Time series sensor data anomaly detection using Statistical and Machine Learning Techniques: the INFN CNAF data center use case

Luca Torzi

September 13, 2023

Problem

Detection of the anomalies in the INFN CNAF Data Center [a].

An **anomaly** is an event that deviates from what is standard, normal, or expected.

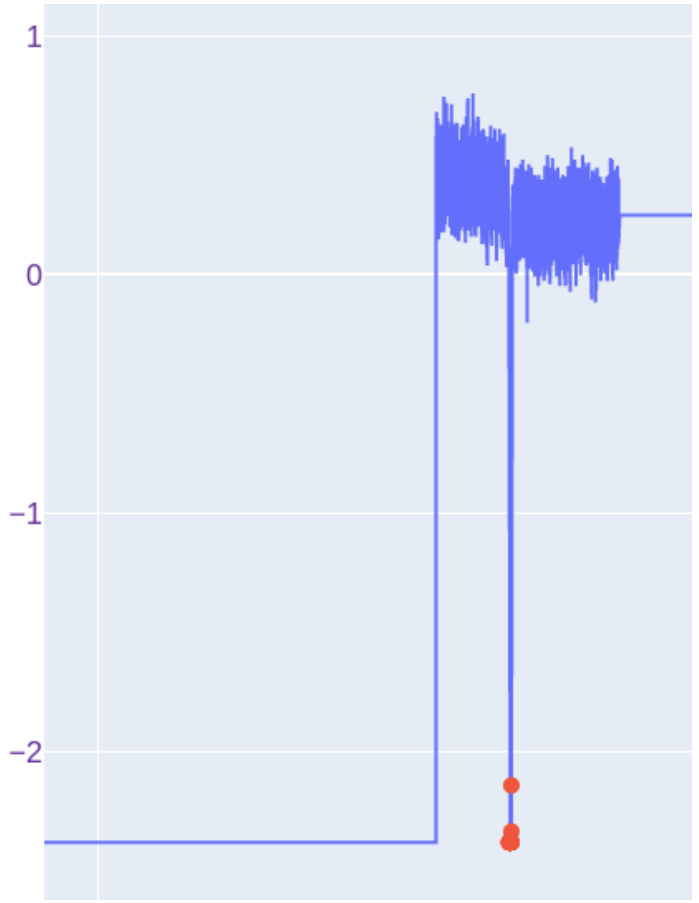

Figure 1: Example of anomalies on the values of a sensor.

[a] Sotware used: Pandas [1], Matplotlib [2], Seaborn [3], Plotly [4], SciPy [5], scikit-learn [6][7], PyTorch [8].

Composition of the Data Center

Composition of the data center

There are 4 main different components:

- the **electrical implant**
- the **UPS**
- the **cooling system**
- the **IT equipment**

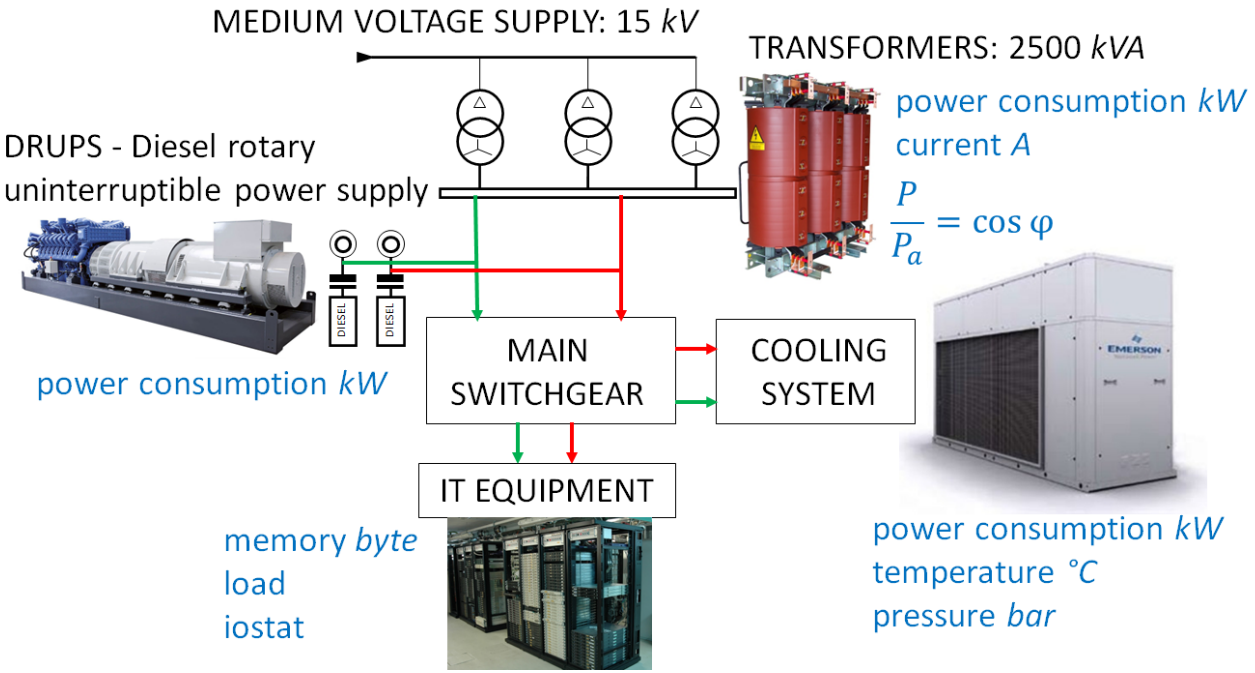

Figure 2: Components of the data center.

Electrical Implant

There are **three transformers**.
They works in couples, so two work and one is off.

In total there are **19 sensors**

- 6 sensors for each transformer
- 1 is global.

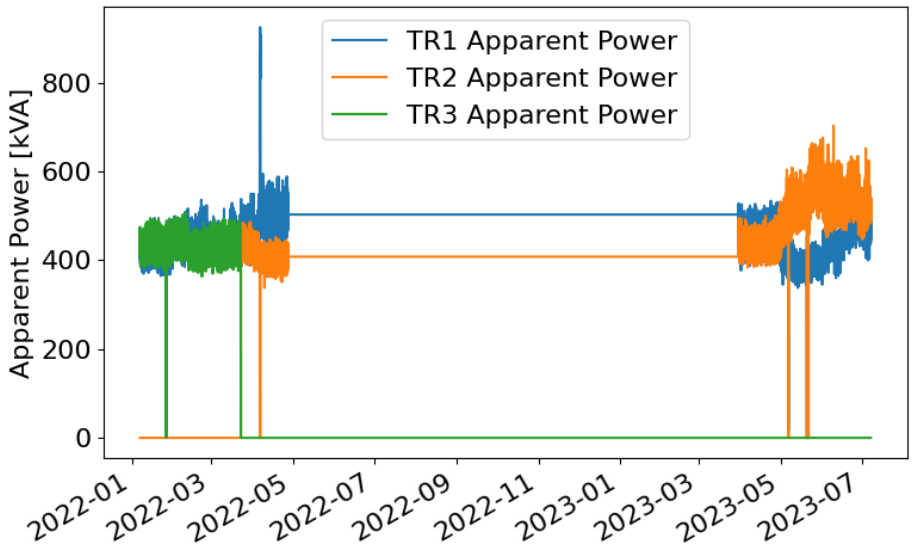

Figure 3: Apparent Power of the transformers.

Electrical Implant

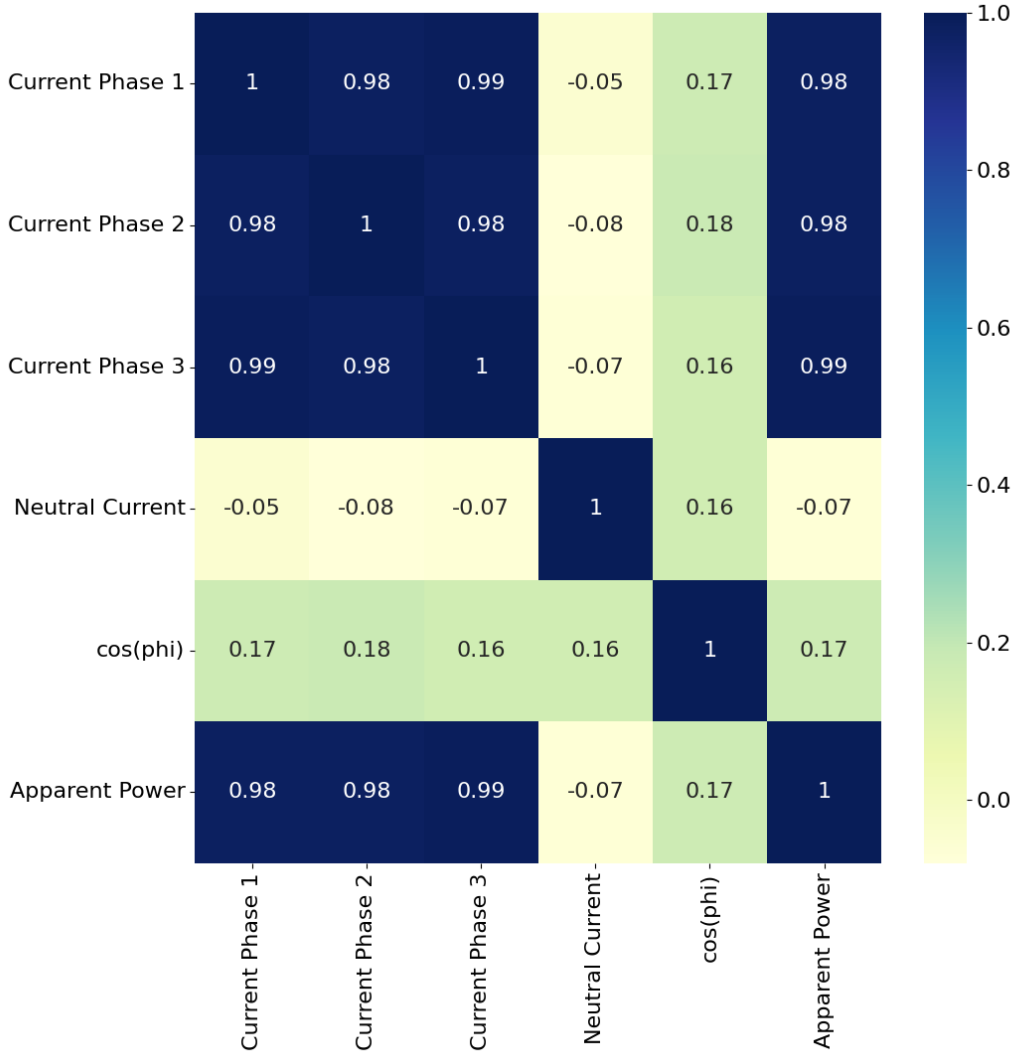

Figure 4: Correlation matrix of the transformer 1 sensors.

UPS

There is only one sensor for this part of the system.

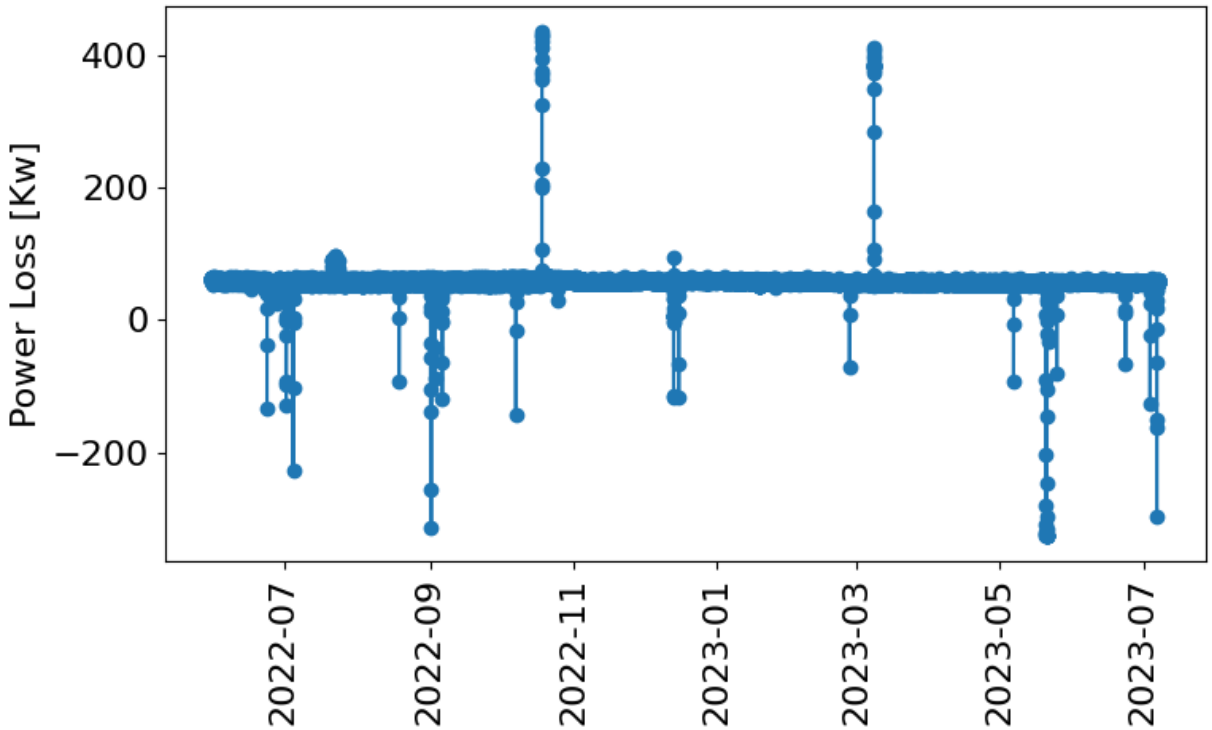

Figure 5: Values of the sensor over the time.

Cooling system

There are **six refrigerator units**.
In total there are **51 sensors**

- 7 sensors for each refrigerator unit
- 9 are global.

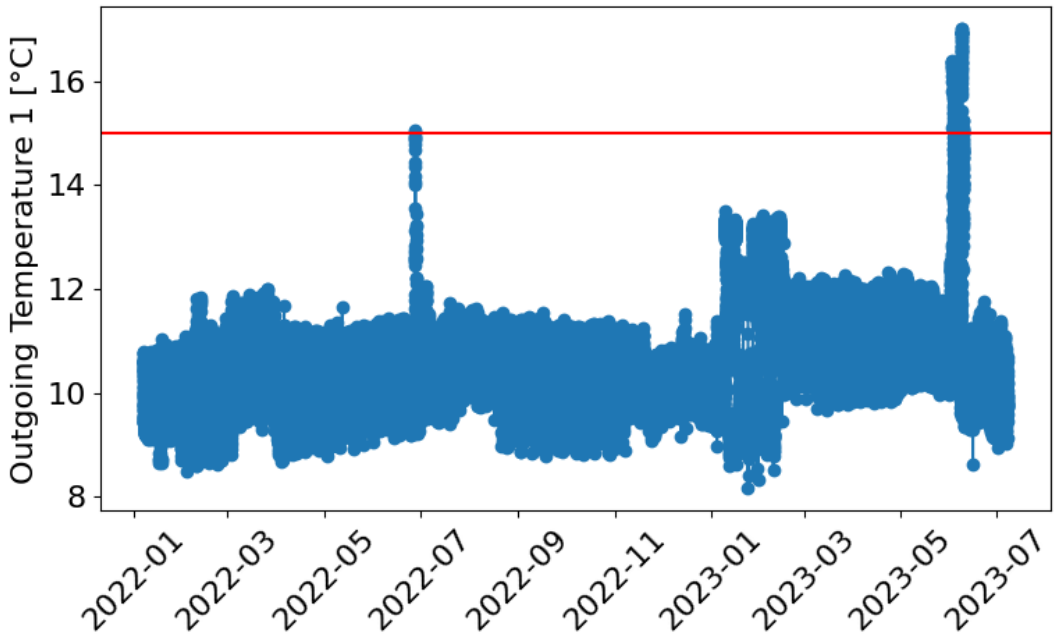

Figure 6: Outgoing temperature (first group of pumps). The red line marks the threshold of 15°C, because alarms are sent if this temperature is exceeded.

Cooling system

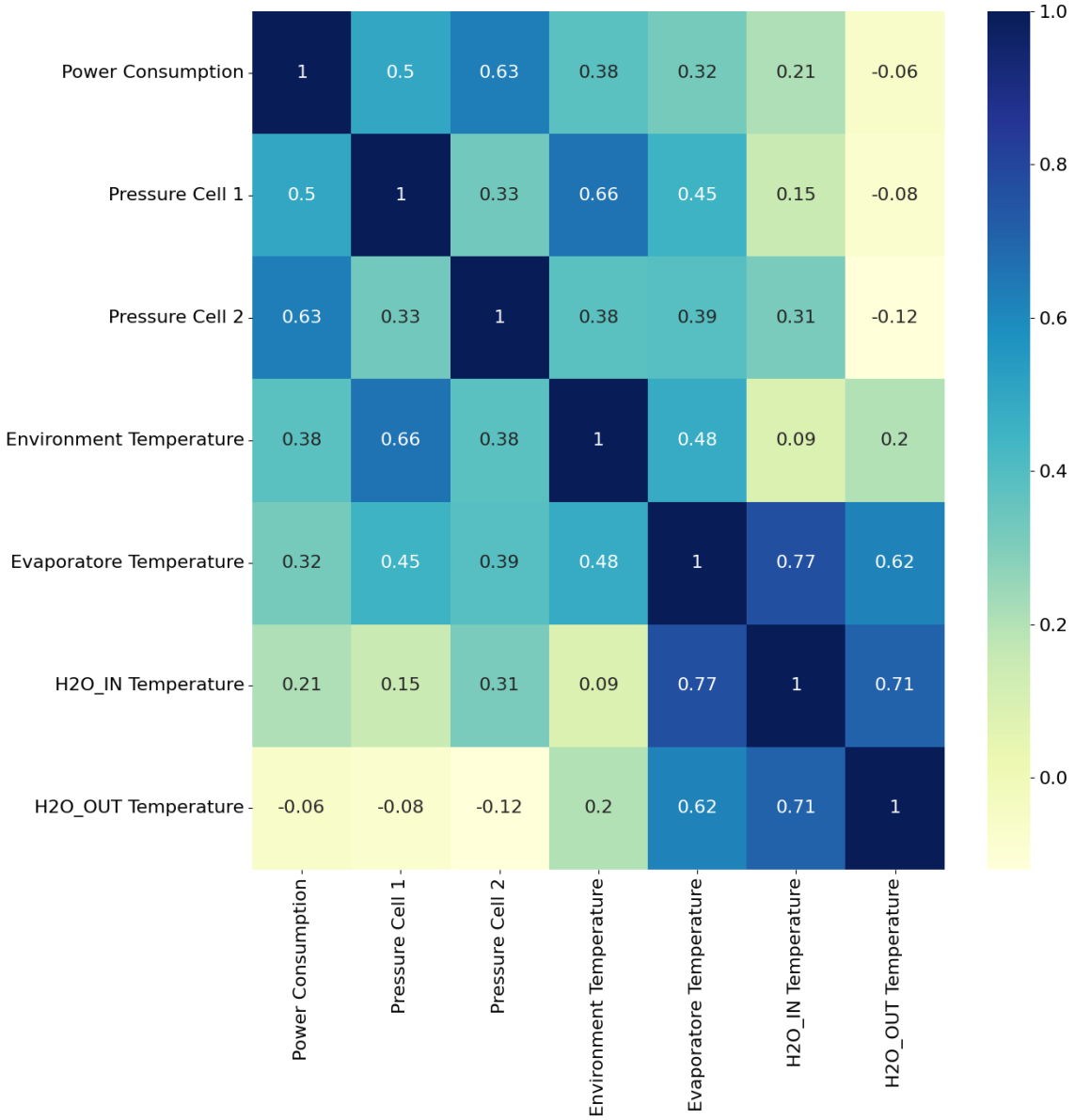

Figure 7: Correlation matrix of the sensors of the refrigerator unit 1.

Statistical approaches to detect the anomalies:

- ► Z-score
- ► Interquartile range

Z-score

It measures the number of standard deviations by which a data point is above or below the mean value.

$$Zscore = \frac{x - \mu}{\sigma}$$

μ is the mean of the population, σ is the standard deviation.

Every point that has a Z-score bigger than 4 (in absolute value) is considered as anomalous.

Interquartile range

The **IQR** is a measure of statistical dispersion.

Every value that is smaller than the 0.2-percentile or bigger than the 99.8-percentile is detected as anomalous.

This type of measurement is more robust against the outliers because we are not considering values at the borders [9].

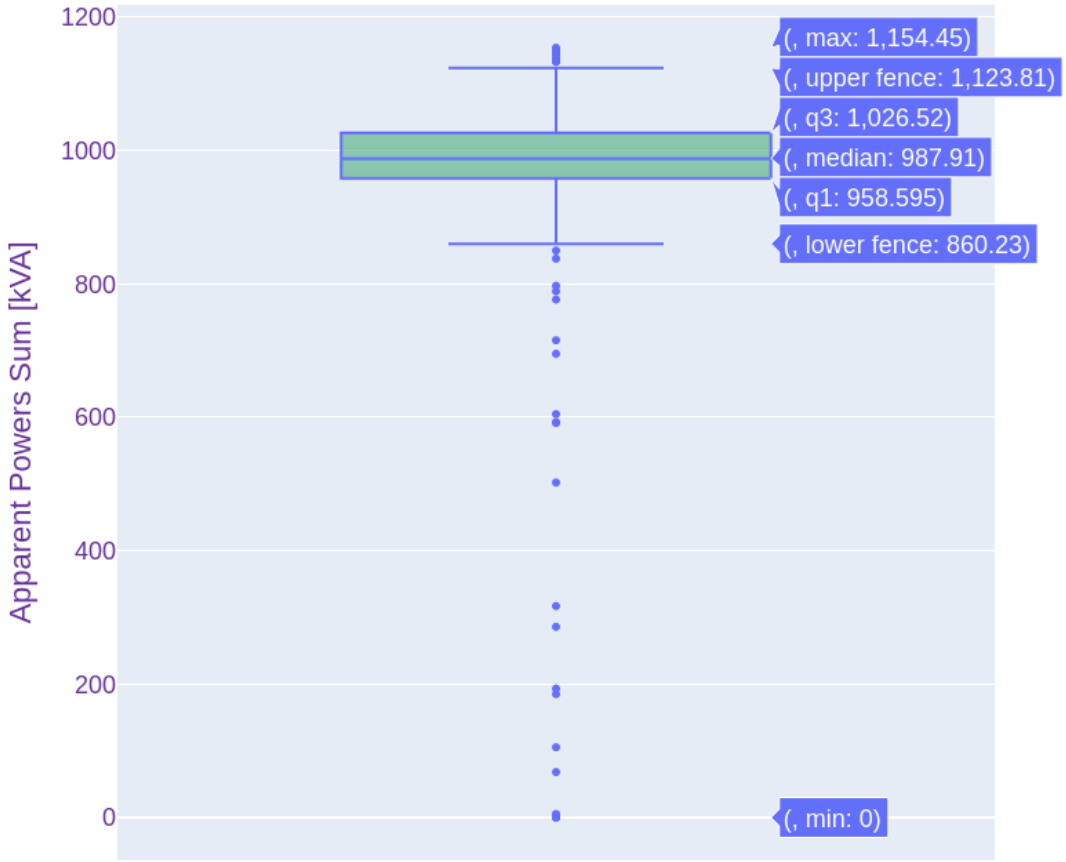

Figure 8: Boxplot of the sum of the transformers apparent powers. In the image it is possible to see the quartiles (25th-percentile and 75th-precentile) and the median.

Results on the cooling system sensors

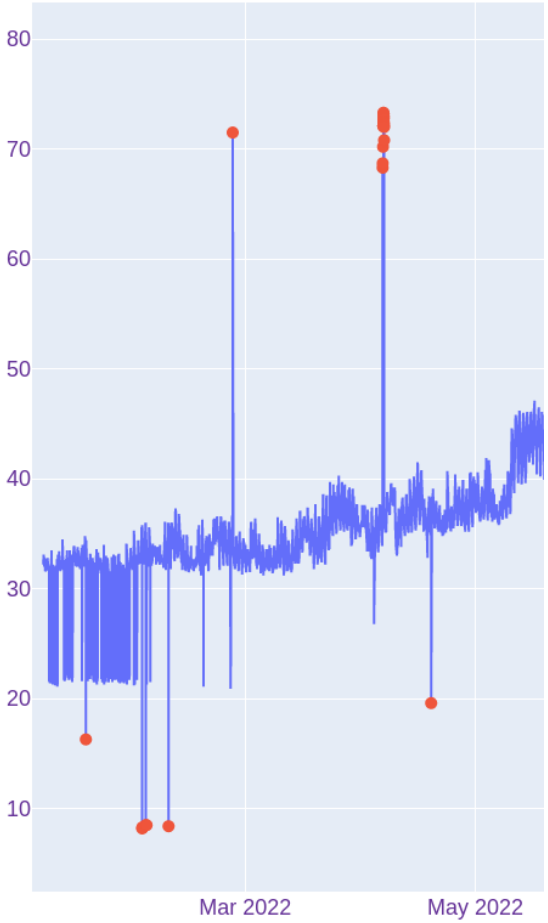

Figure 9: Anomalies detected by the Z-score method on the refrigerator unit 2 **Power Consumption values** [kW].

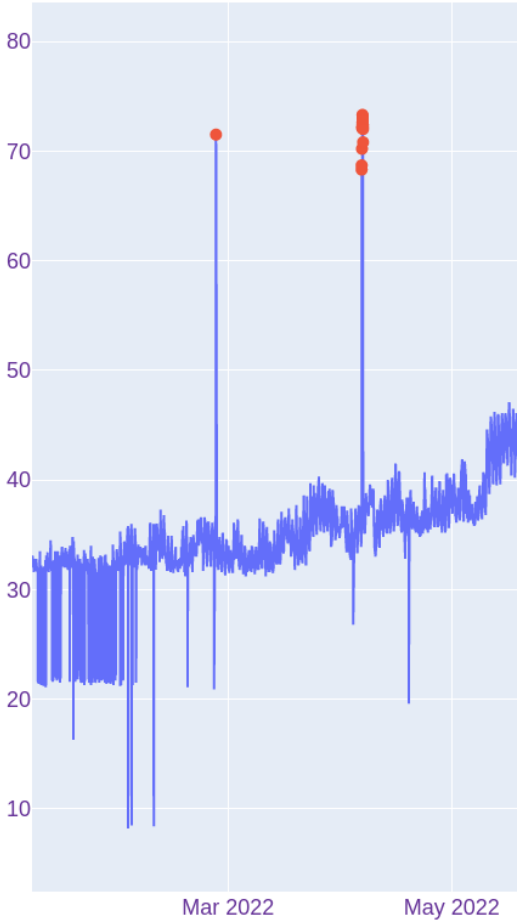

Figure 10: Anomalies detected by the IQR method on the refrigerator unit 2 **Power Consumption values** [kW].

Results on the cooling system sensors

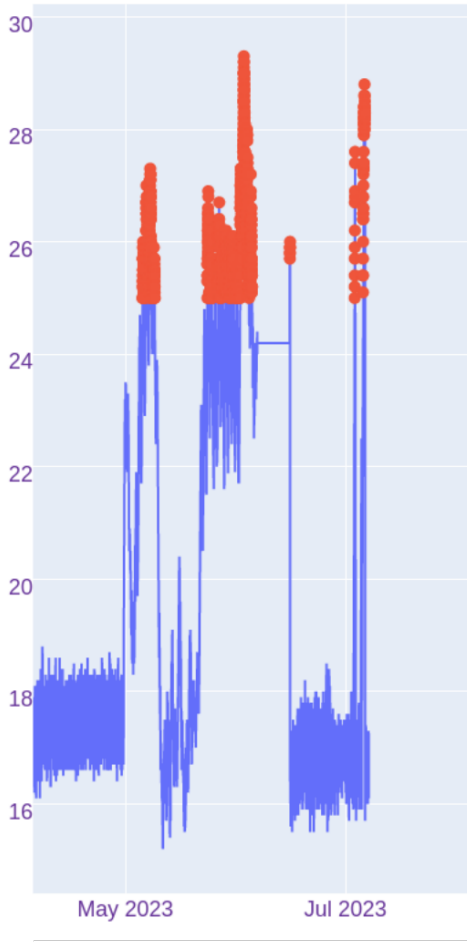

Figure 11: Anomalies detected by the Z-score method on the sensor that measures the **incoming temperature** to the refrigerator unit 4 [°C].

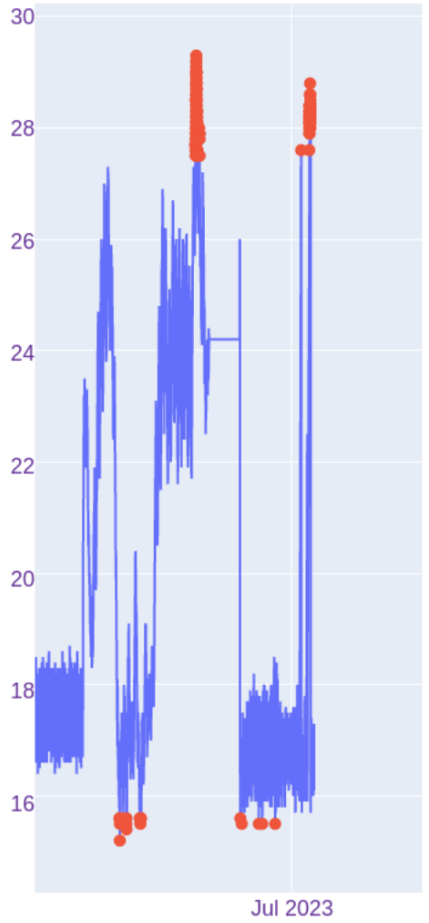

Figure 12: Anomalies detected by the Z-score method on the sensor that measures the **incoming temperature** to the refrigerator unit 4 [°C].

Result on the electrical implant sensors

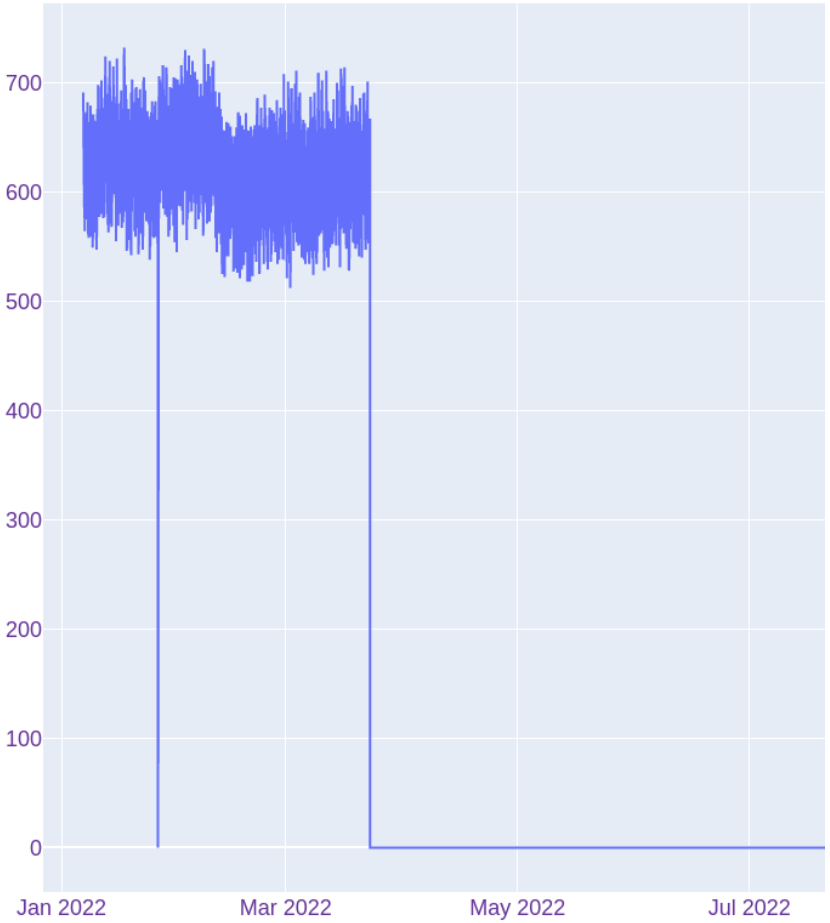

Figure 13: Anomalies detected by the Z-score method on the transformer 3 **current phase 1 values** [A].

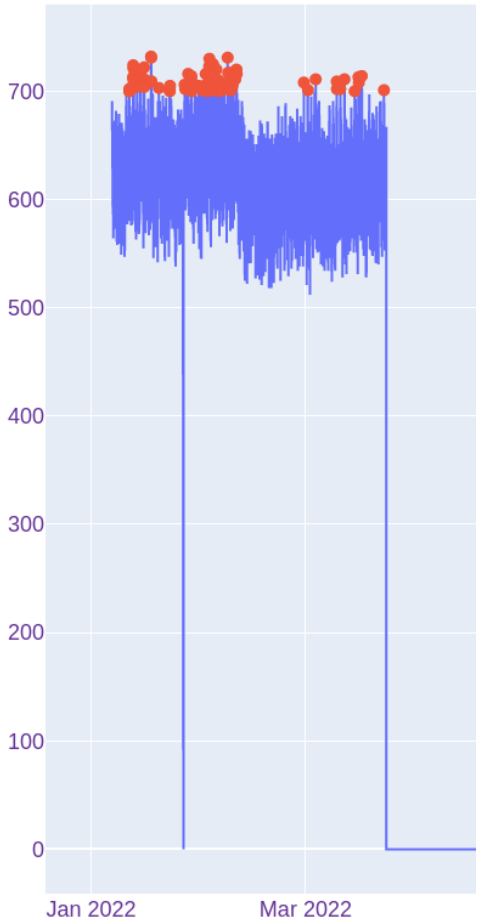

Figure 14: Anomalies detected by the IQR method on the transformer 3 **current phase 1 values** [A].

Result on the UPS system sensor

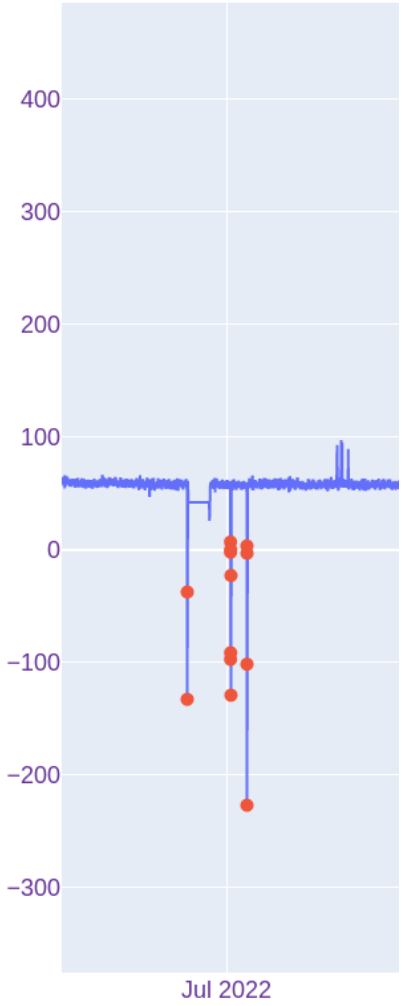

Figure 15: Anomalies detected by the Z-score method on the **ups sensor values** [kW].

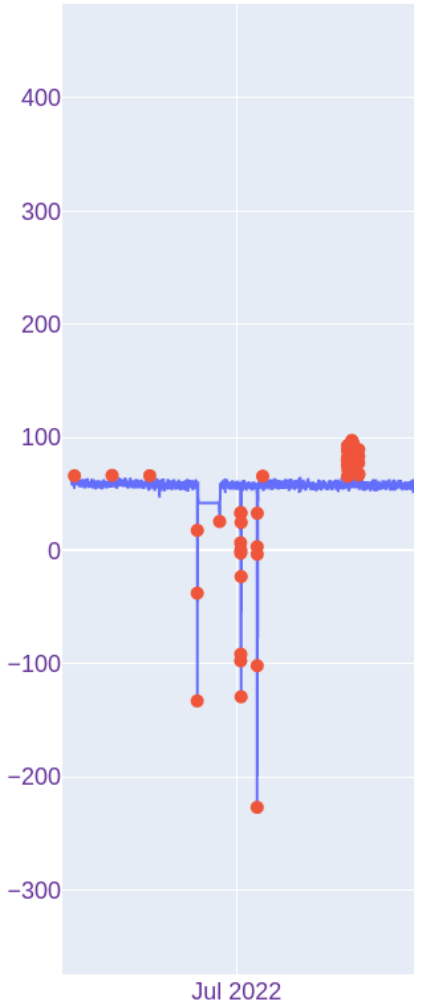

Figure 16: Anomalies detected by the IQR method on the **ups sensor value** [kW].

Machine Learning approach to detect the anomalies:

DBSCAN

DBSCAN

DBSCAN [10] is an algorithm that creates clusters of datapoints and the points that do not belog to any cluster are detected as anomalous.

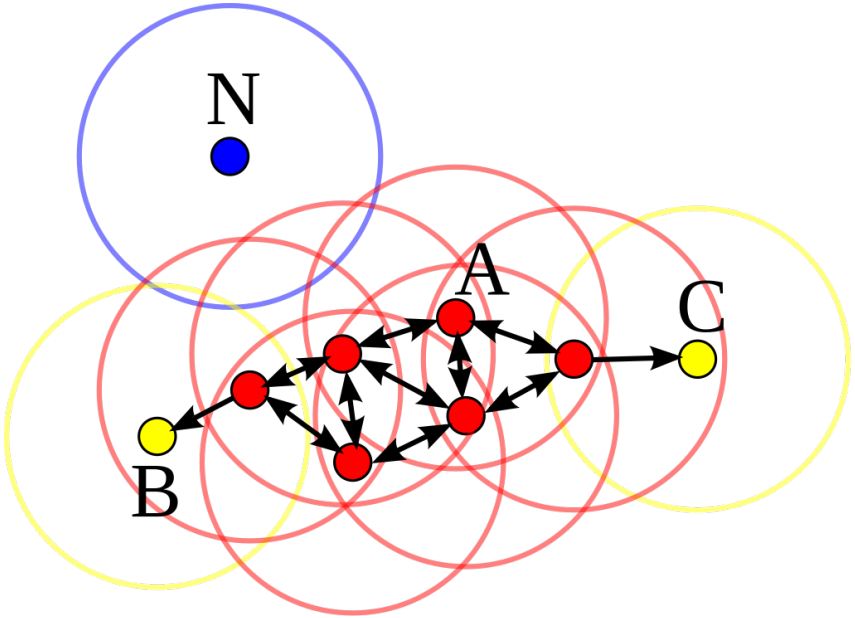

Figure 17: In this diagram, $minPts = 4$. Point A and the other red points are core points, because the area surrounding these points in an ε radius contain at least 4 points (including the point itself). Because they are all reachable from one another, they form a single cluster. Points B and C are not core points, but are reachable from A (via other core points) and thus belong to the cluster as well. Point N is a noise point that is neither a core point nor directly-reachable. [11]

DBSCAN on the sensors values of the cooling system

The **values** in the dataframes are **normalized over both the axes** (the time and the values), such that it is easier for the algorithm to create the clusters. To capture the rapid evolution of some sensors the **timestamps values** have been **scaled up by a factor of 15**, such that the values of the time could be represented in a larger space and have less importance in the creation of the clusters.

The **dbscan parameters** are $\epsilon = 0.8$ and $min\_sample = 250$, moreover all the clusters that do not respect equation 2 are considered as anomalous.

$$num\_sample\_in\_cluster > \frac{len(df)}{30} \tag{1}$$

DBSCAN on the sensors values of the cooling system

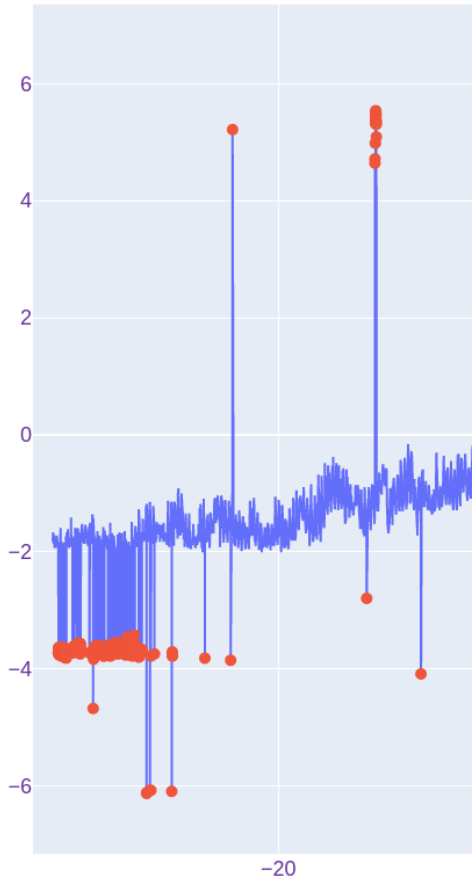

Figure 18: Anomalies detected by DBSCAN on the refrigerator unit 2 **Power Consumption values** [kW].

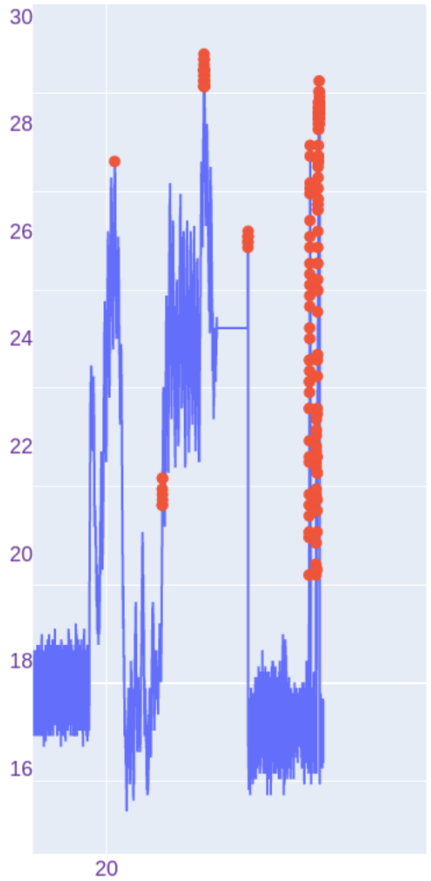

Figure 19: Anomalies detected by DBSCAN on the sensor that measures the **incoming temperature** to the refrigerator unit 4 [°C].

DBSCAN on the electrical implant and the UPS system

The **values** in the dataframes are **normalized over both the axes** (the time and the values), such that it is easier for the algorithm to create the clusters. Here the **timestamps values** are been **scaled up by a factor of 6**.

The **dbscan parameters** are $\epsilon = 0.5$ and $min\_sample = 5$, moreover all the clusters that do not respect equation 2 are considered as anomalous.

$$num\_sample\_in\_cluster > \frac{len(df)}{100} \tag{2}$$

DBSCAN on the sensors values of the electrical implant

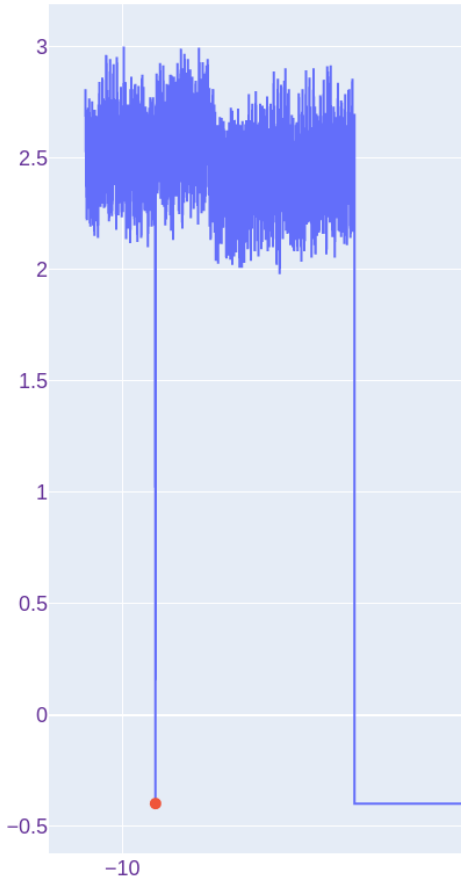

Figure 20: Anomalies detected by DBSCAN on the transformer 3 **current phase 1 values** [A].

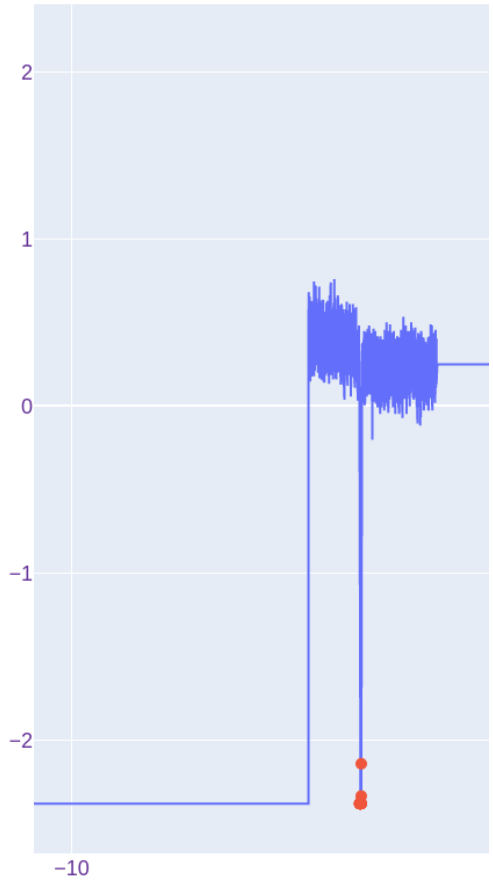

Figure 21: Anomalies detected by DBSCAN on the transformer 2 **apparent power values** [kW].

DBSCAN on the sensor values of the UPS system

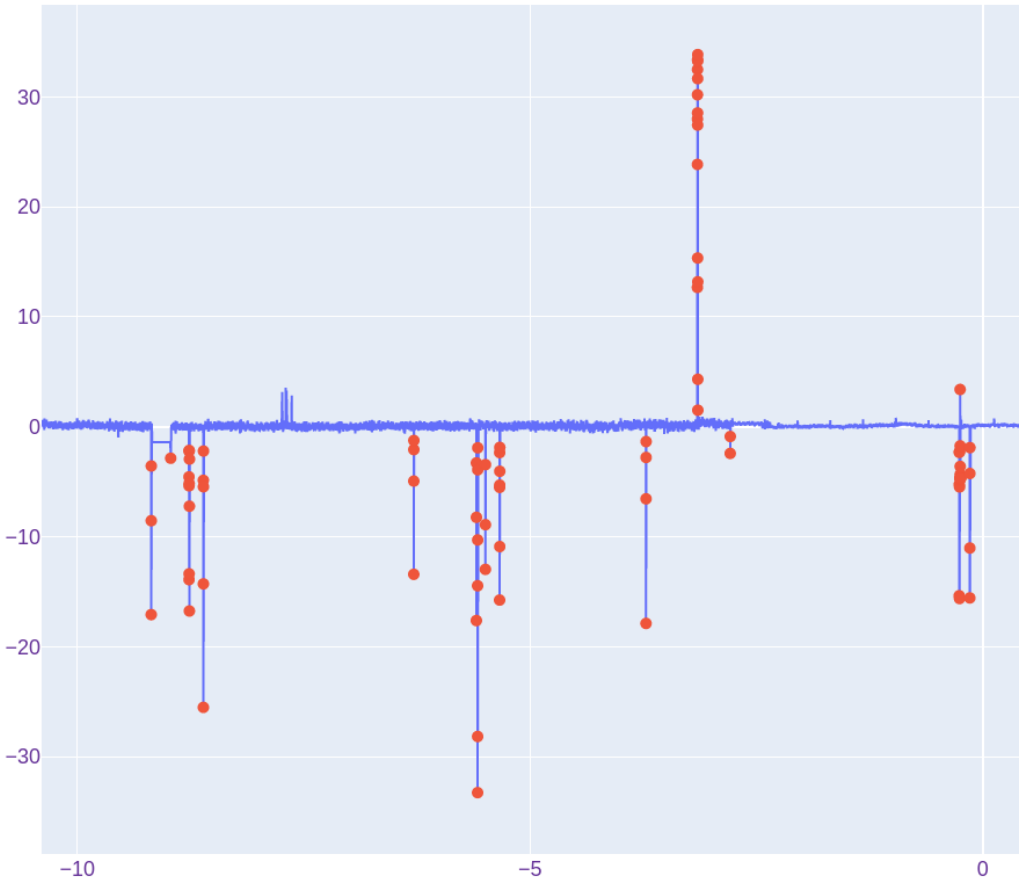

Figure 22: Anomalies detected by DBSCAN on the **ups system sensor values** [kW].

Deep Learning approach to detect the anomalies

What is Deep Learning?

Deep learning is a method that uses artificial neural networks with many layers to learn from data. A layer is a collection of neurons that process input data and produce output data. It enables computers to learn from large amounts of data and make predictions or decisions based on that data.

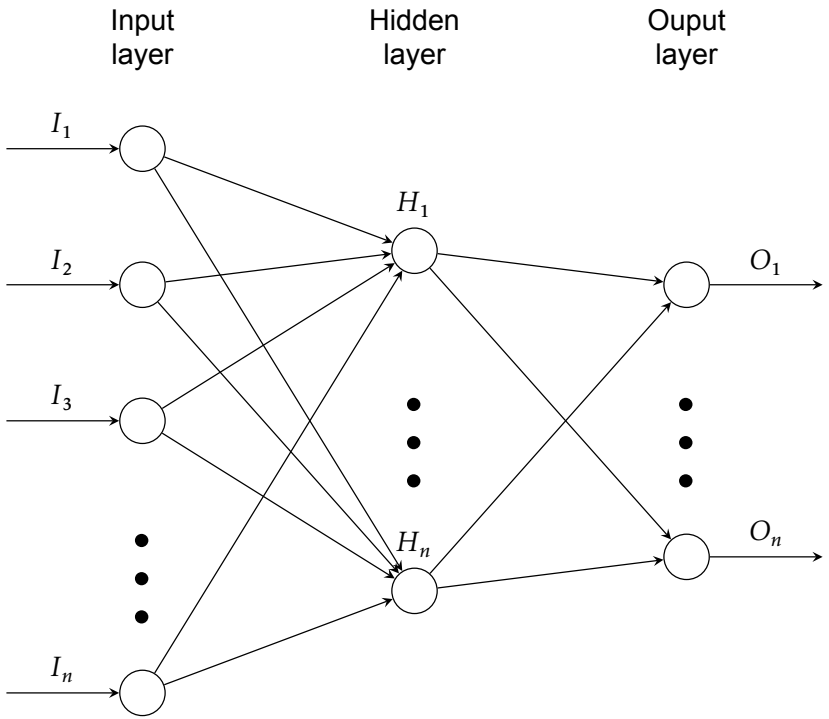

Figure 23: Example of neural network.

What a layer do?

$$f(x;\theta) = \Phi(W_1 x + b_1)$$

Φ is a **non-linear function**, called **activation function**, the most used is the ReLU.

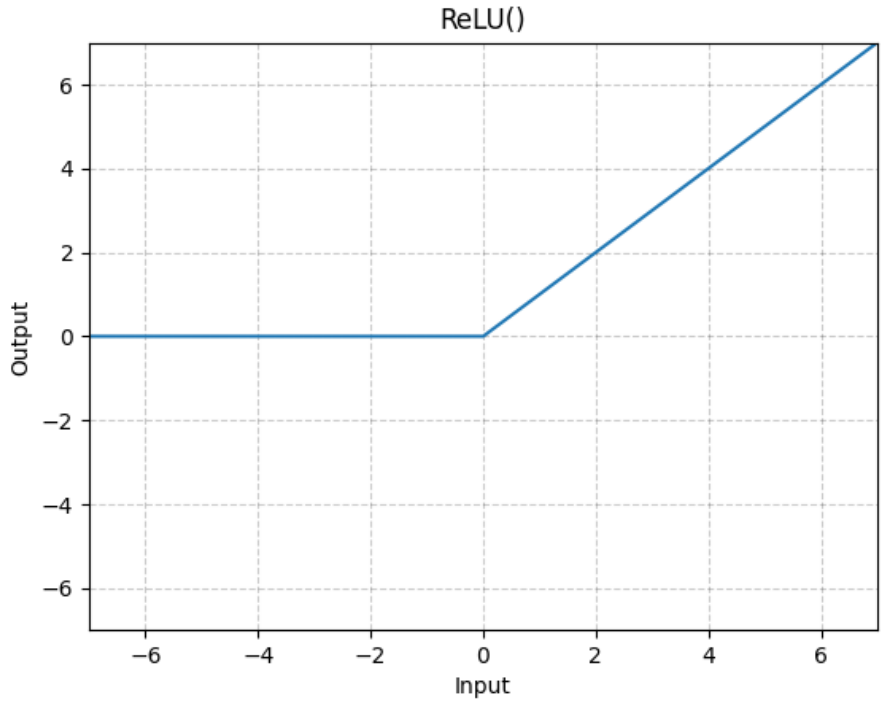

Figure 24: Rectified linear unit activation function.

Supervised and Unsupervised learning

To make the network learn usually two different approaches are used (depending on the available data and on the task):

- supervised learning
- unsupervised learning

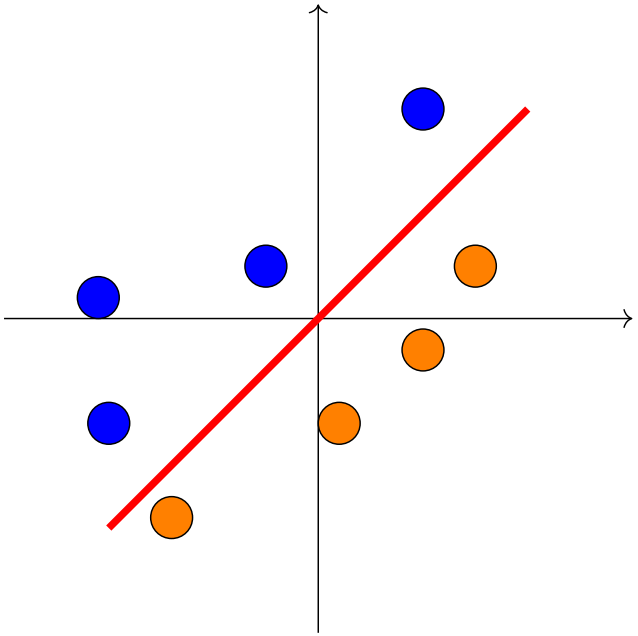

Figure 25: Example of binary classification.

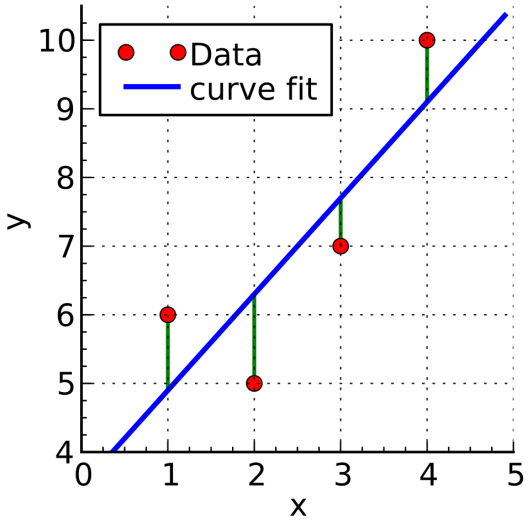

Figure 26: Example of linear regression [12].

Construction Dataset

To construct the dataset to train a deep network, we need to **merge the values** of all the sensors together.
Moreover we need to know if each **row is anomalous or not**.

The data are not labelled, so the results given by dbscan (+ the results of the Z-score method for certain sensors) are used to assign a **label** to each sensor.

A row is considered anomalous if at least one sensor in each block has anomalous values or at least three sensors overall have anomalous values.

In this phase only the **cooling system sensors** and the **electrical implant sensors** are considered.

Dataset

All the sensors values are merged together in a unique dataframe.

Index	Timestamp	TR3_curr_1	…	Temp_OUT_RU1	anomaly
0	2022-01-06 21:41:03	6410	...	18	0
1	2022-01-06 21:56:03	6360	...	18	0
2	2022-01-06 21:56:03	5820	...	20	1
...	...	...	...	...	...
52455	2023-07-07 10:11:03	0	...	18	0
52456	2023-07-07 10:26:03	0	...	18	0

Table 1: Final dataframe built. There are 52457 rows and 69 columns. **50568 rows are not anomalous and 1889 are anomalous**. The sensors used are 66.

Graph Neural Network

What is a graph?

The graph is an ordered binary group $G = (V, E)$, V denotes the set of feature nodes and E denotes the set of edges.

- **Definition (Neighborhood Node)**
  Let $v \in V$ and $w \in V$ denote two feature nodes, and $e = (v, w) \in E$ denotes an edge pointing from v to w. The neighbor nodes of the feature node v is defined as $N(v) = \{w \in V | (v, w) \in E\}$.

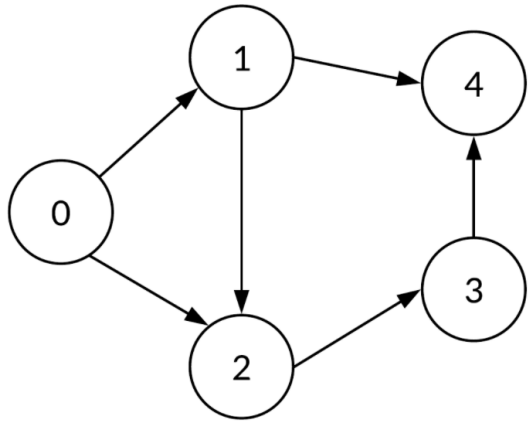

Figure 27: Example of directed graph [13].

What is a Graph Neural Network?

It is a particular type of neural network that tries to merge together the information of the nodes in the neighborhood to obtain a new representation of the node itself.
This method is called message passing:

$$z_i = \Phi\left(Wx_i + \sum_{j \in N(i)} Wx_j\right)$$

But how could it be used in our task?

Graph neural network-based anomaly detection in multivariate time series

The network

This is a Graph Neural Network designed to handle time series [14].

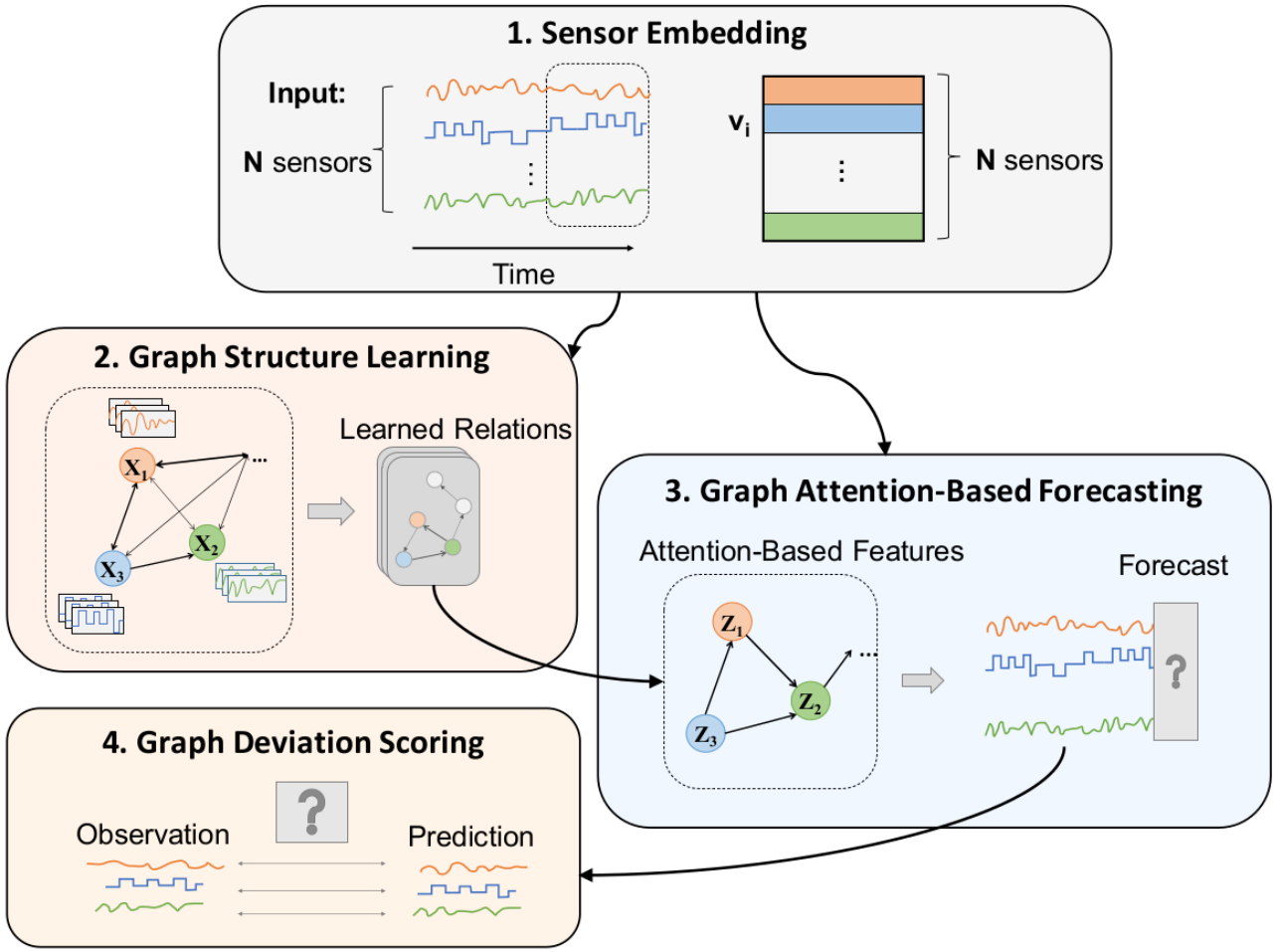

Figure 28: Overview of the framework (the code is available here).

1. Sensor Embedding

The first component produces an **embedding vector** $v_i \in \mathbb{R}^d$ for each sensor, representing the characteristics of the sensor itself.

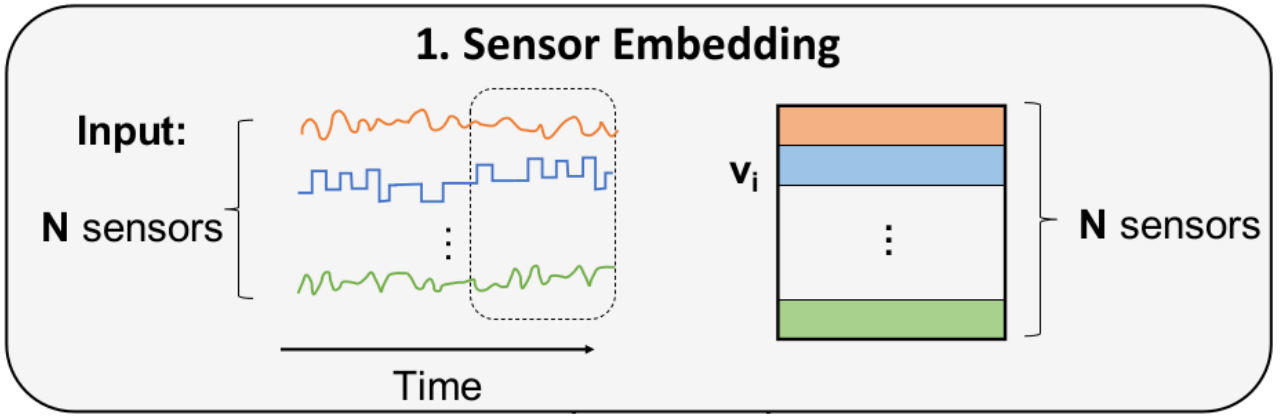

Figure 29: Sensor Embedding module.

2. Graph Structure Learning

The second component constructs a **directed graph**, whose nodes represent sensors, to model the relationship between them.

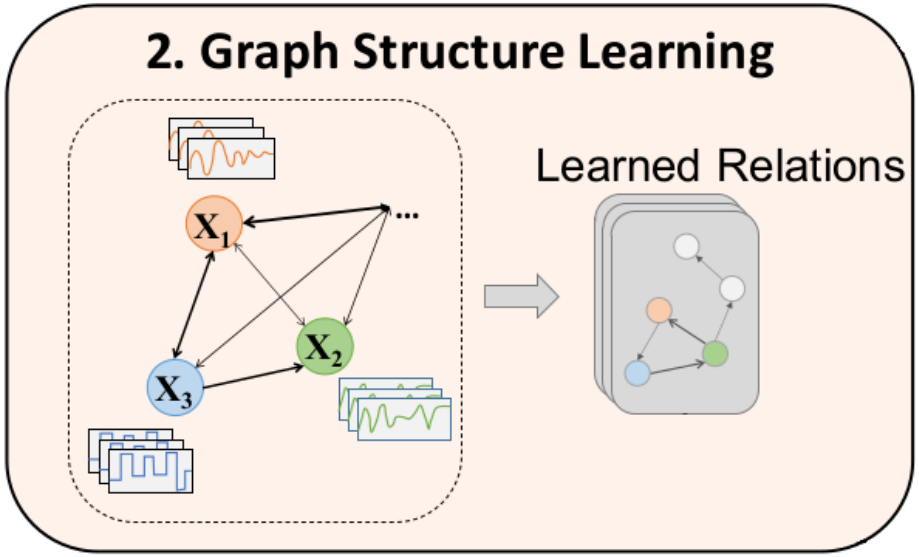

Figure 30: Graph Structure Learning module.

3. Graph Attention-Based Forecasting

The expected behavior of each sensor, at time t, is forecasted using a sliding window, of size w, over the historical time series data $x^{(t)} \in \mathbb{R}^{Nxw}$.

Moreover the graph computed in the previous component is used to merge the information of the nodes in the neighborhood, using the message passing mechanism.

$$z_i^{(t)} = \Phi\left(\alpha_{i,i} W x_i^{(t)} + \sum_{j \in N(i)} \alpha_{i,j} W x_j^{(t)}\right)$$

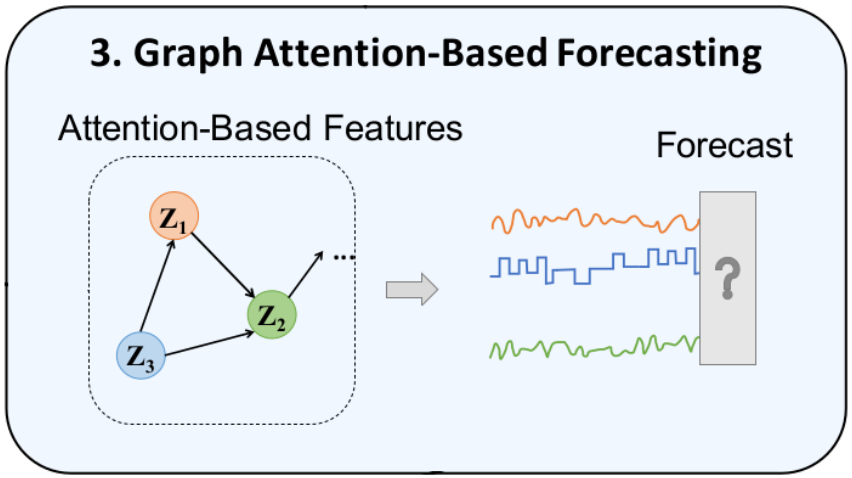

Figure 31: Graph Attention-Based Forecasting module.

4. Graph Deviation Scoring

Then the expected and the observed behaviors at time t are compared.

$$Err_i(t) = |s_i^{(t)} - \hat{s}_i^{(t)}|$$

To compute the overall anomalousness at time tick t, these values are aggregated using the **max function**.

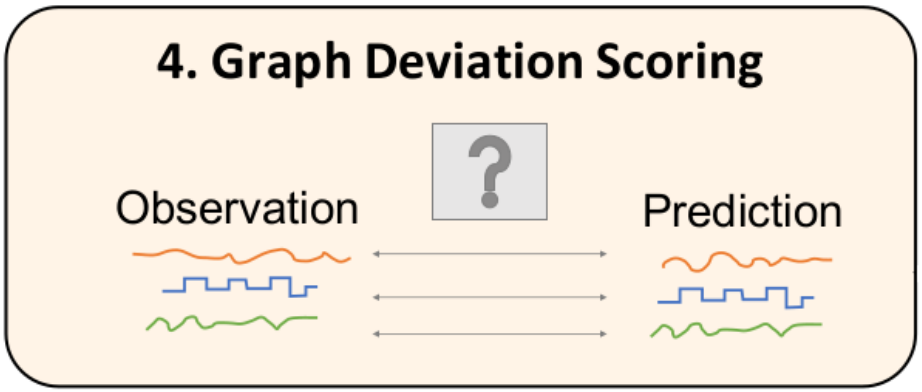

Figure 32: Graph Deviation Scoring module.

Test on GDN architecture

To use this network the dataset is divided in two different dataframes, one used for training and another one used for testing.

At the end of the training process it is possible to evaluate the results.

The dataset is unbalanced, so the metric used for evaluation is the F1-Score:

- Precision (Prec)
- Recall (Rec)
- **F1-Score** (F1)

$$\frac{TP}{TP + FP}$$

$$\frac{TP}{TP + FN}$$

$$\frac{2 \cdot Prec \cdot Rec}{Prec \cdot Rec}$$

At the end of the tests done trying to tune the parameters of this network, I obtained a score of **0.57** (the minimum is 0 and the maximum is 1).

Problems of this architecture

- After the calculation of the error for each sensor, it retrieve the maximum and uses a fixed threshold
- This architecture use a layer called **GAT**, and in the work called *How Attentive are Graph Attention Networks?* [15] it is shown that it has some problem during the learning phase.
  They propose a new formulation (**GATv2**):

$$z_i' = \alpha_{i,i} W x_i + \sum_{j \in N(i)} \alpha_{i,j} W x_j$$

$$\alpha_{i,j} = \frac{exp(LeakyReLU(W[x_i || x_j]))}{\sum_{k \in N(i) \cup i} exp(LeakyReLU(W[x_i || x_k))}$$

New Architecture

New Architecture

I tried to define a new architecture that could overcame these problems.

I build two networks:

- a **regression network** that must learn the behavior of the sensors, to predict the next future values
- a **classification network** that given the predicted and the observed behaviors, could output if the actual timestamp is anomalous or not

Regression network

A bit inspired by the work of Xie et al. [16], I define this architecture with two main blocks:

- one able to capture the relations with other sensors (spatial operation)
- one able to capture the trend of the sensor itself, considering the time series (temporal operation)

To do it I used the **GATv2** layer and the **Long-Short term memory** layer [17].

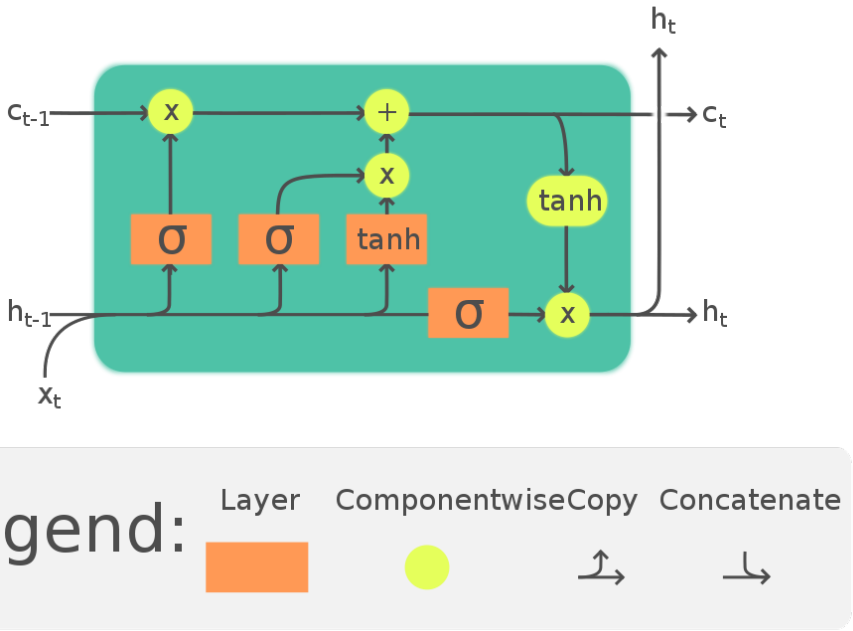

Figure 33: Long-Short term memory layer [18].

Regression network

Before passing the values of the sliding window to this part of the network, we need to define an **embedding** to obtain a new representation of the sensors in a larger space. This is done by a **Linear Layer**.

During the performing of the spatial operations, to avoid that information coming from different timestamps are merged together, n different GATv2 are used, **one for each timestamp** in the sliding window.

For the same reason, when computing the temporal operation, 66 different LSTM layers are used, such that each one could be easily learn to the behavior of **one single sensor**.

At the end the activation of the LSTM layers is processed by a last **Linear Layer** that will output the results.

Regression network

As introduced in the paper of Ioffe and Szegedy [19], an important component of a neural network is the normalization, such that the training is faster and less influenced from the initial random initialization of the parameters.

Here it is used a *Batch Normalization* after each GATv2 layer and after each LSTM layer.

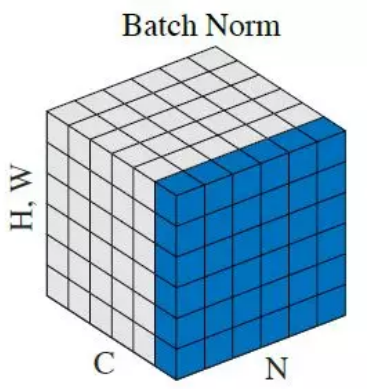

Figure 34: Batch Normalization [19].

Classification network

Once that the regression network has output the expected behavior of the sensors, we could compare it with the observed one.

To do it i defined a simple neural network with only **two Linear layers**.
After the first one there is a *Batch Normalization* and a *ReLU*.

The network outputs two numbers between 0 and 1, representing the probability that the timestamp is anomalous and the probability that it is not anomalous.

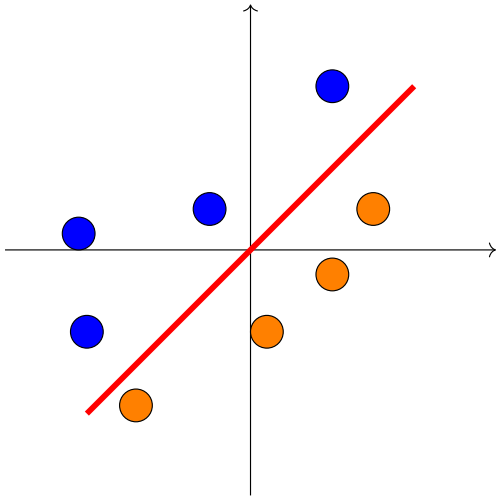

Figure 35: Example of binary classification.

Dataset

To train the first network we must use only the not anomalous timestamps, instead to train the classifier we will use both.

To avoid that there could be holes between timestamps, we cannot random sample the dataset to create the train, validation and test sets. Moreover it is better if the ratio between not anomalous and anomalous timestamps is preserved in all the sets.

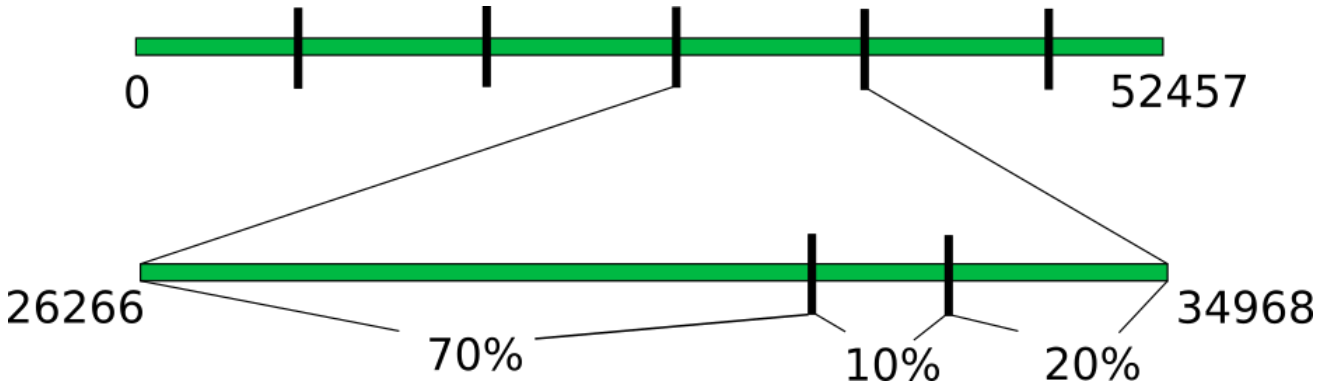

Figure 36: The whole dataframe is divided in 6 equally long parts. Then each part is divided in three parts: the first 70% is for the training set, the second 10% is for the validation set and the last 20% is for the test set. Then these parts are merged with the respective parts of the other division to create the three sets.

Training

The first network has been trained using as loss function the Mean Squared Error.

$$MSE = \frac{1}{n}\sum_{i=0}^{|n|-1}(Y_i - \hat{Y}_i)^2$$

For the second one the Cross Entropy is used.

$$l_n = -w_n \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C}\exp(x_{n,c})}$$

where w_n is the weight given to the class n.
Such that the network is able to properly learn from this unbalanced dataset, the minority class is weighted 10 times more than the majority class

Results

The regression network is able to predict the behavior of the sensor with a mean squared error of 0.025.

The classifier reaches an F1-Score of 0.997 on the test set.

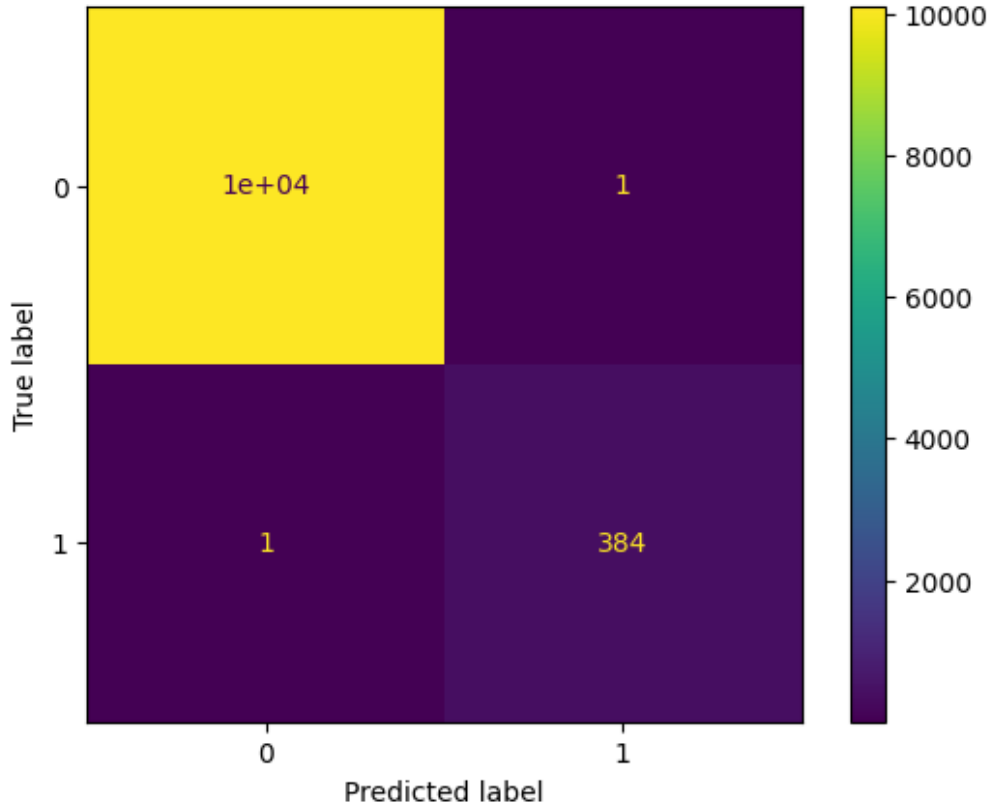

Figure 37: Confusion matrix of the classifier testing part. **The F1-Score is 0.997**.

Ablation Studies

Do we really need this complex architecture?

What if we use only the embedding layer and **only one GATv2** layer (processing the time series as if it is part of the features of the node itself)?

The regression network is able to predict the behavior of the sensor with a mean squared error of 0.029 (+0.004).

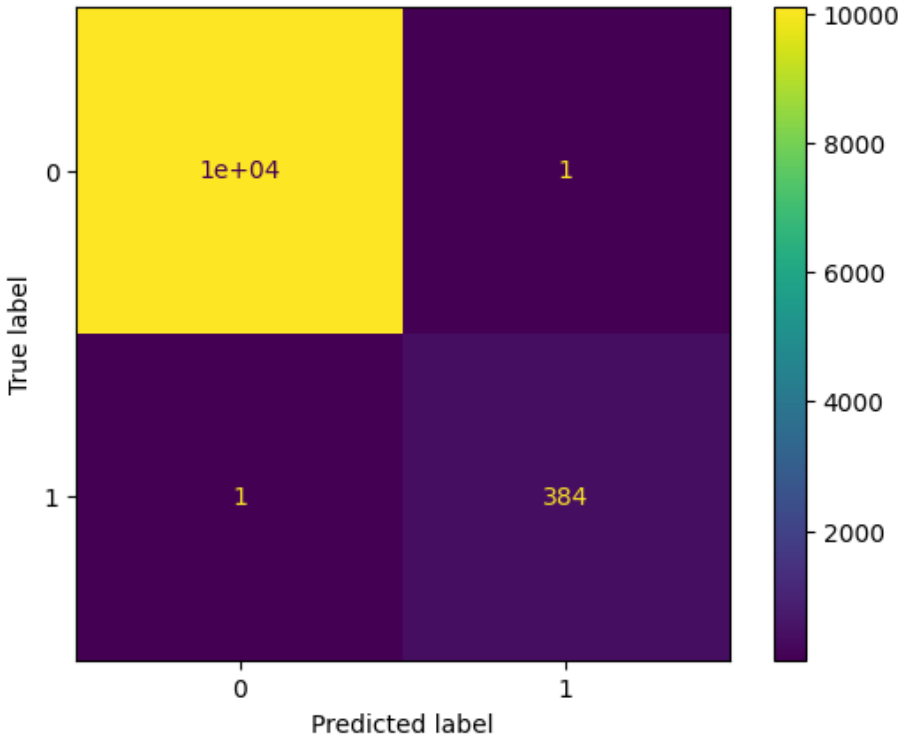

Figure 38: Confusion matrix of the classifier testing part. The F1-Score is 0.997.

Do we really need to use a graph neural network?

What if we use only the embedding layer, **a single LSTM** layer and an output layer?
The regression network is able to predict the behavior of the sensor with a mean squared error of 0.025 (+0.000).

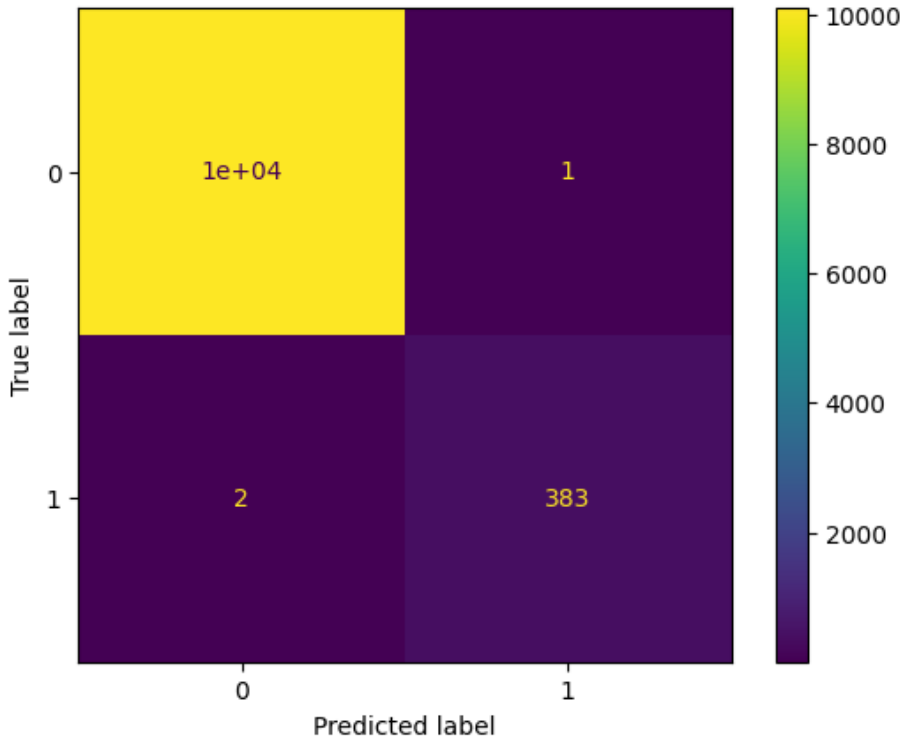

Figure 39: Confusion matrix of the classifier testing part. The F1-Score is 0.996.

Do we really need something more than Linear Layers?

What if we use only the embedding layer and an output layer?
The regression network is able to predict the behavior of the sensor with a mean squared error of 0.008 (-0.021).

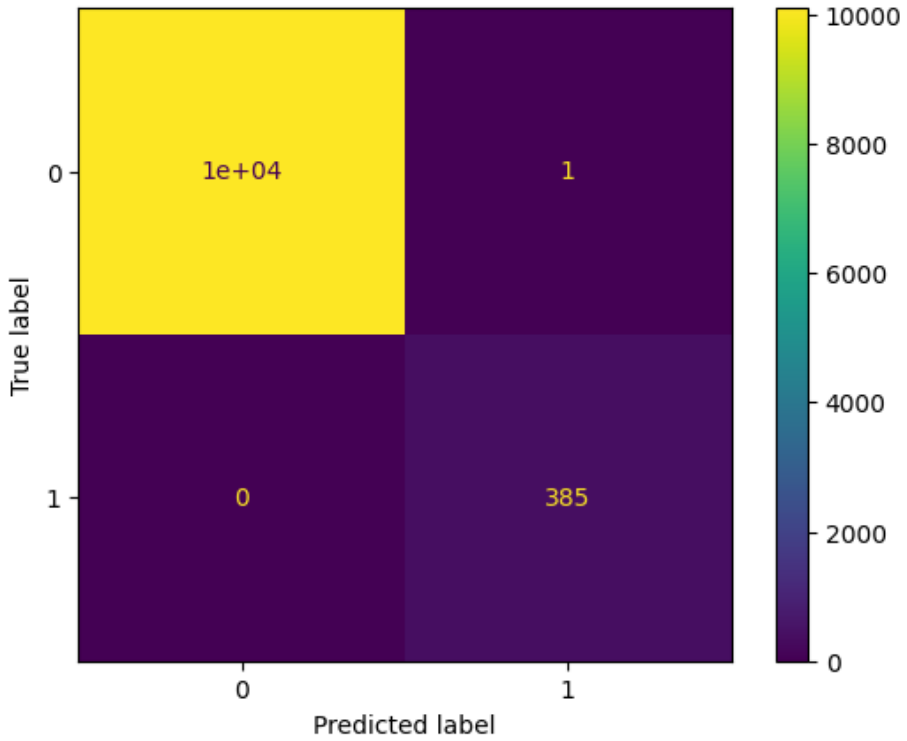

Figure 40: Confusion matrix of the classifier testing part. The F1-Score is 0.997.

Conclusions

Conclusions

We don’t need a so complex network to achieve good results in this task, probably because the behavior of the sensors do not change frequently during time, so it is simple to predict them.
Moreover **the label are assigned algorithmically**, so the task is simplified.

The results of all these networks are near to the perfection.

Moreover with the weights used in the Cross Entropy function it is able to learn also from an unbalanced dataset like this one (only the 3.6% of the rows are considered as anomalous!).

Thanks for the attention!


The pandas development team.
pandas-dev/pandas: Pandas, February 2020.


J. D. Hunter.
Matplotlib: A 2d graphics environment.
Computing in Science & Engineering, 9(3):90–95, 2007.


Michael L. Waskom.
seaborn: statistical data visualization.
Journal of Open Source Software, 6(60):3021, 2021.


Plotly Technologies Inc.
Collaborative data science, 2015.


Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors.
SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python.
Nature Methods, 17:261–272, 2020.


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, 12:2825–2830, 2011.


Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux.
API design for machine learning software: experiences from the scikit-learn project.
In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.


Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala.
Pytorch: An imperative style, high-performance deep learning library, 2019.


Rob Hyndman and Yanan Fan.
Sample quantiles in statistical packages.
The American Statistician, 50:361–365, 11 1996.


Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu.
A density-based algorithm for discovering clusters in large spatial databases with noise.
In *Proc. of 2nd International Conference on Knowledge Discovery and*, pages 226–231, 1996.


Wikimedia Commons.
Dbscan-illustration, 2011.


Krishnavedala.
Linear_least_squares_example2.
https://en.wikipedia.org/wiki/File:Linear_least_squares_example2.svg#filelinks, 10/6/2011.
Accessed: 2023-07-14.


https://www.cs.mtsu.edu/~xyang/3080/images/adjDirectedGraph.png.
Accessed: 2023-07-11.


Ailin Deng and Bryan Hooi.
Graph neural network-based anomaly detection in multivariate time series, 2021.


Shaked Brody, Uri Alon, and Eran Yahav.
How attentive are graph attention networks?, 2022.


Lingqiang Xie, Dechang Pi, Xiangyan Zhang, Junfu Chen, Yi Luo, and Wen Yu.
Graph neural network approach for anomaly detection.
Measurement, 180:109546, 2021.


Sepp Hochreiter and Jürgen Schmidhuber.

Long short-term memory.
Neural computation, 9:1735–80, 12 1997.

Wikimedia Commons.

Lstm_cell, 2021.

Sergey Ioffe and Christian Szegedy.

Batch normalization: Accelerating deep network training by reducing internal covariate shift.
In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.