Outlier Detection of Generalized Deduplication Compressed IoT Data

Something

Morten Lyng Rosenquist
Faculty of Technical Sciences
Aarhus University
Aarhus, Denmark
201706031

May 28, 2022

Write abstract

***Abstract*—**

***Index Terms*—**

Write keywords

write introduction

I. Introduction

II. Background

A. Generalized Deduplication

Deduplication is a technique to perform compression in storage systems. The technique works by utilizing the simarlity of file chunks. Each unique file chunk is stored once. Subsequent copies of the chunks are then replaced with a reference to the stored chunk. The method is established and shown to have good compression gain on various practical scenarios [4]. However, if there are minor discrepancies in the file chunks, the technique will not leverage any of the similarities. Resulting in the near-identical chunks being stored in full. Sensor data from IoT devices is one example of the data potentially being near-identical.

To utilize the similarities in the almost identical data, a generalization of deduplication has been studied. This method consider the chunks at the bit level and splits them into two parts, the *base* and *deviation*. The *base* is the identical part that is to be stored once and herafter referenced with pointers. The *deviation* is the disparity between the chunks. Looking at a simple example with four 6-bit numbers, 100000, 100001, 100010 and 100011. It can be identified that the four most significant bits of the numbers are identical. Hence, leading to all having a shared *base* of 1000. The two least significant bits are then the *deviation*[6].

B. Isolation Forest

Anomaly detection is a combination of outlier- and novelty detection. Including both identifying outliers in the training data and determining if unseen observations are outliers. Isolation Forest (iForest) is an anomaly detection method. It differs from other popular techniques in the way that it identifies anomalies explicitly instead of profiling ordinary data points[1]. IForest utilizes decision trees similar to other tree ensemble methods. The main principle is to recursively split each data point, and then evaluate the amount of splits necessary to split each data point. The logic is that anomalies will requires less splits to be isolated than an ordinary point. Trees are built by selecting a random feature and then selecting a random value between the minimum and maximum value of that feature. The process is then repeated untill all data points are isolated or a maximum height of the tree is reached. An illustration can be seen on Figure 1. The graphic shows an example of a decision tree and how an anomaly is at a lower depth of the tree. When determining if an observation is an outlier iForest calculates a score, it is defined as:

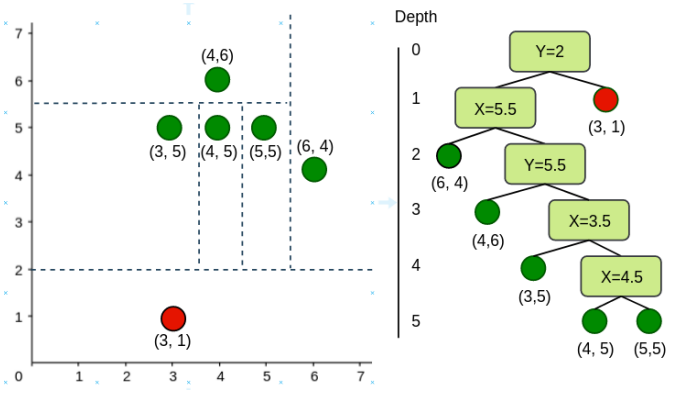

Fig. 1

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}} \quad (1)$$

$E(h(x))$ is the average length from the root node to the specific data point. This is the average over a group of trees. $c(n)$ is the average length from the root node to an external node. The anomaly score s is between 0 and 1. Scores close to 1 is seen as anomalies while values close to 0 is seen as normal data points.

III. Related work

Write related work

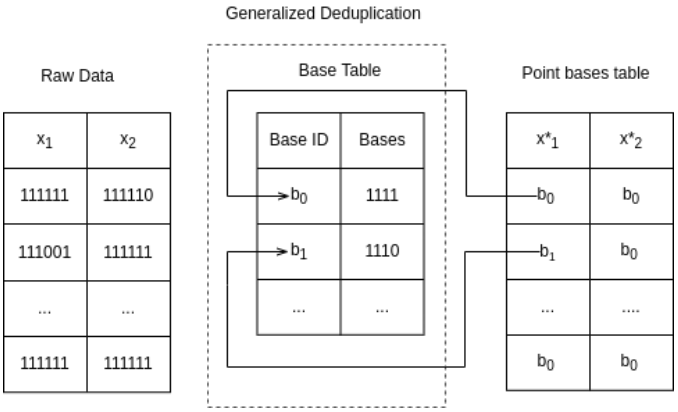

Fig. 2

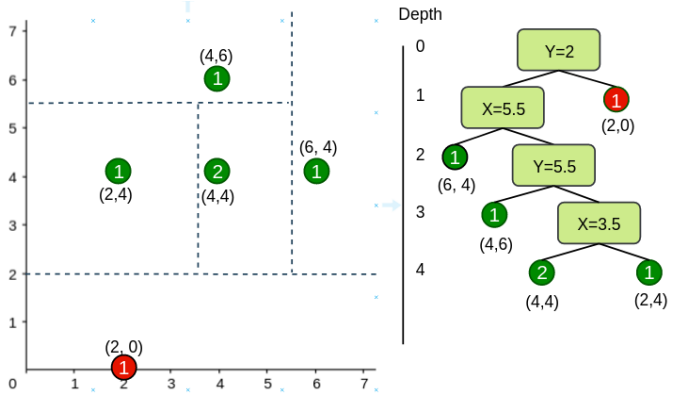

Fig. 3

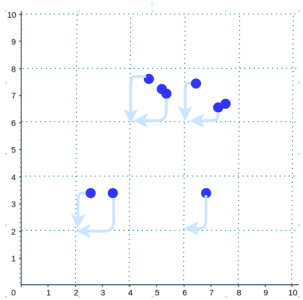

Fig. 4: first figure

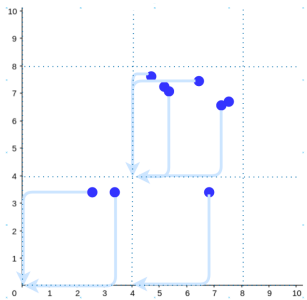

Fig. 5: first figure

Performing analytics on compressed data is not an untouched subject.

A collection of models and algorithms are developed to perform classification and anomaly detection within network communication on compressed data[5].

Another paper looks into anomaly detection based on compressed data. They do it on the edge of the cloud on compressed data. Lots of formulas regarding rate vs. distortion. Dont know compression or anomaly detection method[10].

Direct analytics on data compressed using generalized deduplication has been carried out. It was studied how clustering(K-Means, IMM, DTC) could be performed on synthetic, synthetic with noise and a power consumption data set [9].

Since isolation forest only performs horizontal and vertical splits certain anomalies will not be detected. Imagining a two dimensional data set. Then the ones having the same x and y values might not be isolated correctly. This is extended by allowing diagonal splits in the extended isolation forest[8].

IV. Methods

A. Isolation Forest on GD compressed Data

Data compressed with generalized deduplication results in having a set of bases, deviation and references linking a data point to its base and deviation. In the following example the deviation will be omitted. Say we have the data set S where $S \in \mathbb{R}^2$. Performing GD on S will result in each feature of the points being mapped to their bases. Having an point $x = [x_1, x_2]$ where $x \in S$ and some computed bases ids $b_0, b_1, ..., b_n$,then the transformed version $x* = [x_1^*, x_2^*]$ will hold the computed bases. This is depicted on Figure 2. The bases are computed on the raw data and referenced in the features x_1^* and x_2^*.

Isolation forest is then to be performed on the transformed version of the data set. The isolation forest splits before compression could be seen on Figure 1. Figure 3 is similar but is instead performing the splits on the bases. It is seen that certain data points will map to identical bases on both features.

B. DupRes Isolation Forest

The bases of GD compressed data will inherintly be grouped. Stripping the deviation of each data point will result in data points being placed in bins. This binning is illustrated on Figure 4 and 5. The graphics shows the bins created with different amount of deviation bits. The circles are samples. The dotted lines are enclosing areas where data points in an area will be mapped to the closest base in the negative direction. The arrows illustrate the base each data point will be mapped to. Having a larger amount of deviation bits is leading to larger bins.

Having larger bins might lead to a better compression rate however it could lead to undesired behaviour when trying to detect anomalies with iForest. An outlier could be mapped to the same base as an inlier on all or some of it features. Having the same base on some features will make it harder to isolate the outlier meanwhile having identical bases on all features makes it impossible. The binning aswell leads to inliers being grouped on fewer points. This causes them to be isolated more easily, and thus labeled as outliers.

Isolation Forest is not fit for the large amount duplicates that is potentially created by compressing with generalized deduplication. Therefore, a more duplicate resistant (DupRes)

version is proposed. The core idea of the new version is to utilize the amount of duplicates when building the tree. The amount is then used to adjust the score of an observation. A revised version of the score function is:

$$s(x, n) = 2^{-\frac{E(h(x))+log_2(x_{count})}{c(n)}} \quad (2)$$

The new function differs from Equation 1 by the introduction of the $log_2(x_{count})$ term. x_{count} is the amount of occurences of the given sample. The reason behind using the binary logarithm is firstly that having one occurance will not modify the score, $log_2(1) = 0$. Secondly, it is a strictly increasing function. Resulting in the higher the x_{count}, the larger adjustments will be made to the score. The modification makes no changes in the range of s and in how it should be interpreted. For further details, see the original paper [1].

The change implies that the count of each sample is known. This requires extending what is done in the training phase of the model. Beside building the decision trees, the model must store each unique sample with the amount of occurences. Worst case the training set contains no duplicates and will store all training samples with the count of one. Hereby, the model is not optimal if the data is expected to have a low amount of duplicates. The flow during the evaluation of unseen observations, is to identify the ones that was seen in the training phase and retrive their counts. The adjusted score is then computed and can be used to identify anomalies.

V. Experiment

The experimental setup will be described in this section. Three models will be evaluated, Isolation Forest on the original data before compression, Isolation Forest on the compressed data and the proposed DupRes Isolation Forest on the compressed data. We will refer the different model in the same order as, **original**, **bases** and **DupRes**. The models are evaluated on various datasets containing anomalies. The datasets are described in the following subsections.

Synthetic

The synthetic dataset is self-generated. It is two dimensional where each feature is an integer. The dataset contains three clusters where the samples are normally distributed with a standard devation of 2, $N(\mu, 2^2)$ where μ varies on the cluster center. The data set is represented before and after compression on Figure 6. Each cluster consists of 150 samples and 20 outliers are generated between the clusters. There is then a total of 470 samples.

Pendigits

Pendigts is a database for handwritten digits. Originally it contains 16 integer features with 10 classes (one for each digit). The datasets 6870 samples is mapped to inliers and outliers to be utilized for anomaly detection [12].

Shuttle

The shuttle dataset contains 49097 samples collected from a shuttle. It has 9 dimensions all being integers [12].

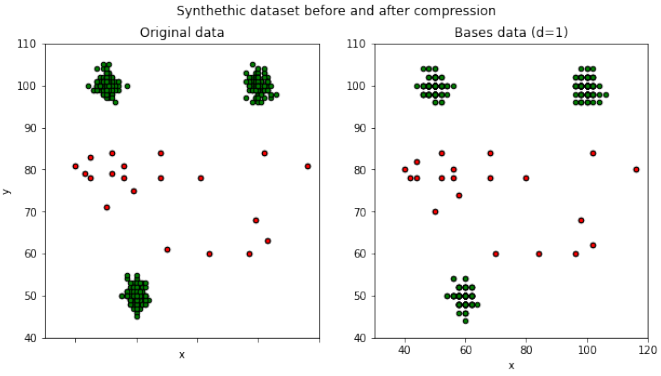

Fig. 6

Waveform

Waveform contains 3443 samples with 100 outliers. It has 21 features that are all numeric [11].

WBC

The WBC dataset contains data of measurements from breast cancer cases. The inliers is the benign class, while the outliers are the malignant class. It contains 278 samples with 30 dimensions [12].

A. Preprocessing

All of the features of the synthethic, pendigits and shuttle datasets can be represented as an one byte integer. However, the WBC and waveform datasets contains floating numbers. Therefore, the features are scaled with a factor of 10.

B. Procedure

The experiments is realized in Python and can be found for replication on github[1]. DupRes is implemented by extending *sklearn*'s existing Isolation Forest functionality. Thus, a fork is as well to be found on github[2]. Each dataset is split in a training set and test set with a 80%/20% proportionality. The split is done in a stratified manner, meaning it is ensured to be inliers and outliers in both sets. We had the three models; **original**, **bases** and **DupRes**. The original requires no additional transformation of the data. However, the other two requires the GD compressed version of the data. Hence, we store both the original and the compressed data. The models are then trained on the training data and evaluated on the test data in sequence.

C. Performance evaluation

As Isolation Forest by nature introduces randomness in building its decision trees, we train and evaluate each model 50 times. At each iteration various performance metrics are stored. When all iterations are complete we remove the lowest and highest scoring of each metric and take the mean of the remaining values. The mean value is then the resulting

[1] https://github.com/mlRosenquist/au-mlr-research-and-development-dedup
[2] https://github.com/mlRosenquist/scikit-learn

value for that metric on a given model. The different metrics that is looked at is training time, testing time, accuracy, f1, recall and precision. The training and testing time tells, how much processing time is used to fit the model on the training data and to predict unseen samples in form of the test data. Accuracy show in percentage how many observations are predicted correctly. Recall relate to the ability of finding all positive samples. In the experiments conducted we have set the inliers to be the positives while the outliers to be the negatives. Precision includes the false positives in its calculation. This means in our case that precision can intuitively be used to see if the outliers are correctly detected. The f1-score is the harmonic mean of the precision and recall [3].

VI. Results

This section will present and interpret the results from the conducted experiments. There are many experimental setups. Three different models evaluated on five datasets with varying amount of compression. Then in addition to that, there is many metrics to be looked at. To illustrate this comprehensive amount of results, Table I has been constructed. The Table contains the results of three of the five datasets. The Table will be reffered to repeatedly in the following examination of the results. The values that are reffered to is marked with bold.

A. Execution time

Starting off with looking at the execution time. The training and test time on the synthetic dataset can visually be seen on Figure 7. It shows no apparent difference between neither the training time or testing of the original and bases model. However, there is a clear difference between those two and DupRes. Both the training in which the uniques are found and in the testing in which the duplicates are looked up adds additional execution time. This performance cost can also clearly be seen in the table for the pendigits dataset. Here we have 93.7/55.3 ms train/test time for the original model, while DupRes with 1 deviation bit has 135/282 ms train/test time. Additionally, it can be seen that DupRes with 6 deviation bits has 145/131 train/test time. This implies that, as the amount of deviation bits rise the testing time decreases. This makes sense as less uniques are found during the training phase, and thus less entries in the table need to be looked through during testing. This incentivize to not use the model if a low amount of duplicates is expected in the data or a low amount of deviation bits is utilized. Originally an implementation utilizing pandas'[2] dataframes was used. However, this performed notably worse then the current implementation which uses numpy[7] arrays. To optimize this further an implementation could be made in c or c++ and utilize the interoperability to Python.

B. Metrics and Scoring

Then we look at the performance of the models in terms of metrics and scoring. Figure 8 illustrates the performance metrics of the models on the datasets. In general it can be seen on the figure that the models are performing quite similar.

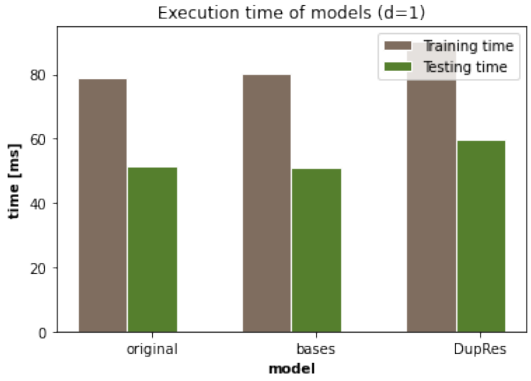

Fig. 7

Investigating the metrics for the synthetic dataset a clear pattern can be seen. The bases model performs better than the original, meanwhile DupRes performs better than the bases. The cause of this can be derived from the recall. Since the positives are the inliers the original is simply classifying more inliers as outliers than the others. It is using too few splits in isolating them. As we compress the data the inliers are harder to isolate due to clustering. Utilizing the many duplicates in DupRes we are then classifiying less inliers as outliers. Similar trends is not be seen on the other datasets. On the recall of the pendigit dataset it is seen that the original model predicts less false positives than the others.

Looking at the table there is several interesting observations. All models except two detect every outlier. The two instances are the bases model and DupRes on the synthetic dataset with 6 deviation bits. The bases model predicts every observation as an outlier resulting in 0 scores. Since a lot of data points is grouped with this amount of deviation bits, DupRes will not do the same action as the other model, and therefore have a precision score of 0.98. This implies that one outlier observation might have grouped with the inliers. DupRes will wrongfully push this to be an inlier. Investigating the performance regarding the WBC set it is seen that the performance is almost identical. There are minor discrepancies in the accuracy however the rest is identical. This implies that even though compression is performed, the analytical capabilities is still intact. This is a general trend across all datasets, as was also depicted on Figure 8.

The amount of deviation bits shows a pattern. Looking at the Pendigits dataset the higher the amount of deviation bits the more inliers will be classified as an outlier. This is natural as the compression will place the observations in larger bins that are easier to separate. DupRes does not classify these inliers as outlier as the large binning has resulted in more duplicates and therefore pushing their score towards being an inlier.

C. Compression and Memory

The two right-most column in the table describes the compression rate and memory accesed for each model. As the features of all datasets is 8-bit numbers the compression

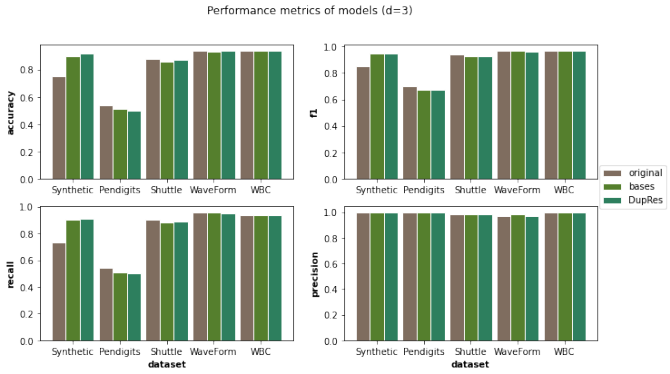

Fig. 8

rate is quite simple to calculate: $c_{rate} = \frac{d}{8}$, where d is the amount of deviation bits. The original model naturally has no compression. The two others compression varies on the their amount of deviation bits. $d = 6$ for both of the models catches all outliers except on the synthetic data set as described previously. This is a quite significant compression rate of 75%. Memory Accessed describe how much of the original data that is accesed. It is kind of the reverse of the compression rate. Most interestingly is the DupRes which also has the count table. This is to illustrate that it needs to store the table of uniques and their counts in memory. This shows the tradeoff with DupRes and performing original Isolation Forest on the bases. They have the same compression rate. However, DupRes needs additional memory to store the table. As seen in the results, the training time spent creating the table and the testing time looking up in the table is not to be neglected.

VII. Conclusion

This paper looked into performing anomaly detection on compressed data. Generalized deduplication was the compression algorithm of choise, that enables direct analysis of the bases derived from the transformed data. The amount of compression controlled the clustering and performance of the anomaly detection models. The duplicates created by the algorithm is taken advantage of in an extended version of Isolation Forest. Comparing the proposed model with original Isolation Forest it showed that utilizing the duplicates improves classification performance. However, the improved scoring is a tradeoff with memory usage and execution time. A general observation is that performing Isolation Forest on the uncompressed data versus compressed had no large descrepancies in performance.

VIII. Future work

DupRes currently uses the $log_2(x_{count})$ term to tune the score towards being an inlier based on the count. This is potentially a hyperparameter that can be tuned. Maybe a completely different term fits better. This could be something that also includes the dimensionality of the data or other factors. This report looked at a limited amount of datasets. It would therefore be interesting to see how the different models perform on others. In regards to generalized deduplication we only investigated 8 bit integers. Thus, a task is to research the anomaly detection behaviour on floating numbers.

References

[1] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation Forest". In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[2] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[3] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[4] Wen Xia et al. "A Comprehensive Study of the Past, Present, and Future of Data Deduplication". In: *Proceedings of the IEEE* 104.9 (2016), pp. 1681–1710. DOI: 10.1109/JPROC.2016.2571298.

[5] Christina Ting et al. "Compression Analytics for Classification and Anomaly Detection Within Network Communication". In: *IEEE Transactions on Information Forensics and Security* 14.5 (2019), pp. 1366–1376. DOI: 10.1109/TIFS.2018.2878172.

[6] Rasmus Vestergaard, Daniel Enrique Lucani Rötter, and Qi Zhang. "Generalized Deduplication: Lossless Compression for Large Amounts of Small IoT Data". English. In: *European Wireless Conference*. VDE Verlag GmbH, 2019, pp. 67–71. ISBN: 978-3-8007-4948-5. URL: https://www.vde-verlag.de/proceedings-en/564948016.html.

[7] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[8] Sahand Hariri, Matias Carrasco Kind, and Robert J. Brunner. "Extended Isolation Forest". In: *IEEE Transactions on Knowledge and Data Engineering* 33.4 (Apr. 2021), pp. 1479–1489. DOI: 10.1109/tkde.2019.2947676. URL: https://doi.org/10.1109%2Ftkde.2019.2947676.

[9] Aaron Hurst et al. "Direct Analytics of Generalized Deduplication Compressed IoT Data". English. In: *IEEE Global Communications Conference (GLOBECOM)*. IEEE Global Communications Conference (GLOBECOM). IEEE Conference and Exhibition on Global Telecommunications, GLOBECOM ; Conference date: 07-12-2021 Through 11-12-2021. IEEE, 2021. DOI: 10.1109/GLOBECOM46510.2021.9685589. URL: https://globecom2021.ieee-globecom.org/.

[10] Alex Marchioni et al. "Anomaly Detection based on Compressed Data: an Information Theoretic Characterization". In: (Oct. 2021). DOI: 10.36227/techrxiv.16738171.v1.

Model	Dataset	Dev. bits	Metric				Time		Compression Rate	Memory Accessed
			acc.	f1	rec.	prec.	T(ms)	E(ms)		
Original	Synthetic		0.75	0.85	0.73	1	78.8	51.4	0%	100%
	Pendigits		0.54	0.70	0.54	1	**93.7**	**55.3**	0%	100%
	WBC		0.94	0.97	0.94	1	75.2	22.0	0%	100%
Bases	Synthetic	1	0.72	0.83	0.71	1	80.2	51.2	12.5%	87.5%
		3	0.90	0.95	0.90	1	74.6	46.4	37.5%	62.5%
		6	0.04	0	0	**0**	74.3	45.0	75.0%	25.0%
	Pendigits	1	**0.53**	0.70	0.53	1	95.1	55.7	12.5%	87.5%
		3	**0.51**	0.67	0.51	1	94.3	55.9	37.5%	62.5%
		6	**0.42**	0.59	0.41	1	95.3	56.0	75.0%	25.0%
	WBC	1	0.94	0.97	0.94	1	75.3	21.6	12.5%	87.5%
		3	0.94	0.97	0.94	1	75.6	21.6	37.5%	62.5%
		6	0.95	0.97	0.94	1	74.8	21.4	75.0%	25.0%
DupRes	Synthetic	1	0.87	0.93	0.87	1	90.4	59.7	12.5%	87.5% + count table
		3	0.92	0.95	0.91	1	85.8	55.4	37.5%	62.5% + count table
		6	0.98	0.99	1.00	**0.98**	86.5	53.4	75.0%	25.0% + count table
	Pendigits	1	**0.54**	0.70	0.53	1	**135**	**282**	12.5%	87.5% + count table
		3	**0.50**	0.67	0.50	1	135	286	37.5%	62.5% + count table
		6	**0.80**	0.89	0.80	1	**145**	**131**	75.0%	25.0% + count table
	WBC	1	0.94	0.97	0.94	1	86.9	23.4	12.5%	87.5% + count table
		3	0.94	0.97	0.94	1	87.7	23.7	37.5%	62.5% + count table
		6	0.95	0.97	0.94	1	85.6	23.5	75.0%	25.0% + count table

TABLE I: Table of stuff

[11] LMU. *Waveform*. https://www.dbs.ifi.lmu.de/research/outlier-evaluation/DAMI/literature/Waveform/. [Online; accessed 28-May-2022]. 2022.

[12] Stony Brook University. *Outlier Detection DataSets (ODDS)*. http://odds.cs.stonybrook.edu/. [Online; accessed 28-May-2022]. 2022.