

Unsupervised Outlier Detection with Isolation Forest

Isolation forest - an unsupervised anomaly detection algorithm that can detect outliers in a data set with incredible speed.



Yenwee Lim

[Follow](#)

7 min read · Mar 17, 2022

 363

...

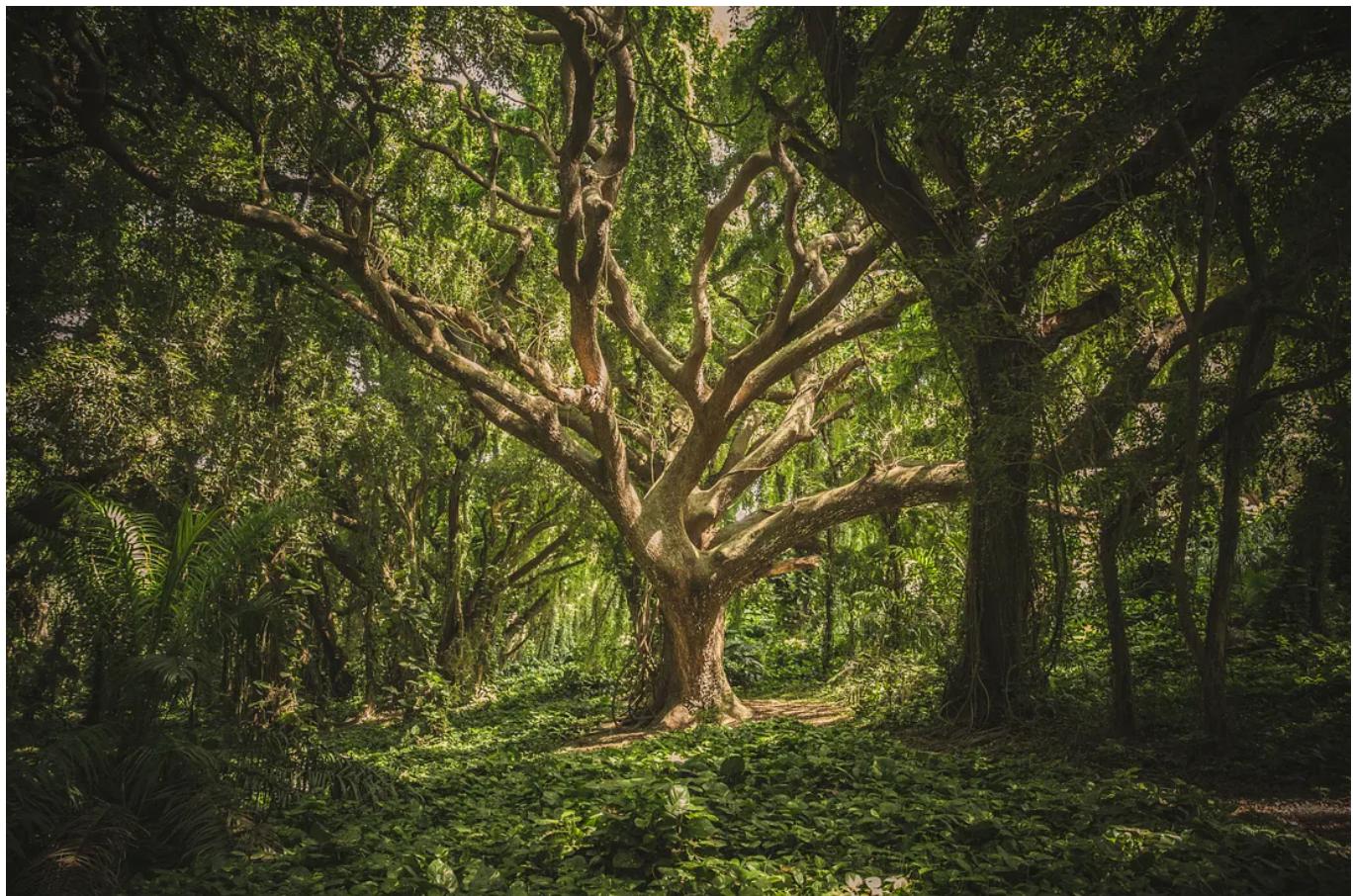


Photo by [veeterzy](#) on [Unsplash](#)

Isolation Forest is a simple yet incredible algorithm that is able to spot outliers or anomalies in a data set very quickly. I should say understanding this tool is a must for any aspiring data scientist. In this article, I will briefly go through the theories behind the algorithm and also its implementation.

Its Python implementation from *Scikit Learn* has been gaining tons of popularity due to its capabilities and ease of use. But before we jump right

into the implementation. It's always best practice for us to study about it's use cases and it's theory behind. However, feel free to skip straight below to the implementation part !

Some Theories

First off, what is an anomaly ?

An anomaly (outlier) can be described as a data point in the data set that differs **significantly** from the other data or observations. This can happen because of several reasons:

- An outlier may indicate **bad data** that is incorrect or an experiment may not have been run correctly.
- An outlier may be due to **random variation** or may indicate something **scientifically interesting**.



Photo by [Will Myers](#) on [Unsplash](#)

That is quite a simplified version of anomaly. Nevertheless, both events are something that data scientists would like to understand and further dive into.

Why anomaly detection?

The reason why we would like to dive deeper into anomalies is because these data points will either screw you up or allow you to identify something meaningful.

In the case of a simple linear regression, a bad outlier can increase the variance in our model and further reduce the power of our model to grasp onto the data. Outliers cause regression models (especially linear ones) to learn a skewed understanding towards the outlier.

Overview of How Isolation Forest Works

Traditionally, other methods have been trying to construct a **profile of normal data**, then further identify which data points that do not conform to the profile as anomalies.

The brilliant part of Isolation Forest is that it can directly detect anomalies using **isolation** (how far a data point is to the rest of the data). This means that the algorithm can run in a **linear time complexity** like other distance-related models such as K-Nearest Neighbors.

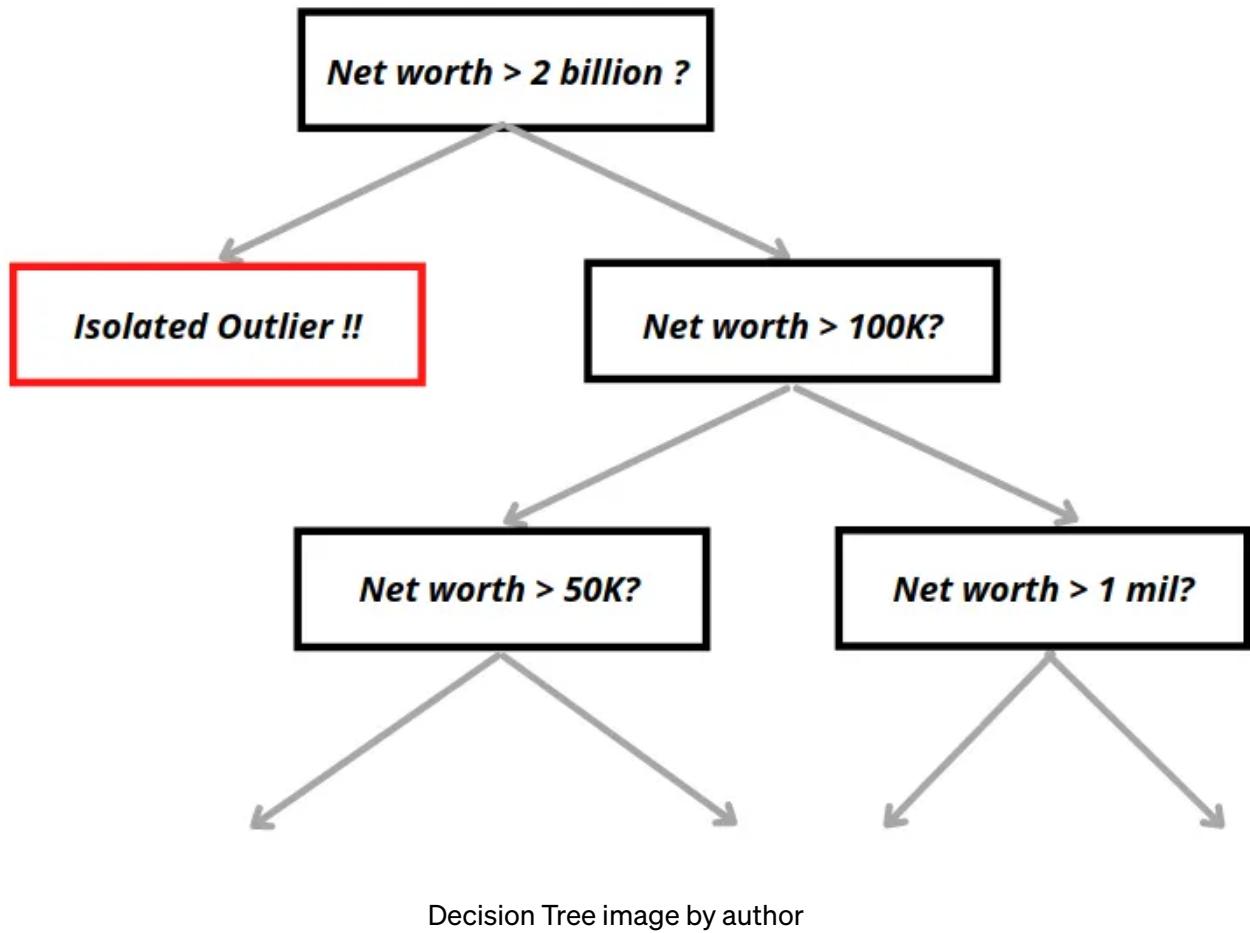
The algorithm works by pivoting on the most obvious attributes of an outlier:

- There will only be *few* outliers
- There outliers will be *different*

Isolation Forest does it by introducing (an ensemble of) **binary trees** that recursively generates **partitions** by randomly selecting a feature and then randomly selecting a split value for the feature. The partitioning process will continue until it separates all the data points from the rest of the samples.

Since only one feature is selected from an instance for each tree. We can say that the max depth of the decision tree is actually one. In fact, the base estimator of an Isolation Forest is actually an extremely random decision tree (ExtraTrees) on various subsets of data.

An example of a single tree in an Isolation Forest can be seen below.



Given the attributes of an outlier mentioned above, we can observe that an outlier will require less partitions on average for them to get isolated compared to normal samples. Each data point will then receive a score based on how easily they are isolated after X amounts of rounds. The data points that have abnormal scores will then be marked as anomalies.

Details

On a more formal note, we recursively divide each data instance by randomly selecting an attribute q and a split value p (within the min max of attribute q) until all of them are fully isolated. Isolation forest will then provide a ranking that reflects the degree of anomaly of each data instance according to their path lengths. The ranking or scores are called the anomaly scores which is calculated as follows:

$H(x)$: the number of steps until the data instance x is fully isolated.

$E[H(x)]$: the average of $H(x)$ from a collection of isolation trees.

The metrics make sense but one problem is that the maximum possible steps of iTree grows in the order of n while at the same time the average

steps only grows in the order of $\log n$. This will then lead to problems where the steps cannot be directly compared. Hence, a normalization constant varying by n will need to be introduced.

$c(n)$, the Path length normalization constant with the following formula:

$$c(m) = \begin{cases} 2H(m-1) - \frac{2(m-1)}{n} & \text{for } m > 2 \\ 1 & \text{for } m = 2 \\ 0 & \text{otherwise} \end{cases}$$

Equation Image by author

where $H(i)$ is the harmonic number that can be estimated by $\ln(i) + 0.5772156649$ (Euler's constant).

The full equation of the anomaly score:

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

Equation Image by author

Hence, if we run the whole data set through an Isolation Forest, we can obtain its anomaly score. Using the anomaly score s , we can deduce if there are anomalies whenever there are instances with an anomaly score very close to one. Any score lower than 0.5 will be identified as normal instances.

Note:

In sklearn's implementation, the anomaly scores are the opposite of the anomaly score defined in the original paper. They are further subtracted with a constant of 0.5. This is to easily identify anomalies (negative scores are identified with anomalies) [Read more](#)

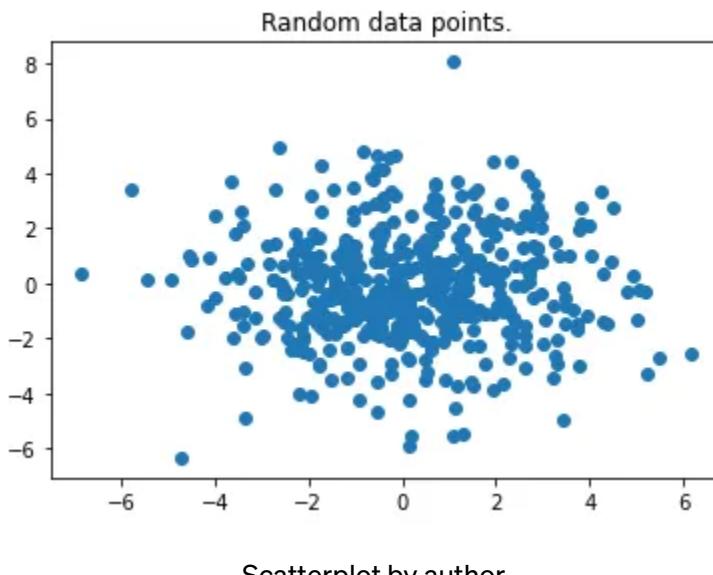
Implementation of Isolation Forest

Done with all the theories? Let's identify some outliers.

First off, we quickly import some useful modules that we will be using later on. We generate a dataset with random data points using the `make_blobs()` function.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

data, _ = make_blobs(n_samples=500, centers=1, cluster_std=2,
center_box=(0, 0))
plt.scatter(data[:, 0], data[:, 1])
plt.show()
```



Scatterplot by author

We can easily eyeball some outliers since this is only a 2-D use case. It is a good choice to proof that the algorithm works. Note that the algorithm can be used on a data set with multiple features without any problem.

We initialize an isolation forest object by calling `IsolationForest()`.

The hyperparameters used here are mostly default and recommended by the original paper.

Number of tree controls the ensemble size. We find that path lengths usually converge well before $t = 100$. Unless otherwise specified, we shall use $t = 100$ as the default value in our experiment.

Empirically, we find that setting subset sample to 256 generally provides enough details to perform anomaly detection across a wide range of data

— Fei Tony Liu, Kai Ming Ting (Author of the original paper, Isolation Forest)

- N_estimators here stands for the number of trees and max sample here stands for the subset sample used in each round.
- Max_samples ='auto' sets the subset size as min(256, num_samples).
- The contamination parameter here stands for the proportion of outliers in the data set. On default, the anomaly score threshold will follow as in the original paper. However, we can manually fix the proportion of outliers in the data if we have any prior knowledge. We set it to 0.03 here for demonstration purpose.

We then fit and predict the entire data set. It returns an array consisting of [-1 or 1] where -1 stands for anomaly and 1 stands for normal instance.

```
iforest = IsolationForest(n_estimators = 100, contamination = 0.03,
max_samples ='auto')
prediction = iforest.fit_predict(data)

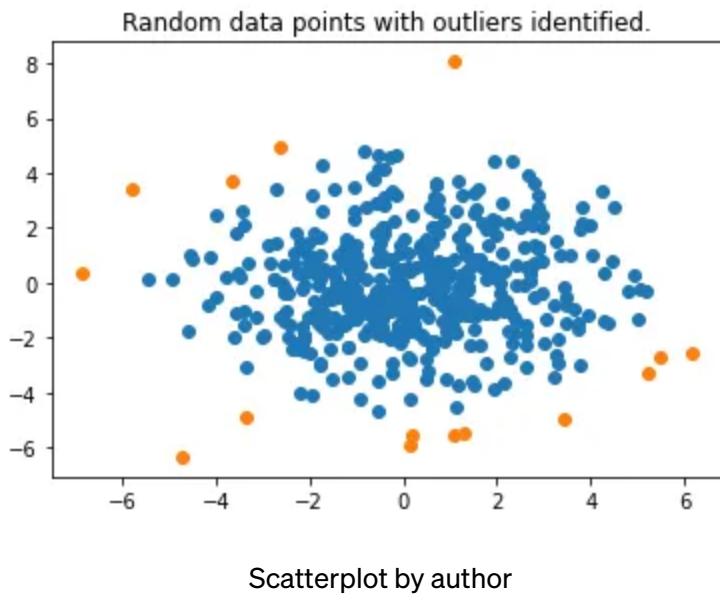
print(prediction[:20])
print("Number of outliers detected: {}".format(prediction[prediction
< 0].sum()))
print("Number of normal samples detected:
{}".format(prediction[prediction > 0].sum()))
```

```
[ 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1]
Number of outliers detected: -15
Number of normal samples detected: 485
```

Code output by author

We will then plot the outliers detected by Isolation Forest.

```
normal_data = data[np.where(prediction > 0)]
outliers = data[np.where(prediction < 0)]
plt.scatter(normal_data[:, 0], normal_data[:, 1])
plt.scatter(outliers[:, 0], outliers[:, 1])
plt.title("Random data points with outliers identified.")
plt.show()
```



We can see that it works pretty well and identifies the data points around the edges.

We can also call `decision_function()` to calculate the anomaly score of each data points. This way we can understand which data points are more abnormal.

```
score = iforest.decision_function(data)
data_scores = pd.DataFrame(list(zip(data[:, 0], data[:, 1], score)), columns = ['X', 'Y', 'Anomaly Score'])

display(data_scores.head())
```

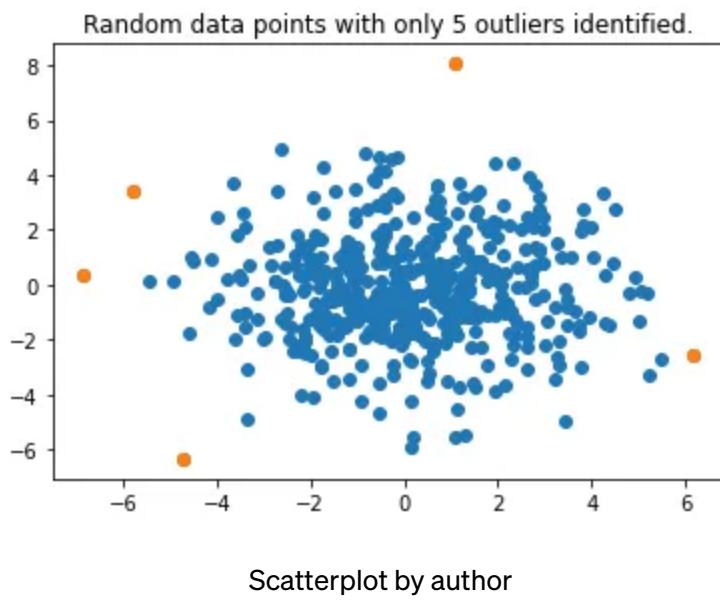
	X	Y	Anomaly Score
0	-1.024335	-1.342475	0.207694
1	0.711119	2.587249	0.155753
2	-0.809386	0.118526	0.205472
3	-2.229571	0.359373	0.188470
4	-1.066572	2.296957	0.158984

Code output by author

We pick the top 5 anomalies using the anomaly scores and then plot it again.

```
top_5_outliers = data_scores.sort_values(by = ['Anomaly Score']).head()

plt.scatter(data[:, 0], data[:, 1])
plt.scatter(top_5_outliers['X'], top_5_outliers['Y'])
plt.title("Random data points with only 5 outliers identified.")
plt.show()
```



Take-away

Isolation Forest is a fundamentally different outlier detection model that can isolate anomalies at great speed. It has a linear time complexity which makes it one of the best to deal with high volume data sets.

It pivots on the concept that since anomalies are “few and different”, they are easier to be isolated compared to normal points. Its Python implementation can be found at [sklearn.ensemble.IsolationForest](#).

Thank you for taking more time out of your busy schedule to sit down with me and enjoy this beautiful piece of algorithm.

Reference

[Isolation Forest](#) by Fei Tony Liu, Kai Ming Ting Gippsland School of Information Technology Monash University, Victoria, Australia.

Mlearning.ai Submission Suggestions

How to become a writer on Mlearning.ai

medium.com