ISSN 1440-771X

On normalization and algorithm selection for unsupervised outlier detection

Sevvandi Kandanaarachchi, Mario A Muñoz, Rob J Hyndman and Kate Smith-Miles

September 2018

Working Paper 16/18

On normalization and algorithm selection for unsupervised outlier detection

Sevvandi Kandanaarachchi, Mario A. Muñoz, Rob J. Hyndman, Kate Smith-Miles

September 13, 2018

Abstract

This paper demonstrates that the performance of various outlier detection methods depends sensitively on both the data normalization schemes employed, as well as characteristics of the datasets. Recasting the challenge of understanding these dependencies as an algorithm selection problem, we perform the first instance space analysis of outlier detection methods. Such analysis enables the strengths and weaknesses of unsupervised outlier detection methods to be visualized and insights gained into which method and normalization scheme should be selected to obtain the most likely best performance for a given dataset.

1 Introduction

An increasingly important challenge in a data-rich world is to efficiently analyze datasets for patterns of regularity and predictability, and to find outliers that deviate from the expected patterns. The significance of detecting such outliers with high accuracy, minimizing costly false positives and dangerous false negatives, is clear when we consider just a few societally critical examples of outliers: e.g. fraudulent credit card transactions amongst billions of legitimate ones, fetal anomalies in pregnancies, chromosomal anomalies in tumours, 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 methods 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 & Macready 1997, Culberson 1998, Ho & Pepyne 2002, Igel & Toussaint 2005). As such, it is advantageous to know the strengths and weaknesses of any method, and how specific properties of a dataset might render it more or less ideally suited to detect outliers than other methods. What kinds of properties would enable a given method to perform well on one dataset, but maybe poorly on another? How sensitive are the existing methods to variations in dataset characteristics? How can we objectively evaluate a portfolio of outlier detection methods to learn these relationships? Given a problem can we learn to predict the best-suited outlier detection method(s)? And given that normalization of a dataset is a typical pre-processing step adopted by all outlier detection methods, but rescaling the data can change the relationships between the data points, what impact does a normalization scheme have on outlier detection accuracy? These are some of the questions that motivate this work.

When evaluating outlier detection methods, an important issue that needs to be considered is the definition of an outlier - according to both an algorithm's definition of an outlier and a human who may have labelled 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 & Lewis (1974) define an outlier as *an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data*. Both these definitions indicate that outliers are quite

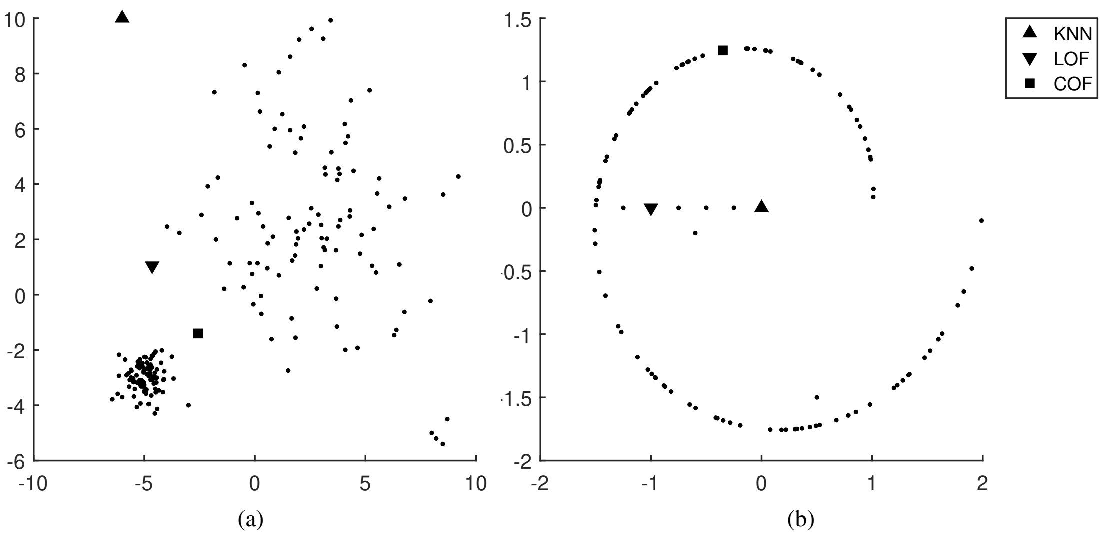


Figure 1: First outlier detected in two different datasets by three different methods: KNN (▲), LOF (▼) and COF (■).

different from non-outlying observations. Barnett & Lewis (1974) also note that it is a *a matter of subjective judgement 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 judgement, but extends to differences in how outlier detection methods define an outlier. Indeed, there are many instances where a set of outlier detection methods may not agree on the location of outliers due to their different definitions, whether they be related to nearest neighbour distances, density arguments or other quantitative metrics. Figure 1 illustrates the lack of consensus of three popular outlier detection methods namely, KNN (Ramaswamy et al. 2000), LOF (Breunig et al. 2000) and COF (Tang et al. 2002), and highlights the opportunity to exploit knowledge of the combination of dataset characteristics and the algorithm's definition of an outlier to enhance selection of the most suitable method.

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 neighbours using the ELKI software suite (Achtert et al. 2008). While the methods considered all used a similar nearest neighbor distance definition of an outlier, the study is relevant to ours since it contributed a useful repository of around 1000 benchmark datasets generated by modifying 23 source datasets that can be used for further outlier detection analysis. It is common practice to test outlier detection algorithms on datasets with known ground truth labels, for example classification datasets where observations of the minority class have been down-sampled. We will be extending their approach to dataset generation for our comprehensive experimental study. Goldstein & Uchida (2016) conducted a comparative evaluation of 19 unsupervised outlier detection methods, which fall into three categories, namely nearest neighbour based, clustering based, and based on other algorithms such as one class SVM and robust PCA. They have used 10 datasets for their evaluation. Their algorithms are released on RapidMiner data mining software. Emmott et al. (2015) conducted a meta-analysis of 12 outlier detection methods, which fall into four categories; namely nearest neighbours based, density based, model based or projection based. These studies focus on the evaluation of outlier detection methods, which is much needed in the contemporary literature due to the sheer volume of new methods being developed. However, they do not address the critical algorithm selection problem for outlier detection, i.e. 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 & Bowly 2015, Muñoz et al. 2018). We will use this framework to gain an understanding of strengths and weaknesses of the outlier detection methods discussed by Campos et al. (2016).

In addition to tackling the algorithm selection problem for outlier detection for the first time, we will also focus on a topic that is generally over-looked; namely normalization. One of the main pre-processing steps in outlier detection is normalizing or standardizing the data. Traditionally min-max normalization method, which normalizes each column of a dataset to the interval $[0, 1]$ is used routinely in outlier detection (Campos et al. 2016, Goldstein & Uchida 2016). However, there are many different methods that can be used for normalizing or standardizing the data. Whether the choice of normalization method impacts the effectiveness of the outlier detection method is a question which has not been given much attention. In fact, we have not come across any studies which focus on the effect of normalization on outlier detection. 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, we make available a repository of more than 12000 datasets, which is generated from approximately 200 source datasets, providing a comprehensive basis for future evaluation of outlier detection methods.

This paper is organized as follows. We start by investigating the impact of normalization on outlier detection methods in Section 2. Firstly, from a theoretical perspective we present mathematical arguments in Section 2.1 to show how various normalization schemes can change the nearest neighbours and densities of the data, and hence why we intuitively expect that the impact of normalization can be significant depending on the definition of an outlier adopted by an algorithm. In Section 2.2 we present comprehensive experimental evidence that this theoretical sensitivity is observed in practice across a set of over 12000 datasets. We show that both the normalization method and the outlier detection method, in combination, have variable performance across the datasets, suggesting that some datasets possess properties that some methods can exploit well, while others are not as well suited. 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 outlier detection methods. 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 performance of outlier detection methods. 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 Section 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, 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 general 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 has not been studied to the best of our knowledge. As such, we investigate the effect of four normalization/standardization methods of outlier detection.

1. Minimum and maximum normalization (Min-Max)

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

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.

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

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

4. Median and median absolute deviation normalization (Median-MAD)
   Here MAD(x) = 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 & Hubert (2017).

Generally, normalization scales axes differently causing some axes to compress and some axes to expand, thus changing the nearest neighbour structure. As nearest neighbour 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

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

Here $\boldsymbol{x}_i^*$ is the normalized observation, μ 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

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

giving us

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

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

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

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

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

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 equation (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

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

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 neighbours

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

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

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

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

If $\boldsymbol{x}_{l_1}^*$ is the nearest neighbour 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 $\boldsymbol{x}_{l_{k-1}}^*$ is the $(k-1)^{\text{st}}$ nearest neighbour of $\boldsymbol{x}_i^*$ and $A_{ik} = A_{i(k-1)} \backslash l_{k-1}$. Proceeding in a similar way, the k^{th} nearest neighbour 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 neighbour distance as outliers we write an expression for the point with the highest knn distance:

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

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 neighbour computations.

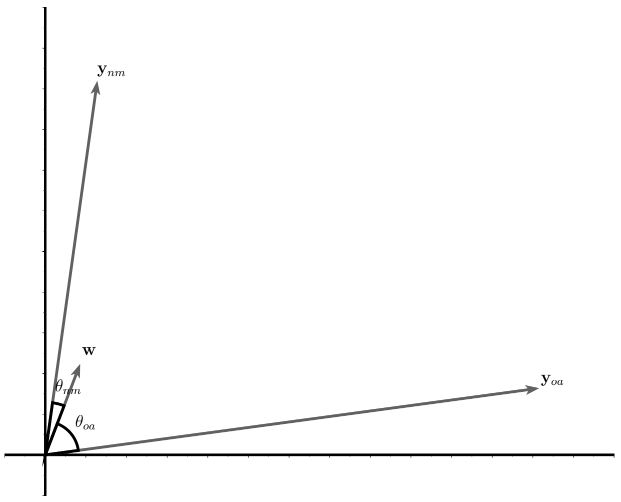


Figure 2: The vectors $\boldsymbol{y}_{oa}$, $\boldsymbol{y}_{nm}$, $\boldsymbol{w}$ with angles θ_{oa} and θ_{nm}.

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 neighbours 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*

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

then

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

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

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

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

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

From equation (11) we have

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

and from equation (12) we have

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

Dividing equation (13) from (14) we obtain

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

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

As illustrated in Figure 2 the angle between the normalization vector $\boldsymbol{w}$ and $\boldsymbol{y}_{nm}$ has an effect in the ordering of k-nearest neighbour distances. Thus the normalization vector $\boldsymbol{w}$ can mask outliers and favour non-outliers, reducing the performance of outlier detection methods.

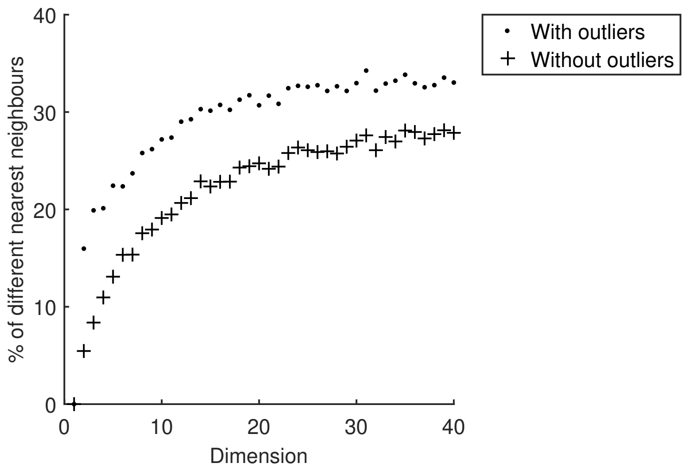


Figure 3: Percentage of observations that do not have the same nearest neighbour after normalizing using the above four methods.

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

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

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

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

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.

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 was scaled using the four normalization methods. For each normalized dataset the nearest neighbour was computed for each observation, and we calculate the percentage of observations that do not have the same nearest neighbour 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 same process. The graphs in figure 3 show the average percentage of observations that do not have the same nearest neighbour 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 neighbours due to normalization increase 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 neighbours of 25% of the data depend on the method of normalization. Similarly, for a 20-dimensional dataset with one outlier, the nearest neighbours 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 neighbours 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 12 outlier detection methods coupled with the above-mentioned 4 normalization methods, across a set of over 12000 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 & Uchida 2016), which takes a classification dataset and down-samples the minority class to label outliers. Campos et al. (2016) start with 23 datasets, from which different variants are obtained mainly by downsampling the outlier class at rates 20%, 10%, 5%, 2% and transforming categorical variables to numeric values. This process results in approximately 1000 datasets. Goldstein & Uchida (2016) use 10 datasets for their evaluation study, with some overlap with Campos et al. (2016). In order to obtain a more comprehesive and diverse set of benchmark test datasets, we extend the approach to utilise 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. Labelling of outliers - as outliers are rare events, the proportion of outliers is typically 5% or less for most outlier datasets. In contrast, the 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 k 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 $k-1$ 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 are 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 $IDF(x) = \ln(N/n_x)$ 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 neighbour 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 are 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 & Uchida 2016), along with the datasets we prepared from Muñoz et al. (2018), our final set of benchmarks for this experimental study contains approximately 12200 datasets suitable for outlier detection evaluation.

2.2.2 Outlier detection methods

We investigate the 12 outlier detection methods studied in Campos et al. (2016) using the ELKI software suite (Achtert et al. 2008). The methods are:

1. KNN - K nearest neighbours (Ramaswamy et al. 2000)
2. KNNW - KNN weight (Angiulli & 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. 2014*b*)
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. 2014*a*)
12. FastABOD - Fast Angle Based Outlier Detection (ABOD), faster version of ABOD (Kriegel et al. 2008)

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 & Uchida (2016).

The parameter that is common to all the above outlier detection methods, and as such deserves some discussion is the value k in k-nearest neighbours. Even though the effect of k is different across methods, the common theme is the k-nearest neighbourhood. The choice of k affects the performance of the method as shown in the following example.

Consider a dataset of 100 observations containing one outlier as shown in figure 4a. For this dataset, we find outlier scores using KNN and LOF for all observations for each value of k from 1 to 99. An outlier can be easily detected if the outlier has a higher score compared to non-outliers. So, for both KNN and LOF we compute the following outlier score ratio:

$$\text{outlier score ratio} = \frac{\text{outlier score of outlier}}{\text{average outlier score of non-outliers}}$$

From figure 4b we see that as k increases, this ratio follows a downward trend for both KNN and LOF. Therefore, noting that no single k would be applicable for all methods and datasets, and to ensure a fair evaluation of methods, 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:

$$k(\text{dataset}) = \min(\text{floor}(5\% \text{ of observations }), 100) \tag{18}$$

Here the maximum of $k = 100$ is a means of limiting the number of computations that can result from a large dataset.

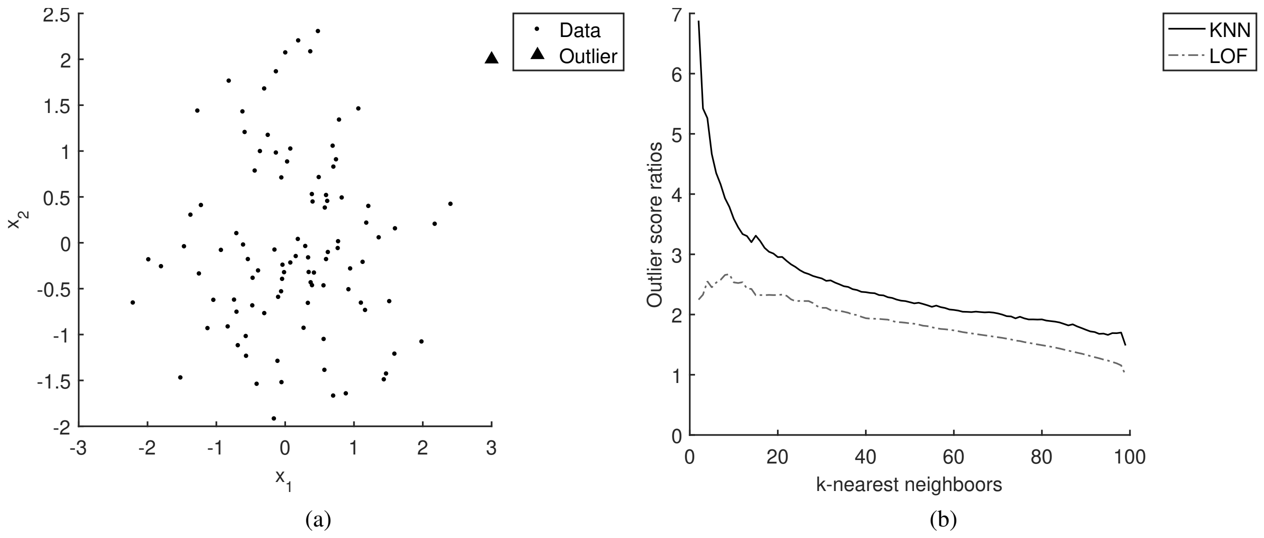


Figure 4: The dataset is plotted in figure (a) with the outlier at $(3, 2)$. The outlier score ratios are plotted in figure (b) for both KNN and LOF. We see the outlier score ratio decrease with increasing k for most k values.

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 very 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. 2014*b*, 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 & Van der Veeken 2008, Wilkinson 2018, Talagala et al. 2018, 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 area under the ROC is more widely used than PR curves. 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 & 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.

2.3 Hypothesis testing

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:

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

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 μ denotes the intercept, c_i the coefficient of the i^{th} outlier detection method, d_j the coefficient of the j^{th} 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 modelled as random variables, i.e. $h_k \sim N\left(0, \sigma_h^2\right)$. The errors ε_{ijkl} are assumed to be normally distributed, i.e. $\varepsilon_{ijkl} \sim N\left(0, \sigma_\varepsilon^2\right)$.

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:

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

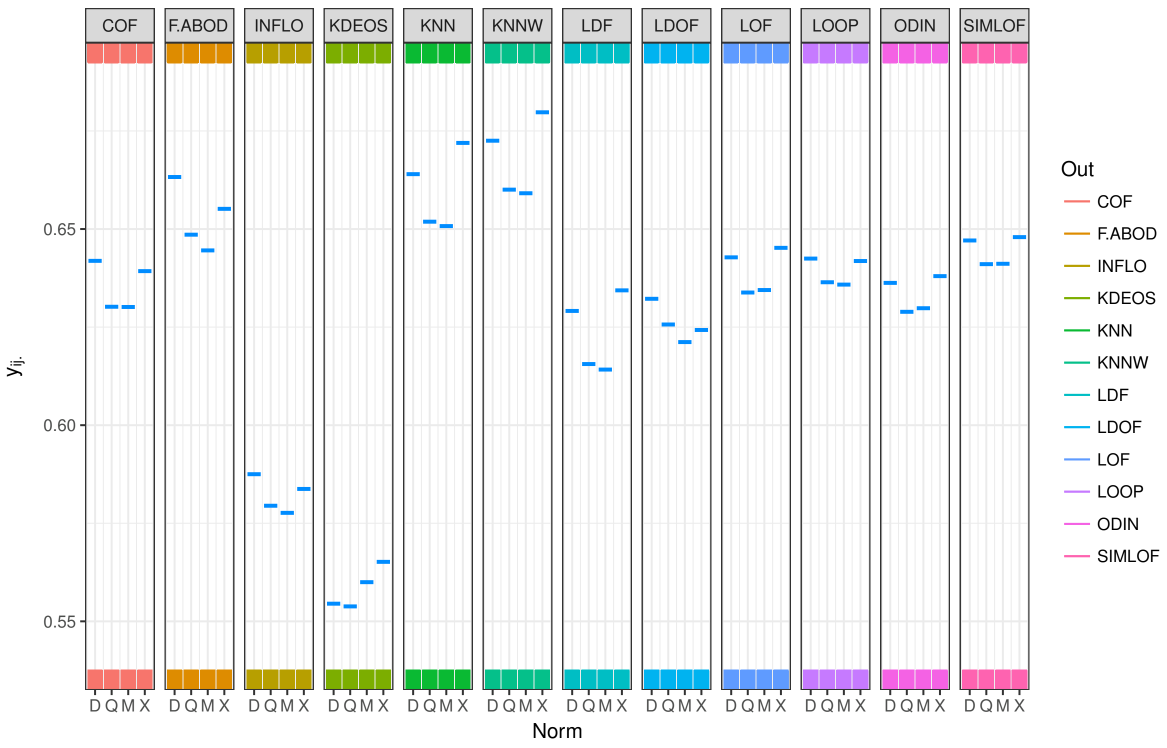
The difference between the first and the second model results from the interaction term, which gives rise to a separate regression coefficient g_{ij} for each pair of outlier and normalization methods, rather than assuming their effects are additive. The second model can be used to determine if normalization 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 favour 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 & Burchett (2012). The letters D, Q, M and X denote the normalization methods Mean-SD, Median-IQR, Median-MAD and Min-Max respectively. The plotted value for each normalization and outlier method is $y_{ij.} = \mu + g_{ij} + \bar{h}$ from equation (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 Figure 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 Figure 5.

1. KNNW has the highest $y_{ij.}$ values, making it the most effective outlier method on average.
2. The three best outlier methods are KNNW, KNN and FAST ABOD.
3. KDEOS has the lowest $y_{ij.}$ values, making it the least effective outlier method on average. The second least effective outlier method is INFLO.
4. For most outlier methods, Min-Max and Mean-SD outperform Median-IQR and Median-MAD.
5. 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.
6. LOF, LOOP and SIMLOF are quite similar in terms of $y_{ij.}$.
7. The effect of the outlier method on $y_{ij.}$is greater than the effect of the normalization method.

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.



(a)

Figure 5: Effect of normalization on outlier detection methods based on model (22). Here, $y_{ij.} = g_{ij} + \bar{h}$ is plotted for normalization methods Mean-SD (D), Median-IQR (Q), Median-MAD (M) and Min-Max (X) for each outlier method. Higher values indicate better performance.

2.4 Preferred normalization methods

In this section we investigate whether each outlier detection method has a preferred normalization method, independent of the dataset. The results of this analysis are given in Table 1. Table 1 gives the percentage of datasets which prefers Min-Max or Median-IQR for each outlier detection method. For a given outlier detection method, we say a dataset prefers a normalization method, if that method gives a higher performance value than other normalization methods.

By inspecting Table 1 we see that 1. there is not much difference between the percentages of datasets that prefer Min-Max to Median-IQR, 2. for all outlier methods apart from INFLO and LDOF the percentage of datasets that prefer Min-Max is higher than that of Median-IQR, although non reach 60%, and 3. it is only for KNN, KNNW and FAST ABOD that this difference is greater than 10%. For all other methods this difference is less than 4%.

As Min-Max is more often preferred than Median-IQR, this analysis somewhat validates the historical preference to use Min-Max for outlier detection. However, for 40–50% of our datasets, Median-IQR was the preferred method. This again brings to light the importance of normalization when performing outlier detection.

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

Table 1: Percentage of datasets for which Median-IQR or Min-Max gives better performance for each outlier method.

Outlier detection method	Median-IQR better performance (%)	Min-Max better performance (%)
COF	48.70	51.30
FAST ABOD	44.64	55.36
INFLO	50.98	49.02
KDEOS	49.05	50.95
KNN	42.84	57.16
KNNW	43.38	56.62
LDF	48.98	51.02
LDOF	51.09	48.91
LOF	49.00	51.00
LOOP	49.34	50.66
ODIN	48.15	51.85
SIMLOF	48.89	51.11

Table 2: Sensitivity to normalization grouped by outlier method

Outlier detection method	$\xi = 0.05$	$\xi = 0.10$	$\xi = 0.15$	$\xi = 0.20$
COF	7956 (68%)	4927 (52%)	2860 (25%)	1617 (14%)
FAST_ABOD	8977 (77%)	5778 (50%)	3643 (31%)	2356 (20%)
INFLO	7366 (63%)	4290 (37%)	2459 (21%)	1473 (13%)
KDEOS	8171 (70%)	5443 (47%)	3453 (30%)	2212 (19%)
KNN	6057 (52%)	3024 (26%)	1709 (15%)	1050 (9%)
KNNW	5545 (48%)	2803 (24%)	1544 (13%)	926 (8%)
LDF	7581 (65%)	4975 (43%)	3241 (28%)	2168 (19%)
LDOF	6687 (58%)	3782 (33%)	2130 (18%)	1186 (10%)
LOF	7028 (61%)	4126 (36%)	2382 (21%)	1377 (12%)
LOOP	6502 (56%)	3543 (31%)	1872 (16%)	1037 (9%)
ODIN	6705 (58%)	3640 (31%)	1899 (16%)	990 (9%)
SIMLOF	6325 (54%)	3427 (30%)	1796 (15%)	989 (9%)

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 dataset, we say that an outlier detection method is $\xi-$sensitive to normalization if the difference between the maximum performance and the minimum performance across all normalization schemes for that outlier detection method is greater than ξ.*

We use this definition with $\xi = 0.05, 0.10, 0.15$ and 0.20 to investigate the effects of normalization. Table 2 reports the number of our datasets that are $\xi-$ sensitive to normalization for each outlier detection method. By definition 2.2 the number of datasets sensitive to normalization decreases as ξ increases.

By looking at Table 2 we seek to identify if there are common datasets that are sensitive to normalization for multiple outlier detection methods. To this end, we compute the number of datasets that are sensitive to normalization for exactly n outlier methods for $n \in \{0, 1, 2, \ldots, 12\}$. Table 3 summarizes these results for $\xi = 0.05, 0.10, 0.15$ and 0.20. For $\xi = 0.05$ we see that 2029 datasets are sensitive to normalization for all 12 outlier methods and 412 datasets are not sensitive to normalization for any outlier method. In addition to these two extremes, there are different numbers of datasets sensitive to normalization for n number of outlier methods for $n \in \{1, \ldots, 11\}$. In particular, there are 1002 datasets that are sensitive to normalization for exactly 1 outlier method. Similarly there are 656 datasets that are

Table 3: Number of datasets that are ξ-sensitive to normalization for n outlier methods.

Number of outlier methods	$\xi = 0.05$	$\xi = 0.10$	$\xi = 0.15$	$\xi = 0.20$
0	412 (3.5%)	1977 (17.0%)	4087 (35.2%)	5966 (51.3%)
1	1002 (8.6%)	2009 (17.2%)	2350 (20.2%)	2330 (20%)
2	656 (5.6%)	1142 (9.8%)	1182 (10.2%)	939 (8.1%)
3	627 (5.4%)	899 (7.7%)	854 (7.4%)	584 (5.0%)
4	692 (6.0%)	719 (6.2%)	626 (5.4%)	426 (3.7%)
5	573 (4.9%)	689 (5.9%)	501 (4.3%)	361 (3.1%)
6	530 (4.6%)	697 (6.0%)	441 (3.8%)	281 (2.4%)
7	695 (6.0%)	671 (5.8%)	398 (3.4%)	230 (2.0%)
8	839 (7.2%)	672 (5.8%)	322 (2.8%)	154 (1.3%)
9	994 (8.6%)	642 (5.5%)	295 (2.5%)	152 (0.3%)
10	1205 (10.3%)	532 (4.6%)	231 (2.0%)	111 (1.0%)
11	1361 (11.7%)	498 (4.3%)	206 (1.8%)	66 (0.6%)
12	2029 (17.5%)	468 (4.0%)	122 (1.1%)	15 (0.1%)

Table 4: Unique sensitivity of outlier detection methods to normalization: the number of datasets sensitive to normalization for exactly one outlier method grouped by the outlier method. Each method may additionally have shared sensitivities to other datasets with other methods.

Outlier detection method	$\xi = 0.05$	$\xi = 0.10$	$\xi = 0.15$	$\xi = 0.20$
COF	62 (6.2%)	205 (10.2%)	264 (11.2%)	224 (9.6%)
FAST ABOD	691 (69.0%)	964 (48.0%)	913 (38.9%)	888 (38.1%)
INFLO	48 (4.8%)	72 (3.6%)	127 (5.4%)	122 (5.2%)
KDEOS	147 (14.7%)	424 (21.1%)	594 (25.3%)	585 (25.1%)
KNN	4 (0.4%)	6 (0.3%)	15 (0.6%)	31 (1.3%)
KNNW	1 (0.1%)	2 (0.1%)	3 (0.1%)	3 (0.1%)
LDF	28 (2.8%)	209 (10.4%)	259 (11.0%)	273 (11.7%)
LDOF	10 (1.0%)	40 (2.0%)	45 (1.9%)	66 (2.8%)
LOF	0 (0.0%)	17 (0.8%)	36 (1.5%)	46 (2.0%)
LOOP	4 (0.4%)	9 (0.4%)	8 (0.3%)	14 (0.6%)
ODIN	6 (0.6%)	61 (3.0%)	84 (3.6%)	71 (3.0%)
SIMLOF	1 (0.1%)	0 (0.0%)	2 (0.1%)	7 (0.3%)
Total	1002	2009	2350	2330

sensitive to normalization for exactly 2 outlier methods for $\xi = 0.05$. Again, each dataset may have a different combination of outlier methods that are sensitive to normalization.

From Table 3 we observe a subtle interplay of dataset characteristics and outlier detection methods affecting the sensitivity to normalization. In order to understand which outlier methods are more sensitive to normalization, we examine the datasets which are sensitive to normalization for only one outlier method. Table 4 contains these results.

From Table 4 we see that FAST ABOD is the method most sensitive to normalization followed by KDEOS. This is consistent for $\xi = 0.05, 0.10, 0.15$ and 0.20. The outlier detection methods KNNW and SIMLOF are the least sensitive to normalization with LOOP and KNN and achieving comparable results. This outcome further validates the results of the second mixed model in section 2.3 given by equation (21). In other words, we explicitly see evidence of normalization affecting outlier detection methods differently.

This section has provided comprehensive evidence, both theoretical and experimental, that normalization can have 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

We use the extended methodology developed by Smith-Miles et al. (2014), which is based on the Algorithm Selection Problem framework proposed by Rice (1976). This methodology enables the visualization of the broadest possible test instance space, beyond the test instances under study, and how the difficulty of the instances and performance of algorithms varies across the instance space. Analysis within this instance space enables an objective measurement of algorithmic power, and the strengths and weaknesses of algorithms to be visualized.

Figure 6 illustrates the framework and its component spaces. The first 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 projects an instance x from a high-dimensional feature space to a two-dimensional space, which is called the *instance space*. The methods used to learn this mapping, and to project from a high-dimensional feature space to a $2 - d$ 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, hence 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 meta-data from which we now construct the outlier detection instance space is described by the problem instances (see datasets in Section 2.2.1), the algorithms (see outlier detection methods in Section 2.2.2), and the performance metric described in Section 2.2.3, as well as a set of outlier detection dataset features we propose below.

3.1 Features

Given that our outlier detection datasets have been generated by down-sampling classification datasets, we are able to borrow many of the features that have been used to summarize interesting properties of classification datasets, and then add some additional features that are unique to the outlier detection challenge. We start with a set of standard classification meta-features categorised as follows:

1. *Simple features* - These are related to the basic structure of a dataset. For our study these include the number of observations, number of attributes, ratio of observations to attributes, number of binary attributes, number of numerical attributes and the ratio of binary to numerical attributes.

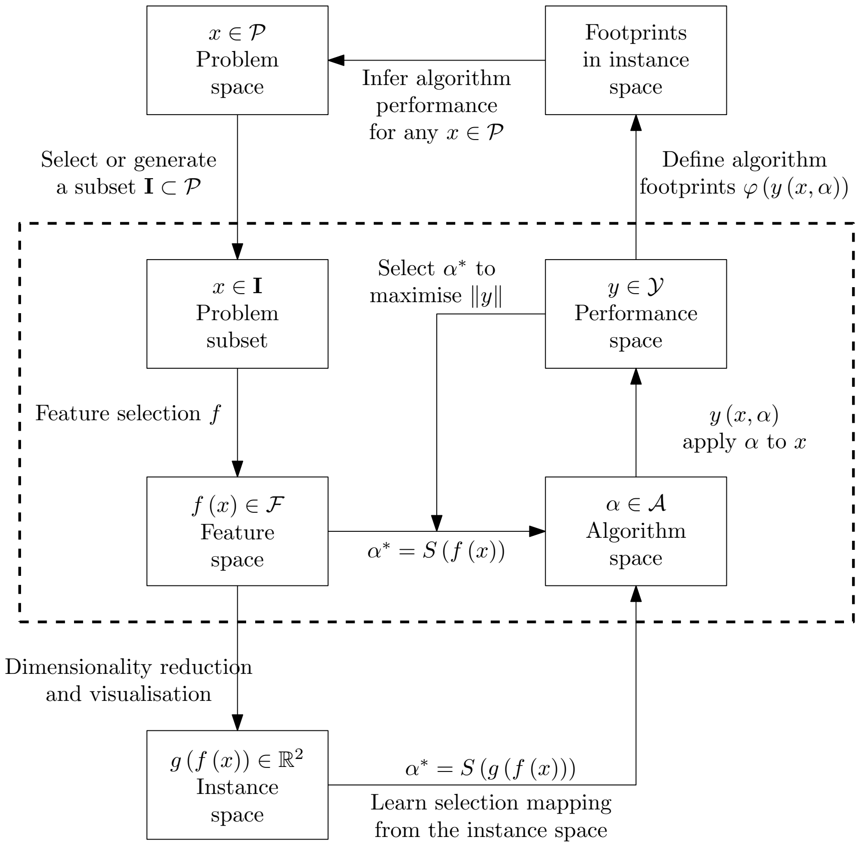

$x \in \mathcal{P}$
Problem space
Footprints in instance space
Infer algorithm performance for any $x \in \mathcal{P}$
Select or generate a subset $\mathbf{I} \subset \mathcal{P}$
Define algorithm footprints $\varphi(y(x,\alpha))$
$x \in \mathbf{I}$
Problem subset
Select α^* to maximise $\|y\|$
$y \in \mathcal{Y}$
Performance space
Feature selection f
$y(x,\alpha)$
apply α to x
$f(x) \in \mathcal{F}$
Feature space
$\alpha^* = S(f(x))$
$\alpha \in \mathcal{A}$
Algorithm space
Dimensionality reduction and visualisation
$g(f(x)) \in \mathbb{R}^2$
Instance space
$\alpha^* = S(g(f(x)))$
Learn selection mapping from the instance space

Figure 6: 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).

2. *Statistical features* - These include statistical properties of skewness, kurtosis, mean to standard deviation ratio, and IQR to standard deviation ratio for all attributes of a dataset, i.e. if a dataset has d attributes, then there are d values for each statistical property. For skewness and kurtosis we include the mean, median, maximum and the 95% percentile of these d values as features. For IQR to standard deviation ratio we include the maximum and the 95% percentile. We also include the average mean to standard deviation ratio. As a correlation measure, we compute the correlation between all attributes and include the mean absolute correlation. We also perform Principal Component Analysis and include the standard deviation explained by the first Principal Component.
3. *Information theoretic features* - for measures of the amount of information present in a dataset, we compute the entropy of each attribute and include the mean in our feature set. Also we include the entropy of the whole dataset and the mutual information.

The above set of features are generic features which measure various aspects of a dataset but are not particularly tailored towards outlier detection. While it is relevant for us to consider these features, they shed little light on the outlier structure of a dataset.

4. *Outlier features* - In order to make our feature set richer we include density-based, residual-based and graph-based features. We also include features that are based on section 2.1, which are related to the normalization vector $\boldsymbol{w}$. We compute these features for different subsets of the dataset, namely outliers, non-outliers and proxi-outliers. We define proxi-outliers as data points that are either far away from "normal" data or residing in low density regions. If there is a significant overlap between proxi-outliers

and actual outliers, then we expect outlier detection methods to perform well on such datasets. Formally, we define proxy-outliers as data-points which have the top 3% of knn-distances for k defined in (18). The density, residual and graph-based features we consider are all ratios. It is either a ratio between proxi-outliers and outliers, or a ratio between outliers and non-outliers. An example is the ratio between average density of non-outliers and average density of outliers. We explain the outlier features below:

i) Density based features
The density based features are computed either using the density based clustering algorithm DBSCAN (Ester et al. 1996) or kernel density estimation as follows:

   (a) DBSCAN features
   We perform principal component analysis (PCA) and use DBSCAN for clustering in a lower-dimensional space. We focus on data-points that either belong to very small clusters, or do not belong to any cluster. Let us call these points dbscan-proxi-outliers, henceforth named dbscan-proxies. Once again, if dbscan-proxies are outliers then we expect density based outlier algorithms to perform well on such datasets. As features we include i. the percentage of dbscan-proxies that are outliers, ii. the percentage of dbscan-proxies that are outliers which do not belong to any cluster and iii. the percentage of dbscan-proxies that are outliers which belong to very small clusters.

   (b) Kernel density estimate (KDE) related features
   Here too, we perform PCA and compute kernel density estimates (KDE) on two dimensional PC spaces to reduce computational burden. We compute KDE as detailed by Duong (2018) for the first 10 principal component (PC) pairs, and for each PC pair we find proxi-outliers. We compute the mean, median, standard deviation, IQR, minimum, maximum, 5^{th}, and 95^{th} percentiles of KDE values for outliers, non-outliers, proxi-outliers and non-proxi-outliers in each PC space. Next we take the computed summary statistics ratios of outliers to non-outliers, and proxi-outliers to non-proxi-outliers; for example the ratio between the mean KDE of non-outliers and the mean KDE of outliers. These ratios are computed for the first 10 two-dimensional PC spaces. As features, we include the average of each ratio for the set of PC spaces. In addition we also include the percentage of proxi-outliers that are outliers in our feature set.

   (c) Local density features
   We compute all features explained in i)(b) using a local density measure based on KDE instead of using KDE itself. The local density is computed by dividing the KDE value of a point by the mean KDE value of its k-nearest neighbours. Here for each data-point its k-nearest neighbours are computed with k as in equation (18).

ii) Residual based features
These features include summary statistics of residuals from linear models. First, we fit a series of linear models by randomly choosing the dependent variable and treating the rest of attributes as independent variables. For each model, data-points which have the the top 3% of absolute residual values are deemed as proxi-outliers. Similar to KDE features, the mean, median, standard deviation, IQR, minimum, maximum, 5^{th}, and 95^{th} percentiles of residual values for outliers, non-outliers, proxi-outliers and non-proxi-outliers are computed. Next the respective ratios are computed for each linear model. Finally, the average of each ratio for all the models is included in the feature set. We also include the percentage of proxi-outliers that are outliers as a feature.

iii) Graph based features
These features are based on graph-theoretic measures such as vertex degree, shortest path and connected components. First, we convert the dataset to a directed-graph based on each data-points' k-nearest neighbours using the software *igraph* (Csardi & Nepusz 2006). Next, we compute the degree of the vertices and label ones with the lowest degree as proxi-outliers. Then, similar to residual based features, we find the summary statistics of degree values for outliers, non-outliers, proxi-outliers and non-proxi-outliers and include ratios of outliers to non-outliers and proxi-outliers to non-proxi-outliers in our feature set. We also include the percentage of proxi-outliers which are actual outliers. Another set of graph features come from connected components. We compute

the number of vertices in each connected component and, similar to degree calculations, compute summary statistics of these values for outliers, non-outliers, proxi-outliers and non-proxi-outliers and include the ratios as above. We also compute the shortest distances from outlier vertices to non-outlier vertices. Here the shortest distance from vertex a to vertex b is the minimum number of edges that connect a and b. We include some statistics about these distances: the percentage of outliers which have infinite shortest distance to all non-outlier vertices, i.e. outlier vertices which are not connected. For outliers that have finite shortest distances to some non-outlying vertex, we compute the percentage of outliers for which the shortest distance is 1.

iv) Normalization related features
These features are related to quantities described in section 2.1 and Proposition 2.1. First, we compute the normalization vector $\boldsymbol{w}$ as in equation (4) for Min-Max and Median-IQR. Next, we compute vectors $\boldsymbol{y}_{ij}$ as in equation (4) for outliers and non-outliers based on each data-point's k-nearest neighbours. Then we compute $\langle \boldsymbol{w}, \boldsymbol{y}_{ij} \rangle$ for the two different normalization vectors $\boldsymbol{w}$ that correspond to Min-Max and Median-IQR. The purpose of this exercise is to compare the quantity $\langle \boldsymbol{w}, \boldsymbol{y}_{ij} \rangle$ obtained from outliers with that of non-outliers for each normalization technique. So we compute $\langle \boldsymbol{w}, \boldsymbol{y}_{ij} \rangle$ ratios of outliers to non-outliers and include the minimum, maximum, mean, median, standard deviation and IQR as features. We also include percentage of ratio values less than 1, as this is a quantity of interest and relates to equation (15) in Proposition 2.1.

This concludes the list of features - both classification and outlier based - that we compute for each dataset. From this list of features, those that are based on density, residuals and graphs depend on nearest neighbours and as such are sensitive to the method of normalization. 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 5 provides a summary of features by category.

Table 5: 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

Before we can proceed with the Instance Space Analysis, it is important to validate that the features contain sufficient information about the similarities, differences and difficulties of datasets that they are reliable as instance summaries. To this end, we first demonstrate that reasonable accuracy can be obtained using the features to predict sensitivity to normalization, and outlier detection method performance, given the characteristics of a dataset summarized by the features.

3.2 Predicting sensitivity to normalization

We have already demonstrated in Section 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 section 3.1. Using 10-fold cross validation (Bischl et al. 2012), we train and test

12 random forest classifiers (Liaw & 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 6. As shown in Table 6 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 6. 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.

Table 6: Prediction results for ξ-sensitivity to normalization with $\xi = 0.05$ using 10-fold cross validation

Outlier detection method	Actual Percentage sensitive to normalization(%)	Prediction accuracy of sensitivity to normalization (%)
COF	67.36	76.78
FAST ABOD	77.45	80.15
INFLO	62.76	74.78
KDEOS	68.48	76.87
KNN	50.63	73.17
KNNW	45.88	72.74
LDF	63.89	71.63
LDOF	57.70	74.42
LOF	59.84	74.02
LOOP	55.46	73.19
ODIN	57.28	73.06
SIMLOF	53.77	73.61

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 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 5-fold cross-validation, we predict the normalization method that gives better performance with results shown in Table 7. From Table 7 we see 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 section 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 7 may be because the set of features do not sufficiently explain these effects for all outlier methods equally. Indeed, the features were pooled together 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 5. When comparing with Table 6 which predicts the sensitivity to normalization, Table 7 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 Section 3.4.

Table 7: Best normalization method prediction accuracy

Outlier detection method	$\xi = 0.05$ (%)	$\xi = 0.10$ (%)	$\xi = 0.15$ (%)
COF	57.69	61.72	68.52
FAST ABOD	76.40	83.62	82.84
INFLO	58.94	63.49	59.32
KDEOS	61.76	63.24	69.85
KNN	71.73	74.18	93.51
KNNW	74.25	85.85	93.33
LDF	60.42	62.68	63.87
LDOF	66.36	62.71	73.54
LOF	63.54	65.17	70.89
LOOP	64.45	67.22	69.43
ODIN	59.78	57.23	67.47
SIMLOF	63.66	64.74	73.47

Table 8: Outlier method performance prediction - average cross validation accuracy %

Outlier detection method	Default accuracy(%) of AUC > 0.8	Prediction accuracy of AUC > 0.8
COF	75.58	83.48
FAST ABOD	67.77	86.07
INFLO	83.22	89.29
KDEOS	90.96	92.81
KNN	68.16	86.56
KNNW	67.13	86.13
LDF	75.65	85.28
LDOF	80.08	87.36
LOF	74.63	84.07
LOOP	77.19	85.88
ODIN	79.09	87.00
SIMLOF	75.85	85.21

3.3 Predicting outlier method performance using features

To confirm that our set of 346 features are also predictive of outlier method performance, we use the complete set of features as input to a Random Forest classifier, which predicts whether a method's area under the ROC curve is greater than 0.8, indicating a good performance for an outlier detection method. Table 8 presents the average cross-validation accuracy over 10 folds for each outlier detection method. The default accuracy is the percentage of the majority class. As the minimum accuracy is 83.48%, we conclude that these features are reasonable predictors of outlier detection method performance.

3.4 Constructing an outlier detection instance space

Having validated that the features contain sufficient information to be predictive of both normalization effects and outlier detection method performance, we now use the features to construct an instance space to visualize the relationships between instance features and strengths and weaknesses of methods.

3.4.1 Feature subset selection

Critical to both algorithm performance prediction and instance space construction is the selection of features that are both distinctive (i.e. uncorrelated to other features) and predictive (i.e correlated with algorithm performance). Therefore, we follow a systematic procedure to select a small subset of features that best meet these requirements, using a subset of 2018 instances for which there are three well performing algorithms or less. 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 of datasets sufficient for performance prediction and instance space construction.

We pre-process the data such that it becomes amenable to machine learning and dimensionality projection method. Given that some features are ratios, there is often the case that one feature can produce excessively large or infinite values. Hence, we bound all the features between their median plus or minus five times their interquartile range. Any not-a-number value is converted to zero, while all infinite values are equal to the bounds. Next, we apply Box-Cox transformation to each feature to stabilize the variance and make the data more normal-like. Finally, we apply z-transform to standardize the feature.

We start with the full set of 346 features. First, we determine whether the feature has unique values for most instances. A feature provides little information if it produces the same value for the majority of the datasets. Hence, we discard those that have less than 30% unique values. After this step we are left with 255 features. Then, we check the correlation between the features and the algorithm performance measure, which is the area under the ROC curve. Sorting the absolute value of the correlation from highest to lowest, we pick the top three features per algorithm, some of which can be repeated. After this step we are left with 26 features. Next, we identify groups of similar features. We use a clustering approach, with k-means as the clustering algorithm and $1 - |\rho|$, where ρ is the correlation between two features, as the dissimilarity measure. As result, we obtain eight clusters. All the possible combinations of eight features out of 26, taking only one feature from each cluster, is 7200. Finally, we determine the best of these 7200 subsets for the dimensionality reduction. We use PCA with two components to project the datasets into a 2-d instance space using a candidate feature subset. Then, we fit a random forest model per algorithm to classify the instances in the trial instance space into groups of ϵ-good and bad performance, which is defined as follows:

Definition 3.1. *For a given dataset x, an outlier detection algorithm α gives ϵ-good performance if the difference in area under the ROC curve to the best algorithm is less than ϵ. Formally:*

$$\max_{a \in \mathcal{A}} (AUC(x, a)) - AUC(x, \alpha) < \epsilon \,,$$

where $\mathcal{A}$ denotes the algorithm space and AUC the area under the ROC curve.

That is, we fit 12 models per each feature combination. We define the best combination as the one producing the lowest average out-of-the-bag (OOB) error across all models. Table 9 shows the combinations which produce the lowest OOB error for each algorithm, and the one selected as the best compromise. We observe that: (a) the most selected feature in a cluster does not always belong to the lowest average set; (b) some features never belong to a best performing subset; and (c) each algorithm has a unique subset of preferred features, even if it shares underlying principles with other algorithms.

The final set of selected features from which we construct the instance space is listed in table 10.

3.4.2 Projection method

Our approach to constructing the instance space is based on the most recent implementation of the methodology described in (Muñoz et al. 2018). We use the Prediction Based Linear Dimensionality Reduction (PBLDR) method (Muñoz et al. 2018) to find a projection from 8-d to 2-d that creates the most linear trends of algorithm performance and feature values across the instance space, to assist visualization of directions of hardness and feature correlations. The resulting method requires global optimization techniques to solve the multi-objective optimization problem of finding the optimal linear transformation matrix, and the optimization algorithm BIPOP-CMA-ES (Hansen 2009) is used. Given this method's stochasticity, we calculate 30 different projections. We then select the one with the highest topological preservation, defined as the correlation between high- and low-dimensional distances. The

Table 9: Feature combinations that produce the lowest out-of-the-bag error (OOB) for each algorithm, and the one that produces the lowest average across all models (LO-AVG).

cluster #	Feature	COF	FAST ABOD	INFLO	KDEOS	KNN	KNNW	LDF	LDOF	LOF	LOOP	ODIN	SIMLOF	% selected	LO-AVG
1	OPO_DenOut_Out_95P_1							✓	✓					17	
	OPO_LocDenOut_Out_95P_1	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	83	✓
2	OPO_Out_DenOut_1_3	✓	✓			✓		✓			✓	✓		50	✓
	OPO_Out_LocDenOut_1_3			✓			✓			✓			✓	33	
	OPO_Out_LocDenOut_2_3				✓				✓					17	
3	Skew_95					✓				✓			✓	25	
	Kuto_Max													0	
	OPO_Res_KNOut_95P_1	✓	✓	✓				✓	✓					42	✓
	OPO_Res_ResOut_Median_3				✓		✓				✓	✓		33	
4	OPO_Res_Out_SD_1							✓						8	
	OPO_Res_Out_IQR_1													0	
	OPO_Res_Out_95P_1	✓				✓				✓	✓	✓		42	
	OPO_Res_Out_IQR_3				✓		✓		✓					25	
	OPO_Res_Out_95P_3		✓	✓									✓	25	✓
5	Total_Entropy_Dataset	✓		✓	✓				✓	✓	✓	✓	✓	67	
	OPO_ResOut_Out_Min_1													0	
	OPO_ResOut_Out_Min_3													0	
	OPO_GComp_PO_Mean_3		✓					✓						17	
	OPO_GComp_PO_Q95_3					✓	✓							17	✓
6	OPO_Den_Out_SD_3		✓				✓		✓				✓	33	✓
	OPO_Den_Out_IQR_3	✓			✓	✓				✓		✓		42	
	OPO_Den_Out_95P_3			✓				✓			✓			25	
7	SNR	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓		83	✓
	OPO_LocDen_Out_IQR_1									✓			✓	17	
8	OPO_GDeg_Out_Mean_1	✓					✓	✓		✓	✓	✓	✓	58	✓
	OPO_GDeg_Out_Mean_3		✓	✓	✓	✓			✓					42	
	OOB error (%)	19	25	2	12	24	24	20	10	11	8	8	10		16

final projection matrix is defined by Equation 23 to represent each dataset as a 2-d vector **Z** depending on its 8-d feature vector.

$$\mathbf{Z} = \begin{bmatrix} -0.0506 & 0.1731 \\ -0.0865 & -0.1091 \\ -0.1697 & 0.0159 \\ 0.0041 & -0.0465 \\ -0.2158 & 0.0398 \\ 0.0087 & -0.0053 \\ 0.0420 & -0.2056 \\ -0.0661 & -0.1864 \end{bmatrix}^{\top} \begin{bmatrix} \text{SNR} \\ \text{OPO\_Res\_KNOut\_95P\_1} \\ \text{OPO\_Out\_DenOut\_1\_3} \\ \text{OPO\_Den\_Out\_SD\_3} \\ \text{OPO\_Res\_Out\_95P\_3} \\ \text{OPO\_LocDenOut\_Out\_95P\_1} \\ \text{OPO\_GDeg\_Out\_Mean\_1} \\ \text{OPO\_GComp\_PO\_Q95\_3} \end{bmatrix} \tag{23}$$

Figure 7 illustrates the resulting instance space, including the full set of more than 12000 instances, discriminated by their source. The sets by Campos et al. (2016) and Goldstein & Uchida (2016) are mostly located in the lower left area of the space; whereas the set produced by down-sampling the

Table 10: Feature descriptions

#	Feature	Description	Other details
1	OPO_LocDenOut_Out_95P_1	$\frac{95^{\text{th}}\text{ percentile of local density for proxi-outliers}}{95^{\text{th}}\text{percentile of local density for outliers}}$	Local density computed using KDE on 2D PC space. Min-Max used.
2	OPO_Out_DenOut_1_3	$\frac{\text{number of density proxi-outliers that are also outliers}}{\text{number of outliers}}$	Density computed using KDE on 2D PC space. Median-IQR used.
3	OPO_Res_KNOut_95P_1	$\frac{95^{\text{th}}\text{ percentile of residuals for non-proxi-outliers}}{95^{\text{th}}\text{percentile of residuals for proxi outliers}}$	Residuals of many linear models computed with Min-Max normalization.
4	OPO_Res_Out_95P_3	$\frac{95^{\text{th}}\text{ percentile of residuals for non-outliers}}{95^{\text{th}}\text{percentile of residuals for outliers}}$	Residuals of many linear models computed with Median-IQR normalization.
5	OPO_GComp_PO_Q95_3	$\frac{95^{\text{th}}\text{percentile of connected component size of graphs for non-proxi-outliers}}{95^{\text{th}}\text{percentile of connected component size of graphs for proxi-outliers}}$	The distribution of connected components of KNN graphs computed using Median-IQR normalizaiton.
6	OPO_Den_Out_SD_3	$\frac{\text{Standard deviation of density for non-outliers}}{\text{Standard deviation of density for outliers}}$	Density computed using KDE on 2D PC space. Median-IQR used.
7	SNR	Signal to noise ratio	Averaged across dataset attributes.
8	OPO_GDeg_Out_Mean_1	$\frac{\text{Mean graph-degree for non-outliers}}{\text{Mean graph-degree for outliers}}$	The distribution of vertex degree of KNN graphs computed using Min-Max normalizaiton.

UCI repository provides a greater coverage of the instance space and hence more diversity of features. Finally, Figures 8 and 9 show the distribution of feature values and outlier method performance across the instance space respectively, based only on the subset of 2018 instances. The scale has been adjusted to the $[0, 1]$ range. We observe that:

1. Low values of the feature SNR and high values of OPO_Res_KNOut_95P_1 are found at the bottom of the space, which correlates with good performance of LDF.
2. Both OPO_Out_DenOut_1_3 and OPO_Res_Out_95P_3 tend to decrease from left to right of the space. Both features tend to correlate with high performance of KDEOS and low performance of KNN and KNNW.
3. Performance of FAST ABOD tends to increase from the bottom up, which tends to be correlated with the feature OPO_GDeg_Out_Mean_1.
4. There are no distinguishable linear patterns for some algorithms, such as COF and LOF. This indicates that either PBLDR cannot find a predictive projection for these algorithms, or that we lack representative instances for these algorithms. This will be reflected in weaker footprint results for these methods.

3.5 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 & Tan 2012). To construct a footprint, we follow the approach first introduced in (Smith-Miles & Tan 2012) and later refined in (Muñoz & Smith-Miles 2017): (a) we measure the distance between all instances, and eliminate those with a distance lower than a threshold, δ; (b) we calculate a Delaunay triangulation within the convex hull created formed by the remaining instances; (c) we create a concave hull, by removing any triangle with edges larger than another threshold, Δ; (d) we calculate the density and purity of each triangle in the concave hull; and, (e) we remove any triangle that does not fulfil the density and purity thresholds. The values for parameters for the lower and upper distance thresholds, $\{\delta, \Delta\}$, are set to 1% and 25% of the

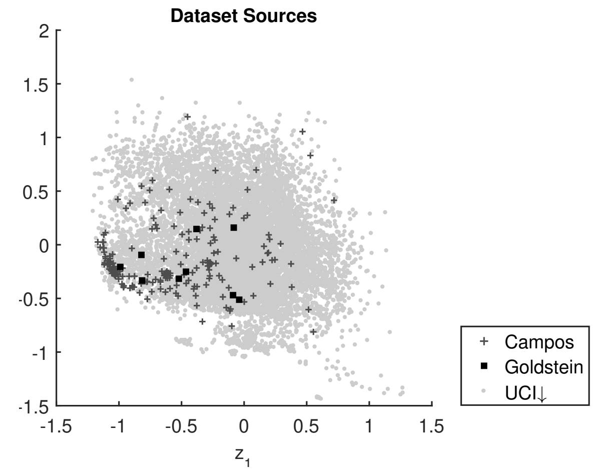


Figure 7: Instance space including the full set of more than 12000 instances, discriminated by their source.

maximum distance respectively. 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 area lost by the overlapping footprints when the contradicting sections are removed. The algorithm that would loose the largest area in a contradiction gets to keep it, as long as it maintains the density and purity thresholds.

Table 11 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, assuming that the most suitable normalization method was selected. The results are expressed as a percentage of the total area (3.5520) and density (565.8808) of the convex hull that encloses all instances. Further results are also illustrated in Figure 10, which shows the ϵ-good footprint as black areas and ϵ-bad instances as grey marks. Table 11 demonstrates that most algorithms have very small footprints. This can be corroborated by Figure 10, which shows that some algorithms do not have pockets of good performance. Instead, some algorithms such as INFLO, LDOF and SIMLOF present good performance in scattered regions of the instance space; hence, we fail to find a well-defined area that fulfils the density and purity requirements. On the other hand, FAST ABOD, KNN and KNNW possess the largest footprints. FAST ABOD, with a footprint covering 29.8%, of the instance space tends to dominate the upper left areas, while KNN and KNNW tend to dominate the lower left areas of the instance space. KDEOS and LDF are special cases. If we only consider their ϵ-good performance, we could think that both are unremarkable, as their footprints only cover 4.9% and 2.2% of the space respectively. However, their footprint increase to 12.9% and 6.4% respectively when considering their best performance, suggesting that they have some unique strengths. Observing Figures 10d and 10g, we observe that KDEOS and LDF tend to dominate the upper right and lower areas of the instance space respectively. Given that the footprint calculation minimises contradictions, their ϵ-good performance is diminished when it is compared with the three dominating algorithms, FAST ABOD, KNN and KNNW. In fact, most algorithms have areas of superior performance, which are masked by good performance of the three dominating ones.

3.6 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.

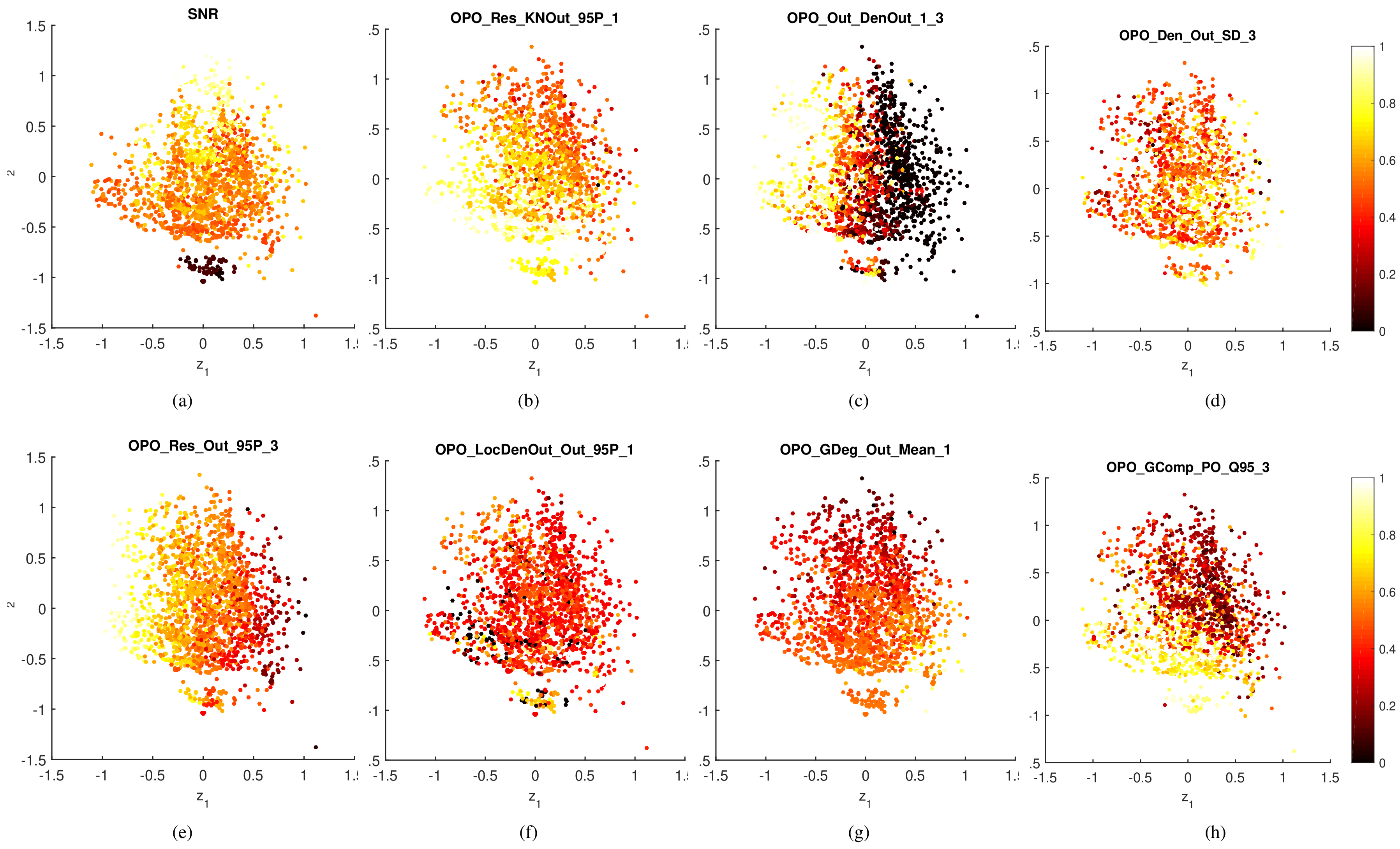


Figure 8: Distribution of normalized features on the projected instance space.

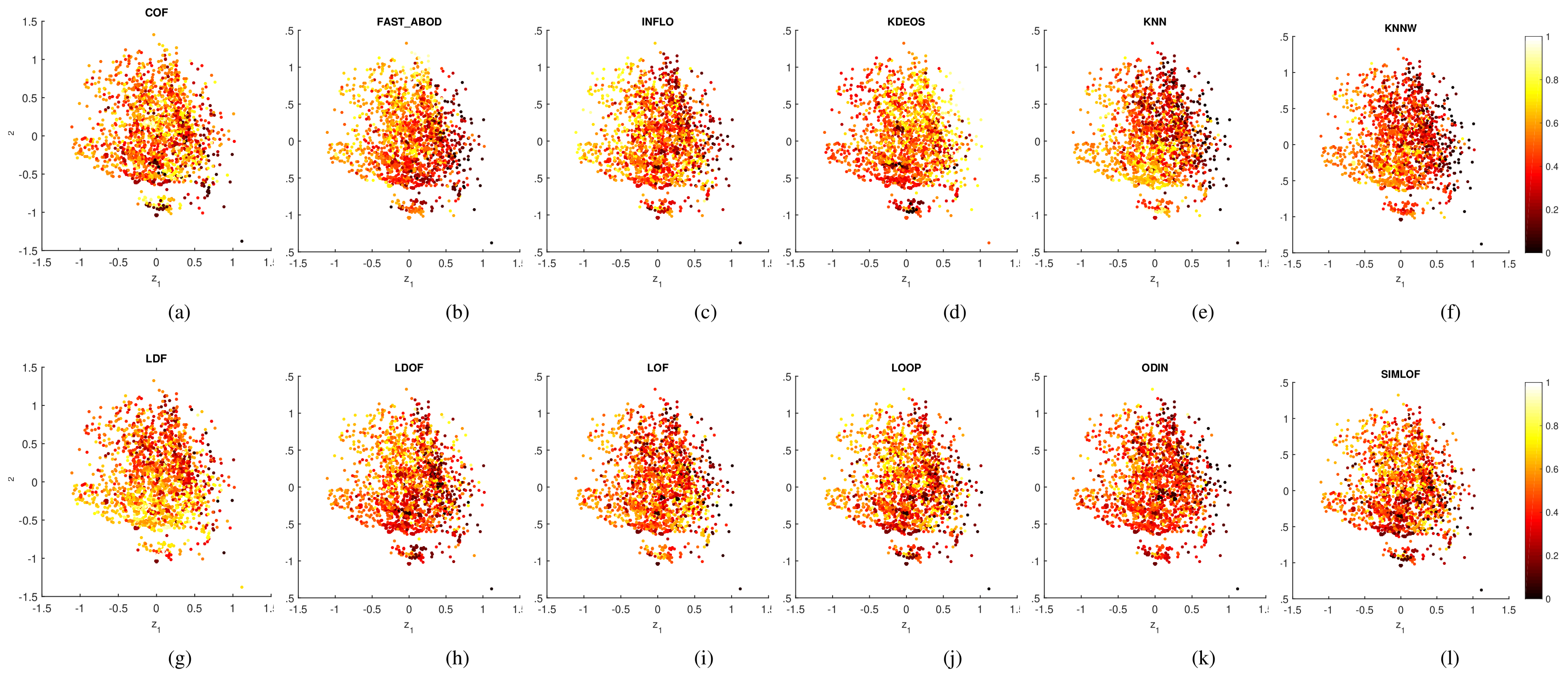


Figure 9: Scaled area under the curve for each outlier detection algorithm on the instance space.

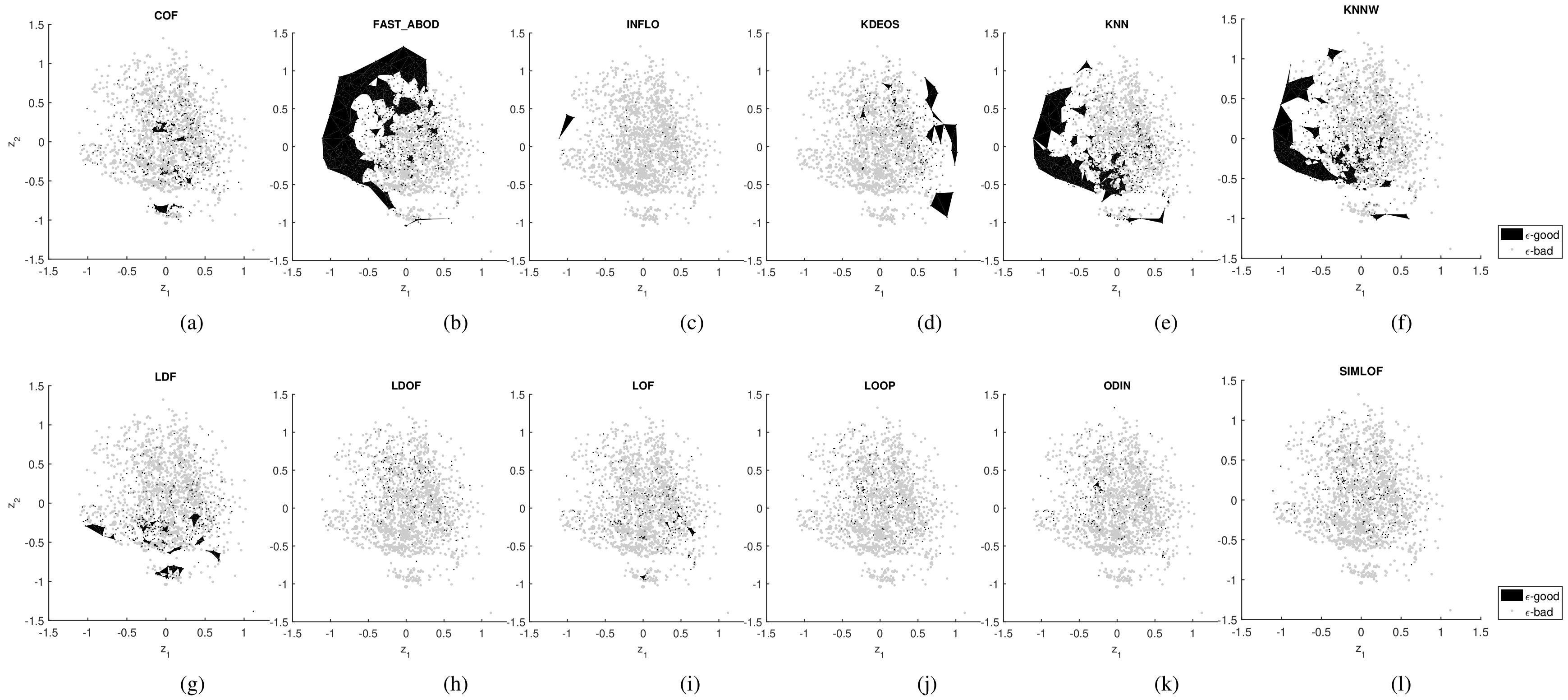


Figure 10: Footprints of the algorithms in the instance space, assuming ϵ-good performance.

Table 11: Footprint analysis of the algorithms. 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 ϵ-good and best, with $\epsilon = 0.05$.

	ϵ-good			Best algorithm		
	α_N	d_N	p	α_N	d_N	p
COF	0.9%	464.2%	94.3%	1.7%	317.1%	86.0%
FAST ABOD	29.8%	81.2%	95.1%	5.8%	181.7%	82.5%
INFLO	0.5%	31.2%	100.0%	6.4%	34.1%	88.6%
KDEOS	4.9%	50.4%	88.0%	12.9%	46.8%	78.5%
KNN	15.7%	130.5%	96.4%	1.3%	237.5%	80.3%
KNNW	12.3%	128.6%	96.6%	2.5%	171.4%	83.9%
LDF	2.2%	380.4%	95.2%	6.4%	200.6%	80.3%
LDOF	0.0%	4180.1%	100.0%	2.0%	123.7%	82.0%
LOF	0.2%	386.5%	100.0%	1.7%	155.9%	79.6%
LOOP	0.0%	8164.9%	100.0%	0.9%	96.8%	83.3%
ODIN	0.1%	720.4%	90.9%	3.2%	80.4%	76.9%
SIMLOF	0.0%	0.0%	00.0%	0.9%	196.5%	79.4%

Table 12: Accuracy of SVM prediction of ϵ-good performance based on instance space location of test sets.

Outlier detection method	Prediction Accuracy (%)	Actual percentage of majority class (%)
FAST ABOD	71	58
KDEOS	87	87
KNN	71	60
KNNW	68	64

We use support vector machines (SVM) for this partitioning. Of the 12 outlier methods we consider FAST ABOD, KDEOS, KNN and KNNW, as these methods have bigger footprints that span a contiguous region of the instance space. For these outlier methods, we use ϵ-good performance as the output and the instance space coordinates as the input for the SVM. In this way, we train 4 SVMs, each SVM on a single outlier method. The prediction accuracies using 5-fold cross validation along with the percentage of instances in the majority class are given in Table 12. From Table 12 we see that for FAST ABOD, KNN and KNNW that the SVM accuracy is greater than the majority class percentage and for KDEOS, it is equal.

The regions of strength resulting from this experiment are given in Figure 11. From Figure 11 we see an overlap of regions for FAST ABOD, KNN and KNNW. By combining these regions of strength we obtain a partitioning of the instance space shown in Figure 12. To break ties, we use the prediction probability of the SVM and choose the method with the highest prediction probability. One can also use a different approach such as the sensitivity to normalization criteria to break ties.

From Figure 12 we see that no outlier method is recommended for a large part of the instance space. This highlights the opportunity for new outlier methods which 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 Figure 5) has a niche in the instance space where no outlier method performs well. This insight was missed by the standard statistical analysis.

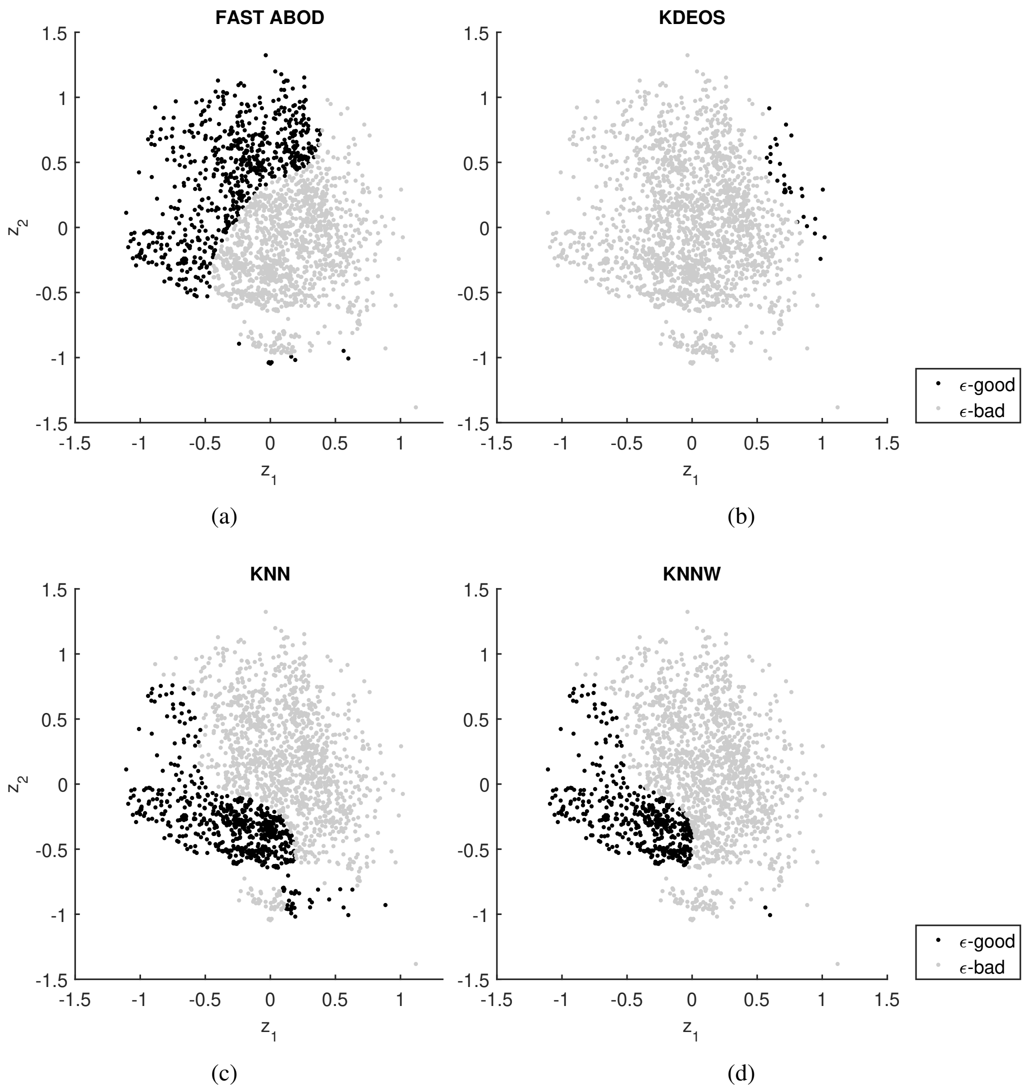


Figure 11: Regions of strength for FAST ABOD, KNN family and LOF family.

4 Conclusions

In this study we have investigated the effect of normalization and the algorithm selection problem for 12 unsupervised outlier methods. Normalization is a topic which 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.

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.

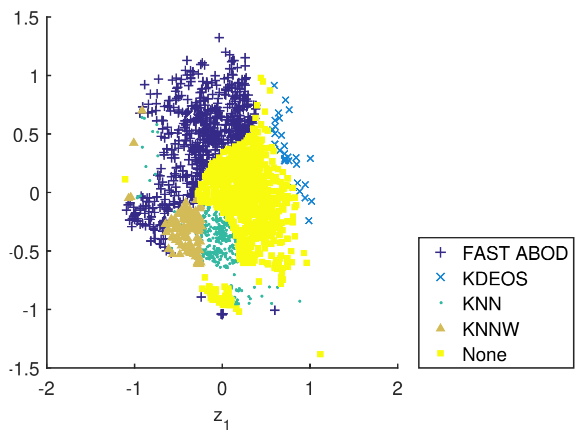


Figure 12: A partition of the instance space showing recommended outlier detection methods.

In addition to normalization we also studied the algorithm selection problem for unsupervised outlier detection. Given measurable features of a dataset, we find the outlier method best suited for it with reasonable accuracy. This is important because each method has its strengths and weaknesses and no single method out-performs all others for all instances. We have explored the strengths and weaknesses of outlier methods by analysing their footprints in the constructed instance space. Moreover, we have identified different regions of the instance space that reveal the relative strengths of different outlier detection methods. Our work clearly demonstrates for example that KDEOS, which gives poor performance on average, has a region of strength in the instance space where no other algorithm excels.

In addition to these contributions, we hope to have laid some important foundations for future research into new and improved outlier detection methods, in the following ways: 1. enabling evaluation of the sensitivity to normalization for new outlier methods; 2. rigorous evaluation of new methods using the comprehensive corpus of over 12000 datasets with diverse characteristics we have made available; 3. using the instance space, the strengths and weaknesses of new outlier methods can be identified, and their uniqueness compared to existing methods described. Equally valuable, the instance space analysis can also reveal if a new outlier method is similar to existing outlier methods, or offers a unique contribution to the available repertoire of techniques.

As a word of caution, we note that our current instance space is computed using our set of datasets, outlier methods and features. Thus, we do not make claim to have constructed the definitive instance space for all unsupervised outlier detection methods. Hence, the selected features for the instance space may change with the expansion of the corpus of datasets and outlier methods. Future research paths include the expansion of the instance space by generating new and diverse instances and considering other classes of outlier detection methods, such as subspace approaches. To aid this expansion and future research, we make all of our data and implementation scripts available on our website.

Broadening the scope of this work, we have been adapting the instance space methodology to other problems besides outlier detection. For example, machine learning (Muñoz et al. 2018) and time series forecasting (Kang et al. 2017). Part of this larger project is to build freely accessible web-tools that carry out the instance space analysis automatically, including testing of algorithms on new instances. Such tools will be available at `matilda.unimelb.edu.au` in the near future.

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.

Billor, N., Hadi, A. S. & Velleman, P. F. (2000), 'Bacon: blocked adaptive computationally efficient outlier nominators', *Computational Statistics & Data Analysis* **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*, Cognitive Technologies, Springer.

Breheny, P. & Burchett, W. (2012), 'Visualizing regression models using visreg'.

Breunig, M. M., Kriegel, H.-P., Ng, R. T. & Sander, J. (2000), Lof: identifying density-based local outliers, *in* 'ACM sigmod record', Vol. 29, ACM, pp. 93–104.

Campos, G. O., Zimek, A., Sander, J., Campello, R. J., Micenková, B., Schubert, E., Assent, I. & Houle, M. E. (2016), 'On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study', *Data Mining and Knowledge Discovery* **30**(4), 891–927.

Craswell, N. (2009), Precision at n, *in* 'Encyclopedia of database systems', Springer, pp. 2127–2128.

Csardi, G. & Nepusz, T. (2006), 'The igraph software package for complex network research', *InterJournal, Complex Systems* **1695**(5), 1–9.

Culberson, J. C. (1998), 'On the futility of blind search: An algorithmic view of no free lunch', *Evolutionary Computation* **6**(2), 109–127.

Duong, T. (2018), 'ks: Kernel density estimation for bivariate data'.

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* .

Ester, M., Kriegel, H.-P., Sander, J., Xu, X. et al. (1996), A density-based algorithm for discovering clusters in large spatial databases with noise., *in* 'Kdd', Vol. 96, pp. 226–231.

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.

Hansen, N. (2009), Benchmarking a bi-population CMA-ES on the BBOB-2009 function testbed, *in* 'GECCO '09', ACM, pp. 2389–2396.

Hautamaki, V., Karkkainen, I. & Franti, P. (2004), Outlier detection using k-nearest neighbour graph, *in* 'Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on', Vol. 3, IEEE, pp. 430–433.

Hawkins, D. M. (1980), *Identification of outliers*, Vol. 11, Springer.

Ho, Y.-C. & Pepyne, D. L. (2002), 'Simple explanation of the no-free-lunch theorem and its implications', *Journal of optimization theory and applications* **115**(3), 549–570.

Hubert, M. & Van der Veeken, S. (2008), 'Outlier detection for skewed data', *Journal of chemometrics* **22**(3-4), 235–246.

Igel, C. & Toussaint, M. (2005), 'A no-free-lunch theorem for non-uniform distributions of target functions', *Journal of Mathematical Modelling and Algorithms* **3**(4), 313–322.

Jin, W., Tung, A. K., 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.

Kang, Y., Hyndman, R. & Smith-Miles, K. (2017), 'Visualising forecasting algorithm performance using time series instance spaces', *Int. J. Forecast* **33**(2), 345–358.

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., Zimek, A. et al. (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, L. J., 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.

Leyton-Brown, K., Nudelman, E., Andrew, G., McFadden, J. & Shoham, Y. (2003), A portfolio approach to algorithm selection, *in* 'IJCAI', Vol. 3, pp. 1542–1543.

Liaw, A. & Wiener, M. (2002), 'Classification and regression by randomforest', *R News* **2**(3), 18–22.
URL: *http://CRAN.R-project.org/doc/Rnews/*

Muñoz, M. A., Villanova, L., Baatar, D. & Smith-Miles, K. (2018), 'Instance spaces for machine learning classification', *Machine Learning* **107**(1), 109–147.

Muñoz, M. & Smith-Miles, K. (2017), 'Performance analysis of continuous black-box optimization algorithms via footprints in instance space', *Evol. Comput.* **25**(4), 529–554.

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.

Rousseeuw, P. J. & Hubert, M. (2017), 'Anomaly detection by robust statistics', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* .

Schubert, E., Zimek, A. & Kriegel, H.-P. (2014*a*), 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. (2014*b*), 'Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection', *Data Mining and Knowledge Discovery* **28**(1), 190–237.

Smith-Miles, K. A. (2009), 'Cross-disciplinary perspectives on meta-learning for algorithm selection', *ACM Computing Surveys (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', *Computers & Operations Research* **63**, 102–113.

Smith-Miles, K. & Tan, T. T. (2012), Measuring algorithm footprints in instance space, *in* 'Evolutionary Computation (CEC), 2012 IEEE Congress on', IEEE, pp. 1–8.

Talagala, P., Hyndman, R., Smith-Miles, K., Kandanaarachchi, S., Munoz, M. et al. (2018), Anomaly detection in streaming nonstationary temporal data, Technical report, Monash University, Department of Econometrics and Business Statistics.

Tang, J., Chen, Z., Fu, A. W.-C. & Cheung, D. W. (2002), Enhancing effectiveness of outlier detections for low density patterns, *in* 'Pacific-Asia Conference on Knowledge Discovery and Data Mining', Springer, pp. 535–548.

Wilkinson, L. (2018), 'Visualizing big data outliers through distributed aggregation', *IEEE transactions on visualization and computer graphics* **24**(1), 256–266.

Wolpert, D. H. & Macready, W. G. (1997), 'No free lunch theorems for optimization', *IEEE transactions on evolutionary computation* **1**(1), 67–82.

Wolpert, D. H., Macready, W. G. et al. (1995), No free lunch theorems for search, Technical report, Technical Report SFI-TR-95-02-010, Santa Fe Institute.

Zhang, E. & Zhang, Y. (2009), Average precision, *in* 'Encyclopedia of database systems', Springer, 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', *Statistical Analysis and Data Mining: The ASA Data Science Journal* **5**(5), 363–387.