Eurographics Conference on Visualization (EuroVis) 2019
M. Gleicher, H. Leitte, and I. Viola
(Guest Editors)

Volume 38 (2019), Number 3

Oui! Outlier Interpretation on Multi-dimensional Data via Visual Analytics

Xun Zhao[1,2], Weiwei Cui[2], Yanhong Wu[3], Haidong Zhang[2], Huamin Qu[1], and Dongmei Zhang[2]

[1]The Hong Kong University of Science and Technology, Hong Kong
[2]Microsoft Research Asia, China
[3]Visa Research, USA

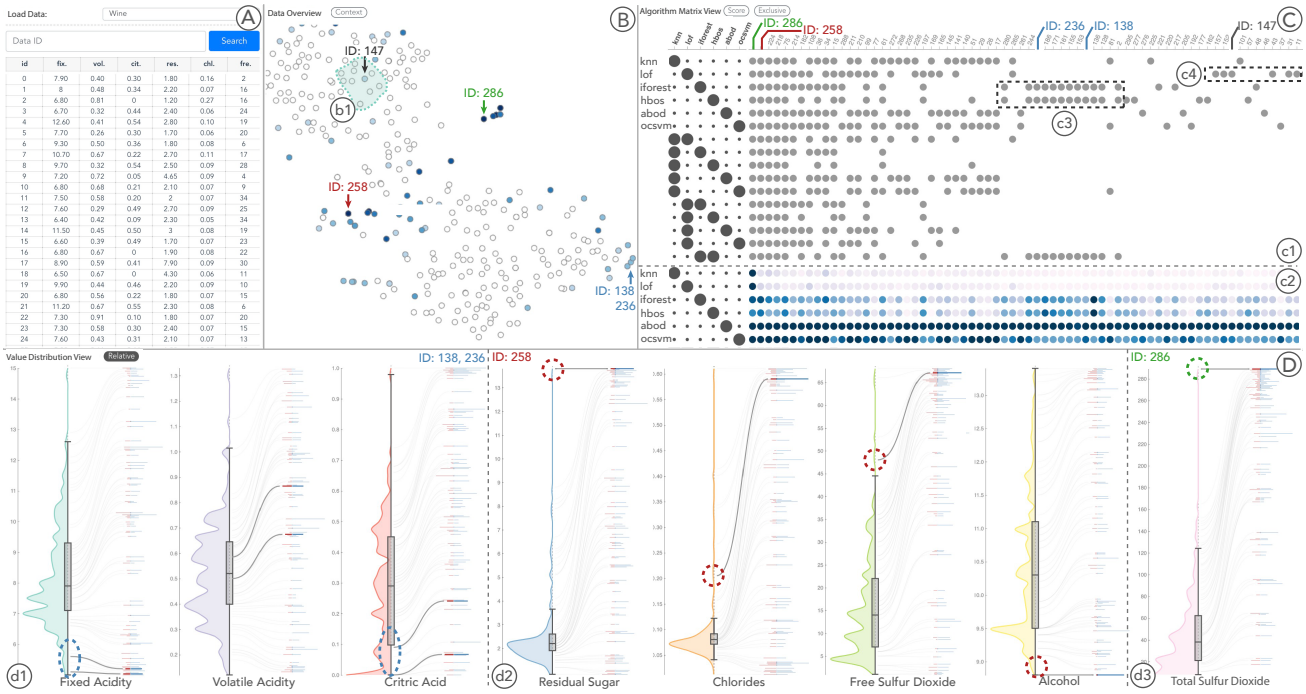

Figure 1: Understanding and interpreting outliers of the Wine dataset using Oui: (A) a Data Table showing raw data and allowing quick searching; (B) a Data Overview displaying an overview of the whole dataset including both outliers and normal data; (C) an Algorithm Matrix View displaying the diverse sets of outliers detected by different algorithms as well as the similarities between these outlier sets; (D) a Value Distribution View allowing users to understand and interpret outliers by showing the attribute values of outliers and statistics of each dimension in diverse contexts.

Abstract

Outliers, the data instances that do not conform with normal patterns in a dataset, are widely studied in various domains, such as cybersecurity, social analysis, and public health. By detecting and analyzing outliers, users can either gain insights into abnormal patterns or purge the data of errors. However, different domains usually have different considerations with respect to outliers. Understanding the defining characteristics of outliers is essential for users to select and filter appropriate outliers based on their domain requirements. Unfortunately, most existing work focuses on the efficiency and accuracy of outlier detection, neglecting the importance of outlier interpretation. To address these issues, we propose Oui, a visual analytic system that helps users understand, interpret, and select the outliers detected by various algorithms. We also present a usage scenario on a real dataset and a qualitative user study to demonstrate the effectiveness and usefulness of our system.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentations]: User Interfaces—Graphics user interfaces (GUI)

1. Introduction

Outliers, the data instances that do not conform with normal patterns in a dataset [CBK09], are widely studied in different domains, such as cybersecurity [BG16], social analysis [SKV10], and public health [WMCW02]. For instance, in financial domain, typical outliers include fraudulent transactions and tax evasion [PLSG10]. By analyzing outliers, experts can better understand suspicious behaviors and improve the security and robustness of financial systems [HPK11]. Another application of outlier identification is data cleaning, *i.e.*, removing noisy or erroneous values as a preprocessing step of data analytics. Numerous statistical and machine learning techniques have been proposed to efficiently detect outliers from different perspectives [HPK11]. However, as different domains usually have different considerations about outliers, there is no "one size fits all" algorithm that universally works well. In addition, the boundary between outliers and normal data is usually blurry [CBK09,HPK11], leading to many false positives / negatives during outlier detections. Hence, it is vital for users to further examine detected outliers in a post-detection phase, reason about the defining characteristics of outliers, and justify whether their abnormal patterns are meaningful to specific domain requirements.

However, this is a challenging task as normal behaviors can be defined in different levels of context [CBK09] and outliers may present a wide range of abnormal patterns [DMAN13]. Outliers not only include the data points that significantly deviate from the whole dataset, but also incorporate the ones that have extreme values in its local neighborhoods or a small and isolated data cluster [CBK09,DMAN13,HPK11]. In addition, the underlying reason that makes an outlier abnormal may also vary. For example, an outlier may have an extremely small or large value on a single dimension, or differ significantly on several dimensions [Foo18, KN99]. In the aforementioned financial fraud detection example, a transaction may be detected as an outlier because its transaction amount is exceptionally high, while another outlier may be detected because a joint effect of unfamiliar recipient and irregular transaction time [CBK09]. A successful outlier interpretation requires users to dig deeply into various aspects of the data to identify the root causes of why an outlier is identified and how it deviates from normal data.

Interactive visualization can benefit outlier interpretation in many aspects by providing intuitive visual representations of data and enabling users to flexibly explore outliers. For instance, by visualizing the data distribution of each attribute, users can closely examine whether the outliers deviate significantly on certain dimensions. However, previous visualization techniques either do not focus on outlier interpretations or lack the flexibility to inspect different perspectives of outliers and the corresponding context. Moreover, users may also need to examine potential outliers in various customized contexts. Thus, it is necessary to develop an interactive visualization system that facilitates users to examine different perspectives of outliers for interpretation and justification.

To tackle these challenges, we introduce Oui, an interactive visualization system that helps users explore, interpret, and justify outliers in multi-dimensional datasets. We first build an Algorithm Matrix View to summarize the outliers detected by a variety of algorithms. Users can also observe the diverse abnormal patterns and freely select the context of outliers in the Data Overview. In addition, we propose a novel design that combines violin plots and divergent bar charts in the Value Distribution View to depict the statistical contexts and the attribute values of outliers. This design allows users to inspect the underlying factors contributing to the outliers abnormality. Our contributions are summarized as follows:

- An interactive visualization system, Oui, to help users explore, interpret, and justify outliers from diverse contexts.
- A novel design that combines violin plots and divergent bar charts to visualize individual outliers and the corresponding context, thereby facilitating outlier interpretation.
- One usage scenario and a user study that demonstrate the effectiveness and usefulness of Oui.

2. Related Work

2.1. Outlier Detection Algorithm

For outlier detection, various algorithms have been proposed over the past decades [CBK09]. Based on how they model normal data or outliers, these algorithms can be mainly divided into six categories: statistical, distance-based, density-based, high-dimensional, classification-based, and ensemble approaches [HPK11].

Statistical-based algorithms [BL74], such as Histogram-based Outlier Score (**HBOS**) [GD12], assume that normal data follow a certain distribution, and data items that strongly deviate from this distribution are outliers. Distance-based algorithms [RRS00] assume that normal data are close to their neighbors and can successfully identify outliers that are distant from their neighbors. K^{th} Nearest Neighbor (k**NN**) [HA04] defines the outlier score as the distance between an item and its k-th nearest neighbor. When data exhibit different local densities, distance-based algorithms may fail to capture certain outliers that are distant from their local neighborhoods [HPK11]. Hence, Density-based algorithms (i.e., Local Outlier Factor (**LOF**) [BKNS00]), which assume that data items with much lower density than their neighbors' are outliers, are proposed. As many existing algorithms suffer from the curse of dimensionality, high-dimensional approaches are proposed. These approaches either find outliers in subspace [ZSK12] or build new models for high-dimensional outliers directly, such as Angle-Based Outlier Detection (**ABOD**) [KSZ08]. Instead of making assumptions about outliers, classification-based approaches [HHWB02] attempt to detect outliers by building a classification model that can distinguish outliers from normal data. Since labels of outliers are usually not available, classification-based algorithms adapt to build a one-class model (e.g., One-Class Support Vector Machine (**oc-SVM**) [SSWB00]), which learns only the normal class. In this case, outliers are data items that do not belong to the normal class. Apart from using a single algorithm for outlier detection, outlier ensembles [GU16,ZCS14] combine multiple individual unsupervised outlier detection algorithms to achieve non-trivial performance improvements. Isolation Forest (**iForest**) [LTZ08] explicitly builds an ensemble of trees for the dataset, then identifies data items that have short average path lengths as outliers.

In our system, to provide users with a comprehensive picture of outliers, we select one representative algorithm from each category, including HBOS, kNN, LOF, ABOD, oc-SVM, and iForest.

2.2. Machine learning Interpretability

As machine learning models have been applied to many critical fields, such as financial and medical domain, the understanding and trust of these models become important for the successful deployment of these models [Lip16, Kim15]. Many work has been proposed to interpret machine learning models, which can mainly be categorized into three classes: feature analysis, surrogate model, and example-based explanations. Feature analysis methods include analyzing which features contribute most to the model decisions [Bre01] or describing the relations between features and prediciton through partial dependence plots (PDP) [Fri01]. Based on PDP, Prospector [KPN16] proposes a heatmap-based design to support users understand how the feature change affects the predictions. A surrogate model is an interpretable model, such as decision trees [SFP*07] and decision rules [Den18], that is learned to approximate the original complex model. For instance, the method proposed by Schetinin et al. [SFP*07] interprets random forests by selecting the tree with highest accuracy and confidence as the surrogate model. Apart from approximating models globally, Riberiro et al. [RSG16] proposed local interpretable model-agnostic explanations (LIME) to explain individual predictions of machine learning models by training local surrogate models. Example-based explanation methods explain the model decisions of individual instances by referring to similar examples [Lip16]. This method originates from case-based reasoning, which proposes that a new case can be solved based on the solutions of similar cases. Among these categories, our paper incorporates the example-based reasoning and feature analysis. By visualizing each attribute's statistics, we support users to justify whether potential outliers are valid and which attributes contribute most to the outlierness score. In addition, we support users to compare a potential outlier with its neighborhood points to understand whether and why a data point is an outlier.

Apart from general machine learning model interpretation, some work focuses on interpreting outliers specifically. Existing outlier interpretation work mainly focuses on finding the deterministic subspaces that distinguish outliers from normal data items [KN99]. For instance, Knorr and Ng [KN99] attempted to provide explanations by finding out in which subspace the outliers are exceptional. Rather from the perspective of deterministic feature subspaces, Kriegel et al. [KKSZ11] attempted to interpret outliers by unifying the outlier scores generated by different algorithms. Ninghao et al. [LSH17] proposed a Contextual Outlier INterpretation method (COIN) that takes the local context of outliers into the consideration. Though these methods attempt to interpret outliers from different perspectives, they are incapable of incorporating users' domain knowledge and of allowing users to closely examine the differences between outliers and normal data.

2.3. Outlier Visualization

Multi-dimensional data visualizations are often used to help users understand and justify whether the auto-detected outliers satisfy the domain-specific requirements. For example, dimensional reduction techniques, such as principal component analysis (PCA) [Jol11] or multidimensional scaling (MDS) [Dav91], project high dimensional data into low dimension to identify clusters and distinguish outliers. However, they often cause information loss. Meanwhile, parallel coordinate plots [Ins85] and scatter plot matrices [BCS96] can visualize all the dimensions of data at once. By depicting the overall data patterns, these two methods can explore and detect outliers to a certain degree. Unfortunately, they are insufficient for many domains with more diverse data types, such as social media.

In addition to general multi-dimensional data visualization techniques, many designated analytic systems have been developed for specific domains or data types. FluxFlow [ZCW*14] is an interactive visualization system for detecting and analyzing anomalous information diffusion on social media like twitter. Similarly, Cao et al. proposed TargetVue [CSL*16] to detect and visualize the anomalous users in online communication systems. Combining the powerfulness of outlier detection algorithms and the expressiveness of visualizations, these systems have demonstrated their usefulness in various domains and complex datasets. Recently, Cao et al. [CLGD18] has further introduced a general visualization design, Z-Glyph, a family of glyphs to support the outlier detection of multi-dimensional data from various domains. Nevertheless, these aforementioned methods only focus on differentiating the individual outliers from the normal data, while neglecting a special kind of outliers called *rare categories*, which lie in between the individual outliers and the predominant patterns. To tackle this issue, Lin et al. [LGG*18] proposed a visualization system called RCLens that assists users in exploring and identifying rare categories in the datasets by leveraging active-learning algorithms. Xu et al. [XXM*19] recently proposed a visual analytics system that utilizes ensemble techniques to facilitate users accurately identify outliers by aggregating multiple algorithms' results while our approach focuses more on interpreting how outliers differ from normal data in different context. In summary, most visualization systems mainly focus on helping users detect outliers, while our system aims to enable users to interpret the auto-detected outliers and justify whether these outliers satisfy domain-specific requirements.

3. Design Goals

Based on a literature review of papers from interpretable emachine learning, data mining, and human-computer interaction fields and an informal interview with a researcher focuses on outlier detection in an industry research lab, we distilled the following design goals.

G1: Highlight the differences between outliers and normal data from different perspectives. As the boundary between outliers and normal data is usually fuzzy, users need to examine how outliers behave abnormally from different perspectives for better decision making [Leo98]. To start, outlier score [BKNS00] can help users quickly grasp the outlierness degrees of data that considered by algorithms. It is also necessary to examine both raw attribute values of outliers and statistics of attributes, including the value distributions, the minimum and maximum values, etc. By comparing the values with the statistics, users can assess whether and how the detected outliers deviate from the normal data on certain attributes. Another crucial perspective is the relationships between an outlier and other normal data points, which allow users to examine whether they form a cluster or how the outlier deviates from the local clusters. Hence, our visualization system needs to support highlighting the differences between outliers and normal data from various perspectives.

G2: Understand and verify outliers in different contexts. Outlier interpretation not only requires highlighting the differences between outliers and normal data, but also needs to examine outliers in different contexts [LSH17]. Specifically, the context of an outlier is defined as a set of reference data points that are considered as normal [KKZ10]. Accordingly, an outlier can be verified by comparing it with its context to examine whether this data point has extreme values of certain attributes. Inspecting outliers in different contexts helps users better justify whether the detected outliers deviate from normal data in a finer granularity. For example, apart from the outliers that have extreme attribute values, some outliers may only deviate from a local data cluster or a few neighbors. Capturing these "outliers in a local context" is useful for detecting subtle anomalies such as credit frauds. By allowing user to investigate outliers with respect to different contexts, users can infer how outliers act abnormally in details.

G3: Compare outliers detected by different algorithms. Since different outlier detection algorithms have different assumptions of outlier characteristics, the detected outliers usually vary [FAP*17, ZCS14]. Instead of considering all the results as outliers with blind faith, users usually need to select and filter appropriate ones based on specific domain requirements. However, manually examining outliers one by one can be time-consuming and takes great efforts. As the outliers that are detected by the same algorithm may share similar anomaly patterns, a more plausible way is to first observe the defining outlier characteristics of different algorithms and then exclude inapplicable results. For example, *k*NN method focuses on detecting the outliers that significantly deviate from most data in the dataset [DMAN13], while LOF method performs better on detecting outliers that only deviate from its local neighbors [GU16]. Users can then narrow down to a smaller subset of data items for further exploration and analysis. Thus, comparing detected outliers of different algorithms is necessary in filtering and selecting outliers that satisfy domain requirements.

G4: Support an interactive exploration and verification of outliers. During the analysis process, the system needs to support various interactions to dynamically explore outliers and normal data. For example, users may need to select certain data points as normal context to compare with outliers. Furthermore, as users deepening their understanding of the data during the exploration process, they may want to narrow down to a subset of outliers in a local context for detailed examination. Therefore, it is crucial to allow users to customize context and select outliers of interest. In addition, when comparing outliers detected by different algorithms, user may pay special attention to the outliers detected by certain algorithms of interest. Hence, allowing users to highlight the shared or exclusive outliers detected by these algorithms is useful for comparison. To fulfill these requirements, a variety of interactions, such as filtering, searching, and linking need to be supported.

4. Visual Design

Motivated by above design goals, we designed Oui, a web-based interactive visualization system that supports users to flexibly identify, and more importantly, to interpret the detected outliers. Oui contains a data processing module and a visual analysis module. The data processing module computes a set of outlier candidates for each selected outlier detection algorithm we selected. As shown in Fig. 1, the visual analysis module contains three major views: 1) a *Data Overview* (Fig. 1B) which provides users with an overview of the whole dataset including both outlier candidates and normal data; 2) an *Algorithm Matrix View* (Fig. 1C) which displays the diverse sets of outliers detected by different algorithms as well as the similarities and differences among these outlier sets; 3) a *Value Distribution View* (Fig. 1D) which aims to help users justify whether the detected outlier candidates are reasonably abnormal by showing the distribution and statistics of each dimension in diverse contexts. We also provide a *Data Table* (Fig. 1A) to allow users to load data into our system and select outliers. A rich set of interactions is also supported to link these different views and support users to dynamically explore data. We use Python to implement the data processing module and utilize Flask to serve as the back-end server. The visual analysis module is developed using D3.

4.1. Data Overview

The Data Overview aims at capturing the overall patterns of data similarity and distribution to guide further exploration (**G1**). In addition, users can flexibly examine and select data points of interest during their exploration (**G4**).

Many techniques can be utilized to visualize multi-dimensional data, such as scatter plot matrices (SPM) [BCS96], parallel coordinate plots (PCP) [Ins85], and dimension reduction techniques [Dav91, MH08]. SPM and PCP can reveal the correlations between attributes and accurately represent a large number of data points, but they are not space-efficient when dimension size increases. In contrast, although dimension reduction methods may not accurately reflect the exact data distances in the original feature space, it preserves similarities between data with good scalability. Since this view is designed for users to get an overview of data, we adopt dimensional reduction techniques to visualize data with higher scalability. Specifically, we choose t-SNE [MH08] as it preserves a strong clustering effect on similar data items compared with other methods such as Multidimensional Scaling (MDS) and Principal Component Analysis (PCA). In outlier interpretation settings, this feature helps highlight the abnormal data points that deviate from clusters. The multi-dimensional data are projected onto a 2D space, in which similar points are placed closely to each other. Each data point is encoded as a circle, where the circle color saturation encodes the number of algorithms that consider it as an outlier. Specifically, we use a sequential color scheme in which a white color may indicate a normal data point while a dark blue circle may suggest a highly suspicious candidate.

We also support various interactions to guide users further explore data in details. For example, the Data Overview supports panning and zooming to let users focus on a region of interest. This also reduces the visual clutter problem when many similar data points are positioned in a small area. When users hovering over a circle, a radar chart [CVW11] will pop up to show the detailed information of the data point. Users can also click on a circle to highlight the corresponding data point. The previous selected data points will also appear as light gray polygons in the radar chart for users to compare in detail. To support users to flexibly customize various contexts for outlier interpretation (**G2**), Oui also enables users to

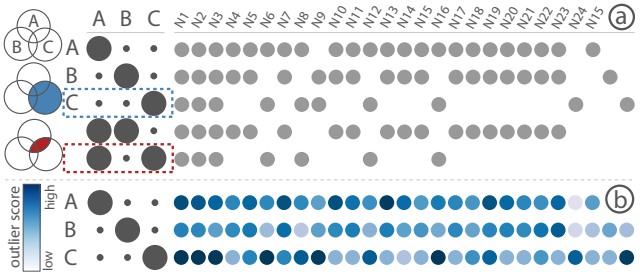

Figure 2: Visual encoding of the Algorithm Matrix View: (a) the *default mode* that displays the outliers detected by a single algorithm and different algorithm combinations and (b) the *outlier score mode* that displays outlier score distribution of each algorithm.

select a subset of data points in the Data Overview as the context and to update the Value Distribution View accordingly. Specifically, the *context mode* is triggered by clicking the context button (as shown in Fig. 1B). In this mode, users can draw a lasso to select points of interest as the current context. If no point is selected by the lasso, the whole dataset will be used as the context.

4.2. Algorithm Matrix View

Since different outlier detection algorithms have different assumptions on outlier characteristics, their detected outlier sets can be different and overlap with each other. Thus, we design the Algorithm Matrix View to allow users to explore and compare the outlier detection results of different algorithms (**G3**). Users can explore outliers from the algorithm's perspective and gain a preliminary understanding of different outlier categories to guide exploration (**G1**).

As shown in Fig. 2a, we employ a matrix-based design to display the outliers detected by each single algorithm and different algorithm combinations. For an algorithm combination, the corresponding outliers are defined as the intersection of outliers detected by individual algorithms contained in the combination. Each column represents a data point and we list all the data points that are detected as outliers by at least one algorithm. Each row represents an algorithm or an algorithm combination. At the beginning of each row, we use an algorithm combination glyph that follows the set visualization design proposed in UpSet [LGS*14] to demonstrate the corresponding algorithm combination. Specifically, each circle in the glyph represents a specific algorithm. All the algorithms that belong to the corresponding row's combination are marked as solid black circles while the algorithms that are not in the combination are represented as small gray dots. For example, as shown in Fig. 2a, the algorithm combination glyph in blue rectangle indicates that this row displays outliers detected by algorithm C. The glyph in red rectangle indicates that the corresponding row shows the outliers detected by both algorithm A and C. The order of the algorithms is fixed and consistent for each row. The rows are sorted based on the number of algorithms in a combination in an ascending order from top to bottom. By default, each matrix cell indicates whether the corresponding data point is detected as outliers by all the algorithms indicated by the algorithm glyph. The matrix cells are sorted based on the number of algorithms that detect the corresponding data point as an outlier in a descending order from left to right. By clicking on the *Exclusive* button (Fig. 1C), we can switch each row to show the outliers that are exclusively detected by the corresponding algorithm or algorithm combinations. In this mode, the algorithm glyph is shown as a white circle with black stroke instead of small gray dots when the corresponding algorithms are not in the combination.

The Algorithm Matrix View also supports various interactions to help users highlight outliers of interest and examine detailed information. When users hover on the algorithm glyph, the corresponding row will be highlighted in red. Similarly, when users hover on the matrix cell, the corresponding algorithm glyph and the whole outlier column will also be highlighted in red. Instead of a single "yes-or-no" answer, we support users to examine the outlier score generated by each algorithm to compare outliers to a finer granularity. By clicking on the outlier score button (Fig. 1C), the rows that represent a single algorithm will switch to the *outlier score mode* as shown in Fig. 2b. In the *outlier score mode*, we use a sequential color scheme to fill each matrix cell circle where a darker blue indicates a higher outlier score.

Design Alternative. Before adopting the current design, we have also considered other design alternatives such as the radial design. Each algorithm is represented by an uniformly-shaped non-overlapping region arranged radially on a circle. To represent the statistical information of the outliers detected by different algorithms, each region also includes a glyph that divide the corresponding outliers into multiple sets based on the number of algorithms that detect them as outliers concurrently. We aggregate the outliers detected by an algorithm into groups according to their degrees, which corresponds to how many algorithms have detected this outlier. For each set, outlier groups are represented by bars, which are arranged radially inside the corresponding region. The length of bar represents the size of the outlier group, where longer bar indicates larger size. The curves linking different regions inside the circle depicts the intersections of results between any two algorithms. The curve width represents the cardinality of the intersections. However, this design suffers from severe visual clutter problem resulting from link crossing when the number of algorithms is large. Compared to the matrix design, the radial design is less space-efficient. In addition, it is difficult to inspect how many algorithms have simultaneously detected a specific data item as an outlier. For these reasons, we discard this design alternative and adopt our current matrix design.

4.3. Value Distribution View

Though the Data Overview and the Algorithm Matrix View provide users an overview of the detected outliers, users still need to drill into a subset of interesting outliers to discover their anomalous patterns in detail. Therefore, we design the Value Distribution View to allow users to interpret outliers by closely examining on which attributes the data points significantly deviate from the distribution of normal context (**G1**). Apart from considering the whole dataset as the context, users can also customize the context by lassoing nodes of interest in the Data Overview (**G2**).

To compare outliers with normal data, we design this view to display the information of normal context and outliers side by side. The Value Distribution View is a list of horizontally arranged

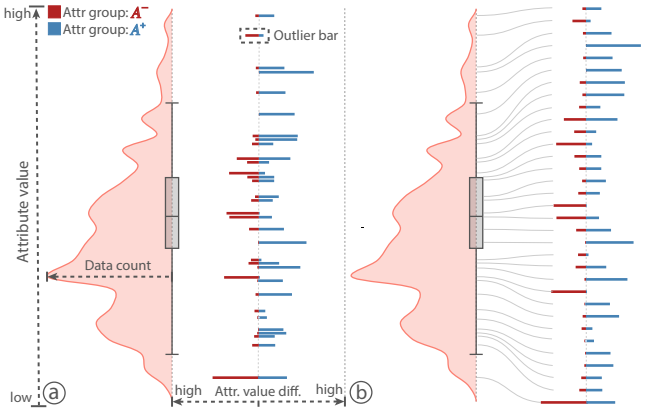

Figure 3: Visual encoding of the Value Distribution View in (a) the *absolute position mode* and (b) the *relative position mode*. The *context component* is a violin diagram that illustrates the statistics and distribution of the context on a dimension; The *outlier component* part utilizes divergent bar charts to display the values and deviations of the selected outliers.

columns in which each column corresponds to an attribute. As shown in Fig. 3a, each column consists of two components: the left *context component* represents the statistics and distribution of the context on this attribute while the right *outlier component* displays the selected data points that are considered as potential outliers. We also place a vertical axis between these two components to indicate attribute values (increase from bottom to top).

In the context component, we use a violin diagram that consists of a box plot [HN98] and a density plot to display the statistics of the context. By default, the context refers to the whole dataset while users can switch the context to any group of data items by lassoing points of interest in the Data Overview. We position a vertical box plot along the axis to encode the statistics of the context such as the first quartile, the median, and the third quartile. The box plot is used to indicate the critical attribute value thresholds which help user assess whether an outlier has an abnormal value in the corresponding attribute. We also use a density plot to show the detailed value distribution of context in which the flow width indicates the number of data points residing in the corresponding value ranges.

In the outlier component, each horizontal bar, which consists of a left red part and a right blue part, represents a specific data item. Each bar is positioned vertically based on the actual value of the corresponding attribute (*absolute position mode*). By default, the outlier component shows all the detected outliers. Users can further select the data items that they consider as potential outliers in the Data Overview or the Algorithm Matrix View to highlight them. After selection, only the bars that represent the selected data items are displayed along the vertical axis (sorted based on their values). This enables users to observe the value distributions of the selected data on a single attribute. We also enable users to quickly observe whether the selected data have abnormal values on other attributes from the horizontal bar. For each attribute, we choose the median value of the current context as the baseline value of this attribute. In this case, for each outlier, we can divide its attribute values into two groups: one group A^- that includes the attribute values below the baseline values, and the other A^+ that includes the attribute values above the baseline values. As shown in Fig. 3a, its aggregated value differences between the attribute values and the baselines in A^- and A^+ are represented by the lengths of left and right bars, respectively. For an outlier p_i, its normalized value difference from the baseline on any attribute a_l can be calculated as:

$$\delta_{a_l}(p_i) = |p_i^l - baseline^l| / range^l \quad (1)$$

where p_i^l is the value of p_i at attribute a_l, $baseline^l$ is the baseline value of the current context on attribute a_l, and $range^l$ represents the total value range of the current context on a_l. Therefore, the aggregated differences for the attribute values in group A^- can be calculated as: $\Delta(p_i)^- = \sum_{a_l \in A^-} \delta_{a_l}(p_i)$. Similarly, the aggregated differences for the attribute values in group A^+ can be computed as: $\Delta(p_i)^+ = \sum_{a_l \in A^+} \delta_{a_l}(p_i)$. These bars help users obtain an overall understanding of the deviation of the outlier in other attributes.

One potential issue of this view is that the horizontal bars can overlap with each other when multiple selected points have similar values on an attribute. To improve the scalability, we enable users to examine their relative order by uniformly spreading the bars vertically (*relative position mode*). In this mode, we use light gray curves to connect the horizontal bars to their real attribute values on the axis as shown in Fig. 3b. When hovering on a bar or a connecting curve, all bars and curves that correspond to the same outlier in all columns will be highlighted while the opacity of all other bars and curves will be set to zero.

The *absolute position mode* can provide the accurate values of outliers on each attribute directly via the positions of bars, thereby allowing users to easily perceive the distributions of detected outliers. On the other hand, the *relative position mode* can avoid visual clutter, so that users can clearly observe the outlier bars. Although the *relative position mode* may require users to track the connecting lines to know the exact values, users reported that it is preferred in our experiment. Therefore, we use the *relative position mode* by default. However, users can switch to the *absolute position mode* by clicking a button as shown in Fig. 1D.

Design Alternative. When designing the Value Distribution View, we considered several design choices. We considered the vanilla density plot and violin diagrams in designing the context component. We choose violin diagram over density plot, as the violin diagram can provide more informative statistical information, such as the median and min-max value, than the vanilla density plot without occupying more spaces. In addition, we have considered to overlay the bars on the violin diagram. Despite this design is more space-efficient, we discard it considering several reasons. First, the design would cause severe visual clutter problems that bars with same absolute value would overlay on each other. In addition, users may find it difficult to perceive the bar lengths from the background distribution information, which might leads to interpretation errors. Hence, we select current design as the final design decision based on the above reasons.

5. Usage Scenario

In this usage scenario, we describe Alex, a wine collector, plans to dispose of some low-quality wines in his collection. He loads the Red Wine Quality dataset [UCI18] into Oui and aims to verify

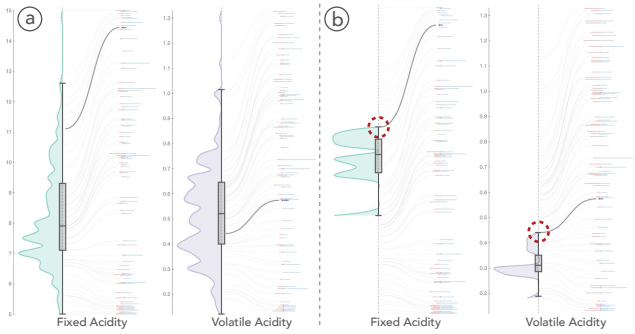

Figure 4: Examining wine *147* in the Value Distribution View: (a) the context is the whole dataset; (b) the context is the selected local neighborhood points of wine *147*.

whether the detected outliers have low quality. The dataset contains 11 numerical attributes, such as *Fix Acidity*, *Sulphates*, and so on.

From the dark blue circles (Fig. 1B), Alex identifies a wine (*ID = 258*), which was bought before his wine cellar was renovated. He suspects that this wine has been contaminated during the renovating period, so he clicks on the corresponding circle to highlight wine *258* in both the Value Distribution View and the Algorithm Matrix View to examine the detailed information (**G4**). In the Value Distribution View, he observes that this wine has values outside the whiskers of the box plots on several attributes. As shown in Fig. 1*D*, wine *258* has extremely large values on *Residual Sugar*, *Chlorides*, and *Free Sulfur Dioxide*, while extremely small value on *Alcohol* (**G1, G2**). He thinks these indicators suggest that this wine may go bad and wants to further examine which outlier detection algorithms have detected this wine as an outlier, so that he can further examine whether the other outliers found by these algorithms also have low quality.

Then, he switches to the Algorithm Matrix View (Fig. 1c_1), and observes that all the outlier detection algorithms detect this wine as an outlier. Apart from wine *258*, four other wines are also detected as outliers by all algorithms. To quickly observe the differences between these five wines, he clicks the *Score* button to display the generated outlier scores of these wines. By observing the circle color distribution of each row (Fig. 1*c*2), he finds that kNN and LOF share a similar but special score distribution where only one circle (wine *286*) is colored in dark blue while other circles all have a low saturation. This indicates that these two algorithms both exhibit an extremely unbalanced score distribution, in which one data item has been assigned with a very high outlier score while other data items have small outlier scores (**G1, G3**).

To investigate why wine *286* obtains this high score, he clicks on the circle to add it into the Value Distribution View for further examination. From the Value Distribution View, he observes that *Total Sulfur Dioxide* has a value of 289 (Fig. 1d_3), which deviates significantly from the top whiskers of the box plot (**G2**) and exceeds the maximum legal limits for this attribute [JLMdO11]. As the value is so abnormal, he infers that the data of wine *286* might be inaccurately documented.

From the Algorithm Matrix View, Alex has also observed two interesting patterns (Fig. 1c_1). First, many data points are exclusively identified as outliers by both iForest and HBOS (Fig. 1c_3). Second, although LOF shares many outliers with other algorithms, it has many exclusively ones (Fig. 1c_4). Alex decides to further investigate these two interesting patterns respectively (**G3**).

He first examines the outliers exclusively detected by both iForest and HBOS. By clicking on the corresponding rows in the Algorithm Matrix View, the outliers detected by both algorithms are highlighted in the Data Overview (**G4**). In Fig. 1B, He observes that a small cluster has been formed among these highlighted circles (*e.g.*, wine *138* and *236*). By examining the detailed information in the Value Distribution View, he also observes that these outliers have similar values on most attributes (**G2**). From the vertical positions of highlighted outlier bars, Alex finds that these wines all have very small values on both *Fixed Acidity* and *Citric Acid* (Fig. 1d_1), which indicates that these wines have "flat flavor" as these attributes are critical for the taste and flavor of wine.

He also investigates the outliers exclusively detected by LOF. He hovers on the corresponding row to highlight these outliers in the Data Overview. In Fig. 1b_1, He observes that wine *147* is surrounded by normal data (**G1**). To verify whether it is an outlier, He clicks on the corresponding circle and highlight it in the Value Distribution View (**G4**). Then, he surprisingly finds that this wine does not have extreme values on any dimensions (Fig. 4a). Furthermore, the outlier bar length is rather short, which indicates that wine *147* is not significantly different from other wines in general. To ensure that wine *147* still maintains a high quality, he further draws a lasso to select some neighbor points (Fig. 1b_1) as the local context in which he can inspect wine *147* in a finer granularity (**G2**). The violin plots in the Value Distribution View are then updated to display the distribution and statistics of the selected context. From the vertical position of the outlier bar of wine *147*, he observes that it has rather large *Fixed Acidity* and *Volatile Acid*, which indicates that wine *147* may be too sour compared to its neighboring wines (Fig. 4b). Since Alex prefers sweet flavors, he considers wine *147* as an outlier. Therefore, Alex decides to dispose of the abnormal wines analyzed above, such as wine *147* and *258*.

6. User Study

We conducted a user study to assess the effectiveness of our system for outlier interpretation. A formal comparative study between Oui and a baseline outlier visualization system is not applicable because the existing systems either do not focus on outlier interpretation or are limited to specific data types. In addition, tasks that involve outlier interpretation may require users to inspect different perspectives of outliers in various contexts, which cannot be simplified to yes or no questions. Thus, we chose to conduct a qualitative user study rather than a controlled quantitative experiment.

6.1. Participants and Apparatus

We recruited 12 participants (ten males and two females, aged 19 to 28 (mean = 24, SD = 2.4)) with normal to corrected-to-normal vision. All the participants were undergraduate or postgraduate students. Most of the participants majored in computer science, while one participant majored in chemistry. All the participants have a technical background, but with limited knowledge of outliers. The

experiment was conducted on a laptop computer with an external 24-inch display of resolution 1920 × 1080 pixels.

6.2. Dataset

We adopted the Numbeo quality-of-life dataset [Num18], which describes the overall living conditions of 176 cities worldwide. This dataset contains eight numerical attributes: *Purchasing Power*, *Safety*, *Health Care*, *Living Cost*, *Affordability* (a measurement for house purchase affordability), *Traffic*, *Environment*, and *Climate*.

6.3. Tasks and Design

Ten tasks were designed to cover all the important perspectives of the design goals for outlier interpretation (Table 1). We intended to evaluate whether our system can assist users in understanding and examining why a data item is an outlier as well as identifying the different characteristics of the outlier. We had carried out several pilot studies to ensure the tasks were designed properly. The participants needed to utilize all views in Oui to successfully accomplish all the tasks. Users are required to perform all tasks in a sequential order. Specifically, Task 1 and Task 3 mainly focus on evaluating whether the Data Overview can help users identify the diverse types of outliers. Tasks 4–6 assess whether the Algorithm Matrix View can provide users an overview and allow them to compare outliers detected by different algorithms for further data exploration. Task 2 and Tasks 7–8 evaluate whether the Value Distribution View helps users analyze whether and why a data point is an outlier from the global context. Tasks 9 and 10 are follow-up questions of Task 8. Specifically, Task 9 focuses on identifying a local context and analyzing its distribution, and Task 10 focuses on evaluating whether the selected data point is an outlier deviating from the local context.

6.4. Procedure

We began each study with a brief introduction on the outliers and the visual interfaces of our system. During the introduction, we used the Wine dataset to help participants to get familiar with our system. After the introduction, we encouraged the participants to freely explore our system and try diverse interactions with the Wine dataset. The participants were asked to think aloud and raise questions when they encountered problems during the exploration stage.

In the formal study, to avoid the memorization effect on the Wine dataset, we adopted the Numbeo quality-of-life dataset for users to perform all the tasks. After the participants finished all ten tasks, they were asked to fill out a questionnaire including 15 questions about the effectiveness and aesthetics of our system. Each question is designed to evaluate our system using a 7-likert scale ranging from 1 (totally disagree) to 7 (totally agree). Furthermore, we conducted a post-study interview to gather participants' feedback on our system, such as which view is their favorite. The whole process which includes the introduction, tasks, questionnaire, and the post-study interview lasted around one hour.

6.5. Results

We now present the results obtained from the controlled user study. For each participant, we record the completion time of each task and the rating of each question in the questionnaire. Then, we computed the means and 95% confidence intervals (CI) of all participants. We also took notes of the questions and feedback raised by participants for further analysis.

6.5.1. Completion Time

The detailed completion time of each task is shown in Fig. 5. In general, all participants managed to complete the tasks in a short period of time. However, three tasks (Tasks 2, 4, and 9) took longer time than the other tasks. This is reasonable for Tasks 2 and 9 because Task 2 requires participants to first select one outlier and examine all attribute columns to discover on which attributes the data points significantly deviates from the normal data, and Task 9 requires participants to manually draw a lasso to define the local context of a selected data point. On the other hand, the fact that Task 4 took a rather longer time is unexpected. Our hypothesis is that participants can identify any two similar sets of outliers with a quick glimpse, but in the user study, many of the participants spent a lot of time to find the most similar pair of outlier sets instead of any two similar ones by cross-checking several times.

6.5.2. Questionnaire Results

The questionnaire is designed to evaluate whether our system has fulfilled all the design goals as mentioned in Sec. 3 and the overall performance. Specifically, the questionnaire contains goal-related questions and overall questions. For the goal-related questions, they have been separated into four parts and each part (three questions) corresponds to a goal. The overall questions inquiry users about the overall characteristics of our system, such as the interactions, aesthetics, and usefulness. Many participants reported that the system is useful for understanding and interpreting outliers, the visualization is visually pleasing, and the interactions are easy in general.

6.5.3. Subjective Feedback

We also conducted post-study interviews to inquiry more feedback from the participants on the Oui. In this post-study interviews, most participants appreciated the usefulness and effectiveness of Oui in helping them understand whether and why a data item is an outlier. The questions asked are listed as follows: What views or features do you like most? Which parts do you think need to be improved? Which scenarios or domains do you think the system may help with outlier interpretation?

Favorite Features. Most participants (75%) like the Value Distribution View the best. Many participants commented that the Value Distribution View provides a straightforward and clear way for them to interpret outliers. A participant commented that " *From the Value Distribution View, I can inspect exactly which attributes are the reasons that cause the data item as an outlier.*" Other participants added that "*I can see the normal range, so that I can determine whether the data deviates significantly on each attribute.*" One participants also conveyed that this view is visually pleasing.

Some participants also valued the insights provided by the Algorithm Matrix View. One participants commented that "*I like that the Algorithm Matrix View allows me to view the results of different algorithms simultaneously. This improves the credibility of results*

Task	Description
T1	Can you identify any data points that are most strongly predicted as outliers? If yes, please specify the data points.
T2	Select one outlier you identified in T1; can you identify on which attributes that this point significantly deviates from the distribution of normal data?
T3	Can you identify any rare categories (outliers that form a tiny cluster)?
T4	Can you identify two algorithms that detect a similar set of outliers?
T5	For the two algorithms you identified in T4, how many outliers are detected by both of them?
T6	For the two algorithms you identified in T4, do they also share a similar outlier score distribution?
T7	Given a data point A, does it deviate significantly from other data points on Attribute *Affordability*?
T8	Considering all the attributes excluding *Affordability*, does A deviate significantly from the normal data? If so, how does it deviate from the normal data?
T9	Draw a lasso around the neighborhood points of A to specify its local context. Can you describe the distribution of the local context on Attribute *Affordability* in the Value Distribution View?
T10	Can you identify on which attributes A deviates significantly from the distribution of the local context selected in T9?

Table 1: Experimental tasks conducted to cover all the important perspectives of the design goals for outlier interpretation.

as I can hardly trust the results of any single algorithm." On the other hand, some participants valued the importance of the Data Overview, e.g., "*The Data Overview can provide a starting point to explore the outliers and link with other views for further exploration.*" Three participants also demonstrated that the local context is useful for interpret outliers, e.g., "*I like the setting of local context, so that I can focus on comparing the outliers with its neighborhood points, which is more effective and meaningful.*"

Limitations and Improvement. Apart from the positive feedback, the participants also provided several suggestions on improving our system. For the Data Overview, three participants who spent relatively long time to select the local context of a selected outlier, commented that it would take a while for them to get familiar with how to draw a lasso efficiently. This suggests that there might be a small learning curve for some participants to draw a lasso fluently. For the Algorithm Matrix View, three participants can flexibly choose the combinations of outlier sets detected by different algorithms, instead of searching through all the combinations each time. We added a function in the Algorithm Matrix View to allow users to issue any query of the algorithm combinations. Two participants requested that, after they selected one point in the Data Overview, they wanted to examine the details of the corresponding highlighted bars in the Value Distribution View. However, when they hover the mouse inside the Value Distribution View, other bars are highlighted whenever the mouse went. Therefore, when any point has been selected in the Data Overview, we keep the corresponding bars in the Value Distribution View highlighted unless the point is unselected in the Data Overview.

Applications. Apart from the visual design, most participants also appreciated that our system can be useful in many domain application scenarios. Many participants commented that the system can be very useful in financial and medical area. In addition, three participants commented that our system could be useful for data cleaning, e.g., "*The system can be used to analyze the input data. For example, whether the data is biased or contains noise.*" Two participants whose research direction is system development also reported that "*The system can be very useful for debugging which parts went wrong in the large scale system.*"

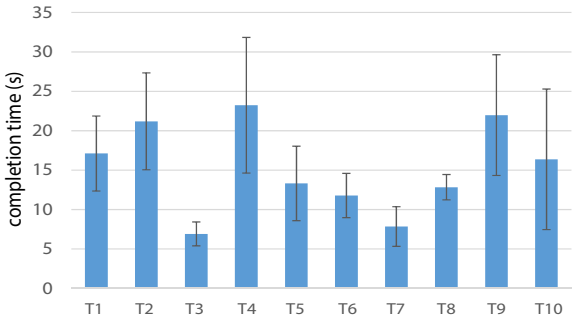

Figure 5: Average completion times for each task. Error bars show 95% confidence intervals ($n = 12$).

7. Discussion

7.1. Scalability

Similar to many visualization systems, one crucial problem we considered during the process of designing Oui is the scalability.

In the Data Overview, too many data points could cause severe overlapping problems. To mitigate this problem, we increase the distances between overlapped circles using an overlap removal technique. In addition, we support users to interact, including zooming and panning, to focus on certain regions of interest. In our system, the Data Overview can support hundreds to thousands of data points overall. We can further improve the scalability by adopting clustering, sampling, or highlighting techniques. For instance, when the data size grows larger, we can adopt an outlier-based random sampling approach to address the scalability issue [LXL*18]. By assigning a higher sampling rate for outliers than for the normal data items, we can efficiently alleviate the visual clutter problem while still maintaining the presentation of outliers.

In the Value Distribution View, we support visualization of eight attributes in the same window. This number of attribute columns is usually sufficient as the visual capacity of human is limited to about three to seven objects [SK12]. For datasets with more attributes, users can use the scroll bar to inspect all the attributes. Thus, the cardinality of total attributes in the datasets is not limited. To further improve the scalability, one design choice is to aggregate

attributes with similar distributions together and show hierarchical levels of attributes. Users can obtain an overview of the attribute distributions and further examine the attributes of interest via interactions. When many outliers have similar values of an attribute, the bars in the corresponding column may encounter visual clutter problem. To alleviate it, we offer an option to arrange the bars according to their relative values rather than absolute values. In case the number of outliers grows even larger, we can select the bars of higher interest or cluster the bars to further reduce visual clutter.

In the Algorithm Matrix View, the intersections of outlier sets detected by different algorithms may grow exponentially with the number of algorithms. Therefore, we support users to use the vertical scroll bar to browse the intersections. When the number of algorithms becomes even larger, we can aggregate these intersections according to the number of algorithms in the intersection. Users can expand the rows to further examine the intersections of interest and can also collapse certain rows to filter out the non-interested results. In addition, we also plan to support users to flexibly issue a query to directly obtain the result of any intersections. On the other hand, the number of outliers displayed in a row is limited by the space of the Algorithm Matrix View. To address this issue, we sort the outliers according the number of algorithms that detected them in an descending order from left to right. Therefore, the outliers that are more strongly predicted are displayed by default. We can also aggregate the column of outliers to further improve the scalability.

The Data Table is designed for displaying and accessing raw data. We support interactions such as searching and scrolling. Users can glance at the data by scrolling, and locate the data of interests quickly by searching when the data size increases.

7.2. Generalization

First, Oui can be easily extended to support ordinal or categorical attributes. For ordinal attributes, we can calculate the universal distances for both numerical attributes and ordinal attributes simultaneously. For categorical attributes, we can use the one-hot encoding to represent categorical attribute values. Then, we can calculate the distances between data points using these vectors.

Although Oui mainly focuses on outlier interpretation, it can also be applied to data cleaning to improve the data quality. In machine learning area, the quality of training data has strong influence on algorithm developments. We can use Oui to identify the noise in input training datasets and remove inappropriate data points. Then we can feed the processed data into downstream machine learning models to improve model performances.

7.3. Target Users

We consider our target users as the data analysts from various domains who need to identify the domain-specific outliers based on their specific requirements. We assume that they have sufficient domain knowledge and basic skills on data analytics and understanding common visualization such as scatter plots and violin plots. By navigating through different views in Oui and exploring the data of interests, we aim to facilitate them to further examine the detected outliers and interpret the defining characteristics via our system to make decisions.

8. Reflections on Design

Switching between local and global contexts. When interpreting outliers, users usually need to switch between local and global contexts. For example, a user's exploration often starts with obtaining an overview on outlier detection results and the differences between various algorithms. Afterwards, they need to drill down to a set of interested outliers to understand how they deviate from the normal data. And users may also return to a global perspective to examine normal data distribution or to explore other outliers. This requires the system to support a smooth navigation through different views to enable such exploration processes. In our system, all the views are linked together so that users can select an outlier from the Data Overview, observing which algorithms detect this data instance as an outlier from the Algorithm Matrix View, and identifying its defining characteristics from the Value Distribution View. We suggest that following work may also support users to switch between local and global contexts smoothly.

Enabling flexible and effective comparison. As different algorithms detect outliers using different assumptions, they usually capture diverse sets of outliers. A visualization system on outlier interpretation should enable a flexible and effective comparison to understanding the characteristics of both outliers and algorithms For example, in our Algorithm Matrix View, we utilize the algorithm combination glyph to enable users quickly identifying the common outlier groups detected by multiple algorithms.

Scalability. Scalability is a major concern when designing outlier interpretation system. As the data size increases, the number of detected outliers also grows, which may impose crucial challenges to the design. We have adopted different methods such as sampling, aggregation, and interactions, to improve scalability. We suggest future research on outlier interpretation needs to consider the scalability problem carefully before designing the system.

9. Conclusion and Future Work

In this paper, we have presented Oui, an interactive visualization system that helps users interpret and justify the outliers that are automatically detected by different outlier detection algorithms. Oui provides an overview and an algorithm matrix view to support the browsing and exploration of all outliers. The system also helps users interpret the outliers by offering the detailed information of each attribute. A case study with real-world data and a user study demonstrate the effectiveness of Oui in outlier interpretation.

Oui has many promising directions in the future research. First, we aim to further analyze in which subspaces the outliers are most separated from the normal data for better interpretation. Second, we want to enhance the scalability of our system for larger datasets by integrating the methods we have discussed in this paper. In addition, we plan to further evaluate the effectiveness of our system in real-world application scenarios and conduct more comprehensive user studies such as engaging more participants.

10. Acknowledgments

The authors would like to thank all the anonymous reviewers for their valuable comments and constructive suggestions. This research was supported in part by Hong Kong Theme-based Research Scheme grant T41-709/17N.

References

[BCS96] BUJA A., COOK D., SWAYNE D. F.: Interactive high-dimensional data visualization. *Journal of Computational and Graphical Statistics 5*, 1 (1996), 78–99. 3, 4

[BG16] BUCZAK A. L., GUVEN E.: A survey of data mining and machine learning methods for cyber security intrusion detection. vol. 18, IEEE, pp. 1153–1176. 2

[BKNS00] BREUNIG M. M., KRIEGEL H.-P., NG R. T., SANDER J.: Lof: identifying density-based local outliers. In *Proceedings of SIGMOD International Conference on Management of Datad* (2000), vol. 29, pp. 93–104. 2, 3

[BL74] BARNETT V., LEWIS T.: *Outliers in Statistical Data*. 1974. 2

[Bre01] BREIMAN L.: Random forests. *Machine learning 45*, 1 (2001), 5–32. 3

[CBK09] CHANDOLA V., BANERJEE A., KUMAR V.: Anomaly detection: A survey. *ACM Computing Surveys 41*, 3 (2009), 15. 2

[CLGD18] CAO N., LIN Y.-R., GOTZ D., DU F.: Z-glyph: Visualizing outliers in multivariate data. *IEEE Trans. on Visualization and Computer Graphics 17*, 1 (2018), 22–40. 3

[CSL*16] CAO N., SHI C., LIN S., LU J., LIN Y.-R., LIN C.-Y.: Targetvue: Visual analysis of anomalous user behaviors in online communication systems. *IEEE Trans. on Visualization and Computer Graphics 22*, 1 (2016), 280–289. 3

[CVW11] CLAESSEN J. H., VAN WIJK J. J.: Flexible linked axes for multivariate data visualization. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2310–2316. 4

[Dav91] DAVISON M. L.: Multidimensional scaling. 3, 4

[Den18] DENG H.: Interpreting tree ensembles with intrees. *International Journal of Data Science and Analytics* (2018), 1–11. 3

[DMAN13] DANG X. H., MICENKOVÁ B., ASSENT I., NG R. T.: Local outlier detection with interpretation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2013), pp. 304–320. 2, 4

[FAP*17] FU Y., AGGARWAL C., PARTHASARATHY S., TURAGA D. S., XIONG H.: Remix: Automated exploration for interactive outlier detection. In *Proceedings of SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017), pp. 827–835. 4

[Foo18] FOORTHUIS R.: A typology of data anomalies. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* (2018), pp. 26–38. 2

[Fri01] FRIEDMAN J. H.: Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232. 3

[GD12] GOLDSTEIN M., DENGEL A.: Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *Poster and Demo Track of the German Conference on Artificial Intelligence* (2012), 59–63. 2

[GU16] GOLDSTEIN M., UCHIDA S.: A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one 11*, 4 (2016), e0152173. 2, 4

[HA04] HODGE V., AUSTIN J.: A survey of outlier detection methodologies. *Artificial Intelligence Review 22*, 2 (2004), 85–126. 2

[HHWB02] HAWKINS S., HE H., WILLIAMS G., BAXTER R.: Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery* (2002), pp. 170–180. 2

[HN98] HINTZE J. L., NELSON R. D.: Violin plots: a box plot-density trace synergism. *The American Statistician 52*, 2 (1998), 181–184. 6

[HPK11] HAN J., PEI J., KAMBER M.: *Data mining: concepts and techniques*. 2011. 2

[Ins85] INSELBERG A.: The plane with parallel coordinates. *The Visual Computer 1*, 2 (1985), 69–91. 3, 4

[JLMdO11] JACKOWETZ N., LI E., MIRA DE ORDUÑA R.: Sulphur dioxide content of wines: the role of winemaking and carbonyl compounds. *Research Focus 3* (2011), 1–7. 7

[Jol11] JOLLIFFE I.: Principal component analysis. In *International Encyclopedia of Statistical Science*. Springer, 2011, pp. 1094–1096. 3

[Kim15] KIM B.: *Interactive and interpretable machine learning models for human machine collaboration*. PhD thesis, Massachusetts Institute of Technology, 2015. 3

[KKSZ11] KRIEGEL H.-P., KROGER P., SCHUBERT E., ZIMEK A.: Interpreting and unifying outlier scores. In *Proceedings of SIAM International Conference on Data Mining* (2011), pp. 13–24. 3

[KKZ10] KRIEGEL H.-P., KRÖGER P., ZIMEK A.: Outlier detection techniques. *Tutorial of SIGKDD International Conference on Knowledge Discovery and Data Mining* (2010). 4

[KN99] KNORR E. M., NG R. T.: Finding intensional knowledge of distance-based outliers. In *Proceedings of International Conference on Very Large Data Bases* (1999), pp. 211–222. 2, 3

[KPN16] KRAUSE J., PERER A., NG K.: Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (2016), ACM, pp. 5686–5697. 3

[KSZ08] KRIEGEL H., SCHUBERT M., ZIMEK A.: Angle-based outlier detection in high-dimensional data. In *Proceedings of SIGKDD International Conference on Knowledge Discovery and Data Mining* (2008), pp. 444–452. 2

[Leo98] LEONG T. Y.: Multiple perspective dynamic decision making. *Artificial Intelligence 105*, 1-2 (1998), 209–261. 3

[LGG*18] LIN H., GAO S., GOTZ D., DU F., HE J., CAO N.: Rclens: Interactive rare category exploration and identification. *IEEE Trans. on Visualization and Computer Graphics 24*, 7 (2018), 2223–2237. 3

[LGS*14] LEX A., GEHLENBORG N., STROBELT H., VUILLEMOT R., PFISTER H.: Upset: visualization of intersecting sets. *IEEE Trans. on Visualization and Computer Graphics 20*, 12 (2014), 1983–1992. 5

[Lip16] LIPTON Z. C.: The mythos of model interpretability. *arXiv preprint arXiv:1606.03490* (2016). 3

[LSH17] LIU N., SHIN D., HU X.: Contextual outlier interpretation. In *Proceedings of International Joint Conference on Artificial Intelligence* (2017), pp. 2461–2467. 3, 4

[LTZ08] LIU F. T., TING K. M., ZHOU Z.-H.: Isolation forest. In *IEEE International Conference on Data Mining* (2008), pp. 413–422. 2

[LXL*18] LIU S., XIAO J., LIU J., WANG X., WU J., ZHU J.: Visual diagnosis of tree boosting methods. *IEEE transactions on visualization and computer graphics 24*, 1 (2018), 163–173. 9

[MH08] MAATEN L. V. D., HINTON G.: Visualizing data using t-sne. *Journal of Machine Learning Research* (2008), 2579–2605. 4

[Num18] NUMBEO: Numbeo. https://www.numbeo.com/quality-of-life/, 2018. 8

[PLSG10] PHUA C., LEE V. C. S., SMITH-MILES K., GAYLER R. W.: A comprehensive survey of data mining-based fraud detection research. *CoRR abs/1009.6119* (2010). arXiv:1009.6119. 2

[RRS00] RAMASWAMY S., RASTOGI R., SHIM K.: Efficient algorithms for mining outliers from large data sets. In *Proceedings of SIGMOD International Conference on Management of Data* (2000), vol. 29, pp. 427–438. 2

[RSG16] RIBEIRO M. T., SINGH S., GUESTRIN C.: Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (2016), ACM, pp. 1135–1144. 3

[SFP*07] SCHETININ V., FIELDSEND J. E., PARTRIDGE D., COATS T. J., KRZANOWSKI W. J., EVERSON R. M., BAILEY T. C., HERNANDEZ A.: Confident interpretation of bayesian decision tree ensembles for clinical applications. *IEEE Transactions on Information Technology in Biomedicine 11*, 3 (2007), 312–319. 3

[SK12] SØRENSEN T. A., KYLLINGSBÆK S.: Short-term storage capacity for visual objects depends on expertise. *Acta Psychologica 140*, 2 (2012), 158–163. 9

[SKV10] STRINGHINI G., KRUEGEL C., VIGNA G.: Detecting spammers on social networks. In *Proceedings of Annual Computer Security Applications Conference* (2010), pp. 1–9. 2

[SSWB00] SCHÖLKOPF B., SMOLA A. J., WILLIAMSON R. C., BARTLETT P. L.: New support vector algorithms. *Neural Computation 12*, 5 (2000), 1207–1245. 2

[UCI18] UCI: Wine quality data set. https://archive.ics.uci.edu/ml/datasets/wine+quality, 2018. 6

[WMCW02] WONG W.-K., MOORE A., COOPER G., WAGNER M.: Rule-based anomaly pattern detection for detecting disease outbreaks. In *Proc. of National Conference on Artificial Intelligence* (2002), pp. 217–223. 2

[XXM*19] XU K., XIA M., MU X., WANG Y., CAO N.: Ensemblelens: Ensemble-based visual exploration of anomaly detection algorithms with multidimensional data. *IEEE Trans. on Visualization and Computer Graphics 25*, 1 (Jan 2019), 109–119. doi:10.1109/TVCG.2018.2864825. 3

[ZCS14] ZIMEK A., CAMPELLO R. J., SANDER J.: Ensembles for unsupervised outlier detection: challenges and research questions a position paper. *SIGKDD Explorations 15*, 1 (2014), 11–22. 2, 4

[ZCW*14] ZHAO J., CAO N., WEN Z., SONG Y., LIN Y.-R., COLLINS C.: # fluxflow: Visual analysis of anomalous information spreading on social media. vol. 20, IEEE, pp. 1773–1782. 3

[ZSK12] ZIMEK A., SCHUBERT E., KRIEGEL H.-P.: A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining 5*, 5 (2012), 363–387. 2