Data Mining and Knowledge Discovery
https://doi.org/10.1007/s10618-019-00661-z

On normalization and algorithm selection for unsupervised outlier detection

Sevvandi Kandanaarachchi[1] · Mario A. Muñoz[2] · Rob J. Hyndman[1] · Kate Smith-Miles[2]

Received: 12 September 2018 / Accepted: 22 October 2019

Abstract
This paper demonstrates that the performance of various outlier detection methods is sensitive to both the characteristics of the dataset, and the data normalization scheme employed. To understand these dependencies, we formally prove that normalization affects the nearest neighbor structure, and density of the dataset; hence, affecting which observations could be considered outliers. Then, we perform an instance space analysis of combinations of normalization and detection methods. Such analysis enables the visualization of the strengths and weaknesses of these combinations. Moreover, we gain insights into which method combination might obtain the best performance for a given dataset.

Keywords Unsupervised outlier detection · Effect of normalization on outlier detection · Algorithm selection problem for outlier detection · Instance space analysis · Instance space analysis for outlier detection

1 Introduction

An increasingly important challenge in a data-rich world is to efficiently detect outliers that deviate from regular patterns in a dataset. The significance of detecting outliers with high accuracy, minimizing costly false positives and dangerous false negatives, is clear when we consider a few socially critical examples of outliers: fraudulent credit

Responsible editor: Srinivasan Parthasarathy.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10618-019-00661-z) contains supplementary material, which is available to authorized users.

✉ Sevvandi Kandanaarachchi
sevvandi.kandanaarachchi@monash.edu

1 Monash University, Clayton, VIC, Australia

2 School of Mathematics and Statistics, The University of Melbourne, Parkville, VIC 3010, Australia

Published online: 21 November 2019

card transactions among billions of legitimate ones, fetal anomalies in pregnancies, chromosomal anomalies in tumors, emerging terrorist plots in social media, and early signs of stock market crashes.

There are many outlier detection methods already available in the literature, with new ones emerging at a steady rate (Zimek et al. 2012). The diversity of applications makes it unlikely that a single method will out-perform all others in all scenarios (Wolpert et al. 1995; Wolpert and Macready 1997; Culberson 1998; Ho and Pepyne 2002; Igel and Toussaint 2005). As such, it is advantageous to know their strengths and weaknesses, and how specific properties of a dataset might affect their performance. What properties would enable a given method to perform well on one dataset but poorly on another? How sensitive are the existing methods to variations in dataset characteristics? How can we objectively evaluate a portfolio of detection methods to learn these relationships? Given a problem, can we learn to predict the best-suited detection method(s)? And given that normalization of a dataset is a typical pre-processing step adopted by most detection applications, what impact does a normalization scheme have on detection accuracy? These are some of the questions that motivate this work.

When evaluating outlier detection methods, we must consider the definition of an outlier used by both the algorithm and a human who may have labeled training data. Generically, Hawkins (1980) defines an outlier *as an observation which deviates so much from other observations as to arouse suspicion it was generated by a different mechanism*. Barnett and Lewis (1974) define an outlier as *an observation (or a subset of observations) which appears to be inconsistent with the remainder of that set of data*. Both these definitions indicate that outliers are quite different from non-outlying observations. Barnett and Lewis (1974) also note that it is a *a matter of subjective judgment on the part of the observer whether or not some observation is picked out for scrutiny*. The subjectivity of outlier detection is not only due to human judgment but extends to differences in how each method defines an outlier. Indeed, disagreements between methods on the location of outliers may be due to differences on definitions, whether they are related to nearest neighbor distances, density arguments or other quantitative metrics. For example, Fig. 1 illustrates the lack of consensus between three popular methods, namely KNN (Ramaswamy et al. 2000), LOF (Breunig et al. 2000) and COF (Tang et al. 2002). As such, to select the most suitable method, we should exploit the knowledge gained from analyzing both the dataset characteristics and the algorithm's definition of an outlier.

Evaluation of unsupervised outlier detection methods has received growing attention in recent years. Campos et al. (2016) conducted an experimental evaluation of 12 methods based on nearest neighbors using the ELKI software suite (Achtert et al. 2008). While all the methods considered used a similar definition of an outlier, this study contributed a repository of around 1000 benchmark datasets generated by modifying 23 source ones that can be used for further analysis. It is common practice to test detection methods on datasets with known ground truth labels, for example, classification datasets where observations of the minority class have been down-sampled. We extend their approach to dataset generation for our study. Goldstein and Uchida (2016) conducted a comparative evaluation of 19 methods, which fall into three categories, namely nearest neighbor based, clustering-based, and based on other algorithms such as one class SVM and robust PCA. They have used 10 datasets for their evaluation.

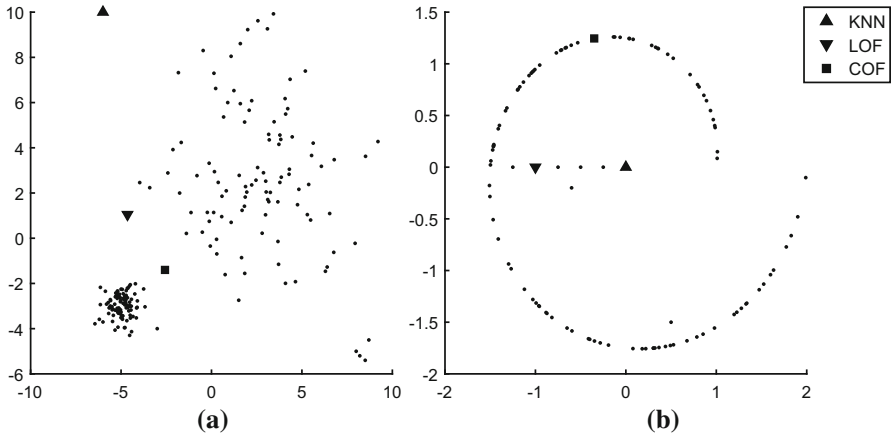

Fig. 1 First outlier detected in two different datasets by three different methods: KNN (▲), LOF (▼) and COF (■)

Their algorithms were released on the RapidMiner data mining software. Emmott et al. (2013) studied the construction of benchmark datasets for outlier detection using classification datasets. Emmott et al. (2015) conducted a meta-analysis of 12 methods, which fall into four categories; namely, nearest neighbors based, density-based, model-based or projection-based. However, none of these studies address the critical algorithm selection problem for outlier detection: given a dataset, which outlier detection method(s) is expected to give the best performance, and why? This is one of the main contributions of our work.

The algorithm selection problem has been extensively studied in various research communities (Rice 1976; Smith-Miles 2009) for challenges such as meta-learning in machine learning (Brazdil et al. 2008), black-box optimization (Bischl et al. 2012), and algorithm portfolio selection in SAT solvers (Leyton-Brown et al. 2003). Smith-Miles and co-authors have extended the Rice (1976) framework for algorithm selection and developed a methodology known as *instance space analysis* to visualize and gain insights into the strengths and weaknesses of algorithms across the broadest possible set of test instances, rather than a finite set of common benchmarks (Smith-Miles et al. 2014; Smith-Miles and Bowly 2015; Muñoz et al. 2018). In Kandanaarachchi et al. (2019a), we extended the study by Campos et al. (2016) to explore how their considered outlier detection methods, with a fixed normalization scheme, could be analyzed from an instance space perspective. In the current paper, we augment this proof-of-concept study, by considering a larger collection of outlier methods, using the instance space framework to gain an understanding of the strengths and weaknesses of these algorithms. Moreover, we recognize that an algorithm often includes the normalization of the dataset. The complex relationship between normalization and detection methods create a combined effect that is not well understood. Thus, a further contribution of this paper is to comprehensively tackle the algorithm selection problem for outlier detection in the context of the often-overlooked impact of normalization.

Routinely, the min–max normalization method, which bounds each variable to the interval [0, 1], is used in outlier detection (Campos et al. 2016; Goldstein and Uchida 2016). However, there are alternative methods. To the best of our knowledge, no previous study has focused on the effect that the choice of normalization method has on outlier detection performance. As such, we explore this relationship and show that the performance of outlier methods can change significantly depending on the normalization method. This is a further contribution of our work. In addition, our R package *outselect* (Kandanaarachchi 2018) contains dataset features and outlier method performance values for all datasets and can be used to predict suitable outlier detection methods for a new dataset, and plot it in the instance space. Furthermore, we make available a repository of over 12,000 datasets (Kandanaarachchi et al. 2019b), which is generated from approximately 200 source datasets, providing a comprehensive basis for future evaluation of outlier detection methods. Moreover, the code used for instance space analysis is available at Muñoz (2019).

This paper is organized as follows. We start by investigating the impact of normalization on outlier detection methods in Sect. 2. First, we present mathematical arguments in Sect. 2.1 to prove how normalization changes the nearest neighbors and densities of the data, and hence why we expect that the impact of normalization can be significant depending on the definition of an outlier adopted by an algorithm. In Sect. 2.2, we present experimental evidence of this theoretical sensitivity through the analysis of a set of over 12,000 datasets. We show that combinations of normalization and detection methods have variable performance across datasets, suggesting that some combinations can exploit the properties of the datasets, while others cannot. This experimental and theoretical evidence then motivates the remainder of the paper, where we adapt instance space analysis to gain insights into the strengths and weaknesses of normalization and outlier detection method combinations. Section 3 first describes the methodological framework for the algorithm selection problem and instance space analysis introduced by Smith-Miles et al. (2014). This section then discusses a novel set of features that capture properties of outlier detection datasets and shows how these features can be used to predict the performance of a detection method. The instance space is then constructed, and objective assessment of outlier detection method strengths and weaknesses is presented in the form of footprint analysis. The instance space shows that the datasets considered in this study are more diverse and comprehensive than previous studies, and suitable outlier detection methods are identified for various parts of the instance space. Finally, in Sect. 4 we present the conclusions of this work and future avenues of research.

2 Impact of normalization on outlier detection

One of the main pre-processing steps for many statistical learning tasks is normalizing the data. Normalization[1] is especially important in unsupervised outlier detection because different attributes of a dataset may have different measurement units. In fact,

[1] Generally normalization refers to scaling each attribute to [0, 1] while standardization refers to scaling each attribute to $\mathcal{N}(0, 1)$. For the sake of simplicity, and without loss of generality, we use the term normalization to refer to both re-scalings in this paper.

Campos et al. (2016) show that outlier detection methods on normalized datasets give higher performance values compared to the performance on un-normalized datasets. Even though there seems to be a consensus in the research community that normalization is a necessary pre-processing step for outlier detection, the effects of different normalization methods on outlier detection have not been studied to the best of our knowledge. As such, we investigate the effect of four normalization/standardization methods of outlier detection.

Minimum and maximum normalization (Min–Max)

Each column x is transformed to $\frac{x-\min(x)}{\max(x)-\min(x)}$ where $\min(x)$ and $\max(x)$ are the minimum and maximum values of x respectively.

Mean and standard deviation normalization (Mean–SD)

Each column x is transformed to $\frac{x-\text{mean}(x)}{\text{sd}(x)}$, where $\text{mean}(x)$ and $\text{sd}(x)$ are the mean and standard deviation values of x respectively.

Median and the IQR normalization (Median–IQR)

Each column x is transformed to $\frac{x-\text{median}(x)}{\text{IQR}(x)}$, where $\text{median}(x)$ and $\text{IQR}(x)$ are the median and IQR of x respectively.

Median and median absolute deviation normalization (Median–MAD)

Here $\text{MAD}(x) = \text{median}(|x - \text{median}(x)|)$ and each column x is transformed to $\frac{x-\text{median}(x)}{\text{MAD}(x)}$.

We note that Min–Max and Mean–SD are influenced by outliers while Median–IQR and Median–MAD are more robust to outliers. A detailed account of the usage of robust statistics in outlier detection is covered by Rousseeuw and Hubert (2017).

Generally, normalization scales axes differently causing some to compress and others to expand, thus changing the nearest neighbor structure. As nearest neighbor distances play an important role in many outlier detection techniques, such normalization impacts outlier detection method results, as will be explained theoretically and then demonstrated experimentally in the following sections.

2.1 Mathematical analysis

In this section we look at the effect of normalization on a dataset from a mathematical view-point. Let D be a dataset containing N observations and d numerical attributes. Let us denote the ith observation by $\boldsymbol{x}_i$ where $\boldsymbol{x}_i \in \mathbb{R}^d$. The four normalization techniques described above can be written as

$$\boldsymbol{x}_i^* = S^{-1}(\boldsymbol{x}_i - \boldsymbol{m}). \tag{1}$$

Here $\boldsymbol{x}_i^*$ is the normalized observation, $\boldsymbol{m}$ is either the minimum, mean or median of the data and S is a diagonal matrix containing column-wise range, standard deviation, IQR or MAD. Let $S = \text{diag}(s_1, s_2, s_3, \ldots, s_d)$.

Let dist $(\boldsymbol{x}, \boldsymbol{y})$ denote the Euclidean distance between the two points $\boldsymbol{x}$ and $\boldsymbol{y}$, i.e. dist $(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$, where we use the L_2 norm. So we have

$$\text{dist}\left(\boldsymbol{x}_i^*, \boldsymbol{x}_j^*\right) = \left\| S^{-1}\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)\right\|, \tag{2}$$

giving us

$$\begin{aligned}\text{dist}^2\left(\boldsymbol{x}_i^*, \boldsymbol{x}_j^*\right) &= \left\langle S^{-1}\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right), S^{-1}\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)\right\rangle, \\ &= \left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)^\top S^{-2}\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right), \\ &= \sum_{k=1}^{d} \frac{1}{s_k^2}\left(x_{ik} - x_{jk}\right)^2,\end{aligned} \tag{3}$$

where x_{ik} is the kth coordinate of $\boldsymbol{x}_i$. By defining

$$\begin{aligned}\boldsymbol{w} &= \left(\frac{1}{s_1^2}, \frac{1}{s_2^2}, \ldots, \frac{1}{s_d^2}\right)^\top \quad \text{and} \\ \boldsymbol{y}_{ij} &= \left(\left(x_{i1} - x_{j1}\right)^2, \left(x_{i2} - x_{j2}\right)^2, \ldots, \left(x_{id} - x_{jd}\right)^2\right)^\top,\end{aligned} \tag{4}$$

we can write the distance between $\boldsymbol{x}_i^*$ and $\boldsymbol{x}_j^*$ as

$$\text{dist}^2\left(\boldsymbol{x}_i^*, \boldsymbol{x}_j^*\right) = \left\langle \boldsymbol{w}, \boldsymbol{y}_{ij}\right\rangle. \tag{5}$$

The advantage of this representation is that we can explore the effect of normalization without restricting ourselves to the normalized space. That is, suppose we want to compare two normalization methods given by matrices S_1 and S_2. By working with the corresponding vectors $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ we can stay in the space of $\boldsymbol{y}_{ij}$ for different normalization methods. Here, the space where $\boldsymbol{y}_{ij}$ lives is different from the space of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. From Eq. (4) the components of $\boldsymbol{y}_{ij}$ corresponds to the squared component differences between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. As such the vector $\boldsymbol{y}_{ij}$ cannot contain negative values, i.e. $\boldsymbol{y}_{ij} \in \mathbb{R}^{d+}$ where $\mathbb{R}^{d+}$ is the positive orthant or hyperoctant in $\mathbb{R}^d$. Similarly, $\boldsymbol{w}$ has positive coordinates and $\boldsymbol{w} \in \mathbb{R}^{d+} \backslash \{\boldsymbol{0}\}$.

To understand more about the space of $\boldsymbol{y}_{ij}$, we give it separate notation. Let us denote the space of $\boldsymbol{y}_{ij}$ by Y and the space of observations by O. It is true that Y is isomorphic to $\mathbb{R}^{d+}$ and O to $\mathbb{R}^d$. However, because the original observations in $O \times O$ map to Y in a slightly different way when compared with the standard partitioning of $\mathbb{R}^{d+}$ from $\mathbb{R}^d$, it makes sense to detach Y from $\mathbb{R}^{d+}$ and O from $\mathbb{R}^d$ for a moment. From (4) we have

$$\begin{aligned}\boldsymbol{y}_{ij} &= \left(\left(x_{i1} - x_{j1}\right)^2, \left(x_{i2} - x_{j2}\right)^2, \ldots, \left(x_{id} - x_{jd}\right)^2\right)^\top, \\ &= \left(\left(x_{i1} + \eta_1 - x_{j1} - \eta_1\right)^2, \left(x_{i2} + \eta_2 - x_{j2} - \eta_2\right)^2, \ldots, \left(x_{id} + \eta_d - x_{jd} - \eta_d\right)^2\right)^\top.\end{aligned}$$

As such, if the points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ give rise to $\boldsymbol{y}_{ij}$ so does the points $\boldsymbol{x}_i + \eta$ and $\boldsymbol{x}_j + \eta$ for any $\eta \in \mathbb{R}^d$. Thus, the mapping from $O \times O$ to Y is translation-invariant. This shows that Y is obtained from $O \times O$ in a different way to the standard partitioning of $\mathbb{R}^{d+}$ from $\mathbb{R}^d$. However, we will not use the translation invariant property of Y in the next sections.

2.1.1 Nearest neighbors

Let the kth nearest neighbor of a point $\boldsymbol{x}$ be denoted by $\text{nn}\,(\boldsymbol{x}, k)$ and the kth nearest neighbor distance be denoted by $\text{nnd}\,(\boldsymbol{x}, k)$. With the above notation, let us write down the expression for the nearest neighbor of a point $\boldsymbol{x}_i^*$:

$$\text{nn}\left(\boldsymbol{x}_i^*, 1\right) = \underset{\boldsymbol{x}_j,\, j \neq i}{\text{argmin}} \left(\text{dist}\left(\boldsymbol{x}_i^*, \boldsymbol{x}_j^*\right)\right) \tag{6}$$

Let $A_{i1} = \{1, 2, \ldots, i-1, i+1, \ldots, N\}$. Then using (5) we can re-write this as

$$\begin{aligned} \text{nn}\left(\boldsymbol{x}_i^*, 1\right) &= \underset{\boldsymbol{x}_j,\, j \in A_{i1}}{\text{argmin}} \left(\text{dist}\left(\boldsymbol{x}_i^*, \boldsymbol{x}_j^*\right)^2\right) \\ &= \underset{\boldsymbol{x}_j,\, j \in A_{i1}}{\text{argmin}} \left\langle \boldsymbol{w}, \boldsymbol{y}_{ij} \right\rangle \end{aligned} \tag{7}$$

If $\boldsymbol{x}_{l_1}^*$ is the nearest neighbor of $\boldsymbol{x}_i^*$ we define A_{i2} as $A_{i2} = A_{i1} \backslash l_1$, giving us

$$\text{nn}\left(\boldsymbol{x}_i^*, 2\right) = \underset{\boldsymbol{x}_j,\, j \in A_{i2}}{\text{argmin}} \left\langle \boldsymbol{w}, \boldsymbol{y}_{ij} \right\rangle.$$

Similarly we can write

$$\text{nn}\left(\boldsymbol{x}_i^*, k\right) = \underset{\boldsymbol{x}_j,\, j \in A_{ik}}{\text{argmin}} \left\langle \boldsymbol{w}, \boldsymbol{y}_{ij} \right\rangle, \tag{8}$$

where $\boldsymbol{x}_{\ell_{k-1}}^*$ is the $(k-1)^{\text{st}}$ nearest neighbor of $\boldsymbol{x}_i^*$ and $A_{ik} = A_{i(k-1)} \backslash \ell_{k-1}$. Proceeding in a similar way, the kth nearest neighbor distance can be written as

$$\text{nnd}\left(\boldsymbol{x}_i^*, k\right) = \min_{j \in A_{ik}} \sqrt{\left\langle \boldsymbol{w}, \boldsymbol{y}_{ij} \right\rangle}. \tag{9}$$

As the outlier detection method KNN declares the points with the highest k-nearest neighbor distance as outliers we write an expression for the point with the highest knn distance:

$$\text{point with highest knn distance} = \underset{i}{\text{argmax}} \left(\text{nnd}\left(\boldsymbol{x}_i^*, k\right)\right) = \underset{i}{\text{argmax}} \left(\min_{j \in A_{ik}} \sqrt{\left\langle \boldsymbol{w}, \boldsymbol{y}_{ij} \right\rangle}\right). \tag{10}$$

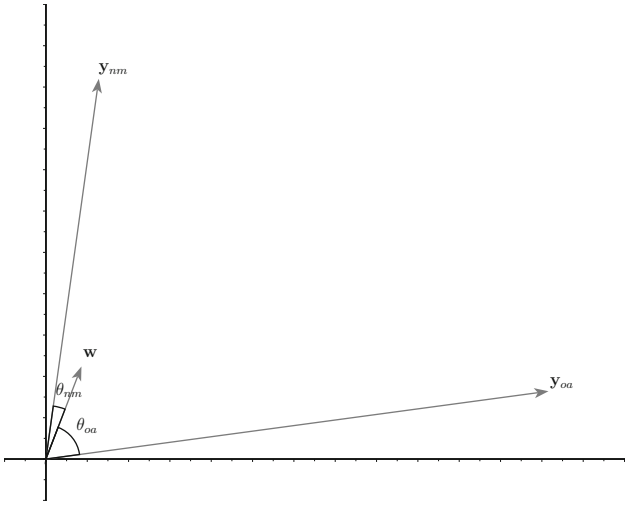

Fig. 2 The vectors $\boldsymbol{y}_{oa}$, $\boldsymbol{y}_{nm}$, $\boldsymbol{w}$ with angles θ_{oa} and θ_{nm}

From (10) we can see that $\boldsymbol{w}$ has a role in determining the data-point with the highest knn distance. A different $\boldsymbol{w}$ may produce a different data-point having the highest knn distance. Therefore, the method of normalization plays an important role in nearest neighbor computations.

Proposition 2.1 *Let $\boldsymbol{x}_o$ be an outlier and $\boldsymbol{x}_n$ a non-outlier. Let $\boldsymbol{x}_a$ and $\boldsymbol{x}_m$ be $\boldsymbol{x}_o$ and $\boldsymbol{x}_n$'s respective k-nearest neighbors according to the normalization scheme defined by $\boldsymbol{w}$. Let θ_{oa} and θ_{nm} be the angles that $\boldsymbol{y}_{oa}, \boldsymbol{y}_{nm} \in Y$ make with $\boldsymbol{w}$. If*

$$\frac{\left\| \boldsymbol{y}_{oa} \right\|}{\left\| \boldsymbol{y}_{nm} \right\|} < \frac{\cos \theta_{nm}}{\cos \theta_{oa}},$$

then

$$nnd\left(\boldsymbol{x}_o^*, k\right) < nnd\left(\boldsymbol{x}_n^*, k\right),$$

where $\boldsymbol{x}_o^*$ and $\boldsymbol{x}_n^*$ are the normalized coordinates of $\boldsymbol{x}_o$ and $\boldsymbol{x}_n$ according to $\boldsymbol{w}$. Thus a non-outlier has a higher knn distance that an outlier with respect to $\boldsymbol{w}$.

Proof From (9) the knn distance of $\boldsymbol{x}_o^*$ is

$$\begin{aligned} \text{nnd}\left(\boldsymbol{x}_o^*, k\right) &= \min_{j \in A_{ok}} \sqrt{\langle \boldsymbol{w}, \boldsymbol{y}_{oj} \rangle}, \\ &= \sqrt{\langle \boldsymbol{w}, \boldsymbol{y}_{oa} \rangle}, \end{aligned} \tag{11}$$

as $\boldsymbol{x}_a$ is the k-nearest neighbour of $\boldsymbol{x}_o$. Similarly

$$\text{nnd}\left(\boldsymbol{x}_n^*, k\right) = \min_{j \in A_{nk}} \sqrt{\langle \boldsymbol{w}, \boldsymbol{y}_{nj} \rangle} = \sqrt{\langle \boldsymbol{w}, \boldsymbol{y}_{nm} \rangle}. \tag{12}$$

From Eq. (11) we have

$$\text{nnd}\left(\boldsymbol{x}_o^*, k\right)^2 = \langle \boldsymbol{w}, \boldsymbol{y}_{oa} \rangle = \|\boldsymbol{w}\| \left\|\boldsymbol{y}_{oa}\right\| \cos\theta_{oa}, \tag{13}$$

and from Eq. (12) we have

$$\text{nnd}\left(\boldsymbol{x}_n^*, k\right)^2 = \langle \boldsymbol{w}, \boldsymbol{y}_{nm} \rangle = \|\boldsymbol{w}\| \left\|\boldsymbol{y}_{nm}\right\| \cos\theta_{nm}. \tag{14}$$

Dividing Eq. (13) from (14) we obtain

$$\frac{\text{nnd}\left(\boldsymbol{x}_o^*, k\right)^2}{\text{nnd}\left(\boldsymbol{x}_n^*, k\right)^2} = \frac{\left\|\boldsymbol{y}_{oa}\right\| \cos\theta_{oa}}{\left\|\boldsymbol{y}_{nm}\right\| \cos\theta_{nm}} < 1 \tag{15}$$

by the condition of the proposition. This makes $\text{nnd}\left(\boldsymbol{x}_o^*, k\right) < \text{nnd}\left(\boldsymbol{x}_n^*, k\right)$. □

As illustrated in Fig. 2 the angle between $\boldsymbol{w}$ and $\boldsymbol{y}_{nm}$ has an effect in the ordering of k-nearest neighbor distances. Thus, $\boldsymbol{w}$ can mask outliers and favor non-outliers, reducing the performance of a detection method.

2.1.2 Density computations

Density can be defined as the number of data-points in a unit ball. Using this definition, the density of point $\boldsymbol{x}_i^*$ is

$$\text{density}(\boldsymbol{x}_i^*) = \#\left\{\boldsymbol{x}_j^* : \left\|\boldsymbol{x}_j^* - \boldsymbol{x}_i^*\right\| \leq 1\right\} \tag{16}$$

where # denotes the number of points satisfying the given condition. Using the notation defined in (4) and (5) we can rewrite this as

$$\begin{aligned}\text{density}\left(\boldsymbol{x}_i^*\right) &= \#\left\{\boldsymbol{x}_j^* : \left\|\boldsymbol{x}_j^* - \boldsymbol{x}_i^*\right\|^2 \leq 1\right\} \\ &= \#\left\{\boldsymbol{x}_j^* : \text{dist}\left(\boldsymbol{x}_j^*, \boldsymbol{x}_i^*\right)^2 \leq 1\right\} \\ &= \#\left\{\boldsymbol{y}_{ij} : \langle \boldsymbol{w}, \boldsymbol{y}_{ij} \rangle \leq 1\right\}\end{aligned} \tag{17}$$

Again we see that vector $\boldsymbol{w}$ which comes from the method of normalization plays a role in determining the density of data-points. As many outlier detection methods are based on density estimates, we see that normalization affects density-based outlier detection methods as well.

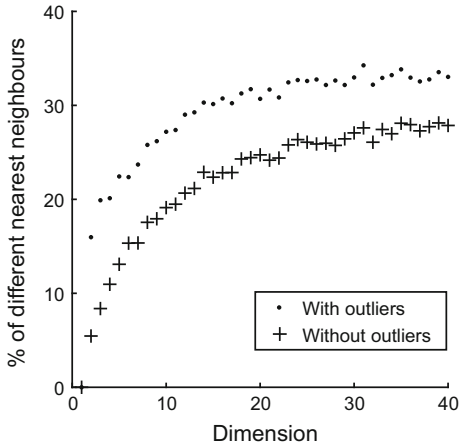

Fig. 3 Percentage of observations that do not have the same nearest neighbour after normalizing using the above four methods

We now show that this theoretical sensitivity is observed when using common benchmark datasets and that the performance of outlier detection methods depends on normalization as well as characteristics of the datasets.

2.2 Experimental evidence of impact of normalization

As an initial experiment, we generate a dataset of 100 data points of dimension d from the uniform distribution, where d ranges from 1 to 40. Next, the data were scaled using the four normalization methods. For each normalized dataset, the nearest neighbor was computed for each observation, and we calculate the percentage of observations that do not have the same nearest neighbor across the whole dataset. This percentage is the quantity of interest. For each dimension d, we run the experiment for 100 repetitions and compute this percentage. For a second experiment, we add one outlier to the dataset and repeat the process. The graphs in Fig. 3 show the average percentage of observations that do not have the same nearest neighbor from different normalizations with and without the outlier added.

We observe that in both the above scenarios the percentage of observations that have different nearest neighbors due to normalization increases with dimension. For the case with no outliers, this percentage changes from 5 to 25% as the dimension changes from 2 to 20. That is for a 20-dimensional dataset without outliers, the nearest neighbors of 25% of the data depend on the method of normalization. Similarly, for a 20-dimensional dataset with one outlier, the nearest neighbors of 30% of the data depend on the normalization method. This observation has the important implication that as the dimensionality of the dataset increases while keeping the number of observations constant, the nearest neighbors of a data-point are highly sensitive to the method of normalization. Thus, given an outlier detection problem, the normalization method, as well as the outlier detection technique, play an important role. Of course, it is important to validate this hypothesis on other datasets, rather than

randomly generated data, to see if structured data from benchmark datasets is also sensitive to normalization.

In the remainder of this section, we evaluate the impact of normalization on 14 outlier detection methods coupled with the above-mentioned 4 normalization methods, across a set of over 12,000 datasets described below.

2.2.1 Datasets

We generate outlier detection datasets by adopting the approach used in recent studies (Campos et al. 2016; Goldstein and Uchida 2016), which takes a classification dataset and down-samples the minority class to label outliers. Campos et al. (2016) started with 23 datasets, from which different variants are obtained mainly by down-sampling the outlier class at rates 20%, 10%, 5%, 2% and transforming categorical variables to numeric values. This process results in approximately 1000 datasets. Goldstein and Uchida (2016) used 10 datasets for their evaluation study, with some overlap with Campos et al. (2016). In order to obtain a more comprehensive and diverse set of benchmark test datasets, we extend the approach to utilize a set of 170 base classification datasets recently used by Muñoz et al. (2018) obtained primarily from the UCI machine learning repository. These classification datasets were not intended for outlier detection evaluation, and so the following issues need to be addressed to generate meaningful outlier detection benchmarks:

1. Labeling of outliers: since outliers are rare events, its proportion is typically 5% or less for most datasets. In contrast, classification datasets have sometimes more than 2 classes and the proportion of observations belonging to each class is often similar and much larger than 10%.
2. Categorical variables: while some classification algorithms such as random forests and decision trees are capable of handling categorical variables, most outlier detection methods need distances or densities to find outliers, which requires only numerical attributes.
3. Duplicate observations and missing values: the classification datasets contain data challenges that we wish to eliminate at this stage to focus on understanding how the underlying mechanism of outlier detection behaves in the presence of complete data.

Therefore, we modify the 170 classification datasets used in Muñoz et al. (2018) to make them more applicable for outlier detection, as described below:

Down-sampling If a dataset has observations belonging to c classes, then each class, in turn, is designated the outlier-class and observations belonging to that class are down-sampled, while the observations belonging to the other $c - 1$ classes are deemed non-outliers. We conduct the down-sampling such that the percentage of outliers is $p\%$ for $p \in \{2, 5\}$. For a given outlier class and for each value of p, the down-sampling is randomly carried out 10 times. Hence, for a given outlier class there are 20 down-sampled files generated. This procedure is done for all classes in the dataset, e.g. if a base classification dataset has 3 classes, then there are $3 \times 2 \times 10$ down-sampled files generated from that base dataset.

Categorical variables While a range of techniques for transforming categorical variables to numerical variables is available, there is little consensus on which approach is best suited for a given task. For each source down-sampled dataset, we create two versions: one with categorical variables removed, and one with categorical variables converted using the method of inverse data frequency (Campos et al. 2016), which creates a new variable $\mathrm{IDF}(x) = \ln(N/n_x)$ where N is the total number of observations in the dataset and n_x is the number of times the categorical variable takes the value x. IDF maps the rarer values to higher numbers and common values to lower numbers.

Duplicate observations As the nearest neighbor distance for a duplicate observation is zero, this can create division by zero errors causing numerical instability when computing densities and other metrics. As such, we remove duplicate observations from the datasets.

Missing values We use the method in Campos et al. (2016) to treat missing values. For each attribute in each dataset, the number of missing values is computed. If an attribute has less than 10% of missing values, the observations containing the missing values are removed, otherwise, the attribute is removed.

The above procedures were followed on the 170 base classification datasets used in Muñoz et al. (2018). In addition, we augmented our benchmark collection to considering the 1000 datasets used in Campos et al. (2016) and selected the ones with 5% and 2% outliers (but not the ones with 10% and 20% outliers). With the datasets from Campos et al. (2016), Goldstein and Uchida (2016), along with the datasets we prepared from Muñoz et al. (2018), our final set of benchmarks for this experimental study contains approximately 12,200 datasets suitable for outlier detection evaluation.

2.2.2 Outlier detection methods

We investigate 14 outlier detection methods: 12 outlier detection methods studied in Campos et al. (2016), and an ensemble of LOF based outlier methods using the ELKI software suite (Achtert et al. 2008), and isolation forest (iForest) using the R implementation of Liu (2009). The methods are:

1. KNN: K nearest neighbours (Ramaswamy et al. 2000)
2. KNNW: KNN weight (Angiulli and Pizzuti 2002)
3. ODIN: Outlier Detection using In-degree Number (Hautamaki et al. 2004)
4. LOF: Local Outlier Factor (Breunig et al. 2000)
5. Simplified LOF: (Schubert et al. 2014b)
6. COF: Connectivity based Outlier Factor (Tang et al. 2002)
7. INFLO: Influenced Outlierness (Jin et al. 2006)
8. LOOP: Local Outlier Probabilities (Kriegel et al. 2009)
9. LDOF: Local Density based Outlier Factor (Zhang et al. 2009)
10. LDF: Local Density Factor (Latecki et al. 2007)
11. KDEOS: Kernel Density Estimation Outlier Score (Schubert et al. 2014a)
12. FAST ABOD: Fast Angle Based Outlier Detection (ABOD), faster version of ABOD (Kriegel et al. 2008)
13. iForest: Isolation Forest (Liu et al. 2008)

14. An ensemble using LOF, LDOF and LOOP

For a brief description of the above methods we refer the reader to Campos et al. (2016) and for a pictorial explanation of KNN, LOF, COF, INFLO, and LOOP, to Goldstein and Uchida (2016).

2.2.3 Evaluation metric

Unlike classification or regression, outlier detection offers challenges in finding acceptable evaluation metrics. As outliers are rare, even if a method does not detect outliers, it still has a high level of overall accuracy. The lack of a universally accepted evaluation metric is evident from the different standards adopted by different research communities. While it is common for outlier methods to rank observations ordered by the level of *outlierness* (Breunig et al. 2000; Schubert et al. 2014b; Tang et al. 2002; Jin et al. 2006), it is also common for methods to find outliers and declare them as binary output—outlier or not (Hubert and Van der Veeken 2008; Wilkinson 2018; Talagala et al. 2019; Billor et al. 2000). In the first instance, when the observations are ranked, finding the outliers becomes the task of the user as a threshold is needed to separate outliers from non-outliers. While this may be preferred for some applications, others might prefer the second approach where observations are either declared outliers or not.

While there is no single accepted metric to evaluate an outlier detection method, two popular evaluation metrics are 1. the area under the Receiver Operator Characteristic (ROC) curve, and 2. The Precision–Recall (PR) curve. Of these two methods, it is fair to say that the area under the ROC is more widely used than PR curves. Davis and Goadrich (2006) discuss the relationship between ROC and PR curves, and show that for a given dataset if a curve dominates in the ROC space, it also dominates in the PR space. In addition to these two measures, it is also quite common to report false positives and false negatives rather than an uninformative overall accuracy measure. In addition to these, there are also other methods such as precision at n (Craswell 2009), average precision (Zhang and Zhang 2009) and excess-mass and mass–volume curves (Goix 2016).

In our study, we use the area under ROC curve (AUC) as the evaluation metric and define good performance as AUC $\geq$ 0.8 as part of our experimental setting. Alternative performance metrics can be considered, such as Area under the Precision–Recall curve—and we have tested these—but we have found that AUC provides the most discriminating performance metric.

2.2.4 Other experimental settings

Our goal is to demonstrate that we can find regions of "good" performance for different methods, for a given experimental setting, which includes algorithm parameters, an evaluation metric, a definition of good performance and classifiers used for later analysis. Thus we do not expend effort on parameter selection, which is a non-trivial study in itself. In addition, there are many evaluation metrics, definitions of good performance and classifiers that can be used. We fix each of these variables for our experimental

setting, aiming to demonstrate that the instance space methodology can yield useful insights into the relationships observed within a given experimental setting, rather than seeking to generalize these relationships across many possible experimental settings. Naturally, the methodology and analysis can be repeated for any chosen experimental setting. For the analysis in Sect. 3.2 we use random forests and in Sects. 3.4 and 3.5 support vector machines for classification.

All outlier methods apart from iForest use a k-nearest neighborhood giving rise to a common parameter k. Noting that no single k would be applicable for all methods and datasets, we choose a tailored value of k based on the dataset and not on the method. That is, for a given dataset, we choose the same k for all methods as follows:

$$k\,(\text{dataset}) = \min\left(\lfloor N/20 \rfloor\,,\,100\right), \tag{18}$$

where N is the number of observations. Here the maximum of $k = 100$ is a means of limiting the number of computations that can result from a large dataset. The motivation for using $\lfloor N/20 \rfloor$, which is 5% of the number of observations, is explained as follows: As k relates to the the size of the neighborhood, choosing $k = N$ would make outliers and non-outliers indistinguishable for some methods such as KNN and LOF, as we are considering too big neighborhoods. On the other hand choosing $k = 1$ may not be effective as it considers a too small neighborhood and would miss outliers in small clusters. Thus, we need a small neighborhood, which is not very small. While this choice of k may not be optimal for all algorithms, this is our choice of k in our experimental setting.

2.2.5 Difficulty and diversity of instances

In order to understand the difficulty and diversity of instances, we compute scores similar to Campos et al. (2016). For computational simplicity we use the following definitions:

$$\begin{aligned}\text{difficulty}(x) &= 1 - \operatorname*{mean}_{\alpha_i} \text{AUC}\left(x, \alpha_i\right),\\ \text{diversity}(x) &= \operatorname*{sd}_{\alpha_i} \text{AUC}\left(x, \alpha_i\right),\end{aligned}$$

where x is the dataset or instance, α is an algorithm and sd is the standard deviation. Figure 4 shows the difficulty–diversity space with instances colour-coded by the source dataset.

From Fig. 4 we see that there are many instances of high difficulty and high diversity, indicating that some algorithms perform well on these instances. These instances are the most useful for algorithm selection as these are generally hard instances, but some algorithms perform well on these. The other set of instances that are also useful are the low difficulty and high diversity instances. These are generally easy instances, but certain algorithms do not perform well on these. As such, selecting a suitable algorithm is important. The low difficulty and low diversity instances are generally the easy instances for which any algorithm performs reasonably well. While these instances are suitable for outlier detection, they do not lend insights into the strengths

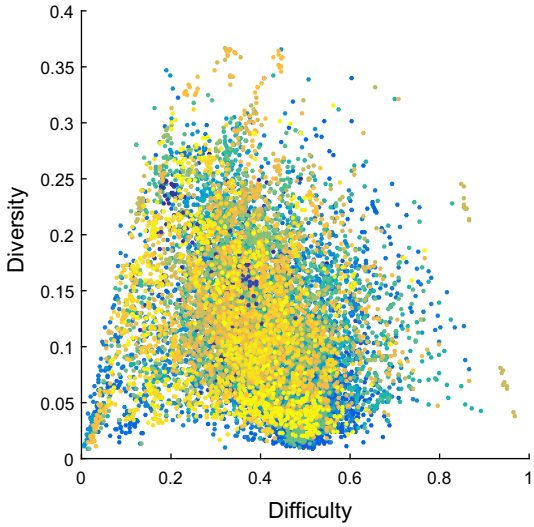

Fig. 4 The difficulty and diversity of instances with each color representing datasets from the same source (Color figure online)

and weaknesses of algorithms. Similarly, the instances with high difficulty and low diversity are the hard instances and they do not differentiate among the algorithms. However, these instances highlight the need for new algorithms to be developed.

In addition, we also see the variety of instances generated from the same source in terms of difficulty and diversity. It is interesting to note that instances from the same source are not always clustered together. Therefore, we believe that the set of instances have sufficient difficulty and diversity to create an interesting instance space and gain insights into the strengths and weaknesses of combinations of outlier detection methods and normalization schemes.

2.3 Hypothesis testing

2.3.1 Does normalizing achieve better results?

To test whether normalization achieves better results we perform Student's *t* tests. We compare the non-normalized performance values with the normalized performance values for each combination of outlier and normalization methods. Table 1 shows the results of this analysis. A + sign signifies that normalization achieves significantly better results at the 5% level, while a − sign denotes significantly worse results. A 0 depicts that there is no significant difference as a result of normalizing.

From Table 1 we see that for all methods apart from iForest and KDEOS, normalization has a positive impact, i.e. there is at least one normalization method that performs significantly better than its non-normalized counterpart. For methods FAST ABOD, INFLO, KNN, KNNW, LOOP, ODIN, and SIMLOF all normalization schemes perform better. The method iForest does not perform distance or density calculations to find outliers and thus normalizing does not achieve a better or worse outcome.

Table 1 Comparison of outlier methods with and without normalizing

Outlier method	Min_Max	Mean_SD	Median_IQR	Median_MAD
COF	+	+	0	0
Ensemble	+	+	+	0
FAST ABOD	+	+	+	+
iForest	0	0	0	0
INFLO	+	+	+	+
KDEOS	0	−	−	0
KNN	+	+	+	+
KNNW	+	+	+	+
LDF	+	+	0	0
LDOF	+	+	+	0
LOF	+	+	0	0
LOOP	+	+	+	+
ODIN	+	+	+	+
SIMLOF	+	+	+	+

The symbols + and − indicates normalizing achieves significantly better and worse results respectively at a level of 5% using Student's t test, while 0 indicates no significant difference

2.3.2 Mixed models

To determine the effects of outlier and normalization methods on performance, we use mixed-effects models. Mixed-effects models are typically used when there is dependence in the data, such as in hierarchical structures. They are well suited for our case since:

1. dependencies arise from dataset variants, as all datasets in our corpus are generated from approximately 200 source datasets; and
2. the structure of the experiment involves the combination of normalization and outlier detection methods, whereby each dataset is normalized using 4 methods, and each outlier detection method is performed on all 4 normalized versions of each dataset.

Thus, we have a structure where the outlier detection method, normalization method and the source dataset play a combined role in influencing performance that we seek to understand.

We use two mixed models to ascertain the significance of normalization. The first model uses outlier detection methods and normalization methods as fixed effects, and source datasets as a random effect. We do not have any interaction terms for this model. We write the first model (using the R formula notation) as:

$$y \sim \text{Out} + \text{Norm} + (1|\text{Source})\,. \tag{19}$$

Here y is the performance, Out is the outlier detection method, Norm is the normalization method and Source is the source dataset. The term (1|Source) means that source

is a random effect and the intercept changes according to the source dataset. We can also write this model in the following way:

$$y_{ijkl} = \lambda + c_i + d_j + h_k + \varepsilon_{ijkl}, \tag{20}$$

where y_{ijkl} is the performance of outlier detection method i using normalization method j on a dataset variant l from source k. The term λ denotes the intercept, c_i the coefficient of the ith outlier detection method, d_j the coefficient of the jth normalization method, h_k the random effect due to the source dataset, and ε_{ijkl} the error term. While the fixed effects coefficients c_i and d_j are parameters, the random effects coefficients h_k are modeled as random variables, i.e. $h_k \sim N(0, \sigma_h^2)$. The errors ε_{ijkl} are assumed to be normally distributed, i.e. $\varepsilon_{ijkl} \sim N(0, \sigma_\varepsilon^2)$.

The second model uses an additional interaction term as follows:

$$y \sim \text{Out} * \text{Norm} + (1|\text{Source})\,. \tag{21}$$

We can also write the second model as follows:

$$y_{ijkl} = \lambda + g_{ij} + h_k + \varepsilon_{ijkl}. \tag{22}$$

The difference between the first and the second model results from the interaction term, which gives rise to a separate regression coefficient g_{ij} for each pair of outlier and normalization methods, rather than assuming their effects are additive. The second model can be used to determine if normalization affects each outlier detection method differently.

As the two models are nested, we perform a likelihood-ratio test and obtain a p-value of 2.2×10^{-16} in favor of the second model making it clear that there are significant interactions between normalization methods and outlier detection methods. In other words, the effect of normalization is different from one outlier method to another.

Figure 5 shows the effect of normalization methods on each outlier detection method using plotting tools described in Breheny and Burchett (2017). The letters O, D, Q, M and X denote the normalization methods None, Mean–SD, Median–IQR, Median–MAD and Min–Max respectively. The plotted value for each normalization and outlier method is $y_{ij.} = \lambda + g_{ij} + \bar{h}$ from Eq. (22) where $\bar{h}$ denotes the mode of h_k, pertaining to the source *connectionist_vowel*. For any other source, the values $y_{ij.}$ is a vertical translation of values shown in Fig. 5. A higher value of $y_{ij.}$ denotes better performance while a lower value denotes a poorer performance. The main quantity of interest of the second model constitutes of the values $y_{ij.}$. As such, we make the following remarks about $y_{ij.}$ using Fig. 5.

1. KNNW has the highest $y_{ij.}$ values, making it the most effective outlier method on average.
2. The four best outlier methods are KNNW, KNN, iForest, and FAST ABOD.
3. On average, iForest does not have a preferred normalization method.
4. KDEOS has the lowest $y_{ij.}$ values, making it the least effective outlier method on average. The second least effective outlier method is INFLO.

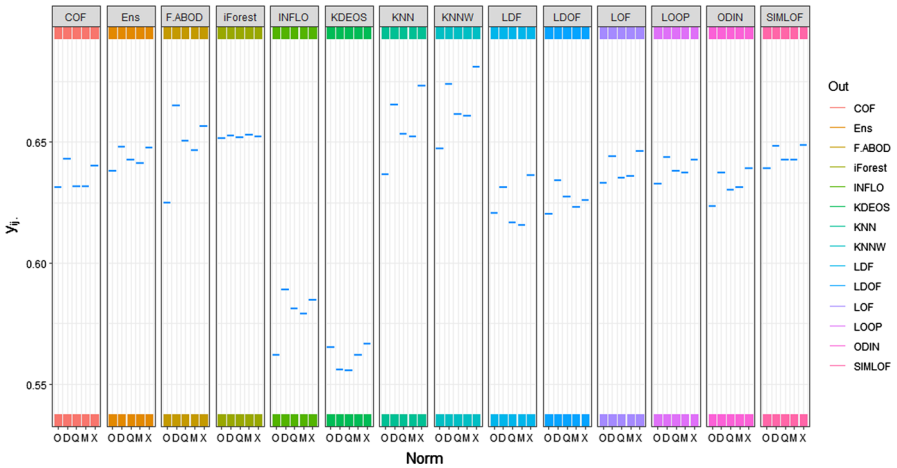

Fig. 5 Effect of normalization on outlier detection methods based on model (22). Here, $y_{ij.} = g_{ij} + \bar{h}$ is plotted for normalization methods None (O), Mean–SD (D), Median–IQR (Q), Median–MAD (M) and Min–Max (X) for each outlier method. Higher values indicate better performance

5. For most outlier methods, Min–Max and Mean–SD outperform Median–IQR and Median–MAD.
6. For most outlier methods, Min–Max and Mean–SD give similar $y_{ij.}$ values, and Median–IQR and Median–MAD also give similar $y_{ij.}$ values.
7. LOF, LOOP, and SIMLOF are quite similar in terms of $y_{ij.}$.
8. The effect of the outlier method on $y_{ij.}$ is greater than the effect of the normalization method.

We repeated the above experiment using a subset of the datasets to alleviate any bias resulting from datasets with more variants and obtained similar results.

As a result of these insights, we only consider normalization methods Min–Max and Median–IQR in the following sections, so as to elicit higher contrasts in performance arising from normalization.

2.4 Preferred normalization methods

In this section, we investigate whether the Min–Max normalization method is significantly superior to Median–IQR. We perform this analysis because Min–Max is the normalization scheme traditionally performed in outlier detection, and to ascertain if we are justified in making this choice. For example, if 80% of the datasets prefer Min–Max as the outlier method, then it is justifiable to use Min–Max as the standard method, but not if it is only 50%.

We test the null hypothesis that Min–Max and Median–IQR are equally preferred. If Min–Max is superior, the null hypothesis will be rejected and the 99% confidence intervals will not include the 50% mark. Table 2 gives the percentage of datasets which prefer Min–Max for each outlier detection method and the associated 99% confidence intervals, and the p-values for the hypothesis test. We see that for most outlier methods

Table 2 Comparing Min–Max to Median–IQR

Outlier method	Min–Max better performance (%)	99% Confidence interval	p-value
COF	51.16	(49.99, 52.36)	1.001×10^{-2}
Ensemble	49.97	(48.80, 51.13)	9.569×10^{-1}
FAST ABOD	55.39	(54.23, 56.54)	4.169×10^{-33}
iForest	50.08	(48.91, 51.24)	8.570×10^{-1}
INFLO	49.03	(47.87, 50.20)	3.358×10^{-2}
KDEOS	50.79	(49.62, 51.95)	8.067×10^{-2}
KNN	57.06	(55.91, 58.21)	1.272×10^{-55}
KNNW	56.39	(55.23, 57.54)	8.004×10^{-46}
LDF	51.05	(49.89, 52.22)	1.922×10^{-2}
LDOF	48.81	(47.64, 49.97)	8.557×10^{-3}
LOF	50.92	(49.75, 52.08)	4.185×10^{-2}
LOOP	50.43	(49.27, 51.59)	3.398×10^{-1}
ODIN	51.70	(50.54, 52.86)	1.555×10^{-4}
SIMLOF	50.87	(49.70, 52.03)	5.399×10^{-2}

apart from FAST ABOD, KNN, and KNNW, the confidence intervals include the 50% mark, making it clear that Median–IQR is a strong contender for Min–Max. This is confirmed by the p-values. Thus, if one were to naively choose Min–Max as the normalization method, the probability of achieving better performance by choosing Median–IQR is between 0.4 and 0.5. This again brings to light the importance of normalization when performing outlier detection, and its complex relationship with dataset characteristics.

2.4.1 Possible conditions for min–max suitability

From Table 2 we see that KNN, KNNW, and FAST ABOD prefer Min–Max to Median–IQR. To give some intuition why Min–Max performs better for these methods, we recall the normalization vector:

$$\boldsymbol{w} = \left(\frac{1}{s_1^2}, \frac{1}{s_2^2}, \ldots, \frac{1}{s_d^2}\right)^T,$$

and the kth nearest neighbor distance

$$\text{nnd}\left(\boldsymbol{x}_i^*, k\right) = \min_{j \in A_{ik}} \sqrt{\langle \boldsymbol{w}, \boldsymbol{y}_{ij} \rangle},$$

and note that when a scaling parameter s_ℓ is high, the ℓth coordinate of the vector $\boldsymbol{w}$ is low. For Min–Max as s_l denotes the range, a high range in the ℓth axis will result in a low ℓ^{th} coordinate in $\boldsymbol{w}$, making $\boldsymbol{w}$ point away from the ℓth axis. For a

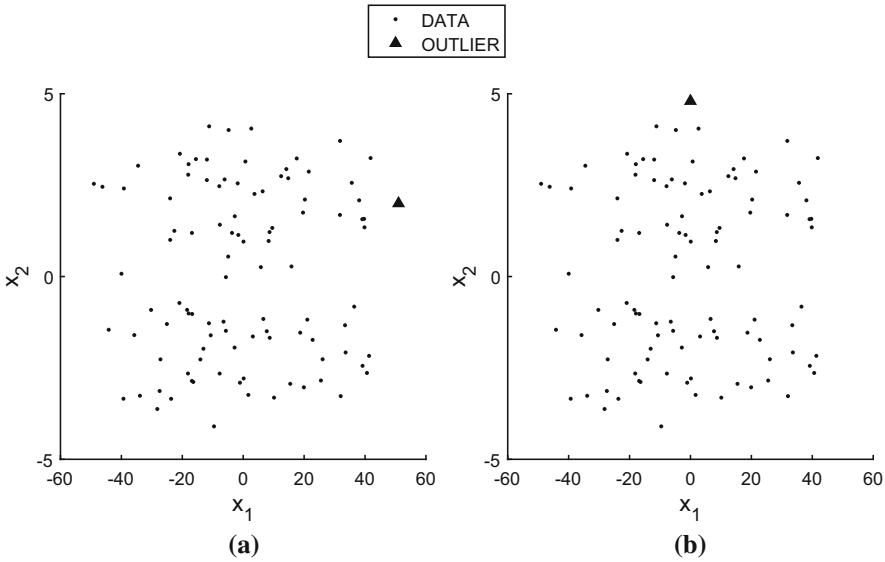

Fig. 6 The dataset above has a comparatively high range in the x_1 axis compared to x_2 axis. An outlier is marked as (▲). Using KNN we compute the performance using Min–Max and Median–IQR normalization methods

given dataset, suppose the ℓth axis has the highest range. As nearest neighbors are found using the above equation, if outliers have bigger values for the ℓth coordinate compared to other coordinates, this would give rise to vectors y_{om} with relatively bigger ℓ^{th} coordinates and relatively smaller other coordinates. As the vector $\boldsymbol{w}$ has a relatively small ℓ^{th} coordinate, unless the ℓth coordinate of y_{om} is sufficiently large to counteract its smaller $\boldsymbol{w}$ counterpart, this would result in low inner product values for outliers, making Min–Max less effective. On the other hand if outliers have low ℓ^{th} coordinates, this gives rise to vectors y_{om} with other coordinates being larger than the ℓth coordinate, making Min–Max more effective than Median–IQR. If a dataset has more than 2 attributes, then there is a higher chance that the outliers are in a different direction compared to the axis of highest range, making Min–Max perform better on average than Median–IQR.

We illustrate this with a simple example. Figure 6 shows a two dimensional dataset having the range 85 and 8 in the x_1 and x_2 axes respectively. In Fig. 6a we place an outlier, depicted in red in the direction of the highest range (i.e., the x_1 axis), and in Fig. 6b in the direction of the x_2 axis. The position of this outlier is slightly varied in each iteration of the experiment. For each iteration, the dataset is normalized using Min–Max and Median–IQR and we compute the KNN ranks with $k = 1$. Table 3 shows the results of these two experiments.

We see that when the outlier is placed in the x_1 direction, which is the direction of the highest range, Median–IQR performs better. And when the outlier is placed in the other direction, Min–Max performs better. Thus when outliers occur in directions other than that of the highest range, Min–Max performs better. However, for a dataset

Table 3 Two simple experiments to illustrate the difference between Min–Max and Median–IQR

Experiment 1	x_1 value	45	46	47	48	49	50	51
	AUC ROC—Min–Max	82.82	87.87	92.92	92.92	96.96	96.96	97.97
	AUC ROC—Median–IQR	82.82	89.89	93.93	95.95	96.96	97.97	98.98
Experiment 2	x_2 value	4.2	4.3	4.4	4.5	4.6	4.7	4.8
	AUC ROC—Min–Max	63.13	55.05	59.09	64.14	77.27	86.36	90.90
	AUC ROC—Median–IQR	55.05	53.03	53.03	61.11	74.24	77.27	83.33

with many attributes, outliers may occur in directions of high variation. For such cases, Median–IQR may give better results.

In addition, one may note that outliers in this example do not deviate a lot from other observations. However, this occurrence is not uncommon when outlier datasets are generated from classification datasets and ground truth is used in labeling outliers.

2.5 Sensitivity to normalization

By inspecting the performance results for a given outlier detection method we see that for some datasets normalization has an effect on performance while for others it does not. How can we determine if normalization affects outlier method performance for a given dataset? If we think of "sensitivity to normalization" as an attribute, is it an intrinsic attribute of the dataset, or is it an attribute of the combination of dataset and outlier detection method? For example, if the performance of an outlier method α_1 on dataset x fluctuates due to normalization, will a different outlier method α_2 on x give fluctuating results as well? We start this investigation by offering a definition of the dataset attribute "sensitivity to normalization".

Definition 2.2 For a given outlier detection method α and a dataset x, we say that the sensitivity to normalization is the difference between the maximum performance and the minimum performance across all normalization schemes for that outlier detection method, i.e.

$$\text{sensitivity}\,(x, \alpha) = \max_{n \in \mathcal{N}} \text{AUC}\,(x, \alpha, n) - \min_{n \in \mathcal{N}} \text{AUC}\,(x, \alpha, n)\,,$$

where $\mathcal{N}$ denotes the set of normalization methods and AUC the area under the curve for dataset x, using normalization method n and outlier method α. Furthermore, we say that an outlier detection method is $\xi-$sensitive to normalization for that dataset, if the difference between the maximum performance and the minimum performance across all normalization schemes for that outlier detection method is greater than ξ.

Figure 7 shows the sensitivity to normalization for each outlier method in terms of densities and box plots. We perform the Friedman test to find out if there is are significant differences in outlier methods with respect to normalization sensitivity. To adjust for dataset variants, we compute the median sensitivity to normalization for each

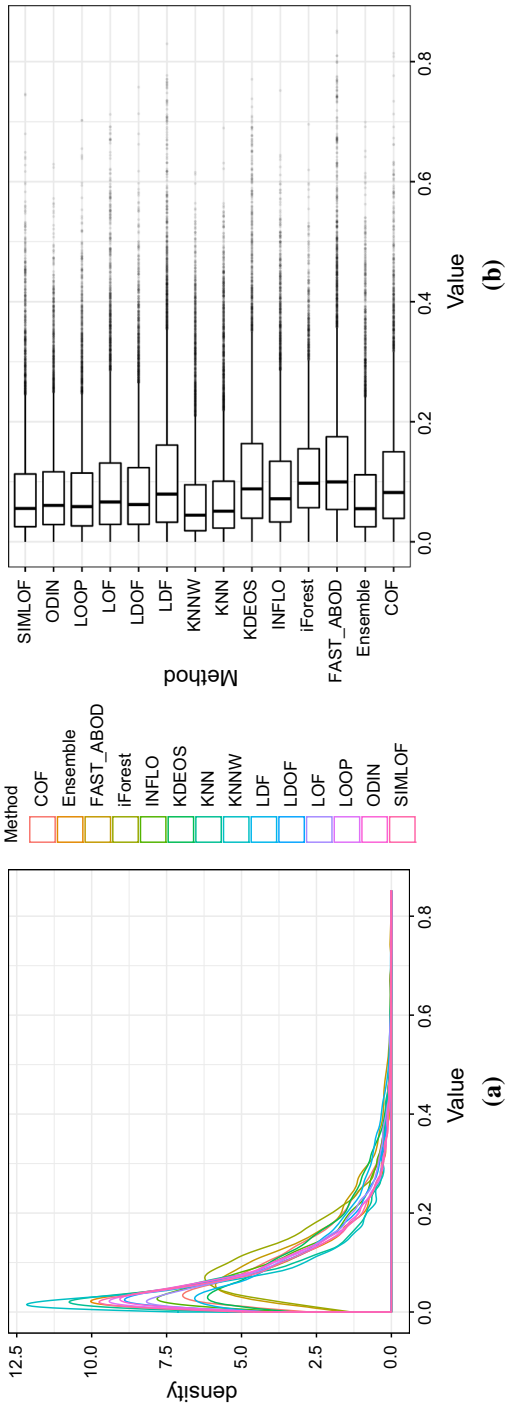

Fig. 7 Sensitivity to normalization for each outlier method. The sensitivity to normalization value is on the x axis for both graphs

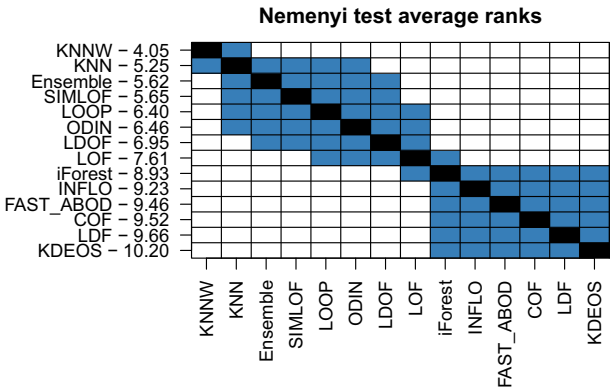

Fig. 8 Nemenyi test output. The average rank on the y axis with low ranks signifying robustness to normalization methods and high ranks signifying sensitivity to normalization methods. The blue-colored cells denote the methods that are not significantly different from the cell in black (Color figure online)

Table 4 Confidence intervals of Nemenyi test ranks

Outlier detection method	Confidence intervals of ranks
KNNW	(2.55, 5.54)
KNN	(3.75, 6.74)
Ensemble	(4.12, 7.12)
SIMLOF	(4.15, 7.14)
LOOP	(4.90, 7.89)
ODIN	(4.96, 7.95)
LDOF	(5.45, 8.44)
LOF	(6.11, 9.10)
iForest	(7.43, 10.42)
INFLO	(7.73, 10.72)
FAST_ABOD	(7.96, 10.95)
COF	(8.02, 11.01)
LDF	(8.16, 11.15)
KDEOS	(8.70, 11.70)

source dataset for each outlier method. The null hypothesis for this test is that there is no difference in sensitivity to normalization between the outlier methods. The Friedman test gives a p-value of 2.2×10^{-16}, rejecting the null hypothesis. As the outlier methods respond differently for sensitivity to normalization, we perform a Nemenyi post-hoc test. The Nemenyi test ranks the outlier methods, with low ranking corresponding to more robust methods and high ranking corresponding to more sensitive methods. Figure 8 shows the average ranks of methods with vertical lines denoting methods which are not significantly different from each other. Table 4 gives the 95% confidence intervals of these ranks.

Table 5 ξ-sensitivity to normalization for data sources

ξ	Least sensitive		Most sensitive	
	Dataset source	Num. outlier methods	Dataset source	Num. outlier methods
0.05	Ring	0.1794	Annthyoid	13.909
0.10	Ring	0	Annthyoid	13.181
0.15	Abalone_ori	0	Wilt	12.454
0.20	Abalone_ori	0	Wilt	11.454

From Table 4 and Fig. 8 we see that KDEOS is the method most sensitive to normalization followed by LDF. The outlier detection methods KNNW is the least sensitive to normalization with KNN and Ensemble methods achieving comparable results. The method KNNW uses the average of KNN distances, and as such is more robust to normalization methods. Similarly, Ensemble methods are also robust to normalization as they use the combined output of many methods. Somewhat unexpectedly, iForest appears in the middle of the list, even though it does not compute distances or densities. This can be due to the random selection of attributes which is an integral part of the iForest algorithm. Indeed from Fig. 5 we see that it does not have any preferred normalization method. However, due to the random selection of attributes, each normalization method can give rise to different performance values.

This outcome further validates the results of the second mixed model in Sect. 2.3 given by Eq. (21). In other words, we explicitly see evidence of normalization affecting outlier detection methods differently.

Next, we investigate if certain source datasets are more sensitive to normalization compared to others. For a fixed ξ we compute the number of outlier methods that are $\xi-$sensitive for each dataset. For example, dataset x_1 might be ξ-sensitive for 2 outlier methods and dataset x_2 might be ξ-sensitive for 10 outlier methods. We want to test if there are significant differences among the dataset sources with respect to ξ-sensitivity to normalization. The null hypothesis is that there is no difference.

We group the number of ξ-sensitive outlier methods by its source dataset and perform a Kruskal Wallis test. We perform this experiment for each value of $\xi \in \{0.05, 0.10, 0.15, 0.2\}$. For all 4 experiments we obtain p-values of 2.26×10^{-16}, rejecting the null hypothesis. Table 5 gives the least and the most ξ-sensitive dataset source, with corresponding the average number of outlier methods that are sensitive to normalization. As there are approximately 180 dataset sources, we do not perform a post-hoc analysis to differentiate the pairwise differences between the dataset sources.

Table 5 shows that on average an Annthyoid dataset is ξ-sensitive to normalization for nearly all outlier methods for $\xi = 0.05$ and a ring dataset is not sensitive to normalization for any outlier method. Thus, the percentage of datasets that is ξ-sensitive to normalization for n outlier methods is of interest to us. Figure 9 shows the density distributions of the datasets that are ξ-sensitive to n outlier methods for $\xi \in \{0.05, 0.10, 0.15, 0.20\}$ and $n \in \{1, \ldots, 14\}$.

From Fig. 9 we see that some datasets are sensitive to normalization for all outlier methods. As shown in Table 5 this is likely due characteristics of the dataset than that

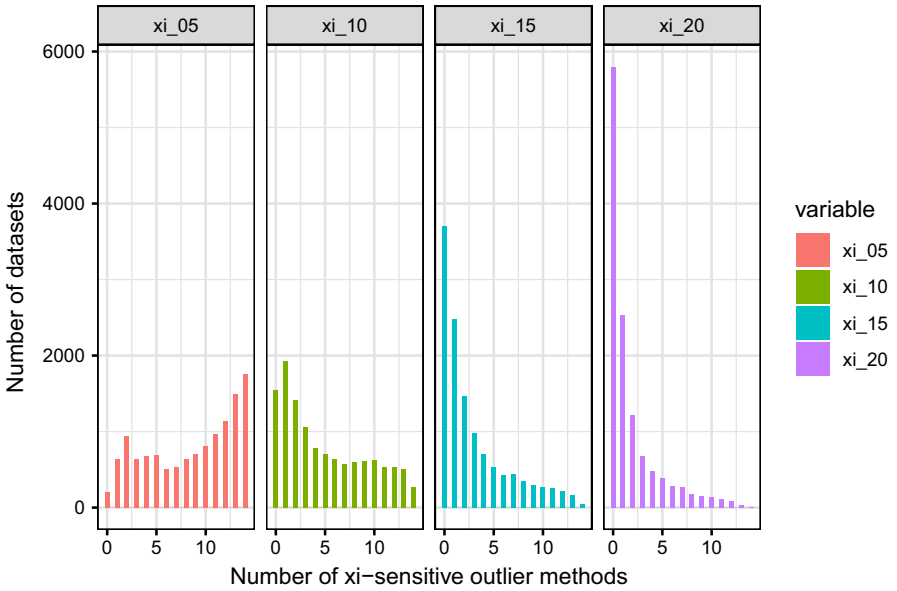

Fig. 9 The number of datasets that is ξ-sensitive to normalization for n outlier methods for $\xi \in \{0.05, 0.10, 0.15, 0.20\}$ and $n \in \{1, \ldots, 14\}$

of the outlier method. Similarly, there are datasets that are only sensitive to one outlier method. This is likely due to the outlier method characteristics than that of the dataset as we know that some outlier methods are more sensitive to normalization than others (Fig. 8). Thus the sensitivity to normalization of the datasets at each end of the density graph in Fig. 9 is explained by either the dataset or the outlier method. However, this explanation does not carry over for datasets that lie in the middle of the graph in Fig. 9. For example there are some datasets that are sensitive to normalization for 6 outlier methods. It is a combination of dataset and outlier method characteristics that make these datasets sensitive to normalization. Thus there is a subtle interplay between the datasets and the outlier methods in determining sensitivity to normalization.

This section has provided comprehensive evidence, both theoretical and experimental, that normalization can have a significant impact on some outlier detection methods, and that the complex interplay of dataset characteristics, outlier detection method, and normalization scheme makes it challenging to ensure the best algorithm is selected for a given dataset. We now turn to recent advances in instance space analysis to address the challenges of this algorithm selection problem.

3 Instance space analysis

The *instance space* models the relationship between structural properties of an instance to the performance of a group of algorithms. It was first proposed by Smith-Miles et al. (2014), using as a foundation the Algorithm Selection Problem framework developed

by Rice (1976). Through the analysis of the instance space it is possible to determine the strengths and weaknesses of an algorithm; thus, facilitating objective assessments of comparative algorithm power.

Figure 10 illustrates the framework underpinning the development of the instance space. At its core, there are five component *spaces*. The first one is the ill-defined *problem space*, $\mathcal{P}$, which contains all the relevant problems in the application domain (e.g. outlier detection). However, we only have instances and computational results for a subset, $\boldsymbol{I}$. Second is the algorithm space, $\mathcal{A}$, which is composed of a portfolio of algorithms applied to the problems in $\boldsymbol{I}$. Third is the performance space, $\mathcal{Y}$, which is the set of metrics $y(\alpha, x)$, measuring the performance of an algorithm $\alpha \in \mathcal{A}$ to solve a problem $x \in \boldsymbol{I}$. Fourth is the *feature space*, $\mathcal{F}$, which contains multiple measures that characterize the properties that can distinguish similarities and differences between instances in $\boldsymbol{I}$, and that may correlate with difficulty for various algorithms. These features are represented by the vector $\boldsymbol{f}(x)$. The meta-data, composed of the features and algorithm performance for all the instances in $\boldsymbol{I}$, is used to learn the mapping $g(\boldsymbol{f}(x), y(\alpha, x))$ that can be used to predict the performance of an algorithm, given a feature vector summary of an instance. Finally, the instance space can be visualized and algorithm performance inspected across the instance space, once we project an instance x from a high-dimensional feature space to the two-dimensional instance space. The methods used to learn the performance mapping, and to project from a high-dimensional feature space to a 2D instance space are flexible. In this paper, we adopt the approach from Muñoz et al. (2018) to obtain an optimal projection that encourages linear trends in both features and algorithm performance to be visualized across the resulting instance space.

Instance Space Analysis involves a study of the instances described by their location in the instance space, thus by their features, and the performance of algorithms in various parts of the instance space. In particular, we are able to construct *footprints* for each algorithm, defined as the region in instance space where we statistically infer good performance of the algorithm, for a user-defined criteria of good. Furthermore, instance space allows us to: [(a)]visualize the distribution and diversity of existing benchmark and real-world instances; [(b)] assess the adequacy of the features; [(c)] describe the unique strengths and weaknesses of algorithms; [(d)] identify and measure the algorithm's *footprint* to objectively compare algorithms; [(e)] partition the instance space into recommended regions for automated algorithm selection; and [(f)] distinguish areas of the instance space where it may be useful to generate additional instances to gain further insights. The unique advantage of visualizing algorithm performance in the instance space, rather than as a small set of summary statistics averaged across a large collection of instances, is the nuanced analysis that becomes possible to examine interesting variations in performance that may be hidden by tables of summary statistics.

The meta-data from which we now construct the instance space is described by the problem instances (see datasets in Sect. 2.2.1), the algorithms (see outlier detection methods in Sect. 2.2.2), and the performance metric described in Sect. 2.2.3, as well as a set of outlier detection dataset features we propose below.

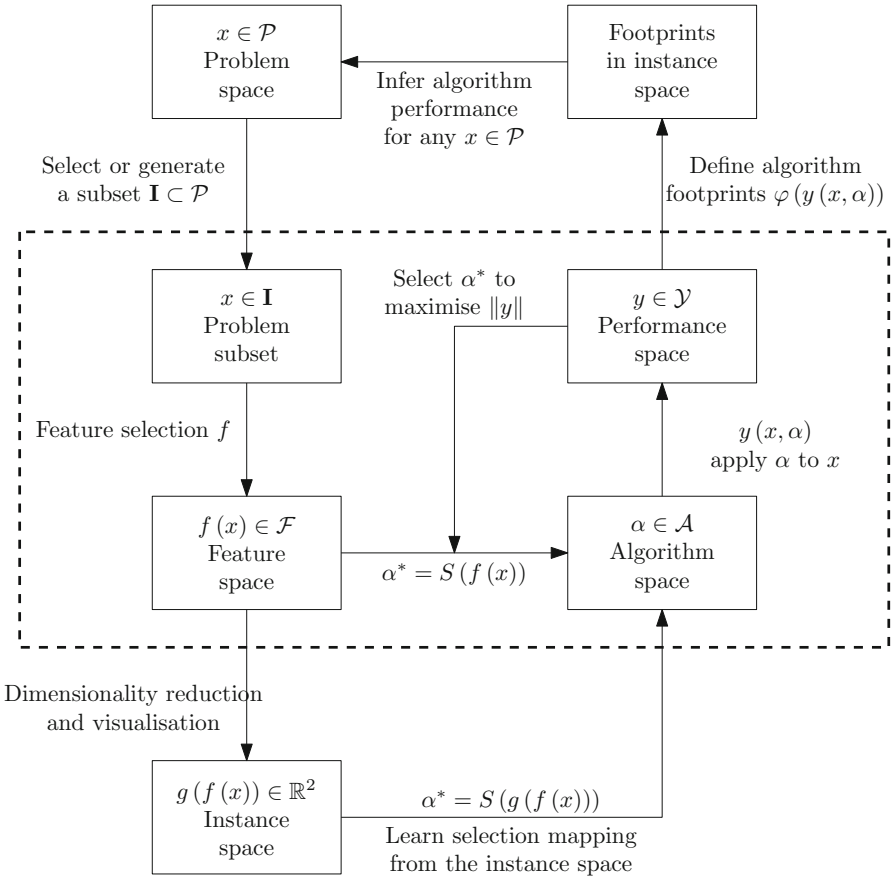

Fig. 10 Summary of the Instance Space methodology proposed by Smith-Miles et al. (2014), underpinned by the Algorithm Selection framework (in the dotted box) by Rice (1976)

3.1 Features

We compute a broad set of candidate features, from which we select the most predictive subset later in Sect. 3.3. While some features are more general, others are geared toward outlier detection and use class labels in the feature computation. The reason for using class labels is because outlier detection methods often disagree on outliers as shown in Fig. 1. In particular, by taking ground truth in labeling outliers, each problem has its own internal definition of an outlier. As such we need training data to know what each problem calls an outlier so that we can recommend an appropriate outlier method for that problem and future data from a similar source. This is common in industrial applications, where we work with training data with labeled outliers with the aim of finding methods to elicit these outliers (Leigh et al. 2019; Talagala et al. 2019). Furthermore, we use a supervised methodology to find suitable unsupervised outlier detection methods, for a given dataset, but the recommended method remains

unsupervised. A detailed description of these features is contained in the GitHub repository Kandanaarachchi (2018), and the associated R package *outselect* has the functionality to compute them. Broadly, we categorize the features as follows:

1. *Standard meta-features*—These range from simple and statistical features to information-theoretic features. Examples of simple features include the number of observations and attributes of a dataset, while statistical features describe properties such as skewness and kurtosis, and information-theoretic features are drawn from concepts such as entropy.
2. *Outlier features*—These take into account the outlier structure of a dataset. In addition to outliers, we also consider proxy-outliers. We define proxy-outliers using distance, density, residuals, and graph-based metrics. Distance-based proxy-outliers are the points with the top 3% KNN distances. Density-based proxy-outliers are the points with the lowest 3% density estimates on an arbitrarily chosen plane. Residual based proxy-outliers are points contributing to the highest residuals of a linear model where the dependent variable is arbitrarily selected and the other variables are used as predictors. Graph-based proxy-outliers are the points with the lowest degree when the dataset is transformed into a graph. For a given dataset if proxy-outliers and outliers agree, then we can expect outlier methods that are associated with the proxy-outlier definition to perform well on that dataset. Proxy-outlier based features are important because we solicit features that explain outlier method performance.
   Features based on proxy-outliers fall into the category of landmarking features, which has been popular in meta-learning studies (Pfahringer et al. 2000; Peng et al. 2002; Smith-Miles 2009). The concept of *landmarking* is to obtain a rough picture of the problem space with the aid of some simple tools.
   The outlier features can be further classified as follows:

   (a) Density-based features: For this collection of features, we compute densities on arbitrary subspaces using DBSCAN, kernel density estimates, and local density methods. An example feature is the ratio of density estimates between proxy-outliers and non-proxy-outliers.
   (b) Residual based features: These features are based on residuals of linear models, where a variable is randomly selected as the dependent variable and others as independent variables. An example feature is the ratio of the median residual value of outliers to that of non-outliers.
   (c) Graph-based features: These features are based on graph-theoretic properties such as vertex degree and shortest path. We generate a graph associated with each dataset using distances between points. From this graph, we compute the degree of vertices, shortest path between points and connected components. An example feature is the ratio of standard deviation of degree of all vertices to that of proxy-outliers.
   (d) Normalization based features: These features are derived from quantities described in Sect. 2.1.

From this list of features, those that are based on density, residuals, and graphs depend on nearest neighbors and as such are sensitive to the method of normalization.

Table 6 Types of features calculated

Feature category	Number of features	Normalization methods	Total features
Standard meta-learning	25	NA	25
Density based	77	2	154
Residual based	35	2	70
Graph based	41	2	82
Normalization based	15	NA	15
Total			346

Hence we calculate features for each of the two selected normalization methods that we have earlier shown are not correlated, namely Min–Max and Median–IQR. The choice of these two is justified since: 1. Min–Max is the most commonly used normalization method for outlier detection, 2. Median–IQR is one of the methods which is robust to outliers. By combining density, residual and graph-based features computed on datasets normalized by 2 different methods with standard features and normalization based features we end up with a total of 346 candidate features. Table 6 provides a summary of features by category.

In our previous work (Kandanaarachchi et al. 2019a) we validated these features by showing that they can predict the performance of some outlier methods with min–max normalization. Given the different focus of this paper, we now demonstrate that we can predict sensitivity to normalization with a reasonable accuracy.

3.2 Predicting sensitivity to normalization

We demonstrated in Sect. 2 that some combinations of datasets and outlier methods are sensitive to normalization, but can we predict these combinations? That is, given a dataset and an outlier method, can we predict if the dataset is sensitive to normalization with respect to that outlier method and if it is sensitive, which normalization method should be used? To investigate this question we use features discussed in Sect. 3.1. Using tenfold cross-validation (Bischl et al. 2012), we train and test 12 random forest classifiers (Liaw and Wiener 2002), one for each outlier method, with all 346 features as input to predict the binary output of ξ-sensitivity to normalization with $\xi = 0.05$. The results are given in Table 7. As shown in Table 7 prediction accuracy of sensitivity to normalization ranges from 71 to 80% with FAST ABOD, which was the method most sensitive to normalization, achieving the highest prediction accuracy. Also, it is insightful to compare these prediction accuracies with the actual percentages of datasets that are sensitive to normalization, which is given in column 2 of Table 7. In general, we can correctly predict if a dataset is sensitive to normalization with respect to an outlier detection method with an accuracy greater than 70%, suggesting that the feature set must contain some useful summaries of relevant dataset properties.

Next, we investigate which normalization method gives better performance if a dataset is sensitive to normalization for a given outlier detection method. We only consider the normalization methods Min–Max and Median–IQR, and datasets that

Table 7 Prediction results for ξ-sensitivity to normalization with $\xi = 0.05$ using tenfold cross validation

Outlier detection method	Actual percentage sensitive to normalization (%)	Prediction accuracy of sensitivity to normalization (%)
COF	67.36	77.23
Ensemble	53.38	74.88
FAST ABOD	77.43	80.02
iForest	78.97	80.62
INFLO	62.74	75.07
KDEOS	68.51	76.75
KNN	50.59	73.08
KNNW	54.13	73.22
LDF	63.90	71.63
LDOF	57.64	73.65
LOF	59.82	74.27
LOOP	55.42	73.66
ODIN	57.27	73.30
SIMLOF	53.74	73.92

are ξ-sensitive to normalization for each outlier method with $\xi = 0.05, 0.10$, and 0.15. Using features of ξ-sensitive datasets as input to a random forest classifier using fivefold cross-validation, we predict the normalization method that gives better performance with results shown in Table 8. From this table we observe that prediction accuracy generally increases with ξ. This is to be expected because it is easier for the classifier to predict the preferred normalization method as the sensitivity to normalization increases. Also, prediction accuracy is higher for KNN, KNNW and FAST ABOD than for other outlier methods.

From the results of the mixed models in Sect. 2.3 we know that normalization methods affect outlier methods differently. As such, one of the reasons for high fluctuations in prediction accuracy seen in Table 8 might be because the set of features does not sufficiently explain these effects for all outlier methods equally. Indeed, the features were pooled with the intent of discovering strengths and weaknesses of outlier detection methods, not of normalization methods. Only a handful of features focus on normalization as seen in Table 6. When comparing with Table 7 which predicts the sensitivity to normalization, Table 8 has higher contrasts in terms of accuracy. However, from both these tables we see that we can reasonably predict if a dataset is sensitive to normalization given an outlier method, and if it is sensitive to normalization which normalization method to recommend.

In effect, we are proposing a strategy to select the normalization method to maximize performance. First for a preferred outlier method, we find if a dataset is sensitive to normalization using features and a classifier. If it is sensitive, then we find which normalization method gives better performance. One may ask how one selects the preferred outlier method. This question will be answered in detail in Sect. 3.3.

Table 8 Best normalization method prediction accuracy

Outlier detection method	$\xi = 0.05$ (%)	$\xi = 0.10$ (%)	$\xi = 0.15$ (%)
COF	59.15	62.35	62.95
Ensemble	61.01	67.46	69.64
FAST ABOD	77.40	80.14	83.45
iForest	51.42	50.86	51.19
INFLO	60.34	62.45	62.65
KDEOS	59.37	63.24	68.96
KNN	71.75	75.73	76.04
KNNW	74.82	76.65	84.49
LDF	60.44	63.89	68.42
LDOF	67.25	68.16	70.87
LOF	65.65	60.37	58.83
LOOP	60.95	66.65	67.16
ODIN	60.44	62.35	67.67
SIMLOF	65.38	62.85	67.59

3.3 Constructing an outlier detection instance space

Critical to fitting the instance space is the identification of a portfolio of complementary algorithms (i.e. with uncorrelated performance), and the selection of a subset of features that are distinctive (i.e. uncorrelated with each other) and predictive (i.e. correlated with algorithm performance). This is not only to reduce the computational cost of fitting the space but also because we want to focus on modeling the most meaningful relationships. Our first step is to reduce the number of candidate algorithms. As such, from this point forward we define an algorithm as the combination of a normalization scheme and a detection method. Considering the two normalization methods deemed significantly different in Sect. 2.3.2, i.e., Min–Max and Median–IQR, and the 14 detection methods listed in Sect. 2.2.2, we consider 28 candidate algorithms. Using as dissimilarity measure $1 - \left|\rho_{\alpha,\beta}\right|$, where $\rho_{\alpha,\beta}$ is the correlation between the performance of two algorithms, we find the following eight groups using clustering (Maechler et al. 2018). From each one, we select the algorithm in italics, such that each family of detection and normalization methods is represented.

Group 1 COF Median IQR—*Ensemble Median IQR*—INFLO Median IQR—LDOF Median IQR—LOF Median IQR—LOOP Median IQR—ODIN Median IQR—SIMLOF Median IQR

Group 2 COF Min Max—Ensemble Min Max—INFLO Min Max—LDOF Min Max—*LOF Min Max*—LOOP Min Max—ODIN Min Max—SIMLOF Min Max

Group 3 FAST ABOD Median IQR—*KNN Median IQR*—KNN Min Max—KNNW Median IQR—KNNW Min Max—LDF Median IQR

Group 4 *FAST ABOD Min Max*

Group 5 *iForest Median IQR*—iForest Min Max

Group 6 *KDEOS Median IQR*
Group 7 *KDEOS Min Max*
Group 8 *LDF Min Max*

In Sect. 3.2 and Kandanaarachchi et al. (2019a) we provided evidence that the proposed features are predictive of both normalization effects and outlier detection method performance. Hence, our next step is to reduce the initial set of 346 features into a smaller set. For this purpose, we employ a subset of 3142 instances for which there are three well-performing algorithms or less, i.e., AUC $\geq$ 0.8. This choice is somewhat arbitrary, motivated by the fact that we are less interested to study datasets, where many algorithms perform well or poorly, given our aim, is to understand strengths and weaknesses of outlier detection methods. We will later project the full set of datasets into the constructed instance space, but consider this reduced set sufficient for performance prediction and instance space construction. The details of feature selection procedure are presented in Kandanaarachchi et al. (2019a). In summary:

1. Pre-process the data by bounding large and infinite values between their median plus or minus five times their interquartile range. Not-a-number values are converted to zero.
2. Normalize and standardize each feature through Box–Cox and Z transformations.
3. Discard those features which only have a limited number of unique values or are not one of the top three features by correlation with the performance of any algorithm.
4. Identify groups of highly correlated features using a clustering algorithm with the dissimilarity measure being $1 - |\rho|$, where ρ is the correlation between two features.
5. Selecting one feature from each group, find the combination that, once projected into 2D, allows the most accurate prediction of the performance of all algorithms.

As a result, we obtain a set of seven features from which we construct the instance space. We then use the Prediction Based Linear Dimensionality Reduction (PBLDR) method (Muñoz et al. 2018) to find a projection from 7D to 2D, such that algorithm performance and feature values increase linearly from one edge of the instance space to the opposite; hence, assisting the visualization of directions of hardness and feature correlation. To find the projection, we make use of BFGS as optimization algorithm. However, given that PBLDR has infinite solutions, we calculate 30 different projections and select the one with the highest topological preservation, defined as the correlation between high- and low-dimensional distances. The final projection matrix is defined by Eq. 23, which transforms the 7D feature vector into a 2D vector $\mathbf{Z}$. The L_2–norm of each row from this matrix provides an indication of the most influential features in the space, with the top three being OPO_GDeg_Out_Mean_3 (0.4190),

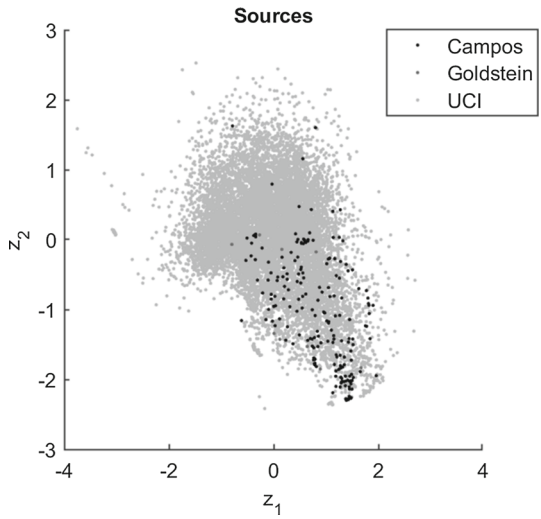

Fig. 11 Instance space including the full set of more than 12,000 instances, discriminated by their source

OPO_Res_Out_Mean_3 (0.4180) and OPO_Out_LocDenOut_2_3 (0.3333).

$$\mathbf{Z} = \begin{bmatrix} -0.2334 & -0.2324 \\ 0.0798 & -0.3236 \\ -0.3712 & -0.1945 \\ 0.2201 & -0.0993 \\ 0.2089 & -0.1657 \\ 0.2907 & -0.3003 \\ -0.0125 & -0.2422 \end{bmatrix}^{\top} \begin{bmatrix} \text{OPO\_GComp\_PO\_Mean\_1} \\ \text{OPO\_Out\_LocDenOut\_2\_3} \\ \text{OPO\_GDeg\_Out\_Mean\_3} \\ \text{OPO\_Res\_ResOut\_95P\_1} \\ \text{OPO\_Res\_ResOut\_Mean\_3} \\ \text{OPO\_Res\_Out\_Mean\_3} \\ \text{MEAN\_PROD\_IQR} \end{bmatrix} \tag{23}$$

Figure 11 illustrates the resulting instance space, including the full set of more than 12,000 instances, discriminated by their source. The sets by Campos et al. (2016) and Goldstein and Uchida (2016) are mostly located in the lower right area of the space; whereas the set produced by down-sampling, the UCI repository provides greater coverage of the instance space and hence more diversity of features. Finally, Figs. 12 and 13 show the distribution of feature values and outlier method performance across the instance space respectively, based only on the subset of 3142 instances. The scale has been adjusted to the [0, 1] range. To understand the construction of the instance space, we now briefly describe the details of each selected feature, their weights in the projection matrix and their trends across the instance space.

OPO_GComp_PO_Mean_1 This feature captures information of the inner-points of the dataset. While there is not much correlation between this feature with any of the outlier methods, it is the only one that increases from top-right to bottom-left, as observed in Fig. 12e. It is calculated as follows:

1. Normalize the dataset using Min–Max.
2. Generate a graph from the dataset using distances between points using Csardi and Nepusz (2006). Suppose data point v_1 is the nearest neighbour of data

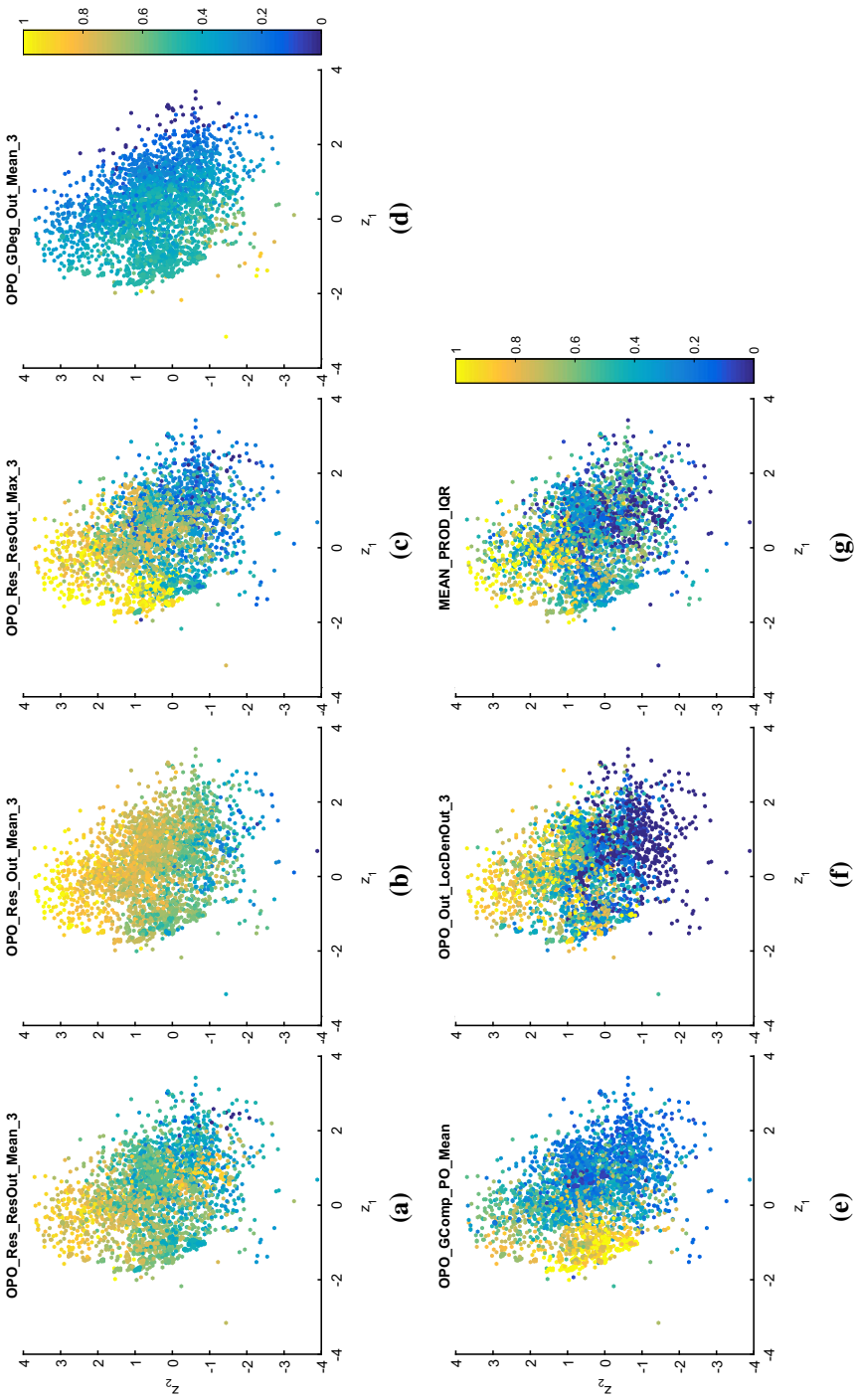

Fig. 12 Distribution of normalized features on the projected instance space

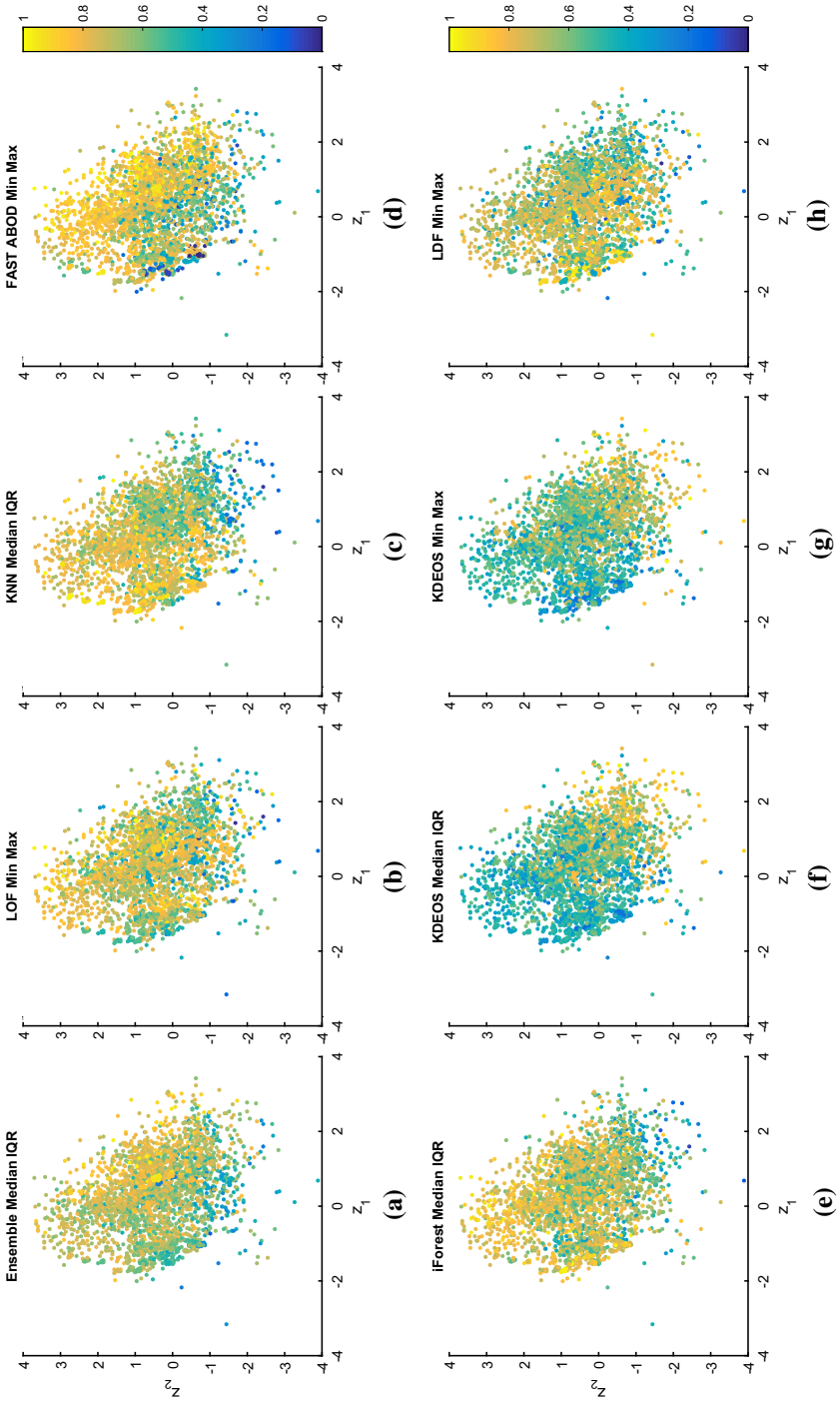

Fig. 13 Scaled area under the curve for each outlier detection method on the instance space

point v_2. Then the associated graph will have vertices v_1 and v_2 connected with an edge.
3. Compute the connected components and the degree of vertices of this graph.
4. Define the potential inner points ω as points with the top 3% of degree values.
5. Find the associated connected components having these inner points ω and find the number of vertices γ in each connected component.
6. OPO_GComp_PO_Mean_1 $=$ mean$(\gamma[\omega])/$ mean$(\gamma[\bar{\omega}])$, where $\bar{\omega}$ denotes the non-inner points.

OPO_Out_LocDenOut_2_3 This feature computes a ratio of subspace density proxy-outliers to actual outliers. It is calculated as follows:

1. The dataset is normalized using Median–IQR.
2. Compute Principle Components of the dataset
3. Compute kernel density estimates for the two dimensional subspaces, spanned by the principle component pairs (1, 2), (2, 3), (3, 4) etc …stopping with the 10th principle component.
4. For each point, consider a neighbourhood $\mathcal{N}$.
5. For each point, define the local density as the density of that point divided by the average density of the points in the neighbourhood.
6. Define the density based proxy-outliers as the points with the smallest 3% density values.
7. Let $\omega =$ number of density proxy-outliers that are actual outliers/number of density proxy-outliers.
8. OPO_Out_LocDenOut_2_3 $=$ mean(ω) where ω is computed for each principle component subspace defined above.

This feature decreases with z_2 values in Fig. 12f. This feature correlates inversely with KDEOS.

OPO_GDeg_Out_Mean_3 is a ratio of the mean degree of outliers to mean degree of non-outliers. As shown in Fig. 12d, low values of this feature indicate that outliers have a lower degree compared to non-outliers and are depicted in blue. It is calculated as follows:

1. Normalize the dataset using Median–IQR.
2. Generate a graph from the dataset as in OPO_GComp_PO_Mean_1.
3. Compute the degree θ of each vertex v.
4. Let ω be the set of outliers.
5. OPO_GDeg_Out_Mean_3 $=$ mean$(\theta[\omega])/$ mean$(\theta[\bar{\omega}])$ where $\bar{\omega}$ denotes non-outliers.

This feature captures the performance of FAST ABOD because outliers that have a lower degree are generally isolated and as such will make smaller angles with their neighbors. Thus we see the algorithm FAST ABOD performing well on datasets that have low values of OPO_GDeg_Out_Mean_3. Even though we are not finding the angle between points, the associated graph degree has a correlation with the performance.

OPO_Res_ResOut_Mean_3 is the ratio between the mean of residuals of proxy-outliers to the mean of residuals of non-proxy-outliers. It is shown in Fig. 12a and it is calculated as follows:

1. Normalise the dataset x using Median–IQR.
2. Randomly choose a set of non-binary attributes s from x.
3. For each attribute a in s, fit a linear model l with a as the dependent variable and others as the independent variables.
4. For each model l, compute the residuals r. Define the residual based proxy-outliers ω as the residuals with the top 3% absolute residual values.
5. For each model l, compute $\kappa = \text{mean}(r[\omega])/\,\text{mean}(r[\bar{\omega}])$, where $\bar{\omega}$ denotes non proxy-outliers.
6. The feature is the average of κ.

If certain points of a dataset have large residuals for many linear models, we can expect these points to be outliers. If they are indeed outliers according to ground truth, it should be relatively easy to find these outliers for many methods, since they have large residuals. As such, datasets that have outliers that have large residuals, i.e. large values of OPO_Res_ResOut_Mean_3 are easy instances for many methods as seen by Fig. 12a. However, datasets that have non-outliers with large residuals have also large values of OPO_Res_ResOut_Mean_3. These are not easy datasets. These datasets are colored in yellow, in the bottom half of Fig. 12a.
OPO_Res_ResOut_95P_1 is a similar to OPO_Res_ResOut_Mean_3, with the exception that $\kappa = \text{Perc}(r[\omega], 95)/\text{Perc}(r[\bar{\omega}], 95)$, where $\text{Perc}(\cdot, 95)$ denotes the 95th percentile and the dataset is normalized using Min–Max. It is shown in Fig. 12c.
OPO_Res_Out_Mean_3 is the ratio between the mean of residuals of outliers to the mean of residuals of non-outliers. It is shown in Fig. 12b and it is calculated following a similar approach to that of OPO_Res_ResOut_Mean_3 with the exception that ω denotes actual outliers instead of proxy-outliers. Hence, if the outliers have large residuals then it is an easy dataset for many methods.
MEAN_PROD_IQR Here we try to see if outliers are masked by non-outliers. As in Proposition 2.1 we compute $\frac{\text{nnd}(\boldsymbol{x}_o^*,k)^2}{\text{nnd}(\boldsymbol{x}_n^*,k)^2} = \frac{\langle \boldsymbol{w}, \boldsymbol{y}_{oa}\rangle}{\langle \boldsymbol{w}, \boldsymbol{y}_{nm}\rangle}$ for outliers and non-outliers using Eqs. (12) and (13) for the normalization vector $\boldsymbol{w}$ obtained by using Median–IQR. The feature MEAN_PROD_IQR is the mean of this quantity, which is computed for many outliers and non-outliers. High values of this feature indicate that outliers are not masked by non-outliers using that normalization scheme and low values indicate the opposite.

With this information and thorough inspection of the figures, we can conclude that of these seven features, three are related to residual-based ratios. This shows the relationship between residuals and outliers in terms of determining easy datasets for multiple outlier methods. Moreover, we can distinguish three broad groups of algorithms. The first one, composed of the Ensemble, LOF and FAST ABOD variants, the second one, composed of KNN and iForest variants and the third one consisting of KDEOS variants. The performance of the first group increases from left to right of the space, which is well described by OPO_GDeg_Out_Mean_3. On the other hand,

Table 9 Footprint analysis of the algorithms

	AUC ≥ 0.8			Best algorithm		
	α_N (%)	d_N (%)	p (%)	α_N (%)	d_N (%)	p (%)
Ensemble median IQR	3.4	150.2	93.7	0.4	434.6	86.8
LOF min Max	3.3	318.2	91.2	1.1	350.4	89.1
KNN median IQR	8.5	194.6	93.4	4.0	169.5	86.4
FAST ABOD min max	12.0	246.6	89.4	9.2	248.6	81.5
iForest median IQR	3.1	336.2	94.5	1.8	293.6	85.2
KDEOS median IQR	2.3	81.0	96.6	1.6	78.0	85.0
KDEOS min max	1.0	66.0	95.0	3.6	34.4	76.9
LDF min max	1.8	357.8	94.1	1.7	410.7	84.1

α_N is the area, d_N the density and p the purity. The footprint areas (and their density and purity) are shown where algorithm performance is AUC ≥ 0.8

the performance of the second group increases from the top to the bottom of the space, which is well described by OPO_Res_ResOut_Mean_3. The performance of KDEOS variants is described by OPO_Out_LocDenOut_2_3.

3.4 Footprint analysis of algorithm strengths and weaknesses

We define a footprint as an area of the instance space where an algorithm is expected to perform well based on inference from empirical performance analysis (Smith-Miles and Tan 2012). To construct a footprint, we follow a simplified version of the approach first introduced by Smith-Miles and Tan (2012): (a) we take a sub-sample of 50% of the instances, including those in the convex hull; (b) we calculate a Delaunay triangulation; (c) we calculate the density and purity of each triangle; and, (d) we discard any triangle that does not fulfill the density and purity thresholds. The density threshold, ρ, is set to 10, and the purity threshold, π, is set to 75%. We then remove any contradictions that could appear when two different conclusions could be drawn from the same section of the instance space due to overlapping footprints, e.g., when comparing two algorithms. This is achieved by comparing the density and purity of the contradicting sections. The algorithm whose contradicting section has higher density and purity gets to keep it.

Table 9 presents the results from the footprint analysis. The best algorithm is the one with the largest area under the ROC curve for the given instance. The results are expressed as a percentage of the total area (14.7379) and density (213.1922) of the convex hull that encloses all instances. The table demonstrates that the dominant algorithm is FAST ABOD, with a footprint covering 9.2% of the space. KDEOS and LDF have regions of unique strength. If we only consider their average performance, as discussed in Sect. 2.3.2, we could think that both are unremarkable. However, their footprints are the third and fifth largest with 3.6% and 1.7% respectively.

3.5 Automated algorithm selection in the instance space

One of the main advantages of the instance space is that we can see regions of strength for some outlier methods. In addition, the instance space can also be used for automated algorithm selection for untested instances. Given the instance space coordinates of an untested instance, we can find outlier methods suited for it by exploring the instance space. In fact, machine learning methods can be used to partition the instance space into regions where different outlier methods are dominant.

We use support vector machines (SVM) for this partitioning. For each one of the eight algorithms, we train an SVM with the label AUC ≥ 0.8 as output and the instance space coordinates as the input, using 10-fold cross-validation. Table 10 shows the results for each algorithm, plus the results of an idealized oracle that always picks the right algorithm, and our selector that combines the decisions made by the 8 SVMs. The average ROC and its standard deviation are the expected performance if we were to use each algorithm across all instances. The probability of good is the fraction of instances for which AUC ≥ 0.8. We also present the values of SVMs accuracies, precision, and recall, as well as their parameters $\{C, \gamma\}$. The results show that our selector has better performance, both in terms of precision and average ROC than using always FAST ABOD, which is the best performing algorithm. However, the difference in the performance to the Oracle indicates that the selector could be further improved.

The regions of strength resulting from this experiment are given in Fig. 14. From Fig. 14 we see an overlap of many regions. By combining these regions of strength we obtain a partitioning of the instance space shown in Fig. 15. To break ties, we use the prediction probability of the SVM and choose the method with the highest precision. One can also use a different approach such as the sensitivity to normalization criteria to break ties.

For datasets in the top-left part of Fig. 15, the precision of the SVM is quite low. This highlights the opportunity for new outlier methods that perform well in this part of the space to be developed. In addition, we see that KDEOS, which was the overall least effective method (see Fig. 5) has a niche in the instance space where no outlier method performs well. This insight was missed by the standard statistical analysis.

4 Conclusions

In this study, we have comprehensively investigated the effect of normalization and the algorithm selection problem for 14 unsupervised outlier methods. Normalization is a topic that has not received much attention in the literature. We show its relevance to outlier detection mathematically and further illustrate experimentally that performance of an outlier method may significantly change depending on the normalization method. In fact we show that the effect of normalization changes from one outlier method to another. Furthermore, certain datasets and outlier methods are more sensitive to normalization than others, creating a subtle interplay between the datasets and the outlier methods that affects their sensitivity to normalization.

Table 10 Performance of SVM prediction models based on instance space location of test sets

Outlier detection method	Average ROC	SD ROC	Probability of good	CV accuracy	CV precision	CV recall	C	γ
Ensemble Median–IQR	0.676	0.132	0.154	61.8	24.2	69.2	0.059	0.058
LOF Min–Max	0.694	0.157	0.283	58.4	35.2	55.9	0.061	0.057
KNN Median–IQR	0.702	0.155	0.295	67.3	46.7	76.6	0.033	0.032
FAST ABOD Min–Max	0.730	0.181	0.451	64.3	58.2	74.2	0.045	0.061
iForest Median–IQR	0.697	0.146	0.264	52.5	32.1	71.4	0.052	0.060
KDEOS Median–IQR	0.536	0.159	0.069	72.4	17.0	77.3	0.038	0.042
KDEOS Min–Max	0.552	0.156	0.066	93.4	–	–	0.031	0.046
LDF Min–Max	0.670	0.167	0.217	59.5	27.8	53.7	0.034	0.032
Oracle	0.870	0.049	1.000					
Selector	0.756	0.151	0.500		50.0	36.9		

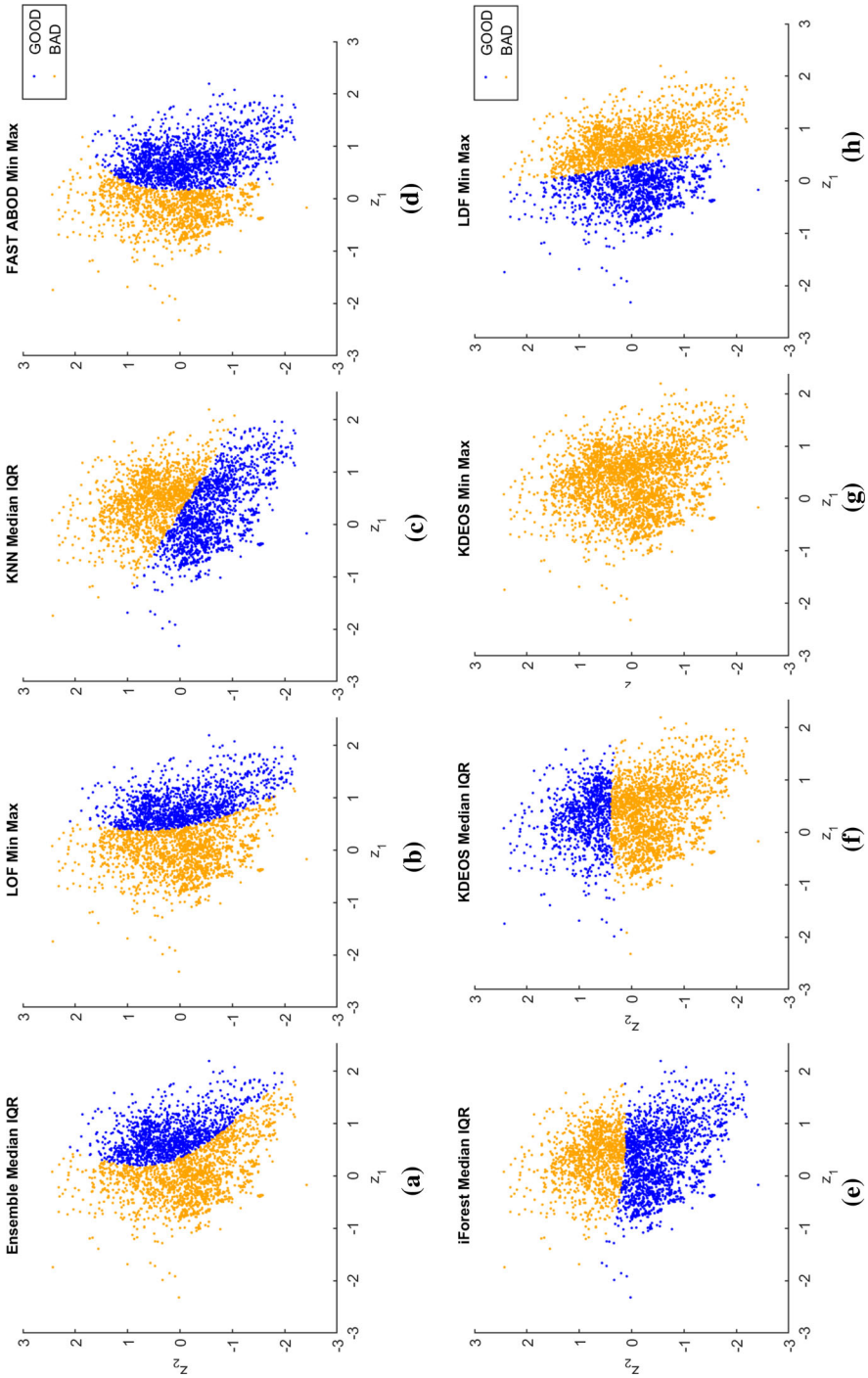

Fig. 14 Regions of strength for each algorithm according to the SVM predictions

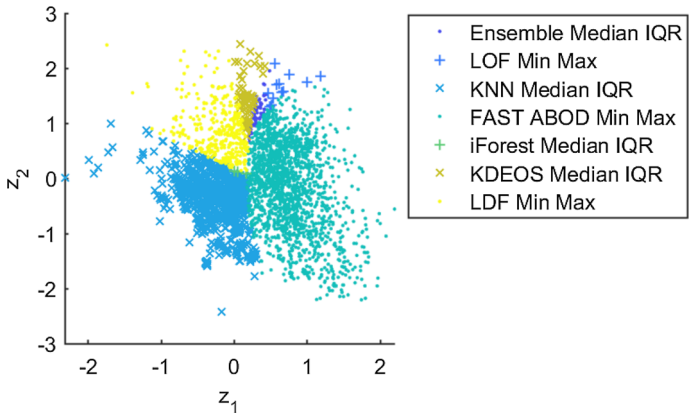

Fig. 15 A partition of the instance space showing recommended outlier detection methods

One main conclusion of this research is that normalization should not be treated as a fixed strategy, and a normalization method should be selected to maximize performance. To aid with this selection, we have proposed an approach whereby we first predict the sensitivity to normalization of a dataset, and then the normalization method best suited for a given outlier detection method. Our models predict with reasonable accuracy, with some outlier methods having higher accuracy than others.

In addition to normalization, we also investigated the algorithm selection problem for 8 different normalization-outlier method combinations. These algorithms were chosen via a clustering process so that the selected algorithms were different from each other. Using dataset features, we predict the best algorithm—normalization-outlier method combination with reasonable accuracy. We achieve better results than using any single algorithm consistently. This shows the validity of the "no free lunch" concept, where no single algorithm is suited for all problems. Furthermore, we have investigated the strengths and weaknesses of algorithms using the instance space and computed their algorithm footprints. We have shown that the algorithm KDEOS, which gave the lowest average performance has a niche in the instance space where none of the other algorithms perform well. This is the kind of insight that is hidden from standard summary statistical reporting in tables of results.

Our R package *outselect* can be used to find suitable outlier algorithms for new datasets and plot them in the instance space. This is another contribution of this work to the broader community. Additionally, this work can be used by future researchers when a new outlier detection method is developed in the following way: 1. Evaluate the sensitivity to normalization for the new outlier method. 2. Evaluate the new method using the corpus of datasets made available. This enables thorough testing of the outlier method as we have approximately 12,000 datasets of diverse characteristics. 3. Using the instance space, find the strengths and weaknesses of the new outlier method. This is useful because a new outlier method can potentially generate a footprint in parts of the instance space not yet occupied by any other method, i.e. it can have unique strengths benefiting many datasets and applications. On the other hand the instance

space can also insightfully reveal if a new outlier method is actually similar to existing outlier methods.

We note that we cannot claim our instance space to be the definitive instance space for all unsupervised outlier algorithms and datasets. Indeed, our instance space is a function of the datasets, normalization and outlier algorithms and the features we have chosen according to our experimental settings. Similar to extrapolating a curve, additional datasets, algorithms, and features have the potential to change the instance space. Future research avenues to broaden the instance space include incorporating other classes of outlier algorithms such as subspace methods, clustering-based methods, and PCA based methods. Indeed, normalization can be also thought of as a process of selecting appropriate weights for dataset attributes, which links to feature selection and subspace outlier detection. The transition between normalization and subspace outlier detection is an interesting avenue of research that is worth pursuing. In addition, the instance space can be expanded by generating even more intentionally diverse instances. Furthermore, the outlier methods we have considered, apart from iForest, use distances and densities which are computed using numerical features. As such, we converted the non-numeric attributes to numeric in our analysis. Another avenue of research is to expand the instance space to use datasets with non-numeric attributes.

The instance space methodology has been studied in other applications such as time series forecasting (Kang et al. 2017) and classification (Muñoz et al. 2018). As part of the project MATILDA (Smith-Miles 2019), a comprehensive set of tools to support instance space analysis is available at https://matilda.unimelb.edu.au/. All meta-data and code to reproduce the results presented in this paper are available for download at https://matilda.unimelb.edu.au/matilda/problems/learning/anomaly_detection, enabling this study to be expanded over time as new outlier detection algorithms are proposed.

Supplementary Material

R package outselect This package contains the functionality to reproduce graphs and computation in this manuscript, apart from the instance space generation.
Instance space scripts The code used for instance space analysis is available at the GitHub repository Muñoz (2019).
Datasets Datasets are available at Kandanaarachchi et al. (2019b).
Scripts The script Supp_Mat_1.R contains the R code that uses *outselect* in Sections 2 and 3. The script Supp_Mat_2.R contains the R code for Section 2, using area under the pre cision recall curve instead of area under ROC.
Other R-packages We have used the following R-packages either in this paper or within the package *outselect*: *randomForest* (Liaw and Wiener 2002), *infotheo* (Meyer 2014), *moments* (Komsta and Novomestky 2015), *ks* (Duong 2018), *igraph* (Csardi and Nepusz 2006), *e1071* (Meyer et al. 2018), *ggplot2* (Wickham 2016), *quantmod* (Ryan and Ulrich 2018), *dbscan* (Hahsler and Piekenbrock 2018), *FNN* (Beygelzimer et al. 2018), *cluster* (Maechler et al. 2018), *lme4* (Bates et al. 2015), *reshape* and *reshape2* (Wickham 2007), *multcomp* (Hothorn et al. 2008), *tsutils* (Kourentzes 2019), *visreg*

(Breheny and Burchett 2017), *latex2exp* (Meschiari 2015) and *pROC* (Robin et al. 2011).

Acknowledgements Funding was provided by the Australian Research Council through the Australian Laureate Fellowship FL140100012, and Linkage Project LP160101885. This research was supported in part by the Monash eResearch Centre and eSolutions-Research Support Services through the MonARCH HPC Cluster.

References

Achtert E, Kriegel H-P, Zimek A (2008) Elki: a software system for evaluation of subspace clustering algorithms. In: International conference on scientific and statistical database management. Springer, pp 580–585

Angiulli F, Pizzuti C (2002) Fast outlier detection in high dimensional spaces. In: European conference on principles of data mining and knowledge discovery. Springer, pp 15–27

Barnett V, Lewis T (1974) Outliers in statistical data. Wiley, Hoboken

Bates D, Mächler M, Bolker B, Walker S (2015) Fitting linear mixed-effects models using lme4. J Stat Softw 67(1):1–48

Beygelzimer A, Kakadet S, Langford J, Arya S, Mount D, Li S (2018) FNN: Fast nearest neighbor search algorithms and applications. R package version 1.1.2.2. https://CRAN.R-project.org/package=FNN

Billor N, Hadi AS, Velleman PF (2000) Bacon: blocked adaptive computationally efficient outlier nominators. Comput Stat Data Anal 34(3):279–298

Bischl B, Mersmann O, Trautmann H, Preuß M (2012) Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In: Proceedings of the 14th annual conference on genetic and evolutionary computation. ACM, pp 313–320

Brazdil P, Giraud-Carrier C, Soares C, Vilalta R (2008) Metalearning: applications to data mining. Springer, Berlin

Breheny P, Burchett W (2017) Visualization of regression models using visreg. R J 9(2):56–71

Breunig MM, Kriegel H-P, Ng RT, Sander J (2000) LOF: identifying density-based local outliers. In: ACM sigmod record, vol 29. ACM, pp 93–104

Campos GO, Zimek A, Sander J, Campello RJ, Micenková B, Schubert E, Assent I, Houle ME (2016) On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. Data Min Knowl Discov 30(4):891–927

Craswell N (2009) Precision at n. Springer, Boston, pp 2127–2128

Csardi G, Nepusz T (2006) The igraph software package for complex network research. InterJ Complex Syst 1695(5):1–9

Culberson JC (1998) On the futility of blind search: an algorithmic view of "no free lunch". Evol Comput 6(2):109–127

Davis J, Goadrich M (2006) The relationship between Precision–Recall and ROC curves. In: Proceedings of the 23rd international conference on machine learning. ACM, pp 233–240

Duong T (2018) ks: Kernel smoothing. R package version 1.11.3. https://CRAN.R-project.org/package=ks

Emmott A, Das S, Dietterich T, Fern A, Wong W-K (2015) A meta-analysis of the anomaly detection problem. ArXiv preprint arXiv:1503.01158

Emmott AF, Das S, Dietterich T, Fern A, Wong W-K (2013) Systematic construction of anomaly detection benchmarks from real data. In: Proceedings of the ACM SIGKDD workshop on outlier detection and description. ACM, pp 16–21

Goix N (2016) How to evaluate the quality of unsupervised anomaly detection algorithms? arXiv preprint arXiv:1607.01152

Goldstein M, Uchida S (2016) A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PLoS ONE 11(4):e0152173

Hahsler M, Piekenbrock M (2018) dbscan: Density based clustering of applications with noise (DBSCAN) and related algorithms. R package version 1.1-3. https://CRAN.R-project.org/package=dbscan

Hautamaki V, Karkkainen I, Franti P (2004) Outlier detection using k-nearest neighbour graph. In: Proceedings of the 17th international conference on pattern recognition, ICPR 2004, vol 3. IEEE, pp 430–433

Hawkins DM (1980) Identification of outliers, vol 11. Springer, Berlin
Ho Y-C, Pepyne DL (2002) Simple explanation of the no-free-lunch theorem and its implications. J Optim Theory Appl 115(3):549–570
Hothorn T, Bretz F, Westfall P (2008) Simultaneous inference in general parametric models. Biom J 50(3):346–363
Hubert M, Van der Veeken S (2008) Outlier detection for skewed data. J Chemom 22(3–4):235–246
Igel C, Toussaint M (2005) A no-free-lunch theorem for non-uniform distributions of target functions. J Math Modell Algorithms 3(4):313–322
Jin W, Tung AK, Han J, Wang W (2006) Ranking outliers using symmetric neighborhood relationship. In: Pacific-Asia conference on knowledge discovery and data mining. Springer, pp 577–593
Kandanaarachchi S (2018) Outselect: algorithm selection for unsupervised outlier detection. R package version 0.0.0.9000. https://github.com/sevvandi/outselect
Kandanaarachchi S, Munoz MA, Smith-Miles K (2019) Instance space analysis for unsupervised outlier detection. In: Proceedings of the 1st workshop on evaluation and experimental design in data mining and machine learning co-located with siam international conference on data mining (SDM 2019), Calgary, Alberta, Canada, May 4th, 2019, pp 32–41. http://ceur-ws.org/Vol-2436/article_4.pdf
Kandanaarachchi S, Muñoz MA, Smith-Miles K, Hyndman R (2019) Datasets for outlier detection. https://monash.figshare.com/articles/Datasets_12338_zip/7705127/4
Kang Y, Hyndman R, Smith-Miles K (2017) Visualising forecasting algorithm performance using time series instance spaces. Int J Forecast 33(2):345–358
Komsta L, Novomestky F (2015) Moments: moments, cumulants, skewness, kurtosis and related tests. R package version 0.14. https://CRAN.R-project.org/package=moments
Kourentzes N (2019) tsutils: time series exploration, modelling and forecasting. R package version 0.9.0. https://CRAN.R-project.org/package=tsutils
Kriegel H-P, Kröger P, Schubert E, Zimek A (2009) LoOP: local outlier probabilities. In: Proceedings of the 18th ACM conference on information and knowledge management. ACM, pp 1649–1652
Kriegel H-P, Schubert M, Zimek A (2008) Angle-based outlier detection in high-dimensional data. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 444–452
Latecki LJ, Lazarevic A, Pokrajac D (2007) Outlier detection with kernel density functions. In: International workshop on machine learning and data mining in pattern recognition. Springer, pp 61–75
Leigh C, Alsibai O, Hyndman RJ, Kandanaarachchi S, King OC, McGree JM, Neelamraju C, Strauss J, Talagala PD, Turner RD et al (2019) A framework for automated anomaly detection in high frequency water-quality data from in situ sensors. Sci Total Environ 664:885–898
Leyton-Brown K, Nudelman E, Andrew G, McFadden J, Shoham Y (2003) A portfolio approach to algorithm selection. In: 2003 International joint conference on artificial intelligence (IJCAI), vol 3. pp 1542–1543
Liaw A, Wiener M (2002) Classification and regression by randomForest. R News 2(3):18–22
Liu FT (2009) Isolationforest: Isolation forest. R package version 0.0-26/r4. https://R-Forge.R-project.org/projects/iforest/
Liu FT, Ting KM, Zhou Z-H (2008) Isolation forest. In: 2008 Eighth IEEE international conference on data mining. IEEE, pp 413–422
Maechler M, Rousseeuw P, Struyf A, Hubert M, Hornik K (2018) Cluster: cluster analysis basics and extensions. R package version 2.0.7-1
Meschiari S (2015) latex2exp: Use LaTeX expressions in plots. R package version 0.4.0. https://CRAN.R-project.org/package=latex2exp
Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F (2018) e1071: Misc functions of the department of statistics, probability theory group (formerly: E1071), TU Wien. R package version 1.7-0. https://CRAN.R-project.org/package=e1071
Meyer PE (2014) Infotheo: information-theoretic measures. R package version 1.2.0. https://CRAN.R-project.org/package=infotheo
Muñoz MA (2019) Instance space analysis: a toolkit for the assessment of algorithmic power. https://github.com/andremun/InstanceSpace
Muñoz MA, Villanova L, Baatar D, Smith-Miles K (2018) Instance spaces for machine learning classification. Mach Learn 107(1):109–147
Peng Y, Flach PA, Soares C, Brazdil P (2002) Improved dataset characterisation for meta-learning. In: International conference on discovery science. Springer, pp 141–152

Pfahringer B, Bensusan H, Giraud-Carrier CG (2000) Meta-learning by landmarking various learning algorithms. In: International conference on machine learning (ICML), pp 743–750
Ramaswamy S, Rastogi R, Shim K (2000) Efficient algorithms for mining outliers from large data sets. In: ACM sigmod record, vol 29. ACM, pp 427–438
Rice J (1976) The algorithm selection problem. In: Advances in computers, vol 15. Elsevier, pp 65–118
Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez J-C, Müller M (2011) pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinform 12:77
Rousseeuw PJ, Hubert M (2017) Anomaly detection by robust statistics. Wiley Interdiscip Rev Data Min Knowl Discov 8:e1236
Ryan JA, Ulrich JM (2018) quantmod: Quantitative financial modelling framework. R package version 0.4-13. https://CRAN.R-project.org/package=quantmod
Schubert E, Zimek A, Kriegel H-P (2014a) Generalized outlier detection with flexible kernel density estimates. In: Proceedings of the 2014 SIAM international conference on data mining. SIAM, pp 542–550
Schubert E, Zimek A, Kriegel H-P (2014b) 'Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection'. Data Min Knowl Discov 28(1):190–237
Smith-Miles K (2019) MATILDA: melbourne algorithm test instance library with data analytics. https://matilda.unimelb.edu.au
Smith-Miles KA (2009) Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Comput Surv (CSUR) 41(1):6
Smith-Miles K, Baatar D, Wreford B, Lewis R (2014) Towards objective measures of algorithm performance across instance space. Comput Oper Res 45:12–24
Smith-Miles K, Bowly S (2015) Generating new test instances by evolving in instance space. Comput Oper Res 63:102–113
Smith-Miles K, Tan TT (2012) Measuring algorithm footprints in instance space. In: 2012 IEEE congress on evolutionary computation. IEEE, pp 3446–3453
Talagala PD, Hyndman RJ, Smith-Miles K, Kandanaarachchi S, Munoz MA (2019) Anomaly detection in streaming nonstationary temporal data. J Comput Graph Stat. https://doi.org/10.1080/10618600.2019.1617160
Tang J, Chen Z, Fu AW-C, Cheung DW (2002) Enhancing effectiveness of outlier detections for low density patterns. In: Pacific-Asia conference on knowledge discovery and data mining. Springer, pp 535–548
Wickham H (2007) Reshaping data with the reshape package. J Stat Softw 21(12):1–20
Wickham H (2016) ggplot2: Elegant graphics for data analysis. Springer, New York. http://ggplot2.org
Wilkinson L (2018) Visualizing big data outliers through distributed aggregation. IEEE Trans Vis Comput Graph 24(1):256–266
Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evolut Comput 1(1):67–82
Wolpert DH, Macready WG et al (1995) No free lunch theorems for search. Technical report, SFI-TR-95-02-010, Santa Fe Institute
Zhang E, Zhang Y (2009) Average precision. In: Encyclopedia of database systems. Springer, Berlin, pp 192–193
Zhang K, Hutter M, Jin H (2009) A new local distance-based outlier detection approach for scattered real-world data. In: Pacific-Asia conference on knowledge discovery and data mining. Springer, pp 813–822
Zimek A, Schubert E, Kriegel H-P (2012) A survey on unsupervised outlier detection in high-dimensional numerical data. Stat Anal Data Min ASA Data Sci J 5(5):363–387