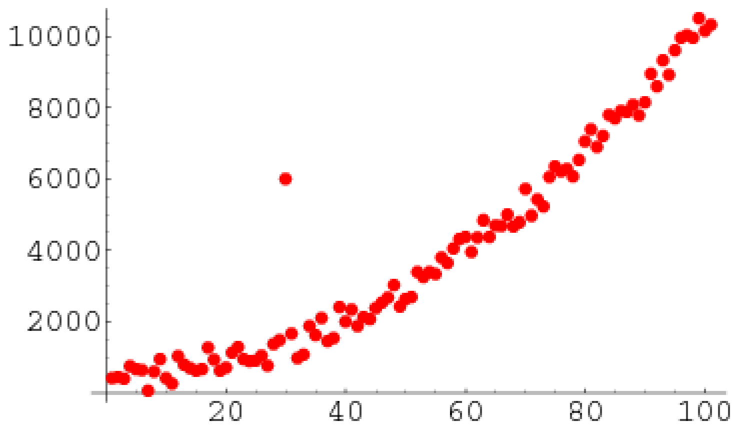
5 Ways to Detect Outliers/Anomalies That Every Data Scientist Should Know (Python Code)

Detecting Anomalies is critical to any business either by identifying faults or being proactive. This article discusses 5 different ways to identify those anomalies.

Will Badr Follow
Mar 5 · 6 min read ★

Photo by Will Myers on Unsplash

What is Anomaly/Outlier?



In statistics, outliers are data points that don't belong to a certain population. It is an abnormal observation that lies far away from other values. It's an observation that diverges from otherwise unstructured data.

For Example, you can clearly see the outlier in this list: [20,24,22,19,29,18,**4300**,30,18]

It is easy to identify it when the observations are just a bunch of numbers and it is one dimensional but when you have thousands of observations or multi-dimensions, you will need more clever ways to detect those values. This is what this article will cover.

Why Do We Care About Anomalies?

Detecting outliers or anomalies is one of the core problems in data mining. The emerging expansion and continued growth of data and the spread of IoT devices, make us rethink the way we approach anomalies and the use cases that can be built by looking at those anomalies.

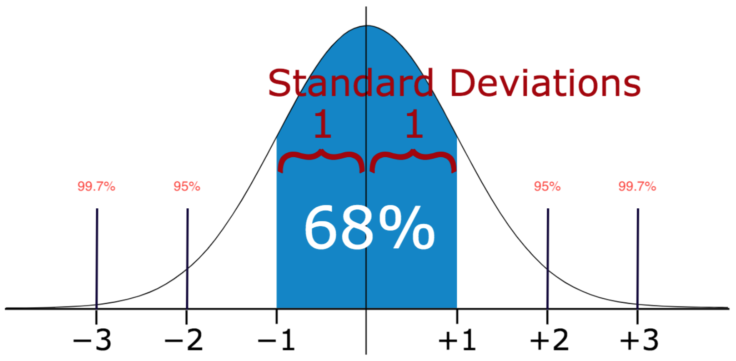
We now have smart watches and wristbands that can detect our heartbeats every few minutes. Detecting anomalies in the heartbeat data can help in predicting heart diseases. Anomalies in traffic patterns can help in predicting accidents. It can also be used to identify bottlenecks in network infrastructure and traffic between servers. Hence, the use cases and solution built on top of detecting anomalies are limitless.

Another reason why we need to detect anomalies is that when preparing datasets for machine learning models, it is really important to detect all the outliers and either get rid of them or analyze them to know why you had them there in the first place.

Now, let’s explore 5 common ways to detect anomalies starting with the most simple way.

Method 1—Standard Deviation:

In statistics, If a data distribution is approximately normal then about 68% of the data values lie within one standard deviation of the mean and about 95% are within two standard deviations, and **about 99.7%** lie within three standard deviations



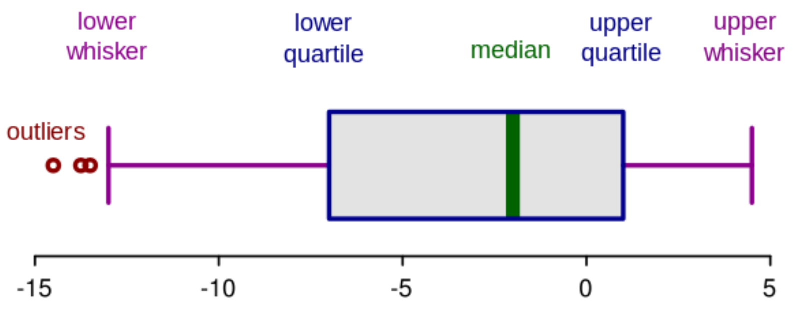
Therefore, if you have any data point that is more than 3 times the standard deviation, then those points are very likely to be anomalous or outliers.

Let’s see some code.

```
import numpy as np
import matplotlib.pyplot as plt
seed(1)
anomalies = []

# multiply and add by random numbers to get some real
random_data = np.random.randn(50000)  * 20 + 20

# Function to Detection Outlier on one-dimentional dat
def find_anomalies(data):
    # Set upper and lower limit to 3 standard deviatic
    random_data_std = std(random_data)
    random_data_mean = mean(random_data)
    anomaly_cut_off = random_data_std * 3

    lower_limit  = random_data_mean - anomaly_cut_off
    upper limit = random data mean + anomaly cut off
```

The output of this code is a list of values above 80 and below -40. Notice that the dataset I am passing is a one-dimensional dataset. Now, let's explore more advanced methods for multi-dimensional datasets.

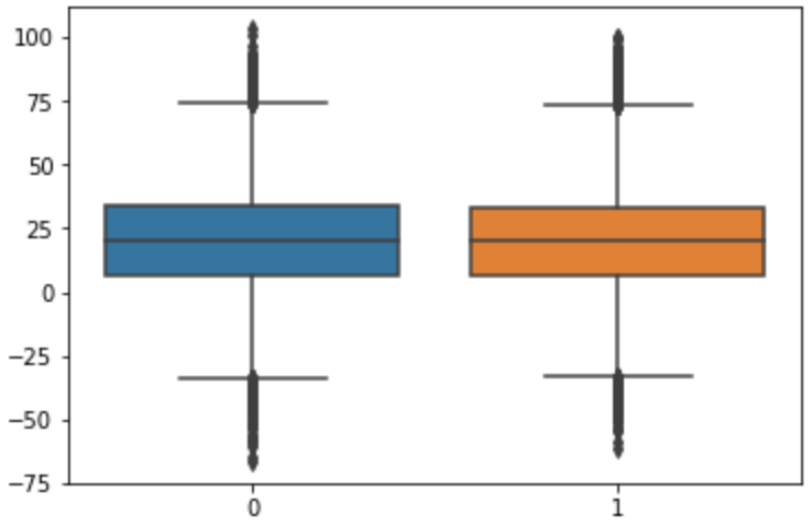
Method 2—Boxplots



Box plots are a graphical depiction of numerical data through their quantiles. It is a very simple but effective way to visualize outliers. Think about the lower and upper whiskers as the boundaries of the data distribution. Any data points that show above or below the whiskers, can be considered outliers or anomalous. Here is the code to plot a box plot:

```python
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(data=random_data)
```

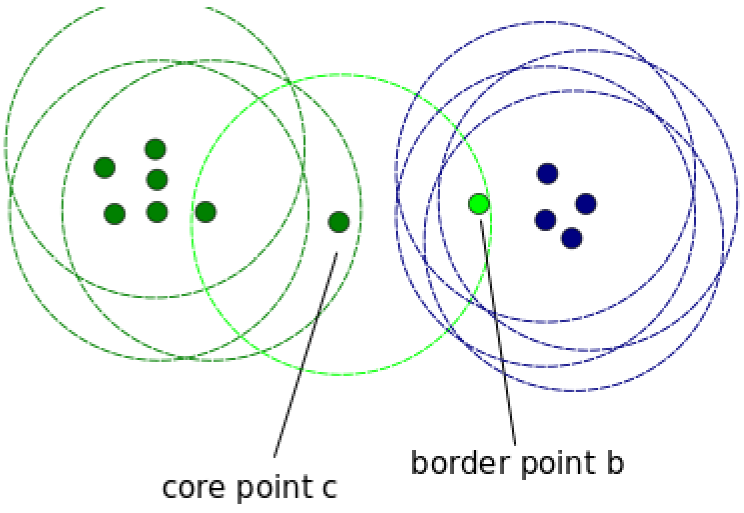
The above code displays the plot below. As you can see, it considers everything above 75 or below ~ -35 to be an outlier. The results are very close to method 1 above.



Method 3— DBScan Clustering:

DBScan is a clustering algorithm that's used cluster data into groups. It is also used as a density-based anomaly detection method with either single or multi-dimensional data. Other clustering algorithms such as k-means and hierarchal clustering can also be used to detect outliers. In this instance, I will show you an example of using DBScan but before we start, let's cover some important concepts. DBScan has three important concepts:

- ***Core Points:*** In order to understand the concept of the core points, we need to visit some of the hyperparameters used to define DBScan job. First hyperparameter (HP)is **min_samples.** This is simply the minimum number of core points needed in order to form a cluster**.** second important HP is **eps. eps** is the maximum distance between two samples for them to be considered as in the same cluster.
- ***Border Points*** are in the same cluster as core points but much further away from the centre of the cluster.



Source:https://stackoverflow.com/questions/34394641/dbscan-clustering-what-happens-when-border-point-of-one-cluster-is-considered

- Everything else is called ***Noise Points,*** those are data points that do not belong to any cluster. They can be anomalous or non-anomalous and they need further investigation. Now, let's see some code.

```
from sklearn.cluster import DBSCAN
seed(1)
random_data = np.random.randn(50000,2)  * 20 + 20

outlier_detection = DBSCAN(min_samples = 2, eps = 3)
clusters = outlier_detection.fit_predict(random_data)
```

The output of the above code is **94.** This is the total number of noisy points. SKLearn labels the noisy points as (-1). The downside with this method is that the higher the dimension, the less accurate it becomes. You also need to make a few assumptions like estimating the right value for **eps** which can be challenging.

Method 4— Isolation Forest:

Isolation Forest is an unsupervised learning algorithm that belongs to the ensemble decision trees family. This approach is different from all previous methods. All the previous ones were trying to find the normal region of the data then identifies anything outside of this defined region to be an outlier or anomalous.

This method works differently. It explicitly isolates anomalies instead of profiling and constructing normal points and regions by assigning a score to each data point. It takes advantage of the fact that anomalies are the minority data points and that they have attribute-values that are very different from those of normal instances. This algorithm works great with very high dimensional datasets and it proved to be a very effective way of detecting anomalies. Since this article is focusing on the implementation rather than the know-how, I will not go any further on how the algorithm works. However, the full details on how it works are covered in this paper.

Now, let's explore the code:

```
from sklearn.ensemble import IsolationForest
import numpy as np
np.random.seed(1)
random_data = np.random.randn(50000,2)  * 20 + 20

clf = IsolationForest( behaviour = 'new', max_samples=1
```

This code will output the predictions for each data point in an array. If the result is -1, it means that this specific data point is an outlier. If the result is 1, then it means that the data point is not an outlier

Method 5— Robust Random Cut Forest:

Random Cut Forest (RCF) algorithm is Amazon's unsupervised algorithm for detecting anomalies. It works by associating an anomaly score as well. Low score values indicate that the data point is considered "normal." High values indicate the presence of an anomaly in the data. The definitions of "low" and "high" depend on the application but common practice suggests that scores beyond three standard deviations from the mean score are considered anomalous. The details of the algorithm can be found in this paper.

The great thing about this algorithm is that it works with very high dimensional data. It can also work on real-time streaming data (built in AWS Kinesis Analytics) as well as offline data.

I explain the concept in much more details in the video below:

Detect Anomalies in Your Data with Amazo...

The paper shows some performance benchmarks when compared with Isolation Forest. Here are the results from the paper which shows that RCF is much more accurate and faster than Isolation Forests.

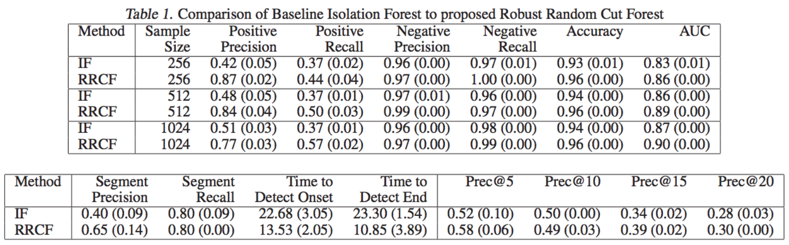


Table 1. Comparison of Baseline Isolation Forest to proposed Robust Random Cut Forest

Method	Sample Size	Positive Precision	Positive Recall	Negative Precision	Negative Recall	Accuracy	AUC
IF	256	0.42 (0.05)	0.37 (0.02)	0.96 (0.00)	0.97 (0.01)	0.93 (0.01)	0.83 (0.01)
RRCF	256	0.87 (0.02)	0.44 (0.04)	0.97 (0.00)	1.00 (0.00)	0.96 (0.00)	0.86 (0.00)
IF	512	0.48 (0.05)	0.37 (0.01)	0.97 (0.01)	0.96 (0.00)	0.94 (0.00)	0.86 (0.00)
RRCF	512	0.84 (0.04)	0.50 (0.03)	0.99 (0.00)	0.97 (0.00)	0.96 (0.00)	0.89 (0.00)
IF	1024	0.51 (0.03)	0.37 (0.01)	0.96 (0.00)	0.98 (0.00)	0.94 (0.00)	0.87 (0.00)
RRCF	1024	0.77 (0.03)	0.57 (0.02)	0.97 (0.00)	0.99 (0.00)	0.96 (0.00)	0.90 (0.00)

Method	Segment Precision	Segment Recall	Time to Detect Onset	Time to Detect End	Prec@5	Prec@10	Prec@15	Prec@20
IF	0.40 (0.09)	0.80 (0.09)	22.68 (3.05)	23.30 (1.54)	0.52 (0.10)	0.50 (0.00)	0.34 (0.02)	0.28 (0.03)
RRCF	0.65 (0.14)	0.80 (0.00)	13.53 (2.05)	10.85 (3.89)	0.58 (0.06)	0.49 (0.03)	0.39 (0.02)	0.30 (0.00)

Full example code can be found here:

awslabs/amazon-sagemaker-examples

Example notebooks that show how to apply machine learning and deep learning in Amazon...

github.com

Conclusion:

We live in a world where the data is getting bigger by the second. The value of the data can diminish over time if not used properly. Finding anomalies either online in a stream or offline in a dataset is crucial to identifying problems in the business or building a proactive solution to potentially discover the problem before it happens or even in the exploratory data analysis (EDA) phase to prepare a dataset for ML. I hope that you find the article useful, let me know what you think in the comments section below.