_{i,l}$ and its nearest neighbors. The value of $s_{i,l}(a_m)$ is used to compute the significance of attributes a_m for each outlier o_i that is defined as $s_i(a_m) = \frac{\sum_l |C_{i,l}| s_{i,l}(a_m)}{|C_{i,l}|}$. The attributes with the large $s_i(a_m)$ values are considered as the outlying attributes of the outlier o_i.

7.1.3 Decision tree-based techniques

We can obtain outlying attributes of an individual outlier from decision trees. We now describe the techniques in this category.

Kopp et al. [52] introduce Explainer, an algorithm to develop sapling random forests (SRF), an ensemble of specifically trained binary classification trees. SRF explains outliers detected by any arbitrary anomaly detector. The approach splits the dataset into two disjoint sets X^a and X^n, containing anomalous and normal samples accordingly. For each anomaly x^a, the approach selects a training set consisting of the anomaly as one class and a randomly chosen subset of the normal samples as the other class. The size of the training set depends on the user input. Explainer generates multiple training sets for each anomaly by repeating the sampling process for the other class. It uses the training sets to train multiple binary decision trees or saplings to classify each anomaly. The sampling growth stops when the leaf containing x^a is pure or if the height is over a threshold. This threshold is estimated from the average heights of single samplings trained per anomaly.

Saplings/decision trees for each anomaly x^a are used to derive decision rules from the splitting conditions and attribute thresholds at each node on the path from the root to the leaf containing x^a. The following conjunction (c) of these rules is used to explain $x^a : c = (x_{j1} > \Theta_1) \wedge (x_{j2} > \Theta_1) \wedge \cdots \wedge (x_{jd} > \Theta_d)$ where $x_{j1}, \ldots, x_{jd}$ are features and $\Theta_1, \ldots, \Theta_d$ are feature thresholds used in the splitting conditions of the saplings. For anomalies with multiple saplings, the algorithm aggregates conjunctions of these rules by grouping the rules of the attribute set. Thus, unlike other algorithms, SRF produces rules containing the most important and discriminative attributes for detecting outliers. The typical height of the saplings in SRF is 1-3. Thus, the length

of the association rules derived from these saplings is short, leading to compact and intuitive explanations of anomalies. Finally, the outlying attributes are the features included in the conjunction form of the SRF.

EXAD [89] identifies the outlying attributes of the outliers detected using a neural network (LSTM/Autoencoder). EXAD uses deep learning because it addresses feature engineering and anomaly detection in the same mechanism. It approximates the neural network using a decision tree to provide explanations about outliers. An explanation is formed by the paths starting from the tree's root leading to the leaves marked as outliers. The final explanation is presented as a disjunctive normal form (DNF) of atomic predicates of the form $(v\ o\ c)$ where v is a feature, c is a constant, and o is one of the operators $\{<, \leq, =, \geq, >\}$. Here the outlying attributes are the features included in the DNF. If another object in the test set satisfies the condition in the DNF, then the object is labeled an outlier. Thus, EXAD can use the explanation to predict future outliers.

7.1.4 Layer-wise relevance propagation (LRP) techniques

Amarasinghe et al. [10] propose a technique focusing on predicting outliers using a depth neural network (DNN) and provides explanations about features relevant in making the prediction. Before deploying the DNN model to predict outliers in data streams, they build it offline using some training sets. The DNN model is trained for multi-class classification where some of the classes represent the anomalous category.

During the offline training phase, the relevance of each input feature to the DNN prediction on each class is computed using layer-wise relevance propagation (LRP). LRP uses backward propagation to calculate the relevance of each dimension of the data point in every layer. It starts from the output layer (the highest layer), hidden layer, and all the way to the input layer (the lowest layer). In other words, the relevance of the lower layer is computed based on the relevance of the higher layer. The features' relevance scores of each class are calculated by summing all the feature relevance scores of the data points belonging to the class in the input layer and dividing it by the number of the class members.

Keep in mind that LRP is done during the DNN training phase. When the deployed DNN model classifies a data point as an outlier, the outlying attributes of the data point are derived from the relevant features of the anomalous class where the outlier is classified.

7.1.5 Entropy-based reward techniques

To find outlying attributes of each outlier, Exstream [104] applies an entropy-based distance function to measure the reward for every feature and ranks it based on the reward. This algorithm provides explanations for user annotated outliers in complex events monitoring (CEP). Consider a stream producer generating events of multiple types. Each event type follows a schema consisting of a set of attributes, including a time-stamp attribute. CEP receives event streams continuously and monitors the events using user-defined queries.

Users can interact with the CEP monitoring system through the provided visualizations. The visualization usually shows the time stamp on the X-axis and one of the derived event's attributes on the Y-axis. Each visualization represents a partition which is the result of a user-defined query monitored by a Hadoop job. Users can drag and draw rectangles on the visualization to annotate the abnormal interval (I_{A}) and the regular interval (I_{R}) that demonstrate the normal behavior. Given the I_{A} and I_{R}, Exstream generates the relevant attributes/features of the abnormal behavior from the events during the I_{A} and I_{R}.

Exstream assumes that the CEP system archives the raw data streams. Exstream computes the reward for every single feature using a distance-based function that measures the difference between the feature's values in the I_{A} and I_{R}. It then ranks the features based on their rewards. Sharp changes between successive features in the ranking indicate that the features that rank below the sharp drops are not significant for explanations. The insignificant features are filtered out. The algorithm identifies and purges the so-called false-positive features that demonstrate similar behavior in other partitions without abnormal indication. It searches the archived streams to find similar partitions. Such partitions should be the results generated by the same query and monitored by the same Hadoop program on the same dataset. The retrieval is fast because Exstream maintains a record of partitions. Once it discovers the related partitions, it aligns the annotated region I_{A} and I_{R} to each partition on temporal-based or point-based. It selects the alignments in which two partitions have the slightest relative difference. Alignments map the annotation to all related partitions.

Exstream uses hierarchical clustering to label the intervals in the related partitions. If a period is placed in the same cluster as the annotated anomaly, it is labeled an outlier. It then recomputes the entropy-based distance for each feature left and filters out the false-positive ones. It uses pairwise correlation to identify similar features. Each feature is represented as a node. Two nodes are connected when their pairwise correlation exceeds a threshold. Connected nodes are considered as a cluster, and Exstream selects one node from each cluster. Once it makes the final selection, it uses the conjunctive normal form (CNF) to connect the selected features. The value boundaries of the selected features during the abnormal interval (I_{A}) are used as constants in the CNF, for example, $feature_1 > 10 \wedge feature_5 < 5$. The CNF represents the complete explanations. Furthermore, when a future event stream satisfies the condition of the CNF, it can be classified as an anomalous event.

7.1.6 Game theory-based techniques

Takeshi [90] proposes a technique using the Shapley values of the reconstruction errors of PCA to explain outliers detected using PCA. The Shapley value is a method from game theory to distribute the gain of a game to the players. In the context of PCA-based anomaly detection, a gain is an outlier score, which is the reconstruction error of the PCA, and a player is a feature used by the model. In order to compute the contribution of a feature, the algorithm needs the following inputs: the principal component matrix W, the observation noise variance, a data point x, a target feature's index i, and the total number of Monte Carlo iterations.

The algorithm approximates the Shapley value of an individual feature using Monte Carlo approximation. The feature i's Shapley value is initially set to 0. In every iteration, an order of feature indices D is drawn randomly from the set of permutations of $(1, \ldots, d)$ where d is the total number of features. The algorithm computes the Shapley value of the current iteration by subtracting the value function of the set of feature indices that precede i in the order D from the value function of the set of feature indices that precede i in the order D union i. It then divides the Shapley value of the current iteration by the total number of iterations and adds the result into the feature i 's Shapley value. It considers the features whose Shapley values are higher than a threshold as the outlying attributes of x.

7.1.7 Discussion of the surveyed techniques on outlying attributes of an individual outlier

We now discuss the aspects of the techniques presented in Sects. 7.1.1–7.1.7 based on the challenges explained in Sect. 3.4.

- Challenge (3.4.a): *Limiting subspace search*
  Searching the subspace (Sect. 7.1.1) to find the optimum subset of attributes responsible for the abnormality of outliers is an NP-hard problem [86]. Knorr and Ng's technique [50] and Refout [47] apply some heuristics to find relevant subspaces that represent the outlying attributes for an individual outlier; however, they do not guarantee to find the optimum subspace (among all subspaces, in that subspace, the outlier is the most abnormal). On the other hand, Lookout [39] uses a submodular objective to quantify the explanation quality and guarantees finding the optimum 2-dimensional subspace. The algorithm scales linearly with the number of input outliers. SFE [86] adopts four greedy-based approaches and a branch-and-bound approach to find the outlying attributes. In all the approaches, SFE needs multiple scans of the entire dataset in order to compute the joint PDF for a single outlier object. SFE's branch-and-bound approach prunes the search space, and it is aimed to find the optimum subspace. Nevertheless, their experiments show that the greedy approach is preferred because it is less computationally intensive and yields similar results to the branch-and-bound approach in actual evaluations. Other techniques presented in Sects. 7.1.2–7.1.7 do not rely on the subspace search. However, they do deal with some overhead. For example, for every single outlier, COIN [61] requires finding so-called local context and training classifiers based on it in order to find the outlying attributes.
- Challenge (3.4.b): *Generating readily interpretable output*
  Most of the subspace search-based techniques described in Sect. 7.1.1 produce a set of original attributes relevant to the abnormality of each outlier. However, they do not include the attributes' values that characterize the outlier. Lookout [39] provides pictorial explanations. It shows scatterplots of 2-dimension subspace where users can easily tell whether each outlier is isolated from other objects.
  The local-neighborhood-based techniques (Sect. 7.1.2), the layer-wise relevance propagation (LRP) technique (Sect. 7.1.4), and the game theory-based technique (Sect. 7.1.7) generate real number values associated with each original attribute. The values represent the contribution of each attribute to the abnormality of an outlier where the non-contributing attributes are supposed to have zero or minimal values. This kind of output requires users to define a threshold to limit the number of attributes of nonzero weight they want to examine further.
  The decision tree-based techniques (Sect. 7.1.3) and the entropy-based reward technique (Sect. 7.1.5) use the disjunctive normal form (DNF) or conjunctive normal form (CNF) to represent the outlying attributes of an outlier. This kind of output does not require users to specify a threshold; however, users' efforts to verify the result are equivalent to the number of rules in the DNF/CNF.
- Challenge (3.4.c): *Incorporating user's prior knowledge about the attributes*
  COIN [61] (Sect. 7.1.2) is the only technique that addresses this challenge. It allows the user to set a prior knowledge of feature importance by assigning a prior value $p_m \in \{-1, 0, 1\}$ to each attribute. If the user expects an outlier to have a small value on an attribute a_m, then its corresponding $p_m = -1$, $p_m = 1$ means the opposite, while $p_m = 0$ means no expectation. The value of each p_m is then incorporated into the computation of the attribute's contribution score.

In Table 2, we summarize the outlier explanation techniques that find the subset of attributes responsible for the abnormality of an individual outlier based on the following

Table 2 Summary of techniques that find the outlying attributes of an individual outlier

Name	Reference	Outlier detection model	Explanation methodology	Output	Data type supported	Future outlier prediction	Challenges addressed
–	Knorr and Ng [50]	Distance-based	Heuristic-based subspace search	Sets of outlying attributes	Numerical	No	Challenge 3.4.a & b
LODI	Dang et al. [25]	Model agnostic	Local neighborhood	Contribution score for each attribute	Numerical	No	Challenge 3.4.b
–	Micenková et al. [68]	Model agnostic	Local neighborhood	Contribution score for each attribute	Numerical	No	Challenge 3.4.b
Refout	Keller et al. [47]	Model agnostic	Heuristic-based subspace search	Sets of outlying attributes	Any	No	Challenge 3.4.a & b
Explainer	Kopp et al. [52]	Model agnostic	Decision Tree	DNF	Any	No	Challenge 3.4.b
LOGP	Dang et al. [24]	Graph	Local neighborhood	Contribution score for each attribute	Numerical	No	Challenge 3.4.b
Exstream	Zhang and He [104]	User-annotated CEP Monitoring	Entropy-based reward	CNF	Numerical	Yes	Challenge 3.4.b
COIN	Liu et al. [61]	Model agnostic	Local neighborhood	Contribution score for each attribute	Numerical	No	Challenge 3.4.b & c
EXAD	Song et al. [89]	Neural network	Decision Tree	DNF	Numerical	No	Challenge 3.4.b
–	Amarasinghe et al. [10]	Neural network	Layer-wise relevance propagation	Contribution score for each attribute	Numerical	Yes	Challenge 3.4.b
Lookout	Gupta et al. [39]	Model agnostic	Heuristic-based subspace search	Set of focus plots	Numerical	No	Challenge 3.4.a & b
SFE	Siddiqui et al. [86]	Density-based	Branch-and-bound search	Sets of outlying attributes	Numerical	No	Challenge 3.4.a & b
–	Takeishi [90]	PCA	Shapley values	Shapley values of all attributes	Numerical	No	Challenge 3.4.b

aspects: (i) the outlier detection model supported, (ii) the main methodology used, (iii) the type of output generated, (iv) the data type supported, (v) whether the explanations can be used to predict future outliers, and (vi) the challenges addressed.

7.2 Techniques to find outlying attributes of a group of outliers

Finding the outlying attributes of a group of outliers is basically finding the similarity among outliers. We survey the techniques proposed for this type of outlier explanation in this section. The main methodologies used by these techniques to generate the explanations can be summarized as: (i) *Apriori-based*, (ii) *Graph-based*, (iii) *Attribute reduction-based*, and (iv) *Subspace clustering-based*. They are described in Sects. 7.2.1–7.2.4. The discussion on how the techniques address the challenges explained in Sect. 3.5 is given in Sect. 7.2.5.

7.2.1 Apriori-based

Chen et al. [21] propose an algorithm to find outlying attributes in distance-based outliers. They first identify outliers in multidimensional spaces using the distance-based outlier detection algorithm proposed in [49]. Then, they group the detected outliers based on the times they were observed (G_1 to G_n for n observed times. Each group G_i has a different number of outliers). For each group of outliers (G_i), the algorithm generates a knowledge set using the Mine-Knowledge Set algorithm, which resembles the apriori association rule mining algorithm [6]. A knowledge set is a minimum subspace in which an object is detected as a non-trivial outlier (an outlier is non-trivial in an attribute set A_p if it is not an outlier in a proper subset of A_p [50]). Before generating knowledge sets, the algorithm assumes that the users have categorized data attributes into three types: identification (I), indication (C), and observation (O). Identification attributes are the attributes that together uniquely identify each object in the dataset. Indication attributes are the attributes whose values are used to calculate the category and behavior rate. Observation attributes are the attributes used to group objects based on the time the objects were observed.

The construction of knowledge sets is based on indication attributes. Each outlier found in an observation can have several knowledge sets. Any pair of outlier groups $\{G_i, G_j\}$ is considered categorically similar if the number of outliers in G_i sharing the same knowledge sets with the outliers in G_j exceeds some predefined threshold. Thus, categorical similarity indicates a high percentage of common categories of outliers between G_i and G_j. Any pair of outlier groups $\{G_i, G_j\}$ is considered behaviorally similar if the number of categorically similar outliers in G_i and G_j lying very close to each other exceeds some predefined threshold. Any pair of outlier groups $\{G_{i-1}, G_i\}$ are similar if they are both categorically and behaviorally similar. Thus, n groups of outliers $(G_1, G_2, \ldots, G_n)$ are similar if every two consecutive groups are similar. The groups that are categorically and behaviorally similar are considered as one big group. The outlying attributes of this big group are the knowledge sets that make the members categorically similar.

7.2.2 Graph-based

Silveira and Diot [88] suggest URCA that narrows down the flows that trigger an outlier detection algorithm. URCA flags alarms in a given time bin to find the root causes of the anomalous bin. A flow is a set of packets that share the same attributes. The attributes of a flow consist of source IP, source port, source AS, previous AS, input router interfaces, output router interfaces, next AS, destination AS, destination IP, and destination port. This method consists of the following steps: (i) identify flows that are primarily responsible for the anomalies in a time bin; (ii) build undirected graphs that characterize the isolated flows; and (iii) classify the graphs from Step ii using hierarchical clustering on labeled root cause events.

URCA uses an algorithm called partition-reduce to group alarm-triggering flows based on their attribute values. It sets the grouped flows as inputs to a function that generates scores for each group. It then sorts those groups of flows in ascending order based on their scores. If the value of the first minimum score is less than a threshold t and the value of the second minimum score is greater than a threshold T, then the algorithm will return the first group as a subset of flows, which has contributed to an anomaly; otherwise, it will return the whole groups of flows. T is a threshold to flag a bin as an anomalous bin, while t is a threshold to evaluate how low the anomaly scores should be for a group of flows to be considered normal.

URCA repeatedly executes the partition-reduce algorithm for each attribute of flows. It uses the isolated flows generated by the partition-reduce algorithm to build graphs. Each node in the graphs represents an attribute flow and its value. The edges on the graphs describe how the flows go from the source hosts to the destination hosts. Graphs with a similar structure tend to be caused by the same event. The root cause events could be: (i) unknown, (ii) large file transfers, (iii) routing changes, (iv) port scans, (v) network scans, (vii) gap/failures, and (viii) DoS attacks. The root cause events are labeled based on 22 coordinates: the average flow size in packets, the average packets' size in bytes, the entropy of the distribution of packets per attribute value of the flows, and the percentage of attribute values in the entire link traffic that also appear in the root cause traffic. Using the assumption that

similar types of events would produce similar graph structures, the authors use hierarchical clustering [45] to build a taxonomy tree where similar anomalies are nested in subtrees, and each leaf node corresponds to an anomaly. When there is an unknown anomaly, the algorithm will find subtrees that contain the siblings of the anomaly based on its coordinates. It considers the outliers nested under the same subtree as a group. Since each node in the graph represents an attribute, one can traverse each node in a graph corresponding to a group of outliers to get the outlying attributes of the group.

7.2.3 Attribute reduction-based

Huang and Yang [44] propose a technique to find sets of attributes having most of the detected outliers. The proposed method consists of (i) identifying outliers in the total attribute space (the outlier detection task), (ii) finding the Knowledge Attribute Subspace (KAS), which is the set of attributes where an identified object has abnormal values, and (iii) detecting the key KAS (KKAS), which is the sets of attributes having a high percentage of outliers. The algorithm to find KAS is similar to the method to find the knowledge set proposed in [21], which we described in the apriori-based outlying attributes technique in Sect. 7.2.1.

In order to find KKAS, the algorithm computes the outlying partition similarity, which is the sum of the weighted percentages of the cardinality of outliers found in a KAS compared to the cardinality of outliers found in the full attribute space. The *outlying partition similarity* for each KAS is calculated using the following equation:

$$\text{outlying partition similarity} = w_1 f_{sup} + w_2 f_{con} + w_3 f_{inc}$$

where $f_{sup} = \frac{card(XO_s \cap XO_d)}{card(XO_s \cup XO_d)}$, $f_{con} = \frac{card(XO_s \cap XO_d)}{card(XO_s}$, $f_{sup} = \frac{card(XO_s \cap XO_d)}{card(XO_d)}$, w_1, w_2, w_3 are weights defined by the user, $card(XO_d)$ is the cardinality of outliers detected in the full attribute space, $card(XO_s)$ is the cardinality of outliers detected in a given KAS, $card(XO_s \cup XO_d)$ is the cardinality of outliers found both in a given KAS and in the full attribute space, and $card(XO_s \cap XO_d)$ is the cardinality of outliers found in a given KAS or in the full attribute space.

The concept of maximum outlying partition is based on the fact that if an object is an outlier in the entire attribute space, its projection in a subspace is not always an outlier and vice versa. The algorithm determines the Key Knowledge Attribute Subspace (KKAS) by finding the maximum outlying partition similarity among KAS. KKAS is the set of attributes that contains most of the outliers. In other words, KKAS is the outlying attributes for the group representing most of the detected outliers.

7.2.4 Subspace clustering-based

We now describe XPACS [64], whose goal is to cluster outliers in an arbitrary subspace. By finding the clusters, it also finds the outlying attributes for each cluster. XPACS finds the groups/clusters of outliers following three steps. Step 1 is to apply subspace clustering to group outliers into a set of hyper-rectangles. In an attribute subset of length k, a hyper-rectangle R has k sides where each side represents a value interval/range of an attribute. The subspace clustering starts by initializing candidates of hyper-rectangles R in 1-dimension using kernel density estimation (KDE). Outliers can be concentrated in more than one interval in each attribute, meaning that there could be more than one candidate hyper-rectangle in every attribute. In order to find the multiple intervals in an individual attribute, the subspace clustering-based algorithm varies the quantile thresholds.

The algorithm progresses level by level. If a candidate hyper-rectangle in k-dimension meets the mass threshold, which is the minimum number of outliers required to form a cluster, it will be examined further; otherwise, it will be discarded. XPACS then checks the candidate hyper-rectangle R to see whether the number of normal points included is less than the maximum number of inliers allowed. If yes, it includes R in the set of pure hyper-rectangles in k-dimension denoted as $\mathscr{R}^k_{pure}$; otherwise, XPACS includes R in a non-pure set denoted as $\mathscr{R}^k_{\neg pure}$. The algorithm joins all candidate hyper-rectangles in $\mathscr{R}^k_{pure}$ into the set of hyper-rectangles denoted by $\mathscr{R}$. It generates the next candidates of hyper-rectangles in k+*1*-dimension by joining $\mathscr{R}^k_{pure}$ and $\mathscr{R}^k_{\neg pure}$. Two hyper-rectangles in k-dimension (R^k_a and R^k_b) can form a candidate of hyper-rectangle in k+1-dimension if they both have the same interval (side) in k-*1* attributes.

Step 2 of the algorithm refines the set of rectangles found in Step 1 into a set of axis-aligned hyper-ellipsoid called pack. A pack is defined as a set of data points (mostly outliers) whose squared distance from the center is equal to or less than 1. Formally, a pack formed in an attribute subset of length k is defined as $p_k = \{x \| (x - c_k) M_k^{-1} (x - c_k)\}$, where c_k represents the center of the hyper-ellipsoid, and M_k is a diagonal matrix of $k \times k$ size. When a hyper-rectangle R is transformed into a hyper-ellipsoid (pack), the goals are to: (i) keep the outliers that are already in R denoted as x_i, (ii) exclude normal data points denoted as x_l, and (iii) include more outliers that are not in R denoted as x_j. A pack found in an attribute subset of length k is represented by a conjunction of k attribute rules where each rule is an attribute value interval $(c^z - \text{radius}_z, c^z + \text{radius}_z)$. They can be used to predict future outliers.

The same outliers can be covered by multiple packs generated in Step 2. Hence in Step 3, the goal is to find packs that describe outliers in the dataset as briefly as possible while

avoiding packs that cover largely overlapping groups of outliers or contain too many inliers. Step 3 uses the random greedy algorithm by Buchbinder et al. [19] to select a subset of packs that yields the fewest bits. Recall that each pack is a group of outliers; therefore, the outlying attributes of the pack are the attributes used in the pack's conjunction representation.

7.2.5 Discussion of the surveyed techniques on outlying attributes of a group of outliers

We now discuss the aspects of the techniques presented in Sects. 7.2.1–7.2.4 based on the challenges explained in Sect. 3.5.

- Challenge (3.5.a): *Limiting subspace search*
  XPACS [64] (Sect. 7.2.4) is the only technique that employs subspace search to group outliers. It limits the search by pruning some subspaces that do not meet the mass threshold.
- Challenge (3.5.b): *Generating readily interpretable output*
  XPACS [64] (Sect. 7.2.4) uses the conjunctive normal form (CNF) to represent each group of outliers and uses a greedy approach to determine the shortest CNF. CNF is considered as an interpretable machine learning output [38]. It is easier to understand by users than the graph form by URCA [88] (Sect. 7.2.2)
- Challenge (3.5.c): *Incorporating user's prior knowledge about the attributes*
  Only Chen et al.'s technique [21] (Sect. 7.2.1) requires users to define three categories of dataset's attributes: the attributes used to identify an object from other objects, the attributes used to indicate the normal/abnormal behavior, and the attributes used to observe the behavior. This technique incorporates the user's prior knowledge; however, it only provides a general setting to find outliers' groups. It does not include prior knowledge about a particular group.
- Challenge (3.5.d): *Finding discriminative outlying attributes*
  In all the techniques described in Sects. 7.2.1-7.2.4, a subset of attributes can cover multiple groups of outliers. Chen et al.'s technique [21] (Sect. 7.2.1), URCA [88] (Sect. 7.2.2), and XPACS [64] (Sect. 7.2.4) distinguish the groups based on their corresponding attributes' values; however, only URCA and XPACS use the obtained groups' information to predict future outliers. Furthermore, XPACS ensures that no normal instances fit within the packs (outlier groups).

We summarize the outlier explanation techniques that find the outlying attributes of a group of outliers in Table 3. The

Table 3 Summary of techniques that find the outlying attributes of a group of outliers

Name	Reference	Outlier detection model	Explanation methodology	Output	Data type supported	Future outlier prediction	Challenges addressed
–	Chen et al. [21]	Distance-based	Apriori-based	Set of outlying attributes and their values	Any	No	Challenge 3.5.c, 3.5.d
URCA	Silveira and Diot [88]	Model agnostic	Graph-based	Graph of attribute flows and their values	Any	Yes	Challenge 3.5.d
KKAS	Huang and Yang [44]	Distance-based	Attribute Reduction-based	Set of outlying attributes	Numerical	No	Challenge 3.5.d
XPACS	Macha and Akoglu [64]	Model agnostic	Subspace Clustering-based	Conjunction of attributes' value intervals	Numerical	Yes	Challenge 3.5.a, 3.5.b, 3.5.d

summary is based on the following aspects: (i) the outlier detection model supported, (ii) the main methodology used, (iii) the type of output generated, (iv) the data type supported, (v) whether the explanations can be used to predict future outliers, and (vi) the challenges addressed.

8 Summary of properties of the surveyed outlier explanation techniques

We summarize the properties of the surveyed outlier explanation techniques in Table 4. The checkmark symbol (✓) indicates whether a technique satisfies the criteria listed in the corresponding columns. The columns in this table represent the following:

1. the name of the proposed techniques (if it is available);
2. the referenced papers;
3. whether the technique is outlier detection model-agnostic, meaning it can explain outliers detected by any outlier detection model;
4. whether the technique is for domain-specific applications;
5. the types of outlier explanations produced: (i) a numerical ranking of outliers, (ii) a categorical ranking of outliers, (iii) causal interactions among outliers, (iv) outlying attributes of an individual outlier, and (v) outlying attributes of a group of outliers;
6. whether the technique includes some prior knowledge about the data set;
7. whether the technique incorporates feedback from users;
8. whether the technique can be applied in stream environments;
9. whether the technique exploits parallel processing;
10. whether the technique applies to distributed systems;
11. whether the source code is shared publicly for reproducibility.

At this point, readers are familiar with Items 1–5 in the list above. However, before we review the content of Table 4, we will first discuss why we include Items 6–10. One of the challenges in finding the outlying attributes of outliers in Sect. 3.5 is incorporating the user's prior knowledge about the attributes. We notice that some techniques for outlier explanation types allow users to include some prior knowledge about the datasets (Item 6). In addition, some techniques can improve the quality of their outputs by taking into account feedback from users (Item 7).

We include whether the technique is applied in stream environments (Item 8) because many emerging applications generate infinite sequences of data called data streams. Researchers and practitioners have applied outlier detection algorithms in data stream applications such as network traffic monitoring [13], financial transactions [7], environmental sensors [76,103], and Internet of Things (IoT) devices [22].

Processing data streams is different from processing batch or static data due to the characteristics of data streams. Sadik & Gruenwald [82] discuss the following general characteristics of data streams: (i) *transiency*: the importance of a data point is decaying over time, (ii) *notion of time*: each data point has a temporal context, (iii) *notion of infinity*: number of data points is infinite such that only a small percentage of data can be stored in memory, (iv) *arrival rate*: data points are continuously coming with a fixed or dynamic high arrival rate such that real-time processing is needed, (v) *concept drift*: the underlying distribution of incoming data can change over time, (vi) *uncertainty*: the unreliability of the data points (missing or out-of-date data points, the attribute values cannot be measured with sufficient confidence, the lack of complete semantics, etc.), and (vii) *multi-dimensionality*. In addition, they also discuss the following characteristics of multiple data streams: (viii) *cross-correlation*: data from multiple streams can be cross-correlated even though their values are not close to one another, (ix) *asynchronous data points*: data sources can be independent of one another, and thus, they can generate data points with different arrival rates, (x) *dynamic relationship* among data points from multiple streams because of concept drift, and (xi) *heterogeneous schema*: streams from multiple sources can have a different schema. We need to take these characteristics into consideration when designing outlier explanation algorithms for data streams.

Parallel processing is crucial as we deal with Big Data [11,93]. Big data is characterized by a large volume, remarkable velocity, variety of data formats gathered from multiple sources, and uncertainty. We can accelerate outlier explanations on Big Data using parallelism, achieved by running the outlier explanation task on numerous computing cores [30,53]. Thus, we include parallelism in Item 9.

As many applications deal with ever-growing data, distributed systems provide a solution to handle increasing application demands [11]. The distributed computing paradigm allows applications to get large-scale computation by combining many computer units via fast, reliable networks in a more affordable way than a supercomputer. It also allows resource sharing among users in a transparent manner. Some algorithms have been proposed to detect outliers in distributed systems [9,35,41]; hence, including distributed systems as a property of outlier explanation techniques (Item 10) is necessary.

Now, we summarize Table 4. We can see that some of the authors do not name their proposed techniques, so we mark their names as '_'. Some techniques can only be applied using specific outlier detection algorithms (outlier detection model-specific). Some can generate explanations for outliers detected by any outlier detector (outlier detection model-

Table 4 Summary of the properties of the surveyed outlier explanation techniques

(1) Name	(2) Reference	(3) Model agnostic	(4) Domain-specific application	(5) Explanations provided					(6) Including prior knowledge of the dataset	(7) Incorporating users' feedback	(8) Data stream	(9) Parallelism	(10) Distributed system	(11) Public availability of source code
				Numerical ranking	Categorical ranking	Causal interaction among outliers	Outlying attributes of an individual outlier	Outlying attributes of a group of outliers						
–	Knorr and Ng [50]				√		√							
–	Chen et al. [21]							√	√					
URCA	Silveira and Diot [88]	√	√					√						
KKAS	Huang and Yang [44]							√						
STOTree	Liu et al. [62]		√			√								
–	Kriegel et al. [54]	√		√										
–	Viswanathan et al. [97]	√	√	√							√		√	
LODI	Dang et al. [25]	√					√							
–	Micenková et al. [68]	√					√							
Refout	Keller et al. [47]	√					√							
Explainer	Kopp et al. [52]	√					√							
LOGP	Dang et al. [24]						√							
–	Xing et al. [100]		√			√								
Exstream	Zhang and He [104]		√				√				√		√	
COIN	Liu et al. [61]	√		√			√		√					√
EXAD	Song et al. [89]		√				√				√		√	
–	Amarasinghe et al. [10]						√							
XPACS	Macha and Akoglu [64]	√						√				√		√
GLAD	Siddiqui et al. [87]	√		√						√			√	√
Lookout	Gupta et al. [39]	√					√					√		√
SFE	Siddiqui et al. [86]						√							
–	Takeishi [90]						√							

agnostic). The majority of the techniques are for general applications, and only a few are for domain-specific applications: spatial-temporal traffic, data center, and network monitoring. The majority of the techniques provide outlying attributes for an individual outlier. Furthermore, most of the surveyed techniques generate one type of outlier explanation. Only one technique [61] provides both the numerical ranking of outliers and outlying attributes of each outlier. Another technique [50] provides a categorical ranking of outliers and outlying attributes of each outlier. Two techniques [21,61] include prior knowledge of the dataset. Only one technique [87] incorporates feedback from users to generate outlier explanations. Three techniques [89,97,104] are for data streams. Two techniques [39,64] can be applied in parallel. Four techniques [87,89,97,104] are applicable in distributed systems. Finally, there are only four of these surveyed techniques [39,61,64,87], the source codes of which are shared publicly.

9 Methods to evaluate outlier explanations

In this section, we describe how the surveyed outlier explanation techniques address the evaluative aspects of explanations (Sect. 9.1). Then, as the ground truth of outlier explanations are needed in order to evaluate them, we survey methods that are proposed to generate the ground truth (Sect. 9.2).

9.1 Evaluative aspects of outlier explanations

Evaluating outlier explanations is a non-trivial task. In Sect. 2, we discussed that we could view four aspects of human-friendly explanations [71] as evaluative aspects. Those aspects are used to assess the performance of the explanations generated by the outlier explanation algorithms. As discussed in Sect. 2, there are four evaluative aspects. First, Aspect (4), *explanations are social*, implies that the explanations should meet the needs of the users investigating the outliers. Next, Aspect (5), *explanations are truthful*, infers that the explanations should reflect high accuracy. Then, Aspect (6), *good explanations are consistent with prior beliefs of the explainee*, suggests the explanations should be coherent with the user's prior opinion about the detected outliers. Finally, Aspect (7), *good explanations are general and probable*, relates to the percentage of the current and future outliers that the task can explain. However, these aspects are abstract metrics; one needs to quantify them when evaluating the results of the outlier explanation task. We describe how the surveyed techniques address these aspects in the following.

When evaluating the *numerical ranking of outliers*, Kriegel et al. [54], Viswanathan et al. [97], Liu et al. [61], and Siddiqui et al. [87] focus only on Aspect (5). They use ROC AUC, accuracy rate, AUC, and the number of true anomalies placed in the top ranking as metrics, respectively. Next, Knorr and Ng's technique [50], which is the only surveyed technique discussing the *categorical ranking of outliers*, does not evaluate any of the aspects. Instead, they focus on the I/O performance, such as the minimum number of I/Os saved and the maximum number of times a data page has to be read when computing the categories of outliers.

Liu et al. [62] and Xing et al. [100] evaluate *the causal interactions among outliers* based on Aspect (5). They use real-world traffic data, and they conclude that the outlier causal relationship discovered by their algorithms coincides with the causal events in real life. However, they do not report the exact numbers of the matched causal events.

When evaluating *the outlying attributes of an individual outlier,* Dang et al. [25] conclude that the interpretation form over outliers is intuitive and meaningful based on the feature visualization over the top 5 outliers found. However, this claim is not quantified. In [24], Dang et al. evaluate Aspect (5) by showing whether the outlying attributes are correctly identified using image data sets where pixels/locations of anomalous images are recognized by human eyes. Furthermore, they use visualization to compare the outlying attributes with the actual ones. Micenková et al. [68] use Jaccard Index and Precision to assess the accuracy rate of the explanation task's outputs (Aspect (5)). Keller et al. [47] use the outlying attributes to improve detection. Hence, the quality of the outlying attributes affects the quality of the outlier detection measured using the AUC metric (Aspect (5)). Kopp et al. [52] measure Aspect (5) using accuracy and average silhouette. Zhang and He [104] measure the consistency between ground truth and the generated outlying attributes using F-measure (Aspect (5)). They also use F-measure to quantify the predictive power (Aspect (7)). Liu et al. [61] use image datasets and real-world datasets with ground truth to assess Aspect (5) using ROC AUC. Amarasinghe et al. [10] conduct experiments on one data set (NSL-KDD) and conclude that the generated outlying attributes are consistent with the commonly known factors of DoS attacks. This evaluation is based on Aspect (5); however, it is very limited. Gupta et al. [39] deal with Aspect (5) by using the so-called incrimination score. Siddiqui et al. [86] evaluate Aspect (5) using the minimum number of attributes required by analysts to accept that a detected outlier is a true positive.

When assessing *the outlying attributes of a group of outliers*, Chen et al. [21] and Huang & Yang [44] focus only on the execution time. Silveira and Diot [88] apply manually labeled anomalies and anomaly injection to evaluate accuracy (Aspect (5) of their classification algorithm, which includes discovering the outlying attributes. Macha and Akoglu [64] evaluate Aspect (5) using image datasets and digit data set to compare XPACS' outlying attributes with what human eyes can recognize as abnormal pixels. They

also measure the interpretability of XPACS's outputs based on the number of groups generated, the average length of the groups' descriptions (rules), the average fraction of normal points included in the groups' descriptions, the average interval width of the groups' descriptions, and the detection performance (AUPRC) which is related to Aspect (7).

In summary, most of the surveyed techniques address Aspect (5), *explanations are truthful*. None of the surveyed techniques addresses Aspect (4), *explanations are social*, and Aspect (6), *good explanations are consistent with prior beliefs of the explainee*. Only two techniques address Aspect (7), *good explanations are general and probable*.

9.2 Methods to generate outlier explanation ground truth

Ideally, when evaluating outlier explanations, we need a benchmark with ground truth to verify that the explanations provided are correct and can be used to understand the detected outliers. However, often the ground truth about the knowledge of outliers (especially outlying attributes) is neither available in real-world datasets nor publicly available. Thus, we need to make some adjustments to verify the results. From the studies we have surveyed, three primary methods are used to generate the ground truth: (i) modifying real-world datasets, (ii) generating synthetic datasets from scratch, and (iii) simulating analysts. We describe these three methods in Sects. 9.2.1–9.2.3 and discuss their advantages and disadvantages in Sect. 9.2.4.

9.2.1 Modifying real-world datasets

Liu et al. [61] state that the ground truth of outlying attributes in real-world datasets is not available. To verify that the attributes identified by their algorithm, COIN, are indeed outlying, they use a real-world dataset, append M noise attributes to all the data instances in the dataset and assume that all the original attributes are outlying attributes and noise attributes are not. COIN chooses the local context of an outlier from 8% of the outlier's nearest neighbors.

To avoid overlap between the outlier class and the inlier classes, the authors set the radius of the synthetic sampling for building the outlier class as half of the distance from the outlier point to the inlier classes. They determine the hyperparameters in the SVM model through validation by randomly selecting some samples from the outlier class and inlier classes for the validation. They report the precision, recall, and F1-score for accuracy. They incorporate prior knowledge of the practical meaning of attributes and evaluate their proposed method with two sets of experiments. Experiment 1 assumes that all the original attributes are equally relevant to the problem of interest while appending some simulated attributes to each instance. Experiment 2 presumes that the original attributes are not equally relevant to the problem of interest. Liu et al. set the ground truth score as 1 for each true outlier and 0 for each inlier to evaluate the outlier ranking. They randomly generate a sample of an equal number of inliers and outliers and feed them into the explainers (COIN and baseline methods). For each query instance, the explainers estimate the outlier score and rank the instances in descending order based on their scores. They use the area under the curve (AUC) as the evaluation metric for the ranking.

Micenková et al. [68] adjust a real-world dataset that is not primarily an outlier detection dataset by selecting all data of one class to be inliers and then one instance from each remaining class to be an outlier. The data do not have the ground truth of the outlying attributes. They run the Local Outlier Factor (LOF) [18] algorithm exhaustively on all possible subspace projections of the data to verify that their method indeed finds the explanatory subspaces. They normalize the outlier scores generated by LOF to overcome the dimensionality bias. Then, they create a ranking of subspaces for each outlier and select the top subspace as a reference to compare with the results of their proposed technique. They also use a KDD Cup 99 dataset [94] to validate that their proposed method finds meaningful explanations. Since there are too many anomalies in the dataset, they first remove identical attacks and then choose 3500 anomalies such that all attack types are covered. Then, they derive the explanations for those anomalies using LARS-lasso [91]. They group the attacks by their types and compare their derived explanations. Assuming that the same attributes can often explain the attacks of the same kind, they build a 2D histogram of the occurrences of each attribute in the explanations of attacks of a specific type. The rows in the histogram represent the dataset's attributes, and the columns represent the types of attacks. They normalize the values of each column by the number of attacks of that type. Thus, we can see whether all attacks of a specific type (column) have a particular attribute (row) in their explanation from the histogram.

Dang et al. [25] evaluate their proposed method LODI using the datasets: Ionosphere, Image segmentation, and Vowel from the UCI Machine Learning Repository [94]. Instead of adding artificial outliers to the dataset, they reduce the number of points in several dataset classes and treat them as hidden outliers. They provide feature visualization of the top-5 ranked outliers returned by the LODI algorithm from every dataset. The x-axis of the plot represents the index of the attributes, while the y-axis represents their degree of importance. They compare LODI with SOD [55] which is not designed directly for outlier explanations. SOD uncovers the axis-parallel subspaces that can be used to select outliers' relevant attributes.

9.2.2 Generating synthetic datasets from scratch

Chen et al. [21] do not use a real-world dataset but instead conduct an empirical analysis on anomalous patterns using synthetic data. They determine the size of a knowledge set using a Poisson distribution and randomly assign attributes to the knowledge set. Recall that the knowledge set is the minimum subset of attributes in which an object is detected as a non-trivial outlier. They generate the data points for the features that are not in the knowledge sets using a uniform distribution. Furthermore, they create a small fraction of the data points within the subspaces consisting of the potential knowledge sets to smoothen the overall distribution of the data points.

Dang et al. [25] generate three synthetic datasets; each consists of 50K points generated from multivariate normal distributions. They vary the number of attributes from 15–50 in each dataset. They randomly select the mean of each attribute from {10, 20, 30, 40, 50} and set the variance to be either 10 or 100. They vary 1%, 2%, 5%, and 10% of the whole data as the number of randomly generated outliers within the range of the data space. They also set the percentages of the large variance to be 40%, 50%, and 60%. The intuition is that when the dataset has more attributes with higher variance, the dimensionality of the subspaces in which an outlier can be identified will be narrowed down due to the wide overlapping of outliers and inliers projected on the attributes with large variance. Conversely, when more attributes have smaller variance, the number of relevant attributes used to explain each outlier increases.

9.2.3 Simulating analysts

Most of the surveyed techniques do not involve expert users in evaluating their proposed strategies. However, Siddiqui et al. [87] evaluate the outlier ranking produced by GLAD, using six benchmark datasets based on audit logs generated during the attack campaigns carried out by a red team as part of the DARPA Transparent Computing program [26]. In their experiments, the outlier ranking was reported to the red team that later released a description of each attack scenario, which outlined the key entities and events. The descriptions tell whether each entity and event are part of the attack or not. The authors use the descriptions to produce a ground truth encoding of the attacks. They then use the ground truth to evaluate their algorithm and simulate feedback provided by a system analyst.

Siddiqui et al. [86] construct a simulated analyst to evaluate their proposed technique, SFE. They model the simulated analyst as a conditional distribution of the inlier class given a data point and a subset of attributes. The analyst, which is a classifier, is formulated as a function $A(x, S) = P(\text{normal}|x_S)$ where x is a data point, S is a subspace, and x_S is the value of x on S. It returns the probability of point x being normal considering only the attributes specified by the subset S. When the classifier identifies the data points correctly given the attribute ordering (the attribute subset based on SFE), it means the SFE is correct. The authors use Regularized Random Forests (RRFs) [27] as the classifier for the analyst model. They first construct a training set of inliers and outlier classes over the points from a classification or regression dataset. Then, they train one RRF for each possible subset of attributes (up to some maximum size) and use tenfold cross-validation to tune the RRF regularization parameters.

9.2.4 Discussion on the ground truth generation methods

We now discuss the advantages and disadvantages of the ground truth generation methods described in Sects. 9.2.1–9.2.3.

Modifying real-world datasets This approach allows us to evaluate outlier explanation techniques on a real-world dataset without ground truth. However, adding noise attributes and treating all the original attributes in the dataset as outlying ones [61] has a pitfall. What if some of the original attributes are not relevant to the outliers' abnormality? If the outlier explanation algorithms do not include those irrelevant original attributes in their outputs, the presumed ground truth will lower the reported accuracy of the algorithms.

Modifying classification datasets and then running an outlier detection algorithm in all possible subspaces to determine outlying ground truth are possible when the total number of attributes is small. However, when dealing with high-dimensional data, the computation is time-consuming. It requires running the outlier detection for $\sum_{k=1}^{d} C(d,k) = \sum_{k=1}^{d} \frac{d!}{k!(d-k)!}$ times, where d is the total number of attributes and k is the cardinality of the attribute subset. Suppose we have a dataset of 10 or 15 attributes, then we need to run the outlier detector 1,023 or 32,767 times, respectively, on the dataset to generate the ground truth. We can speed up the computation if we run the detector on every subspace in parallel. Also, using this approach, we rely on the quality of the outlier detection algorithm [72].

Generating synthetic datasets from scratch When applying this approach, we do not need to worry about the limitation of modifying real-world datasets. We can specify which attribute subset has a high percentage of abnormal data points; therefore, searching all possible subsets is unnecessary. However, we need to vary the type of data distribution and its parameters when evaluating the outlier explanation performance.

Simulating analysts Employing a human analyst seems ideal for evaluating the outlier explanation accuracy; how-

ever, not every researcher has this privilege. Applying classifiers as simulated analysts allows researchers to validate their proposed outlier explanation techniques. While the reported accuracy depends on the quality of the classifiers, we need to note that classifiers have their own biases and limitations. For example, the decision tree classifier is sensitive to overfitting when it is trained on small datasets. The support vector machine (SVM) classifier works well in small datasets, yet it can be slow for large datasets. It is also sensitive to the hyperparameters and the kernel selection.

10 Conclusion and research directions

In this paper, we have defined three categories of outlier explanations: (i) *importance level of outliers*, (ii) *causal interactions among outliers*, and (iii) *outlying attributes of outliers*. Furthermore, the *importance level of outliers* is classified into the *numerical ranking of outliers* and the *categorical ranking of outliers*. We also distinguished the *outlying attributes of an individual outlier* from the *outlying attributes of a group of outliers*. With these explanations, users can gain a better understanding of their data generally and outliers particularly. Thus, they can take appropriate actions based on their applications.

We identified challenges related to the generation of each outlier explanation category in Sect. 3. We reviewed the existing techniques in Sects. 5–7 and discussed whether they address the challenges. None of the techniques proposed for the *numerical ranking of outliers* addresses all the challenges (Sect. 5.1.3 and Table 1). The only surveyed technique for the *categorical ranking* of outliers covers two out of three challenges with some limitations (Sect. 5.2). Two techniques for finding causal interactions among outliers deal with all its challenges (Sect. 6.3). Most of the techniques for outlying attributes of an individual outlier only address one of the challenges (Sect. 7.1.7 and Table 2). None of the techniques for finding outlying attributes of a group of outliers satisfies all three challenges.

Most of the techniques focus on the outlying attributes of an individual outlier and provide only one type of outlier explanation. However, some applications, like irregular heartbeat detection [59] and structural health monitoring (SHM) [14,66], can benefit from more than one type of explanation. Irregular heartbeat detection can benefit from the *categorical ranking of irregular heartbeats (outliers)* and the *outlying attributes of each irregular heartbeat*. SHM can benefit from the *outlying attributes of each outlier* because some anomalies can be caused by sensor system malfunctions but not actual faults on the structure [66]; SHM can also benefit from the *causal interactions among outliers* because certain kinds of damage can lead to other damages, for example, cracks in building walls can trigger other cracks. However, since many real-world applications are time-sensitive, it might not be feasible to run two different algorithms to produce two types of outlier explanations because they are likely to repeat some works, for example, scanning or preprocessing the dataset; instead, it is possible to save work by having a single algorithm generate multiple types of explanations. For this reason, additional research to develop techniques that can provide multiple types of outlier explanations is needed.

Further study is needed to incorporate prior knowledge about the application domain and use the user's feedback to improve outlier explanations. Only two surveyed techniques apply prior knowledge, and one technique includes the user's feedback. To include humans in the loop, applying the active learning (AL) approach can be a way to go. AL is commonly used in rare class classification [3] since it is expensive to manually label all testing data to identify the rare class. In AL, the domain expert is queried to label a subset of the available data samples using an iterative approach. AL has been applied for outlier detection [81], and it is promising to use for the outlier explanation task.

In pursuit of real-time analysis in applications where data arrive in streaming fashion, e.g., network intrusion detection, credit card fraud detection, and structural health monitoring, the ability to efficiently perform computations is necessary. Outlier explanation in Big Data applications can be done faster when using parallel architectures like multi-core CPUs, GPUs [36], and distributed systems like Spark [102]. Parallel algorithms have many performance benefits, but their design is non-trivial because each parallel architecture has its specific research challenges. Furthermore, porting existing serial algorithms in a naïve manner that does not address the research challenges of these architectures can bring significant performance penalties. Among the issues of multi-core CPUs are the efficient uses of shared caches, load balancing among cores, false sharing of caches, etc. Besides the issues just mentioned, GPUs also have a small memory space [75], are connected to the computer through a low throughput PCIe bus, and deal with global memory coalescing, etc. [48]. Finally, distributed architectures like Spark deal with, in addition to the issues of multi-core algorithms, the issue of data partitioning across nodes, minimizing inter-node communication, etc. Despite the potential performance benefits of algorithms designed for these architectures, only two of the surveyed techniques are designed for multi-core CPUs, none is designed for GPUs and four for distributed systems; therefore, more research on parallel outlier explanation algorithms is needed.

Real-time outlier detection in data streams is a growing research area [15,92], but only three of the surveyed outlier explanation techniques are designed for streaming environments. However, those techniques do not address some of the characteristics of data streams, such as concept drift and data

uncertainty. The data distribution in data streams changes over time due to the changes in environments, trends, etc. [82]. As a result, what is considered as an outlier at a certain period of time can be regarded as an inlier in other period of time. Hence, the outlier explanation techniques should adapt according to the concept drift. Furthermore, data sources in stream applications, i.e., wireless sensor networks (WSN), are vulnerable to external events [66,82]. This vulnerability causes data uncertainty where some attributes' values cannot be measured with complete confidence or where data points are missing or out of date. The uncertainty will affect the output of the outlier explanation algorithm. For example, when the algorithm depends on the local neighborhood of the outliers [61], the explanation can be wrong because the actual neighbors are out of date.

Moreover, in multiple-source data streams, data from different sources can be cross-correlated [82]. For example, data streams generated by WSN in harsh environments, i.e., underground oil, forest, and volcanic sites, have attribute correlations, temporal correlation, and spatial correlation [84]. Attribute correlation is the dependency between various attributes, e.g., a sudden rise of temperature is characterized by a sudden drop in humidity. Temporal correlations mean the measured data at a current time is also dependent on past measurements. Spatial correlations imply that the data measurement on a particular sensor node is related to the data measurement of its neighboring nodes (cross-correlation). In this case, outlier explanation algorithms should capture all the correlations to arrive at proper explanations. However, none of the surveyed techniques addresses this trait.

We also reviewed how the surveyed techniques addressed the evaluative aspects of outlier explanations and concluded that most of the techniques focus on Aspect (5), *explanations are truthful*. Ideally, to verify whether the proposed outlier explanation techniques provide correct explanations about outliers, we need benchmark datasets with ground truth concerning which data points are outliers and their explanations. However, the ground truth is not always available in real-world datasets. Thus, we also reviewed some methods used to generate outlier explanation ground truth. We classified the existing ground truth generations into three groups: (i) modifying real-world datasets, (ii) generating synthetic data, and (iii) simulating analysts. Furthermore, we discussed the advantages and disadvantages of each group.

Finally, it is possible to produce outlier explanations by examining the relationships between outliers identified from one dataset and external data references. However, to the best of our knowledge, no current work automatically discusses how to generate outlier explanations with additional data references. Future research on this topic is needed.

References

1. Abdallah, A., Maarof, M.A., Zainal, A.: Fraud detection system: a survey. J. Netw. Comput. Appl. (2016). https://doi.org/10.1016/j.jnca.2016.04.007
2. Adams, J., Hayunga, D., Mansi, S., Reeb, D., Verardi, V.: Identifying and treating outliers in finance. Financ. Manag. **48**(2), 345–384 (2019). https://doi.org/10.1111/fima.12269
3. Aggarwal, C.: Outlier Analysis. Springer, New York (2017)
4. Aggarwal, C.C.: An Introduction to Outlier Analysis, pp. 1–34. Springer, Berlin (2017). https://doi.org/10.1007/978-3-319-47578-3_1
5. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. SIGMOD Rec. **22**(2), 207–216 (1993). https://doi.org/10.1145/170036.170072
6. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1994)
7. Ahmed, M., Mahmood, A.N., Islam, M.R.: A survey of anomaly detection techniques in financial domain. Future Gener. Comput. Syst. **55**, 278–288 (2016). https://doi.org/10.1016/j.future.2015.01.001
8. Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data Min. Knowl. Discov. **29**, 626–688 (2014)
9. Alvarez Cid-Fuentes, J., Szabo, C., Falkner, K.: Adaptive performance anomaly detection in distributed systems using online SVMs. IEEE Trans. Dependable Secure Comput. **17**(5), 928–941 (2020). https://doi.org/10.1109/TDSC.2018.2821693
10. Amarasinghe, K., Kenney, K., Manic, M.: Toward explainable deep neural network based anomaly detection. In: 2018 11th International Conference on Human System Interaction (HSI), pp. 311–317 (2018). https://doi.org/10.1109/HSI.2018.8430788
11. Amato, A.: On the role of distributed computing in big data analytics. In: Distributed Computing in Big Data Analytics, Scalable Computing and Communications, pp. 1–10. Springer International Publishing (2017)
12. Avritzer, A., Tanikella, R., James, K., Cole, R.G., Weyuker, E.: Monitoring for security intrusion using performance signatures. In: WOSP/SIPEW'10 - Proceedings of the 1st Joint WOSP/SIPEW International Conference on Performance Engineering (2010). https://doi.org/10.1145/1712605.1712623
13. Bialas, A., Michalak, M., Flisiuk, B.: Anomaly Detection in network traffic security assurance. In: Advances in Intelligent Systems and Computing, vol. 987 (2020). https://doi.org/10.1007/978-3-030-19501-4_5
14. Bigoni, C., Hesthaven, J.S.: Simulation-based anomaly detection and damage localization: an application to structural health monitoring. Comput. Methods Appl. Mech. Eng. (2020). https://doi.org/10.1016/j.cma.2020.112896
15. Bonial, P., Paparrizos, J., Palpanas, T., Franklin, M.J.: SAND: streaming subsequence anomaly detection. Proc. VLDB Endow. **14**(10), 1717–1729 (2021). https://doi.org/10.14778/3467861.3467863
16. Borowski, S.: The origin and popular use of Occam's razor (2012). https://www.aaas.org/origin-and-popular-use-occams-razor
17. Boukerche, A., Zheng, L., Alfandi, O.: Outlier detection: methods, models, and classification. ACM Comput. Surv. **53**(3), 1–37 (2020). https://doi.org/10.1145/3381028
18. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. SIGMOD Rec. **29**(2), 93–104 (2000). https://doi.org/10.1145/335191.335388
19. Buchbinder, N., Feldman, M., Naor, J.S., Schwartz, R.: Submodular maximization with cardinality constraints. In: Proceedings

of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1433–1452 (2014)
20. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. **41**, 15:1-15:58 (2009)
21. Chen, Z., Tang, J., Fu, A.W.C.: Modeling and efficient mining of intentional knowledge of outliers. In: Seventh International Database Engineering and Applications Symposium, pp. 44–53 (2003)
22. Cook, A.A., Misirli, G., Fan, Z.: Anomaly detection for IoT time-series data: a survey. IEEE Internet Things J. **7**(7), 6481–6494 (2020). https://doi.org/10.1109/JIOT.2019.2958185
23. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**, 273–297 (2004)
24. Dang, X.H., Assent, I., Ng, R.T., Zimek, A., Schubert, E.: Discriminative features for identifying and interpreting outliers. In: 2014 IEEE 30th International Conference on Data Engineering, pp. 88–99. IEEE (2014)
25. Dang, X.H., Micenková, B., Assent, I., Ng, R.T.: Local outlier detection with interpretation. In: Machine Learning and Knowledge Discovery in Databases, pp. 304–320 (2013)
26. Defense Advanced Research Projects Agency. https://www.darpa.mil/program/transparent-computing. Accessed 4 Oct 2021
27. Deng, H.: Guided random forest in the RRF package. arXiv:1306.0237 (2013)
28. Dervilis, N., Cross, E., Barthorpe, R., Worden, K.: Robust methods of inclusive outlier analysis for structural health monitoring. J. Sound Vib. **333**(20), 5181–5195 (2014). https://doi.org/10.1016/j.jsv.2014.05.012
29. Desai, S.: Research guides: fake news, lies and propaganda: how to sort fact from fiction: what is fake news (2018)
30. Dutta, K.: Distributed computing technologies in big data analytics. In: Distributed Computing in Big Data Analytics, Scalable Computing and Communications, pp. 57–82. Springer (2017)
31. Farrar, C.R., Worden, K.: Structural Health Monitoring: A Machine Learning Perspective. Wiley, New York (2012). https://doi.org/10.1002/9781118443118
32. Feng, Q., Dou, Z., Li, C., Si, G.: Anomaly detection of spectrum in wireless communication via deep autoencoder. In: Lecture Notes in Electrical Engineering, vol. 421 (2017). https://doi.org/10.1007/978-981-10-3023-9_42
33. Fernando, T., Gammulle, H., Denman, S., Sridharan, S., Fookes, C.: Deep learning for medical anomaly detection—a survey. ACM Comput. Surv. **54**(7), 1–37 (2021). https://doi.org/10.1145/3464423
34. Fleming, N.: Coronavirus misinformation, and how scientists can help to fight it. Nature (2020). https://doi.org/10.1038/d41586-020-01834-3
35. Fu, Q., Lou, J.G., Wang, Y., Li, J.: Execution anomaly detection in distributed systems through unstructured log analysis. In: Proceedings—IEEE International Conference on Data Mining, ICDM (2009). https://doi.org/10.1109/ICDM.2009.60
36. Garland, M., Kirk, D.B.: Understanding throughput-oriented architectures. Commun. ACM **53**(11), 58–66 (2010). https://doi.org/10.1145/1839676.1839694
37. Guidotti, R., Monreale, A., Matwin, S., Pedreschi, D.: Black box explanation by learning image exemplars in the latent feature space. In: Machine Learning and Knowledge Discovery in Databases, pp. 189–205 (2020)
38. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Comput. Surv. **51**(5), 1–42 (2018). https://doi.org/10.1145/3236009
39. Gupta, N., Eswaran, D., Shah, N., Akoglu, L., Faloutsos, C.: Beyond outlier detection: lookout for pictorial explanation. In: Machine Learning and Knowledge Discovery in Databases, pp. 122–138. Springer (2019)
40. Hawkins, D.: Identification of outliers. In: Monographs on Applied Probability and Statistics (1980)
41. Herath, J.D., Yang, P., Yan, G.: Real-time evasion attacks against deep learning-based anomaly detection from distributed system logs. In: CODASPY 2021—Proceedings of the 11th ACM Conference on Data and Application Security and Privacy (2021). https://doi.org/10.1145/3422337.3447833
42. Hill, D.J., Minsker, B.S.: Anomaly detection in streaming environmental sensor data: a data-driven modeling approach. Environ. Model. Softw. **25**, 1014–1022 (2010)
43. Hodge, V.J., Austin, J.: A survey of outlier detection methodologies. Artif. Intell. Rev. **22**, 85–126 (2004)
44. Huang, B., Yang, P.: Finding key knowledge attribute subspace of outliers in high-dimensional dataset. Expert Syst. Appl. **38**(8), 10147–10152 (2011). https://doi.org/10.1016/j.eswa.2011.02.077
45. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**, 264–323 (1999)
46. Jiang, J., Chen, J., Gu, T., Choo, K.K.R., Liu, C., Yu, M., Huang, W., Mohapatra, P.: Anomaly detection with graph convolutional networks for insider threat and fraud detection. In: IEEE Military Communications Conference (MILCOM), pp. 109–114 (2019). https://doi.org/10.1109/MILCOM47813.2019.9020760
47. Keller, F., Müller, E., Wixler, A., Böhm, K.: Flexible and adaptive subspace search for outlier analysis. In: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, pp. 1381–1390 (2013). https://doi.org/10.1145/2505515.2505560
48. Kirk, D.B., Hwu, W.M.W.: Programming Massively Parallel Processors: A Hands-on Approach, 3rd edn. Morgan Kaufmann, Burlington (2016)
49. Knorr, E.M., Ng, R.T.: Algorithms for mining distance-based outliers in large datasets. In: Proceedings of the 24rd International Conference on Very Large Data Bases, pp. 392–403 (1998)
50. Knorr, E.M., Ng, R.T.: Finding intensional knowledge of distance-based outliers. In: Proceedings of the 25th International Conference on Very Large Data Bases, pp. 211–222 (1999)
51. Kopp, M., Holena, M.: Evaluation of association rules extracted during anomaly explanation. In: ITAT (2015)
52. Kopp, M., Pevný, T., Holena, M.: Interpreting and clustering outliers with sapling random forests. In: Information Technologies—Applications and Theory (2014)
53. Koutris, P., Salihoglu, S., Suciu, D.: Algorithmic aspects of parallel data processing (2018). https://doi.org/10.1561/1900000055
54. Kriegel, H.P., Krager, P., Schubert, E., Zimek, A.: Interpreting and unifying outlier scores. In: Proceedings of the SIAM International Conference on Data Mining, pp. 13–24 (2011)
55. Kriegel, H.P., Kröger, P., Schubert, E., Zimek, A.: Outlier detection in axis-parallel subspaces of high dimensional data. In: Advances in Knowledge Discovery and Data Mining, pp. 831–838 (2009)
56. Leigh, C., Alsibai, O., Hyndman, R.J., Kandanaarachchi, S., King, O.C., McGree, J.M., Neelamraju, C., Strauss, J., Talagala, P.D., Turner, R.D.R., Mengersen, K.L., Peterson, E.E.: A framework for automated anomaly detection in high frequency water-quality data from in situ sensors. Sci. Total Environ. **664**, 885–898 (2019)
57. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 420–429 (2007)
58. Li, D., Guo, H., Wang, Z., Zheng, Z.: Unsupervised fake news detection based on autoencoder. IEEE Access **9**, 29356–29365 (2021). https://doi.org/10.1109/ACCESS.2021.3058809
59. Li, H.Z., Boulanger, P.: A survey of heart anomaly detection using ambulatory electrocardiogram (ECG). Sensors **20**(5), 1461 (2020)

60. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining, pp. 413–422 (2008)
61. Liu, N., Shin, D., Hu, X.: Contextual outlier interpretation. In: 27th International Joint Conference on Artificial Intelligence (2018)
62. Liu, W., Zheng, Y., Chawla, S., Yuan, J., Xing, X.: Discovering spatio-temporal causal interactions in traffic data streams. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1010–1018 (2011). https://doi.org/10.1145/2020408.2020571
63. Luo, Z., Xiong, Y., Zuo, R.: Recognition of geochemical anomalies using a deep variational autoencoder network. Appl. Geochem. (2020). https://doi.org/10.1016/j.apgeochem.2020.104710
64. Macha, M., Akoglu, L.: Explaining anomalies in groups with characterizing subspace rules. Data Min. Knowl. Discov. **32**, 1444–1480 (2018)
65. Mahalanobis, P.C.: On the generalized distance in statistics. Proc. Natl. Inst. Sci. **2**, 49–55 (1936)
66. Mao, J., Wang, H., Spencer, B.F.: Toward data anomaly detection for automated structural health monitoring: exploiting generative adversarial nets and autoencoders. Struct. Health Monit. (2021). https://doi.org/10.1177/1475921720924601
67. Meliou, A., Gatterbauer, W., Halpern, J.Y., Koch, C., Moore, K.F., Suciu, D.: Causality in databases. IEEE Data Eng. Bull. **33**(EPFL-ARTICLE-165841) (2010)
68. Micenková, B., Ng, R.T., Dang, X.H., Assent, I.: Explaining outliers by subspace separability. In: IEEE 13th International Conference on Data Mining, pp. 518–527 (2013)
69. Milenkoski, A., Vieira, M., Kounev, S., Avritzer, A., Payne, B.D.: Evaluating computer intrusion detection systems: a survey of common practices. ACM Comput. Surv. **48**(1), 1–41 (2015). https://doi.org/10.1145/2808691
70. Miller, T.: Explanation in artificial intelligence: insights from the social sciences. Artif. Intell. (2019). https://doi.org/10.1016/j.artint.2018.07.007
71. Molnar, C.: A guide for making black box models explainable. Leanpub. https://leanpub.com/interpretable-machine-learning
72. Myrtakis Nikolaos Christophides Vassilis, S.E.: A comparative evaluation of anomaly explanation algorithms. In: 24th International Conference on Extending Database Technology, pp. 97–108. OpenProceedings.org (2021)
73. Ng, R.: Outlier detection in personalized medicine. In: Proceedings of the ACM SIGKDD workshop on outlier detection and description, p. 7. Association for Computing Machinery (2013). https://doi.org/10.1145/2500853.2500856
74. Niu, Z., Shi, S., Sun, J., He, X.: A survey of outlier detection methodologies and their applications. In: Artificial Intelligence and Computational Intelligence. Lecture Notes in Computer Science, pp. 380–387. Springer, Berlin (2011)
75. NVIDIA: Geforce GPUs. https://www.nvidia.com/en-gb/graphics-cards/ (2021). Accessed 4 Oct 2021
76. Otsuka, T., Torii, Y., Ito, T.: Anomaly weather information detection using wireless pressure-sensor grid. J. Inf. Process. **23**(6), 745–752 (2015). https://doi.org/10.2197/ipsjjip.23.745
77. Pascual, A.L., Kyle, M., Van Aleia, D.: Overcoming False Positives: Saving the Sale and the Customer Relationship. Technical report, Javelin strategy and research reports (2015)
78. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In: Lecture Notes in Computer Science, pp. 398–416 (1999)
79. PNPOLY-Point Inclusion in Polygon Test W. Randolph Franklin (WRF). https://wrf.ecse.rpi.edu//Research/Short_Notes/pnpoly.html. Accessed 4 Oct 2021
80. Pozo-Pérez, J.A., Medina, D., Herrera-Pinzón, I., Heßelbarth, A., Ziebold, R.: Robust outlier mitigation in multi-constellation GNSS positioning for waterborne applications. In: Proceedings of the 2017 International Technical Meeting of The Institute of Navigation, pp. 1330–1343 (2017). https://doi.org/10.33012/2017.14936
81. Russo, S., Lürig, M., Hao, W., Matthews, B., Villez, K.: Active learning for anomaly detection in environmental data. Environ. Model. Softw. (2020). https://doi.org/10.1016/j.envsoft.2020.104869
82. Sadik, M.S., Gruenwald, L.: Research issues in outlier detection for data streams. SIGKDD Explor. **15**, 33–40 (2014)
83. Schubert, E., Wojdanowski, R., Zimek, A., Kriegel, H.P.: On evaluation of outlier rankings and outlier scores. In: Proceedings of the SIAM International Conference on Data Mining (2012)
84. Shahid, N., Naqvi, I.H., Qaisar, S.B.: Characteristics and classification of outlier detection techniques for wireless sensor networks in harsh environments: a survey. Artif. Intell. Rev. **43**(2), 193–228 (2015). https://doi.org/10.1007/s10462-012-9370-y
85. Shalev-Shwartz, S.: Online learning and online convex optimization. Found. Trends Mach. Learn. **4**, 107–194 (2012)
86. Siddiqui, M.A., Fern, A., Dietterich, T.G., Wong, W.K.: Sequential feature explanations for anomaly detection. ACM Trans. Knowl. Discov. Data **13**, 1–22 (2019)
87. Siddiqui, M.A., Fern, A., Dietterich, T.G., Wright, R., Theriault, A., Archer, D.W.: Feedback-Guided Anomaly Discovery via Online Optimization. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2018)
88. Silveira, F., Diot, C.: Urca: Pulling out anomalies by their root causes. In: 2010 Proceedings IEEE INFOCOM, pp. 1–9 (2010)
89. Song, F., Diao, Y., Read, J., Stiegler, A., Bifet, A.: Exad: A system for explainable anomaly detection on big data traces. In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 1435–1440 (2018). https://doi.org/10.1109/ICDMW.2018.00204
90. Takeishi, N.: Shapley values of reconstruction errors of PCA for explaining anomaly detection. In: 2019 International Conference on Data Mining Workshops, pp. 793–798 (2019)
91. Tibshirani, R.: Regression shrinkage and selection via the Lasso. J. R. Stat. Soc. Ser. B (Methodol.) **58**, 267–288 (1996)
92. Tran, L., Mun, M.Y., Shahabi, C.: Real-time distance-based outlier detection in data streams. Proc. VLDB Endow. **14**(2), 141–153 (2021). https://doi.org/10.14778/3425879.3425885
93. Tsai, C.W., Lai, C.F., Chao, H.C., Vasilakos, A.: Big data analytics: a survey. J. Big Data **2**(1), 1–32 (2015)
94. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/index.php. Accessed 4 Oct 2021
95. US7346471B2—Web Data Outlier Detection and Mitigation. https://patents.google.com/patent/US7346471. Accessed 4 Oct 2021
96. Valko, M., Cooper, G., Seybert, A.L., Visweswaran, S., Saul, M., Hauskrecht, M.: Conditional anomaly detection methods for patient-management alert systems. In: Proceedings of the International Conference on Machine Learning. International Conference on Machine Learning, vol. 2008 (2008)
97. Viswanathan, K., Lakshminarayan, C., Talwar, V., Wang, C., Macdonald, G., Satterfield, W.: Ranking anomalies in data centers. In: 2012 IEEE Network Operations and Management Symposium, pp. 79–87 (2012)
98. Wedge, R., Kanter, J.M., Veeramachaneni, K., Rubio, S.M., Perez, S.I.: Solving the false positives problem in fraud prediction using automated feature engineering. In: Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, pp. 372–388. Springer International Publishing (2019)
99. Weller-Fahy, D.J., Borghetti, B.J., Sodemann, A.A.: A survey of distance and similarity measures used within network intrusion anomaly detection. IEEE Commun. Surv. Tutor. **17**, 70–91 (2015)

100. Xing, L., Wang, W., Xue, G., Yu, H., Chi, X., Dai, W.: Discovering traffic outlier causal relationship based on anomalous DAG. In: Advances in Swarm and Computational Intelligence, pp. 71–80 (2015)
101. Yu, R., Qiu, H., Wen, Z., Lin, C., Liu, Y.: A survey on social media anomaly detection. ACM SIGKDD Explor. Newsl. **18**(1), 1–14 (2016). https://doi.org/10.1145/2980765.2980767
102. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: Cluster computing with working sets. In: 2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2010 (2010)
103. Zemicheal, T., Dietterich, T.G.: Anomaly detection in the presence of missing values for weather data quality control. In: Proceedings of the 2019 Conference on Computing and Sustainable Societies (2019). https://doi.org/10.1145/3314344.3332490
104. Zhang, H., Diao, Y., Meliou, A.: EXstream: explaining anomalies in event stream monitoring. In: 20th International Conference on Extending Database Technology (2017). https://doi.org/10.5441/002/edbt.2017.15
105. Zhang, J.: Advancements of outlier detection: a survey. EAI Endorsed Trans. Scalable Inf. Syst. **1**, e2 (2013)
106. Zhang, K., Hutter, M., Jin, H.: A New Local Distance-Based Outlier Detection Approach for Scattered Real-World Data, pp. 813–822. Springer, Berlin (2009)
107. Zimek, A., Schubert, E., Kriegel, H.P.: A survey on unsupervised outlier detection in high-dimensional numerical data. Stat. Anal. Data Min. **5**, 363–387 (2012)