# Evaluating the Sensitivity of Isolation Forest Parameters in Anomaly Detection

DT Nicolay 26296918
*Computer Science Division*
*Stellenbosch University*
Stellenbosch, South Africa
26296918@sun.ac.za

***Abstract*—TODO**

***Index Terms*—TODO**

## I. Introduction

## II. Background

### A. Isolation Forests

The majority of existing model-based approaches to anomaly detection construct a profile of normal instances, then they identify instances that do not conform to this normal profile as anomalies [1]. Liu et al. (2008) proposed a fundamentally different model-based method that explicitly isolates anomalies instead of profiles normal points, Isolation Forests. Here, isolation refers to separating an instance from the rest of the instances. This is ideal for an anomaly detection problem context, since anomalies are by nature sparse and diverse.

Normal profile methods, since not optimised for anomaly detection, often lead to too many false positives or little to no anomalies detected at all. These methods are also constrained to low dimensional data and small data size since they require significant computational power. Isolation Forests on the other hand take advantage of anomaly datasets consisting of fewer observations for the target class, and anomalies having feature values distinct from the rest of the data. Due to the nature of anomaly observations, they are isolated closer to the root of the tree. This forms the foundation of Isolation Trees.

Isolation Forests involve an ensemble of isolation trees where the predicted anomalies are the observations with the shortest average paths across the trees. Using this model for anomaly detection involves two stages. The first stage constructs isolation trees by recursively partitioning. This training stage is described in Algorithm 1 and Algorithm 2 [2].

**Algorithm 1** iForest($X$, $t$, $\psi$)

**Require:** $X$ – input data, $t$ – number of trees, $\psi$ – subsampling size
**Ensure:** A set of $t$ isolation trees
Initialize Forest $\leftarrow \emptyset$
**for** $i = 1$ to $t$ **do**
  $X' \leftarrow$ sample$(X, \psi)$
  Forest $\leftarrow$ Forest $\cup$ iTree$(X')$
**end for**
**return** Forest

The iTree algorithm recursively partitions the dataset by randomly selecting a feature and a random split value until the data can no longer be divided. The resulting tree structure isolates individual points, where anomalies tend to have shorter average path lengths because they are easier to isolate.

**Algorithm 2** iTree($X'$)

**Require:** $X'$ – input data
**Ensure:** an isolation tree
**if** $X'$ cannot be divided **then return** exNode{Size $\leftarrow |X'|$}
**else**
  Let $Q$ be the list of attributes in $X'$
  Randomly select an attribute $q \in Q$
  Randomly select a split point $p$ between the max and min values of attribute $q$ in $X'$
  $X_l \leftarrow$ filter$(X', q < p)$
  $X_r \leftarrow$ filter$(X', q \geq p)$ **return** inNode{
   Left $\leftarrow$ iTree$(X_l)$,
   Right $\leftarrow$ iTree$(X_r)$,
   SplitAtt $\leftarrow q$,
   SplitValue $\leftarrow p$}
**end if**

### B. Control Parameters

There are five control parameters to consider namely: the number of estimators, the maximum samples, the contamination, the maximum features, and whether to first bootstrap sample.

The number of trees parameter determines the number of base estimators in the ensemble. This parameter is denoted

by $t$ in Algorithm 1. The performance of Isolation Forests converges quickly with a very small number of trees [1].

The maximum samples describes the number of sample observations to draw from the training data for each base estimator. The subsampling size is denoted by $\psi$ in Algorithm 1. Only a small sampling size is required to achieve high detection performance with high efficiency [1]. Setting $\psi$ to 256 often suffices for anomaly detection across a wide range of data [2].

Contamination refers to the proportion of anomalies present in the dataset. It is defined as the number of anomalies divided by total number of observations. When set to a specific value between 0 and 0.5, it determines the threshold on the anomaly scores such that approximately that fraction of the training samples are labelled as outliers. In the original paper, the threshold is automatically fixed at an offset of -0.5, following the original Isolation Forest formulation, where inliers typically yield scores near 0 and outliers near -1, allowing the model to separate them without prior knowledge of the true contamination level. Since this parameter directly influences the threshold for anomaly detection, mis-tuning will directly increase false positives or false negative rates.

The maximum features describes how many features are selected at random before tree construction to train each base estimator. Introducing randomness can help with with high-dimensional data, but may increase instability if set too low.

The bootstrap parameter controls the manner in which sample observations are drawn for each tree. This determines whether sampling is done with or without replacement. That is, sub-sample $X'$ is randomly sampled with or without replacement in Algorithm 1 from $X$. Bootstrap resampling can lead to marginal improvements across classification metrics [3].

### C. Evaluation Stage

For this stage, a single path length $h(\boldsymbol{x})$ is computed by counting the number of edges from the root node to an external node as an instance $\boldsymbol{x}$ traverse each iTree. There is a predefined height limit, *hlim*, that when reached, the algorithm returns the value $e$ plus an adjustment $c(Size)$. This process is described in Algorithm 3 [2]. The worst case time complexity of the evaluation stage for a dataset size $n$ is $O(nt\psi)$.

**Algorithm 3** PathLength($\boldsymbol{x}, T, h_{lim}, e$)

**Require:** $\boldsymbol{x}$ – an instance; $T$ – an iTree; $h_{lim}$ – height limit; $e$ – current path length (initialized to 0 when first called)
**Ensure:** Path length of $\boldsymbol{x}$

```
if T is an external node or e ≥ h_lim then
    return e + c(T.size) ▷ c(·) is defined in Equation (1)
else
    a ← T.splitAtt
    if x_a < T.splitValue then
        return PathLength(x, T.left, h_lim, e + 1)
    else
        return PathLength(x, T.right, h_lim, e + 1)
    end if
end if
```

The anomaly score $s(\boldsymbol{x}, \psi)$ for instance $\boldsymbol{x}$ is computed using

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

where $c(\psi)$ is the average path length of unsuccessful searches and $E(h(\boldsymbol{x}))$ is the average depth from the forest of isolation trees. If the score of an observation $s$ is close to 1, the observation is an anomaly. If an observation has a score much smaller than 0.5, then it is regarded as a normal instance. However, if all observations return a score of approximately 0.5, then there are no distinct anomalies.

## III. Implementation

### A. Isolation Forest Algorithm

The Scikit-learn library [4] implementation of the Isolation Forest algorithm described in the background section is used to detect anomalies across three varied datasets. This implementation returns the anomaly score for each observation. The path length to an observation in the a tree, averaged over a forest of isolation trees, is used as a measure of normality and the decision function. When the Isolation Forest collectively produces shorter path length for a particular observation, it is likely to be an anomaly. The maximum depth of each tree is $\lceil log_2(n) \rceil$ where $n$ is the number of samples used to build the tree.

### B. Experimental Setup

A random seed of 12 is used for all stochastic procedures to ensure reproducibility. All tests are conducted on a Fedora Linux 41 system with a 13th Gen Intel Core i3-13100 CPU with 16 GiB of memory. The computationally significant portions of the algorithm are running the algorithm with a large ensemble size and also a large parameter grid for some parameter combinations. The code for this paper is available on GitHub [5].

## IV. Empirical Process

### A. Datasets

### B. Evaluation Metrics

### C. Experimental Design

#### 1) Single-Parameter Analysis:

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

[2] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, Mar. 2012. [Online]. Available: https://doi.org/10.1145/2133360.2133363

[3] H. Choi and K. Jung, "Impact of data distribution and bootstrap setting on anomaly detection using isolation forest in process quality control," *Entropy*, vol. 27, no. 7, p. 761, July 2025.

[4] 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*, vol. 12, pp. 2825–2830, 2011.

[5] D. Nicolay, "Ml441 assignments," https://github.com/Voltzz9/ml441_assignments, 2025.