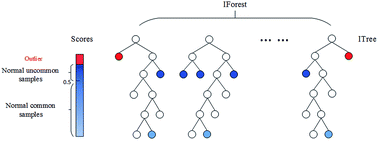
**Isolation Forest** is a technique for finding unusual or abnormal data points in a dataset.

<https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/>



Isolation Forests(IF), similar to Random Forests, are build based on decision trees. And since there are no pre-defined labels here, it is an unsupervised model.

Just like the random forests, **isolation forests** are built using decision trees. They are implemented in an unsupervised fashion as there are no pre-defined labels. Isolation forests were designed with the idea that anomalies are “few and distinct” data points in a dataset.

It works by randomly splitting the data into smaller groups using decision trees. The idea is that normal data points will be grouped together in large groups, while abnormal data points will be separated in small groups. The smaller the group, the more likely it is an anomaly. Isolation Forest assigns an anomaly score to each data point based on how easy it is to isolate it from the rest of the data. The higher the score, the more likely it is an anomaly.

In an Isolation Forest, randomly sub-sampled data is processed in a tree structure based on randomly selected features. The samples that travel deeper into the tree are less likely to be anomalies as they required more cuts to isolate them. Similarly, the samples which end up in shorter branches indicate anomalies as it was easier for the tree to separate them from other observations.

----------------------------------------------------------------------------

KNN

<https://laptrinhx.com/k-nearest-neighbors-knn-for-anomaly-detection-3629720645/>

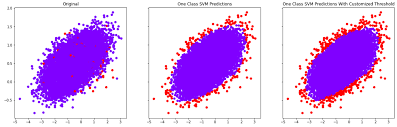
KNN anomaly detection is a technique for finding unusual or abnormal data points in a dataset. It works by comparing each data point to its nearest neighbors and measuring how similar or different they are. The idea is that normal data points will have many similar neighbors, while abnormal data points will have few or no similar neighbors. KNN anomaly detection assigns an anomaly score to each data point based on how many neighbors it has and how far they are. The lower the score, the more likely it is an anomaly.



Although kNN is a supervised ML algorithm, when it comes to anomaly detection it takes an unsupervised approach. This is because there is no actual “learning” involved in the process and there is no pre-determined labeling of “outlier” or “not-outlier” in the dataset, instead, it is entirely based upon threshold values.

One Class SVM

[One-Class SVM For Anomaly Detection | by Amy @GrabNGoInfo | GrabNGoInfo | Medium](https://medium.com/grabngoinfo/one-class-svm-for-anomaly-detection-6c97fdd6d8af)



One Class SVM anomaly detection is a technique for finding unusual or abnormal data points in a dataset. It works by using a Support Vector Machine (SVM) to learn a boundary that separates normal data points from abnormal ones. A Support Vector Machine is a machine learning algorithm that can find a line or a curve that best divides two groups of data points. One Class SVM uses only one group of data points, which are assumed to be normal, and tries to find a boundary that encloses them as tightly as possible. Any data point that lies outside the boundary is considered an anomaly,

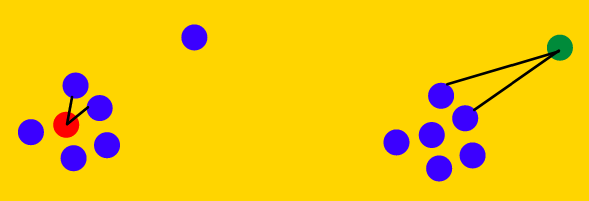
One-Class Support Vector Machine (SVM) is an unsupervised model for anomaly or outlier detection. Unlike the regular supervised SVM, the one-class SVM does not have target labels for the model training process. Instead, it learns the boundary for the normal data points and identifies the data outside the border to be anomalies.

# tep 4: Train One-Class Support Vector Machine (SVM) Model

When training the one-class SVM, there are a few critical hyperparameters.

* nu is to specify the percentage of anomalies. nu=0.01 means that we have around 1% outliers in the dataset.
* Kernel specifies the kernel type. The radial basis function (rbf) kernel is a commonly used kernel type. It maps data from a low dimensional space to a high dimensional space to help the SVM model draw a decision boundary.
* gamma is a kernel coefficient, and it is for 'rbf', 'poly', and 'sigmoid' kernels. When setting it to 'auto', the kernel coefficient is 1 over the number of features.

**Angle-based Outlier Detection (ABOD)**



ABOD based anomaly detection is a technique that uses angles between data points to measure how different they are from the rest of the data. It is a geometric method that works well in high-dimensional spaces. The idea is that outliers have larger angles with their neighbors than normal points. To find the outliers, ABOD calculates the variance of the angles between each point and all pairs of its neighbors, and assigns a score based on how large the variance is. [The higher the score, the more likely the point is an outlier](https://blog.paperspace.com/outlier-detection-with-abod/)

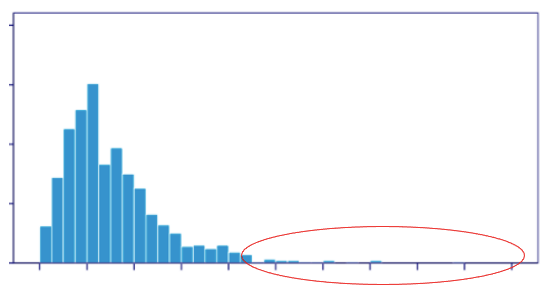
This technique is based on the idea of keeping an eye on the angle formed by a set of any three data points in the multi-variate feature space. The variance in the magnitude of the angular enclosure comes out to be different for outliers and the normal points. Usually the observed variance is higher for the inlier points than for outliers, hence such a measure helps us to cluster normal and outlier points differently. The angle-based outlier (ABOD) technique works pretty well in high-dimensional space, unlike other distance-based measures that suffer from the “[Curse of dimensionality](https://en.wikipedia.org/wiki/Curse_of_dimensionality?ref=blog.paperspace.com)”. The distance between any two points in the high dimension space is almost similar. In such scenarios, angles can give better picture of closeness.

*The algorithm is pretty straightforward and is described below -*

1. Iterate over each data point and calculate the angle it*(pivot)* forms with all other data pairs and store them in the angle list.
2. Calculate the variance of this angle list created as part of Step-1.
3. Variance values less than a certain threshold can be marked as potential anomalies. *(****Low variance****means pivot point is an****anomaly****,****High variance****means pivot point is a****normal point****)*

*HBOS*

[*https://medium.com/dataman-in-ai/anomaly-detection-with-histogram-based-outlier-detection-hbo-bc10ef52f23f*](https://medium.com/dataman-in-ai/anomaly-detection-with-histogram-based-outlier-detection-hbo-bc10ef52f23f)



HBOS anomaly detection is a technique that uses histograms to estimate the probability distribution of each feature in a dataset. It is a statistical method that works well for univariate data and assumes independence of the features. The idea is that outliers have low probability values in the histograms than normal points. To find the outliers, HBOS calculates the histogram-based outlier score (HBOS) for each point by multiplying the inverse of the probability values of its features, and assigns a score based on how low the probability is. [The lower the score, the more likely the point is an outlie](https://medium.com/dataman-in-ai/anomaly-detection-with-histogram-based-outlier-detection-hbo-bc10ef52f23f)

**How Does the HBOS Work?**

The HBOS constructs the histograms independently for all the N variables. The height of the bin is used to measure the “outlier-ness”. Most of the observations belong to the bins of high frequency, and outliers belong to the bins of low frequency. The univariate outlier score is defined as the inverse of the height of a bin.

**Local Outlier Factor (LOF)**

[**https://towardsdatascience.com/anomaly-detection-with-local-outlier-factor-lof-d91e41df10f2**](https://towardsdatascience.com/anomaly-detection-with-local-outlier-factor-lof-d91e41df10f2)

**https://www.geeksforgeeks.org/local-outlier-factor/**

Based on web search results, Local Outlier Factor (LOF) anomaly detection is a technique that uses the distance between data points and their neighbors to measure how different they are from the rest of the data. It is a nearest-neighbor method that works well in multivariate settings. The idea is that outliers have larger distances with their neighbors than normal points. To find the outliers, LOF calculates the local density of each point by comparing its distance to its k nearest neighbors, and assigns a score based on how much the local density deviates from the average density of the neighbors. [The higher the score, the more likely the point is an outlier**1**](https://en.wikipedia.org/wiki/Local_outlier_factor)

LOF compares the density of any given data point to the density of its neighbors. Since outliers come from low-density areas, the ratio will be higher for anomalous data points. As a rule of thumb, a normal data point has a LOF between 1 and 1.5 whereas anomalous observations will have much higher LOF. The higher the LOF the more likely it is an outlier. If the LOF of point X is 5, it means the average density of X’s neighbors is 5 times higher than its local density.

**Working of LOF:** Local density is determined by estimating distances between data points that are neighbors (k-nearest neighbors). So for each data point, local density can be calculated. By comparing these we can check which data points have similar densities and which have a lesser density than its neighbors. The ones with the lesser densities are considered as the outliers

https://www.datatechnotes.com/2020/04/anomaly-detection-with-dbscan-in-python.html

Based on web search results, DBSCAN anomaly detection is a technique that uses clustering to identify outliers in a dataset. It is a density-based method that groups data points based on their proximity and density. The idea is that outliers are the data points that do not belong to any cluster or have very low density. To find the outliers, DBSCAN defines two types of data points: core points and border points. Core points are the data points that have at least a minimum number of neighbors (min\_samples) within a given distance (eps). Border points are the data points that are reachable from core points but have less than min\_samples neighbors. All other data points are outliers. [DBSCAN assigns each data point to a cluster or labels it as an outlier based on these definitions](https://www.datatechnotes.com/2020/04/anomaly-detection-with-dbscan-in-python.html)

To implement DBSCAN in Python, you can use the DBSCAN class from the scikit-learn library. Here are the basic steps:

* Import the DBSCAN class and other libraries you need, such as numpy, pandas and matplotlib.
* Load or create your dataset as a numpy array or a pandas dataframe. You can use the make\_blobs function from scikit-learn to generate some synthetic data for testing.
* Instantiate the DBSCAN object with the parameters you want, such as eps, min\_samples and metric. You can use the default values or tune them according to your data characteristics and desired results.
* Fit the DBSCAN object to your data using the fit method. This will assign each data point to a cluster label or -1 for outliers.
* Optionally, you can use the fit\_predict method to fit and predict the cluster labels in one step.
* Plot or print the results to see the clusters and outliers. You can use matplotlib.pyplot.scatter to plot the data points with different colors based on their labels.

Here is an example code that implements DBSCAN in Python:

# Import libraries

from sklearn.cluster import DBSCAN

from sklearn.datasets import make\_blobs

import numpy as np

import matplotlib.pyplot as plt

# Generate some synthetic data

X, \_ = make\_blobs(n\_samples=100, centers=3, cluster\_std=0.5, random\_state=42)

# Instantiate DBSCAN with eps=0.5 and min\_samples=5

dbscan = DBSCAN(eps=0.5, min\_samples=5)

# Fit and predict cluster labels

y\_pred = dbscan.fit\_predict(X)

# Plot the results

plt.scatter(X[:, 0], X[:, 1], c=y\_pred, cmap="plasma")

plt.xlabel("Feature 0")

plt.ylabel("Feature 1")

plt.title("DBSCAN clustering")

plt.show()

Based on web search results, choosing the best eps and min\_samples values for DBSCAN depends on your data characteristics and desired results. There is no general formula or rule to find the optimal values, but some methods or guidelines are:

* For min\_samples, you can use your domain knowledge or a rule of thumb based on the number of dimensions in your data. [A common choice is min\_samples = D + 1, where D is the number of dimensions**1**](https://stats.stackexchange.com/questions/88872/a-routine-to-choose-eps-and-minpts-for-dbscan).
* For eps, you can use a knn distance histogram and look for a “knee” or an elbow point where the distance increases sharply. [This indicates a suitable distance threshold to separate clusters from noise**2**](https://stackoverflow.com/questions/12893492/choosing-eps-and-minpts-for-dbscan-r)[**3**](https://towardsdatascience.com/how-to-use-dbscan-effectively-ed212c02e62). Alternatively, you can use a grid search or a trial-and-error approach to find a value that produces reasonable results.
* [You can also use some variations or extensions of DBSCAN that do not require eps, such as OPTICS, which produces a hierarchical clustering that can be seen as running DBSCAN with every possible eps](https://stackoverflow.com/questions/12893492/choosing-eps-and-minpts-for-dbscan-r)

Kmeans

K-means anomaly detection is a technique that uses clustering to identify outliers in a dataset. It is a centroid-based method that groups data points based on their similarity or distance to a cluster center. The idea is that outliers are the data points that are far away from any cluster center. To find the outliers, K-means assigns each data point to one of the K clusters based on the minimum distance to the cluster center. Then, it calculates the distance or similarity of each data point to its assigned cluster center. [A threshold value can be used to detect anomalies: if the distance or similarity of a data point to its cluster center is greater than the threshold value, then it is an outlier](https://github.com/gprashmi/Anomaly-Detection-Using-K-means-Clustering)