POLITECNICO
MILANO 1863

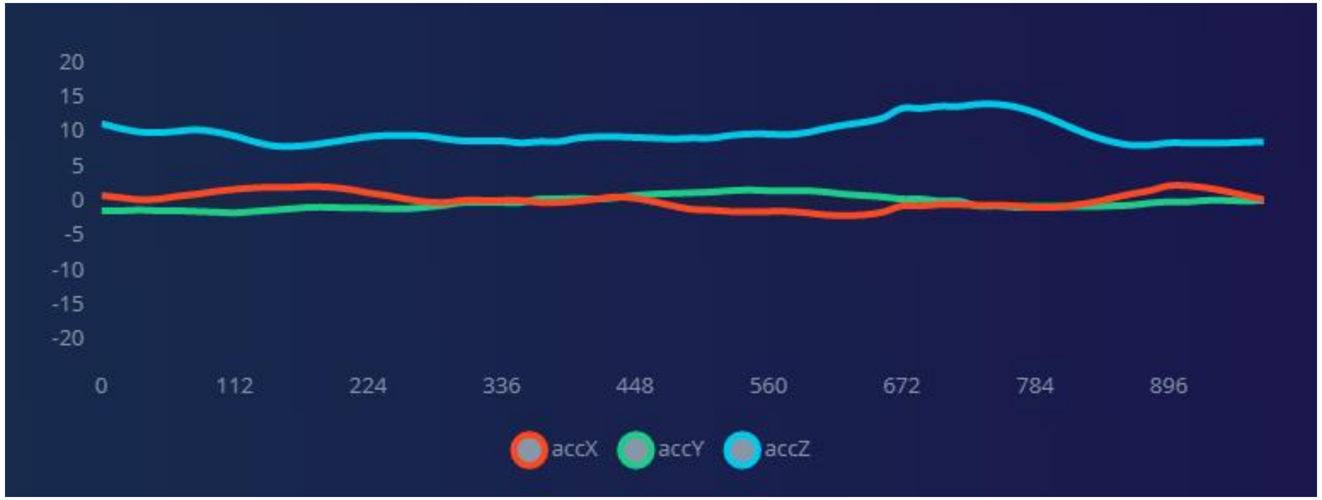
Hardware Architectures for Embedded and Edge AI

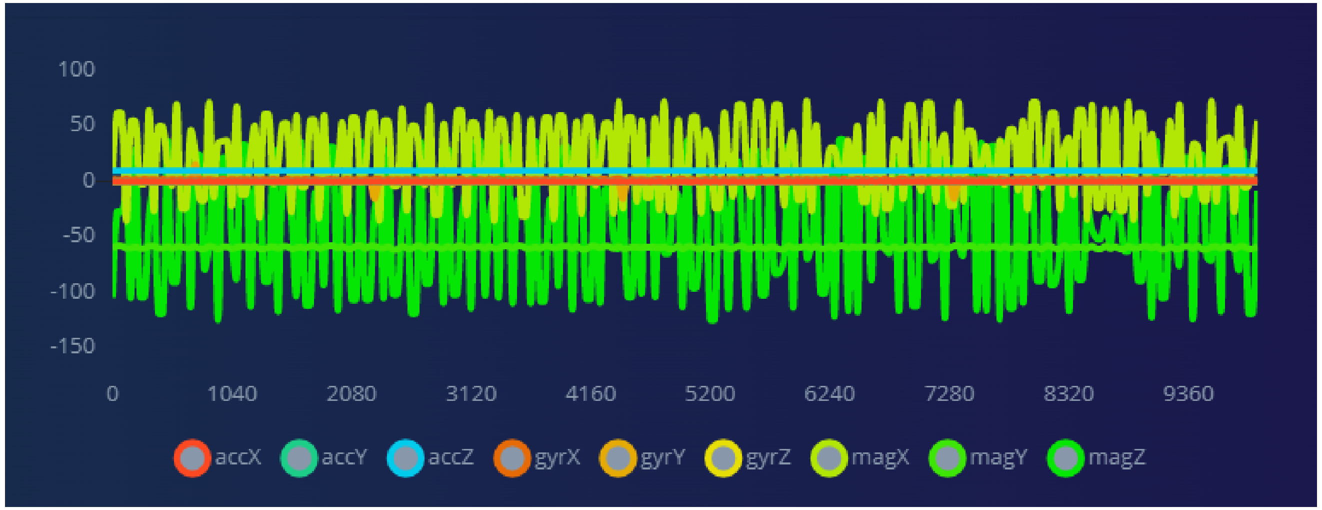
Prof Manuel Roveri – manuel.roveri@polimi.it
Massimo Pavan – massimo.pavan@polimi.it

Exercise session 9 – Continous motion recognition (with IMU) and Anomaly Detection

Motion and vibration: Accelerometers and IMU

- Accelerometers measure the acceleration over three axis (the three spatial dimensions)
- IMUs contain an accelerometer, but also a giroscope and a magnetometer
- It depends on the application how many of these sensors/axis you should include in the model

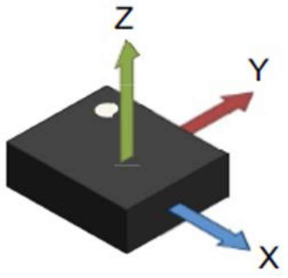




LSM9DS1 (9 axis IMU) description

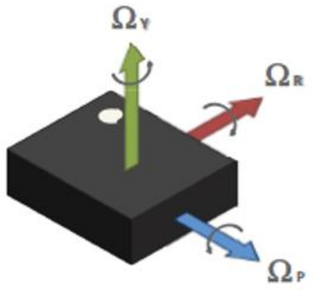
Accelerometer:

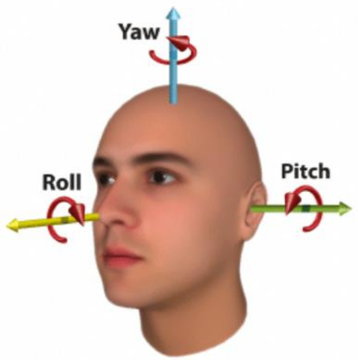
- used to measure the acceleration in m/s2 (meters per second per second) or g's (gravities [about 9.8 m/s2])
- its scale can be set to either ± 2, ± 4, ± 8, or ± 16 g



Gyroscope:

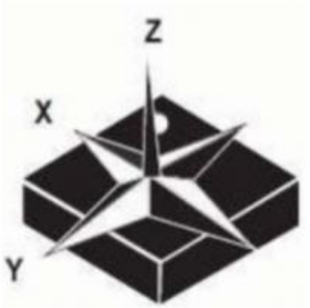
- used to measure the angular velocity in degrees per second (usually abbreviated to DPS or °/s).
- its scale can be set to either to ± 245 ± 500 or ± 2000 DPS



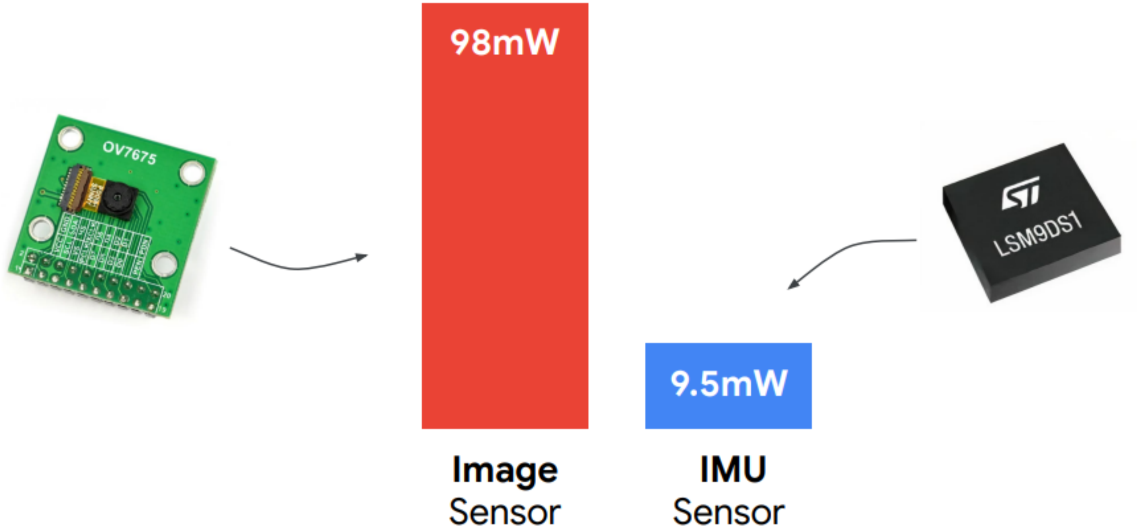


Magnetometer:

- measures the power and direction of magnetic fields in units of gauss (Gs)
- its measurement scale to either ± 4, ± 8, ± 12, or ± 16 Gs.



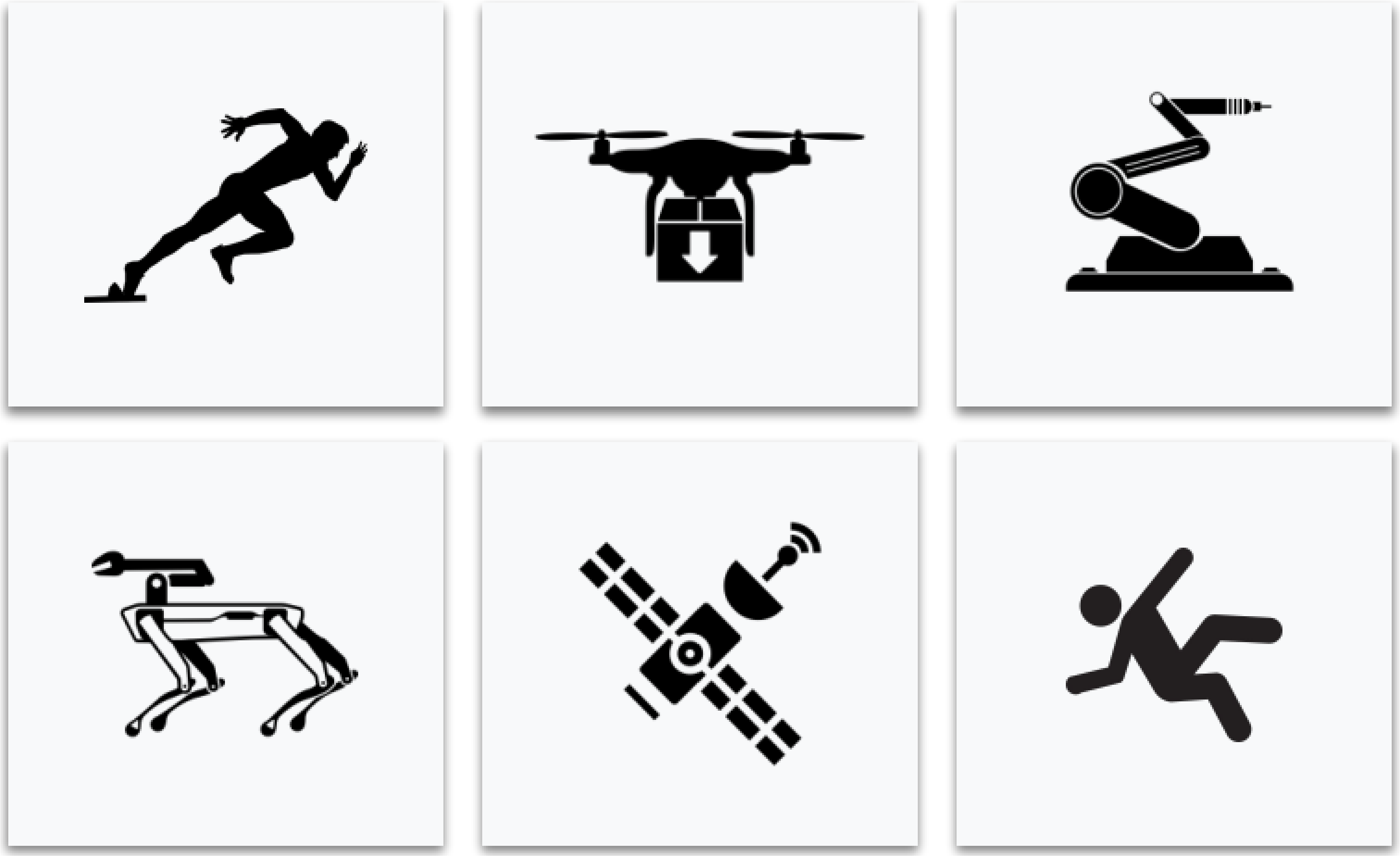
Why IMUs are so interesting?



Low Power consumption

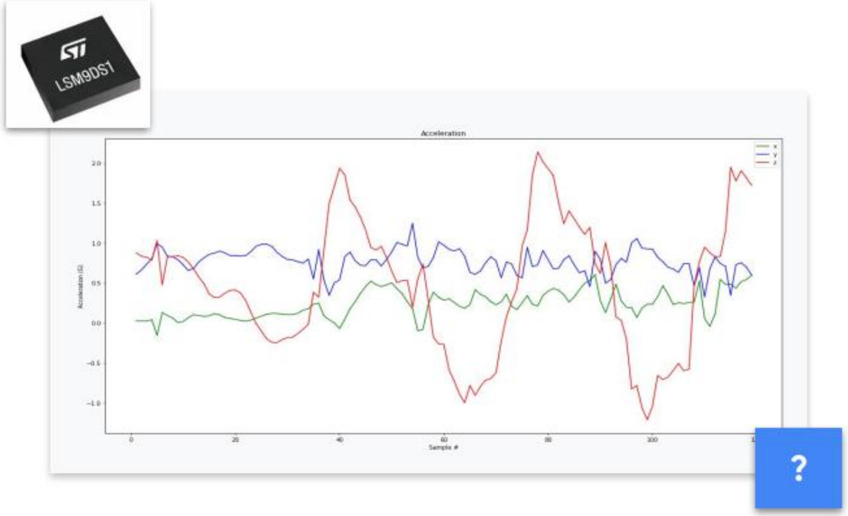
Size, Weight, Price

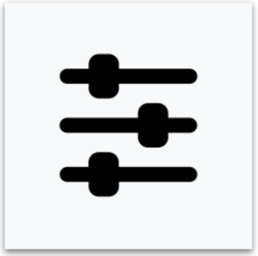
Applications of IMUs



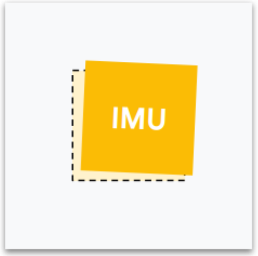
Challenges

- Interpretability:
  - Easy to understand and label pictures, but time series?
- Sensor drift:
  - Over time, the sensitivity and baselines of the sensor changes
- Deployment sensitivity:
  - Each sensor is slightly different from the other, and this may cause problems in the development of ML algorithms





Parameter Tuning



Misalignment

POLITECNICO
MILANO 1863

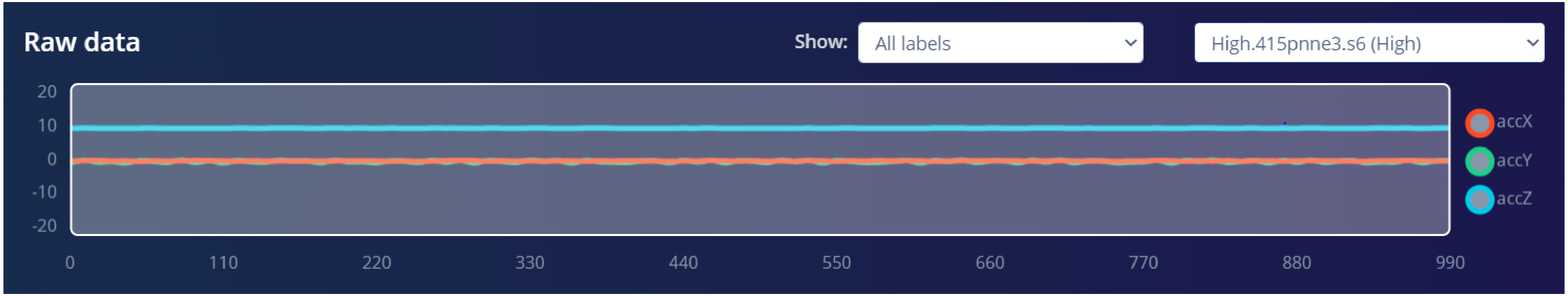
Industry application example: fan working mode classification and anomaly detection

Use case example

- Old industrial fan, not connected to the internet
- It runs faster under some circumstances
- It's possible to monitor its working mode (L – M – H) through vibration

- Write a tinyML algo to classify the working mode through IMU

Input Data

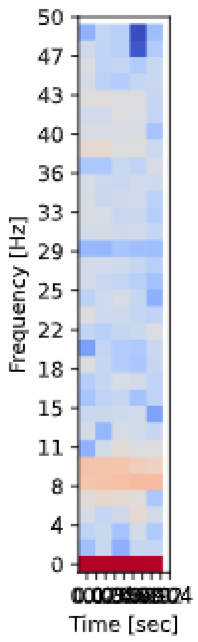


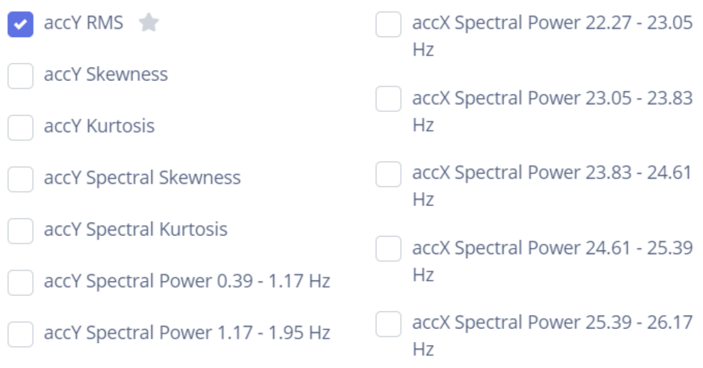
- 10 seconds recording for each class
- Splitted into 1 second batches
- AccY seem to be the most meaningful axis
- 30 seconds per working mode, + 30 seconds «Off»

Preprocessing

Two possible ways:

- Spectrogram
    - Similar to what was done with audio, compute the STFT (short time fourier transform)
    - Only one axis as input
    - 2D output
- Spectral features
    - Apply a filter (low pass, high pass)
    - Compute FFT and some statistical properties (RMS, kurtosis…)
    - Multiple axis as input
    - 1D output, possible to select just the most meaningful





Classifier

A very simple classifier should be good for this kind of problem:

- 2D Convolutional if you are analizing spectrogram
- Fully connected for the extracted spectral features

- Both models and pipelines seem to work pretty good

Model Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 100.0%

LOSS 0,00

Confusion matrix (validation set)

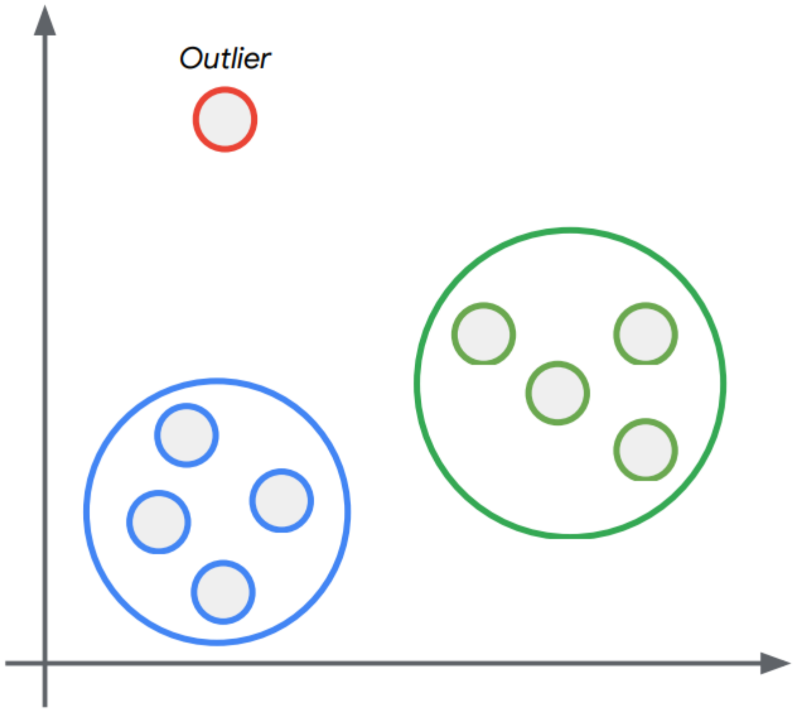
	HIGH	LOW	MEDIUM	OFF
HIGH	100%	0%	0%	0%
LOW	0%	100%	0%	0%
MEDIUM	0%	0%	100%	0%
OFF	0%	0%	0%	100%
F1 SCORE	1.00	1.00	1.00	1.00

POLITECNICO
MILANO 1863

Anomaly Detection

Anomaly detection - definition

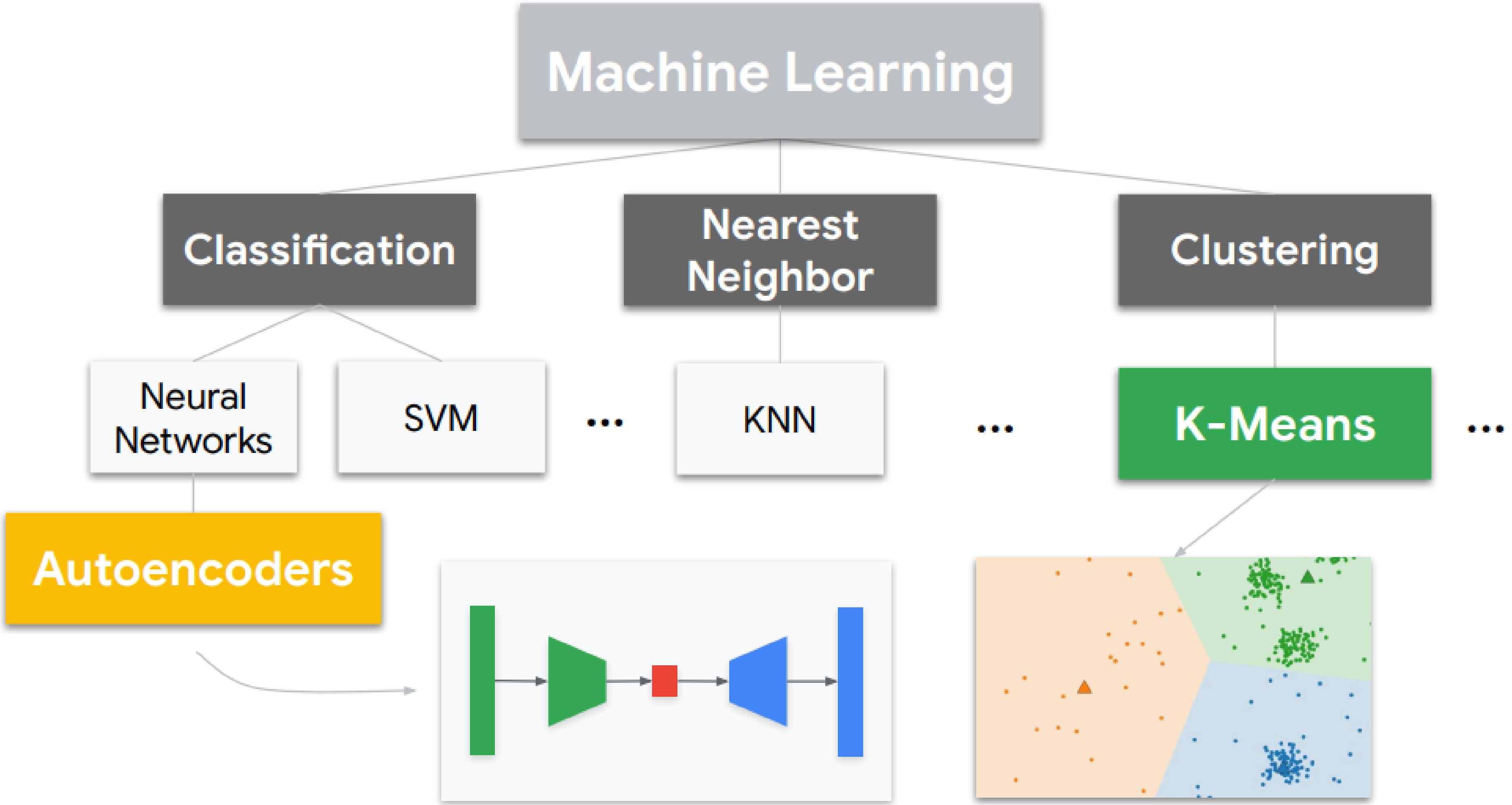
In data analysis, anomaly detection is the identification of **rare items, events or observations** which raise suspicions because they are differing significantly from the majority of the data.



Possible applications of anomaly detection

Health	Industry	Security
ECG Sensor	Accelerometer	IR Motion Sensor

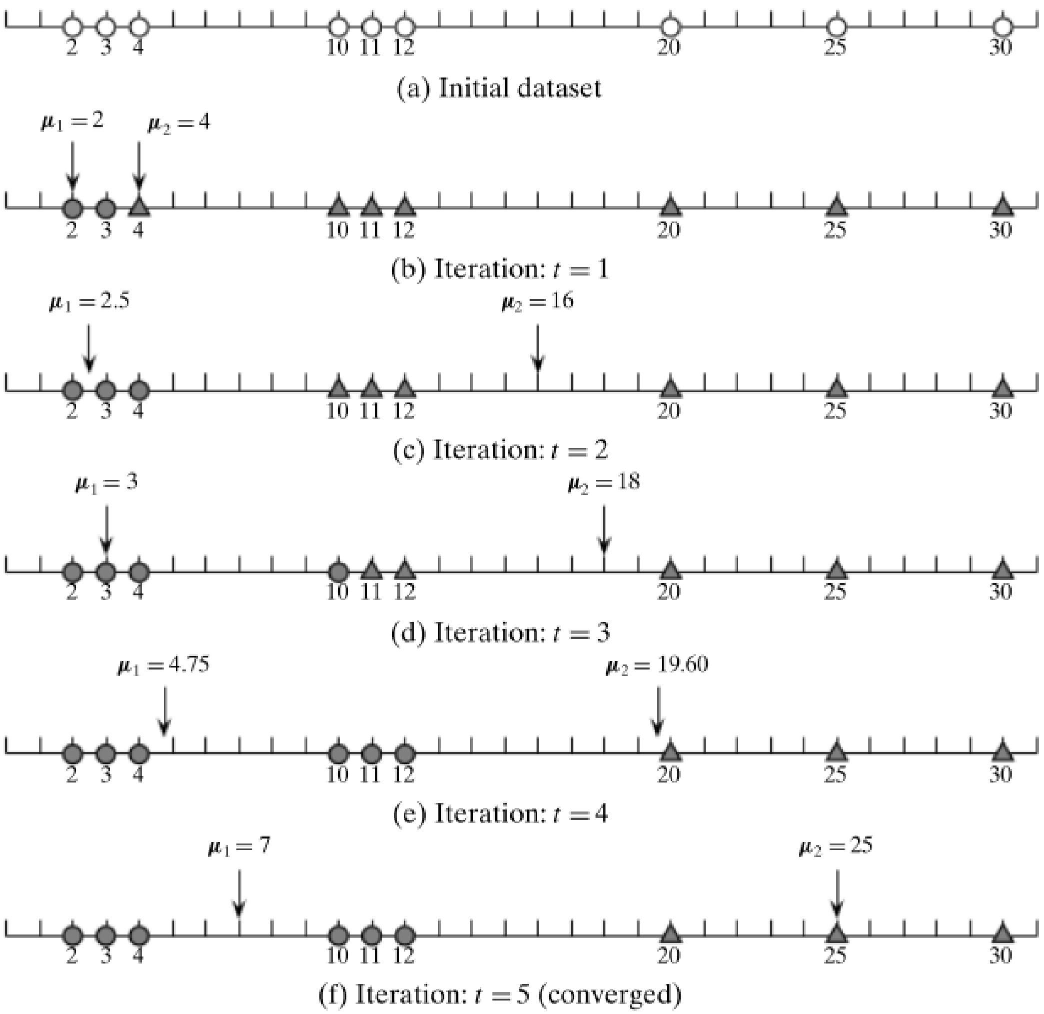
Especially in AD, Neural Networks are not the only option



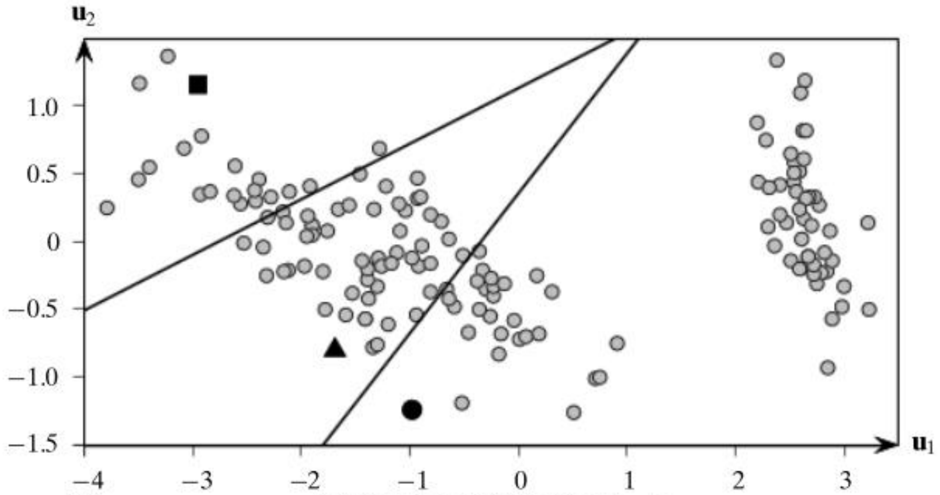
K-means: clustering

- Given a dataset of N instances, and a desired number of clusters k, this class of algorithms generates a partition C of N instances (data points) in k clusters {C1, C2, ..., C_k}
- Greedy iterative approach to find a clustering that minimizes the SSE objective:

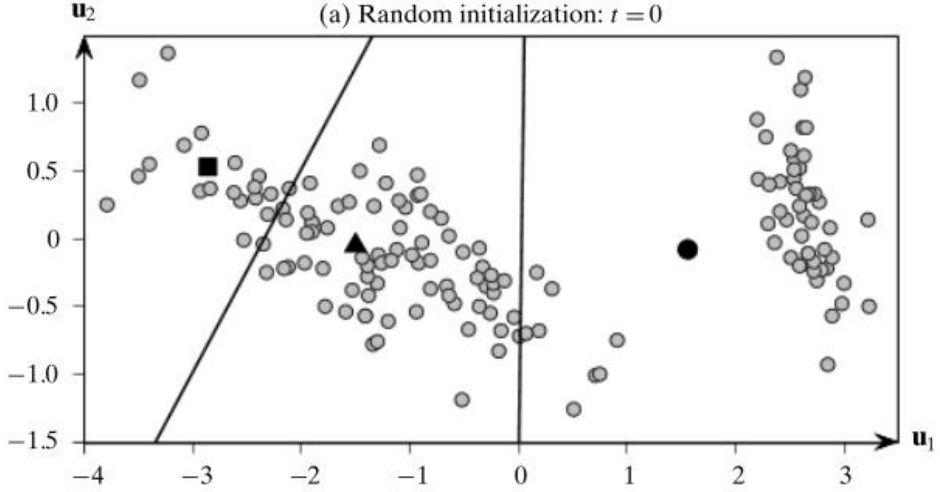
$$\mathrm{SSE}(C) = \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_i||^2$$



2Dimensions

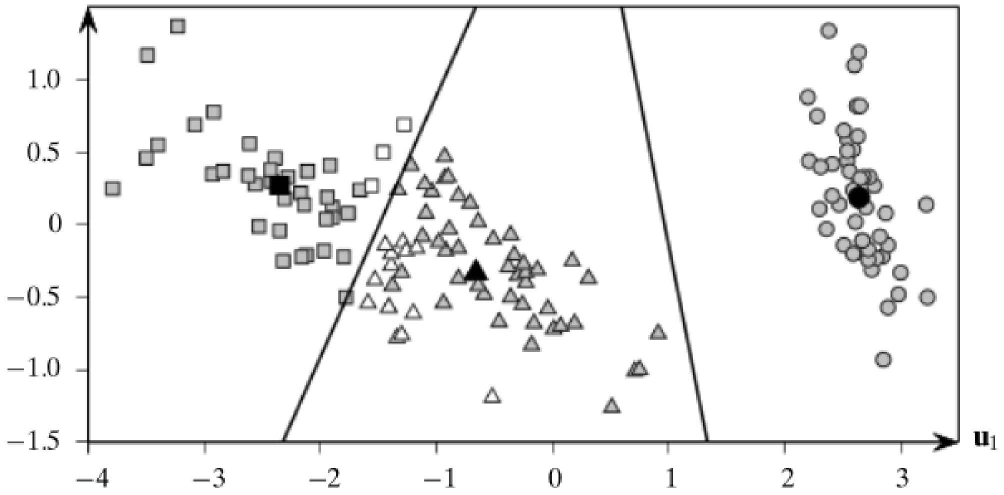


(a) Random initialization: $t = 0$



(b) Iteration: $t = 1$

(b) Iteration: $t = 1$



(c) Iteration: $t = 8$ (converged)

K-means pseudo code

Algorithm 13.1: K-means Algorithm

K-MEANS $(\mathbf{D}, k, \epsilon)$:

$t = 0$

Randomly initialize k centroids: $\boldsymbol{\mu}_1^t, \boldsymbol{\mu}_2^t, \ldots, \boldsymbol{\mu}_k^t \in \mathbb{R}^d$

repeat

  $t \leftarrow t + 1$

  $C_i \leftarrow \emptyset$ for all $i = 1, \cdots, k$

  `// Cluster Assignment Step`

  foreach $\mathbf{x}_j \in \mathbf{D}$ **do**

    $i^* \leftarrow \arg\min_i \left\{ \|\mathbf{x}_j - \boldsymbol{\mu}_i^{t-1}\|^2 \right\}$

    $C_{i^*} \leftarrow C_{i^*} \cup \{\mathbf{x}_j\}$ `// Assign` $\mathbf{x}_j$ `to closest centroid`

  `// Centroid Update Step`

  foreach $i = 1, \cdots, k$ **do**

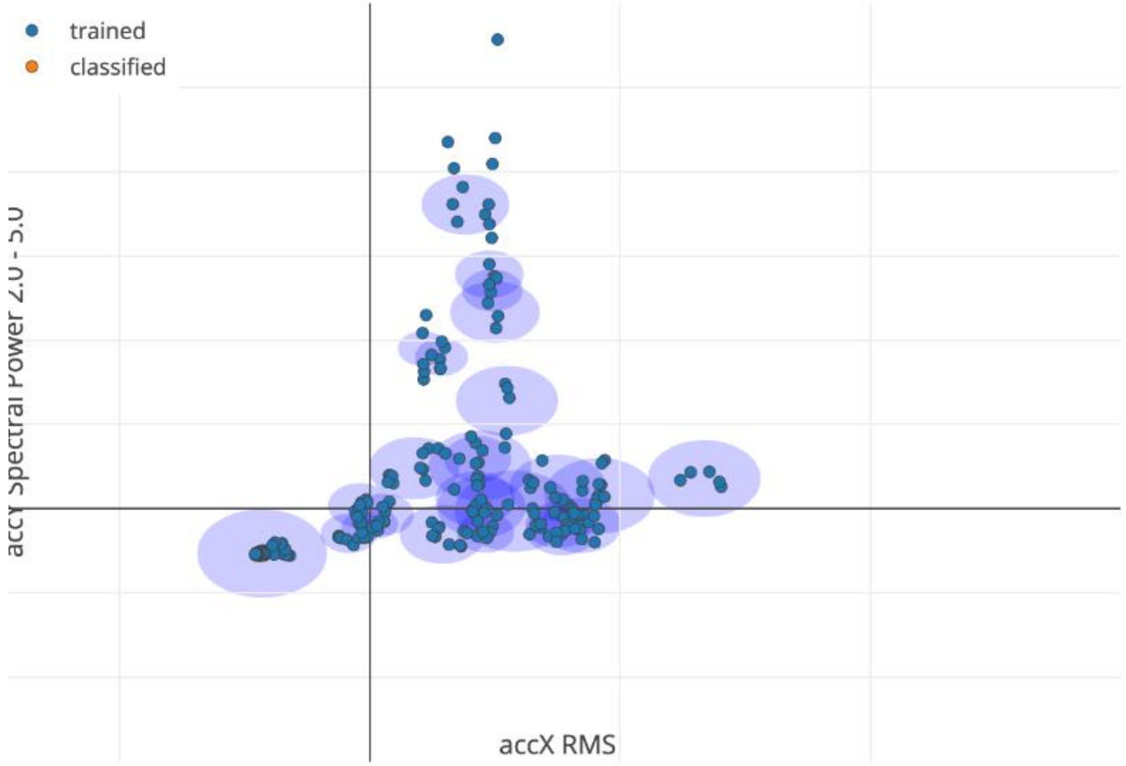
    $\boldsymbol{\mu}_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$

until $\sum_{i=1}^{k} \|\boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1}\|^2 \leq \epsilon$

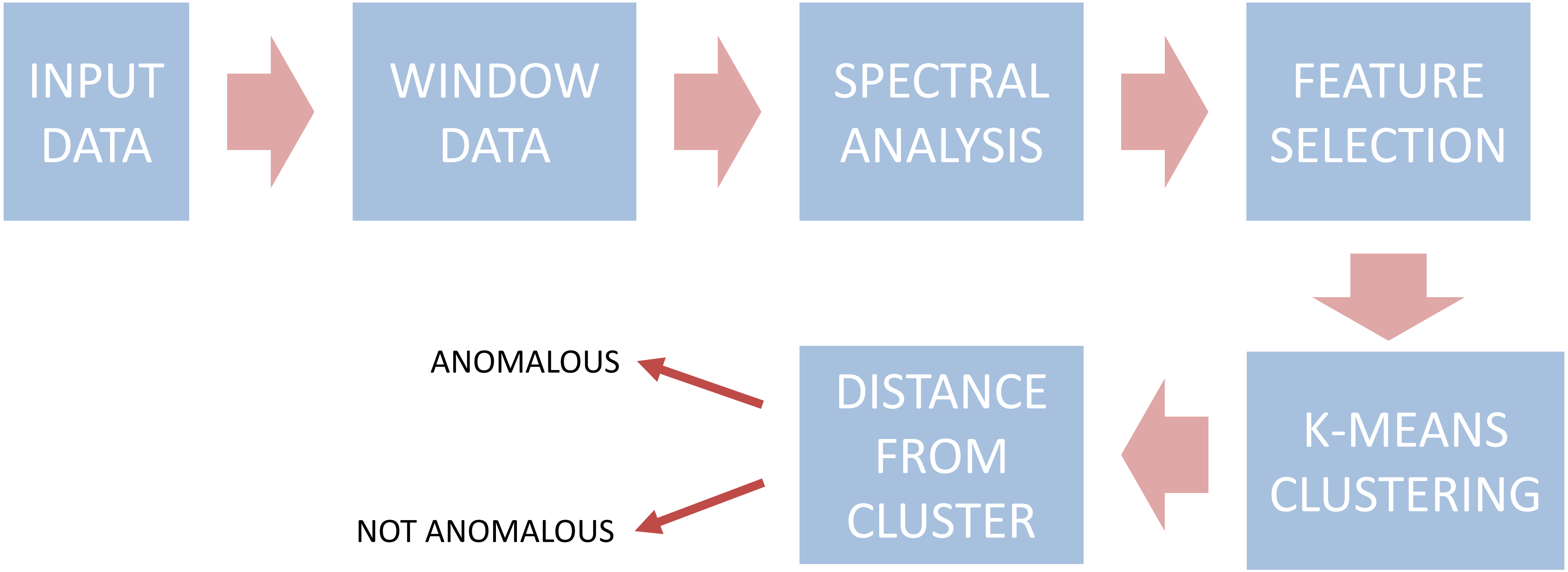
"Data Mining and Machine Learning" by Zaki & Meira - Chapter 13

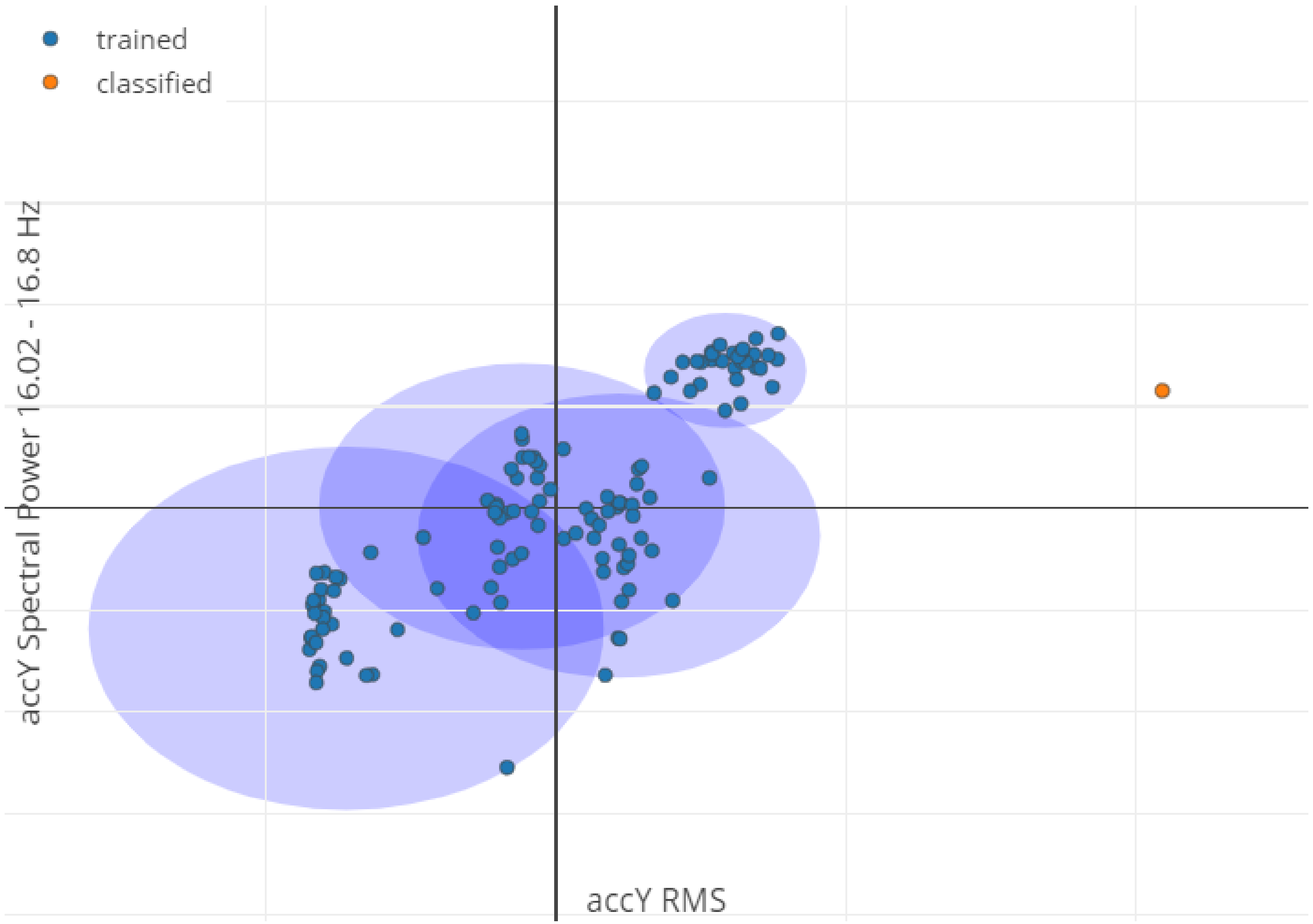
K-means for anomaly detection

- It is possible to compute also a «dimension» for each cluster (based on the variance of the point inside of the cluster), not just the centroid
- If the new data point is far from the boundaries of the cluster, it is considered an anomaly



Fan Fault detection




trained
classified
accY Spectral Power 16.02 - 16.8 Hz
accY RMS

Testing – anomaly detection and classification at the same time

TIMESTAMP	HIGH	LOW	MEDIUM	OFF	ANOMALY
1800	0.29	0.02	0.08	0	-0.26
2000	0.84	0.02	0	0.13	0.41
2200	0.33	0	0	0.67	2.47
2400	0.94	0	0	0.06	3.27
2600	0.95	0	0	0.05	3.34
2800	0.99	0	0	0.01	3.39
3000	0.80	0	0	0.20	2.30
3200	0.95	0	0	0.05	0.01
3400	0.06	0.16	0.73	0.05	-0.53

Deployment

- Gather the data from the sensor and put them into the input buffer

```
// roll the buffer -3 points so we can overwrite the last one
numpy::roll(buffer, EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, -3);

// read to the end of the buffer
IMU.readAcceleration(
    buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE - 3],
    buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE - 2],
    buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE - 1]
);
```

```
// copy the buffer
memcpy(inference_buffer, buffer, EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE * sizeof(float));

// Turn the raw buffer in a signal which we can the classify
signal_t signal;
int err = numpy::signal_from_buffer(inference_buffer, EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &signal);
if (err != 0) {
    ei_printf("Failed to create signal from buffer (%d)\n", err);
    return;
}
```

Deployment

- Run pre-processing and inference
- Run statistics on execution time

```
err = run_classifier(&signal, &result, debug_nn);
if (err != EI_IMPULSE_OK) {
    ei_printf("ERR: Failed to run classifier (%d)\n", err);
    return;
}

// print the predictions
ei_printf("Predictions ");
ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
    result.timing.dsp, result.timing.classification, result.timing.anomaly);
ei_printf(": ");

// ei_classifier_smooth_update yields the predicted label
const char *prediction = ei_classifier_smooth_update(&smooth, &result);
ei_printf("%s ", prediction);
```

Results

COM6

Send

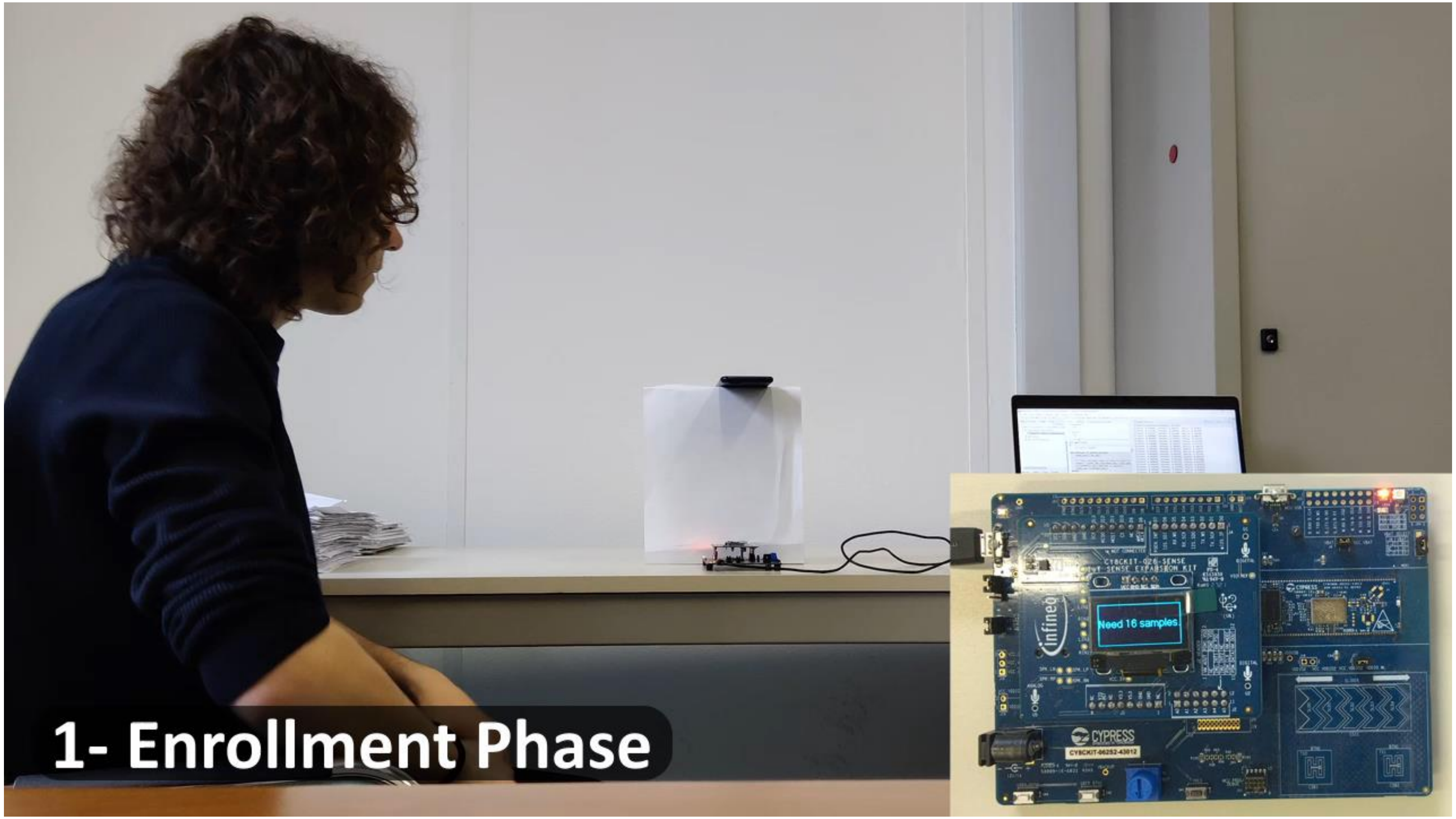
```
Predictions (DSP: 13 ms., Classification: 0 ms., Anomaly: 0 ms.): Medium  [ 0, 0, 8, 0, 2, 0, ]
Predictions (DSP: 13 ms., Classification: 0 ms., Anomaly: 0 ms.): Medium  [ 0, 0, 9, 0, 1, 0, ]
Predictions (DSP: 15 ms., Classification: 0 ms., Anomaly: 0 ms.): Medium  [ 0, 0, 9, 0, 1, 0, ]
Predictions (DSP: 15 ms., Classification: 0 ms., Anomaly: 0 ms.): Medium  [ 0, 0, 9, 0, 1, 0, ]
Predictions (DSP: 15 ms., Classification: 0 ms., Anomaly: 0 ms.): Medium  [ 0, 1, 8, 0, 1, 0, ]
Predictions (DSP: 13 ms., Classification: 2 ms., Anomaly: 0 ms.): Medium  [ 0, 2, 8, 0, 0, 0, ]
Predictions (DSP: 13 ms., Classification: 0 ms., Anomaly: 0 ms.): Medium  [ 0, 3, 7, 0, 0, 0, ]
Predictions (DSP: 13 ms., Classification: 0 ms., Anomaly: 0 ms.): uncertain  [ 0, 4, 6, 0, 0, 0,
```

Appendix

Credits and reference

- "TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers", Daniel Situnayake, Pete Warden, O'Reilly Media, Inc.
- Online course:
  - https://www.edx.org/professional-certificate/harvardx-tiny-machine-learning
- A lot more material on TinyML:
  - http://tinyml.seas.harvard.edu/
- Edge impulse guide
  - https://www.edgeimpulse.com/blog/advanced-anomaly-detection-with-feature-importance
- Mohammed J. Zaki, Wagner Meira Jr - Data Mining and Machine Learning Fundamental concept and algorithms
- https://studio.edgeimpulse.com/public/231558/latest

Anomaly detection for other use cases



SV – using «Anomaly detection» concept for speaker verification