Deleting Trees from Isolation Forest

A new Isolation Forest variant for producing better anomaly scores

Márk Dániel Szalai

Supervisor:
Gábor Horváth dr.

Budapest University of Technology and Economics
Department of Networked Systems and Services
Budapest, Hungary
October 2022

Supported by the **ÚNKP-22-2-III-BME-240** New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.

Contents

1 Introduction

The effective and reliable automatic detection of anomalies is an active field of research and several companies are looking for solutions that can be used in industrial applications. Several deeplearning model has been proposed to handle the problem. In general, the decisions of these models are difficult to explain which makes it more difficult to use them in industrial applications. Therefore, in practice algorithmical approaches are very common of which the ensemble based models are especially promising. These models utilize several weaklearners to produce their predictions, letting them specialize and focus on certain properties of the data set. Isolation Forest exploits this concept by building several trees each of which tries to separate the samples of the data set with hyperplanes perpendicular to the axes. A huge deficiency of the algorithm is the lack of performance measuring and potential deletion of the underperforming weaklearners. The same insight applies to the Extended Isolation Forest algorithm which separates the data set with arbitrary hyperplanes. This deficiency may produce anomaly scores with significant overlap between normal and anomalous samples. Since anomaly detection is typically an unsupervised learning problem we can only rely on the anomaly scores to determine how normal our samples are which makes it crucial to have well separating scores. The goal of the new algorithm is to solve this issue, produce better separating scores, introduce a new deletion process and enable industrial applications.

2 Isolation Forest and its variant

Isolation Forest with the recent introduction of its extended variant is becoming rather popular when it comes to anomaly detection. One of its better qualities is the fact that it allows for the joint handling of categorical and numerical variables. There is only one restriction (to allow splitting data between child nodes, see below): the categorical data must be ordinal, meaning there has to be a clear "smaller than"/ "bigger than" relationship between the values of the categorical variables. This can be easily done with an arbitrary encoding scheme.

2.1 Isolation Forest

Isolation Forest (hereinafter IF), first introduced in [1], builds a set of binary trees (hereinafter forest) on a subset of samples with the goal of separating them with random cuts. The assigned anomaly score of a sample is based on the average length to isolate it in the trees of the forest.

To build an isolation tree, IF first randomly select a subset of the input data set X, denoted by $\widetilde{X}$. Once $\widetilde{X}$ is obtained, a cutting process enfolds. This is done by randomly selecting an attribute q and a value p such that $x \in \widetilde{X} | \min(x[q]) \leq p \leq \max(x[q])$. Attribute q and value p divides the data points into two subsets: the ones whose attribute q value is larger than p and the rest. Once the two smaller data-sets are produced a left and right child node is created which will repeat the cutting process and also creating child nodes with the even smaller data-sets. This process is repeated in a recursive fashion until a node receives a single sample or the same samples, which will make it a leaf node. If $\widetilde{X}$ is has $n = |\widetilde{X}|$ samples then the average path length of the binary tree is [1]

$$c(n) = 2H(n-1) - \frac{2n-2}{n}, \tag{1}$$

where $H(n) = ln(n) - EC$ and $EC = 0.5772156649$ is the Euler's constant. $c(n)$ is used to normalize the anomaly score.

Based on an isolation tree and a data sample x, the procedure evaluates the length of the path, $h(x)$, by performing a binary search in the tree, where the cutting criteria ($q \leq p$) determines which child node should be chosen in the search. Reaching a leaf node on the path (easy to see that there can be one and only one for each sample) will be considered a positive hit and the end of the path.

Having T trees in the isolation forest, the mean of $h(x)$ (i.e. the expected value of $h(x)$, i.e. the average path length of sample x in the forest) is

$$E(h(x)) = \frac{1}{T} \sum_{t=1}^{T} h_t(x),$$

where $h_t(x)$ is the length of the path of sample x in isolation tree t. The anomaly

score of data point x (determined by the isolation forest of trees with n leaves) is

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

where $c(n)$ is the average path length of a binary tree (see Equation 1).

Since anomalies are usually easier to separate from the rest of the data, they should on average be separated earlier with the cutting processes, resulting in shorter path lengths and thus higher anomaly scores. This easier separability phenomena is what being exploited by the algorithm.

The authors of the IF algorithm claim that IF:

- has a linear time complexity (in both the number of samples and the number of trees) with a low memory usage,
- has the capability to scale up to handle extremely large and even high dimensional data sets (the required number of cuts only depends on the size of the initial subset of data),
- utilizes no distance or density measures to detect anomalies thus coping more effectively with the curse of dimensionality.

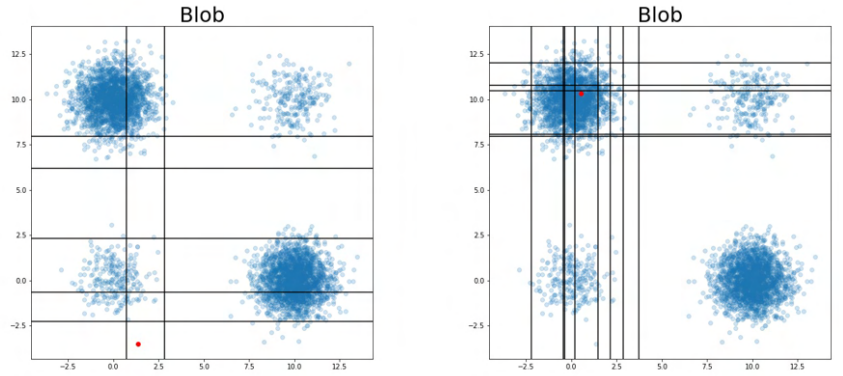
The computational complexity of the algorithm can be reduced by terminating the construction of isolation trees at certain depth, limiting the maximal height of the trees. The limit has to be set to be large enough to isolate anomalous points. We will denote this variable with h_{max}. This way the parameters of the algorithm are

- the size of the forest, i.e., the number of isolation trees to build: T,
- the sub-sampling size, i.e., the number of randomly selected data points to build a tree: $n = |\widetilde{X}|$,
- the maximal height of a tree: h_{max}. This is often set to $h_{max} = \log_2 n$, since we are only interested in the anomalous samples (likely with shorter than average path lengths) the logarithm of the sub-sampling is a reasonable heuristic.

The complete algorithm can be described as a two-stage process.

1. The first (training) stage builds the isolation forest, i.e. constructs T isolation trees, each from a $\widetilde{X}$ subset of n randomly selected data samples with a default maximal height of $\lceil \log_2 n \rceil$.
2. The second (testing) stage computes the anomaly score for each data point by traversing each tree for each sample until the sample reaches a leaf node.

See Figure 1 for an example of Isolation Forest tree traversal and sample isolation in case of an anomalous and a normal sample, the height limit stopped the tree from needlessly achieving complete isolation of the normal sample.



(a) Anomalous sample

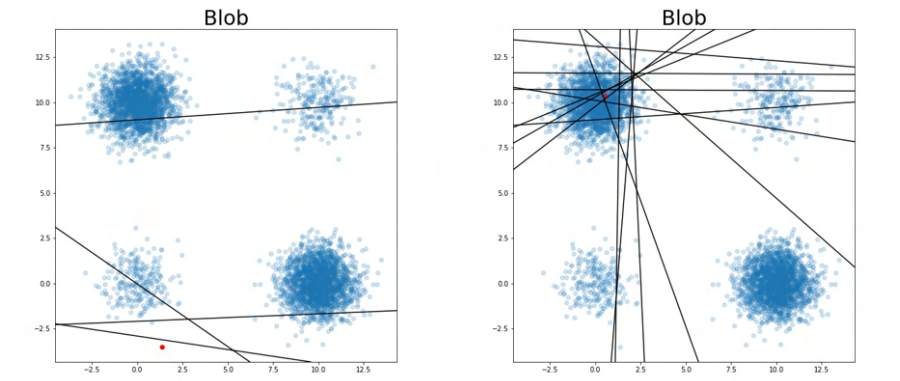
(b) Normal sample

Figure 1: Isolation Forest tree traversal and sample isolation, the examined sample is indicated with red color

2.2 Extended Isolation Forest

EIF, first described in [2], introduces a new separation method to the IF algorithm. While IF selects a single random attribute and a splitting point between the attribute's minimum and maximum values, EIF creates a projection vector, computes the dot product of the vector and the sample, selects a splitting point between the minimum and maximum values of the projected values and splits the samples accordingly. This means that EIF can split the samples along any hyperplane, while with IF the separation is always performed along a hyperplane, parallel with the axes.

See Figure 2 for an example of Extended Isolation Forest tree traversal and sample isolation in case of an anomalous and a normal sample, once again the height limit stopped the tree from needlessly achieving complete isolation of the normal sample. Comparing the results to 1 we can see that EIF required fewer cuts to isolate the anomaly than IF and could significantly narrow down the area of the normal sample.



(a) Anomalous sample (b) Normal sample

Figure 2: Isolation Forest tree traversal and sample isolation, the examined sample is indicated with red color

2.3 Scalability

The complexity of both Isolation Forest and its extended variant is $\mathcal{O}(T * |X| * h_{max})$, where T is the number of trees, $|X|$ is the number of samples (sub-sampling size during the fit phase and the size of the whole data-set during the predictions phase) and h_{max} is the maximum allowed depth of the trees. This means that both algorithm scales incredibly well on high dimensional data as the number of dimensions only affects the projection vectors and dot products of EIF.

Their parallelization is also easy, almost trivial. During the training phase the tree construction can be done parallel and during the prediction phase each sample's anomaly-score can be calculated parallel. This means that both algorithm can be easily deployed in a distributed ecosystem (for example there is a LinkedIn implementation in Apache Spark [3]).

2.4 Online Isolation Forest

It would be great if we could use Isolation Forest on data streams. This requires the ability to adapt to the current behaviour of the data. A trivial way of achieving adaptation is to implement a ring-buffer and for every n sample build a new tree and insert into the buffer. This would mean that after we fill up the buffer with trees, the oldest tree will always be overwritten, thus only the most recent trees will affect our prediction. In the next section we would like to propose an alternate deletion method that may also prove useful on data sets with no concept drifts.

3 The proposed new algorithm

3.1 Data-sets used in the experiments

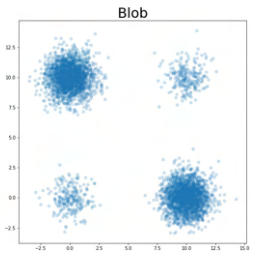
We have used several synthetic and publicly available, real-world data-sets to investigate the behaviour of our modified algorithm and compare it to the original Isolation Forest.

3.1.1 Synthetic data-sets

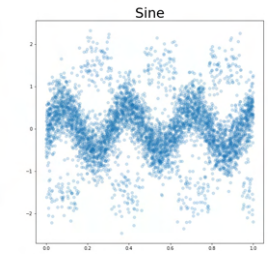
We have generated three different data-sets that were mainly used these to establish and test hypothesises, thus we will not discuss them in detail:

- Blobs: We have used four $2D$ normal distribution, two larger and two smaller and placed them into the corners of a square. The anomalous considered samples are the ones from the smaller distributions.
- Sine wave: the anomalous samples are the ones not on the sine wave.
- Ring: The normal samples form a ring around the anomalous samples. This data set proved to be an unfortunate choice, as the Isolation Forest algorithm tries to isolate the samples and the center is difficult to separate.

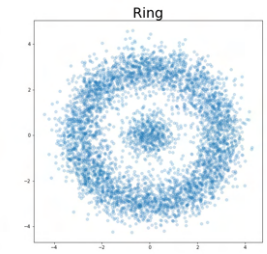
See Figure 3 for a visual representation of these data-sets.



(a) Blob data-set



(b) Sine data-set



(c) Ring data-set

Figure 3: Synthetic data-sets

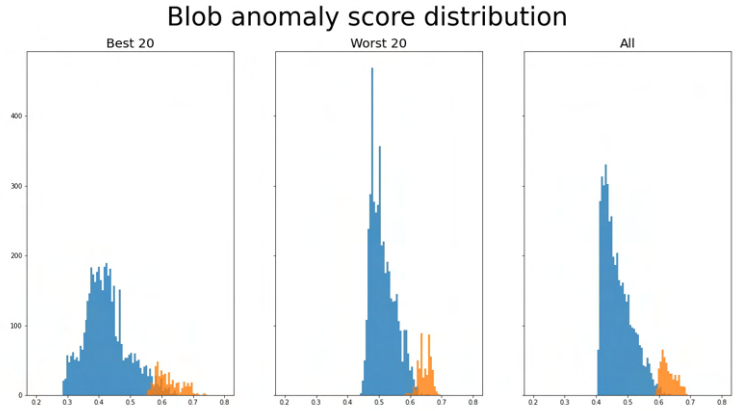
3.1.2 Real-world data-sets

These are the data-sets we used to actually evaluate the algorithm and the proposed changes. We only had one criteria for these data-sets: they must not have non-ordinal categorical values, more strictly: all attributes must have numerical values (but they can be categorical). Obviously other data-sets could be also used, but the categorical values would have to be encoded to numerical values, which might distort the results of our algorithm. Most of them can be found in the UCI Machine Learning Repository [4]. Another source of the data-sets is the Outlier Detection DataSets Library (ODDS) [5]. Some of the data-sets in the ODDS Library originates from the UCI Machine Learning Repository, but they include changes that make them more suitable for anomaly detection, thus we used ODDS for our source of data. There are two exceptions that can not be found in ODDS, these are the CreditCard and the PumpSensor data-sets. Both can be found on Kaggle at [6] and [7].

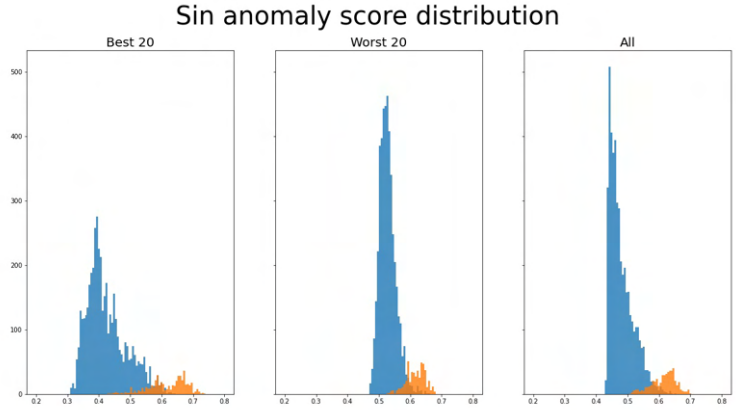
3.2 Inspiration

During our experiments we have noticed an interesting behaviour in the algorithm. Suppose we count the external nodes (leaves) in a tree on each of its level. This would leave us with a vector V, where $V[i]$ is the number of external nodes on the ith level of the tree. Now count the number of samples stopping on each level of the tree. We shall denote it with C. Notice that both V and C has the same length: the depth of the tree. Now if we perform element-wise division on the two vector (numerator: C, denominator: V) we get vector R (note that a sample can not stop on levels with not leaves which means that if the denominator is 0 then so is the numerator, thus let us define $0/0 = 0$ in order to avoid mathematical ambiguity). The idea was simple and it builds on the ideas of the Isolation Forest algorithm: anomalies are easier to separate, thus anomalies will require fewer cuts to be separated, thus they will be closer to the root of the tree. Combining this idea with the fact that in general there are fewer anomalous than normal samples then it can be assumed that the elements of the resulting R vector should follow a uniform distribution. The original idea was to delete those trees which has the R vector that diverges the most from a uniform distribution. This can be measure with a simple KL-divergence, which basically would result in the entropy of R.

The expectation was that the trees with the largest KL-divergence would perform worse, however the exact opposite could be seen during testing: the "worse" trees generally assigned higher anomaly scores to the anomalous samples and lower anomaly-score to the normal samples resulting in fewer overlaps between the anomaly-scores of normal and anomalous samples. This behaviour can be consistently observed on several real-world and synthetic data sets as seen on Figures 4 and 5, where the anomalous samples are colored orange and the normal samples are blue:

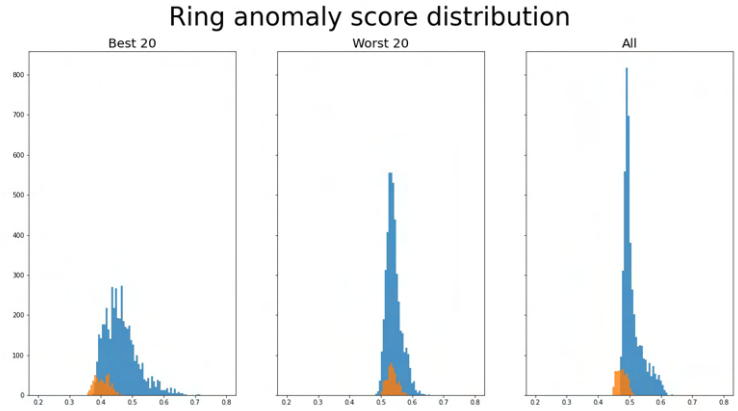


(a) Blob data-set

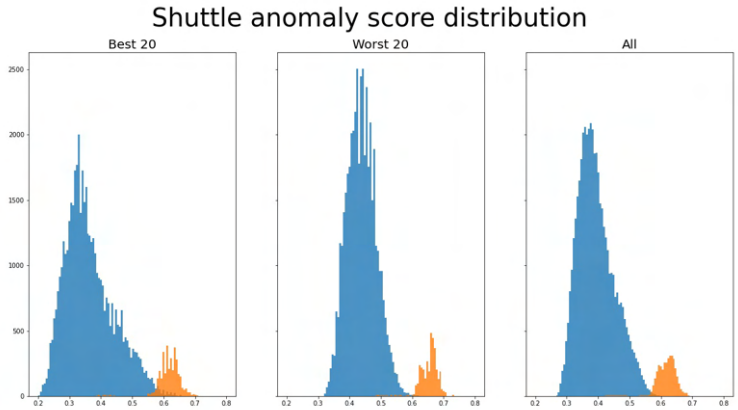


(b) Sine data-set

Figure 4: Distribution of anomaly scores of the best and worst trees on different data-sets



(a) Ring data-set



(b) Shuttle data-set

Figure 5: Distribution of anomaly scores of the best and worst trees on different data-sets

In all of the above examples we can see that the "worst" 20 trees achieved a better separation of the anomalous samples in terms of their assigned scores. One exception could be the Ring data-set, but in this case the "worst" trees still performed better because they assigned a higher score to the anomalous samples. Please note that in unsupervised learning (to which the algorithm belongs) we can only rely on the anomaly-score of the samples to determine whether they are anomalous or not. Higher score should mean higher chance of being an anomaly, in case of the Ring data-set we can see that the "worst" trees assign the better scores to the anomalous samples.

3.3 Foundations of our algorithm

The above described phenomenon led us believe that it is worth examining the structure of the trees of the forest as it seems that there are good indicators of the usefulness of a tree that could be exploited to develop a deleting mechanism for the Isolation Forest algorithm that would ultimately result in better anomaly scores. One important criteria we wanted to set is to not use the actual samples in calculating the usefulness of the trees, thus our investigated metrics are solely based on the tree structures.

3.4 Implementation

We have implemented the algorithm with out modifications in the C++ language. To further increase the speed up the execution of the algorithm we have added OpenMP [8] support to both the fit and predict phases of the algorithm, which allows the parallel construction of trees and parallel scoring of samples. This resulted in a significant speed boost which could be utilized during testing. Our implementation has an extra parameter: *metric*. This parameter can be used to specify witch metric we want to sort our trees and which part of the sorted trees we want to keep (Top,Bot). In the future extra metrics can be easily added to the code-base. The algorithm also accepts an *extension_level* parameter which when set larger than 0 would create trees using the splitting criteria defined in Extended Isolation Forest [2]. We found that the structure of trees built with this splitting method significantly differs from the structures of Isolation Forest trees and thus would require a whole new set of metrics, thus they are out of the scope of this paper.

Finally we implemented a Cython [9] wrapper around our C++ code-base, which makes it possible to run the module directly from a Python environment on both Windows and Linux operating systems.

As a way to help visualize the trees of the forest we also made it possible to generate a Graphviz representation of a tree, which can then be supplied to Graphviz for plotting. See Figure 6 for an example plot, generated by [10].

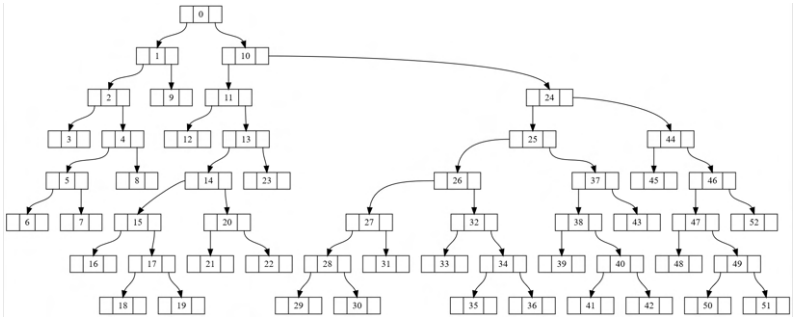


Figure 6: Example of tree visualization

For the Isolation Forest algorithm we use the sklearn [11] implementation.

3.5 Investigated metrics

We have investigated several metrics that are being used to describe binary tree structures. One of them is the Colless' imbalance described in [12], designed to measure the imbalance of trees. It is defined as follows in Equation 3:

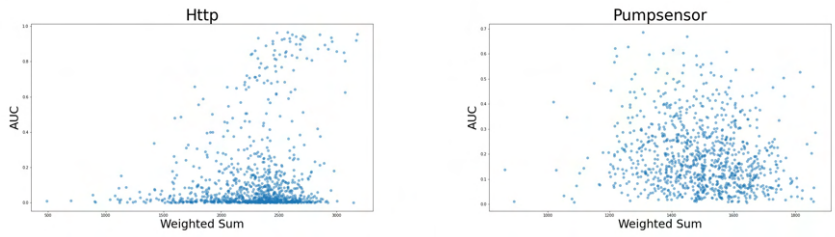
$$I_C = \frac{\sum_{(\text{all interior node})} |T_L - T_R|}{\frac{(n-1)*(n-2)}{2}} \tag{3}$$

Where T_L and T_R are the leaves of the left and right sub-trees respectively. The sum is normalized by the maximum value of the sum: n is the number of leaves and $n-1$ is the number of internal nodes. This metric is often criticized for not weighing the nodes equally (the absolute difference between sub-trees of near-root nodes can be significantly higher than the ones further from the root and the normalization can easily diminish the contribution of the latter nodes).

A more robust alternative of the Colless' imbalance is the I_2 imbalance introduced in [13]. This metric also uses the absolute difference in the number of nodes but it handles normalization differently. Rather than normalizing the resulting sum, it normalizes each member of the sum with the number of leaves in the sub-tree of the focal node (i.e.: $T_L + T_R$). The metric is defined as in Equation 4.

$$I_2 = \frac{\sum_{(\text{all interior node})} \frac{|T_L - T_R|}{T_L + T_R - 2}}{n-1} \tag{4}$$

A third metric was devised by obtaining anomaly-scores from the individual trees and then calculating the ROC-AUC scores of the predictions and then looking for similarities between high AUC yielding trees. We found on several data-sets that there is a positive correlation between the AUC score of the trees and the "deepness" of leaves. We found that the greater the average distance between leaves and the root, the better AUC score a tree produces, see Figure 7: in both data-set the lowest scoring trees produce the worst AUC yielding anomaly-scores.



(a) HTTP data-set (b) Pumpsensor data-set

Figure 7: AUC of each trees prediction in function of the weighted sum of the number of leaves on each level

In the following section we investigate different deletion criteria based on these metrics.

3.6 Deleting Isolation Forest

The algorithm works similarly to the basic Isolation Forest algorithm. The difference is an additional deletion step which makes the algorithm capable of online learning, while potentially increasing the quality of anomaly-scores produced with the more refined deletion criteria that originates from the structure of trees. To simulate an online learning environment we created ten times the trees we called the algorithm with (T) and we investigated how the scores described in 3.5 could be used to enhance the performance of the forest, thus we ordered the trees according to these metrics and evaluated the performance of the forest with both the highest and lowest scoring T trees. Since randomness plays a huge part in the algorithm we run each experiment 20 times and calculated the mean and standard deviation of the resulting AUC scores. Both Isolation Forest and our algorithm (denoted with the used metric and whether the highest (Top) or lowest (Bot) scoring trees are kept for the evaluation). We wanted to see how the maximum allowed affected the usability of the metrics (we have seen significant difference in performance in some cases), thus we performed the experiments with a sub-sampling size of 128 and with the following depth limits: 128 (practically unlimited for 128 samples), 20 (a reasonable middle-ground) and with 7 ($\lceil \log(128) \rceil$, which is the same that Isolation Forest uses). Note that Isolation Forest has only been run with its paper-defined depth limit, it is only included in all of the tables as a reference. The results can be seen in the following Tables 1, 2, 3.

Data set	Criteria [AUC] - mean ± std						
	IF	I2 Top	I2 Bot	IC Top	IC Bot	WAVG Top	WAVG Bot
CreditCard	0.949±0.002	0.949±0.002	0.948±0.002	0.949±0.002	0.947±0.002	0.949±0.002	0.946±0.002
ForestCover	0.853±0.022	0.852±0.021	0.846±0.023	0.886±0.018	0.808±0.023	0.893±0.013	0.791±0.024
HTTP	0.997±0.002	0.054±0.000	0.989±0.004	0.054±0.002	0.990±0.005	0.054±0.000	0.991±0.003
Mammography	0.864±0.005	0.875±0.005	0.870±0.005	0.874±0.005	0.872±0.004	0.878±0.003	0.865±0.006
Musk	0.315±0.033	0.406±0.024	0.394±0.029	0.339±0.017	0.512±0.024	0.385±0.019	0.473±0.026
PumpSensor	0.956±0.011	0.956±0.013	0.959±0.009	0.961±0.007	0.952±0.011	0.956±0.010	0.955±0.011
Shuttle	0.997±0.000	0.997±0.000	0.997±0.000	0.998±0.000	0.995±0.001	0.998±0.000	0.995±0.001
SMTP	0.863±0.009	0.854±0.006	0.874±0.007	0.861±0.006	0.859±0.008	0.850±0.007	0.874±0.005

Table 1: Average AUC scores of 20 runs, each tree built with a 128 depth limit.

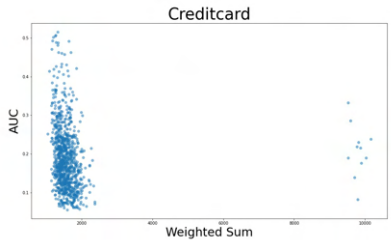
Data set	Criteria [AUC] - mean ± std						
	IF	I2 Top	I2 Bot	IC Top	IC Bot	WAVG Top	WAVG Bot
CreditCard	0.949±0.002	0.948±0.002	0.948±0.002	0.950±0.002	0.946±0.003	0.949±0.002	0.947±0.003
ForestCover	0.853±0.022	0.849±0.023	0.853±0.020	0.891±0.018	0.794±0.024	0.900±0.013	0.790±0.032
HTTP	0.997±0.002	0.957±0.009	0.989±0.003	0.988±0.002	0.969±0.006	0.961±0.010	0.988±0.002
Mammography	0.864±0.005	0.875±0.003	0.874±0.005	0.881±0.003	0.866±0.006	0.874±0.005	0.874±0.004
Musk	0.315±0.033	0.456±0.020	0.356±0.028	0.285±0.020	0.520±0.018	0.366±0.027	0.463±0.023
PumpSensor	0.956±0.011	0.963±0.007	0.956±0.009	0.956±0.007	0.954±0.011	0.972±0.008	0.945±0.012
Shuttle	0.997±0.000	0.997±0.000	0.997±0.000	0.998±0.000	0.996±0.000	0.998±0.000	0.996±0.000
SMTP	0.863±0.009	0.841±0.008	0.876±0.007	0.867±0.005	0.855±0.006	0.860±0.005	0.850±0.005

Table 2: Average AUC scores of 20 runs, each tree built with a 20 depth limit.

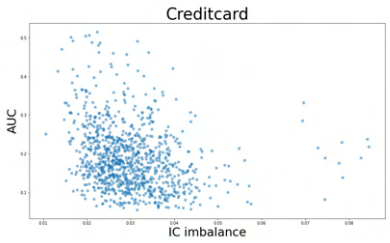
Data set	Criteria [AUC] - mean ± std						
	IF	I2 Top	I2 Bot	IC Top	IC Bot	WAVG Top	WAVG Bot
CreditCard	0.949±0.002	0.949±0.002	0.947±0.003	0.946±0.003	0.949±0.002	0.947±0.003	0.949±0.002
ForestCover	0.853±0.022	0.913±0.011	0.784±0.028	0.788±0.030	0.926±0.013	0.866±0.020	0.873±0.016
HTTP	0.997±0.002	0.993±0.001	0.989±0.004	0.986±0.005	0.994±0.001	0.992±0.005	0.993±0.001
Mammography	0.864±0.005	0.878±0.003	0.865±0.007	0.866±0.006	0.879±0.004	0.878±0.005	0.870±0.006
Musk	0.315±0.033	0.251±0.025	0.463±0.027	0.441±0.031	0.270±0.027	0.339±0.030	0.278±0.023
PumpSensor	0.956±0.011	0.972±0.008	0.932±0.014	0.933±0.015	0.978±0.007	0.969±0.010	0.950±0.008
Shuttle	0.997±0.000	0.999±0.000	0.995±0.001	0.995±0.000	0.999±0.000	0.997±0.000	0.998±0.000
SMTP	0.863±0.009	0.870±0.006	0.845±0.010	0.847±0.010	0.866±0.007	0.851±0.007	0.872±0.007

Table 3: Average AUC scores of 20 runs, each tree built with a 7 depth limit.

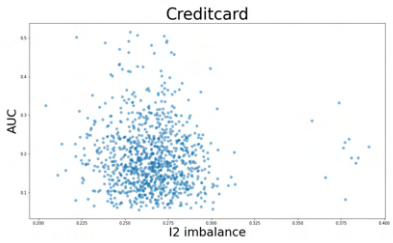
Upon further investigation of the relations of the metrics described in 3.5 it has become clear that it is possible to significantly enhance the performance of the Isolation Forest algorithm by examining the structures of the Isolation Trees. This property seems to be data-set dependent as well as which metric would yield the optimal results. A huge advantage of the described metric is that they do not rely on the actual samples of the data-set, they only use the shape of the trees to predict how well could they contribute to an accurate anomaly-score. Please refer to Figures 8 - 15 to see how the AUC-score of each tree's anomaly-score relates to their metric-scores. Once again, we include the results of the same experiment with different depth limits (128, 20 and 7).



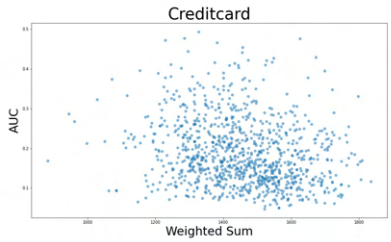
(a) 128 depth, WS



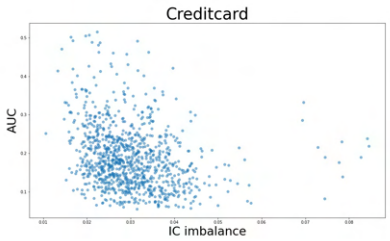
(b) 128 depth, IC



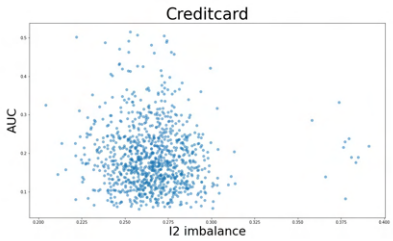
(c) 128 depth, I2



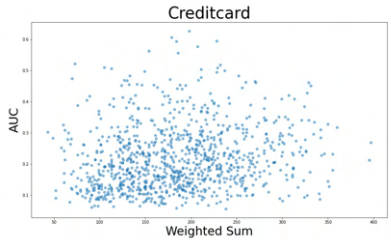
(d) 20 depth, WS



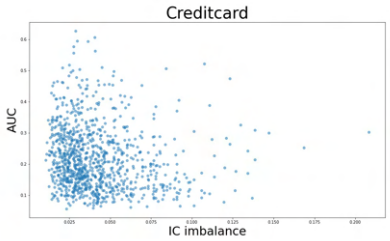
(e) 20 depth, IC



(f) 20 depth, I2



(g) 7 depth, WS

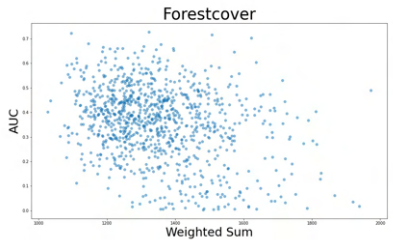


(h) 7 depth, IC

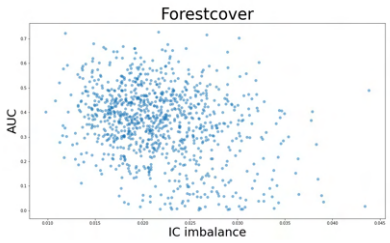
(i) 7 depth, I2

Figure 8: CreditCard data-set: AUC of each trees prediction in function of each metric

As we can see on Figure 8, the highest AUC achieving trees tend to have lower Colless' imbalance score. Unfortunately all the metircs and IF achieved a similar and quite high AUC with all depth limit settings and the different scores are within each others confidence interval, which means that we can not show the effects of our modification on the CreditCard data-set.

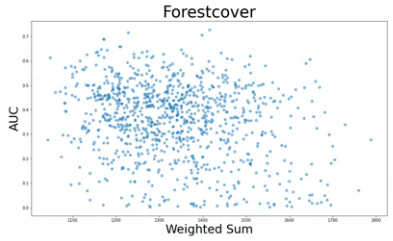


(a) 128 depth, WS

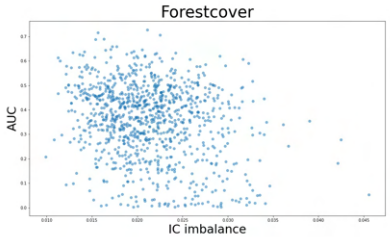


(b) 128 depth, IC

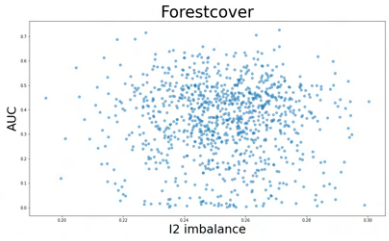
(c) 128 depth, I2



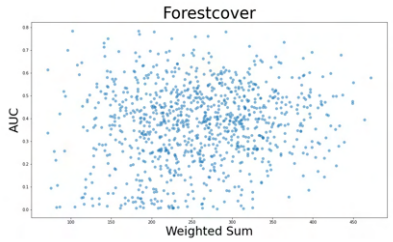
(d) 20 depth, WS



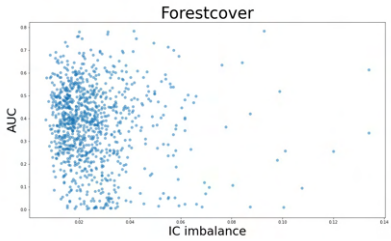
(e) 20 depth, IC



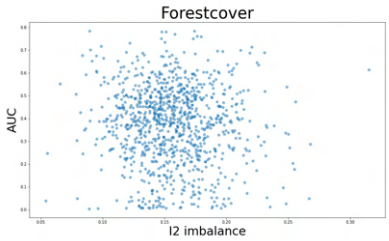
(f) 20 depth, I2



(g) 7 depth, WS



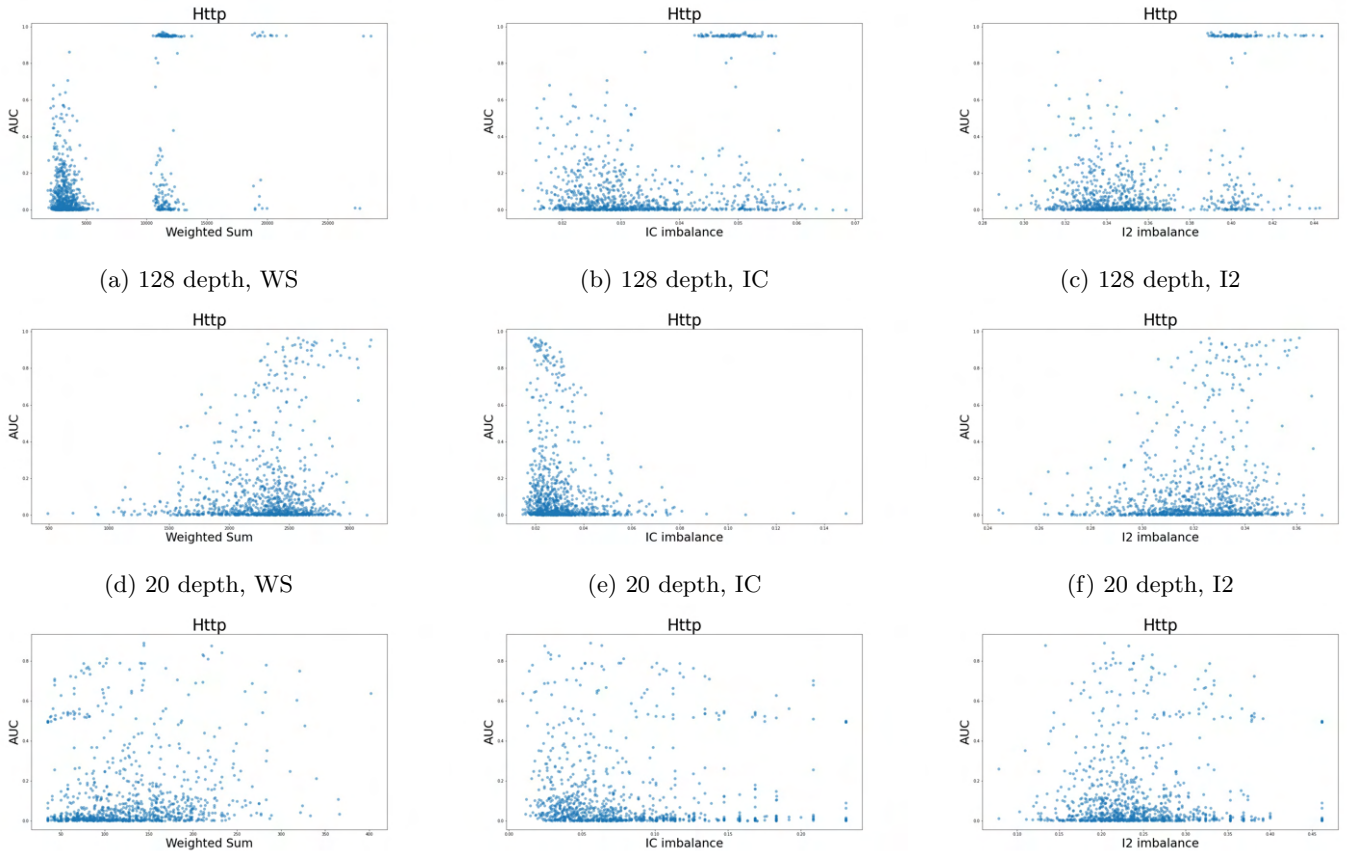
(h) 7 depth, IC



(i) 7 depth, I2

Figure 9: ForestCover data-set: AUC of each trees prediction in function of each metric

On 9 we can see that at 7 depth limit the lower Colless' imbalnce scoring trees tend to have a higher AUC score. If we check the corresponding, 7-depth Table 3 we can see that there is a significant improvement compared to IF, and there confidence interval do not overlap.



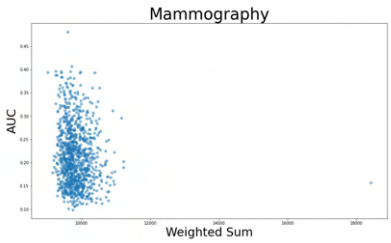
(a) 128 depth, WS (b) 128 depth, IC (c) 128 depth, I2

(d) 20 depth, WS (e) 20 depth, IC (f) 20 depth, I2

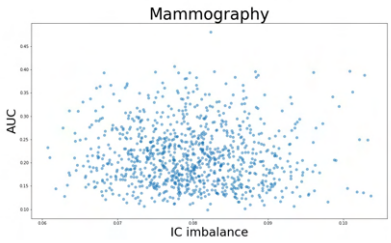
(g) 7 depth, WS (h) 7 depth, IC (i) 7 depth, I2

Figure 10: HTTP data-set: AUC of each trees prediction in function of each metric

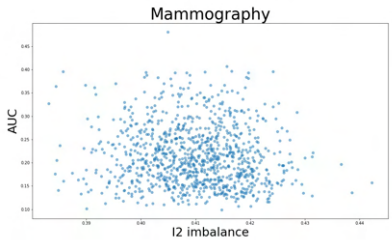
On Figure 10 we can see an interesting behaviour: the plot seems to have the data organised in columns. This is due to the implementation of the algorithm: if a node receives a set of same samples during the fit phase, it will not be able to split them, but it will try. As a result two new nodes are constructed, one of them receives no data, the other one receives the whole set of samples and the cycle continues until the depth limit is reached. This is however no concern due to the way the anomaly-score is formulated and the rest of the forest is capable of compensating the possible miss-classifications of anomalous samples (benefit of ensemble learning).



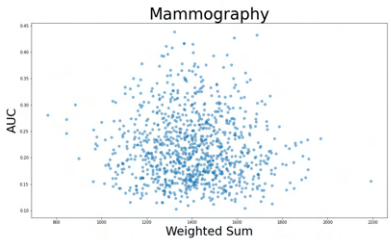
(a) 128 depth, WS



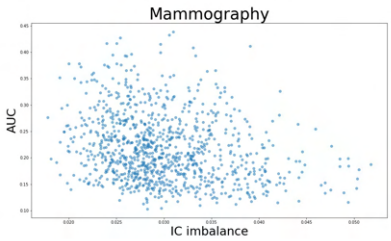
(b) 128 depth, IC



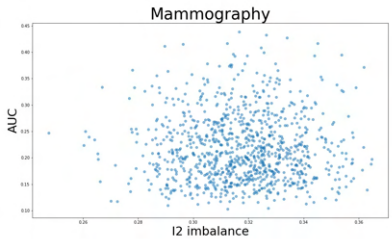
(c) 128 depth, I2



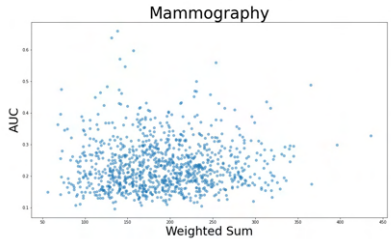
(d) 20 depth, WS



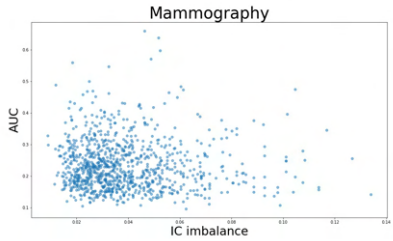
(e) 20 depth, IC



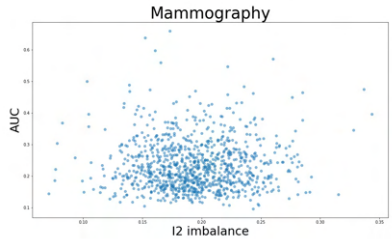
(f) 20 depth, I2



(g) 7 depth, WS



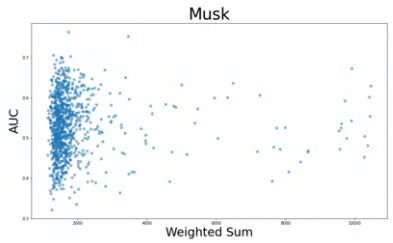
(h) 7 depth, IC



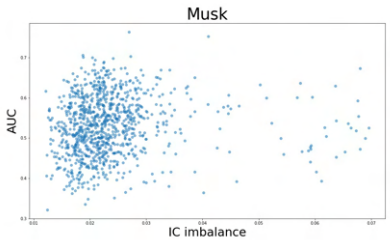
(i) 7 depth, I2

Figure 11: Mammography data-set: AUC of each trees prediction in function of each metric

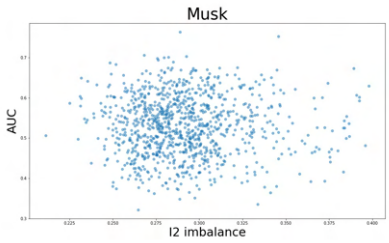
On Figure 11 we can see another example of a data-set where there is not really an evident connection between the AUC- and metric-scores of the trees. This resulted in similar performance between the metrics and IF.



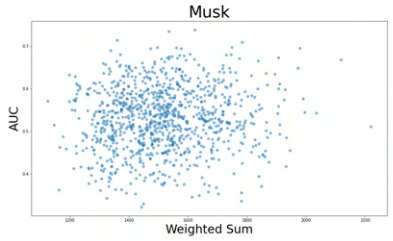
(a) 128 depth, WS



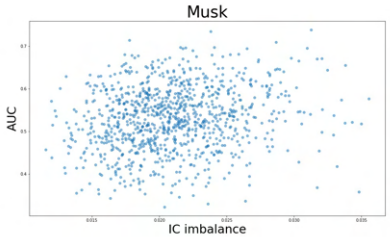
(b) 128 depth, IC



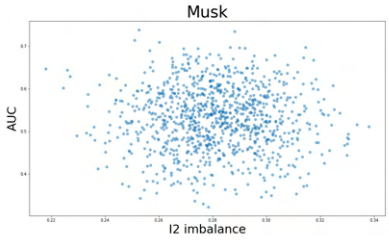
(c) 128 depth, I2



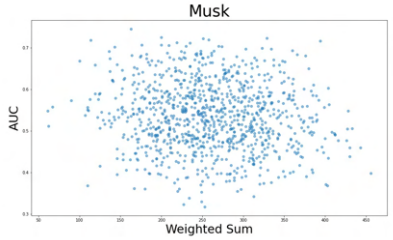
(d) 20 depth, WS



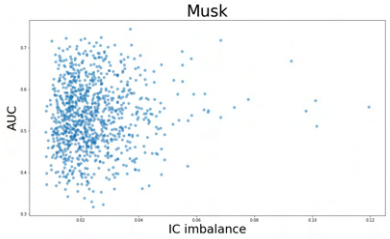
(e) 20 depth, IC



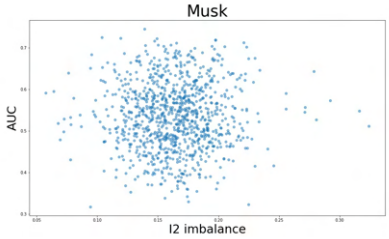
(f) 20 depth, I2



(g) 7 depth, WS



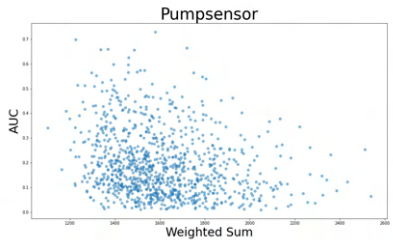
(h) 7 depth, IC



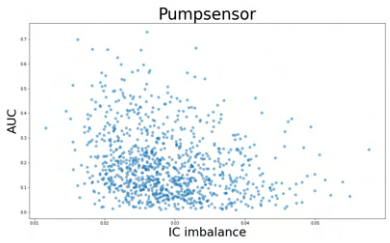
(i) 7 depth, I2

Figure 12: Musk data-set: AUC of each trees prediction in function of each metric

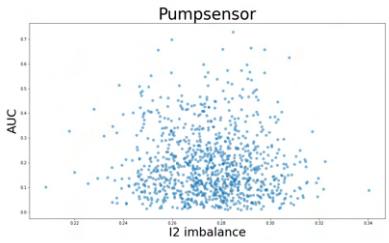
On Figure 12 it is difficult to see the correlation between the AUC-scores and the metric-scores, but as evident from the above tables the right metric selected trees kept significantly outperforming the IF algorithm.



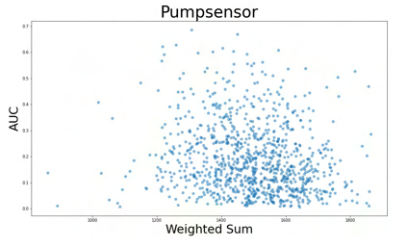
(a) 128 depth, WS



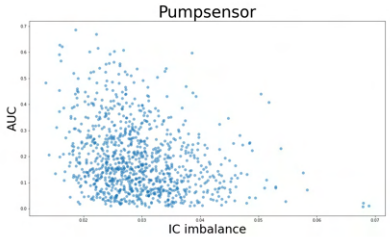
(b) 128 depth, IC



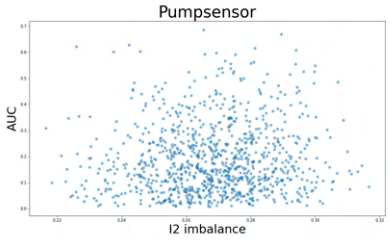
(c) 128 depth, I2



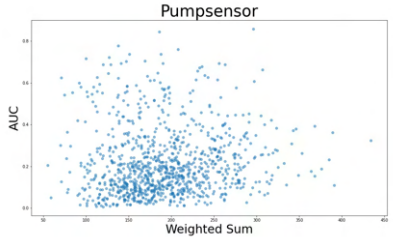
(d) 20 depth, WS



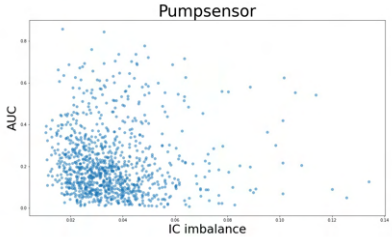
(e) 20 depth, IC



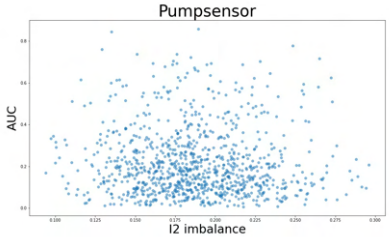
(f) 20 depth, I2



(g) 7 depth, WS



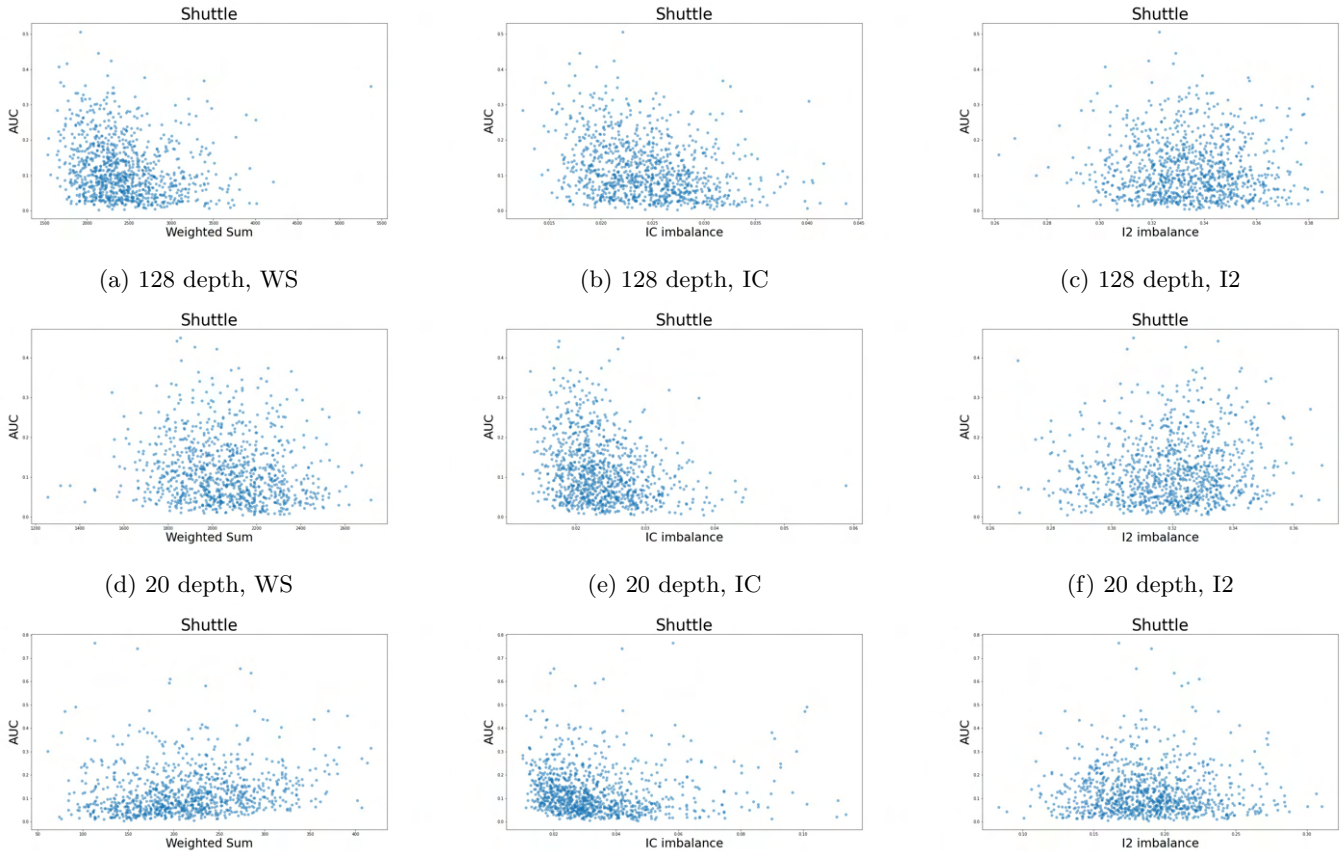
(h) 7 depth, IC



(i) 7 depth, I2

Figure 13: PumpSensor data-set: AUC of each trees prediction in function of each metric

On Figure 13 we can discover a negative correlation between AUCs and ICs of trees. The negative correlation is most clear on the 7 depth limit experiment, which is confirmed by the above tables, as this combination consistently reaches an almost perfect prediction (1.0 AUC).



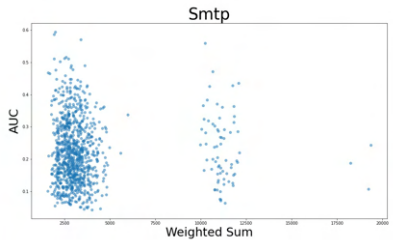
(a) 128 depth, WS (b) 128 depth, IC (c) 128 depth, I2

(d) 20 depth, WS (e) 20 depth, IC (f) 20 depth, I2

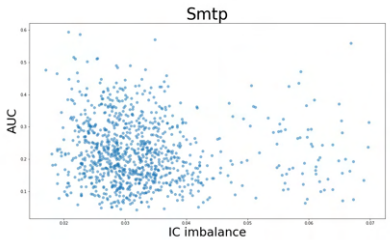
(g) 7 depth, WS (h) 7 depth, IC (i) 7 depth, I2

Figure 14: Shuttle data-set: AUC of each trees prediction in function of each metric

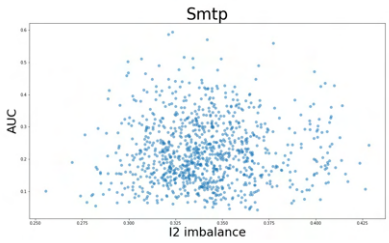
On Figure 14 we can discover a similar negative correlation between the AUCs and ICs of the trees and although the corresponding entry in Table 3 does show that it performed consistently well (1.0 AUC) so did the other variants (they did fell behind a bit).



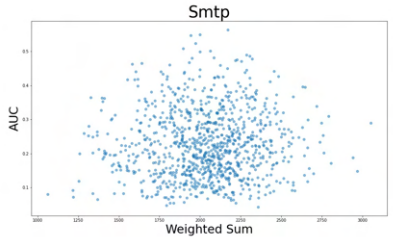
(a) 128 depth, WS



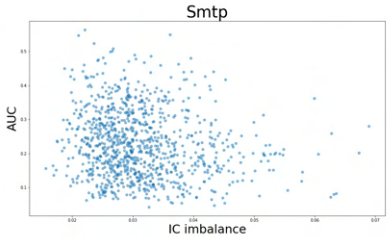
(b) 128 depth, IC



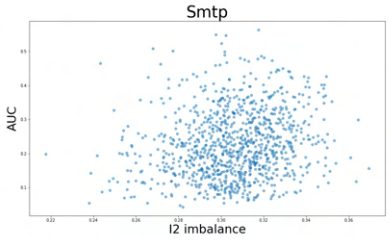
(c) 128 depth, I2



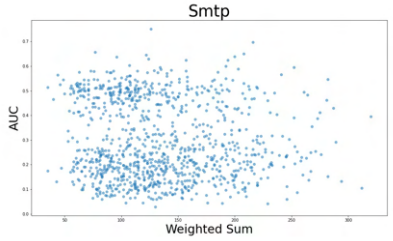
(d) 20 depth, WS



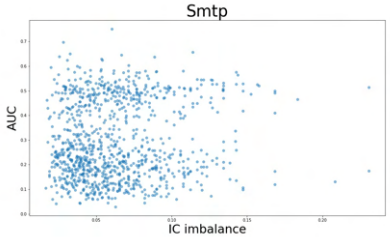
(e) 20 depth, IC



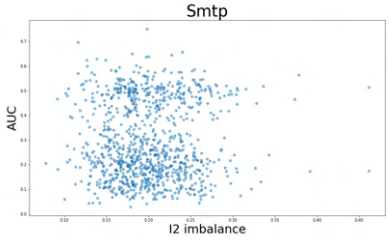
(f) 20 depth, I2



(g) 7 depth, WS



(h) 7 depth, IC



(i) 7 depth, I2

Figure 15: SMTP data-set: AUC of each trees prediction in function of each metric

On 15 we can once again discover the same phenomena as with the HTTP 10 data-set: column organised data at the largest depth limit, the reason is once again the several equal samples. We can also see a weak negative correlation between the AUCs and ICs of the trees on the larger depth limits.

4 Conclusion and Future Work

We have investigated 3 metrics describing the structures of trees and their correlation with the quality of anomaly-score they produced via their ROC-AUC-scores. We have shown that it is worth investigating the structure of trees as their right metric based selection can lead to consistently and significantly higher overall AUC scores, enhancing the predictive power of the Isolation Forest algorithm. Unfortunately none of the investigated metrics could consistently outperform the others, meaning that the best score yielding metric has to be found individually for each data-set and depth limit setting.

In the future we are planning on investigating other metrics with the goal of finding a universally applicable metric that can be applied with success on any Isolation Forest and data-set and can consistently increase the quality of anomaly-scores produced.

References

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

[2] S. Hariri, M. Carrasco Kind, and R. J. Brunner. Extended isolation forest. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2019.

[3] Linkedin isolation forest. https://github.com/linkedin/isolation-forest. Accessed: 2022-09-11.

[4] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[5] Shebuti Rayana. Odds library, 2016.

[6] Credit card fraud detection. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. Accessed: 2022-10-10.

[7] Pump sensor data. https://www.kaggle.com/datasets/nphantawee/pump-sensor-data. Accessed: 2022-10-10.

[8] OpenMP Architecture Review Board. OpenMP application program interface version 3.0, May 2008.

[9] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2011.

[10] Online graphviz. https://dreampuf.github.io/GraphvizOnline/. Accessed: 2022-10-27.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[12] Colless D.H. Review of] phylogenetics: the theory and practice of phylogenetic systematics. *Syst. Zool*, 31:100–104, 1982.

[13] Mooers AO and Heard SB. Inferring evolutionary process from phylogenetic tree shape. *Quarterly Review of Biology*, 72:31–54, 1997.