Journal of Machine Learning Research 25 (2024) 1-34 Submitted 5/23; Revised 1/24; Published 3/24

Unsupervised Anomaly Detection Algorithms on Real-world Data: How Many Do We Need?

Roel Bouman ROEL.BOUMAN@RU.NL
Institute for Computing and Information Sciences
Radboud University
Toernooiveld 212, 6525 EC Nijmegen, The Netherlands

Zaharah Bukhsh Z.BUKHSH@TUE.NL
Information Systems, Industrial Engineering and Innovation Sciences
Eindhoven University of Technology
Groene Loper 3, 5612 AE Eindhoven, The Netherlands

Tom Heskes TOM.HESKES@RU.NL
Institute for Computing and Information Sciences
Radboud University
Toernooiveld 212, 6525 EC Nijmegen, The Netherlands

Editor: Marc Schoenauer

Abstract

In this study we evaluate 33 unsupervised anomaly detection algorithms on 52 real-world multivariate tabular data sets, performing the largest comparison of unsupervised anomaly detection algorithms to date. On this collection of data sets, the EIF (Extended Isolation Forest) algorithm significantly outperforms the most other algorithms. Visualizing and then clustering the relative performance of the considered algorithms on all data sets, we identify two clear clusters: one with "local" data sets, and another with "global" data sets. "Local" anomalies occupy a region with low density when compared to nearby samples, while "global" occupy an overall low density region in the feature space. On the local data sets the kNN (k-nearest neighbor) algorithm comes out on top. On the global data sets, the EIF (extended isolation forest) algorithm performs the best. Also taking into consideration the algorithms' computational complexity, a toolbox with these two unsupervised anomaly detection algorithms suffices for finding anomalies in this representative collection of multivariate data sets. By providing access to code and data sets, our study can be easily reproduced and extended with more algorithms and/or data sets.

Keywords: Unsupervised Anomaly Detection, Anomaly Analysis, Algorithm Comparison, Outlier Detection, Outlier Analysis

1. Introduction

Anomaly detection is the study of finding data points that do not fit the expected structure of the data. Anomalies can be caused by unexpected processes generating the data. In chemistry an anomaly might be caused by an incorrectly performed experiment, in medicine a certain disease might induce rare symptoms, and in predictive maintenance an anomaly can be indicative of early system failure. Depending on the application domain, anomalies have different properties, and may also be called by different names. Within the domain

of machine learning (and hence also in this paper), anomaly detection is often used interchangeably with outlier detection.

Unsupervised, data-driven, detection of anomalies is a standard technique in machine learning. Throughout the years, many methods, or algorithms, have been developed in order to detect anomalies. Some of these algorithms aim to solve specific issues, such as high dimensionality. Other methods try to detect anomalies in the general sense, and focus on high performance or low computational or memory complexity. Due to the many algorithms available, it is hard to determine which algorithm is best suited for a particular use case, especially for a user who is not intimately familiar with the field of anomaly detection. More details on the specific algorithms evaluated in this study can be found in section 3.1.

Several studies have been performed to provide guidelines on when to apply which algorithm. Some review studies (Malik et al., 2014; Ruff et al., 2021), give advice based on the theoretical properties of the algorithms. In recent years, several studies have been conducted that empirically compare a number of anomaly detection algorithms on a range of data sets.

Emmott et al. (2015) study 8 well-known algorithms on 19 data sets. They find Isolation Forest to perform the best overall, but recommend using ABOD (Angle-Based anomaly Detection) or LOF (Local Outlier Factor) when there are multiple clusters present in the data.

Campos et al. (2016) compare 12 k-nearest neighbours based algorithms, on 11 base data sets. They find LOF to significantly outperform a number of other methods, while KDEOS (Kernel Density Estimation anomaly Score) performs significantly worse than most algorithms.

Goldstein and Uchida (2016) compare 19 algorithms on 10 data sets. Unlike Campos et al., Goldstein and Uchida perform no explicit optimization or selection, but rather evaluate the average performance over a range of sensible hyperparameter settings. With methods based on k-nearest neighbours generally giving stable results, Goldstein and Uchida recommend kNN (k-nearest neighbours) for global anomalies, LOF for local anomalies, and HBOS (Histogram-Based anomaly Selection) in general (see 2.2 for an explanation of global and local anomalies). Goldstein and Uchida compare on a data set basis, without any overall statistical analysis.

More recently, Domingues et al. (2018), apply 14 algorithms on 15 data sets, some of which are categorical. They find IF (Isolation Forest) and robust KDE (Kernel Density Estimation) to perform best, but note that robust KDE is often too expensive too calculate for larger data sets.

Steinbuss and Böhm (2021) propose a novel strategy for synthesizing anomalies in real-world data sets using several statistical distributions as a sampling basis. They compare 4 algorithms across multiple data sets derived from 19 base data sets, both using the original and synthesized anomalies. They find kNN and IF to work best for detecting global anomalies, and LOF to work best for local and dependency anomalies. In the same year, Soenen et al. (2021) study the effect of hyperparameter optimization strategies on the evaluation and propose to optimize hyperparameters on a small validation set, with evaluation on a much larger test set. In their comparison of 6 algorithms on 16 data sets, IF performs the best, closely followed by CBLOF/u-CBLOF ((unweighted-)Cluster-Based Local Out-

lier Factor) and kNN, while OCSVM (One-Class Support Vector Machine) performs worst unless optimized using a substantially larger validation set than the other algorithms.

Han et al. (2022) performed an extensive comparison of anomaly detection methods, including supervised and semi-supervised algorithms. They compare 14 unsupervised algorithms on 47 tabular data sets using out-of-the-box, that is suggested by algorithm author or implementation, hyperparameter settings. They subsample larger data sets to a maximum of 10.000 samples, duplicate samples for those data sets smaller than 1000 samples. They find no significant differences between unsupervised algorithms. While real-world data sets are being used, the anomalies in each data set are generated synthetically according to 4 different type definitions (see section 2.2), and they compare the performance for each different type. Additionally, they have analyzed more complex benchmark data sets used in CV and NLP, such as CIFAR10 (Krizhevsky and Hinton, 2009) and the Amazon data set (He and McAuley, 2016) by performing neural-based feature extraction.

Other studies are of a more limited scope, and cover for example methods for high-dimensional data (Xu et al., 2018), or consider only ensemble methods (Zimek et al., 2014).

The studies done by Campos et al. (2016); Goldstein and Uchida (2016); Domingues et al. (2018); Steinbuss and Böhm (2021); Soenen et al. (2021); Han et al. (2022) have several limitations when used as a benchmark. Firstly, with the exception of Han et al., all studies were done on a rather small collection of data sets. Secondly, these studies cover only a small number of methods. Campos et al. compare only kNN-based approaches, while Goldstein and Uchida fail to cover many of the methods that have gained traction in the last few years, such as IF (Liu et al., 2008) and variants thereof (Hariri et al., 2019). Soenen et al. consider just 6 commonly used methods, Steinbuss and Böhm cover 4 methods and Han et al. cover 14 unsupervised methods.

Some of these studies consider the performance on data sets containing specific types of anomalies, such as global or local anomalies. Specifically, Steinbuss and Böhm look at the performance of different algorithms on data sets containing synthesized global, local, and dependency anomalies. Similarly, Han et al. synthesize these three types of anomalies as well as cluster anomalies for use in their comparison. Goldstein and Uchida's study is, to the best of our knowledge, the only one that analyzes real-world, that is, non-synthesized global and local anomalies. In particular, they analyze the 'pen-local' and 'pen-global' data set, two variants of the same data set where different classes were selected to obtain local and global anomalies specifically.

In practice, very little is known regarding what types of anomalies are present in commonly used benchmark data sets, and thus large scale comparisons on real-world data for specific types of anomalies are still missing. In this study we apply a large number of commonly used anomaly detection methods on a large collection of multivariate data sets, to discover guidelines on when to apply which algorithms. We explicitly choose to perform no optimization of hyperparameters, so as to evaluate the performance of algorithms in a truly unsupervised manner. Instead, we evaluate every algorithm over a range of sensible hyperparameters, and compare average performances. This contrasts with Soenen et al. (2021), who perform extensive optimization on a small validation set and thereby supply guidelines for semi-supervised detection or active learning. Our approach rather is similar to that used by Domingues et al. (2018), who also compare out-of-the-box performance. To the best of our knowledge, ours is the largest study of its kind performed so far.

2. Background

2.1 Unsupervised Anomaly Detection

Most anomaly detection tasks, including that done in this study, are conducted unsupervised. That means that no labels are available to the user. Consequently this means that regular optimization, like grid searches for optimal hyperparameters used in supervised learning, are not used within unsupervised anomaly detection. Most unsupervised anomaly detection algorithms produce scores, rather than labels, to samples. The most common convention is that a higher score indicates a higher likelihood that a sample is an anomaly, making unsupervised anomaly detection a ranking problem.

2.2 Types of Anomalies

Many different definitions of anomalies and their properties exist, many of these have been defined in an isolated context, not considering the relationships with other definitions or properties (Breunig et al., 2000). More recently, a review by Foorthuis (2021) tried to unify definitions across multiple subdomains of anomaly detection in order to encompass all types of anomalies. These definitions however do not encompass many properties or types of anomalies, such as clustered or dependency anomalies.

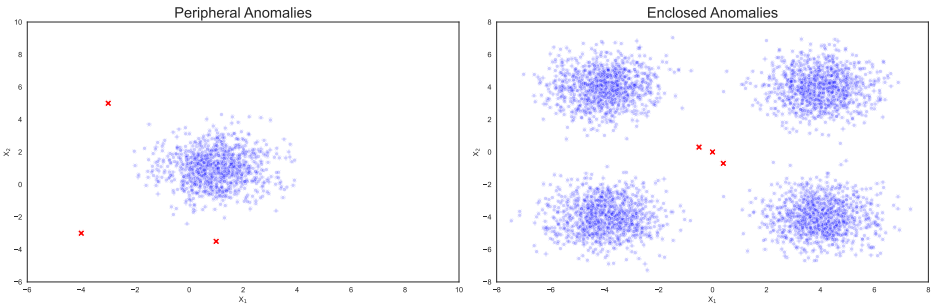
Rather than aiming to redefine every distinct type of anomaly, we treat anomalies as being able to have multiple, sometimes non-exclusive properties. Instead, we define four scales of non-exclusive properties which, when combined, encompass all types of anomalies found in multivariate tabular data in literature known to us.

2.2.1 Enclosed and Peripheral Anomalies

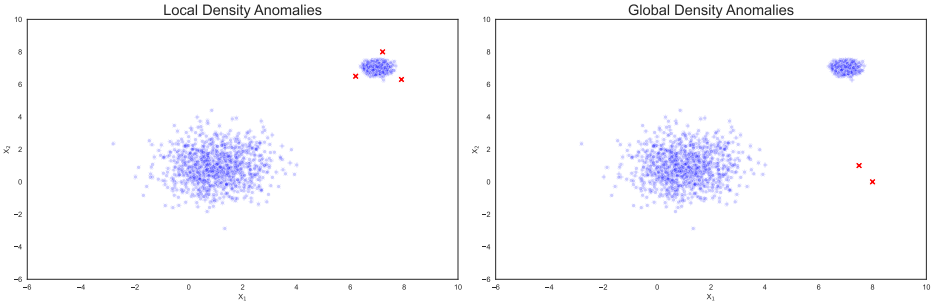
Anomalies can be surrounded in the feature space by normal data. When this occurs, we define them as enclosed anomalies. On the other end of this axis, peripheral anomalies occupy the edges of the feature space, and have one or more attribute scores either below the minimum or above the maximum of the scores of the normal data region. Examples of both enclosed and peripheral anomalies can be found in Figure 1a.

2.2.2 Global and Local Density Anomalies

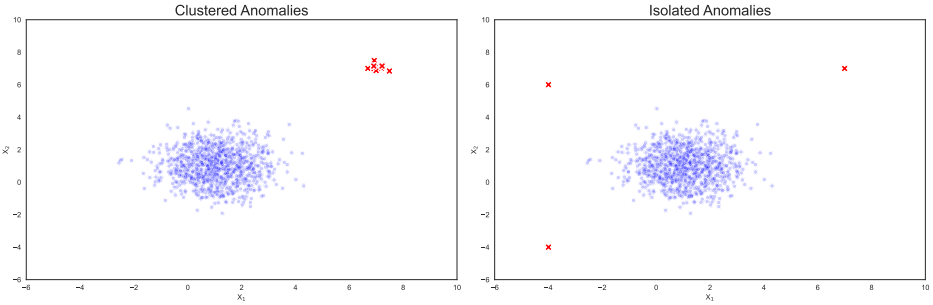
The most commonly discussed types of anomalies are the global and local anomalies. These definitions follow from the work of Breunig et al. (2000). Global anomalies are points which can be isolated from normal data because they occupy a globally low density region of the feature space. Local anomalies however, cannot be separated using just these criteria. This stems from the fact that density estimates are often imperfect, and based on a proxy measure, such as (average) distance. Local anomalies rather are located in regions with a density which is low compared to nearby normal regions, so just the distance as a proxy for density would fail to identify these points. Local anomalies occur when multiple clusters with differing density functions exist in the feature space. Examples of both global and local density anomalies can be found in Figure 1b . Specifically in the "local anomaly example", the red marked anomalies occupy a space close to the dense cluster, but where the density is low.



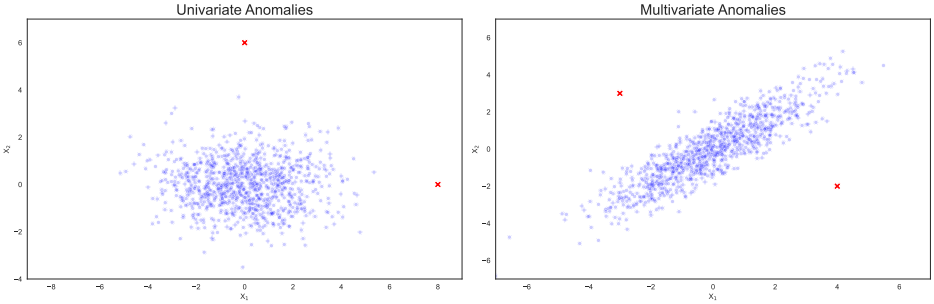
(a) Examples of peripheral (left) and enclosed (right) anomalies



(b) Examples of local (left) and global (right) anomalies



(c) Examples of clustered (left) and isolated (right) anomalies



(d) Examples of univariate (left) and multivariate (right) anomalies

Figure 1: 8 examples of different types of anomalies along the 4 defined property axes. Normal data are visualized as blue points, while anomalies are visualized as red crosses.

2.2.3 Isolated and Clustered Anomalies

Most often, anomalies are isolated, and are single datapoints without any additional, normal or anomaly, datapoints nearby. In many practical cases, anomalies are not that singular, and small groups of anomalies form clusters, leading to clustered anomalies. Clustered anomalies are closely related to the phenomenon known as "masking", where similar anomalies mask each other's presence by forming a cluster (Liu et al., 2008). Examples of both isolated and clustered anomalies can be found in Figure 1c.

2.2.4 Univariate and Multivariate Anomalies

Some anomalies are clearly univariate in nature. That is, they can be identified by just a single feature score in an anomalous range. Other anomalies are multivariate in nature, requiring a specific combination of feature scores to be identified as anomalies. These multivariate anomalies are also often called dependency anomalies, as they differ from the normal dependency, or causal, structure of the data. Examples of both isolated and clustered anomalies can be found in Figure 1d.

3. Materials and Methods

We evaluate the effectiveness of 33 different algorithms, listed in Table 1. We evaluate each algorithm multiple times for each data set, each with a different set of sensible hyperparameters. The results are then averaged across hyperparameter settings, leading to a single average ROC-AUC score for each method-data set combination. We refrain from optimizing hyperparameters, for example, using cross-validation, to reflect the real-world situation in which no labels are available for training the models. It should be noted, that the neural network architectures covered in this study are optimized with loss functions not using the original class labels, autoencoders for example use the mean squared error between the original and reconstructed features. While unsupervised optimization of hyperparameters has been studied by Thomas et al. (2016) it has not been applied to most algorithms considered in this study.

3.1 Algorithms

Of the 33 methods, 27 were used as implemented in the popular Python library for anomaly detection, PyOD (Zhao et al., 2019). As part of this research, we made several contributions to this open source library, such as a memory-efficient implementation of the COF (Connectivity-based Outlier Factor) method, as well as an implementation of the Birgé and Rozenholc (2006) method for histogram size selection, which is used in HBOS and LODA (lightweight on-line detector of anomalies). For EIF (extended isolation forest) we used the implementation provided by the authors in the Python package "eif" by Hariri et al. (2019). We implemented the ODIN (Outlier Detection using Indegree Number) method in Python, and it is being prepared as a submission to the PyOD package. The ensemble-LOF method, which implements the LOF score calculation in line with the original paper Breunig et al. (2000) was implemented using the base LOF algorithm from PyOD. DeepSVDD is applied

Name	Hyperparameters	Publication
ABOD	FastABOD, $k = 60$	Kriegel et al. (2008)
AE	$n_{\text{layers}} = 1, 2, 3$, shrinkage factor= $0.2, 0.3, 0.5$	Japkowicz et al. (1995)
ALAD	$n_{\text{layers}} = 3$, shrinkage factor= $0.2, 0.3, 0.5$	Zenati et al. (2018)
CBLOF	$k = 2, 5, 10, 15$, $\alpha = 0.7, 0.8, 0.9$, $\beta = 3, 5, 7$	He et al. (2003)
COF	$k = 5, 10, 15, 20, 25, 30$	Tang et al. (2002)
COPOD		Li et al. (2020)
DeepSVDD	$n_{\text{layers}} = 1, 2, 3$, shrinkage factor= $0.2, 0.3, 0.5$	Ruff et al. (2019)
DynamicHBOS		Goldstein and Dengel (2012)
ECOD		Li et al. (2022)
EIF	$n_{\text{trees}} = 1000$, $n_{\text{samples}} = 128, 256, 512, 1024$, no replacement, extension levels: $1, 2, 3$	Hariri et al. (2019)
ensemble-LOF	maximum LOF score over $k = 5, ..., 30$	Breunig et al. (2000)
gen2out		Lee et al. (2021)
GMM	$n_{\text{gaussians}} = 2, ..., 14$	Agarwal (2007)
HBOS	n_{bins} based on Birgé-Rozenholc criterium	Goldstein and Dengel (2012)
IF	$n_{\text{trees}} = 1000$, $n_{\text{samples}} = 128, 256, 512, 1024$, no replacement	Liu et al. (2008)
INNE	200 estimators	Bandaragoda et al. (2018)
KDE	Gaussian kernel	Latecki et al. (2007)
kNN	$k = 5, 8, ..., 29$, mean distance	Ramaswamy et al. (2000)
kth-NN	$k = 5, 8, ..., 29$, largest distance	Ramaswamy et al. (2000)
LMDD	$n_{\text{shuffles}} = 100$, MAD dissimilarity function	Arning et al. (1996)
LODA	n_{bins} based on Birgé-Rozenblac criterium, 100 random projections	Pevnỳ (2016)
LOF	$k = 5, 8, ..., 29$	Breunig et al. (2000)
LUNAR	$k = 5, 10, 15, 20, 25, 30$	Goodge et al. (2022)
MCD	subset fraction= $0.6, 0.7, 0.8, 0.9$	Rousseeuw and Driessen (1999)
OCSVM	RBF kernel, $\nu = 0.5, 0.6, 0.7, 0.8, 0.9$, $\gamma = 1/d$	Schölkopf et al. (1999)
ODIN	$k = 5, 8, ..., 29$	Hautamaki et al. (2004)
PCA	selected PCs explain $> 30, 50, 70, 90\%$ of variance	Shyu et al. (2003)
sb-DeepSVDD	$n_{\text{layers}} = 1, 2, 3$, shrinkage factor= $0.2, 0.3, 0.5$	Ruff et al. (2019)
SOD	$k = 20, 30$, $l = 10, 18$, $\alpha = 0.7, 0.9$	Kriegel et al. (2009)
SO-GAAL	stop epochs= 50	Liu et al. (2019)
u-CBLOF	$k = 2, 5, 10, 15$, $\alpha = 0.7, 0.8, 0.9$, $\beta = 3, 5, 7$	Amer and Goldstein (2012)
VAE	$n_{\text{layers}} = 1, 2, 3$, shrinkage factor= $0.2, 0.3, 0.5$	An and Cho (2015)
β-VAE	$n_{\text{layers}} = 1, 2, 3$, shrinkage factor= $0.2, 0.3, 0.5$, $\gamma = 10, 20, 50$	Zhou et al. (2020)

Table 1: Overview of the algorithms, the setting of the hyperparameters, the year of original publication, and the author(s). For the neural networks, the "shrinkage factor" hyperparameter indicates that any subsequent layer in the encoder is defined by: layer size$_{n+1}$ = layer size$_n$ $\times$ shrinkage factor.

based on the publicly available code by its author Lukas Ruff[1], and we modified it to work on general tabular data sets. The DynamicHBOS method was applied using the code by Kanatoko[2]

We left out several of the implemented methods in the PyOD package, such as LOCI and ROD, because they have a time or memory complexity of $\mathcal{O}(n^3)$, with n the number of data points. The PyOD SOS method was also ignored, due to its $\mathcal{O}(n^2)$ memory requirement. None of these methods performed notably well compared to the other algorithms that we included on the smaller data sets where evaluation was feasible. We have thoroughly optimized several of the slower methods in PyOD, specifically the SOD, COF, and LMDD methods.

1. The DeepSVDD Git repository can be found at `https://github.com/lukasruff/Deep-SVDD-PyTorch`.
2. The DynamicHBOS Git repository can be found at `https://github.com/Kanatoko/HBOS-python`.

3.2 Data

3.2.1 Datasets

In this study we consider a large collection of data sets from a variety of sources. We focus on real-valued, multivariate, tabular data, comparable to the data sets used by Fernández-Delgado et al. (2014); Campos et al. (2016); Goldstein and Uchida (2016); Soenen et al. (2021); Domingues et al. (2018). Table 2 contains a summary of the data sets, listing each data set's origin, number of samples, number of features, number and percentage of anomalies. While we recognize that other types of data, such as timeseries, categorical, or visual, are of interest, they cannot be readily compared in a single study.

Our collection consists for the most part of data sets from the ODDS data collection (Rayana, 2016), specifically the multi-dimensional point data sets. It is a collection of various data sets, mostly adapted from the UCI machine learning repository (Dua and Graff, 2017). All data sets are real-valued, without any categorical data. Curation of this collection is sadly not fully up-to-date, causing some of the listed data sets to be unavailable. The unavailable data sets were omitted from this comparison.

In addition to the ODDS data set, we also incorporate publicly available data sets used in earlier anomaly detection research. These include several data sets from the comparison by Goldstein and Uchida (2016), from the comparison of Campos et al. (2016) using ELKI (Schubert and Zimek, 2019), from a study on Generative Adversial Active Learning, or GAAL (Liu et al., 2019), from a study on extended Autoencoders (Shin and Kim, 2020), from the ADBench comparison (Han et al., 2022), and from a study on Efficient Online Anomaly Detection (EOAD) (Brandsæter et al., 2019). data sets from these latter sources that are (near-)duplicates of data sets present in the ODDS collection are left out. In Table 2 we specify exactly where each data set was downloaded or reconstructed from.

Emmott et al. (2013, 2015) present a systematic methodology to construct anomaly detection benchmarks, which is then also extensively applied by Pevnỳ (2016). In this paper, we chose not to construct our own benchmark data sets, which inevitably leads to some arbitrariness and possibly bias, but instead we rely on a large collection of different data sets used in earlier comparison studies. Synthetic data sets are not included in this study, as real-world data sets are generally considered the best available tool for benchmarking algorithms (Emmott et al., 2015; Domingues et al., 2018; Ruff et al., 2021). While real-world data sets are preferred for benchmarking, we note the usefulness of synthetic data in when studying specific properties of anomaly detection algorithms.

3.2.2 Preprocessing

Several steps have been undertaken to be able to compare the performance of the various algorithms on the different data sets. Most importantly all features in all data sets have been scaled and centered. Centering is done for each feature in a data set by subtracting the median. Scaling is performed by dividing each feature by its interquartile range. Our choice of centering and scaling procedure is deliberate, as both the median and interquartile range are influenced less by the presence of anomalies than the mean and standard deviation. This procedure is generally considered to be more stable than standardization when anomalies are known to be present (Rousseeuw and Croux, 1993). Our choice of scaling is further motivated because although some algorithms, such as Isolation Forest, can implicitly handle

Name	Origin	#samples	#features	#outliers	%outliers	#removed features
aloi	Goldstein	49999	27	1507	(3.01%)	0
annthyroid	ODDS	7200	6	534	(7.42%)	0
arrhythmia	ODDS	452	257	66	(14.6%)	17
breastw	ADBench	683	9	239	(34.99%)	0
campaign	ADBench	41188	62	4640	(11.27%)	0
cardio	ODDS	1831	21	176	(9.61%)	0
cover	ODDS	286048	10	2747	(0.96%)	0
donors	ADBench	619326	10	36710	(5.93%)	0
fault	ADBench	1941	27	673	(34.67%)	0
glass	ODDS	214	9	9	(4.21%)	0
hepatitis	ELKI	80	19	13	(16.25%)	0
hrss_anomalous_optimized	ex-AE	19634	18	4517	(23.01%)	0
hrss_anomalous_standard	ex-AE	23645	18	5670	(23.98%)	0
http	ODDS	567498	3	2211	(0.39%)	0
internetads	ELKI	1966	1555	368	(18.72%)	0
ionosphere	ODDS	351	33	126	(35.9%)	0
landsat	ADBench	6435	36	1333	(20.71%)	0
letter	ODDS	1600	32	100	(6.25%)	0
magic.gamma	ADBench	19020	10	6688	(35.16%)	0
mammography	ODDS	11183	6	260	(2.32%)	0
mi-f	ex-AE	25286	40	2161	(8.55%)	5
mi-v	ex-AE	25286	40	3942	(15.59%)	5
mnist	ODDS	7603	78	700	(9.21%)	22
musk	ODDS	3062	166	97	(3.17%)	0
nasa	ex-AE	4687	32	755	(16.11%)	0
optdigits	ODDS	5216	62	150	(2.88%)	2
pageblocks	ELKI	5393	10	510	(9.46%)	0
parkinson	ELKI	195	22	147	(75.38%)	0
pen-global	Goldstein	808	16	90	(11.14%)	0
pen-local	Goldstein	6723	16	10	(0.15%)	0
pendigits	ODDS	6870	16	156	(2.27%)	0
pima	ODDS	768	8	268	(34.9%)	0
satellite	ODDS	6435	36	2036	(31.64%)	0
satimage-2	ODDS	5803	36	71	(1.22%)	0
seismic-bumps	ODDS	2584	21	170	(6.58%)	3
shuttle	ODDS	49097	9	3511	(7.15%)	0
skin	ADBench	245057	3	50859	(20.75%)	0
smtp	ODDS	95156	3	30	(0.03%)	0
spambase	GAAL	4206	57	1678	(39.9%)	0
speech	ODDS	3686	400	61	(1.65%)	0
stamps	ELKI	340	9	31	(9.12%)	0
thyroid	ODDS	3772	6	93	(2.47%)	0
vertebral	ODDS	240	6	30	(12.5%)	0
vowels	ODDS	1456	12	50	(3.43%)	0
waveform	GAAL	3442	21	99	(2.88%)	0
wbc	ODDS	378	30	21	(5.56%)	0
wbc2	ADBench	223	9	10	(4.48%)	0
wilt	ELKI	4819	5	257	(5.33%)	0
wine	ODDS	129	13	10	(7.75%)	0
wpbc	ADBench	198	33	47	(23.74%)	0
yeast	ADBench	1484	8	507	(34.16%)	0
yeast6	EOAD	1484	8	35	(2.36%)	0

Table 2: Summary of the 52 multivariate data sets used in our anomaly detection algorithm comparison: the colloquial name of the data set, origin of the data set, the number of samples, features, and anomalies, as well as the percentage of anomalies, and the number of removed features.

features with different scales, methods that involve, for example, distance or cross-product calculations are strongly affected by the scale of the features.

3.3 Evaluation Procedure

In the unsupervised anomaly detection setting, it is generally more common and useful to evaluate anomaly scores, rather than binary labels as also produced by some algorithms. An anomaly score is a real-valued score, where a higher value indicates a higher likelihood, according to the score producing algorithm, that a specific sample is an anomaly. Using these scores, samples can be ranked according to apparent anomalousness, providing insights into the underlying nature of anomalies. For each data set, we calculate anomaly scores on all available data at once, without using any cross-validation or train-test splits, procedures common in the supervised setting. The scores from this unsupervised analysis are then compared to the ground truth labels, which indicates whether a sample is an anomaly (1), or not (0), to evaluate the performance of the algorithm. In order to compare the different algorithms we calculate the performance for each algorithm-data set combination in terms of the AUC (area under the curve) value resulting from the ROC (receiver operating characteristic) curve. This is the most commonly used metric in anomaly detection evaluations (Goldstein and Uchida, 2016; Campos et al., 2016; Xu et al., 2018), which can be readily interpreted from a probabilistic view. We considered using other metrics, such as the R-precision or average precision and their chance-adjusted variants introduced by Campos et al. (2016), but found these to be less stable, and harder to interpret.

For each data set we rank the AUC scores calculated from the scores produced by each algorithm. Following the recommendations for the comparison of classifiers by Demšar (2006), we use the Iman-Davenport statistic (Iman and Davenport, 1980) in order to determine whether there is any significant difference between the algorithms. If this statistic falls below the desired critical value corresponding to a p-value of 0.05, we apply the Nemenyi post-hoc test (Nemenyi, 1963) to then assess which algorithms differ significantly from each other.

In some of the visualizations in this paper we plot the percentage of maximum AUC, defined as

$$\widetilde{\mathrm{AUC}}(a,d) = \frac{\mathrm{AUC}(a,d)}{\max_{a' \in A} \mathrm{AUC}(a',d)} \times 100\,,$$

with a one of the algorithms and d one of the data sets.

3.4 Reproducibility

In order to reproduce all our experiments, we have provided access to a public GitHub repository[3] containing the code, including the optimized PyOD methods, and data sets used for all experiments as well as for the production of all figures and tables presented in this paper.

3. The Git repository can be found at: `https://github.com/RoelBouman/outlierdetection`.

4. Results

4.1 Overall Algorithm Performance

In order to gauge the performance, we evaluated each algorithm on each data set using the AUC measure corresponding to a ROC curve. To evaluate the performance across multiple sensible hyperparameters the AUC value for a given method is the average of the AUC of each hyperparameter setting evaluated. In our analysis, we found three data sets on which nearly every evaluated algorithm produced close to random results, that is with all AUC values between 0.4 and 0.6. These data sets, the 'hrss_anomalous_standard' and 'wpbc' data sets, were therefore excluded from further analysis. It is likely that these data sets contain no discernible anomalies. The existence of newer versions of these data sets, further motivates our choice of removal. Some data sets showed no AUC values above 0.6, but did show AUC values below 0.4. In these cases, the detector performs better when the labels are inverted. This behaviour was observed in the 'yeast', 'skin' and 'vertebral' data sets. The original construction of the latter two data sets was done based on treating the largest group of samples as the normal (label 0) class, and the smaller group as the anomaly (label 1) class, as is done commonly in anomaly detection research. Yet, for both these sets, the more heterogeneous group is chosen as the normal class, in contrast to normal anomaly definitions. For these data sets, we inverted the labelling and recalculated the AUC values to be more in line with the general anomaly property that anomalies are more heterogenous than normal data.

Figure 2 shows the distribution of the performance for each method. In order to compare the AUC across different data sets, which might have different baseline performances, in a boxplot we express the AUC in terms of its percentage of the maximum AUC value obtained by the best performing algorithm on that particular data set.

It can be seen that many of the algorithms perform comparably, with a median percentage of maximum AUC around 90%. Several lower medians, as well as wider quartiles, can be observed.

To determine whether any of the observed differences in performance from Figure 2 are significant, we apply the Iman-Davenport test. This yielded a test-statistic of 16.395 , far above the critical value of 0.625 , thus refuting the null hypothesis. We then applied the Nemenyi post-hoc test to establish which algorithms significantly outperform which other algorithms. The results of this Nemenyi post-hoc test are summarized in Table 3.

Table 3 reveals that there are indeed several algorithms significantly outperforming many other algorithms. Most notable here is EIF, which significantly outperforms 14/16 algorithms evaluated in this study at the $p = 0.05/p = 0.10$ significance level respectively. Since the computational complexity of Isolation Forest and variants thereof scales linearly with the number of samples n, this may give them a clear further edge over methods such as kNN and derivatives for large data sets, with a computational complexity that scales quadratically or at best with $\mathcal{O}(n \log n)$ when optimized.

From Figure 2 and Table 3 we can also observe that the original CBLOF method is by far the worst performing method based on the mean AUC, being significantly outperformed by 22 at the $p = 0.05$ significance level. This corroborates the results of Goldstein and Uchida (2016), who also found CBLOF to consistently underperform, while its unweighted variant, u-CBLOF, performs comparably to other algorithms. Notable is also

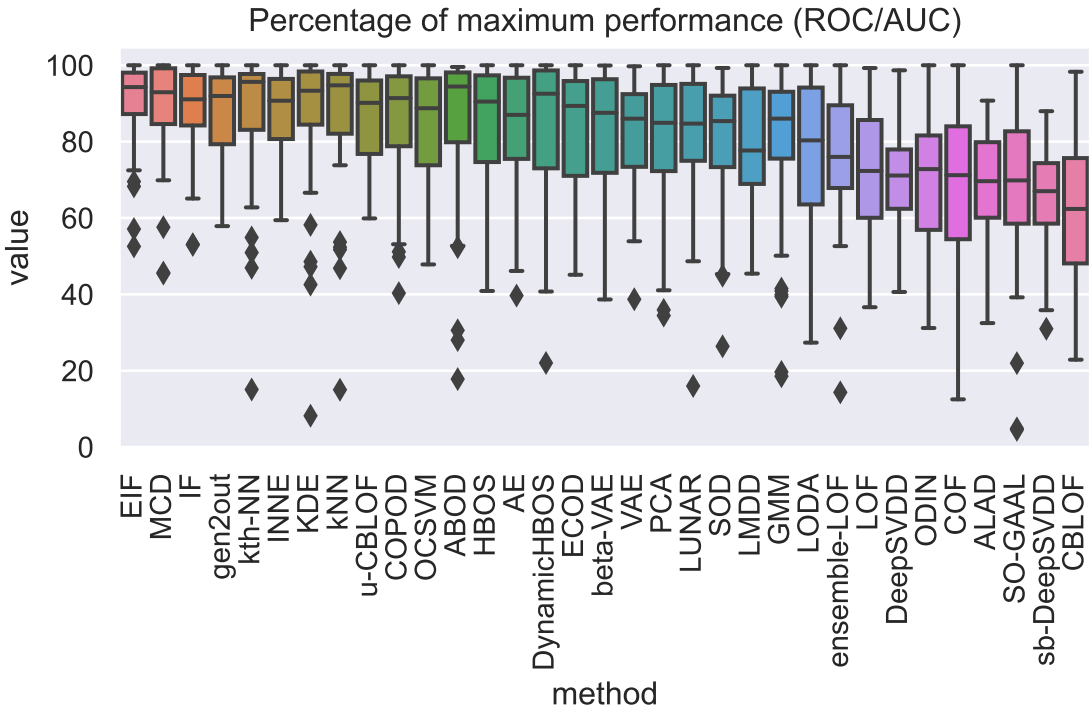


Figure 2: Boxplots of the performance of each algorithm on each data set in terms of percentage of maximum AUC. The maximum AUC is the highest AUC value obtained by the best performing algorithm on that particular data set. The whiskers in the boxplots extend 1.5 times the interquartile range past the low and high quartiles. data set-algorithm combinations outside of the whiskers are marked as diamonds.

	CBLOF	sb-DeepSVDD	SO-GAAL	ALAD	COF	ODIN	DeepSVDD	LOF	ensemble-LOF	LODA	GMM	LMDD	SOD	LUNAR	PCA	VAE	**Mean AUC**
EIF	++	++	++	++	++	++	++	++	++	++	+	++	++	+	++	++	0.770
MCD	++	++	++	++	++	++	++	++	++	++		++					0.762
IF	++	++	++	++	++	++	++	++	++	++		++					0.759
gen2out	++	++	++	++	++	++	++	++		+							0.747
kth-NN	++	++	++	++	++	++	++	++	++	++		++	++		+		0.745
INNE	++	++	++	++	++	++	++	++									0.743
KDE	++	++	++	++	++	++	++	++		++							0.743
kNN	++	++	++	++	++	++	++	++	++	++		++					0.741
u-CBLOF	++	++	++	++	++	++	++	+									0.740
COPOD	++	++	++	++	++	++	++	++									0.729
OCSVM	++	++	++	++	++	++	++										0.723
ABOD	++	++	++	++	++	++	++	++									0.721
HBOS	++	++	++	++	++	++	++										0.720
AE	++	++	++	++		+	++										0.719
DynamicHBOS	++	++	++	++	++	++	++										0.717
ECOD	++	++	++	++		+	++										0.713
beta-VAE	++	++	++	++			++										0.711
VAE	++	++	+	++			+										0.704
PCA	++	++		++													0.700
LUNAR	++	++	++	++			++										0.691
SOD	++	++		+													0.682
LMDD		++															0.675
GMM	++	++	++	++			+										0.664
LODA		+															0.658
ensemble-LOF		++															0.643
LOF																	0.617
DeepSVDD											-			--		-	0.596
ODIN																	0.588
COF																	0.586
ALAD											--		-	--	--	--	0.576
SO-GAAL											--			--		-	0.575
sb-DeepSVDD									--	-	--	--	--	--	--	--	0.551
CBLOF											--		--	--	--	--	0.515

Table 3: Significant differences between algorithms based on Nemenyi post-hoc analysis. ++/+ denotes that the row algorithm outperforms the column algorithm at $p = 0.05/p = 0.10$ respectively, while -- denotes that the row algorithm is outperformed by the column algorithm at $p = 0.05$. Rows and columns are sorted by descending and ascending mean performance, respectively. Columns are not shown when the column algorithm is not outperformed by any other algorithms at $p = 0.05$ or $p = 0.10$. The last column shows the mean AUC.

the sb-DeepSVDD method, which performs slightly better in terms of mean AUC, but is significantly outperformed by 24/25 algorithms at the $p = 0.05/p = 0.10$ significance level respectively.

From these overall results it is clear that many of the neural networks do not perform well. (Soft-boundary) DeepSVDD, ALAD, and SO-GAAL all occupy the lower segment of overall method performance. We surmise that there are three likely reasons for this phenomenon. Firstly, these methods were not designed with tabular data in mind, and they can't leverage the same feature extraction capabilities that give them an edge on their typical computer vision tasks. Secondly, these methods are relatively complex, making it exceedingly hard to specify general hyperparameter settings and architectures which work on a large variety of data sets. Lastly, many of the data sets in this study are likely not sufficiently large enough to leverage the strengths of neural network approaches. Not all neural networks suffer from these problems equally, as the auto-encoder and variants, as well as LUNAR, perform about average. This is likely caused by a more straightforward architecture and optimisation criterion. More specifically, we suspect lack of convergence is a problem for the generative adversarial methods and DeepSVDD.

In addition to the neural networks, the local methods, such as LOF, ODIN, COF, and CBLOF, are some of the most underperforming methods. This result for LOF stands in stark contrast to the results of Campos et al. (2016), who found LOF to be among the best performing methods. This is most likely caused by their evaluation on a small number of data sets with a low percentage of anomalies, which causes LOF to suffer less from swamping or masking (Liu et al., 2008). We further study this finding in Section 4.3.

4.2 Clustering Algorithms and data sets

To visualize the similarities between algorithms on one hand, and the data sets on the other, Figure 3 shows a heatmap of the performance of each data set/algorithm combination and dendrograms of two hierarchical clusterings, one on the data sets, and one on the algorithms. For these clustering steps, the Pearson correlation was used as a distance measure, as this best shows how similar methods or data sets are when looking at the calculated performance. We furthermore used average linkage cluster analysis to construct more robust clusters. For the sake of visualisation the leaf orderings were optimized using the method of Bar-Joseph et al. (2001).

Figure 3 shows that many similar algorithms cluster together in an expected way, with families of algorithms forming their own clusters. Some interesting patterns can be observed at a larger level. For the algorithms, we obtain several fairly distinct clusters. Firstly, CBLOF is distinct, as it underperforms on nearly every data set. Similarly, underperforming methods such as SO-GAAL, GMM, and ALAD only cluster together at a large distance, indicating that they have little correlation to other methods. The local methods, COF, ensemble-LOF, LOF and ODIN, form a separate cluster. These algorithms, which are specifically designed to detect local anomalies, work well on a few (approximately a quarter) of the data sets, but do not perform well for most other data sets. We have a small cluster of kNN and related methods such as LUNAR and SOD, that performs decently for all data sets. Lastly, as can be seen seen in the top left half of Figure 3, a large cluster of

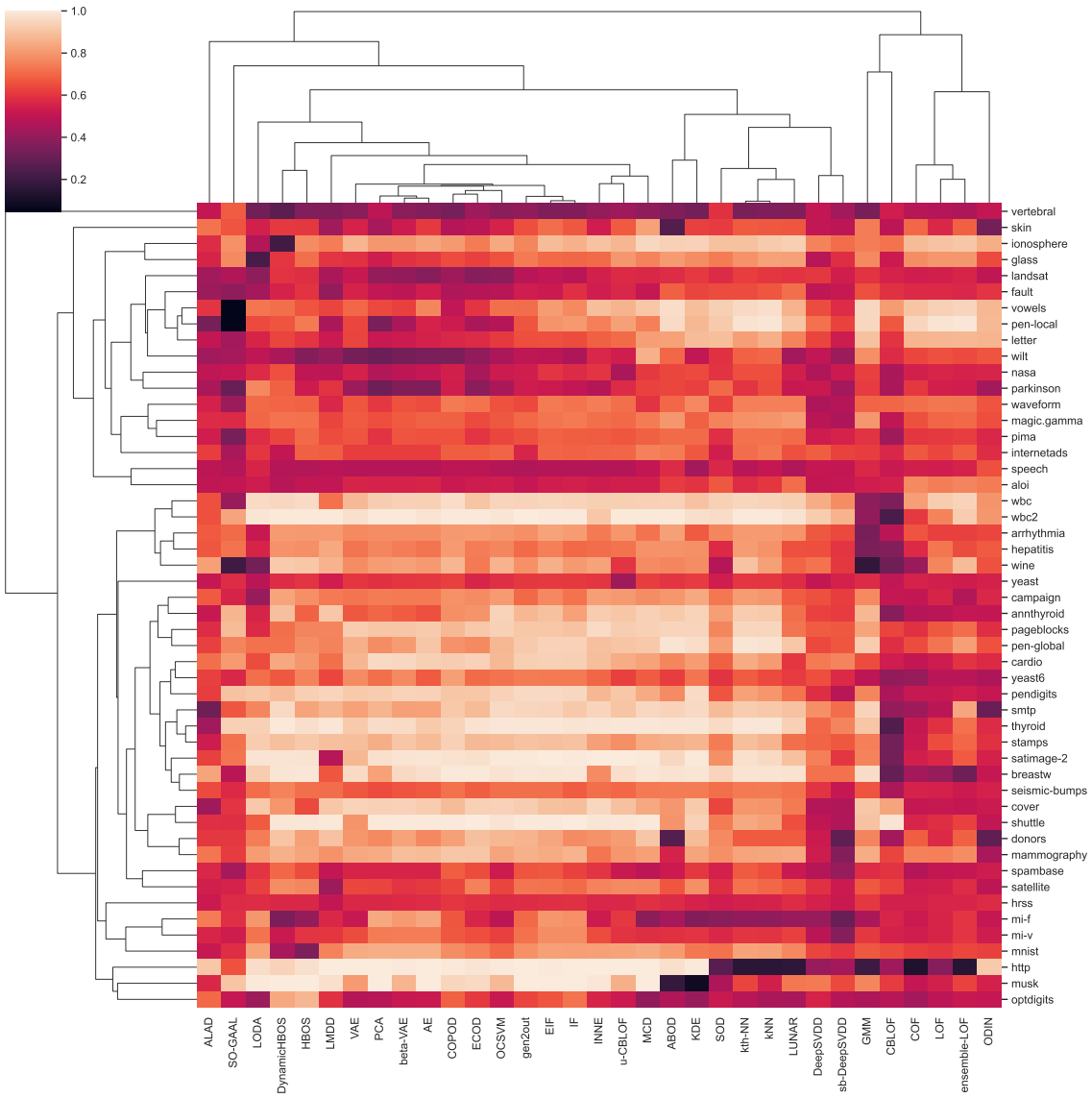


Figure 3: Clustered heatmap of the ROC/AUC performance of each algorithm. The algorithms and data sets are each clustered using hierarchical clustering with average linkage and the Pearson correlation as metric.

methods seems to negatively correlate with the local methods, performing well for most (approximately three-quarters) of the data sets, but less so for the remainder.

The data sets split into two clearly distinct clusters: one cluster of data sets on which the local algorithms perform well, and another cluster of data sets on which the large cluster of algorithms performs well. Combining the two-way clustering with knowledge of the algorithms suggests that approximately one third of the data sets comprises so-called local problems, while the other two-thirds comprises global problems. This is corroborated by specifically constructed local and global sets 'pen-local' and 'pen-global', that clearly belong to their expected clusters. This observation is corroborated by research by Steinbuss and Böhm (2021) and Emmott et al. (2015), who similarly find differences between what they categorize as local/dependency and multi-cluster anomalies respectively, and global anomalies. We cannot clearly observe any other clear patterns of different anomaly properties arising from our analysis. The 'vertebral' data set furthermore seems distinct from either the global or local clusters.

To the best of our knowledge, no previous study on naturally occurring anomalies in real-world data has looked, in detail, into the difference between the performance of algorithms when specifically being applied to either global or local anomaly detection problems.

4.3 Performance on Global and Local Problems

In the previous section, we discovered a clear distinction between two clusters of data sets: one with the "local" data sets 'aloi', 'fault', 'glass', 'internetads', 'ionosphere', 'landsat', 'letter', 'magic.gamma', 'nasa', 'parkinson', 'pen-local', 'pima', 'skin', 'speech', 'vowels', 'waveform', and 'wilt', and another with the remaining "global" data sets, excluding 'vertebral',. Suspecting that different methods may do well on different types of data sets, we repeated the significance testing procedure from Section 4.1 for both clusters separately.

Performance boxplots for all algorithms applied on the collection of local data sets can be found in Figure 4. Figure 4 clearly shows the reversed performance of some of the local methods for anomaly detection. Where COF, ensemble-LOF, and LOF were among the worst performers over the entire collection, they are among the best performers when applied to the problems for which they were specifically developed. This phenomenon is a fine example of Simpson's paradox (Simpson, 1951). This also partially explains the difference in findings of our overall comparison and the comparison of Campos et al. (2016).

We then repeated the Nemenyi-Friedman post hoc test on just the local data sets. The results for this analysis are summarized in Table 4. kNN is the top performers, and significantly outperform 17/18 other methods at the $p = 0.05/p = 0.10$ significance level respectively.

We then repeated the analysis for the global data sets, leading to the performance boxplots in Figure 5 and the significance results in Table 5.

From Figure 5 and Table 5 we can see that the Extended Isolation Forest has the highest mean performance, closely followed by the regular Isolation Forest. The Extended Isolation Forest outperforms 13/14 methods at $p = 0.05/p = 0.10$ respectively. Coincidentally, these methods also have the lowest computational and memory requirement, leaving them as the most likely choices for global anomalies.

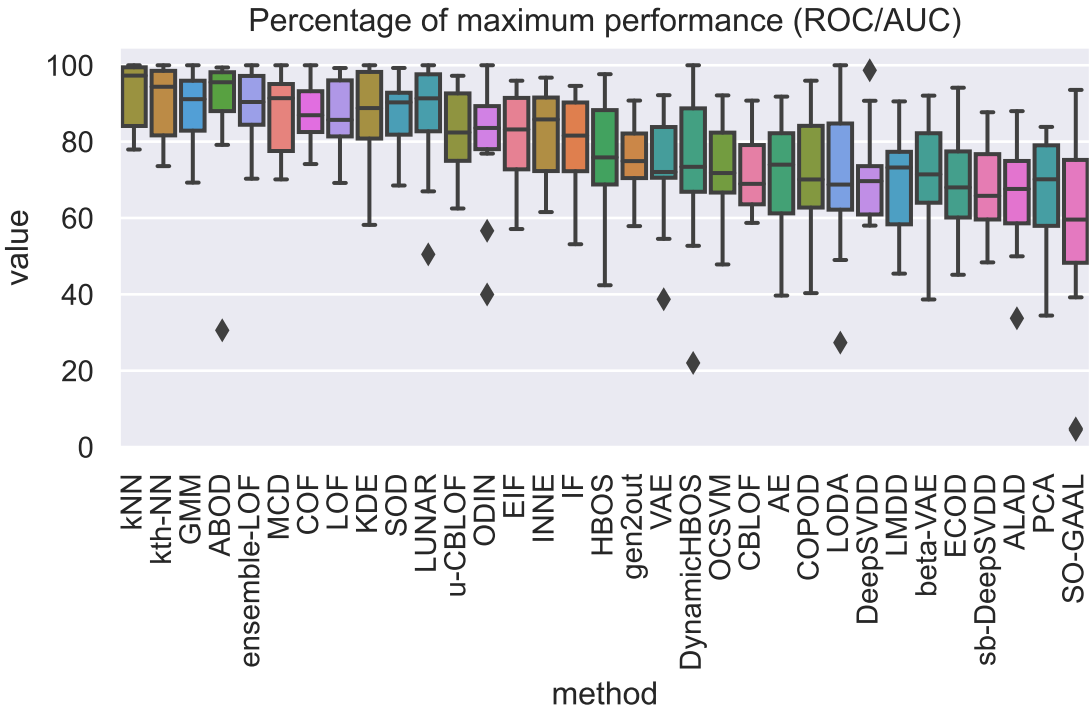


Figure 4: Boxplots of the performance of each algorithm on the "local" data sets in terms of percentage of maximum AUC. The maximum AUC is the highest AUC value obtained by the best performing algorithm on that particular data set. The whiskers in the boxplots extend 1.5 times the interquartile range past the low and high quartiles. data set-algorithm combinations outside of the whiskers are marked as diamonds.

	SO-GAAL	PCA	ALAD	sb-DeepSVDD	ECOD	beta-VAE	LMDD	DeepSVDD	LODA	COPOD	AE	CBLOF	OCSVM	DynamicHBOS	VAE	gen2out	HBOS	ODIN	**Mean AUC**
kNN	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++	+	0.737
kth-NN	++	++	++	++	++	++	++	++	++	++	++	++	++	+	++	++	+		0.722
GMM	++	++	++	++	++	++	++	+			+								0.710
ABOD	++	++	++	++	++	++	++	++	++	++	++	++	++		++	++			0.709
ensemble-LOF	++	++	++	++	++	++	++	++		+	++	+	+						0.708
MCD	++	++	++	++	++	++	++	+			+								0.702
COF	++	++	++	++	++	++	++												0.698
LOF	++	++	++	++	++	++	++												0.695
KDE	++	++	++	++	++	++	++	+			+								0.693
SOD	++	++	++	++	++	++	++												0.693
LUNAR	++	++	++	++	++	++	++												0.693
u-CBLOF	++	++	++	+															0.657
ODIN																			0.646
EIF	++	++	++																0.646
INNE	++	+	+																0.644
IF	+																		0.637
HBOS																			0.609
gen2out																			0.601
VAE																			0.585
DynamicHBOS																			0.578
OCSVM																			0.573
CBLOF																			0.566
AE																			0.565
COPOD																			0.561
LODA																			0.555
DeepSVDD																			0.554
LMDD																			0.539
beta-VAE																			0.539
ECOD																			0.538
sb-DeepSVDD																			0.531
ALAD																			0.514
PCA																			0.513
SO-GAAL																			0.444

Table 4: Significant differences between algorithms on the collection of local problems based on Nemenyi post-hoc analysis. ++/+ denotes that the row algorithm outperforms the column algorithm at $p = 0.05/p = 0.10$. Rows and columns are sorted by descending and ascending mean performance, respectively. Columns are not shown when the column algorithm is not outperformed by any other algorithms at $p = 0.05$ or $p = 0.10$. The last column shows the mean AUC.

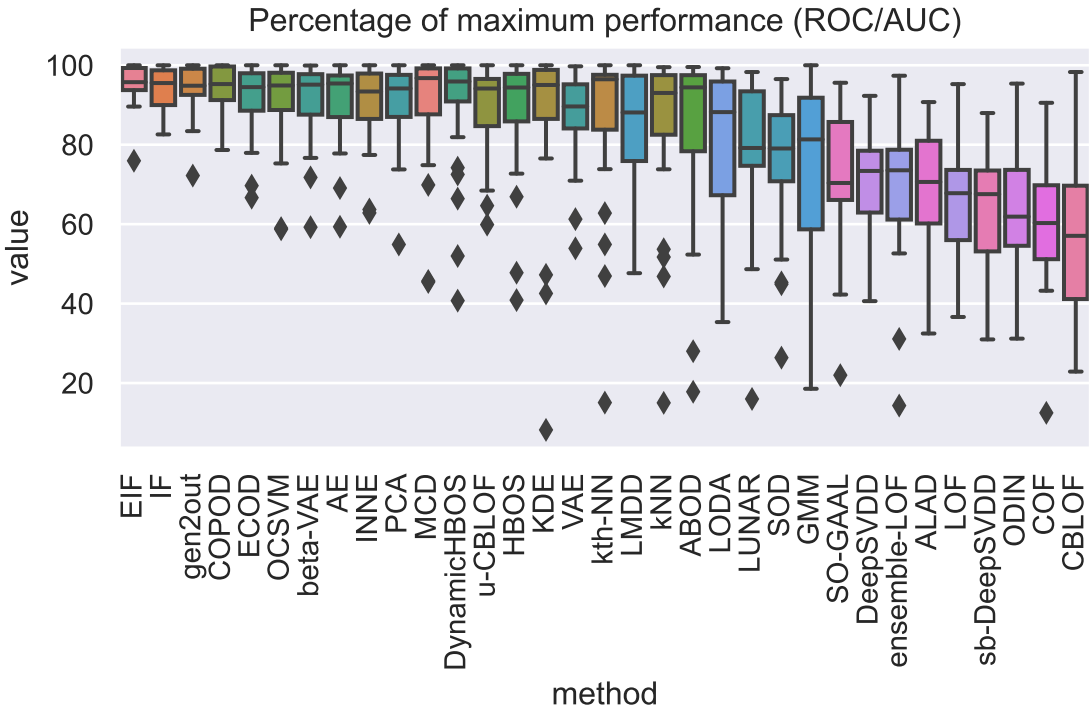


Figure 5: Boxplots of the performance of each algorithm on the global data sets in terms of percentage of maximum AUC. The maximum AUC is the highest AUC value obtained by the best performing algorithm on that particular data set. The whiskers in the boxplots extend 1.5 times the interquartile range past the low and high quartiles. data set-algorithm combinations outside of the whiskers are marked as diamonds.

	CBLOF	COF	ODIN	sb-DeepSVDD	LOF	ALAD	ensemble-LOF	DeepSVDD	SO-GAAL	GMM	SOD	LUNAR	LODA	ABOD	**Mean AUC**
EIF	++	++	++	++	++	++	++	++	++	++	++	++	++	+	0.849
IF	++	++	++	++	++	++	++	++	++	++	++	++	++		0.837
gen2out	++	++	++	++	++	++	++	++	++	++	++	++	++		0.836
COPOD	++	++	++	++	++	++	++	++	++	++	++	++	++		0.831
ECOD	++	++	++	++	++	++	++	++	++	+	++				0.815
OCSVM	++	++	++	++	++	++	++	++	++		++				0.813
beta-VAE	++	++	++	++	++	++	++	++	++		++				0.813
AE	++	++	++	++	++	++	++	++	++		++				0.812
INNE	++	++	++	++	++	++	++	++	++		++				0.807
PCA	++	++	++	++	++	++	++	++	++						0.807
MCD	++	++	++	++	++	++	++	++	++		++				0.805
DynamicHBOS	++	++	++	++	++	++	++	++	++		++				0.804
u-CBLOF	++	++	++	++	++	++	++	++	++						0.793
HBOS	++	++	++	++	++	++	++	++	++		+				0.790
KDE	++	++	++	++	++	++	++	++	++		++				0.782
VAE	++	++	++	++	++	++	+	++							0.778
kth-NN	++	++	++	++	++	++	++	++	++		++				0.770
LMDD	++	++	++	++	++	++		++							0.757
kNN	++	++	++	++	++	++	++	++	+						0.755
ABOD	++	++	++	++	++	++		++							0.740
LODA	++	++		++											0.724
LUNAR		+													0.701
SOD															0.679
GMM															0.650
SO-GAAL															0.642
DeepSVDD														--	0.621
ensemble-LOF															0.615
ALAD														--	0.611
LOF														--	0.580
sb-DeepSVDD													--	--	0.566
ODIN														--	0.560
COF												-	--	--	0.530
CBLOF													--	--	0.487

Table 5: Significant differences between algorithms on the collection of global problems based on Nemenyi post-hoc analysis. ++/+ denotes that the row algorithm outperforms the column algorithm at $p = 0.05/p = 0.10$. Rows and columns are sorted by descending and ascending mean performance, respectively. Columns are not shown when the column algorithm is not outperformed by any other algorithms at $p = 0.05$ or $p = 0.10$. The last column shows the mean AUC.

5. Discussion

In our study we compared the performance of anomaly detection algorithms on 52 semantically meaningful real-world tabular data sets, more than any other recent comparison studies (Campos et al., 2016; Goldstein and Uchida, 2016; Xu et al., 2018; Soenen et al., 2021; Steinbuss and Böhm, 2021; Domingues et al., 2018; Han et al., 2022). A somewhat comparable study by Fernández-Delgado et al. (2014) on classification algorithms easily considered 121 data sets. The main reason for this discrepancy is that data sets for comparing anomaly detection algorithms rarely include categorical features, which are not an issue for comparing classification algorithms. It is certainly possible to further extend the collection of data sets, for example, through data set modifications. Campos et al. (2016), Emmott et al. (2013, 2015), and Steinbuss and Böhm (2021) modified data sets in different ways to create similar data sets with differing characteristics from a single base data set. While such modifications can be useful for targeted studies, near-duplicate data sets are far from independent and then seem detrimental to a proper statistical comparison of anomaly detection algorithms, such as can be observed in Emmott et al. (2015).

In this study we compared 33 of the most commonly used algorithms for anomaly detection. This collection is certainly not exhaustive: many more methods exist (Schubert and Zimek, 2019; Goldstein and Uchida, 2016; Emmott et al., 2015; Ruff et al., 2021; Domingues et al., 2018), and likely even more will be invented. Also along this axis, there is a clear discrepancy with the study by Fernández-Delgado et al. (2014) on classification algorithms, who incorporated 179 classifiers from 17 different families. Apparently, the number of classification algorithms largely exceeds the number of anomaly detection algorithms. But perhaps more importantly, there are many more solid and easy-to-use implementations of classification algorithms in many different machine learning libraries than there are out-of-the-box implementations of anomaly detection algorithms. Before being able to perform the comparison in this study, we had to spend quite some effort to clean up and sometimes re-implement (parts of) existing code.

In this research we chose not to cover meta-techniques for ensembling. While ensembles are of great interest, a better understanding of the performance of base learners is an essential prerequisite before moving on to a study of ensemble methods.

While we evaluated neural networks in our comparison, no general guidelines exist on how to construct a well-performing network for any given data set, which is essential for the unsupervised setup considered in this study. Additionally, the strength of many of these methods comes from high-level feature extraction implicitly performed by the network, which cannot be leveraged on the smaller tabular data sets used in this benchmark. Like Ruff et al. (2021), we recognize that there is a major discrepancy between the availability of classification and anomaly detection benchmark data sets useful for deep learning approaches. More anomaly detection benchmark data sets useful for deep learning based anomaly detection would be a welcome addition to the field.

Cross-comparing the performance of algorithms on data sets, we noticed a clear separation between two clusters of data sets and roughly two clusters of corresponding algorithms. We characterized these clusters as "local" and "global" data sets and algorithms, in correspondence with common nomenclature in the literature (Breunig et al., 2000; Goldstein and Uchida, 2016). However, we are well aware that this characterization may turn out to be

an oversimplification when analyzing more data sets and more algorithms in closer detail. For example, the local and global problems likely have quite some overlap, but need not be fully equivalent with multimodal and unimodal anomaly detection problems, respectively. Overlap between multimodal and local problems occurs when the different modes start having different densities, so that local algorithms that try to estimate these local densities fare better than global algorithms that cannot make this distinction. Further theoretical and empirical studies, for example, on carefully simulated data sets are needed to shed further light. We acknowledge that there is a gap in both theoretical and empirical studies on determining what types of anomalies are present in a data set, which would directly help in selecting an appropriate algorithm in conjunction with this research. Furthermore, we have not readily observed several well-described properties of anomalies. This exemplifies the need for more, varied, benchmark data sets.

6. Conclusion

Based on our research we can establish general guidelines on when users should apply which anomaly detection methods for their problem.

In general, when a user has no *a priori* knowledge on whether or not their data set contains local or global anomalies, EIF, Extended Isolation Forest, is the best choice. It outperforms 14 out of 33 other evaluated methods at $p = 0.05$, and is one the highest performing method based on its mean AUC score.

When a data set is known or suspected to contain local anomalies, which might for example occur when the data is known to contain multiple different density clusters, the best performing method is kNN, which outperforms 17 out of 33 methods at $p = 0.05$.

Datasets containing just global anomalies are best analyzed using EIF, which is the top performing algorithm on the data sets containing global anomalies. COPOD, gen2out, INNE and k-thNN all perform comparably, and these methods all outperform at least 10 other methods at $p = 0.05$. IF and EIF are the algorithms with the lowest computational complexity, which are usually preferable in practice.

Contemplating the above considerations, we are tempted to answer the question in the title of our paper with "two": a toolbox with kNN, and EIF seems sufficient to perform well on the type of multivariate data sets considered in our study. These two algorithms are due to the scope of this study likely to perform well on unseen real-world multivariate data sets. This conclusion is open for further consideration when other algorithms and/or data sets are added to the bag, which should be relatively easy to check when extending the code and the data set pre-processing procedures that we open-sourced.

Future work following this study may seek to extend our comparative analysis with diverse types of data such as raw images, texts and time-series. All of these types of data require specific methods and tailored comparisons. Furthermore, automatically determining properties of anomalies in a data set before further analysis is an unexplored avenue of study which might provide users with even more detailed guidelines on which algorithm to apply.

Acknowledgments

The research reported in this paper has been partly funded by the NWO grant NWA.1160.18.238 (`https://primavera-project.com/`), as well as BMK, BMDW, and the State of Upper Austria in the frame of the SCCH competence center INTEGRATE [(FFG grant no. 892418)] part of the FFG COMET Competence Centers for Excellent Technologies Programme.

Appendix A. AUC Scores for Each Algorithm-data set Combination

Table 6 contains the first half of the combinations and Table 7 contains the second half of the combinations.

Appendix B. Nemenyi Post-hoc Analysis Results

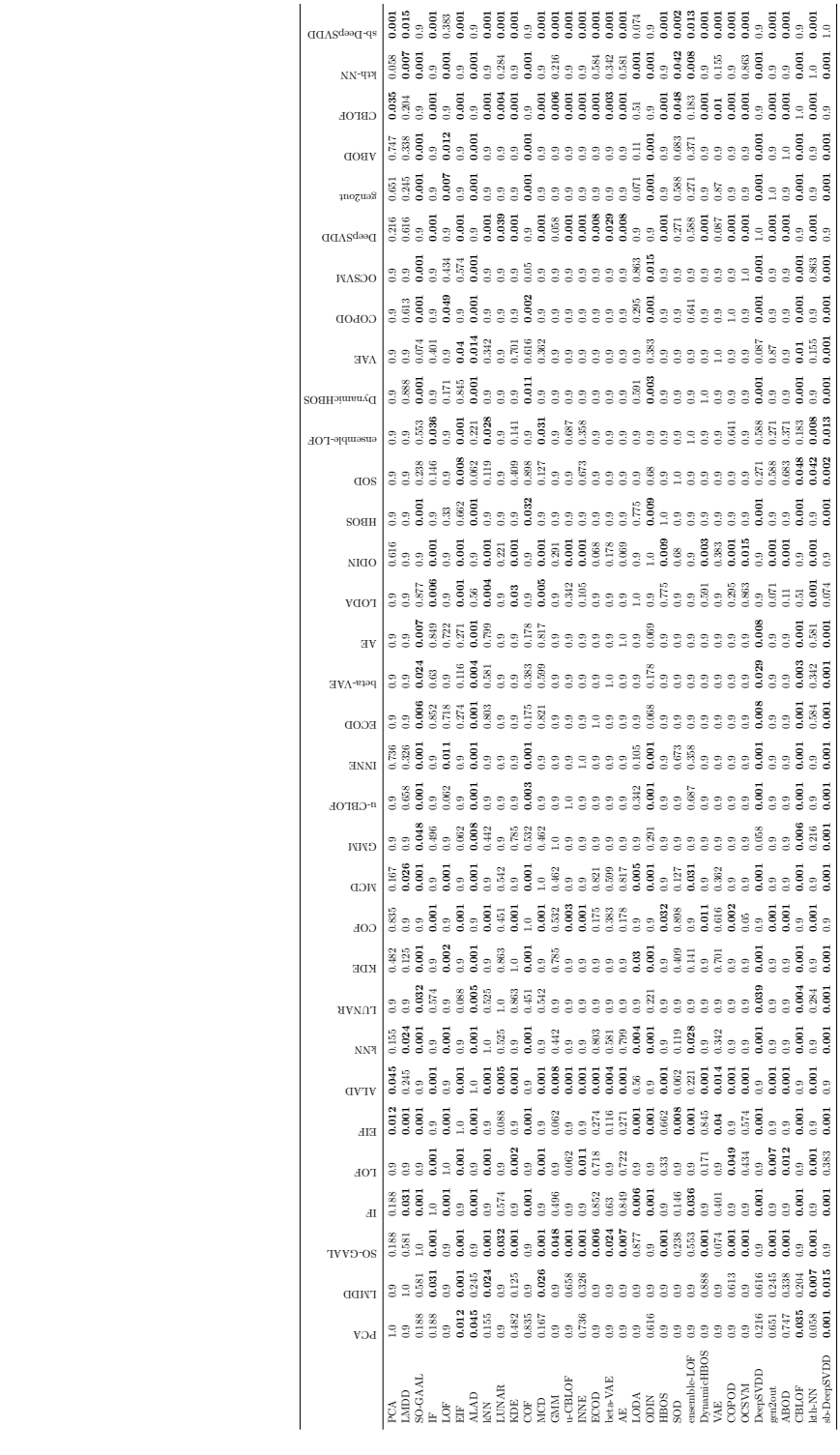
Table 8 shows the results for all data sets, Table 9 shows the results for the local data sets, and Table 10 shows the results for the global data sets.

	ionosphere	yeast	spambase	hrss	glass	seismic-bumps	donors	pageblocks	internetads	wilt	breastw	shuttle	pen-global	skin	satimage-2	fault	pima	mammography	smtp	hepatitis	mi-f	cover	parkinson	pen-local	arrhythmia
MCD	0.95	0.61	0.49	0.59	0.75	0.73	0.91	0.93	0.75	0.86	0.99	1.00	0.91	0.82	0.99	0.51	0.68	0.81	0.95	0.79	0.38	0.84	0.62	0.74	0.72
OCSVM	0.84	0.58	0.52	0.57	0.65	0.71	0.87	0.93	0.69	0.44	0.98	0.98	0.87	0.55	0.99	0.47	0.63	0.83	0.95	0.76	0.49	0.93	0.45	0.47	0.80
ODIN	0.85	0.55	0.53	0.57	0.64	0.55	0.28	0.58	0.57	0.66	0.51	0.50	0.66	0.33	0.55	0.59	0.56	0.44	0.30	0.67	0.51	0.52	0.43	0.88	0.63
SO-GAAL	0.78	0.58	0.44	0.57	0.77	0.57	0.60	0.88	0.45	0.44	0.48	0.58	0.76	0.66	0.72	0.39	0.33	0.61	0.66	0.72	0.58	0.60	0.30	0.05	0.70
LODA	0.47	0.48	0.59	0.58	0.22	0.71	0.74	0.57	0.60	0.49	0.92	0.67	0.72	0.54	0.98	0.43	0.58	0.81	0.76	0.55	0.80	0.91	0.76	0.64	0.51
DynamicHBOS	0.21	0.58	0.70	0.57	0.60	0.75	0.90	0.71	0.50	0.45	0.98	0.99	0.71	0.61	0.98	0.55	0.63	0.86	0.95	0.78	0.34	0.79	0.68	0.66	0.80
COPOD	0.80	0.62	0.68	0.60	0.64	0.71	0.82	0.88	0.68	0.34	0.99	0.99	0.79	0.47	0.97	0.46	0.65	0.91	0.91	0.80	0.66	0.88	0.54	0.52	0.80
kNN	0.92	0.60	0.65	0.55	0.82	0.74	0.67	0.95	0.73	0.69	0.98	0.82	0.98	0.67	0.95	0.66	0.71	0.83	0.91	0.79	0.39	0.79	0.98	0.98	0.80
kth-NN	0.89	0.60	0.67	0.55	0.79	0.73	0.67	0.95	0.71	0.69	0.98	0.84	0.98	0.67	0.97	0.64	0.71	0.83	0.92	0.81	0.39	0.79	0.60	0.98	0.80
PCA	0.80	0.59	0.55	0.57	0.53	0.64	0.77	0.90	0.61	0.31	0.82	0.99	0.78	0.44	0.97	0.50	0.59	0.84	0.87	0.76	0.83	0.93	0.31	0.34	0.78
geu2out	0.76	0.60	0.66	0.57	0.73	0.71	0.86	0.91	0.56	0.49	0.99	0.99	0.92	0.65	0.99	0.53	0.65	0.86	0.94	0.70	0.70	0.92	0.53	0.66	0.78
GMM	0.74	0.57	0.58	0.60	0.76	0.66	0.75	0.83	0.61	0.78	0.96	0.90	0.92	0.74	0.75	0.65	0.59	0.85	0.94	0.35	0.45	0.90	0.61	0.95	0.34
DeepSVDD	0.75	0.50	0.47	0.55	0.48	0.59	0.52	0.67	0.68	0.50	0.71	0.49	0.64	0.50	0.75	0.50	0.53	0.53	0.78	0.65	0.41	0.47	0.52	0.72	0.67
SOD	0.89	0.53	0.54	0.53	0.74	0.71	0.77	0.76	0.53	0.59	0.94	0.74	0.81	0.61	0.78	0.64	0.58	0.78	0.88	0.57	0.37	0.63	0.70	0.92	0.76
KDE	0.93	0.61	0.60	0.55	0.77	0.74	0.88	0.94	0.66	0.50	0.98	0.93	0.96	0.62	0.99	0.63	0.70	0.83	0.95	0.78	0.36	0.92	0.62	0.87	0.67
CBLOF	0.72	0.50	0.59	0.54	0.52	0.45	0.43	0.63	0.56	0.57	0.29	0.98	0.58	0.49	0.33	0.59	0.42	0.63	0.40	0.36	0.56	0.83	0.45	0.67	0.48
INNE	0.90	0.60	0.59	0.56	0.70	0.71	0.79	0.96	0.69	0.56	0.77	0.98	0.89	0.53	1.00	0.53	0.67	0.72	0.94	0.71	0.52	0.95	0.47	0.74	0.73
AE	0.83	0.60	0.55	0.58	0.60	0.67	0.76	0.92	0.61	0.34	0.97	0.99	0.90	0.60	0.98	0.53	0.66	0.88	0.82	0.72	0.83	0.92	0.35	0.55	0.77
LOF	0.90	0.54	0.50	0.56	0.79	0.54	0.57	0.67	0.63	0.65	0.40	0.58	0.72	0.57	0.56	0.57	0.61	0.74	0.48	0.72	0.55	0.51	0.54	0.98	0.62
IF	0.86	0.61	0.62	0.59	0.69	0.67	0.77	0.90	0.68	0.45	0.99	1.00	0.93	0.67	0.99	0.58	0.68	0.86	0.91	0.70	0.78	0.89	0.50	0.80	0.81
ECOD	0.74	0.56	0.64	0.59	0.62	0.68	0.89	0.91	0.68	0.39	0.99	1.00	0.77	0.49	0.97	0.47	0.59	0.91	0.88	0.74	0.56	0.93	0.38	0.45	0.81
ABOD	0.93	0.60	0.52	0.57	0.81	0.74	0.25	0.94	0.74	0.69	0.98	0.79	0.98	0.25	0.95	0.68	0.70	0.55	0.94	0.79	0.44	0.80	0.63	0.96	0.82
ensemble-LOF	0.90	0.53	0.51	0.56	0.80	0.52	0.68	0.71	0.68	0.64	0.31	0.62	0.81	0.68	0.67	0.56	0.61	0.78	0.83	0.64	0.59	0.50	0.54	0.99	0.62
sb-DeepSVDD	0.71	0.54	0.41	0.52	0.58	0.51	0.28	0.66	0.63	0.41	0.72	0.46	0.66	0.49	0.60	0.51	0.55	0.36	0.71	0.60	0.30	0.46	0.44	0.63	0.63
u-CBLOF	0.87	0.42	0.51	0.57	0.80	0.70	0.81	0.92	0.69	0.53	0.97	0.99	0.90	0.67	1.00	0.56	0.66	0.79	0.90	0.72	0.60	0.89	0.57	0.81	0.79
LUNAR	0.93	0.58	0.51	0.58	0.82	0.72	0.67	0.72	0.65	0.43	0.98	0.65	0.91	0.66	0.89	0.71	0.70	0.83	0.90	0.65	0.41	0.73	0.51	0.90	0.79
HBOS	0.77	0.59	0.64	0.56	0.71	0.71	0.81	0.75	0.68	0.36	0.99	0.99	0.77	0.60	0.98	0.59	0.70	0.83	0.83	0.78	0.40	0.64	0.52	0.73	0.80
VAE	0.86	0.60	0.56	0.57	0.59	0.67	0.79	0.92	0.61	0.33	0.96	0.79	0.85	0.55	0.87	0.55	0.66	0.84	0.82	0.72	0.51	0.94	0.42	0.63	0.77
COF	0.88	0.55	0.47	0.54	0.78	0.55	0.68	0.59	0.66	0.63	0.43	0.56	0.64	0.70	0.54	0.57	0.60	0.75	0.42	0.57	0.52	0.51	0.60	0.95	0.66
LMDD	0.74	0.53	0.50	0.50	0.59	0.71	0.90	0.76	0.68	0.40	0.66	0.99	0.75	0.43	0.48	0.40	0.64	0.78	0.86	0.83	0.56	0.91	0.59	0.45	0.80
ALAD	0.57	0.51	0.56	0.52	0.55	0.64	0.60	0.58	0.61	0.43	0.83	0.58	0.62	0.72	0.62	0.41	0.54	0.72	0.31	0.66	0.74	0.42	0.45	0.33	0.67
beta-VAE	0.79	0.60	0.55	0.57	0.59	0.66	0.82	0.91	0.61	0.33	0.95	0.99	0.81	0.52	0.98	0.49	0.66	0.89	0.82	0.74	0.80	0.93	0.36	0.44	0.77
EIF	0.88	0.61	0.67	0.58	0.70	0.71	0.83	0.91	0.69	0.49	0.98	1.00	0.94	0.71	1.00	0.52	0.69	0.84	0.94	0.75	0.79	0.89	0.52	0.79	0.82

Table 6: The AUC values for the first half of the algorithm-data set combinations.

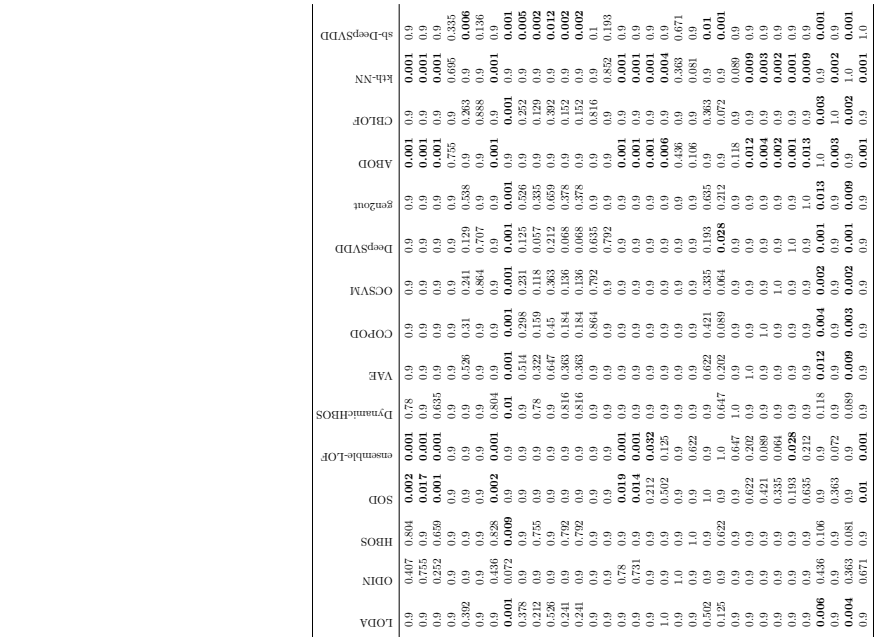
	musk	wbc2	vertebral	mnist	pendigits	stamps	nasa	wbc	optdigits	waveform	landsat	cardio	satellite	yeast6	campaign	speech	letter	wine	ali	mi-v	vowels	thyroid	annthyroid	http	magic.gamma
MCD	1.00	0.99	0.39	0.83	0.84	0.84	0.61	0.93	0.39	0.59	0.56	0.82	0.76	0.67	0.77	0.50	0.81	0.77	0.54	0.58	0.91	0.99	0.93	1.00	0.74
OCSVM	0.87	0.98	0.38	0.70	0.93	0.88	0.52	0.94	0.51	0.65	0.38	0.88	0.63	0.72	0.80	0.47	0.61	0.86	0.54	0.62	0.71	0.99	0.94	0.99	0.67
ODIN	0.55	0.79	0.50	0.62	0.51	0.59	0.56	0.79	0.51	0.66	0.49	0.57	0.49	0.45	0.56	0.65	0.86	0.66	0.74	0.53	0.87	0.57	0.50	0.92	0.65
SO-GAAL	0.90	0.83	0.67	0.58	0.90	0.71	0.51	0.41	0.50	0.41	0.46	0.80	0.55	0.56	0.56	0.47	0.43	0.20	0.50	0.53	0.04	0.93	0.86	0.66	0.57
LODA	0.99	0.98	0.32	0.82	0.91	0.92	0.57	0.96	0.42	0.70	0.39	0.64	0.62	0.71	0.40	0.55	0.55	0.32	0.53	0.72	0.72	0.93	0.55	0.97	0.69
DynamicHBOS	1.00	0.99	0.27	0.44	0.88	0.89	0.65	0.95	0.83	0.69	0.59	0.84	0.77	0.67	0.83	0.47	0.63	0.92	0.47	0.51	0.71	0.99	0.89	0.94	0.72
COPOD	0.95	0.99	0.33	0.77	0.90	0.93	0.54	0.96	0.68	0.73	0.42	0.92	0.63	0.81	0.78	0.49	0.56	0.87	0.51	0.67	0.50	0.94	0.78	0.99	0.68
kNN	0.54	0.99	0.35	0.81	0.79	0.86	0.65	0.94	0.45	0.74	0.60	0.80	0.70	0.64	0.80	0.51	0.85	0.81	0.60	0.59	0.97	0.99	0.92	0.15	0.79
kth-NN	0.63	0.99	0.34	0.82	0.82	0.89	0.65	0.94	0.48	0.74	0.61	0.84	0.71	0.67	0.81	0.48	0.81	0.90	0.58	0.60	0.96	0.99	0.92	0.15	0.78
PCA	1.00	0.99	0.49	0.85	0.92	0.85	0.46	0.92	0.48	0.60	0.40	0.95	0.63	0.71	0.74	0.47	0.52	0.81	0.55	0.74	0.64	0.96	0.69	1.00	0.65
gen2out	1.00	1.00	0.39	0.78	0.95	0.90	0.53	0.94	0.63	0.63	0.48	0.93	0.72	0.74	0.70	0.46	0.65	0.78	0.50	0.71	0.71	0.99	0.87	1.00	0.70
GMM	0.76	0.39	0.34	0.67	0.76	0.74	0.62	0.38	0.45	0.69	0.59	0.64	0.67	0.49	0.77	0.57	0.89	0.17	0.53	0.60	0.95	0.91	0.87	0.20	0.80
DeepSVDD	0.73	0.92	0.50	0.63	0.58	0.69	0.46	0.85	0.55	0.45	0.60	0.73	0.59	0.65	0.65	0.51	0.63	0.72	0.50	0.48	0.64	0.70	0.63	0.41	0.48
SOD	0.45	0.96	0.59	0.71	0.66	0.74	0.59	0.93	0.52	0.62	0.57	0.68	0.58	0.59	0.73	0.56	0.89	0.47	0.66	0.57	0.93	0.96	0.81	0.26	0.75
KDE	0.08	0.97	0.33	0.73	0.96	0.87	0.64	0.90	0.41	0.76	0.60	0.81	0.78	0.72	0.84	0.43	0.88	0.77	0.59	0.59	0.89	0.98	0.94	0.99	0.68
CBLOF	0.61	0.23	0.53	0.64	0.45	0.32	0.45	0.35	0.48	0.69	0.55	0.53	0.62	0.39	0.51	0.52	0.59	0.31	0.54	0.55	0.81	0.24	0.37	0.42	0.49
INNE	0.99	0.91	0.40	0.82	0.87	0.82	0.59	0.93	0.55	0.74	0.54	0.89	0.75	0.69	0.81	0.47	0.68	0.82	0.53	0.64	0.89	0.98	0.92	1.00	0.71
AE	1.00	0.98	0.36	0.85	0.93	0.88	0.50	0.93	0.51	0.64	0.37	0.93	0.60	0.67	0.73	0.47	0.55	0.80	0.55	0.74	0.76	0.91	0.65	1.00	0.69
LOF	0.54	0.75	0.47	0.60	0.51	0.64	0.56	0.92	0.49	0.72	0.54	0.53	0.54	0.48	0.55	0.53	0.87	0.76	0.74	0.55	0.93	0.59	0.48	0.37	0.69
IF	1.00	1.00	0.36	0.81	0.95	0.90	0.57	0.94	0.72	0.72	0.48	0.93	0.70	0.73	0.72	0.47	0.64	0.80	0.54	0.77	0.77	0.98	0.82	1.00	0.73
ECOD	0.96	0.99	0.42	0.75	0.91	0.88	0.44	0.90	0.60	0.72	0.37	0.94	0.75	0.70	0.77	0.49	0.57	0.73	0.53	0.65	0.59	0.98	0.79	0.98	0.64
ABOD	0.18	0.99	0.36	0.81	0.77	0.85	0.63	0.93	0.46	0.70	0.58	0.76	0.66	0.61	0.79	0.57	0.82	0.76	0.61	0.59	0.96	0.98	0.91	0.97	0.80
ensemble-LOF	0.63	0.92	0.45	0.63	0.53	0.70	0.55	0.94	0.51	0.72	0.55	0.59	0.58	0.48	0.46	0.55	0.87	0.88	0.75	0.59	0.94	0.71	0.50	0.14	0.70
sb-DeepSVDD	0.64	0.86	0.43	0.60	0.47	0.67	0.52	0.77	0.47	0.47	0.54	0.76	0.52	0.57	0.61	0.50	0.53	0.60	0.51	0.37	0.58	0.76	0.61	0.43	0.45
u-CBLOF	0.85	0.99	0.42	0.82	0.91	0.78	0.44	0.93	0.52	0.71	0.57	0.84	0.77	0.62	0.80	0.47	0.70	0.59	0.54	0.59	0.86	0.99	0.91	1.00	0.70
LUNAR	0.73	0.97	0.36	0.76	0.72	0.71	0.54	0.93	0.44	0.74	0.59	0.60	0.66	0.61	0.67	0.46	0.75	0.69	0.71	0.62	0.86	0.93	0.71	0.16	0.81
HBOS	1.00	0.99	0.36	0.35	0.93	0.91	0.49	0.96	0.87	0.68	0.58	0.80	0.76	0.75	0.78	0.46	0.60	0.91	0.50	0.59	0.66	0.97	0.68	0.97	0.71
VAE	0.80	0.97	0.38	0.84	0.92	0.88	0.54	0.87	0.47	0.66	0.51	0.87	0.63	0.77	0.72	0.47	0.63	0.77	0.55	0.64	0.70	0.92	0.67	1.00	0.67
COF	0.50	0.61	0.47	0.61	0.51	0.52	0.54	0.81	0.43	0.71	0.53	0.50	0.53	0.41	0.50	0.54	0.86	0.41	0.77	0.54	0.90	0.51	0.47	0.12	0.65
LMDD	0.97	1.00	0.36	0.75	0.94	0.89	0.49	0.72	0.59	0.56	0.45	0.71	0.42	0.63	0.73	0.48	0.52	0.86	0.50	0.60	0.63	0.99	0.91	1.00	0.63
ALAD	0.56	0.65	0.50	0.51	0.61	0.53	0.49	0.65	0.70	0.54	0.43	0.71	0.54	0.62	0.67	0.49	0.50	0.81	0.50	0.56	0.59	0.43	0.51	0.91	0.57
beta-VAE	0.86	0.99	0.37	0.85	0.94	0.90	0.49	0.93	0.51	0.65	0.39	0.95	0.60	0.77	0.73	0.47	0.52	0.81	0.55	0.74	0.62	0.96	0.67	1.00	0.67
EIF	1.00	1.00	0.35	0.81	0.95	0.89	0.59	0.95	0.66	0.73	0.50	0.93	0.72	0.73	0.76	0.47	0.65	0.85	0.53	0.78	0.81	0.99	0.90	0.99	0.72

Table 7: The AUC values for the second half of the algorithm-data set combinations.



	PCA	LMDD	SO-GAAL	IF	LOF	EIF	ALAD	kNN	LUNAR	KDE	COF	MCD	GMM	u-CBLOF	INNE	ECOD	beta-VAE	AE	LODA	ODIN	HBOS	SOD	ensemble-LOF	DynamicHBOS	VAE	COPOD	OCSVM	DeepSVDD	gnn2out	ABOD	CBLOF	kth-NN	sb-DeepSVDD
PCA	1.0	0.9	0.188	0.188	0.9	**0.012**	**0.045**	0.155	0.9	0.482	0.835	0.167	0.9	0.9	0.736	0.9	0.9	0.9	0.9	0.616	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.216	0.651	0.747	**0.035**	0.058	**0.001**
LMDD	0.9	1.0	0.581	**0.031**	0.9	**0.001**	0.245	**0.024**	0.9	0.125	0.9	**0.026**	0.9	0.658	0.326	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.888	0.9	0.613	0.9	0.616	0.245	0.338	0.204	**0.007**	**0.015**
SO-GAAL	0.188	0.581	1.0	**0.001**	0.9	**0.001**	**0.032**	**0.001**	0.9	**0.001**	**0.048**	**0.001**	0.9	**0.001**	**0.001**	**0.006**	**0.024**	**0.007**	0.877	0.9	**0.001**	0.238	0.553	**0.001**	0.074	**0.001**	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9
IF	0.188	**0.031**	**0.001**	1.0	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	**0.002**	0.9	**0.001**	0.9	0.9	0.852	0.9	0.852	0.63	0.849	**0.006**	**0.001**	0.9	0.146	**0.036**	0.9	0.401	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**
LOF	0.9	0.9	0.9	**0.001**	1.0	**0.001**	0.9	**0.001**	0.9	**0.002**	0.9	**0.001**	0.9	0.062	**0.011**	0.9	0.9	0.718	0.9	0.722	0.9	0.9	0.33	0.9	0.9	0.171	0.9	**0.049**	0.434	0.9	**0.007**	**0.012**	0.9	**0.001**	0.383
EIF	**0.012**	**0.001**	**0.001**	0.9	**0.001**	1.0	**0.001**	1.0	**0.005**	0.9	**0.001**	0.9	**0.008**	0.062	0.9	0.9	0.9	0.274	0.116	0.271	**0.001**	**0.001**	**0.004**	**0.008**	**0.001**	0.845	**0.04**	0.9	0.574	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
ALAD	**0.045**	0.245	0.9	**0.001**	0.9	**0.001**	1.0	**0.001**	**0.005**	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9	0.883	0.581	0.799	**0.004**	**0.001**	0.56	0.9	0.9	0.221	**0.014**	**0.001**	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9
kNN	0.155	**0.024**	**0.001**	0.9	**0.001**	0.9	**0.001**	1.0	0.525	0.9	**0.001**	0.9	0.442	0.9	0.9	0.883	0.581	0.799	0.9	0.9	**0.004**	**0.001**	0.9	0.9	0.9	0.342	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
LUNAR	0.9	0.9	**0.032**	0.574	0.9	**0.002**	0.9	0.9	0.863	1.0	**0.001**	0.9	0.785	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
KDE	0.482	0.125	**0.001**	0.9	**0.002**	0.9	**0.001**	0.9	0.863	1.0	**0.001**	0.9	0.785	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
COF	0.835	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.451	1.0	**0.001**	1.0	0.532	**0.003**	**0.001**	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
MCD	0.167	**0.026**	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.542	0.9	**0.001**	1.0	0.462	0.9	0.9	0.9	0.9	0.817	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
GMM	0.9	0.9	**0.048**	0.496	0.9	0.062	**0.008**	0.442	0.9	0.785	0.532	0.462	1.0	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
u-CBLOF	0.9	0.658	**0.001**	0.9	0.062	0.9	**0.001**	0.9	0.9	0.9	**0.003**	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
INNE	0.736	0.326	**0.001**	0.9	**0.011**	0.9	**0.001**	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
ECOD	0.9	0.9	**0.006**	0.852	0.718	0.274	**0.001**	0.883	0.9	0.9	0.175	0.821	0.9	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
beta-VAE	0.9	0.9	**0.024**	0.63	0.9	0.116	**0.004**	0.581	0.9	0.9	0.383	0.599	0.9	0.9	0.9	0.9	0.9	1.0	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
AE	0.9	0.9	**0.007**	0.849	0.9	0.271	**0.001**	0.799	0.9	0.9	0.178	0.817	0.9	0.9	0.9	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
LODA	0.9	0.9	0.877	**0.006**	0.9	**0.001**	0.56	**0.004**	0.9	**0.03**	0.9	**0.005**	0.9	0.342	0.105	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
ODIN	0.616	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	0.9	0.9	0.9	**0.001**	**0.001**	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
HBOS	0.9	0.9	**0.001**	0.9	0.33	0.662	**0.001**	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
SOD	0.9	0.9	0.238	0.146	0.9	**0.008**	0.062	0.119	0.9	0.9	0.409	0.898	0.127	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
ensemble-LOF	0.9	0.9	0.553	**0.036**	0.9	**0.001**	0.221	**0.028**	0.9	0.9	0.141	0.9	**0.031**	0.9	0.687	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
DynamicHBOS	0.9	0.888	0.9	**0.001**	0.9	**0.001**	0.062	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
VAE	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
COPOD	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
OCSVM	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
DeepSVDD	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
gnn2out	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
ABOD	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
CBLOF	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
kth-NN	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
sb-DeepSVDD	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9

Table 8: The p-values from Nemenyi post-hoc analysis on all algorithm pairs based on all 52 data sets. P-values below 0.05 have been printed bold.



	PCA	LMDD	SO-GAAL	IF	LOF	EIF	ALAD	kNN	LUNAR	KDE	COF	MCD	GMM	n-CBLOF	INNE	ECOD	beta-VAE	AE	LODA	ODIN	HBOS	SOD	ensemble-LOF	DynamicHBOS	VAE	COPOD	OCSVM	DeepSVDD	gen2out	ABOD	CBLOF	kth-NN	sb-DeepSVDD
PCA	1.0	0.9	0.9	0.129	**0.001**	**0.042**	0.9	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.028**	0.064	0.9	0.9	0.9	0.9	0.407	0.804	**0.002**	**0.001**	0.78	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9
LMDD	0.9	1.0	0.9	0.436	**0.009**	0.193	0.9	**0.001**	**0.009**	**0.003**	**0.019**	**0.004**	**0.004**	**0.004**	0.144	0.263	0.9	0.9	0.9	0.755	0.9	**0.017**	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9
SO-GAAL	0.9	0.9	1.0	0.068	**0.001**	**0.019**	0.9	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.012**	**0.03**	0.9	0.9	0.9	0.9	0.252	0.659	**0.001**	**0.001**	0.635	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9
IF	0.129	0.436	0.068	1.0	0.9	0.9	0.144	0.263	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.755	0.9	0.696	0.335
LOF	**0.001**	**0.009**	**0.001**	0.9	1.0	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.011**	**0.008**	0.463	0.9	0.392	0.9	0.9	0.9	0.9	0.9	0.526	0.31	0.9	0.241	0.129	0.538	0.9	0.263	0.9	**0.006**
EIF	**0.042**	0.193	**0.019**	0.9	0.9	1.0	**0.047**	0.526	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.212	0.175	0.731	0.9	0.9	0.9	0.436	0.106	0.9	0.9	0.9	0.9	0.9	0.864	0.707	0.9	0.9	0.888	0.9	0.136
ALAD	0.9	0.9	0.9	0.144	**0.001**	**0.047**	1.0	**0.001**	**0.001**	**0.001**	**0.003**	**0.001**	**0.001**	**0.032**	0.072	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9
kNN	**0.001**	**0.001**	**0.001**	0.263	0.9	0.526	**0.001**	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
LUNAR	**0.001**	**0.009**	**0.001**	0.9	0.9	0.9	**0.001**	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.01**	**0.008**	0.378	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.005**
KDE	**0.001**	**0.003**	**0.001**	0.9	0.9	0.9	**0.001**	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	**0.003**	**0.003**	**0.003**	0.212	0.9	0.9	0.755	0.9	0.78	0.322	0.159	0.118	0.057	0.335	0.9	0.9	0.9	0.9	0.9	**0.002**
COF	**0.002**	**0.019**	**0.001**	0.9	0.9	0.9	**0.003**	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	**0.022**	**0.017**	0.231	0.526	0.9	0.9	0.9	0.9	0.9	0.647	0.45	0.363	0.212	0.659	0.9	0.9	0.9	0.392	0.9	**0.012**
MCD	**0.001**	**0.004**	**0.001**	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	1.0	0.9	0.9	0.9	**0.004**	**0.003**	0.076	0.241	0.9	0.9	0.792	0.9	0.816	0.363	0.184	0.184	0.068	0.378	0.9	0.9	0.9	0.152	0.9	**0.002**
GMM	**0.001**	**0.004**	**0.001**	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	1.0	0.9	0.9	0.9	**0.004**	**0.003**	0.076	0.241	0.9	0.9	0.792	0.9	0.816	0.363	0.184	0.136	0.068	0.378	0.9	0.9	0.9	0.152	0.9	**0.002**
u-CBLOF	**0.028**	0.144	**0.012**	0.9	0.9	0.9	**0.032**	0.598	0.9	0.9	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.659	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.792	0.9	0.9	0.816	0.9	0.1	
INNE	0.064	0.263	**0.03**	0.9	0.9	0.9	0.072	0.436	0.9	0.9	0.9	0.9	0.9	1.0	0.9	0.286	0.241	0.816	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.852	0.193
ECOD	0.9	0.9	0.9	0.463	**0.011**	0.212	0.9	**0.001**	**0.01**	**0.003**	**0.022**	**0.004**	**0.004**	0.159	0.286	1.0	0.9	0.9	0.9	0.78	0.9	0.9	0.9	0.78	0.322	0.159	0.118	0.057	0.335	0.9	0.9	0.9	0.816	0.9	
beta-VAE	0.9	0.9	0.9	0.407	**0.008**	0.175	0.9	**0.001**	**0.008**	**0.003**	**0.017**	**0.003**	**0.003**	0.129	0.241	0.9	1.0	0.9	0.9	0.731	0.9	0.9	0.9	0.816	0.363	0.184	0.136	0.068	0.378	0.9	0.9	0.9	0.852	**0.001**	0.9
AE	0.9	0.9	0.9	0.144	0.731	0.9	0.9	**0.001**	0.136	0.9	0.231	0.076	0.076	0.659	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.816	0.363	0.184	0.136	0.068	0.378	0.9	0.9	0.9	0.852	**0.001**	0.9
LODA	0.9	0.9	0.9	0.392	0.9	0.9	0.9	**0.001**	**0.001**	**0.001**	**0.003**	**0.003**	**0.003**	0.129	0.241	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9
ODIN	0.407	0.755	0.252	0.9	0.9	0.9	0.436	0.072	0.9	0.9	0.231	0.076	0.076	0.659	0.816	0.9	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9
HBOS	0.804	0.9	0.659	0.9	0.9	0.9	0.828	**0.009**	0.9	0.755	0.9	0.792	0.792	0.9	0.9	0.9	0.9	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.004**	0.9
SOD	**0.002**	**0.017**	**0.001**	0.9	0.9	0.9	**0.002**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.019**	**0.014**	0.212	0.502	0.9	0.9	1.0	0.9	0.9	0.622	0.421	0.335	0.193	0.635	0.9	0.9	0.9	0.436	0.9	0.363
ensemble-LOF	**0.001**	**0.001**	**0.001**	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	**0.001**	**0.032**	0.125	0.9	0.9	0.622	0.9	0.9	0.9	0.9	0.9	**0.028**	0.212	0.9	0.9	0.9	0.106	0.9	0.081
DynamicHBOS	0.78	0.9	0.635	0.9	0.9	0.9	0.804	**0.01**	0.9	0.78	0.9	0.816	0.816	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
VAE	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.514	0.322	0.647	0.363	0.363	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
COPOD	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.298	0.159	0.45	0.184	0.184	0.864	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.004**	0.9
OCSVM	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.231	0.118	0.363	0.136	0.136	0.792	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.003**	0.9
DeepSVDD	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	**0.001**	0.125	0.057	0.212	0.068	0.068	0.635	0.792	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.002**	0.9
gen2out	0.9	0.9	0.9	0.538	0.9	0.9	**0.001**	0.526	0.9	0.335	0.659	0.378	0.378	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.635	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9
ABOD	**0.001**	**0.001**	**0.001**	0.755	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	**0.001**	**0.001**	**0.006**	0.436	0.106	0.9	0.9	0.118	**0.012**	**0.004**	**0.002**	**0.001**	**0.013**	1.0	**0.003**	0.9	**0.001**		
CBLOF	0.9	0.9	0.9	0.9	0.263	0.888	0.9	**0.001**	0.252	0.129	0.392	0.152	0.152	0.816	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.363	0.072	0.9	0.9	0.9	0.9	0.9	0.9	0.9	1.0	**0.003**	1.0	**0.002**	0.9
kth-NN	**0.001**	**0.001**	**0.001**	0.695	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.852	**0.001**	**0.001**	**0.001**	**0.004**	0.363	0.081	0.9	0.9	0.089	**0.009**	**0.003**	**0.002**	**0.001**	**0.009**	0.9	**0.002**	1.0	**0.001**		
sb-DeepSVDD	0.9	0.9	0.9	0.335	**0.006**	0.136	0.9	**0.001**	**0.005**	**0.002**	**0.012**	**0.002**	**0.002**	0.1	0.193	0.9	0.9	0.9	0.9	0.671	0.9	**0.01**	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	1.0	

Table 9: The p-values from Nemenyi post-hoc analysis on all algorithm pairs based on the 17 local data sets. P-values below 0.05 have been printed bold.

	PCA	LMDD	SO-GAAL	IF	LOF	EIF	ALAD	kNN	LUNAR	KDE	COF	MCD	GMM	u-CBLOF	INNE	ECOD	beta-VAE	AE	LODA	ODIN	HBOS	SOD	ensemble-LOF	DynamicHBOS	VAE	COPOD	OCSVM	DeepSVDD	gen2out	ABOD	CBLOF	klh-NN	sb-DeepSVDD
PCA	1.0	0.9	**0.015**	0.9	**0.001**	0.669	**0.001**	0.9	0.731	0.9	**0.001**	0.9	0.599	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.118	**0.005**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
LMDD	0.9	1.0	0.23	0.634	**0.002**	0.126	**0.026**	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.651	0.105	0.9	0.9	0.638	0.9	**0.009**	0.484	0.9	**0.001**	0.9	**0.001**
SO-GAAL	**0.015**	0.23	1.0	**0.001**	0.9	**0.001**	0.9	0.054	0.9	**0.004**	0.9	**0.001**	0.9	**0.03**	**0.001**	**0.001**	**0.002**	0.9	0.9	0.9	**0.005**	0.9	0.9	**0.001**	0.9	0.216	**0.001**	0.9	**0.001**	0.335	0.9	**0.002**	0.9
IF	**0.001**	0.634	**0.001**	1.0	**0.001**	0.9	**0.001**	0.9	**0.004**	0.9	**0.001**	0.9	**0.002**	0.9	0.9	0.9	**0.021**	**0.001**	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	0.651	0.9	**0.001**	0.9	0.524	**0.001**	0.9	**0.001**
LOF	**0.001**	**0.002**	0.9	**0.001**	1.0	**0.001**	0.9	**0.001**	0.519	0.9	**0.001**	0.9	0.651	0.9	0.9	0.9	0.223	0.9	0.9	0.9	0.9	0.85	**0.001**	0.9	**0.001**	**0.002**	**0.001**	0.9	**0.004**	0.9	**0.001**	0.9	0.9
EIF	0.669	0.126	**0.001**	0.9	**0.001**	1.0	**0.001**	0.424	**0.001**	0.898	**0.001**	0.9	**0.001**	0.9	0.9	0.9	**0.001**	0.9	0.669	0.9	0.9	**0.001**	**0.001**	0.9	**0.023**	0.136	0.9	0.9	0.078	**0.001**	0.9	**0.001**	0.9
ALAD	**0.001**	**0.026**	0.9	**0.001**	0.9	**0.001**	1.0	**0.003**	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	**0.023**	**0.001**	**0.001**	0.9	**0.046**	0.9	**0.001**	0.9	**0.001**
kNN	0.9	0.9	0.054	0.9	**0.001**	0.424	**0.003**	1.0	0.9	0.9	**0.001**	0.9	0.836	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.819	**0.001**	0.9	**0.001**
LUNAR	0.731	0.9	0.9	**0.004**	0.519	**0.001**	0.9	0.9	1.0	0.562	1.0	**0.001**	0.9	0.198	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.195	**0.004**	0.9	0.9	0.775	**0.001**	0.9	0.9
KDE	0.9	0.9	**0.004**	0.9	**0.001**	0.898	**0.001**	0.9	0.502	1.0	**0.001**	0.9	0.198	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
COF	**0.001**	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.056	**0.001**	1.0	**0.001**	0.9	0.035	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
MCD	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.198	0.9	**0.001**	1.0	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
GMM	0.599	0.9	0.9	**0.002**	0.651	**0.001**	0.9	0.836	0.9	0.35	0.105	0.116	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
u-CBLOF	0.9	0.9	**0.03**	0.9	**0.001**	0.546	**0.002**	0.9	0.854	0.9	**0.001**	0.9	0.722	1.0	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
INNE	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.312	0.9	**0.001**	0.9	0.195	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
ECOD	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.312	0.9	**0.001**	0.9	0.722	1.0	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
beta-VAE	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.286	0.9	**0.001**	0.9	0.175	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
AE	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.424	0.9	**0.001**	0.9	0.277	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
LODA	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.784	0.9	**0.001**	0.9	0.616	0.9	0.9	0.9	0.9	0.9	0.9	**0.012**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
ODIN	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.387	0.9	**0.001**	0.9	0.51	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
HBOS	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.832	0.9	**0.001**	0.9	0.832	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
SOD	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.832	0.9	**0.001**	0.9	0.832	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
ensemble-LOF	0.118	0.651	0.105	0.9	0.651	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
DynamicHBOS	**0.005**	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
VAE	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
COPOD	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
OCSVM	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
DeepSVDD	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
gen2out	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
ABOD	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
CBLOF	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
klh-NN	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
sb-DeepSVDD	**0.001**	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9

Table 10: The p-values from Nemenyi post-hoc analysis on all algorithm pairs based on the 32 global data sets. P-values below 0.05 have been printed bold.

References

D. Agarwal. Detecting anomalies in cross-classified streams: a Bayesian approach. *Knowledge and Information Systems*, 11(1):29–44, 2007.

M. Amer and M. Goldstein. Nearest-neighbor and clustering based anomaly detection algorithms for Rapidminer. In *RapidMiner Community Meeting and Conference*, pages 1–12, 2012.

J. An and S. Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

A. Arning, R. Agrawal, and P. Raghavan. A linear method for deviation detection in large databases. *Knowledge Discovery and Data Mining*, 1141(50):972–981, 1996.

T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, Y. Zhu, and J. R. Wells. Isolation-based anomaly detection using nearest-neighbor ensembles. *Computational Intelligence*, 34(4):968–998, 2018.

Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(suppl_1):S22–S29, 2001.

L. Birgé and Y. Rozenholc. How many bins should be put in a regular histogram. *ESAIM: Probability and Statistics*, 10:24–45, 2006.

A. Brandsæter, E. Vanem, and I. K. Glad. Efficient on-line anomaly detection for ship systems in operation. *Expert Systems with Applications*, 121:418–437, 2019.

M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. In *ACM SIGMOD International Conference on Management of Data*, pages 93–104, 2000.

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

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui. A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern Recognition*, 74:406–421, 2018.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

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

A. F. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong. Systematic construction of anomaly detection benchmarks from real data. In *ACM SIGKDD workshop on outlier detection and description*, pages 16–21, 2013.

M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

R. Foorthuis. On the nature and types of anomalies: A review of deviations in data. *International Journal of Data Science and Analytics*, 12(4):297–331, 2021.

M. Goldstein and A. Dengel. Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm. In *KI-2012: Poster and Demo Track*, pages 59–63, 2012.

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

A. Goodge, B. Hooi, S.-K. Ng, and W. S. Ng. LUNAR: Unifying local outlier detection methods via graph neural networks. In *AAAI Conference on Artificial Intelligence*, volume 36, pages 6737–6745, 2022.

S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao. ADBench: Anomaly detection benchmark. In *Neural Information Processing Systems*, 2022.

S. Hariri, M. C. Kind, and R. J. Brunner. Extended Isolation Forest. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1479–1489, 2019.

V. Hautamaki, I. Karkkainen, and P. Franti. Outlier detection using k-nearest neighbour graph. In *International Conference on Pattern Recognition*, volume 3, pages 430–433. IEEE, 2004.

R. He and J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *International Conference on World Wide Web*, pages 507–517, 2016.

Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.

R. L. Iman and J. M. Davenport. Approximations of the critical region of the fbietkan statistic. *Communications in Statistics - Theory and Methods*, 9(6):571–595, 1980.

N. Japkowicz, C. Myers, M. Gluck, et al. A novelty detection approach to classification. In *International Joint Conference on Artificial Intelligence*, volume 1, pages 518–523. Citeseer, 1995.

H.-P. Kriegel, M. Schubert, and A. Zimek. Angle-based outlier detection in high-dimensional data. In *ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 444–452, 2008.

H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Outlier detection in axis-parallel subspaces of high dimensional data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 831–838. Springer, 2009.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

L. J. Latecki, A. Lazarevic, and D. Pokrajac. Outlier detection with kernel density functions. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 61–75. Springer, 2007.

M.-C. Lee, S. Shekhar, C. Faloutsos, T. N. Hutson, and L. Iasemidis. gen2Out: Detecting and ranking generalized anomalies. In *IEEE International Conference on Big Data*, pages 801–811. IEEE, 2021.

Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu. COPOD: copula-based outlier detection. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1118–1123. IEEE, 2020.

Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. H. Chen. ECOD: Unsupervised outlier detection using empirical cumulative distribution functions, 2022.

F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

Y. Liu, Z. Li, C. Zhou, Y. Jiang, J. Sun, M. Wang, and X. He. Generative adversarial active learning for unsupervised outlier detection. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1517–1528, 2019.

K. Malik, H. Sadawarti, and K. G S. Comparative analysis of outlier detection techniques. *International Journal of Computer Applications*, 97(8):12–21, 2014.

P. B. Nemenyi. *Distribution-free multiple comparisons.* Princeton University, 1963.

T. Pevnỳ. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102(2): 275–304, 2016.

S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *International Conference on Management of Data*, pages 427–438, 2000.

S. Rayana. ODDS library, 2016. URL `http://odds.cs.stonybrook.edu`.

P. J. Rousseeuw and C. Croux. Alternatives to the median absolute deviation. *Journal of the American Statistical association*, 88(424):1273–1283, 1993.

P. J. Rousseeuw and K. V. Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft. Deep semi-supervised anomaly detection. *arXiv preprint arXiv:1906.02694*, 2019.

L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller. A unifying review of deep and shallow anomaly detection. *Institute of Electrical and Electronics Engineers*, 2021.

B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt, et al. Support vector method for novelty detection. In *Neural Information Processing Systems*, volume 12, pages 582–588, 1999.

E. Schubert and A. Zimek. Elki: A large open-source library for data analysis. *CoRR*, abs/1902.03616, 2019. URL `https://arxiv.org/abs/1902.03616`.

S. Y. Shin and H.-j. Kim. Extended autoencoder for novelty detection with reconstruction along projection pathway. *Applied Sciences*, 10(13):4497, 2020.

M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. A novel anomaly detection scheme based on principal component classifier. Technical report, University of Miami, department of Eletrical and Computer Engineering, 2003.

E. H. Simpson. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 13(2):238–241, 1951.

J. Soenen, E. Van Wolputte, L. Perini, V. Vercruyssen, W. Meert, J. Davis, and H. Blockeel. The effect of hyperparameter tuning on the comparative evaluation of unsupervised anomaly detection methods. In *Knowledge Discovery and Data Mining Workshop on Outlier Detection and Description*, pages 1–9, 2021.

G. Steinbuss and K. Böhm. Benchmarking unsupervised outlier detection with realistic synthetic data. *ACM Transactions on Knowledge Discovery from Data*, 15(4):1–20, 2021.

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

A. Thomas, A. Gramfort, and S. Clémençon. Learning hyperparameters for unsupervised anomaly detection. In *Conférence sur L'apprentissage automatique-Cap 2016*, 2016.

X. Xu, H. Liu, L. Li, and M. Yao. A comparison of outlier detection techniques for high-dimensional data. *International Journal of Computational Intelligence Systems*, 11(1): 652–662, 2018.

H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, and V. Chandrasekhar. Adversarially learned anomaly detection. In *International Conference on Data Mining*, pages 727–736. IEEE, 2018.

Y. Zhao, Z. Nasrullah, and Z. Li. PyOD: A Python toolbox for scalable outlier detection, 2019.

L. Zhou, W. Deng, and X. Wu. Unsupervised anomaly localization using VAE and beta-VAE. *arXiv preprint arXiv:2005.10686*, 2020.

A. Zimek, R. J. Campello, and J. Sander. Ensembles for unsupervised outlier detection: challenges and research questions a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1):11–22, 2014.