Anomalous Anomaly Detection

Muyeed Ahmed Iulian Neamtiu
Department of Computer Science, *New Jersey Institute of Technology*, Newark, NJ, USA
{ma234, ineamtiu}@njit.edu

***Abstract*—Anomaly Detection (AD) is an integral part of AI, with applications ranging widely from health to finance, manufacturing, and computer security. Though AD is popular and various AD algorithm implementations are found in popular toolkits, no attempt has been made to test the reliability of these implementations. More generally, AD verification and validation are lacking. To address this need, we introduce an approach and study on 4 popular AD algorithms as implemented in 3 popular tools, as follows. First, we checked whether implementations can perform their basic task of finding anomalies in datasets with known anomalies. Next, we checked two basic properties, determinism and consistency. Finally, we quantified differences in algorithms' outcome so users can get a idea of variations that can be expected when using different algorithms on the same dataset. We ran our suite of analyses on 73 datasets that contain anomalies. We found that, for certain implementations, validation can fail on 10–73% of datasets. Our analysis has revealed that five implementations suffer from nondeterminism (19–98% of runs are nondeterministic), and 10 out of 12 implementation pairs are inconsistent.**

***Index Terms*—Anomaly Detection, Outlier Detection, Machine Learning, AI testing, AI reliability, Nondeterminism, Verification**

I. Introduction

Anomaly Detection (AD), aka Outlier Detection, is a technique for detecting rare/unexpected items or events that differ from the dataset's normal or expected behavior. AD has been used in a variety of fields: detecting abnormalities in medical data, fraud detection in the financial sector, or detecting faults during manufacturing. Hence there is a pressing need for approaches that can check the reliability of AD implementations. More generally, AD users should expect basic properties such as getting the same result when (a) running the same AD implementation on the same dataset, or (b) running two implementations of the same AD algorithm on the same dataset. However, so far, AD research has focused on performance and performance metrics, rather than reliability. To address these needs, we introduce the first approach and study that investigate AD reliability and expose several major issues. Specifically, we study reliability along several dimensions. We begin with a basic *validation* task: given a dataset with known anomalies, i.e., ground truth, do AD implementations find the anomalies? Next, we perform two analyses: *determinism* (does running an AD implementation repeatedly on the same dataset yield the same result?) and *consistency* (does running two different implementations on the same dataset yield the same result?). Finally, we quantify variations in outcome due to AD algorithms.

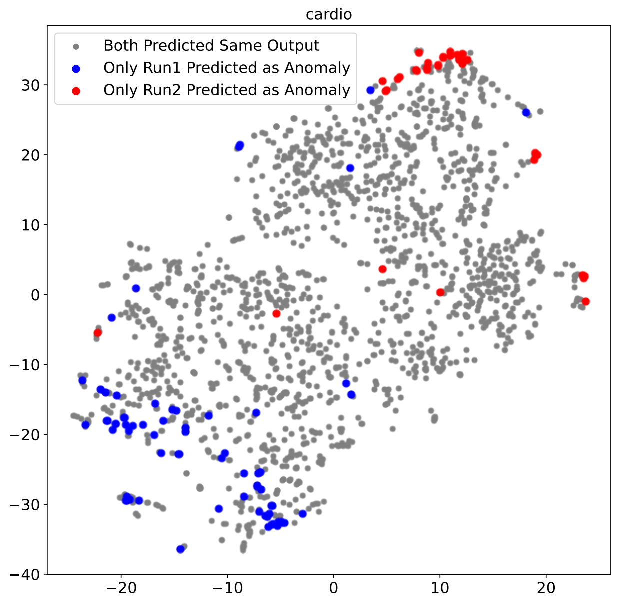


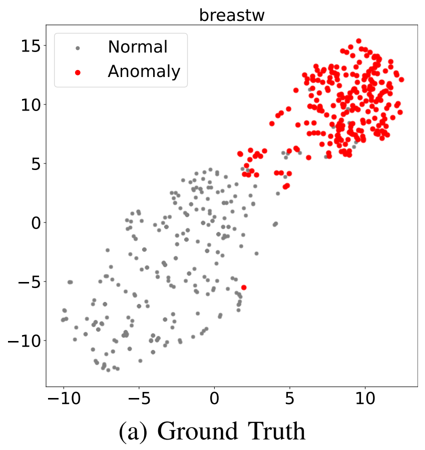
Fig. 1. Nondeterminism: different outputs of two different runs (algorithm: Isolation Forest; toolkit: Scikit-Learn) on dataset cardio.

To motivate our approach, we show how AD implementation nondeterminism and inconsistency can severely affect AD results on two medical datasets.

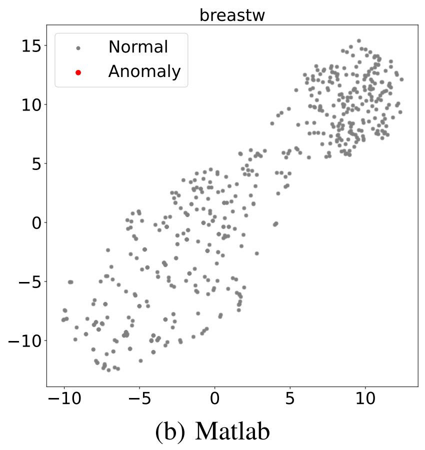
First, consider the dataset cardio, a cardiotocography[1] dataset. This particular dataset has less than 10% data classified as suspect/pathologic (anomaly). We ran Scikit-Learn's Isolation Forest AD algorithm implementation on this dataset, repeatedly, and found that results differ substantially across runs. Figure 1 plots[2] the results of two runs (a single implementation, run twice on the same dataset, with no changes in input settings or parameters). In the figure, in blue we indicate those points identified as anomalous in the first run only; in red, those points identified as anomalous in the second run only; and in gray, those points where the first and the second run concur. Ideally, the runs would concur on 100% of the points, in other words, all points should be gray, because anomalies/outliers should not vary across runs. However, as the figure shows, that is not the case: both red and blue points expose the nondeterminism of the implementation, and

[1]Vitals gathered while monitoring pregnancy and labor. Anomaly status indicates a high probability of having low blood oxygen level or high amount of acid in the body fluids, both requiring immediate action [1], [2].

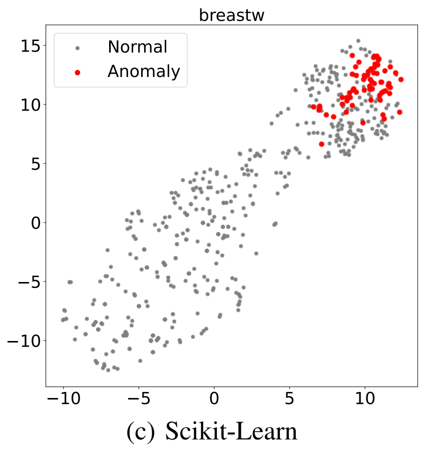
[2]For all figures in this paper that plot datasets, we used t-SNE [3] to reduce dataset dimensionality to 2, for a better visualization.



(a) Ground Truth



(b) Matlab



(c) Scikit-Learn

Fig. 2. Inconsistency example: algorithm Robust Covariance on the breastw dataset.

generally its unreliability.

Second, consider the dataset breastw (breast cancer Wisconsin), whose ground truth anomalies are shown in red in Figure 2 (a). We applied the Robust Covariance AD algorithm on this dataset using two implementations, Matlab and Scikit-learn. The exposed anomalies are shown in Figure 2 (b) and Figure 2 (c). Note that Matlab and Scikit-learn's findings are inconsistent with each other (sets of red points differ) which means two implementations of the same algorithm, run on the same dataset, produce different outcomes. Further, note that Matlab could not detect any anomalies and Scikit-learn's findings differ from ground truth (specifically, find fewer anomalies), which raises validation questions.

Third, we provide a combined, descriptive statistics-based, illustration of both nondeterminism and inconsistency. We ran three different implementations (Matlab, Scikit-learn, R) of the same algorithm, Isolation Forest, on the lymphography oncological dataset. We ran the implementations 30 times and measured the accuracy of different runs against the ground truth. We show the results as boxplots in Figure 3. We can see that the accuracy ranges of these toolkits *have no overlap*, demonstrating inconsistency. For example, the minimum accuracy in Matlab (0.98) was higher than the highest accuracy attained by R (0.86) or Scikit-learn (0.73). Moreover, notice the visible nondeterminism (different results across different runs) in Matlab and Scikit-learn compared to R, which was deterministic.

The rest of the paper is structured as follows. Section II introduces AD, presents the algorithms and toolkits we examined, and discusses our experimental setup in detail. Section III sets the stage via a validation study: five implementations had trouble finding anomalies in datasets that contain confirmed anomalies. In Section IV we focus on nondeterminism: we define nondeterminism in the AD context and present the results of our study that show five implementations exhibit strong nondeterminism. Finally, in Section V we study inconsistency; the results reveal that 10 out of 12 implementation pairs show strong inconsistency.

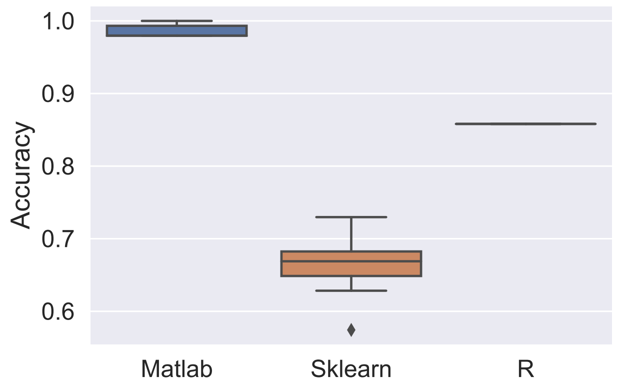


Fig. 3. Isolation Forest accuracy ranges for dataset lymphography.

II. Definitions and Experimental Setup

In this section we first define AD and then describe the setup for our approach (algorithms we explored, the toolkits we investigated, and the datasets we used to conduct our experiment).

A. *Definition and Metrics*

AD is an unsupervised learning technique[3] that, given an unlabeled dataset, aims to find those points whose characteristics or properties differ from most other points. AD algorithms, described in Section II-B, essentially aim to separate "normal" (regular) points from "abnormal" (anomalous) points.

We use several metrics in this paper: accuracy, F1 score, and the Adjusted Rand Index [4] (ARI),[4] for reasons explained shortly. ARI is mainly used to measure similarities of two

[3]While supervised and semi-supervised AD approaches exist, in this paper we focus on the mainstream, unsupervised AD variety.

[4]The ARI of A and B is defined as follows:

$$ARI(A,B)=\frac{2(N_{00}N_{11}-N_{01}N_{10})}{(N_{00}+N_{01})(N_{01}+N_{11})+(N_{00}+N_{10})(N_{10}+N_{11})}$$

where "N_{11} is the number of pairs that are in the same cluster in both A and B; N_{00} is the number of pairs that are in different clusters in both A and B; N_{01} is the number of pairs that are in the same cluster in A but in different clusters in B; and N_{10} is the number of pairs that are in different clusters in A but in the same cluster in B" [5].

clustering results. Assuming two clusterings (partitions) A and B of the same dataset D, the semantics of the ARI between A and B is: $ARI = +1$ indicates perfect agreement, i.e., A and B are identical; $ARI = 0$ corresponds to independent/random clustering, and $ARI = -1$ indicates "perfect disagreement", that is, completely opposite assignment. When A is an AD result and B is ground truth, ARI can be used to measure AD performance. Note that AD does not require labels or training. However when labels (ground truth) do exist, ARI, accuracy, and F1 score can be used to gauge performance.

One of the reasons to use ARI is because two different AD outcomes might have the same accuracy and F1 score. Consider a simple 1-dimensional 4-point dataset where the ground truth is $[0, 0, 0, 1]$, i.e., only the fourth point is an anomaly. Now consider two AD outcomes, $x = [0, 1, 0, 1]$ and $y = [0, 0, 1, 1]$; their accuracy and F1 score are identical, *accuracy = 75%* and *F1 score = 0.67*, but the underlying points identified as anomalies *are different* between x and y. In contrast, when comparing x and y using ARI, the difference is exposed with $ARI = -0.5$.

B. Algorithms

We explore four popular AD algorithms.

Isolation Forest (IF) detects anomalies by isolating objects: IF first selects a random feature and then randomly splits the dataset by selecting a value between the minimum and maximum of that feature [6].

One Class SVM (OCSVM), introduced by Schölkopf et al. [7], uses Support Vector Machines (SVM) to detect novelty objects. SVM maps data into a high-dimensional feature space, then generates a hyperplane to best separate the data into two groups [8]. OCSVM can be viewed as a simple two-class SVM where one class is considered normal and the other class is consider anomalous. Note that one toolkit, Matlab, required a contamination, or outlier fraction, for this algorithm; though the default value was 0, we went with the literature-suggested value of 0.05 (i.e., 5% of the points are outliers).

Local Outlier Factor (LOF) is a density-based AD algorithm proposed by Breunig et al. [9]. LOF uses the distance of an object from its k-nearest neighbors to estimate its local density and then compares with the local densities of its neighbors. An object is considered an anomaly if its density is significantly lower than its neighbors' densities.

Robust Covariance (RobCov) draws a high-dimensional ellipsoid around the "core" of the dataset (encompassing the normal values and leaving out outliers). This algorithm works best on datasets with simple Gaussian distribution [10]. Scikit-learn required a contamination value for this algorithm; we used the default value, 0.1.

C. Toolkits

We studied the implementation of the aforementioned algorithms in three popular toolkits (Table I). **Scikit-learn** (version: 0.21.1), based on Python, has built-in libraries for all 4 algorithms. **Matlab** (version: R2021a) has official libraries for 3 of the 4 algorithms we focused on; for LOF, we used the

TABLE I
TOOLKIT/ALGORITHM CONFIGURATIONS.

Algorithm	MATLAB	R	Scikit-learn
Isolation Forest	✓	✓	✓
LOF	✓	✓	✓
One Class SVM	✓	✓	✓
Robust Covariance	✓		✓

most popular implementation based on GitHub stars [11]. For **R** (version: 4.1.3), we used separate packages for each algorithm [12]–[14]. In the remainder of the paper we use the term "implementation" to denote a toolkit/algorithm combination.

D. Datasets

We used 22 datasets from Outlier Detection DataSets (ODDS) [1]; this is a collection specifically designed to benchmark AD implementations. We also used 51 datasets from OpenML [15]; we selected these datasets by running a skewness analysis to identify those datasets that contained anomalies. The following table summarizes the distribution and characteristics of the 73 datasets.

	Min	Max	Geometric Mean
Instances	63	95,156	817
Features (attributes)	2	166	17.13
# of Anomalies	6	3,511	58.79
Anomaly Ratio	0.03%	35.9%	7.2%

On average, the datasets have 817 instances and 17 features (attributes). The anomaly ratio shows the percentage of a dataset's instances that are anomalies; in our corpus, typically 7.2% of the points in a given dataset are anomalies. As mentioned in Section I, in our figures, we project datasets into 2-D for easier visualization; a drawback of this dimensionality reduction, however, is that points that are outliers in the original space might appear in the dataset's "core" – e.g., center of the figure instead of the expected periphery – in the projection.

III. VALIDATION

The crucial requirement for any AD implementation is to find anomalies in datasets that contain anomalies. However, we found that this requirement was violated in 6 implementations.

TABLE II
OUT OF 73 DATASETS, NUMBER (AND PERCENTAGE) OF DATASETS FOR WHICH AN ALGORITHM FAILS TO DETECT ANY ANOMALIES.

Toolkit	Algorithm	# Datasets	%
Matlab	RobCov	54	73.9
	LOF	19	26.0
Sklearn	LOF	8	10.9
	IF	1	1.4
R	LOF	1	1.4
	OCSVM	2	2.7

We ran each implementation 30 times on the 73 datasets. Table II shows the number (and percentage) of datasets that fail validation, i.e., the implementation failed to find anomalies in at least one run (recall that all 73 datasets contain

TABLE III
NUMBER (PERCENTAGE) OF DATASETS FOR WHICH A IMPLEMENTATIONS SHOW NONDETERMINISM.

Tool	Algorithm	# Datasets	Percentage
Matlab	RobCov	14	19.2%
	IF	71	97.3%
	OCSVM	58	79.5%
Sklearn	RobCov	59	80.8%
	IF	72	98.6%

anomalies). In most cases the 6 implementation that failed validation actually failed to detect anomalies in all 30 runs. There were two exceptions, however: (a) for dataset vertebral, RobCov/Matlab was able to find anomalies in 8 runs out of 30, and (b) for dataset yeast_ml8, IF/Scikit-Learn was able to find anomaly in 7 runs out of 30. As we can clearly see, RobCov on default settings performs very poorly as it fails to detect any anomalies for 73.9% of the datasets. LOF (Matlab and Scikit-Learn) also failed to find anomalies in 10.9% and 26% of datasets, respectively.

The remaining 5 toolkit/algorithm configurations passed validation (as per our definition), i.e., detected *some* anomalies in each of the 30 runs.

IV. NONDETERMINISM

We now turn to verifying whether our studied implementations meet the basic determinism property: running a certain implementation on a certain dataset repeatedly yields the same (deterministic) outcome.

A. Nondeterminism Definition and Test

Measuring Nondeterminism. We ran the implementations 30 times on each dataset. We calculated the accuracy and F1 score of each run w.r.t. ground truth, as well as cross-run ARIs (explained shortly), which expose nondeterminism across runs.

B. Nondeterminism Results

We found that 6 implementations (all three R packages, LOF in Matlab and Scikit-learn, as well as OCSVM in Scikit-learn) are deterministic; however the remaining 5 implementations were strongly nondeterministic. In the remainder of this section we discuss the nondeterminism findings.

Cross-run ARI. For a certain implementation, we count a dataset as nondeterministic if there exist at least 2 runs out of 30 that have mutual (cross-run) $ARI < 1$; in other words, these two runs disagree on which points are outliers and which are not.

Table III shows the count and percentage of datasets for which the 5 implementations showed nondeterminism. IF shows high nondeterminism in both Matlab and Scikit-learn, producing at least one nondeterministic run in $> 97\%$ of datasets. RobCov displayed nondeterminism in both Matlab (19.2%) and Scikit-learn (80.8%).

Table IV shows the widest differences in cross-run ARI for each algorithm. The "1.000" entries are unsurprising ($ARI = 1$ simply means that two runs yield the same outcome). Particularly concerning however, are the entries where

TABLE IV
WIDEST-3 DIFFERENCES IN CROSS-RUN ARI (WITHIN SAME TOOLKIT, I.E., NONDETERMINISM) FOR EACH ALGORITHM.

Algorithm	Toolkit	Dataset	Min	Max	Range
IF	Sklearn	analcatdata_ chlamydia	-0.030	1.000	1.030
	Sklearn	yeast_ml8	-0.001	1.000	1.000
	Matlab	analcatdata_ chlamydia	0.171	1.000	0.829
OCSVM	Matlab	fertility	0.395	1.000	0.605
	Matlab	analcatdata_ chlamydia	0.553	1.000	0.447
	Matlab	mammography	0.556	1.000	0.444
RobCov	Matlab	vertebral	0.000	1.000	1.000
	Sklearn	backache	0.109	0.853	0.744
	Sklearn	mnist	0.048	0.872	0.824

ARI is close to 0 or even negative, because $ARI = 0$ indicates unrelated outcomes and $ARI < 0$ indicates disagreeing outcomes (Section II-A).

TABLE V
TOP-5 WIDEST DIFFERENCE IN ARI VS GROUND TRUTH ACROSS RUNS.

Toolkit	Algo.	Dataset	Min	Max	Range
Sklearn	IF	wine	-0.073	0.361	0.434
Sklearn	IF	musk	0.223	0.600	0.377
Sklearn	RobCov	mnist	0.061	0.403	0.343
Sklearn	RobCov	ar3	0.258	0.558	0.299
Sklearn	RobCov	ar1	0.001	0.281	0.280

ARI vs ground truth. While the cross-run ARI shows the deviations in *outcome* from run-to-run, we also need to understand the run-to-run deviations when compared to ground truth. For that, we use ARI to compare the toolkit outcome (AD result) with the ground truth. Table V shows the 5-widest differences in ARI vs ground truth across runs. We make two observations. First, note the negative ARI values in the table, e.g., for backache we have $ARI = -0.021$ which suggests the AD outcome is worse than unrelated to ground truth and tilts toward opposite assignment. Second, note the wide range, e.g., for musk, the ARI can vary from 0.223 to 0.6, i.e., wide swings in AD outcome across runs.

Accuracy and F1 score. We now quantify nondeterminism in terms of accuracy. Table VII shows the top-5 widest accuracy differences across runs. For example, when we ran RobCov on the dataset vertebral using Matlab, for one run the accuracy was 0.875 whereas a subsequent run, on the same dataset and with the same settings, achieved an accuracy of only 0.633. We also measured F1 scores and found significant difference between runs. Table VI shows top-5 widest F1 score differences across different runs. This table also shows the minimum and maximum precision and recall of different runs. For example, IF/Scikit-learn performed very differently in 2 different runs on dataset wine: in one run it failed to detect any anomalies, hence the F1 score, precision and recall are all 0, whereas in another run the F1 score was 0.483 (with a 0.368 precision and 0.7 recall).

Summary. These findings are concerning, as they indicate that users cannot rely on the results: even when users are aware

TABLE VI
TOP-5 WIDEST F1 SCORE DIFFERENCE ACROSS RUNS.

Tool	Algo.	Dataset	F1			Precision		Recall	
			Min	Max	Range	Min	Max	Min	Max
Sklearn	IF	wine	0.00	0.48	0.48	0.00	0.37	0.00	0.70
Sklearn	RC	mnist	0.16	0.51	0.35	0.16	0.49	0.17	0.53
Sklearn	IF	musk	0.31	0.64	0.33	0.18	0.47	1.00	1.00
Sklearn	RC	ar1	0.09	0.38	0.29	0.08	0.33	0.11	0.44
Sklearn	RC	backache	0.09	0.37	0.28	0.11	0.44	0.08	0.32

TABLE VII
TOP-5 WIDEST ACCURACY DIFFERENCE ACROSS RUNS.

Tool	Algo.	Dataset	Min	Max	Range
Matlab	RC	vertebral	0.633	0.875	0.242
Sklearn	IF	lympho	0.574	0.730	0.155
Sklearn	IF	analcatdata_ neavote	0.610	0.760	0.150
Sklearn	IF	visualizing_ livestock	0.223	0.362	0.138
Sklearn	IF	climate-model-sim-crashes	0.274	0.400	0.126

that results might differ across runs so multiple runs should be considered, it is unclear which run to pick for best AD outcomes.

V. INCONSISTENCY

In this section we study whether our examined implementations meet the basic *consistency* property: running two different implementation of the same algorithm, on the same dataset, yields the same outcome. Users reasonably expect implementations of a certain algorithm to be interchangeable, in other words, the outcome of a certain AD algorithm on a dataset should not depend upon the implementation.

A. Inconsistency Definition and Test

To test inconsistency for a certain algorithm and a certain dataset, we computed the mutual ARI of every run of one implementation against every run of a different implementation; that is, $\binom{30}{2} = 435$ pairs of runs. Among these, we look for the worst-case scenario, i.e., lowest ARI, which indicates that in practice two users that use two toolkits on the same dataset might obtain widely different results.

B. Inconsistency Results

Table VIII shows the 2 lowest consistencies across toolkits for different algorithms. From the table we can see that for each algorithm the toolkits perform very different from each other. In most cases we have $ARI < 0$.

Table IX shows the average mutual ARI for all the datasets on different toolkits for each algorithm. From the table we can clearly see that for all toolkit combinations, the algorithms are generally inconsistent. For all the cases the average is below 0.5 and for all the combinations, more than 60% of the datasets, the mutual ARI values were below 0.5. Results were consistent in just two cases: for 8 datasets we got consistent result when we ran LOF using Matlab and Scikit-learn; for one dataset, we got consistent result when we ran OCSVM using Matlab and Scikit-learn.

We illustrate inconsistency in Figure 4. The figure shows three different algorithms' performance, on three different

TABLE VIII
LOWEST-2 CONSISTENCIES ACROSS TOOLKITS FOR DIFFERENT ALGORITHMS.

Algorithm	Toolkits	Dataset	ARI
IF	Sklearn vs. Matlab	analcatdata_ apnea2	-0.060
		analcatdata_ apnea3	-0.060
	Sklearn vs. R	analcatdata_ apnea3	-0.098
		analcatdata_ apnea2	-0.094
	Matlab vs. R	analcatdata_ challenger	-0.014
		yeast_ml8	-0.002
OCSVM	Sklearn vs. Matlab	analcatdata_ neavote	-0.053
		arsenic-male-bladder	-0.003
	Sklearn vs. R	mnist	0
		oil_spill	0
	Matlab vs. R	optdigits	-0.003
		fertility	-0.001
RobCov	Sklearn vs. Matlab	breastw	0
		glass	0
LOF	Sklearn vs. Matlab	mnist	0
		optdigits	0
	Sklearn vs. R	appendicitis	-0.091
		wbc	-0.079
	Matlab vs. R	glass	-0.076
		ar3	-0.068

TABLE IX
AVERAGE MUTUAL ARI ACROSS DIFFERENT TOOLKITS FOR EACH ALGORITHM.

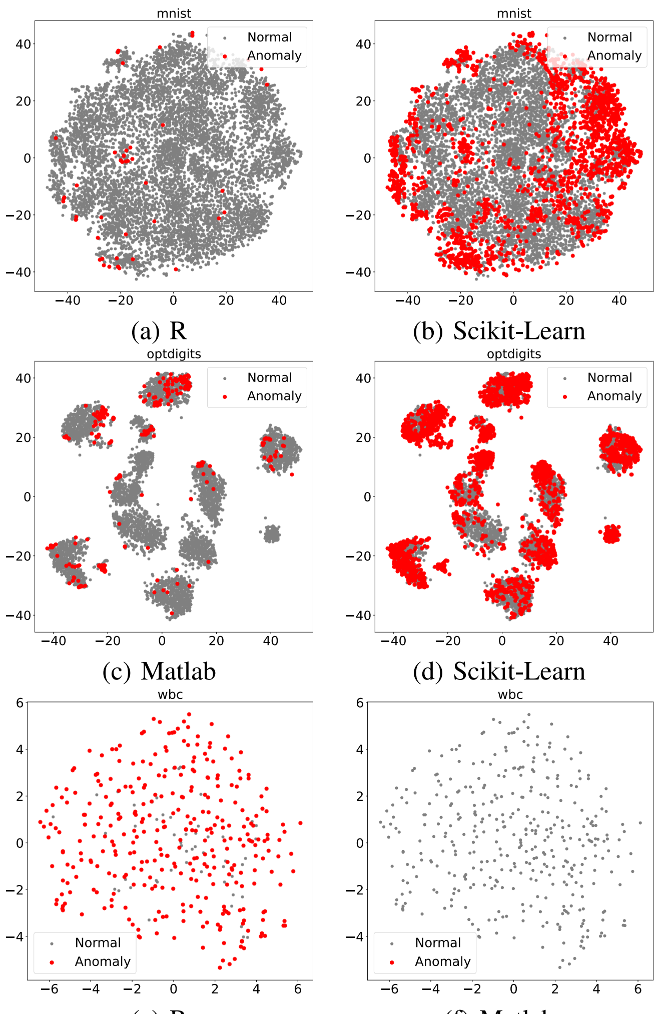
Algorithm	Toolkits	Average ARI	Percentage of Datasets with mutual ARI below 0.5
IF	Sklearn vs. Matlab	0.306	79.4%
	Sklearn vs. R	0.338	72.6%
	R vs. Matlab	0.405	63.9%
OCSVM	Sklearn vs. Matlab	0.006	100%
	Sklearn vs. R	0.283	75%
	R vs. Matlab	0.012	100%
RobCov	Sklearn vs. Matlab	0.067	98.6%
LOF	Sklearn vs. Matlab	0.413	71.2%
	Sklearn vs. R	-0.010	100%
	R vs. Matlab	0.0003	100%

datasets using various toolkits. At the top of the figure, for dataset mnist, using the IF algorithm, outputs differ substantially between toolkits R and Scikit-learm. In the center of the figure, for dataset optdigits, using the OCSVM algorithm, the outputs of Scikit-learn and R (latter not displayed in the figure) are close but the output of Matlab is very different. At the bottom of the figure, for the dataset wbc, the LOF algorithm did not find any anomalies in Matlab, whereas in R a substantial number of points were labeled anomalous.

Summary. These statistics confirm that, for a given algorithm, outcomes differ depending on the toolkit; while root causes differ (toolkits implement a given algorithm differently, toolkits use different default parameter values, etc.) the end-users' assumption that implementations of the same algorithm are interchangeable, is undermined.

VI. RELATED WORK

We are not aware of any studies on the reliability of AD algorithms or their implementations. Work in this area focused on AD performance. Ahmed et al. surveyed AD techniques in the context of detecting network anomalies. They categorized AD methods into four categories (classification,



(a) R (b) Scikit-Learn

(c) Matlab (d) Scikit-Learn

(e) R (f) Matlab

Fig. 4. Inconsistency example: (a-b) IF, (c-d) OCSVM, (e-f) LOF.

statistical, clustering and information theory) and evaluated their effectiveness using different criteria such as output of AD (is the output of AD continuous score or binary labels), attack detection priority (does the AD prioritize detecting collective anomaly or point anomaly) and computational complexity [16]. Maxion et al. explored how datasets' intrinsic structure can affect the performance of AD techniques and introduced a metric for characterizing structural regularities [17]. These efforts are complementary to our work – our findings indicate that experiments with AD implementations in general might lead to unstable results.

The issues of nondeterminism and inconsistency have been studied before in other unsupervised learning contexts, specifically Clustering and Self-organizing Maps [18]–[21]. However those findings do not translate to (neither can they serve as indicative of) Anomaly Detection, because our application area, goals, and experimental setup are different.

VII. Conclusions

Anomaly Detection is widely used but its reliability has not been questioned. We do so in this paper, by investigating 11 implementations of 4 popular AD algorithms. Our findings show that AD implementations can exhibit nondeterminism, that there a strong differences between two different implementations of the same algorithm, and that even core functionality cannot be taken for granted. We make the case that (1) AD reliability needs further scrutiny and (2) the problems affecting current AD implementations should be addressed.

Our experiments indicate that, with default settings, the implementations Isolation Forest/Matlab, Isolation Forest/R, and Robust Covariance/Scikit-learn achieved consistently good performance across datasets.

Acknowledgments

We thank Parth Patel for his help with dataset processing, and the anonymous reviewers for their valuable feedback. This material is based upon work supported by the National Science Foundation under Grant No. CCF-2007730.

References

[1] "ODDS," April 2022, http://odds.cs.stonybrook.edu/.

[2] "Chapter 10 - fetal growth," in *Twining's Textbook of Fetal Abnormalities (Third Edition)*, third edition ed., A. M. Coady and S. Bower, Eds. Churchill Livingstone, 2015, pp. 211–222.

[3] L. Van der Maaten and G. Hinton, "Visualizing non-metric similarities in multiple maps," *Machine learning*, vol. 87, no. 1, pp. 33–55, 2012.

[4] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193–218, 02 1985.

[5] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *JMLR*, vol. 11, no. Oct, pp. 2837–2854, 2010.

[6] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *ICDM*. IEEE, 2008, pp. 413–422.

[7] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12. MIT Press, 1999. [Online]. Available: https://proceedings.neurips.cc/paper/1999/file/8725fb777f25776ffa9076e44fcfd776-Paper.pdf

[8] K. Heller, K. Svore, A. D. Keromytis, and S. Stolfo, "One class support vector machines for detecting anomalous windows registry accesses," 2003.

[9] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM SIGMOD*, 2000, pp. 93–104.

[10] P. J. Rousseeuw and K. V. Driessen, "A fast algorithm for the minimum covariance determinant estimator," *Technometrics*, vol. 41, no. 3, pp. 212–223, 1999.

[11] "Anomaly detection toolbox." [Online]. Available: https://github.com/dsmi-lab-ntust/AnomalyDetectionToolbox/tree/master/Algorithms/distributionBased/LOF

[12] David-Cortes, "David-cortes/isotree: (python, r, c/c++) isolation forest and variations such as sciforest and eif, with some additions (outlier detection + similarity + na imputation)." [Online]. Available: https://github.com/david-cortes/isotree

[13] "Local outlier factor score." [Online]. Available: https://search.r-project.org/CRAN/refmans/dbscan/html/lof.html

[14] "Svm: Support vector machines." [Online]. Available: https://www.rdocumentation.org/packages/e1071/versions/1.7-9/topics/svm

[15] "OpenML," April 2022, https://www.openml.org/.

[16] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[17] R. Maxion and K. Tan, "Benchmarking anomaly-based detection systems," in *Proceeding International Conference on Dependable Systems and Networks. DSN 2000*, 2000, pp. 623–630.

[18] V. Musco, X. Yin, and I. Neamtiu, "Smokeout: An approach for testing clustering implementations," in *ICST 2019*, April 2019.

[19] X. Yin, I. Neamtiu, S. Patil, and S. T. Andrews, "Implementation-induced inconsistency and nondeterminism in deterministic clustering algorithms," in *ICST 2020*, October 2020.

[20] X. Yin, V. Musco, I. Neamtiu, and U. Roshan, “Statistically rigorous testing of clustering implementations,” in *AITEST 2019*, April 2019.

[21] S. Rahaman, R. Samuel, and I. Neamtiu, “Quantifying nondeterminism and inconsistency in self-organizing map implementations,” in *IEEE AITest*, 2021, pp. 85–92.