International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 09 | Sep 2020 www.irjet.net p-ISSN: 2395-0072

Robust and Unsupervised Anomaly Detection for Multivariate Dataset

Ayushi Jhamb[1], Devendra Kumar[2], Himanshu Chauhan[3]

[1-3]*Department of CSE, College of Engineering Roorkee, Roorkee – 247667, Uttarakhand, India*

Abstract: Anomaly detection (also outlier detection [1]) is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data.[1] Typically the anomalous items will translate to some kind of problem such as bank fraud, a structural defect, medical problems or errors in a text. Anomalies are also referred to as outliers, novelties, noise, deviations and exceptions. [2]

In particular, in the context of abuse and network intrusion detection, the interesting objects are often not rare objects, but unexpected bursts in activity. This pattern does not adhere to the common statistical definition of an outlier as a rare object, and many outlier detection methods (in particular unsupervised methods) will fail on such data, unless it has been aggregated appropriately. Instead, a cluster analysis algorithm may be able to detect the micro clusters formed by these patterns. [3]

Three broad categories of anomaly detection techniques exist.[4] Unsupervised anomaly detection techniques detect anomalies in an unlabeled test data set under the assumption that the majority of the instances in the data set are normal by looking for instances that seem to fit least to the remainder of the data set. Supervised anomaly detection techniques require a data set that has been labeled as "normal" and "abnormal" and involves training a classifier (the key difference to many other statistical classification problems is the inherent unbalanced nature of outlier detection). Semi-supervised anomaly detection techniques construct a model representing normal behavior from a given normal training data set, and then test the likelihood of a test instance to be generated by the learnt model.

1. Introduction

In machine learning, the detection of "not-normal" instances within datasets has always been of great interest. This process is commonly known as anomaly detection or outlier detection. The probably first definition was given by Grubbs in 1969 [1]: "An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs". Although this definition is still valid today, the motivation for detecting these outliers is very different now. Back then, the main reason for the detection was to remove the outliers afterwards from the training data since pattern recognition algorithms were quite sensitive to outliers in the data. This procedure is also called data cleansing. After the development of more robust classifiers, the interest in anomaly detection decreased a lot. However, there was a turning point around the year 2000, when researchers started to get more interested in the anomalies itself, since they are often associated with particular interesting events or suspicious data records. Since then, many new algorithms have been developed which are evaluated in this paper. In this context, the definition of Grubbs was also extended such that today anomalies are known to have two important characteristics:

1. Anomalies are different from the norm with respect to their features and

2. They are rare in a dataset compared to normal instances.

Anomaly detection algorithms are now used in many application domains and often enhance traditional rule-based detection systems.

1.1 Categorization of Anomaly Detection

- **Anomaly Detection Setups**

In contrast to the well-known classification setup, where training data is used to train a classifier and test data measures performance afterwards, there are multiple setups possible when talking about anomaly detection. Basically, the anomaly detection setup to be used depends on the labels available in the dataset and we can distinguish between three main types as illustrated in Fig 1:

1) Supervised Anomaly Detection describes the setup where the data comprises of fully labeled training and test data sets. An ordinary classifier can be trained first and applied afterwards. This scenario is very similar to traditional pattern recognition with the exception that classes are typically strongly unbalanced. Not all classification algorithms suit therefore perfectly for this task. For example, decision trees like C4.5 [20] cannot deal well with unbalanced data, whereas Support Vector Machines (SVM) [21] or Artificial Neural Networks (ANN) [22] should perform better. However, this setup is practically not very relevant due to the assumption that anomalies are known and labeled correctly. For many applications, anomalies are not known in advance or may occur spontaneously as novelties during the test phase.

2) Semi-supervised Anomaly Detection also uses training and test datasets, whereas training data only consists of normal data without any anomalies. The basic idea is, that a model of the normal class is learned and anomalies can be detected afterwards by deviating from that model. This idea is also known as "one-class" classification [23]. Well-known algorithms are One-class SVMs [24] and autoencoders [25]. Of course, in general any density estimation method can be used to model the probability density function of the normal classes, such as Gaussian Mixture Models [26] or Kernel Density Estimation [27].
3) Unsupervised Anomaly Detection is the most flexible setup which does not require any labels. Furthermore, there is also no distinction between a training and a test dataset.

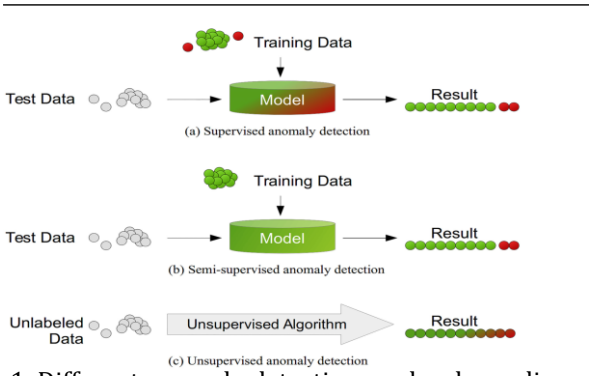

Fig 1. Different anomaly detection modes depending on the availability of labels in the dataset. (a) Supervised anomaly detection uses a fully labeled dataset for training. (b) Semi-supervised anomaly detection uses an anomaly-free training dataset. Afterwards, deviations in the test data from that normal model are used to detect anomalies. (c) Unsupervised anomaly detection algorithms use only intrinsic information of the data in order to detect instances deviating from the majority of the data.

The idea is that an unsupervised anomaly detection algorithm scores the data solely based on intrinsic properties of the dataset. Typically, distances or densities are used to give an estimation what is normal and what is an outlier. This article only focuses on this unsupervised anomaly detection setup.

- **Anomaly Detection Algorithm Output**

As an output of an anomaly detection algorithm, two possibilities exist. First, a label can be used as a result indicating whether an instance is an anomaly or not. Second, a score or confidence value can be a more informative result indicating the degree of abnormality. For super-vised anomaly detection, often a label is used due to available classification algorithms. On the other hand, for semi-supervised and unsupervised anomaly detection algorithms, scores are more common. This is mainly due to the practical reasons, where applications often rank anomalies and only report the top anomalies to the user. In this work, we also use scores as out-put and rank the results such that the ranking can be used for performance evaluation. Of course, a ranking can be converted into a label using an appropriate threshold.

2. Related Work

It could already be inferred from the previous sections that this article primarily deals with multivariate tabular data. Differently structured data, such as graphs or sequential data, is often processed in machine learning using dedicated algorithms. This also holds true in anomaly detection and there exist many algorithms for detecting anomalies in graphs [30], in sequences and time series [31] and for addressing spatial data [32]. However, these specialized algorithms are not evaluated in this work, which focuses on tabular data.

Not many comparative studies on unsupervised anomaly detection do exist today. On the one hand, authors of newly proposed algorithms compare their results with state-of-the-art techniques, for example LOF and k-NN, but often datasets are not published and the evaluation lacks in some other evaluation criteria (local vs. global or parameter k). On the other hand, some studies have been published referring to a specific application scenario, often with a single dataset only. Lazarevic et al. [33] compared LOF, k-NN, PCA and unsupervised SVM for intrusion detection using the KDD-Cup99 dataset. A similar study by Eskin et al. [34] evaluates clustering, k-NN as well as a one-class SVM on the same dataset. A broader study using six different methods for unsupervised anomaly detection was performed by NASA [14] for detecting engine failures of space shuttles. Unfortunately, the dataset is not available and the algorithms used are besides GMM and one-class SVMs four commercially available software systems. Auslander et al. [35] applied k-NN, LOF and clustering on maritime video surveillance data. Ding et al. [36] studied SVDD, a k-NN classifier, k-means and a GMM for detecting anomalies in ten different datasets. Although the authors claim to evaluate unsupervised techniques, the use of a training phase indicates a semi-supervised setup to our understanding. Carrasquilla [37] published a study on comparing different anomaly detection algorithms based on 10 different datasets. Unfortunately, only one unsupervised anomaly detection algorithm was applied, whereas its results were compared to other supervised anomaly detection algorithms. Some of the datasets used in this study are also used as a basis in our evaluation, but with an appropriate preprocessing. All related work concerning the particular algorithms used in this study can be found in the next section.

Besides studies evaluating a single algorithm only, outlier ensembles [38, 39] is a technique of combining multiple unsupervised anomaly detection algorithms in order to boost their joint anomaly detection performance. Since unsupervised anomaly detection does not rely on labeled data, this task is very challenging and often

restricted to simple combinations. In this article we do not evaluate ensembles, although the results reported here might serve as a selection criteria for these algorithms in this currently evolving new research field.

2.1 Unsupervised Anomaly Detection Algorithms

Unsupervised anomaly detection algorithms can be roughly categorized into the following main groups [15] as illustrated in Fig 3: (1) Nearest-neighbor based techniques, (2) Clustering-based methods and (3) Statistical algorithms. Recently, also a new group is emerging based on (4) Subspace techniques. In this work, we cover all of these categories with a focus on nearest-neighbor and clustering-based anomaly detection, the by far most used categories in practice.

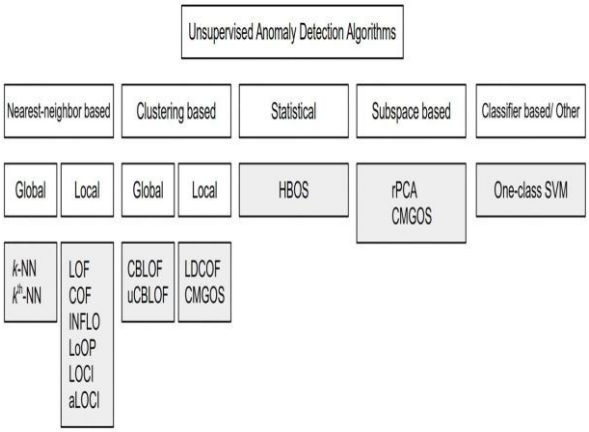

Fig 2. A taxonomy of unsupervised anomaly detection algorithms comprising of four main groups. Note that CMGOS can be categorized in two groups: It is a clustering-based algorithm as well as estimating a subspace of each cluster

Furthermore, other algorithms exist, which are not direct members of these categories, often based on available classification algorithms such as neural networks [25] or support vector machines [40]. It is not an easy task to select a proper subset for this work keeping in mind that dozens of different algorithms have been proposed. However, our selection is based on practical applications in the past and attention in the scientific community. Since one goal of this work is also to standardize datasets, we also welcome other researchers to compare their pro-posed methods with the results presented here. In this section, we shortly introduce the algorithms and their main ideas, but due to the high number of algorithms, the interested reader is referred to the authors original publication for more details.

2.13 One-Class Support Vector Machine

One-class support vector machines [24] are often used for semi-supervised anomaly detection [15]. In this setting, a one-class SVM is trained on anomaly-free data and later, the SVM classifies anomalies and normal data in the test set. One-class SVMs intend to separate the origin from the data instances in the kernel space, which results in some kind of complex hulls describing the normal data in the feature space. Although one-class SVMs are heavily used as a semi-supervised anomaly detection method, it is an unsupervised algorithm by design when using a soft-margin. In particular, it has been shown that they converge to the true density level set [57]. In the unsupervised anomaly detection scenario, the one-class SVM is trained using the dataset and afterwards, each instance in the dataset is scored by a normalized distance to the determined decision boundary [40].

The parameter ν needs to be set to a value lager than zero such that the contained anomalies are correctly handled by a soft-margin. Additionally, one-class SVMs have been modified such that they include further robust techniques for explicitly dealing with outliers during training [40]. The basic idea is that anomalies contribute less to the decision boundary as normal instances. Two different techniques were developed, whereas the η one-class SVM showed superior results. In this enhancement, η is a further optimization objective during training, which estimates the normality of an instance. In our evaluation we are using both unsupervised algorithms, the regular one-class SVM as well as the extended η one-class SVM.

3. Methodology

3.1 Reviewing Algorithm Complexities and Implementation

Concerning the nearest-neighbor based algorithms with the exception of LOCI, the computational complexity of finding the nearest-neighbors is $O(n^2)$. The remaining computations, for example the density or LOF calculations, can be neglected in practice (less than 1% of runtime). Thus, all of these algorithms perform similar in terms of runtime. In our implementation, we used many optimizations so that the algorithms still perform well on large-scale datasets. In particular, first duplicates are removed from the data and a weight matrix is stored. This

procedure might reduce the dataset size and thus save computation time. Memory consumption was reduced by storing only the top-k-neighbors during the search. Additionally, a smart parallelization technique was implemented depending on the number of dimensions. More information about the enhancements can be found in [50]. For LOCI, the computational complexity is $O(n^3)$ and the memory complexity is $O(n^2)$, which makes the algorithm practically too demanding for real-world applications. aLOCI on the contrary is faster and the runtime depends on the number of quad-trees to be used.

Concerning the cluster-based methods (except for CMGOS-MCD), the main computational complexity is due to the clustering algorithm, which is typically faster than $O(n^2)$ if k-means is applied. In practice, when k-means is run several times in order to get stable clustering result, the runtime advance is reduced but still present. For large-scale datasets and big data, clustering-based methods have thus a performance advance compared to nearest-neighbor based methods. However, CMGOS-MCD is an exception here since the MCD computation is again quadratic for each cluster. Furthermore, as already mentioned, HBOS is even faster than the clustering-based algorithms and a good candidate for near real-time large-scale applications.

The complexity of the one-class SVM based algorithms is hard to determine since it depends on the number of support vectors and thus on the structure of the data. Furthermore, the applied gamma tuning of the SVMs has a huge impact on runtime since its computation has a quadratic complexity. Lastly, the complexity of rPCA is $O(d^2 n + d^3)$ and therefore depends heavily on the number of dimensions. If the number of dimensions is small, the algorithm competes in practice among the fastest algorithms in our trials.

3.1 Datasets for Benchmarking

Although unsupervised anomaly detection does not utilize any label information in practice, they are needed for evaluation and comparison. When new algorithms are proposed, it is common practice that an available public classification dataset is modified and the method is compared with the most known algorithms such as k-NN and LOF. There is a set of typically used datasets for classification, which are retrieved from UCI machine learning repository [61]. The typical preprocessing comprises of selecting one class as the anomalous class and sub-sampling some small amount of instances from that randomly. Unfortunately, the resulting datasets are hardly published and cannot be regenerated by other scientists. Since the number of anomalies is typically very low, a different subset might result in very different detection scores. To this end, we only found three different datasets available online [62, 63]. With this work we want to different application domains. A broad spectrum of size, dimensionality and anomaly percentage is covered. They also differ in difficulty and cover local and global anomaly detection tasks.

Table 1. The 17 datasets used for comparative evaluation of the unsupervised anomaly detection algorithms from

Dataset	#points	#di	#outlier
Lympho	148	18	6 (4.1%)
WBC	278	30	21 (5.6%)
Glass	214	9	9 (4.2%)
Vowels	1456	12	50 (3.4%)
Cardio	1831	21	176 (9.6%)
Thyroid	3772	6	93 (2.5%)
Musk	3062	166	97 (3.2%)
Satimage-	5803	36	71 (1.2%)
Letter	1600	32	100
Speech	3686	400	61 (1.65%)
Pima	768	8	268 (35%)
Satellite	6435	36	2036
Shuttle	49097	9	3511 (7%)
BreastW	683	9	239 (35%)
Arrhythm	452	274	66 (15%)
Ionospher	351	33	126 (36%)
Mnist	7603	100	700 (9.2%)
Optdigits	5216	64	150 (3%)
Http	567479	3	2211
ForestCov	286048	10	2747
Mulcross	262144	4	26214
Smtp	95156	3	30 (0.03%)
Mammogr	11183	6	260
Annthyroi	7200	6	534
Pendigits	6870	16	156
Ecoli	336	7	9 (2.6%)

Dataset	#points	#di	#outlier
Wine	129	13	10 (7.7%)
Vertebral	240	6	30 (12.5%)
Yeast	1364	8	64 (4.7%)
Seismic	2584	11	170 (6.5%)
Heart	224	44	10 (4.4%)

Table 1 ODDS dataset overall architecture

Compensate this shortcoming and present and publish more different meaningful datasets, such that algorithms are better comparable in the future. Please note that some of the datasets have already been introduced previously in the first author's Ph.D. Thesis [29] We also put a strong emphasis on a semantic background such the evaluation makes sense. This includes the two assumptions that (1) anomalies are rare and (2) are different from the norm. Additionally, we consider only point anomaly detection tasks as meaningful datasets for benchmarking since a different preprocessing might again lead to non-comparable results. For example, the dataset Poker Hand, which was used for evaluation before [37], is not used because the anomalies (special winning card decks) violate the assumption (2). Similarly, the use of the very popular KDD-Cup99 dataset needs special attention, which was originally used for benchmarking intrusion detection classification systems. Many attacks (anomalies) in the dataset define a collective anomaly detection problem and can thus not be used. Since the dataset is so popular, a point anomaly detection task was extracted as stated below.

If the following, we describe the datasets and our preprocessing in more detail. All modifications have been made publicly available (http://odds.cs.stonybrook.edu/#table1). A summary about the resulting dataset characteristics is given below in Table 1.

3.2 Dataset Summary

All dataset characteristics are summarized in Table 1. With our dataset selection, we cover a broad spectrum of application domains including medical applications, intrusion detection, image and speech recognition as well as the analysis of complex systems. Additionally, the datasets cover a broad range of properties with regard to dataset size, outlier percentage and dimensionality. To our knowledge, this is the most comprehensive collection of unsupervised anomaly detection datasets for algorithm benchmarking. As already stated, we published the datasets to encourage researchers to compare their proposed algorithms with this work and hope to establish an evaluation standard in the community.

4. Result and Discussion

4.1 Comparative Evaluation

Comparing the anomaly detection performance of unsupervised anomaly detection algorithms is not as straight forward as in the classical supervised classification case. In contrast to simply compare an accuracy value or precision/recall, the order of the anomalies should be taken into account. In classification, a wrongly classified instance is for sure a mistake. This is different in unsupervised anomaly detection. For example, if a large dataset contains ten anomalies and they are ranked among the top-15 outliers, this is still a good result, even if it is not perfect. To this end, a common evaluation strategy for unsupervised anomaly detection algorithms is to rank the results according to the anomaly score and then iteratively apply a threshold from the first to the last rank. This results in N tuple values (true positive rate and false positive rate), which form a single receiver operator characteristic (ROC). Then, the area under the curve (AUC), the integral of the ROC, can be used as a detection performance measure. A nice interpretation of the AUC is also given when following the proof from [70] and transform it into the anomaly detection domain: The AUC is then the probability that an anomaly detection algorithm will assign a randomly chosen normal instance a lower score than a randomly chosen anomalous instance. Hence, we think the AUC is a perfect evaluation method and ideal for comparison. However, the AUC only takes the ranking into account and completely neglects the relative difference of the scores among each other. Other measures can be used to cope with this shortcoming by using more sophisticated rank comparisons. Schubert et al. [38] com-pares different rank correlation methodologies, for example Spearman's ρ and Kendall's τ as an alternative to AUC with a focus on targeting outlier ensembles. A second possible drawback of using AUC might be that it is not ideal for unbalanced class problems and methods like area under precision-recall curve or Matthews correlation coefficient could possibly better emphasize small detection performance changes. Nevertheless, AUC based evaluation has been evolved to be the de facto standard in unsupervised anomaly detection, most likely due to its practical interpretability, and thus also serves as the measure of choice in our evaluation.

Please note that the AUC, when it is used in a traditional classification task, typically involves a parameter, for example k, to be altered. In unsupervised anomaly detection, the AUC is computed by varying an outlier threshold in the ordered result list. As a consequence, if a parameter has to be evaluated (for example different k), this yields to multiple AUC values. Another important question in this context is how to evaluate k, the critical parameter for most of the nearest-neighbor and clustering-based algorithms. In most publications, researchers often fix k to a predefined setting or choose "a good k" depending on a favorite outcome. We believe that the latter is not a fair evaluation, because it somehow involves using the test

data (the labels) for training. In our evaluation, we decided to evaluate many different k's between 10 and 50 and finally report the averaged AUC as well as the standard deviation. In particular, for every k, the AUC is computed first by varying a threshold among the anomaly scores as described above. Then, the mean and standard deviation for all these AUCs is reported. This procedure basically corresponds to a random-k-picking strategy within the given interval, which is often used in practice when k is chosen arbitrarily. The lower bound of 10 was chosen because of statistical fluctuations occurring below. For the upper bound, we set a value of 50 such that it is still suitable for our smaller datasets. We are aware, that one could argue to increase the upper bound for the larger datasets or even make it smaller for the small datasets like breast-cancer. Fig 8 shows a broader evaluation of the parameter k illustrating that our lower and upper bound is in fact useful.

4.2 Time Complexity comparison: From the table we easily see that the time complexity is calculated based on 17 datasets in which the OCSVM model is best with respect to time taken by the model.

	Data	#Samples	# Dimensions	Outlier Perc	ABOD	CBLOF	FB	HBOS	IForest	KNN	LOF	MCD	OCSVM	PCA
0	arrhythmia	452	274	14.6018	1.126	0.9786	0.6076	0.8914	0.2507	0.1133	0.0742	1.4578	0.0481	0.0642
0	cardio	1831	21	9.6122	0.3579	0.0983	0.8683	0.005	0.2767	0.1905	0.0993	0.5835	0.0822	0.003
0	glass	214	9	4.2056	0.0361	0.0251	0.0351	0.005	0.1584	0.013	0.003	0.0481	0.001	0.001
0	ionosphere	351	33	35.8974	0.0602	0.0381	0.0642	0.006	0.1785	0.0251	0.007	0.0622	0.004	0.003
0	letter	1600	32	6.25	0.365	0.1052	0.7309	0.007	0.2416	0.1905	0.0922	1.3225	0.0973	0.005
0	lympho	148	18	4.0541	0.0261	0.0321	0.0261	0.005	0.1795	0.011	0.003	0.0531	0.002	0.001
0	mnist	7603	100	9.2069	6.8502	0.9004	44.9604	0.0411	1.6943	6.6327	5.9395	4.1711	4.3626	0.1433
0	musk	3062	166	3.1679	1.8951	0.2998	12.2746	0.0582	1.0528	1.7376	1.5411	21.114	1.1922	0.1504
0	optdigits	5216	64	2.8758	2.243	0.3911	11.9945	0.0271	0.8623	1.9502	1.7256	1.6714	1.4729	0.0521
0	pendigits	6870	16	2.2707	1.3767	0.1955	3.6236	0.008	0.6768	0.8352	0.5886	2.274	0.9646	0.007
0	pima	768	8	34.8958	0.1484	0.0752	0.0982	0.002	0.2176	0.0572	0.011	0.0491	0.01	0.001
0	satellite	6435	36	31.6395	1.6945	0.3288	7.0868	0.017	0.5966	1.1901	0.9465	2.5738	1.3165	0.022
0	satimage-2	5803	36	1.2235	1.3987	0.2396	5.8847	0.015	0.5154	0.9897	0.7229	2.0875	1.0097	0.016
0	shuttle	49097	9	7.1511	13.0948	0.7941	73.3649	0.015	3.2025	9.6808	10.6754	10.3525	44.1575	0.032
0	vertebral	240	6	12.5	0.0431	0.0311	0.029	0.002	0.1575	0.015	0.003	0.0331	0.002	0.001
0	vowels	1456	12	3.4341	0.2336	0.0662	0.2577	0.003	0.2116	0.1113	0.0301	0.755	0.0311	0.002
0	wbc	378	30	5.5556	0.0632	0.0421	0.0792	0.005	0.1564	0.0271	0.007	0.0632	0.005	0.001

Table 2. Time complexity comparison of ODDS dataset

4.3 ROC Performance Comparison: We have perform the ROC performance among various outlier prediction ABOD, CBLOF, FB, HBOS, IForest KNN, LOF, MCD, OCSVM and PCA, For the dataset lympho CBLOF, FB, HBOS, IForest, KNN, LOF, MCD, OCSVM and PCA are results in 100% accuracy and for dataset musk outlier prediction model IForest, MCD, OCSVM and PCA achieved 100 % accuracy. So from table we can conclude that OCSVM is best anomalies / outlier detection model for multivariate dataset.

	Data	#Samples	# Dimensions	Outlier Perc	ABOD	CBLOF	FB	HBOS	IForest	KNN	LOF	MCD	OCSVM	PCA
0	arrhythmia	452	274	14.6018	0.7687	0.7824	0.7796	0.8511	0.8639	0.782	0.7787	0.8228	0.7986	0.7997
0	cardio	1831	21	9.6122	0.5892	0.973	0.6385	0.8373	0.9502	0.734	0.588	0.8195	0.9478	0.9616
0	glass	214	9	4.2056	0.6951	0.7957	0.7073	0.7073	0.7134	0.8384	0.7043	0.8293	0.6585	0.686
0	ionosphere	351	33	35.8974	0.9181	0.795	0.9303	0.6052	0.8486	0.932	0.9227	0.9669	0.8257	0.7941
0	letter	1600	32	6.25	0.8783	0.5301	0.8947	0.6063	0.6201	0.8573	0.8765	0.8061	0.5927	0.5216
0	lympho	148	18	4.0541	0.9831	1	1	1	1	1	1	1	1	1
0	mnist	7603	100	9.2069	0.7628	0.8204	0.7157	0.5766	0.7939	0.8498	0.7195	0.8713	0.854	0.8534
0	musk	3062	166	3.1679	0.2161	0.9899	0.473	0.9999	1	0.8009	0.4629	1	1	1
0	optdigits	5216	64	2.8758	0.4894	0.5329	0.5062	0.8774	0.6735	0.406	0.5277	0.3822	0.5171	0.526
0	pendigits	6870	16	2.2707	0.667	0.9172	0.4889	0.9348	0.9376	0.7371	0.4965	0.8204	0.9235	0.9309
0	pima	768	8	34.8958	0.7163	0.7661	0.6448	0.711	0.6818	0.7395	0.6574	0.7175	0.6561	0.6762
0	satellite	6435	36	31.6395	0.5653	0.5548	0.572	0.7486	0.6825	0.6853	0.572	0.8055	0.6478	0.5923
0	satimage-2	5803	36	1.2235	0.8432	0.9783	0.5235	0.9784	0.9952	0.9515	0.5257	0.9963	0.9997	0.9816
0	shuttle	49097	9	7.1511	0.6171	0.6273	0.4725	0.9871	0.9976	0.6507	0.5556	0.99	0.9934	0.9915
0	vertebral	240	6	12.5	0.5366	0.3937	0.5279	0.3506	0.3772	0.4573	0.4983	0.4103	0.4686	0.4085
0	vowels	1456	12	3.4341	0.9616	0.6496	0.9365	0.6876	0.8209	0.9734	0.9398	0.7243	0.8163	0.6297
0	wbc	378	30	5.5556	0.921	0.8906	0.9271	0.9479	0.9436	0.9444	0.9227	0.9288	0.9358	0.9262

Table 3. ROC performance comparison of ODDS dataset

4.4 Precision Performance Comparison: We have perform the Precision performance among various outlier prediction models ABOD, CBLOF, FB, HBOS, IForest KNN, LOF, MCD, OCSVM and PCA, For the dataset lympho CBLOF, FB, HBOS, IForest, KNN, LOF, MCD, OCSVM and PCA are results in 100% precision and for dataset musk outlier prediction model IForest, MCD, OCSVM and PCA achieved 100 % precision. So from table we can conclude that OCSVM is best anomalies / outlier detection model for multivariate dataset.

	Data	#Samples	# Dimensions	Outlier Perc	ABOD	CBLOF	FB	HBOS	IForest	KNN	LOF	MCD	OCSVM	PCA
0	arrhythmia	452	274	14.6018	0.3571	0.4643	0.4643	0.5714	0.6071	0.5	0.4643	0.4286	0.5	0.5
0	cardio	1831	21	9.6122	0.1918	0.7945	0.1781	0.4521	0.6027	0.3562	0.1507	0.411	0.5342	0.6849
0	glass	214	9	4.2056	0.25	0.25	0.25	0	0.25	0.25	0.25	0	0.25	0.25
0	ionosphere	351	33	35.8974	0.8431	0.549	0.8039	0.3922	0.5882	0.8824	0.7843	0.8627	0.6863	0.5686
0	letter	1600	32	6.25	0.4375	0.0312	0.4062	0.0938	0.0625	0.3125	0.3438	0.1875	0.125	0.125
0	lympho	148	18	4.0541	0	1	1	1	1	1	1	1	1	1
0	mnist	7603	100	9.2069	0.3367	0.3605	0.3741	0.1361	0.2721	0.432	0.3673	0.2653	0.3946	0.3878
0	musk	3062	166	3.1679	0.1	0.65	0.125	0.975	1	0.175	0.125	1	1	1
0	optdigits	5216	64	2.8758	0.0152	0	0.0303	0.2121	0.0303	0	0.0303	0	0	0
0	pendigits	6870	16	2.2707	0.0526	0.1579	0.0526	0.2632	0.3333	0.0702	0.0702	0.0877	0.3158	0.3158
0	pima	768	8	34.8958	0.5253	0.6061	0.4444	0.5354	0.5152	0.5859	0.4646	0.5152	0.5051	0.5354
0	satellite	6435	36	31.6395	0.3962	0.345	0.4	0.57	0.5825	0.4988	0.395	0.6762	0.5225	0.465
0	satimage-2	5803	36	1.2235	0.2333	0.6667	0.1667	0.6	0.8667	0.4333	0.1667	0.6667	0.9	0.7333
0	shuttle	49097	9	7.1511	0.2003	0.2025	0.0257	0.9985	0.9596	0.212	0.1548	0.7395	0.956	0.9516
0	vertebral	240	6	12.5	0.2143	0	0.1429	0	0	0.0714	0.1429	0.0714	0.0714	0
0	vowels	1456	12	3.4341	0.6316	0.1053	0.3684	0.1579	0.1579	0.4737	0.3684	0.1053	0.2632	0.1579
0	wbc	378	30	5.5556	0.375	0.375	0.375	0.5	0.5	0.5	0.375	0.5	0.375	0.375

Table 4. Precision performance comparison of ODDS dataset.

4.5 Computation Time Comparison of Algorithms: If determinable, the theoretical computational complexity of the evaluated algorithms was already discussed. However, in practice, the actual computation times may still be quite differ-ent from each other. For this reason, the computation times were measured and are listed in Table 5. Please note that the listed times are measured in seconds for the first nine datasets and in minutes for the last column, the large kdd99 dataset. The time was measured on a single thread basis. For the clustering-based algorithms, the computation time depends strongly on the number of clusters. For that reason, the times for 10 and 50 clusters are listed separately for each algorithm.

Except for the very demanding LOCI algorithm, it can be seen that the computation for the small datasets is

sufficiently fast, such that the choice of an appropriate algorithm should focus on detection performance, not on runtime. On the contrary, for large datasets, computation time differences are significant. For example, for the largest dataset kdd99, the fastest algorithm HBOS took less than 4 seconds, whereas the slowest GMGOS-MCD took more than 6 days.

In general, it can be observed that nearest-neighbor based algorithms have almost identical runtimes. This is due to the fact, that the nearest-neighbor search is responsible for most of the computation time, whereas the (different) computation of the scores itself has almost no influ-ence. Furthermore, it can be confirmed that the clustering-based algorithms (except for CMGOS-MCD) are faster than the nearest-neighbor based algorithms with the quadratic search complexity. Please keep in mind that the time includes the execution of ten different runs of the underlying k-means algorithm. At this point, we would like to state again, that the use of CMGOS-MCD is not recommended. HBOS is by far the fastest algorithm among all, which is due to its very simple idea of assuming independence of the features. The comparable high runtimes of the SVM based algorithms are mainly based on the automatic gamma tuning, which has a quadratic complexity. For example, for the aloi dataset, the gamma tuning takes about 16 hours, whereas the core SVM training is only 30 seconds for the η one-class SVM and 16 minutes for the regular one-class SVM.

Overall comparison of anomalies detection models have been done using python and dependent libraries, e.g., scikit-learn, we will use Python 3.5 or newer for the latest functions and bug fixes.

Required Dependencies: combo>=0.0.8, numpy>=1.13, numba>=0.35, scipy>=0.19.1, scikit_learn>=0.19.1 and Optional Dependencies keras (optional, required for AutoEncoder), matplotlib (optional, required for running examples) pandas (optional, required for running benchmark),tensorflow (optional, required for AutoEncoder, other backend works), xgboost (optional, required for XGBOD).

Outlier detection often suffers from model instability due to its unsupervised nature. Thus, it is recommended to combine various detector outputs, e.g., by averaging, to improve its robustness. Detector combination is a subfield of outlier ensembles.

Four score combination mechanisms are shown in this demo:

1. **Average**: average scores of all detectors.
2. **Maximization**: maximum score across all detectors.
3. **Average of Maximum (AOM)**: divide base detectors into subgroups and take the maximum score for each subgroup. The final score is the average of all subgroup scores.
4. **Maximum of Average (MOA)**: divide base detectors into subgroups and take the average score for each subgroup. The final score is the maximum of all subgroup score

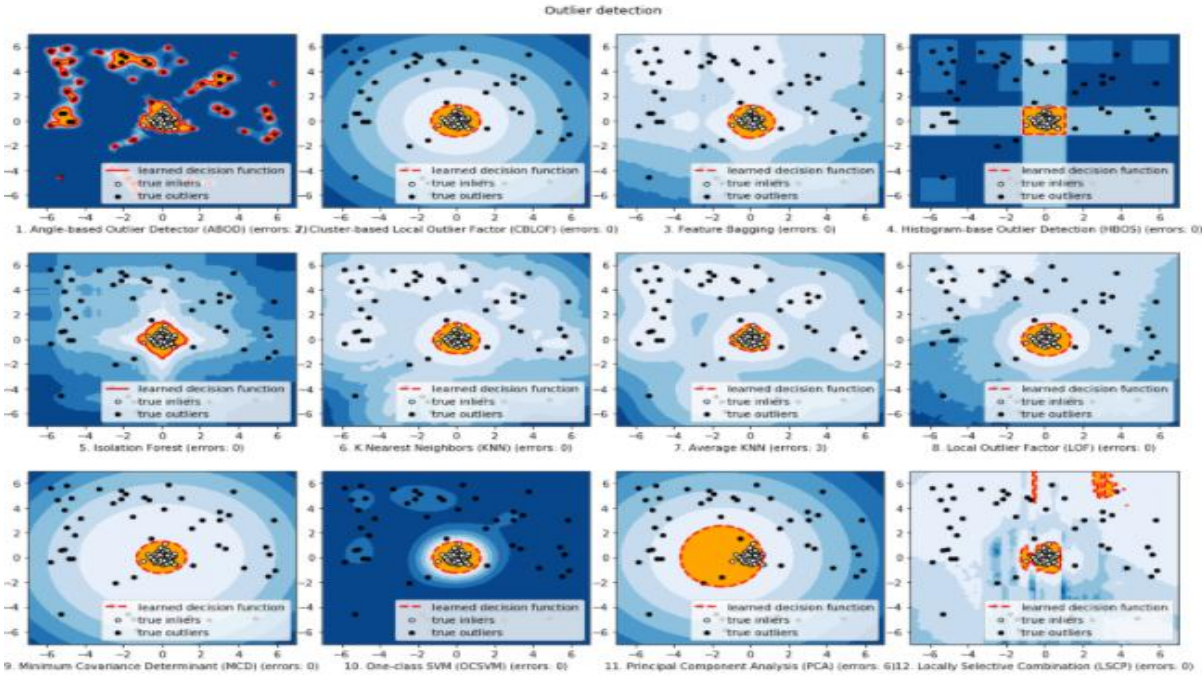

Fig 3. Comparison of all anomalies / outlier detection models

5. Conclusion

A comprehensive evaluation of 19 different unsupervised anomaly detection algorithms on 17 datasets from different application domains has been performed for the first time. The datasets have been made publicly available (http://odds.cs.stonybrook.edu/#table1) and therefore a foundation for a fair and standardized comparison for the community was introduced. Besides supporting the unsupervised anomaly detection research community, we also believe that our study and our implementation is useful for researchers from neighboring fields. Now, it is easy to apply the discussed methods on new data. The broad variety of our evaluation datasets might guide for appropriate algorithm selection in new application domains.

In particular, our findings include that local anomaly detection algorithms, such as LOF, COF, INFLO and LoOP tend to perform poorly on datasets containing global anomalies by generating many false positives. The usage of these algorithms should be avoided if it is known that the task is to detect global anomalies only. On the contrary, global anomaly detection algorithms perform at least average on local problems. This yields in our recommendation to select a global anomaly detection algorithm if there is no further knowledge about the nature of anomalies in the dataset to be analyzed.

As a general detection performance result, we can conclude that nearest-neighbor based algorithms perform better in most cases when compared to clustering algorithms. Also, the stability concerning a not-perfect choice of k is much higher for the nearest-neighbor based meth-ods. The reason for the higher variance in clustering-based algorithms is very likely due to the non-deterministic nature of the underlying k-means clustering algorithm. Despite of this disadvantage, clustering-based algorithms have a lower computation time. As a conclusion, we recommend to prefer nearest-neighbor based algorithms if computation time is not an issue. If a faster computation is required for large datasets, for example in a near real-time setting, clustering-based anomaly detection might be the method of choice. For small datasets, clustering-based methods should be avoided.

Among the nearest-neighbor based methods, the global k-NN algorithm is a good candidate on average. Although LoOP was the best performing nearest-neighbor based algorithm on four datasets, it unfortunately fails significantly one some datasets. Especially for the global anomaly detection problems this algorithm should be totally avoided. Besides our recommendation for k-NN, LOF is also a good candidate if it is previously known that the anomaly detection problem to be solved involves local anomalies.

Concerning the clustering-based algorithms, the simple uCBLOF algorithm also shows on average good performance for all datasets, illustrating that a more sophisticated and compute intense density estimation is not necessarily required. In terms of computational complexity, clustering-based algorithms are faster than their nearest-neighbor competitors. However, in practice, we advice to restart the underlying k-means algorithm multiple times in order to obtain a stable clustering outcome. This procedure unfortunately often takes away the advantage of the theoretical speedup, which leads on the small datasets even to longer runtimes com-pared with the nearest-neighbor based algorithms. Nevertheless, when processing speed is very important or a clustering model can be updated in a data streaming application, a clustering-based algorithm might be used. Besides our recommendation for uCBLOF, CMGOS-Reg also seems to perform reliable on most of the datasets. On the contrary, the original CBLOF algorithm should be avoided due to an algorithm design flaw. Also, the CMGOS with the subspace-based MCD density estimation should not be the first choice, since the density estimation is too slow and detection performance is worse.

The statistical algorithm HBOS, which assumes independence of the features, surprisingly showed very good results in our evaluation. It is even the best performing algorithm on four out of our 10 datasets. Due to the very fast computation time, especially for large datasets, we highly recommend to give it a try on large-scale datasets when looking for global anomalies.

One dataset with 400 dimensions was a big challenge for all of the algorithms, most likely due to the curse of dimensionality. In this context, only nearest-neighbor based algorithms with a very small $k < 5$ were useful at all. Since in unsupervised anomaly detection k can typically not be determined, we might conclude that unsupervised anomaly detection fails on such a high number of dimensions.

As a general summary for algorithm selection, we recommend to use nearest-neighbor based methods, in particular k-NN for global tasks and LOF for local tasks instead of clustering based methods. If computation time is essential, HBOS is a good candidate, especially for larger datasets. A special attention should be paid to the nature of the dataset when applying local algorithms, and if local anomalies are of interest at all in this case. We have summarized our recommendations for algorithm selection in Table 6 with respect to the anomaly detection performance (accuracy), the stability of the scoring (deterministic), the sensitivity to parameters, the computation time for larger datasets (speed) and whether the algorithm is applicable for datasets having global anomalies only. Please note, that the judgments in the table assume that the general recommendations as given above are followed.

References

1. Grubbs FE. Procedures for Detecting Outlying Observations in Samples. Technometrics. 1969; 11 (1):1–21. doi: 10.1080/00401706.1969.10490657
2. Portnoy L, Eskin E, Stolfo S. Intrusion Detection with Unlabeled Data Using Clustering. In: In Proceed-ings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001); 2001. p. 5–8.
3. Garcia-Teodoro P, Diaz-Verdejo JE, Macia-Fernandez G, Vazquez E. Anomaly-based network intru-sion detection: Techniques, systems and challenges. Computers and Security. 2009; 28:18–28. doi: 10.1016/j.cose.2008.08.003
4. Yeung DY, Ding Y. Host-Based Intrusion Detection Using Dynamic and Static Behavioral Models. Pat-tern Recognition. 2003; 36:229–243. doi: 10.1016/S0031-3203(02)00026-2
5. Phua C, Lee V, Smith-Miles K, Gayler R. A Comprehensive Survey of Data Mining-based Fraud Detec-tion Research. Clayton School of Information Technology, Monash University; 2005.
6. Thiprungsri S, Vasarhelyi MA. Cluster Analysis for Anomaly Detection in Accounting Data: An Audit Approach. International Journal of Digital Accounting Research. 2011; 11. doi: 10.4192/1577-8517-v11_4
7. Bolton RJ, Hand DJ. Unsupervised Profiling Methods for Fraud Detection. Statistical Science. 2002; 17 (3):235–255.
8. Sigholm J, Raciti M. Best-Effort Data Leakage Prevention in Inter-Organizational Tactical MANETs. In: Proceedings of IEEE Military Communications Conference (MILCOM 2012). IEEE Computer Society Press; 2012.
9. Lin J, Keogh E, Fu A, Herle HV. Approximations to Magic: Finding unusual Medical Time Series. In: In 18th IEEE Symposium on Computer-Based Medical Systems (CBMS). IEEE Computer Society Press; 2005. p. 23–24.
10. Basharat A, Gritai A, Shah M. Learning Object Motion Patterns for Anomaly Detection and Improved Object Detection. In: Computer Vision and Pattern Recognition. (CVPR 2008). IEEE Conference on. IEEE Computer Society Press; 2008. p. 1–8.
11. Goldstein M, Uchida S. Behavior Analysis Using Unsupervised Anomaly Detection. In: The 10th Joint Workshop on Machine Perception and Robotics (MPR 2014). Online; 2014.
12. Pawling A, Chawla N, Madey G. Anomaly Detection in a Mobile Communication Network. Computa-tional & Mathematical Organization Theory. 2007; 13:407–422. doi: 10.1007/s10588-007-9018-7
13. Gebhardt J, Goldstein M, Shafait F, Dengel A. Document Authentication using Printing Technique Fea-tures and Unsupervised Anomaly Detection. In: Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR 2013). IEEE Computer Society Press; 2013. p. 479–483.
14. Martin RA, Schwabacher M, Oza NC, Srivastava AN. Comparison of Unsupervised Anomaly Detection Methods for Systems Health Management Using Space Shuttle. In: Proceedings of the Joint Army Navy NASA Air Force Conference on Propulsion; 2007.
15. Chandola V, Banerjee A, Kumar V. Anomaly Detection: A Survey. ACM Computing Surveys. 2009; 41 (3):1–58. doi: 10.1145/1541880.1541882
16. Hodge VJ, Austin J. A Survey of Outlier Detection Methodologies. Artificial Intelligence Review. 2004; 22(2):85–126. doi: 10.1023/B:AIRE.0000045502.10941.a9
17. Pimentel MAF, Clifton DA, Clifton L, Tarassenko L. A Review of Novelty Detection. Signal Processing. 2014; 99:215–249. doi: 10.1016/j.sigpro.2013.12.026
18. Markou M, Singh S. Novelty Detection: A Review—Part 1: Statistical Approaches. Signal Processing. 2003; 83(12):2481–2497. doi: 10.1016/j.sigpro.2003.07.018
19. Goldstein M, Asanger S, Reif M, Hutchinson A. Enhancing Security Event Management Systems with Unsupervised Anomaly Detection. In: Proceedings of the 2nd International Conference on Pattern Rec-ognition Applications and Methods (ICPRAM 2013). INSTICC. SciTePress; 2013. p. 530–538.
20. Quinlan JR. C4.5: Programs for Machine Learning. San Francisco, CA, USA: Morgan Kaufmann Pub-lishers Inc.; 1993.
21. Schölkopf B, Smola AJ. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA; 2002.
22. Mehrotra K, Mohan CK, Ranka S. Elements of Artificial Neural Networks. Cambridge, MA, USA: MIT Press; 1997.
23. Moya MM, Hush DR. Network Constraints and Multi-objective Optimization for One-class Classifica-tion. Neural Networks. 1996; 9(3):463–474. doi: 10.1016/0893-6080(95)00120-4
24. Schölkopf B, Platt JC, Shawe-Taylor JC, Smola AJ, Williamson RC. Estimating the Support of a High-Dimensional Distribution. Neural Computation. 2001; 13(7):1443–1471. doi: 10.1162/089976601750264965 PMID: 11440593
25. Hawkins S, He H, Williams GJ, Baxter RA. Outlier Detection Using Replicator Neural Networks. In: Pro-ceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2000). London, UK: Springer-Verlag; 2000. p. 170–180.