PROGRESS REPORT

November 8, 2020

Vibekananda Dutta

Contents

A Modified Isolation Forest Based Anomaly Detection in IoT Dataset

0.0.1 Introduction

The work presented in this report deals with anomaly detection algorithms with a focus on the modified Isolation Forest algorithm. Modified Isolation Forest (mIF) generalizes its predecessor algorithm, the Isolation Forest (IF). The original Isolation Forest algorithm brings a brand new form of detection, although the algorithm suffers from bias coming from tree branching. Improvement of the algorithm removes the bias by adjusting the branching, and the original algorithm becomes just a special case. Modified Isolation Forest is implemented into the IoT-23 open-source dataset platform.

0.0.1.1 Contributions

The work deals with the modified Isolation Forest (mIF). mIF implements a new technique for branching isolation tree, which leads to the improvement of the anomaly score distribution and mitigates the bias. The mIF uses a new hyperparameter "extension level" in order to allow for ordinary IF to run inside the mIF algorithm. The mIF algorithm was developed in Python that would be suitable for processing of big data.

The motivation is to create well tested, and fully supported implementation of modfied Isolation Forest with the possibility to scale the algorithm for big data. The mIF algorithm was implement a study of anomaly score given by IF on IoT-23 data. The following contributions are as follows:

- According to the nature of the dataset IoT-23, initially, the pre-processing approach is employed to process and normalized the data.
- A sparse deep AutoEncoder is used for feature engineering followed by a modified Isolation Forest (mIF).
- The employed mIF is an algorithm for unsupervised anomaly detection based on the Isolation Forest algorithm.
- mIF's branching method allows for slicing of the data by using hyperplanes with random slopes comparing original IF branching provides slicing only parallel to one of the axes.

- Experimental evaluation of the mIF on IoT-23 dataset and compared to the state-of-the-art IF with different settings.
- Scalability Test of the model performance shows the proposed mIF reduces the computational complexity w.r.t. original IF.

0.0.2 Methods and Material

0.0.2.1 Distributions of Applied Data in Training and Testing

In this work, I introduced nature of training and testing data categorically. It is worth mentioning that I did not employed *test-train-split* provided sklearn libraries. The selection and pre-processing of the train and test data was implemented individually.

For the training set, I consider the set of captures from IoT-23: Honeypot-(4,5,7), Malware-(20, 42, 17, 35). The distribution of the test set is illustrates in Fig 1.

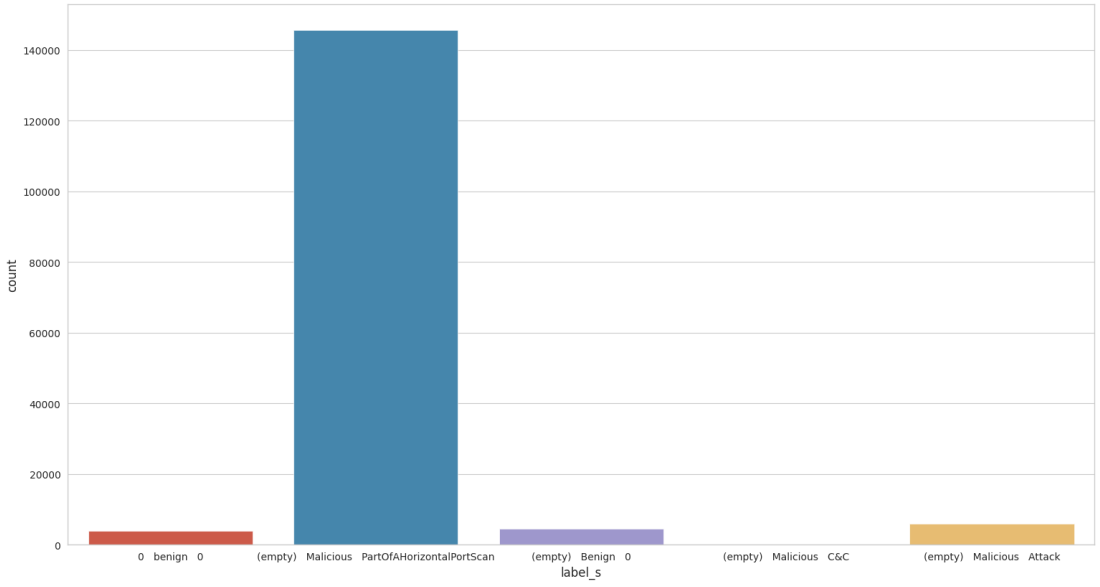


Figure 1: Distribution of the training set

Moreover, the test set, I consider the set of captures from IoT-23 dataset as follows: Malware (1, 34, 43, 35, 17, 36). The distributions of the testing set is visualize in Fig 2.

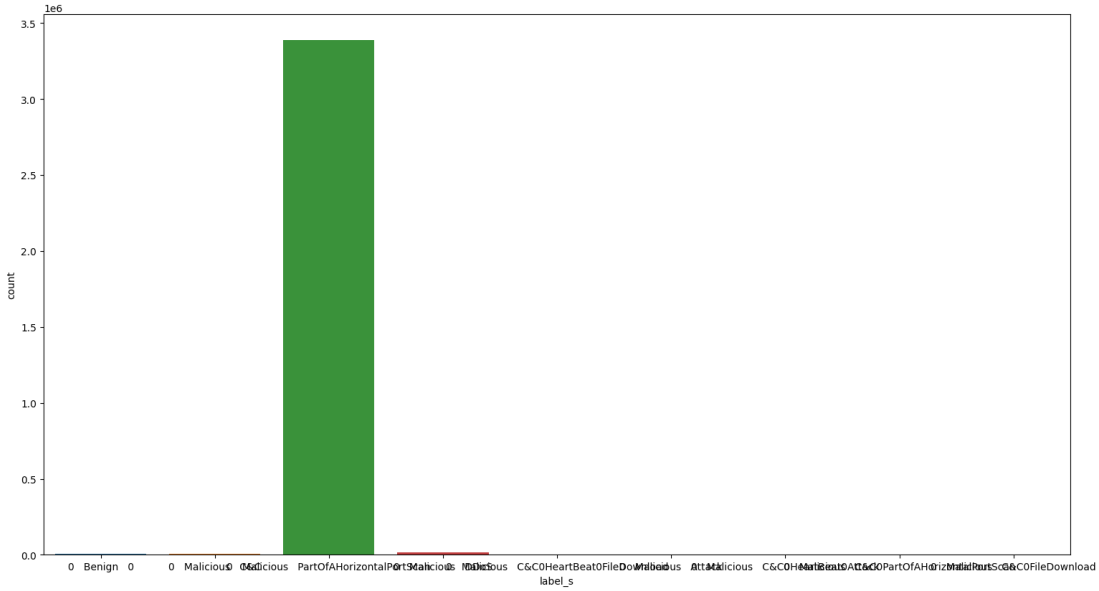


Figure 2: Distribution of the test set

0.0.3 Conceptual Framework of Training and Classification Framework

This section defines the current capabilities of the implemented Isolation Forest and discusses the algorithm's potential. The ability to process big data cannot be affected in any scenario because the IoT-23 data structures are used.

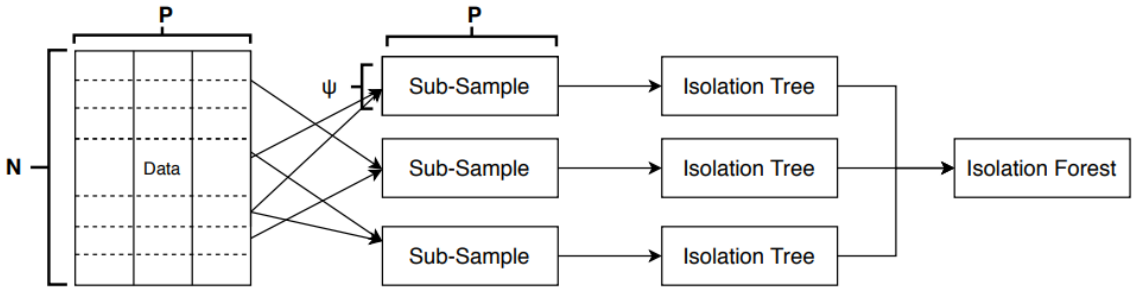


Figure 3: Distribution of the test set

In the training stage process the reconstructed data after feature engineering (based on sparse deep AE) are used as an input vector. mIF builds independent trees with a random sub-sample of data. Each tree can be built separately with reference to a

small amount of data. The Binary Search Tree (BST) building can also be parallelized. The current Isolation Forest implementation uses a structure called DTree developed for distributed Random Forest algorithms. The DTree structure is responsible for computing performance in current Isolation Forest implementation (see Fig 3).

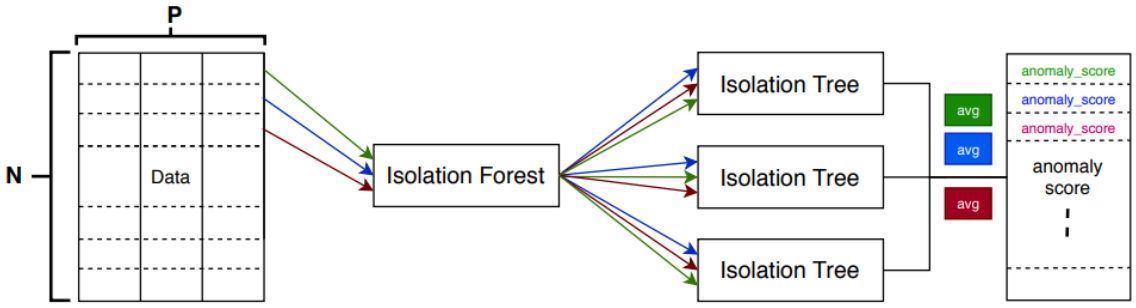


Figure 4: Distribution of the test set

In the classification stage, the input for the Forest are the rows of reconstructed data. The row h(x) is computed in each tree and the average h(x) is the input for the anomaly score. Figure 4 depicts the evaluation stage. Path length in each tree was computed independently. Map/Reduce was used for this. The Isolation Tree can compute the h(x) in the Map method, and the Reduce method adds the h(x) in the correct output row. Second Map/Reduce task computes the average and anomaly score for each row in the output simultaneously.

The Extended Isolation Forest model has following hyperparameters:

- *ntrees*: number of trees in ensemble
- *sample-size* : sub sample size
- *Ex-Level*: value in range [0, P - 1]; where P is the number of features.

0.0.4 Anomaly score computing

The anomaly score is interpreted as follows:

- If instances return very close to 1, then they are identify as anomalies.
- If instances have much smaller than 0, then they can be quite safely regarded as normal instances.

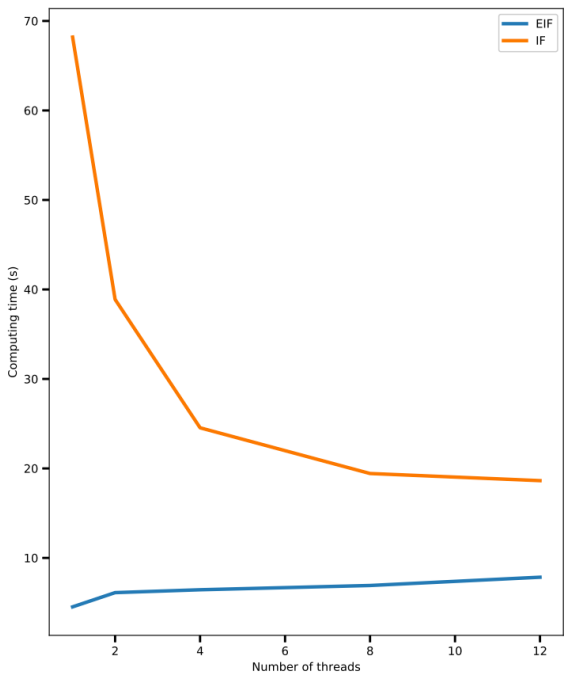
The defined anomaly score is computed as,

$$s(x, N) = 2^{\frac{-E(h(x))}{N}} \tag{1}$$

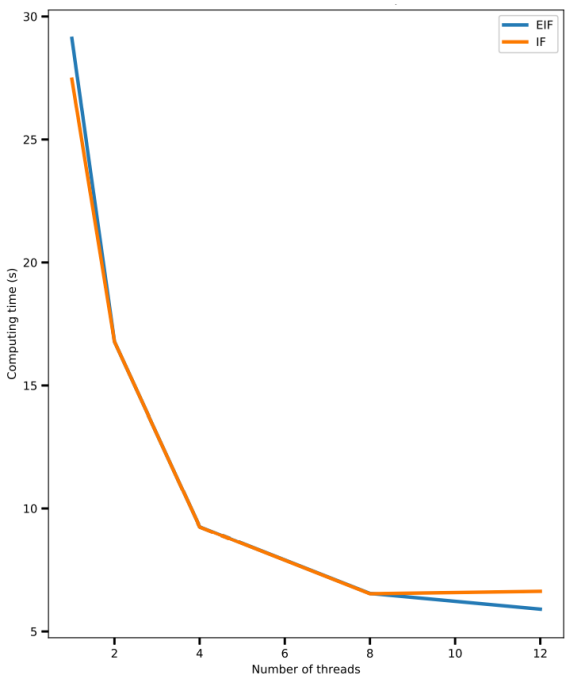
where, x represents any row in the data, N illustrates total number of rows in the data, and finally, $E(h(x))$ defines mean $h(x)$ (i.e., height of tree) in ensemble.

0.0.5 Performance benchmark with four approaches on IoT-2020 data

0.0.5.1 Scalability Test of the model performance with Computing time



(a) Training time with sub-sample size 256



(b) Classification time with sub-sample size 256

Figure 5: Computing time with the number of threads. The following parameters are: sun-sample-256, ntree-1000

0.0.5.2 Anomaly score of the data samples

Note:

- The distribution of **anomaly scores**: by definition, anomalies are those that occur

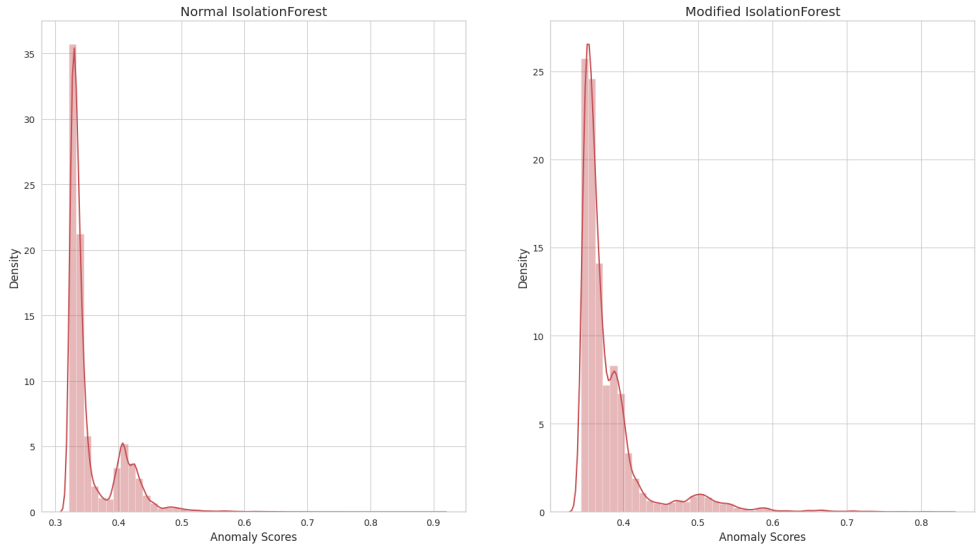


Figure 6: Distribution of anomaly scores for sub-sample 256 and ntree 500

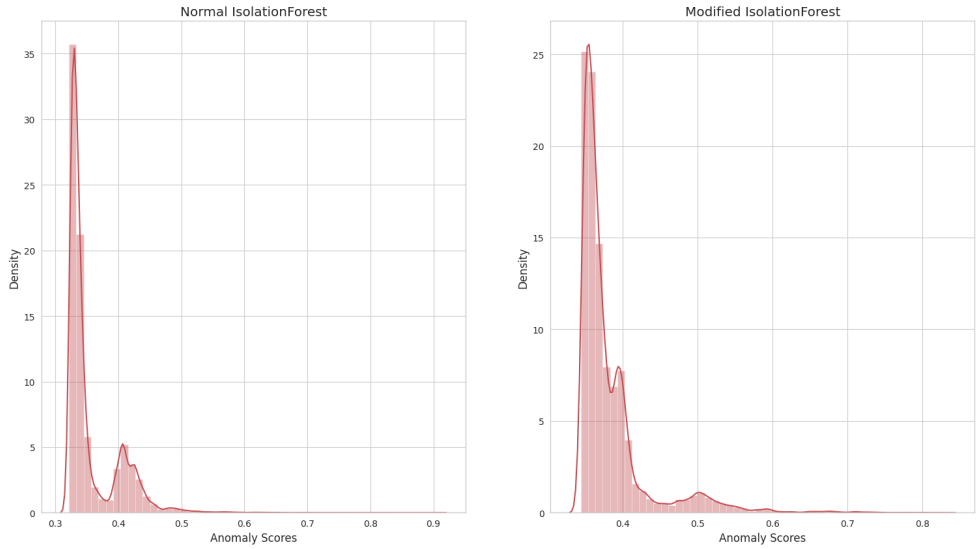


Figure 7: Distribution of anomaly scores for sub-sample 256 and ntree 1000

less frequently. So it makes sense that the number of points with higher anomaly scores reduces as the score increases.

- The distribution of **forest visualization**: by passing a single anomalous and a single nominal point through the forest. Each radial line in the plots below corresponds to

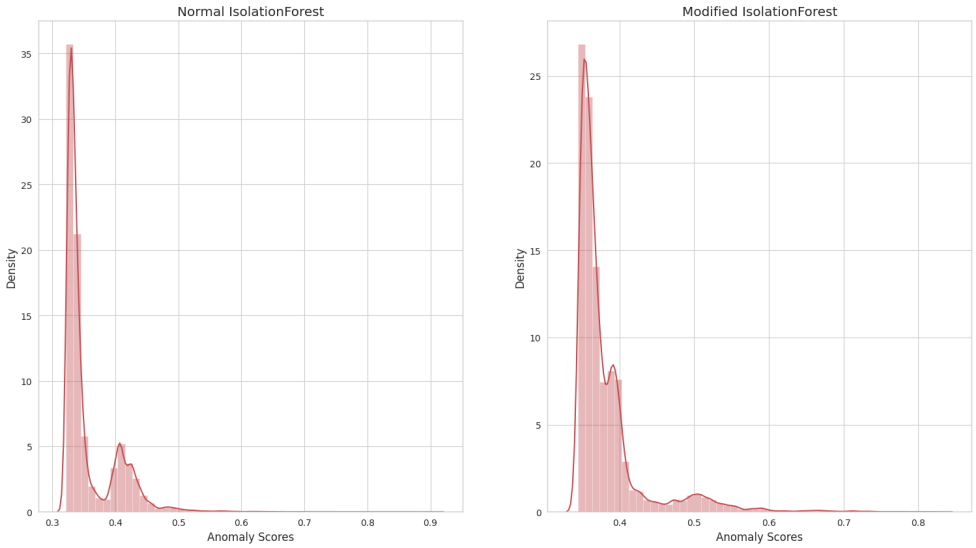


Figure 8: Distribution of anomaly scores for sub-sample 256 and ntree 1000

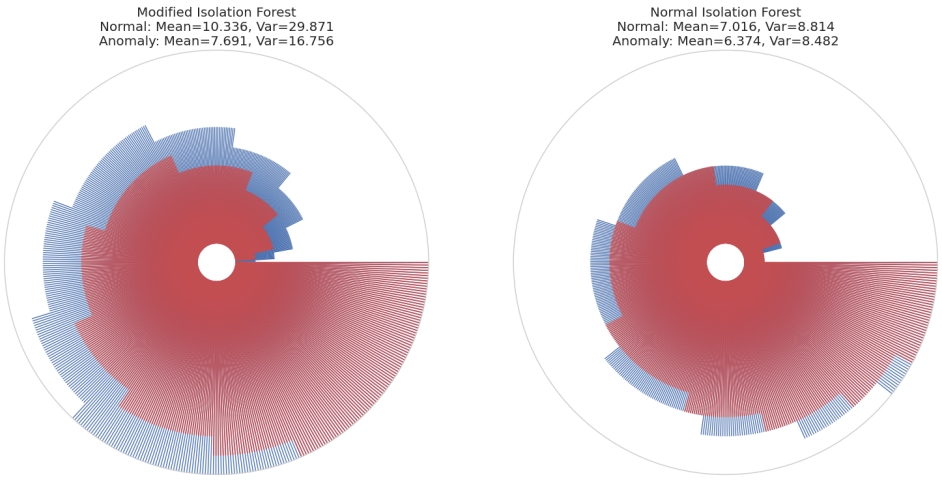


Figure 9: Forest visualization of anomaly scores for sub-sample 256 and ntree 500

a tree. The Gray circle is the depth limit each tree can reach. Blue lines show the depth the nominal point reached on each tree, while the red lines show the depth each anomalous point reaches for each tree. This visualization provides a quick view of how on average anomalous points reach much smaller depths than the nominal

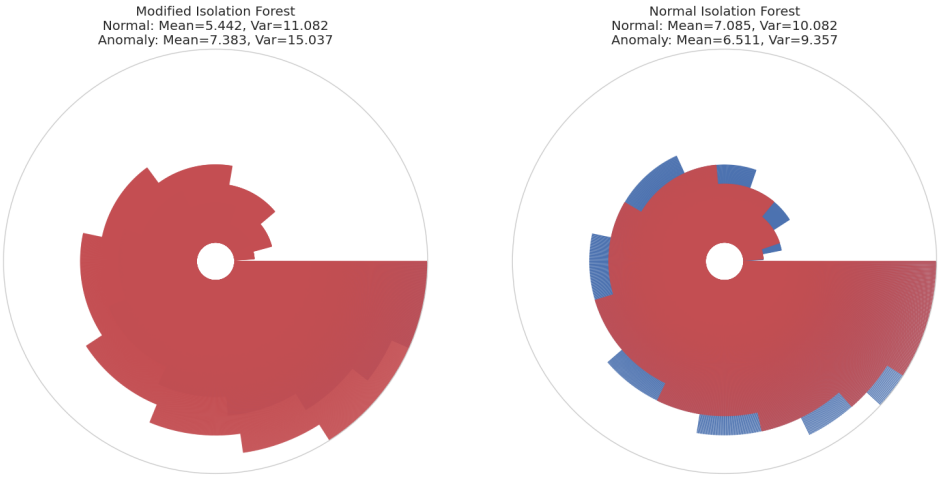


Figure 10: Forest visualization of anomaly scores for sub-sample 256 and ntree 1000

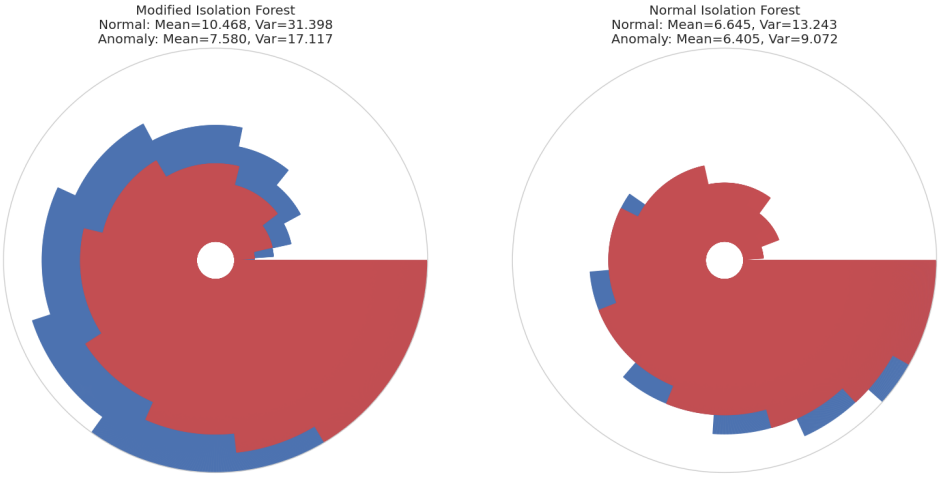


Figure 11: Forest visualization of anomaly scores for sub-sample 256 and ntree 2000

points.

- The best results achieved with **number of tress 1000** and **sub-sample 256** for **mIF** and for **IF**, the best results achieved with **number of tress 2000** and **sub-sample 256**.