The Effect of Hyperparameter Tuning on the Comparative Evaluation of Unsupervised Anomaly Detection Methods

Jonas Soenen*
KU Leuven, Dept. of Computer Science; Leuven.AI
B-3000 Leuven, Belgium
jonas.soenen@kuleuven.be

Elia Van Wolputte*
KU Leuven, Dept. of Computer Science; Leuven.AI
B-3000 Leuven, Belgium
elia.vanwolputte@kuleuven.be

Lorenzo Perini
KU Leuven, Dept. of Computer Science; Leuven.AI
B-3000 Leuven, Belgium
lorenzo.perini@kuleuven.be

Vincent Vercruyssen
KU Leuven, Dept. of Computer Science; Leuven.AI
B-3000 Leuven, Belgium
vincent.vercruyssen@kuleuven.be

Wannes Meert
KU Leuven, Dept. of Computer Science; Leuven.AI
B-3000 Leuven, Belgium
wannes.meert@kuleuven.be

Jesse Davis
KU Leuven, Dept. of Computer Science; Leuven.AI
B-3000 Leuven, Belgium
jesse.davis@kuleuven.be

Hendrik Blockeel
KU Leuven, Dept. of Computer Science; Leuven.AI
B-3000 Leuven, Belgium
hendrik.blockeel@kuleuven.be

ABSTRACT

Anomaly detection aims at finding observations in a dataset that do not conform to expected behavior. Researchers have proposed a large variety of anomaly detection algorithms and their performance is greatly affected by how a user sets each algorithm's hyperparameters. However, the anomaly detection literature does not agree on *how* to set these hyperparameters when experimentally comparing different algorithms. Most papers compare either performance using "default" settings, or maximal performance under optimal settings. In this paper, we argue that both strategies fail to capture what practitioners are actually interested in: how well does the algorithm perform in practice? They are either too pessimistic, assuming no tuning, or unrealistically optimistic, assuming optimal tuning; and they often result in methodologically unsound and irreproducible comparisons between algorithms. We therefore propose to use a small validation set to tune an anomaly detector's hyperparameters on a per dataset basis. We argue this is realistic, striking the balance between keeping the cost of acquiring labeled data low and selecting the hyperparameters in a fair, sound, and reproducible manner. We provide a theoretical lower bound on the validation set size based on probability of an anomaly detector achieving a higher area under the ROC curve than a random detector. Using a benchmark of 16 datasets, we experimentally show that different hyperparameter selection strategies lead to different conclusions about which algorithms perform better than others, and that using a small validation set is a practically feasible and principled way of tuning the hyperparameters for a given dataset.

KEYWORDS

data mining, anomaly detection, outlier detection

ACM Reference Format:
Jonas Soenen, Elia Van Wolputte, Lorenzo Perini, Vincent Vercruyssen, Wannes Meert, Jesse Davis, and Hendrik Blockeel. 2021. The Effect of Hyperparameter Tuning on the Comparative Evaluation of Unsupervised Anomaly Detection Methods. In *ODD '21: 6th Outlier Detection and Description Workshop, Virtual*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

While all unsupervised anomaly detection (AD) algorithms have hyperparameters that can greatly affect performance, deciding how to set them is challenging. The goal is to find hyperparameters that yield good performance. However, measuring performance requires having labeled data, which is assumed not to be available in an unsupervised setting. In fact, the standard motivation for treating anomaly detection as an unsupervised problem is that acquiring labeled data is often difficult, if not infeasible, in practice.

In the literature, researchers tend to cope with this problem in two different ways. The conservative point of view is to perform no tuning and simply use the same hyperparameter configuration on each dataset [2, 6, 8, 9, 11, 13, 16, 18, 19, 23, 25, 27, 30–35, 38–40, 42–47]. This can be thought of as adhering to "reasonable defaults." The opposite point of view is to report results for the hyperparameters that maximize the performance of the detector on each dataset [5, 7, 15, 16, 20, 22–24, 29, 37, 49]. This is akin to assuming

*Both authors contributed equally to the paper

ODD '21, August 15, 2021, Virtual

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

that one has access to an oracle that can always provide the optimal hyperparameters for a dataset.

Unfortunately, both approaches have significant drawbacks. First, neither approach is likely appropriate to answer the standard question asked when empirically evaluating a new anomaly detection method: is the new algorithm useful in practice, i.e., are there datasets on which it outperforms the existing anomaly detection algorithms? On the one hand, using defaults likely provides a pessimistic estimate of performance and disadvantages algorithms whose hyperparameters have a large influence on performance. On the other hand, reporting results that maximize performance literally is "tuning on the test set" which is methodologically unsound. Second, both approaches are likely not indicative of what would happen in practice. Practitioners will likely want to tune the parameters in some way instead of falling back on defaults.

In this paper, we argue that anomaly detection should follow standard ML practice: use a labeled validation set to select appropriate hyperparameters. This offers a principled way to choose reasonable hyperparameter values that enables a fair, reproducible and relevant comparison between algorithms (relevant in the sense that in many practical use cases, it is realistic that the same tuning method can be used). We propose a concrete evaluation strategy, argue for its relevance and conduct an empirical evaluation. First, we demonstrate that existing strategies can lead to substantially different conclusions. Second, as our proposal relies on a validation set which is preferably as small as possible, we also investigate the effect of the size of the validation set on our evaluation strategy. These experiments show that our strategy is feasible in practice, as the majority of AD-methods can be tuned on a relatively small validation set.

2 UNSUPERVISED ANOMALY DETECTION

Viewed from a learning perspective, an unsupervised anomaly detection method is an algorithm that analyses a dataset $D \subset \mathcal{X}$ and returns either a prediction of which instances in D are anomalous (transductive setting), or a function that can predict for any instance in the instance space $\mathcal{X}$ whether it is anomalous (inductive setting). We here make abstraction of whether the method is transductive or inductive; we assume it results in a "model" M, the quality of which is measured by some function q. Often, $q(M)$ is computed by comparing M's predictions with the ground truth on some dataset, which may or may not be equal to D. Most algorithms for anomaly detection have hyperparameters that affect their behavior. We therefore formalize these algorithms as follows: an anomaly detection algorithm A is a function that, given a dataset D and values for the algorithm's hyperparameters θ, returns a model $A(\theta, D)$. Given a quality criterion q, the task of hyperparameter tuning, for a given dataset D, then boils down to: find

$$\theta^* = \arg\max_{\theta} q\left(A\left(\theta, D\right)\right).$$

3 CURRENT METHODOLOGIES

Research on anomaly detection often involves determining which algorithms perform best on certain datasets or under certain conditions. This is typically determined empirically. We discuss two methodologies frequently used in the literature, plus a variation that acts as a reference point in our experiments, and we discuss in what ways they are flawed. We assume a *benchmarking* setup: multiple algorithms A_i are evaluated on b benchmark datasets $\mathcal{B} = \{D_1, \ldots, D_b\}$ using a performance metric q.

Out-of-the-box performance. The most straightforward and popular methodology [2, 6, 8, 9, 11, 13, 16, 18, 19, 23, 25, 27, 30–35, 38–40, 42–47] [1]. is sticking to "default" hyperparameters $\bar{\theta}_j$ as recommended in the literature. Such recommendations come in two forms: as fixed values (e.g. in `iForest` [27], set the number of trees as $t = 100$) or as simple rules of thumb, based on some dataset characteristic (e.g. $\sqrt{|D|}$ bins for `HBOS` [17]). This yields for each algorithm A and dataset D_j the *out-of-the-box performance*

$$q\left(A\left(\bar{\theta}_j, D_j\right)\right).$$

The overall **out-of-the-box performance** of an algorithm A is estimated as the average over all datasets used in the experimental comparison:

$$Q_{out of the box}(A) = \frac{1}{b}\sum_{j=1}^{b} q\left(A\left(\bar{\theta}_j, D_j\right)\right). \tag{1}$$

This method is simple but has multiple drawbacks:

(1) It is relevant if we assume that there is no automatic way of choosing better hyperparameters, and that practitioners will make no effort to find good hyperparameters for a task. This may not be realistic.
(2) If the above assumption is wrong, it tends to underestimate the potential of algorithms whose performance strongly depends on the hyperparameters.
(3) Different papers and different implementations use different "defaults", which still leaves the researcher with having to choose one of them.
(4) It is often not known how these defaults were chosen. If the default for an algorithm A was chosen based on the observation that it works well on some collection of datasets that (unbeknownst to the researcher) happens to overlap with one used in a new investigation, then A is at an advantage, which may lead the researcher to wrong conclusions.
(5) For a newly proposed method, no default exists yet; the question remains how to choose a default. Because of the previous point, it is likely that the new default is chosen based on a different procedure than the other defaults, which jeopardizes the fairness of the comparison between algorithms.

Peak performance. Another popular methodology [5, 7, 15, 16, 20, 22–24, 29, 37, 49] [2] is to use ground truth labels and select the optimal hyperparameters θ_j^* for each problem,

$$\theta_j^* = \arg\max_{\theta} q(A(\theta, D_j)).$$

[1] In all of these references, we observe the use of default hyperparameters. We do not mean to suggest they *exclusively* rely on this methodology. For more details -including excerpts relevant to our categorization- we refer to our full literature review, available at https://github.com/ML-KULeuven/comparative-evaluation-of-anomaly-detection-methods

[2] Same comment as footnote 3, but for optimal hyperparameters instead of defaults.

Averaging this over all datasets gives the **peak performance** of an algorithm:

$$Q_{peak}(A) = \frac{1}{b}\sum_{j=1}^{b} q(A(\theta_j^*, D_j)). \tag{2}$$

This strategy essentially measures the potential of an algorithm. Its drawbacks are:

(1) It is relevant if we assume that practitioners can always select the optimal parameters for a task. This is usually unrealistic.
(2) It overestimates the expected performance of the algorithm when the above assumption is false.

Using this strategy is essentially a version of "tuning on the test set", which is generally considered unsound.

Best-default performance. Unlike the previous strategies, this one is not common in the literature. Rather, it serves as a reference point in our experiments. Instead of selecting the optimal hyperparameters for each problem individually, one can select the hyperparameters that perform best on the whole collection of datasets, on average:

$$\tilde{\theta} = \arg\max_{\theta} \frac{1}{b}\sum_{j=1}^{b} q(A(\theta, D_j))$$

$\tilde{\theta}$ are the "optimal default" hyperparameters for the collection of datasets under consideration. The **best-default performance** of an algorithm is

$$Q_{bestdefault}(A) = \frac{1}{b}\sum_{j=1}^{b} q\left(A\left(\tilde{\theta}, D_j\right)\right). \tag{3}$$

Like out-of-the-box performance, best-default performance assumes no problem-specific tuning is done. The main difference is that it computes a good default for a given collection of datasets, rather than obtaining it from external sources. This makes it unambiguous, and allows treating all algorithms on an equal footing. On the other hand, it is unsound for exactly the same reason as peak performance: it has access to information that would not normally be available to a practitioner. The pessimism of using default hyperparameters and the optimism of "tuning on the test set" partially cancel each other out.

4 A STRATEGY BASED ON VALIDATION SETS

Because it is in their best interest to achieve the best possible performance, our methodology assumes that typical practitioners do an honest effort to find good hyperparameters. In particular, our interpretation of "honest effort" is as follows: the practitioner has access to (or alternatively, spends some time creating) a labeled subset $D^L \subset D$ and uses that information to tune the hyperparameters. In a research setting, given a collection of datasets D_j, this can be simulated by determining

$$\theta_j^\dagger = \arg\max_{\theta} q_{D_j^L}\left(A(\theta, D_j)\right)$$

and the **tuned performance** for an algorithm A is then:

$$Q_{tuned}(A) = \frac{1}{b}\sum_{j=1}^{b} q_{D_j \setminus D_j^L}\left(A\left(\theta_j^\dagger, D_j\right)\right). \tag{4}$$

Note that the quality criterion q is now replaced by two variants $q_{D_j^L}$ and $q_{D_j \setminus D_j^L}$. This reflects the fact that q still measures how close the model's predictions are to the ground truth, but on different subsets of the original dataset: disjoint subsets are used for determining the best hyperparameter values, and for evaluating the model with those values.

In fact, this methodology is nothing new. The use of a labeled validation set to select appropriate hyperparameters is standard practice in supervised ML and is also supported by recent AD toolboxes [26]. Nevertheless, this methodology appears to be largely overlooked in the context of comparative evaluation of AD methods. Unlike the methodologies discussed in Section 3, benchmarking with such tuned performances is accurate, reproducible, and sound. One might ask why such standard practice is not already common in anomaly detection. A possible answer is that it requires a labeled subset of data, which goes against the idea of unsupervised learning. However, in our experience, the assumption that the practitioner is willing to provide a small amount of labeled data, in return for a more accurate anomaly detector, is much more realistic than the assumption that they are not interested in getting better than out-of-the-box performance, or that they can guess the optimal hyperparameter settings necessary to obtain peak performance.

4.1 Validation set size V

One important parameter of this procedure is the size of the validation set. Increasing the size of the validation set increases the likelihood that the model's performance on the validation set is indicative for the performance on the full dataset, because the validation set becomes more representative. On the other hand, the bigger the validation set, the more labels that need to be gathered by the practitioner. This introduces a trade-off: the validation set should be small to limit the labelling effort but not too small as the best performing hyperparameters on a small set might not perform well on the rest of the data.

So, what is the minimum validation set size required to select good hyperparameters? In this paper, we determine this minimum in two ways. First, from a practical point of view, we want to know how many labeled instances are needed in order for this methodology to work. In our experiments, we determine this by investigating the effect of validation set size on our proposed strategy. Second, we also propose a statistical criterion which ensures that, on a given dataset, the validation set size is large enough to distinguish between the performance of a given model and a random classifier. In fact, if the validation set size does not statistically allow inferring that the given model performs differently from random predictions, then comparing different detectors is meaningless. In both cases, we ensure that the validation set contains at least one anomaly and that its contamination (i.e. the fraction of anomalies) is as close as possible to the contamination of the full dataset.

Statistical criterion for the validation set size V. The goal of this criterion is to determine a lower bound on the size of the validation set such that the AUC, i.e., the area under the *receiver operating characteristic* (ROC) curve [3, 14], of a given anomaly detector is statistically different from that of a random predictor. We use AUC because it is a common performance metric in anomaly detection [1,

Table 1: Hyperparameter grid used in our experiments. Random seeds are fixed to ensure reproducibility.

Algorithm	Hyperparameters
LOF and kNN	$k = \{3, 5, 7, \ldots, 299\}$
HBOS	$n\_bins = \{5, 10, 15, \ldots, 100\}$
iForest	$n\_estimators = \{25, 50, 75, \ldots, 300\}$ $max\_samples = \{0.1, 0.2, 0.3, \ldots, 1\}$ $max\_features = \{0.1, 0.2, 0.3, \ldots, 1\}$ $random\_state = 3423452345$
CBLOF	$n\_clusters = \{2, 4, 6, \ldots, 48\}$ $\alpha = \{0.1, 0.2, 0.3, \ldots, 0.9\}$ $\beta = \{2, 4, 6, \ldots, 20\}$ $use\_weights = \{True, False\}$ $random\_state = 123412351$
OCSVM	$kernel =$'rbf' $\nu = \{0.02, 0.04, 0.06, \ldots, 1\}$ $\gamma = \{0.001, 0.005, 0.01, 0.05, \ldots, 5000, 10000\}$

5]. Similarly to [48], we follow three steps to derive this bound. In the first step, we observe that the AUC on a discrete set of examples takes values in $\left\{\frac{i}{m_0 m_1} : 0 \leq i \leq m_0 m_1\right\}$, where m_0 is the number of normal examples (class 0) in the validation set, and m_1 the number of anomalies (class 1). ROC curves can be seen as 2D lattice paths starting from $(0, 0)$ and ending in $(1, 1)$. Every time the threshold used to generate the ROC curve moves, the path takes a direction and makes a step of length $\frac{1}{m_0 m_1}$. The area under any path corresponds to the AUC in our setting. In the second step, we investigate the behavior of the AUC under random paths. According to [41], this area has a normal distribution with mean $\frac{1}{2}$ and variance $\frac{m_0+m_1+1}{12 m_0 m_1}$, to the limit when $m_0, m_1 \to +\infty$. Given that the contamination factor is $\gamma = \frac{m_1}{m_0+m_1}$, and that the size of the validation set is $V = m_0 + m_1$, we can rewrite the variance as $\frac{V+1}{12\gamma(1-\gamma)V^2}$. Finally, when $V \to +\infty$, using $\mathcal{U}$ as the AUC of a random classifier we derive that

$$\mathcal{U} \sim \mathbb{N}\left(\frac{1}{2}, \frac{V+1}{12\gamma(1-\gamma)V^2}\right) \implies \mathbb{P}(\mathcal{U} \geq s) \approx 1 - \Phi\left(\frac{\left(s - \frac{1}{2}\right)V}{\sqrt{\frac{V+1}{12\gamma(1-\gamma)}}}\right), \tag{5}$$

where Φ is the cumulative distribution function (cdf) of the standard normal distribution. Note that $\mathcal{U}$ follows a normal distribution only in the limit [41]. Thus, our approximation may have (small) approximation errors. As a result, given a level of significance p, we choose the size of the validation set V such that $\mathbb{P}(\mathcal{U} \geq s) \leq p$. This means that the AUC achieved by our detector is statistically different from the random prediction with a confidence level of $1 - p$.

In our experiments, for each dataset, we use the average out-of-the-box performance of all algorithms as the AUC value s, the real contamination (i.e., the fraction of anomalies) of the validation set as γ and a significance level p of 0.05, unless explicitly stated otherwise.

5 EXPERIMENTS

We demonstrate the impact of hyperparameter selection on the outcomes of a benchmarking study. In the same context, we investigate the benefits and limitations of our proposal to tune hyperparameters based on a validation set. This yields four research questions:

Q1 Does the methodology for selecting hyperparameters affect the ranking of a benchmarking study?
Q2 How does the methodology for selecting hyperparameters affect the performance of an algorithm?
Q3 Is a small labeled validation set sufficient for identifying a good set of hyperparameters?
Q4 Is an algorithm's performance on the validation set always representative of its performance on the test set?

Table 2: Dataset characteristics of the datasets used in our experiments. The contamination γ is the fraction of anomalous instances.

Dataset name	#attributes m	#instances $\lvert D\rvert$	contamination γ
ALOI	27	49534	3.0%
Annthyroid	21	7129	7.5%
Arrhythmia	259	256	4.7%
Cardiotocography	21	1681	2.0%
InternetAds	1555	1966	18.7%
Ionosphere	32	250	10.0%
KDDCup99	40	48113	0.4%
Lymphography	18	148	4.1%
PageBlocks	10	5393	9.5%
Pima	8	555	9.9%
Shuttle	9	260	5.0%
SpamBase	57	2579	2.0%
Stamps	9	340	9.1%
WBC	9	200	5.0%
WDBC	30	100	10.0%
Waveform	21	3443	2.9%

Datasets. We use a collection of 16 benchmark datasets (Campos et al. [5]) which are often used in the AD-literature. There are multiple versions of each dataset. We use the normalized version, without duplicates. If the contamination (i.e. the fraction of anomalies) exceeds 20%, we use one of the subsampled versions at random (Table 2).

Algorithms. We use six well-known unsupervised anomaly detectors from four different families:

(1) **Density-based:** *local outlier factor* (LOF) [4], *histogram-based outlier detection* (HBOS) [17], and *cluster-based local outlier factor* (CBLOF) [21].
(2) **Proximity-based:** *k-nearest-neighbor-based outlier detection* (kNN) [10].
(3) **Isolation-based:** *isolation forest* (iForest) [28].
(4) **Kernel-based:** *one-class support vector machine* (OCSVM) [36].

We use the implementations available in the PyOD python package [50].

Default hyperparameters. To measure out-of-the-box performance, we need default hyperparameters. As suggested in the literature, we set: $k = max\,(10, 0.03 \cdot |D|)$ for kNN and LOF [12, 33]; fixed-width histograms with $\sqrt{|D|}$ bins for HBOS [17]; number of trees $t = 100$ and number of samples per tree $\phi = 256$ for iForest [27]; $\alpha = 0.90$, $\beta = 5$, and k-means with $k = 10$ as a clustering algorithm for CBLOF [21]. Finally for OCSVM, we use the implementation defaults: rbf kernel with $\nu = 0.5$ and $\gamma = 1/m$, with m the number of features of dataset D.

Optimal & best-default hyperparameters. Optimal and best-default hyperparameters (Section 3) are found via an exhaustive gridsearch (Table 1).

Performance metrics. We report the average area under the *receiver operating characteristic* (ROC) curve (AUC) [3, 14], a canonical choice [1, 5] in anomaly detection, as well as the average rank. To calculate the average rank, for each dataset, we rank the algorithms from best to worst performer according to AUC. The average rank of a method is the average position of the method in each of the rankings.

Experimental setup. For each experiment and dataset, we do the following:

(1) Select a test set, which will be the same for each algorithm.
(2) From the remaining instances select the validation set, its size specified as either a maximum number of instances, or via our statistical criterion (Eq. 5) for a given p-value. The validation set contains at least one anomaly and its contamination is as close as possible to the contamination of the full dataset. Like the test set, the validation set is the same for each algorithm.
(3) When doing hyperparameter selection via our proposal, tune the hyperparameters on the validation set.
(4) Measure the AUC on the test set, regardless of the methodology used to select hyperparameters. This ensures that all reported performance estimates are comparable.
(5) To account for variance in performance estimates due to the random test and validation sets, this procedure is repeated ten times with different test and validation sets. We report the average performance over these ten runs.

5.1 Q1: Does the methodology for selecting hyperparameters affect the ranking of a benchmarking study?

To answer **Q1**, we repeatedly conduct a benchmark study: first based on peak, best-default and out-of-the-box performance, as described in Section 3; then based on tuned performance (our proposal), as explained in Section 4.

Results. Table 3 shows that different hyperparameter selection methodologies lead to different rankings of the algorithms. Based on peak performance, `iForest` performs best, closely followed by `CBLOF` and `OCSVM`. According to best-default performance, the top three should be `iForest`, `CBLOF` and `kNN`. Out-of-the-box performance also indicates these three as top performers, but puts `kNN` ahead of `CBLOF`. Finally, tuned performance also indicates `iForest` and `kNN` as the top performers, followed by `HBOS`. But `CBLOF`, which was among the top performers according to all other methodologies, now drops to second to last in the ranking.

Conclusion. When benchmarking AD algorithms, the hyperparameter selection methodology influences the results to such an extent that the final conclusions of the study are affected. This is due the fact that some algorithms benefit more than others from dataset-specific hyperparameters. In our experiment, `CBLOF` illustrates this nicely: it is the second best performer with optimal hyperparameters, but just the penultimate method with tuned hyperparameters. Similarly, `OCSVM` is among the top performers with optimal hyperparameters, but ranks last in all other evaluation settings.

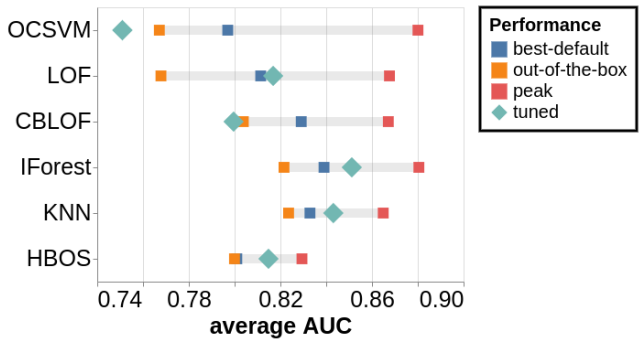


Figure 1: Performance (in AUC) according to each hyperparameter selection methodology, for each algorithm in our study, averaged across all datasets.

5.2 Q2: How does the methodology for selecting hyperparameters affect the performance of an algorithm?

We now know that the ranking produced by a benchmarking study can change, depending on how hyperparameter are selected (**Q1**). Digging deeper, we focus on the impact of hyperparameter selection on the performance of individual algorithms (**Q2**).

Results. Figure 1 shows that some algorithms are more sensitive than others when it comes to hyperparameter selection.

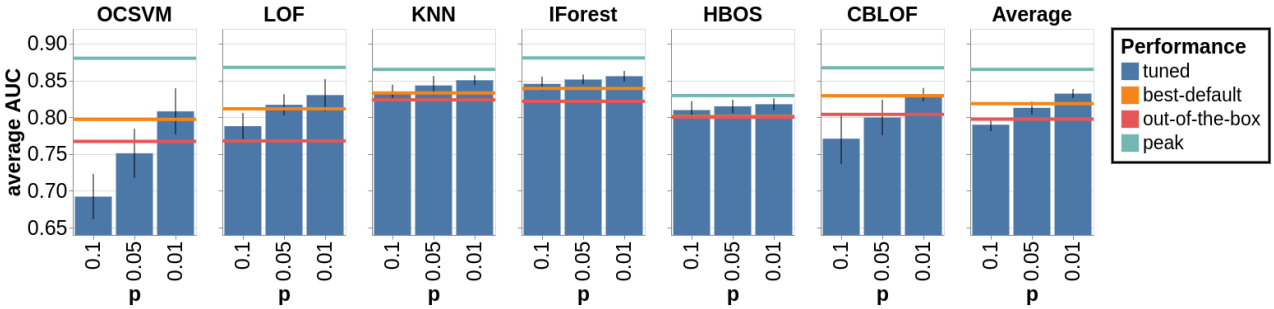
First, the gap between out-of-the-box and peak performance tells us how much an algorithm could *theoretically* benefit from optimal dataset-specific hyperparameters. In general, all algorithms benefit from optimal hyperparameters: on average, the AUC increases with 8.6% when comparing out-of-the-box to peak performance. `OCSVM` and `LOF` benefit the most with AUC improvements of 14.7% and 13.0%, respectively; these algorithms will appear a lot stronger in a a comparative evaluation based on peak performance than they would in one based on out-of-the-box performance. `HBOS` benefits the least from optimal hyperparameters with an improvement of only 3.7%.

Second, the tuned performances tell us how much of those theoretical performance gains can be realized *in practice* with a reasonable amount of tuning. Tuning `LOF`, `iForest`, `kNN`, and `HBOS` yields an improvement over out-of-the-box performance of 6.4%, 3.6%, 2.4%, and 1.9% respectively. For these algorithms, roughly half of the theoretically possible performance gain is realized with a reasonable amount of tuning; on average, the tuned hyperparameters perform similar to or slightly better than the best-default performance. In contrast, for `CBLOF` and `OCSVM` the tuned performance is on-average worse than their out-of-the-box performance.

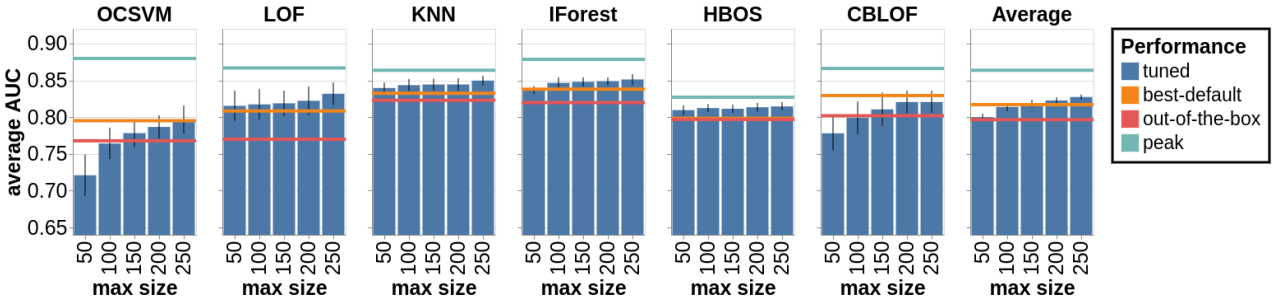
Finally, regardless of how the default hyperparameters are set (fixed values or via rules of thumb), best-default performance always exceeds out-of-the-box performance.

Table 3: Benchmarking several AD-algorithms, based on different methodologies for hyperparameter selection. Size of validation set for tuned performance determined via Eq. 5 with $p = 0.05$. For each algorithm, we report average AUC and average rank (lower rank is better), across the entire benchmark.

Peak (Eq. 2)			Best-default (Eq. 3)			Out-of-the-box (Eq. 1)			Tuned (Eq. 4)		
algorithm	*avg AUC*	*rank*	*algorithm*	*avg AUC*	*rank*	*algorithm*	*avg AUC*	*rank*	*algorithm*	*avg AUC*	*rank*
IForest	0.88	2.72	IForest	0.84	2.75	IForest	0.82	2.66	IForest	0.85	2.44
CBLOF	0.87	3.03	CBLOF	0.83	3.25	KNN	0.82	2.91	KNN	0.84	2.5
OCSVM	0.88	3.28	KNN	0.83	3.34	CBLOF	0.8	3.22	HBOS	0.81	3.56
LOF	0.87	3.5	LOF	0.81	3.59	HBOS	0.8	3.81	LOF	0.82	3.81
KNN	0.86	3.88	HBOS	0.8	3.91	LOF	0.77	4.12	CBLOF	0.8	3.94
HBOS	0.83	4.59	OCSVM	0.8	4.16	OCSVM	0.77	4.28	OCSVM	0.75	4.75



(a) Validation size via Eq. 5 for different p values.



(b) Maximum validation size as an absolute amount of instances.

Figure 2: Tuned performance (averaged over all datasets) for different validation set sizes. Vertical black lines indicate the standard deviation of tuned performance across different runs. Out-of-the-box, best-default and peak performance estimates shown as horizontal lines.

Conclusion. In theory, each algorithm in our study could benefit from optimal hyperparameters for each dataset, as peak performance consistently exceeds out-of-the-box performance. In practice, the performance achieved with reasonable tuning strongly depends on the algorithm. For most (`HBOS`, `iForest`, `kNN` and `LOF`), tuned performance exceeds best-default performance. For others (`CBLOF` and `OCSVM`), tuning is counterproductive, with tuned performance lower than out-of-the-box performance. Furthermore, whether or not tuning helps for a particular algorithm does not depend on potential performance gains: although `OCSVM` and `LOF` both have a large performance gap between their out-of-the-box and peak performance, `LOF` benefits from tuning, but `OCSVM` does not. In Section 5.4, we investigate what makes `CBLOF` and `OCSVM` difficult to tune.

5.3 Q3: Is a small labeled validation set sufficient for identifying a good set of hyperparameters?

In Section 5.2, we show that tuning your hyperparameters on a validation set enables some algorithms to do better than their out-of-the-box performance. Ideally, this validation set is as small as possible, because a smaller validation set requires less labeling effort from the practitioner. So to answer **Q3**, we measure the effect of the size of the validation set on the tuned performance. To do so, we extend the results of Table 3 by computing tuned performance for several other validation set sizes. We determine these validation set sizes in two different manners: First, by using our statistical criterion (Eq. 5, Section 4.1). Second, by using a (maximum) absolute

validation set size n, whilst ensuring that at most 25% of each dataset is used for tuning: i.e., validation set size $V = \min\left(n, \frac{|D|}{4}\right)$. Because the contamination of validation set is kept similar to the contamination of the full dataset (Table 2), the validation set often contains only a few anomalies.

Results. Fig. 2 shows various tuned performances for different validation set sizes. Overall (rightmost panels, Fig. 2), a validation set size of 100 instances, or alternatively, a validation set size determined via our criterion (Eq.5) using $p = 0.05$, is sufficient for tuned performance estimates to consistently exceed the out-of-the-box performances. However, we observe substantial differences between individual algorithms. For HBOS, iForest, LOF and kNN, a small validation set of 50 instances suffices to get a tuned performance similar to their best-default performance. OCSVM and CBLOF need a larger validation set (250 instances) before they reach best-default performance.

Conclusion. On average, in our experiments, hyperparameters acquired by tuning on a validation set of 100 instances (or using our statistical criterion with $p = 0.05$) yield performances that exceed the out-of-the-box performance. Overall, larger validation sets lead to higher tuned performances. As for individual algorithms, some algorithms seem more difficult to tune than others as they need more labelled data before exceeding default-performance. OCSVM and CBLOF are prime examples: they require fairly large validation sets ($p = 0.01$ or 250 instances) to find hyperparameters that perform similar to best-default performance[3].

5.4 Q4: Is an algorithm's performance on the validation set always representative of its performance on the test set?

Using a small validation set to select good hyperparameters assumes that the performance of a detector on the validation set is indicative of its performance on the test set. However, as CBLOF and OCSVM are clearly more difficult to tune than others (**Q2**, **Q3**), it seems that this assumption is not always satisfied. This is what we aim to verify with **Q4**: given a dataset D, is an individuals algorithm's performance on the validation set always representative of its performance on the test set?

Results. Fig. 3 depicts the performance on validation[4] and on test set for the best performing hyperparameters of each algorithm and each dataset. It shows that performance on the validation set is not always indicative of the performance on the test set, and that the size of this effect differs from algorithm to algorithm. For HBOS, kNN, iForest and LOF, the mean absolute difference between the average validation and test set performance over 10 random validation sets is 0.033, 0.041, 0.061 and 0.079 respectively. For these algorithms, the validation set performance matches (reasonably) well with that of the test set. For CBLOF and OCSVM, we observe large discrepancies on some datasets; the mean absolute difference between the average validation and test set performance is 0.109 and 0.164, respectively. This explains why we observed (Sections 5.2, 5.3) that these methods are difficult to tune: hyperparameters that work well on your validation set can completely fail on the test set.

Conclusions. Central to our methodology is the assumption that an algorithm's performance on the validation set provides a reliable estimate of its performance on the test set. For CBLOF and OCSVM, this is clearly not the case, which makes them difficult to tune. Indeed, under realistic conditions (i.e. given a limited amount of labeled data) these methods struggle to realize their full potential (Sections 5.2, 5.3). This shows that the usefulness of a small, labeled validation set is tightly intertwined with the kind of model you are using; e.g. a validation set that is "sufficiently representative" for iForest, can still be inadequate for OCSVM.

6 TAKEAWAYS

We summarize our main observations in five takeaway messages:

(1) When benchmarking anomaly detection algorithms, the final ranking depends on the methodology used to select your hyperparameters (Table 3).
(2) For most algorithms, tuning their hyperparameters on a small validation set yields better performance than using the default hyperparameters (Fig. 1).
(3) None of the algorithms in our study, however, are able to realize their peak performance with a reasonable amount of tuning (Fig. 1).
(4) The potential benefit of tuning does not depend on whether the algorithm has a large gap between out-of-the-box and peak performance, but on whether it is difficult to tune or not (Figs. 2, 3).
(5) A small validation set containing only a few anomalies suffices to achieve those benefits. Concretely, we advise a validation set size of 100 instances, or alternatively, a validation set size determined via our criterion (Eq.5) using $p = 0.05$ (Fig. 2).

Ultimately, we can draw three conclusions. One, it is indeed the case that out-of-the-box performance is an overly pessimistic estimate, whereas peak performance is overly optimistic (points 2-3). Two, our proposed methodology is the only one that yields *realistic* performance estimates, because it takes into account the difficulty of tuning a particular detector on a particular dataset (point 4). Three, our methodology is practically feasible, as a relatively small validation set with a few anomalies is sufficient to tune the hyperparameters (point 5). Moreover, we also derived a theoretical lower bound (Eq. 5) on the validation set size below which it is difficult to distinguish the AUC of a given anomaly detector with that of a random detector.

7 FUTURE WORK

Interesting directions for future work include extending the scope of the current work to include deep AD methods as well as the addition of carefully constructed synthetic datasets to further elucidate what exactly constitutes a good validation set.

[3] Emmott et al. [11] also observed that OCSVM's hyperparameters are hard to optimize.
[4] Validation set size determined via our statistical criterion with $p = 0.05$.

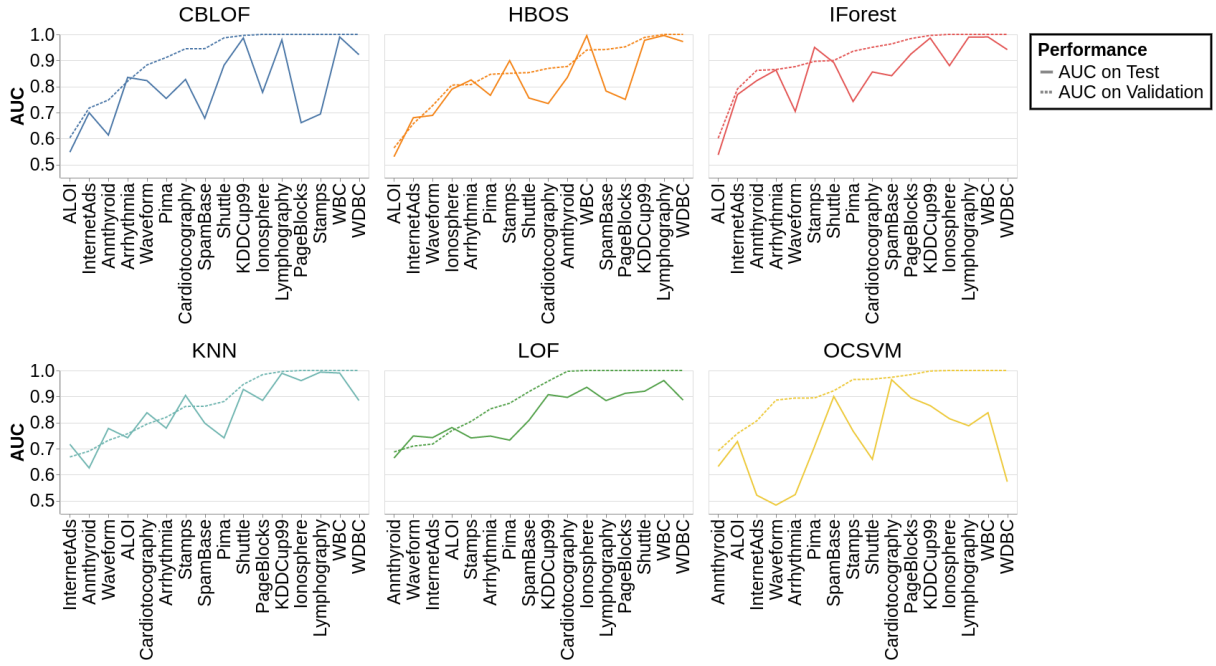


Figure 3: AUC on validation set (dashed line) and on test set (continuous line) for each algorithm and for each dataset. Validation set size via Eq. 5 with $p = 0.05$.

ACKNOWLEDGMENTS

This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme (JS, EVW, LP, VV, WM, JD, HB), the European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme grant agreement No. 694980 "SYNTH: Synthesising Inductive Data Models" (EVW) and KU Leuven research fund C14/17/070 (JD, HB).

REFERENCES

[1] Charu C. Aggarwal. 2017. *Outlier Analysis.* Springer Intl. Publishing AG.
[2] Faruk Ahmed and Aaron Courville. 2020. Detecting Semantic Anomalies. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (Apr. 2020), 3154–3162. https://doi.org/10.1609/aaai.v34i04.5712
[3] Andrew P Bradley. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition* 30, 7 (1997), 1145–1159.
[4] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying Density-based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD Intl. Conference on Management of Data.* 93–104.
[5] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenková, I. Assent E. Schubert, and M. E. Houle. 2016. On the Evaluation of Unsupervised Outlier Detection: Measures, Datasets, and an Empirical Study. *Data Mining and Knowledge Discovery* 30, 4 (2016), 891–927.
[6] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. 2017. Robust, deep and inductive anomaly detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, 36–51.
[7] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. 2018. Anomaly detection using one-class neural networks. *arXiv preprint arXiv:1802.06360* (2018).
[8] Debanjan Datta, M. Raihanul Islam, Nathan Self, Amelia Meadows, John Simeone, Willow Outhwaite, Chen Hin Keong, Amy Smith, Linda Walker, and Naren Ramakrishnan. 2020. Detecting Suspicious Timber Trades. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 08 (Apr. 2020), 13248–13254. https://doi.org/10.1609/aaai.v34i08.7032
[9] Rémi Domingues, Maurizio Filippone, Pietro Michiardi, and Jihane Zouaoui. 2018. A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern Recognition* 74 (2018), 406–421.
[10] Sahibsingh A Dudani. 1976. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-6, 4 (1976), 325–327.
[11] Andrew Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. [n.d.]. A Meta-Analysis of the Anomaly Detection Problem. ([n. d.]). arXiv:1503.01158v2 [cs.AI]
[12] Andrew F Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. 2013. Systematic construction of anomaly detection benchmarks from real data. In *Proceedings of the ACM SIGKDD workshop on outlier detection and description.* 16–21.
[13] Eleazar Eskin. 2000. Anomaly Detection over Noisy Data using Learned Probability Distributions. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, Pat Langley (Ed.). Morgan Kaufmann, 255–262.
[14] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters* 27, 8 (2006), 861–874.
[15] Len Feremans, Vincent Vercruyssen, Boris Cule, Wannes Meert, and Bart Goethals. 2019. Pattern-Based Anomaly Detection in Mixed-Type Time Series. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 11906)*, Ulf Brefeld, Élisa Fromont, Andreas Hotho, Arno J. Knobbe, Marloes H. Maathuis, and Céline Robardet (Eds.). Springer, 240–256. https://doi.org/10.1007/978-3-030-46150-8_15
[16] Izhak Golan and Ran El-Yaniv. 2018. Deep Anomaly Detection Using Geometric Transformations. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/5e62d03aec0d17facfc5355dd90d441c-Paper.pdf
[17] Markus Goldstein and Andreas Dengel. 2012. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: Poster&Demo Track* (2012), 59–63.
[18] Parikshit Gopalan, Vatsal Sharan, and Udi Wieder. 2019. PIDForest: Anomaly Detection via Partial Identification. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/eb6dc8aba23375061b6f07b137617096-Paper.pdf
[19] Eyal Gutflaish, Aryeh Kontorovich, Sivan Sabato, Ofer Biller, and Oded Sofer. 2019. Temporal Anomaly Detection: Calibrating the Surprise. *Proceedings of the*

AAAI Conference on Artificial Intelligence 33, 01 (Jul. 2019), 3755–3762. https://doi.org/10.1609/aaai.v33i01.33013755

[20] Ville Hautamäki, Ismo Kärkkäinen, and Pasi Fränti. 2004. Outlier Detection Using k-Nearest Neighbour Graph. In *17th International Conference on Pattern Recognition, ICPR 2004, Cambridge, UK, August 23-26, 2004*. IEEE Computer Society, 430–433. https://doi.org/10.1109/ICPR.2004.1334558

[21] Zengyou He, Xiaofei Xu, and Shengchun Deng. 2003. Discovering cluster-based local outliers. *Pattern Recognit. Lett.* 24, 9-10 (2003), 1641–1650. https://doi.org/10.1016/S0167-8655(03)00003-5

[22] Ko-jen Hsiao, Kevin Xu, Jeff Calder, and Alfred Hero. 2012. Multi-criteria Anomaly Detection using Pareto Depth Analysis. In *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2012/file/1543843a4723ed2ab08e18053ae6dc5b-Paper.pdf

[23] Tomoharu Iwata and Makoto Yamada. 2016. Multi-view Anomaly Detection via Robust Probabilistic Latent Variable Models. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/0f96613235062963ccde717b18f97592-Paper.pdf

[24] J.H.M. Janssens. 2013. *Outlier selection and one-class classification*. Ph.D. Dissertation. Series: TiCC Ph.D. Series Volume: 27.

[25] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. 2008. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, Ying Li, Bing Liu, and Sunita Sarawagi (Eds.). ACM, 444–452. https://doi.org/10.1145/1401890.1401946

[26] Yuening Li, Daochen Zha, Praveen Venugopal, Na Zou, and Xia Hu. 2020. Pyodds: An end-to-end outlier detection system with automated machine learning. In *Companion Proceedings of the Web Conference 2020*. 153–157.

[27] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 Eighth IEEE Intl. Conference on Data Mining*. IEEE, 413–422.

[28] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 Eighth IEEE Intl. Conference on Data Mining*. IEEE, 413–422.

[29] Yen-Cheng Lu, Feng Chen, Yang Chen, and Chang-Tien Lu. 2013. A Generalized Student-t Based Approach to Mixed-Type Anomaly Detection. *Proceedings of the AAAI Conference on Artificial Intelligence* 27, 1 (Jun. 2013). https://ojs.aaai.org/index.php/AAAI/article/view/8581

[30] Hoang Vu Nguyen, Emmanuel Müller, Jilles Vreeken, Fabian Keller, and Klemens Böhm. 2013. CMI: An information-theoretic contrast measure for enhancing subspace cluster and outlier detection. In *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 198–206.

[31] Guansong Pang, Longbing Cao, Ling Chen, Defu Lian, and Huan Liu. 2018. Sparse Modeling-Based Sequential Ensemble Learning for Effective Outlier Detection in High-Dimensional Numeric Data. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (Apr. 2018). https://ojs.aaai.org/index.php/AAAI/article/view/11692

[32] Heiko Paulheim and Robert Meusel. 2015. A decomposition of the outlier detection problem into a set of supervised learning problems. *Machine Learning* 100, 2-3 (2015), 509–531.

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

[34] Stephen Roberts and Lionel Tarassenko. 1994. A probabilistic resource allocating network for novelty detection. *Neural Computation* 6, 2 (1994), 270–284.

[35] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural computation* 13, 7 (2001), 1443–1471.

[36] Bernhard Schölkopf, Robert C Williamson, Alexander J Smola, John Shawe-Taylor, John C Platt, et al. 1999. Support vector method for novelty detection.. In *NIPS*, Vol. 12. Citeseer, 582–588.

[37] Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. 2014. Generalized Outlier Detection with Flexible Kernel Density Estimates. In *Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014*, Mohammed Javeed Zaki, Zoran Obradovic, Pang-Ning Tan, Arindam Banerjee, Chandrika Kamath, and Srinivasan Parthasarathy (Eds.). SIAM, 542–550. https://doi.org/10.1137/1.9781611973440.63

[38] Xiang-Rong Sheng, De-Chuan Zhan, Su Lu, and Yuan Jiang. 2019. Multi-View Anomaly Detection: Neighborhood in Locality Matters. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 4894–4901. https://doi.org/10.1609/aaai.v33i01.33014894

[39] Georg Steinbuss and Klemens Böhm. 2021. Benchmarking Unsupervised Outlier Detection with Realistic Synthetic Data. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15, 4 (2021), 1–20.

[40] Mahito Sugiyama and Karsten Borgwardt. 2013. Rapid Distance-Based Outlier Detection via Sampling. In *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2013/file/d296c101daa88a51f6ca8cfc1ac79b50-Paper.pdf

[41] Lajos Takács. 1986. Some asymptotic formulas for lattice paths. *Journal of statistical planning and inference* 14, 1 (1986), 123–142.

[42] Holger Trittenbach and Klemens Böhm. 2019. One-Class Active Learning for Outlier Detection with Multiple Subspaces. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 811–820.

[43] Holger Trittenbach, Klemens Böhm, and Ira Assent. 2020. Active Learning of SVDD Hyperparameter Values. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 109–117.

[44] Holger Trittenbach, Adrian Englhardt, and Klemens Böhm. 2020. An overview and a benchmark of active learning for outlier detection with one-class classifiers. *Expert Systems with Applications* (2020), 114372.

[45] Vincent Vercruyssen, Wannes Meert, and Jesse Davis. 2020. Transfer Learning for Anomaly Detection through Localized and Unsupervised Instance Selection. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (Apr. 2020), 6054–6061. https://doi.org/10.1609/aaai.v34i04.6068

[46] Liang Xiong, Barnabás Póczos, and Jeff Schneider. 2011. Group Anomaly Detection using Flexible Genre Models. In *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger (Eds.), Vol. 24. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2011/file/eaae339c4d89fc102edd9dbdb6a28915-Paper.pdf

[47] Xiaodan Xu, Huawen Liu, and Minghai Yao. 2019. Recent progress of anomaly detection. *Complexity* 2019 (2019).

[48] Chris Zeinstra, Raymond Veldhuis, and Luuk Spreeuwers. 2017. How Random is a Classifier given its Area under Curve?. In *2017 International Conference of the Biometrics Special Interest Group (BIOSIG)*. IEEE, 1–4.

[49] Manqi Zhao and Venkatesh Saligrama. 2009. Anomaly detection with score functions based on nearest neighbor graphs. *arXiv preprint arXiv:0910.5461* (2009).

[50] Yue Zhao, Zain Nasrullah, and Zheng Li. 2019. PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research* 20, 96 (2019), 1–7. http://jmlr.org/papers/v20/19-011.html