Title: Anomaly Detection model for IoT Data

Problem Statement: IoT sensors generate vast amounts of data in applications like smart cities, industrial automation, and healthcare. Detecting anomalies in this data is critical to ensure reliability, prevent malfunctions, and address security risks. However, supervised methods are impractical due to the lack of labeled data and the dynamic nature of IoT environments.

This project aims to develop an unsupervised anomaly detection system that learns normal sensor behavior, detects deviations, and adapts to changing data patterns.

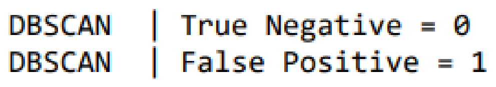
Objective: The goal of this project is to apply anomaly detection models on IoT data in order to find acquisitions that present anomalies, i.e., mistakes in acquiring the data, or unexpected variations of the assesses. We were provided a dataset that contains weather acquisitions of different features, such as 'rain', 'wind' and 'temperature'.

Proposed System: In order to solve an anomaly detection problem, we propose these different approaches:

- Statistical methods: These methods use past measurements to approximate a model of the correct behavior of a sensor, where they mark as an anomaly an incompatible result. In this project, the Average Low-High Pass Filter and Seasonal Extreme Studentized Deviate methods were used.
- Probabilistic methods: These methods rely on estimating the probability of the behavior of a sensor, using different well-known distributions, and compare it with a predefined threshold. In this project, the Univariate Gaussian Predictor was used.
- Clustering-based methods: These methods are dependent on calculating the distances between data measurements to distinguish between anomalous and correct readings by assigning the latter to clusters. In this project, the Local Density Cluster-Based Outlier Factor was used.

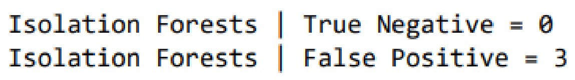
Comparison of Technologies: Algorithms used in this project are:

- DBSCAN



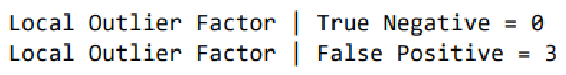
```
DBSCAN  | True Negative = 0
DBSCAN  | False Positive = 1
```

- Isolation forests



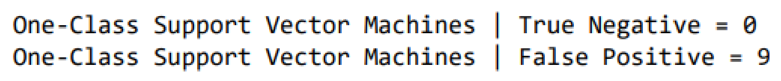
```
Isolation Forests | True Negative = 0
Isolation Forests | False Positive = 3
```

- Local Outlier Factor



```
Local Outlier Factor | True Negative = 0
Local Outlier Factor | False Positive = 3
```

- One-Class Support Vector Machines



```
One-Class Support Vector Machines | True Negative = 0
One-Class Support Vector Machines | False Positive = 9
```

Result:

We will compare the results of each algorithm to the content of this data frame and we will define two values:

- True Negative(TN): The algorithm has predicted an outlier, as an outlier.
- False Positive(FP): The algorithm has predicted an non-outlier, as an outlier.

Let's note that:

- 0 = no outlier = positive
- 1 = outlier = negative
- true = the algorithm predicted correct
- false = the algorithm predicted incorrect

The following table presents the experimental results for the month of January.

Algorithms	**TN**	**TN/Anomalies(%)**	**FP**
DBSCAN	1	33.33	0
IF	1	33.33	2
LOF	1	33.33	2
1CSVM(v=0.02, g=0.005)	3	100	6

The df_anomaly_reference is created based on normal winter values:

‘pluie’>20 or ‘vent’>60 or ‘temp’>30 or ‘temp’<6.

Conclusion: Let's note that False Positive will not be a strong criterion in our comparison. In fact, predicting an acquisition as an outlier, when it is not an outlier, is not a big deal. The results show that One-Class SVM is well accurate in predicting most of the outliers defined by our metric for most of the months. Its hyperparameters slightly vary from one month to another due to the different atmospheric features.

References:

https://mikulskibartosz.name/outlier-detection-with-scikit-learn-d9246b33d352

https://www.depends-on-the-definition.com/detecting-network-attacks-with-isolation-forests/

http://www.cse.ust.hk/~leichen/courses/comp5331/lectures/LOF_Example.pdf

http://sdsawtelle.github.io/blog/output/week9-anomaly-andrew-ng-machine-learning-with-python.html