MACHINE LEARNING for SENSOR DATA

a very short intro

Faktion

Jos Polfliet, VP of Applied AI

Engineering influencer and personal hero

Check out https://erikbern.com/

THE FUNDAMENTAL PROBLEM

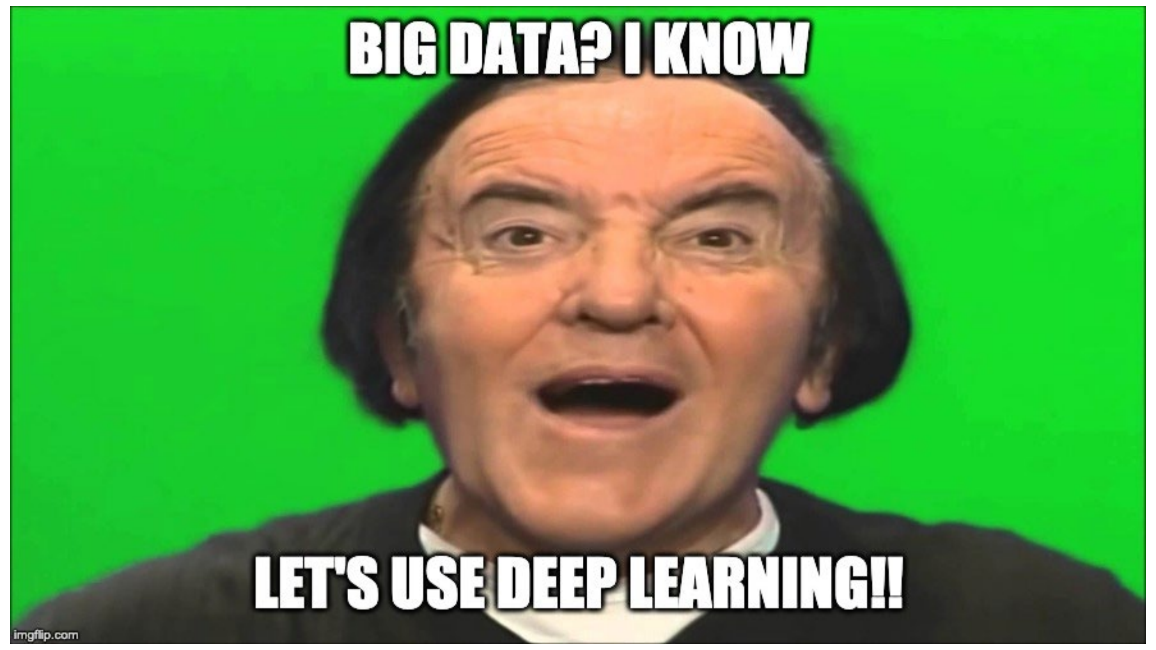
A journey of discovery

What you get

25x Temperature, 25x pressure, 25x flow at 100Hz

75 sensors * 100 Hz * 3600 s * 24 h * 30 days * 3 months = 58,320,000,000 rows

Datetime	Sensor	Value
01/02/2019 00:00:00.00	Temperature 1	120.89121
01/02/2019 00:00:00.01	Temperature 1	120.89118
,,,		
31/05/2019 23:59:59.99	Temperature 1	116.56119
01/02/2019 00:00:00.00	Temperature 2	117.4215
01/02/2019 00:00:00.01	Temperature 2	117.4071
,,,		
31/05/2019 23:59:59.99	Temperature 2	119.14111
…	…	…



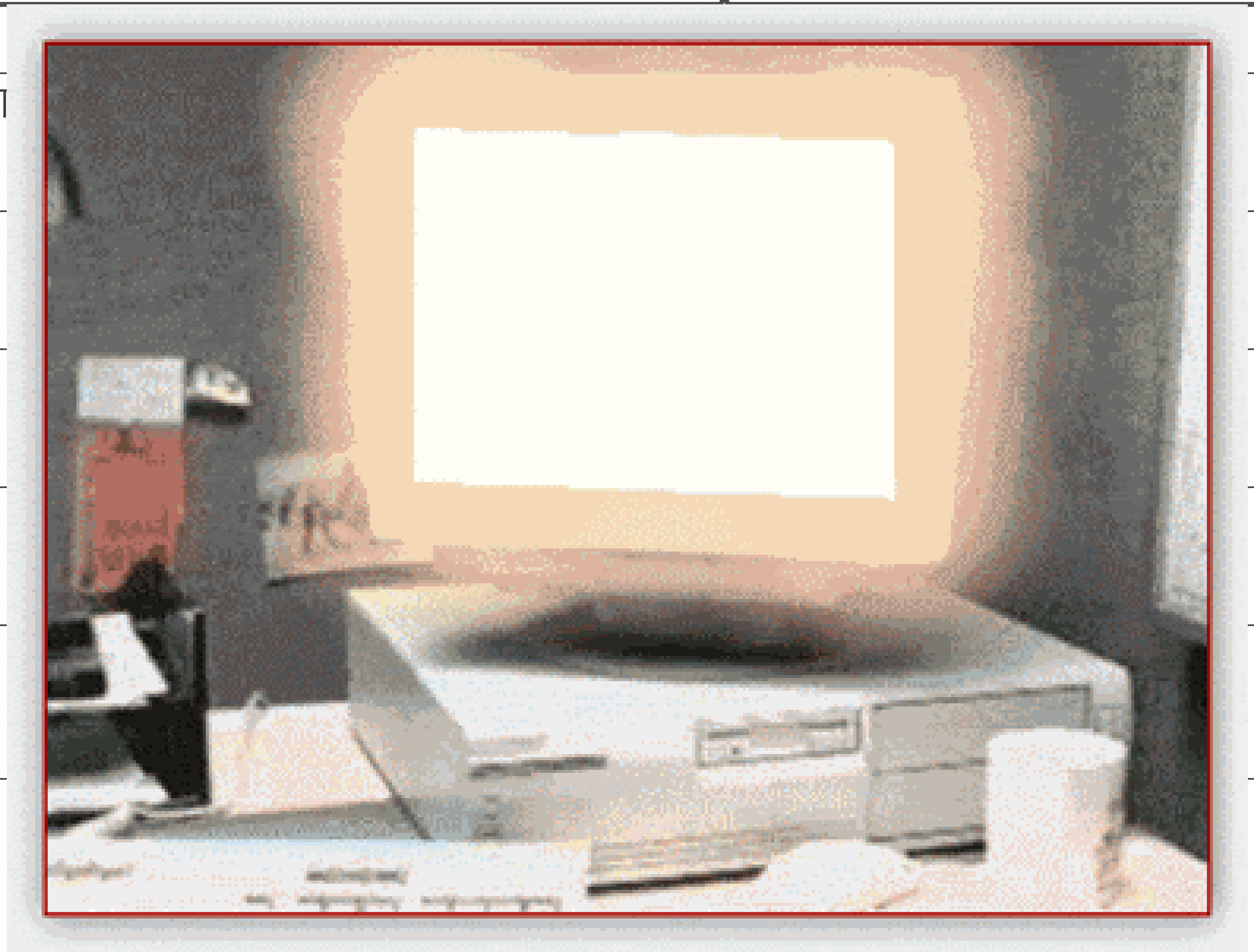
What not to do

25x Temperature, 25x pressure, 25x flow at 100Hz for 1 hour

100 Hz * 3600s * 75 sensors = 27,000,000 columns

24 hours * 30 days * 3 months = 2,160 rows

Datetime	T_0_0	T	…	F_19_600	Target
01/02/2019 00:00					GOOD
01/02/2019 01:00					BAD
31/05/2019 23:00					GOOD



What would you like to learn today?

Learn | Practice | Projects | Pricing | For Business | 0 XP

INTERACTIVE COURSE

Dimensionality Reduction in Python

Start Course For Free

4 hours | 16 Videos | 58 Exercises | 562 Participants | 4,700 XP

Course Description

High-dimensional datasets can be overwhelming and leave you not knowing where to start. Typically, you'd visually explore a new dataset first, but when you have too many dimensions the classical approaches will seem insufficient. Fortunately, there are visualization techniques designed specifically for high dimensional data and you'll be introduced to these in this course. After exploring the data, you'll often find that many features hold little information because they don't show any variance or because they are duplicates of other features. You'll learn how to detect these features and drop them from the dataset so that you can focus on the informative ones. In a next step, you might want to build a model on these features, and it may turn out that some don't have any effect on the thing you're trying to predict. You'll learn how to detect and drop these irrelevant features too, in order to reduce dimensionality and thus complexity. Finally, you'll learn how feature extraction techniques can reduce dimensionality for you through the calculation of uncorrelated principal components.

1 Exploring high dimensional data FREE

0%

You'll be introduced to the concept of dimensionality reduction and will learn when an why this is important. You'll learn the difference between feature selection and feature extraction and will apply both techniques for data exploration. The chapter ends with a lesson on t-SNE, a powerful feature extraction technique that will allow you to visualize a high-dimensional dataset.

VIEW CHAPTER DETAILS

Continue Chapter

Jeroen Boeye

Machine Learning Engineer @ Faktion

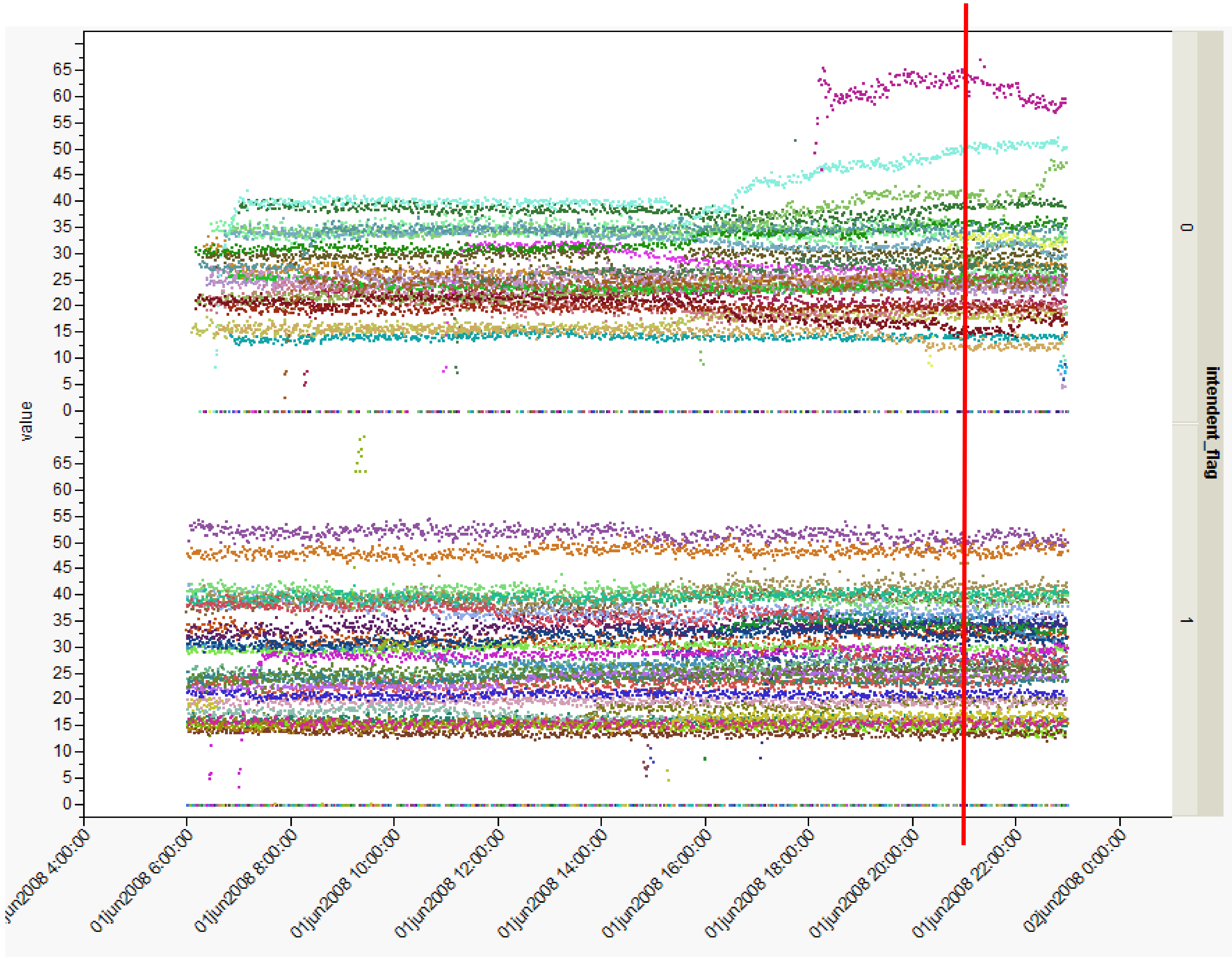
Jeroen is a machine learning engineer working at Faktion, an AI company from Belgium. He uses both R and Python for his analyses and has a PhD background in computational biology. His experience mostly lies in working with structured data, produced by sensors or digital processes.

See More

Pro-tip:
https://www.datacamp.com/courses/dimensionality-reduction-in-python

Time series data mining

Step 1: Prepare timeseries



Step 2: Calculate characteristics

```
for segment in segments:
    for var in segment:
        features_for_this_segment = [
            var.mean(),
            var.median(),
            var.max (),
            var.min (),
            var.stdev (),
            var.kurtosis (),
            var.skewness(),
            *var.percentiles([0.01, 0.02, 0.05, 0.10,  …]),

            ]
```

Do you like features?

```
from faktion.ml.timeseries import extract_features
extracted_features = extract_features(timeseries, column_sort="time")
```

abs_energy(x)	Returns the absolute energy of the time series which is the sum over the squared values
absolute_sum_of_changes(x)	Returns the sum over the absolute value of consecutive changes in the series x
agg_autocorrelation(x, param)	Calculates the value of an aggregation function (e.g.
agg_linear_trend(x, param)	Calculates a linear least-squares regression for values of the time series that were ag sequence from 0 up to the number of chunks minus one.
approximate_entropy(x, m, r)	Implements a vectorized Approximate entropy algorithm.
ar_coefficient(x, param)	This feature calculator fits the unconditional maximum likelihood of an autoregressive AR(k) process.
augmented_dickey_fuller(x, param)	The Augmented Dickey-Fuller test is a hypothesis test which checks whether a unit root is present in a time series sample.
autocorrelation(x, lag)	Calculates the autocorrelation of the specified lag, according to the formula [1]
binned_entropy(x, max_bins)	First bins the values of x into max_bins equidistant bins.
c3(x, lag)	This function calculates the value of
change_quantiles(x, ql, qh, isabs, f _agg)	First fixes a corridor given by the quantiles ql and qh of the distribution of x.
cid_ce(x, normalize)	This function calculator is an estimate for a time series complexity [1] (A more complex time series has more peaks, valleys etc.).
count_above_mean(x)	Returns the number of values in x that are higher than the mean of x
count_below_mean(x)	
cwt_coefficients(x, param)	
energy_ratio_by_chunks(x, pa)	
fft_aggregated(x, param)	Returns the spectral centroid (mean), variance, skew, and kurtosis of the absolute fourier transform spectrum.
fft_coefficient(x, param)	Calculates the fourier coefficients of the one-dimensional discrete Fourier Transform for real input by fast
first_location_of_maximum(x)	Returns the first location of the maximum value of x.
first_location_of_minimum(x)	Returns the first location of the minimal value of x.
friedrich_coefficients(x, param)	Coefficients of polynomial , which has been fitted to
has_duplicate(x)	Checks if any value in x occurs more than once
has_duplicate_max(x)	Checks if the maximum value of x is observed more than once
has_duplicate_min(x)	Checks if the minimal value of x is observed more than once
index_mass_quantile(x, param)	Those apply features calculate the relative index i where q% of the mass of the time series x lie left of i.
kurtosis(x)	Returns the kurtosis of x (calculated with the adjusted Fisher-Pearson standardized moment coefficient G2).
large_standard_deviation(x, r)	Boolean variable denoting if the standard dev of x is higher than 'r' times the range = difference between max and min of x.
last_location_of_maximum(x)	Returns the relative last location of the maximum value of x.
last_location_of_minimum(x)	Returns the last location of the minimal value of x.

linear_trend(x, param)	Calculate a linear least-squares regression for the values of the time series versus the sequence from 0 to length of the time series minus one.
linear_trend_timewise(x, param)	Calculate a linear least-squares regression for the values of the time series versus the sequence from 0 to length of the time series minus one.
longest_strike_above_mean(x)	Returns the length of the longest consecutive subsequence in x that is bigger than the mean of x
	Returns the length of the longest consecutive subsequence in x that is smaller than the mean of x
	arg fixed point of dynamics :math:argmax_x {h(x)=0}` estimated from polynomial ,
	lates the highest value of the time series x.
mean(x)	Returns the mean of x
mean_abs_change(x)	Returns the mean over the absolute differences between subsequent time series values which is
mean_change(x)	Returns the mean over the differences between subsequent time series values which is
mean_second_derivative_central(x)	Returns the mean value of a central approximation of the second derivative
median(x)	Returns the median of x
minimum(x)	Calculates the lowest value of the time series x.
number_crossing_m(x, m)	Calculates the number of crossings of x on m.
number_cwt_peaks(x, n)	This feature calculator searches for different peaks in x.
number_peaks(x, n)	Calculates the number of peaks of at least support n in the time series x.
partial_autocorrelation(x, param)	Calculates the value of the partial autocorrelation function at the given lag.
percentage_of_reoccurring_datapoints_t	than once.
	once.
ratio_beyond_r_sigma(x, r)	Ratio of values that are more than r*std(x) (so r sigma) away from the mean of x.
ratio_value_number_to_time_series_length(x)	Returns a factor which is 1 if all values in the time series occur only once, and below one if this is not the case.
sample_entropy(x)	Calculate and return sample entropy of x.
set_property(key, value)	This method returns a decorator that sets the property key of the function to value
skewness(x)	Returns the sample skewness of x (calculated with the adjusted Fisher-Pearson standardized moment coefficient G1).
spkt_welch_density(x, param)	This feature calculator estimates the cross power spectral density of the time series x at different frequencies.
standard_deviation(x)	Returns the standard deviation of x
sum_of_reoccurring_data_points(x)	Returns the sum of all data points, that are present in the time series more than once.
sum_of_reoccurring_values(x)	Returns the sum of all values, that are present in the time series more than once.
sum_values(x)	Calculates the sum over the time series values
symmetry_looking(x, param)	Boolean variable denoting if the distribution of x *looks symmetric*.
time_reversal_asymmetry_statistic(x, lag)	This function calculates the value of
value_count(x, value)	Count occurrences of *value* in time series x.
variance(x)	Returns the variance of x
variance_larger_than_standard_deviation(x)	Boolean variable denoting if the variance of x is greater than its standard deviation.

ML for sensor data

Step 1: Prepare timeseries

Step 2: Calculate characteristics

Step 3: Train model

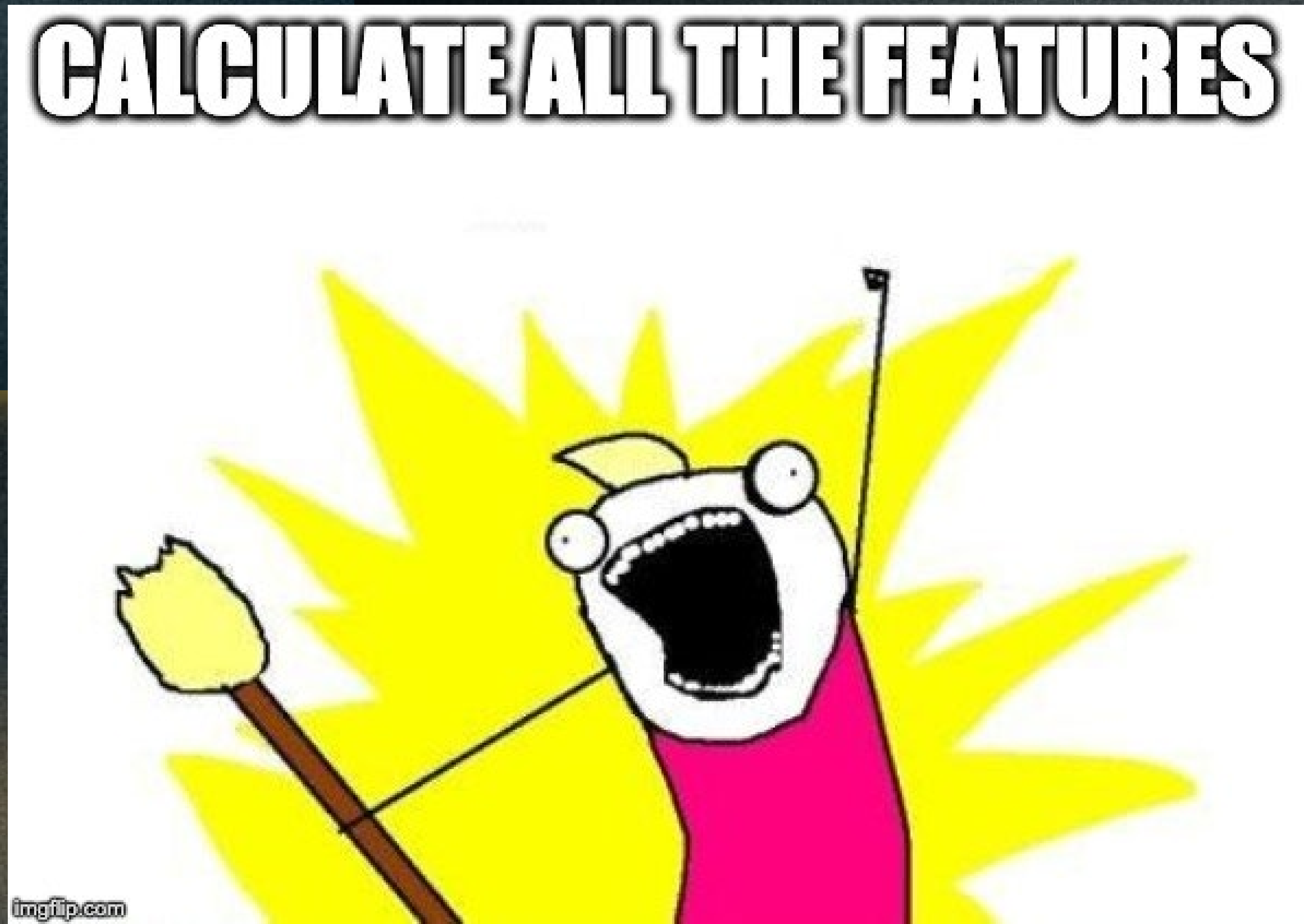
Characteristics become features (+- 1200)

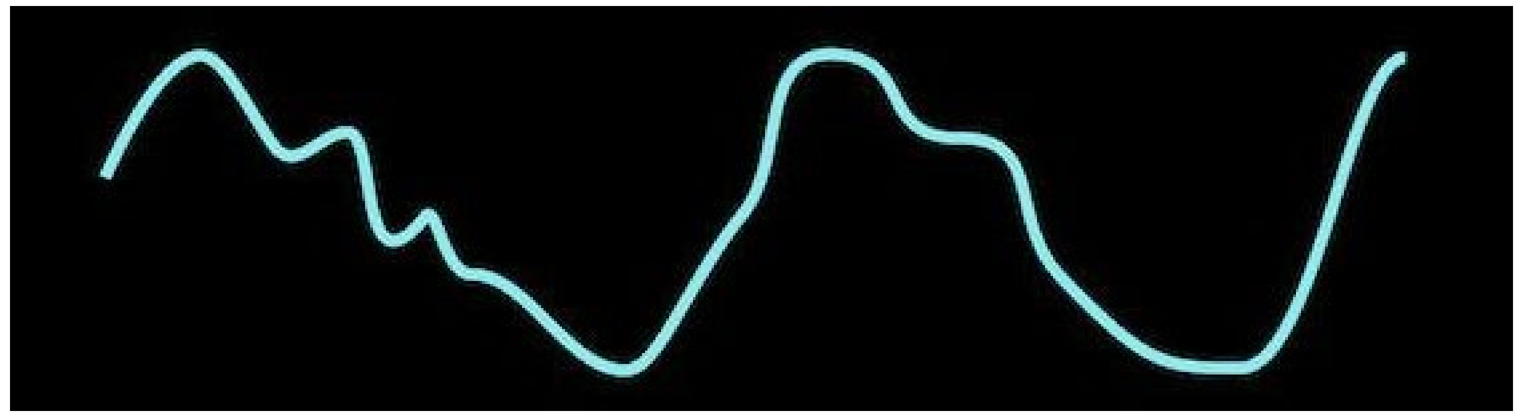
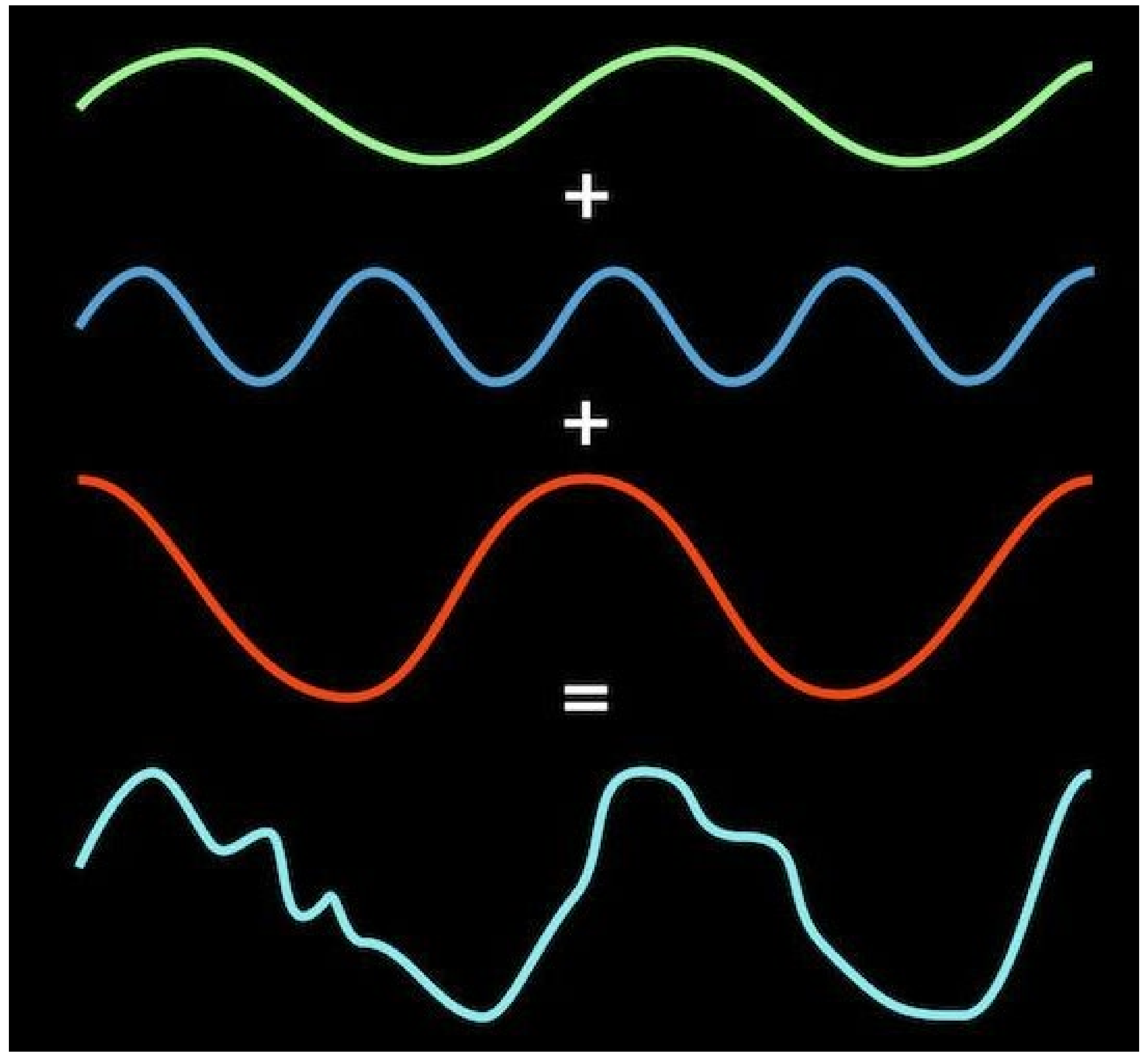
2160 observations

Model type	Use	Examples
Classification	• Good batch / batch batch • Is likely to fail in next 2 hours	Typical ML techniques • PCA • Random forest / XGBoost • Logistic / Linear regression • SVM • …
Regression	• Quality score • Expected yield • Digital Twin	
Anomaly detection	• Anomaly alerts on intervals	• Error analysis of time series forecasting models • Outlier statistics
Survival analysis	• Mean time between failures • Remaining lifetime prediction • Driving factors for failures	• Kaplan-Meier estimates • Cox Proportional Hazards • Aalen additive hazard regression

MORE
FEATURES

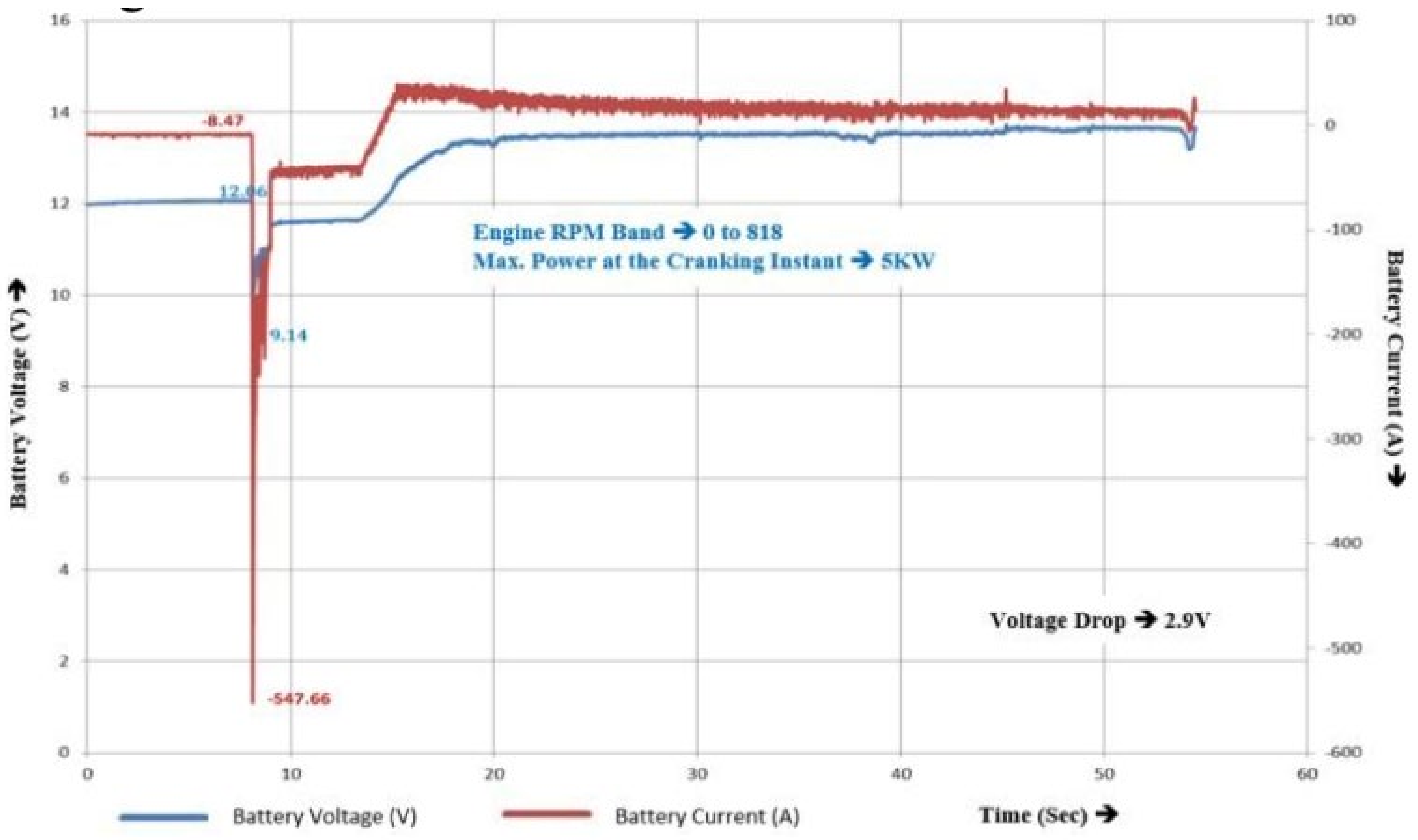
Specific applications require specific models.



	BEFORE	DFT	AFTER
FUNCTION		$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn}$	
DATA	1000 times and values		1000 frequencies and amplitudes
MODEL	Use 1000 values for modelling		Use top 3 freqs and amplitudes for modelling
MP3 ALGO	Store 1000 values in WAV file		Store 50 values in MP3 file

Details on MP3 algorithm: https://www.uio.no/studier/emner/matnat/math/nedlagte-emner/MAT-INF2360/v12/part1.pdf

TALK TO SUBJECT MATTER EXPERTS



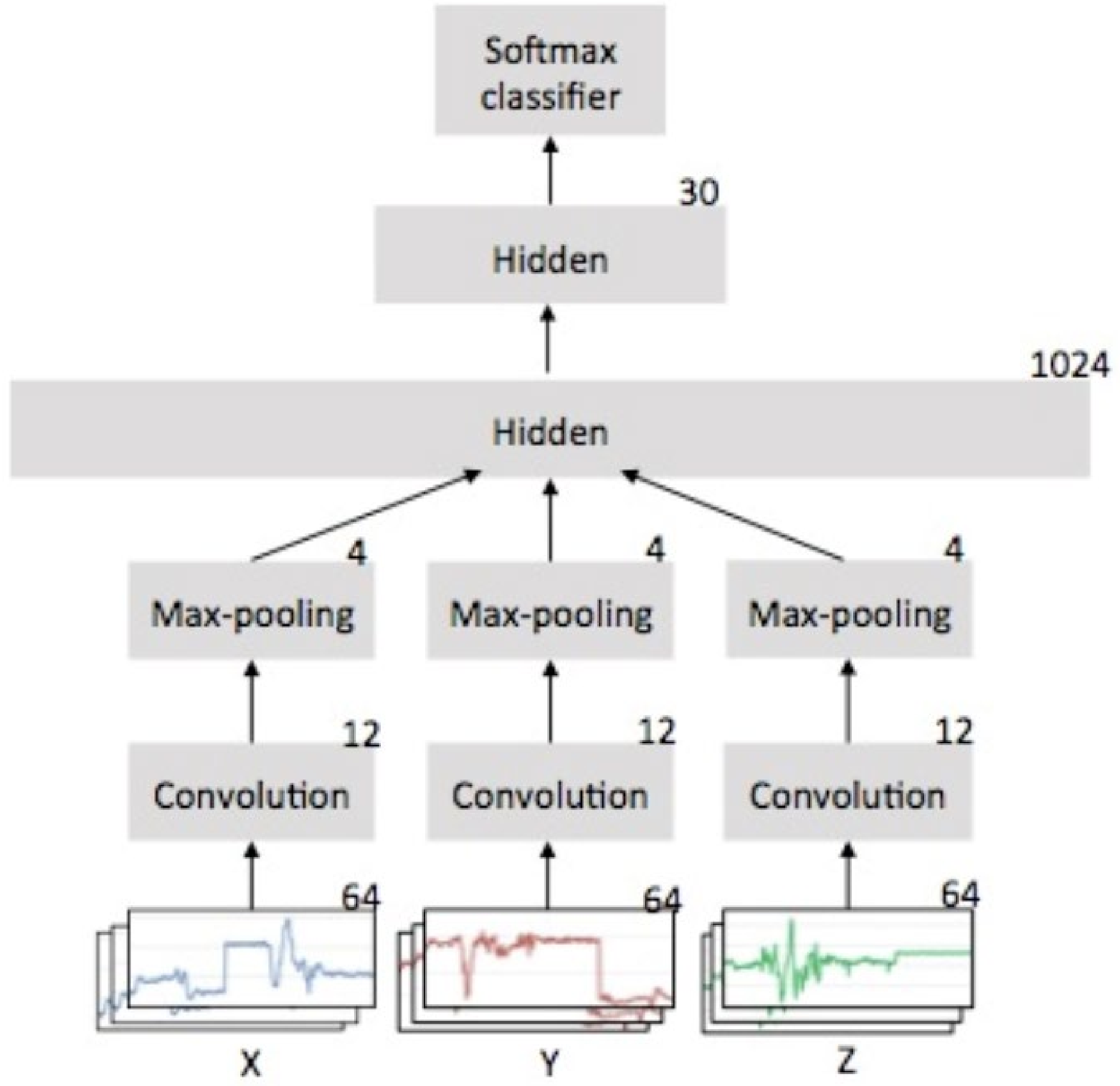
https://ijsea.com/archive/volume7/issue8/IJSEA07081005.pdf

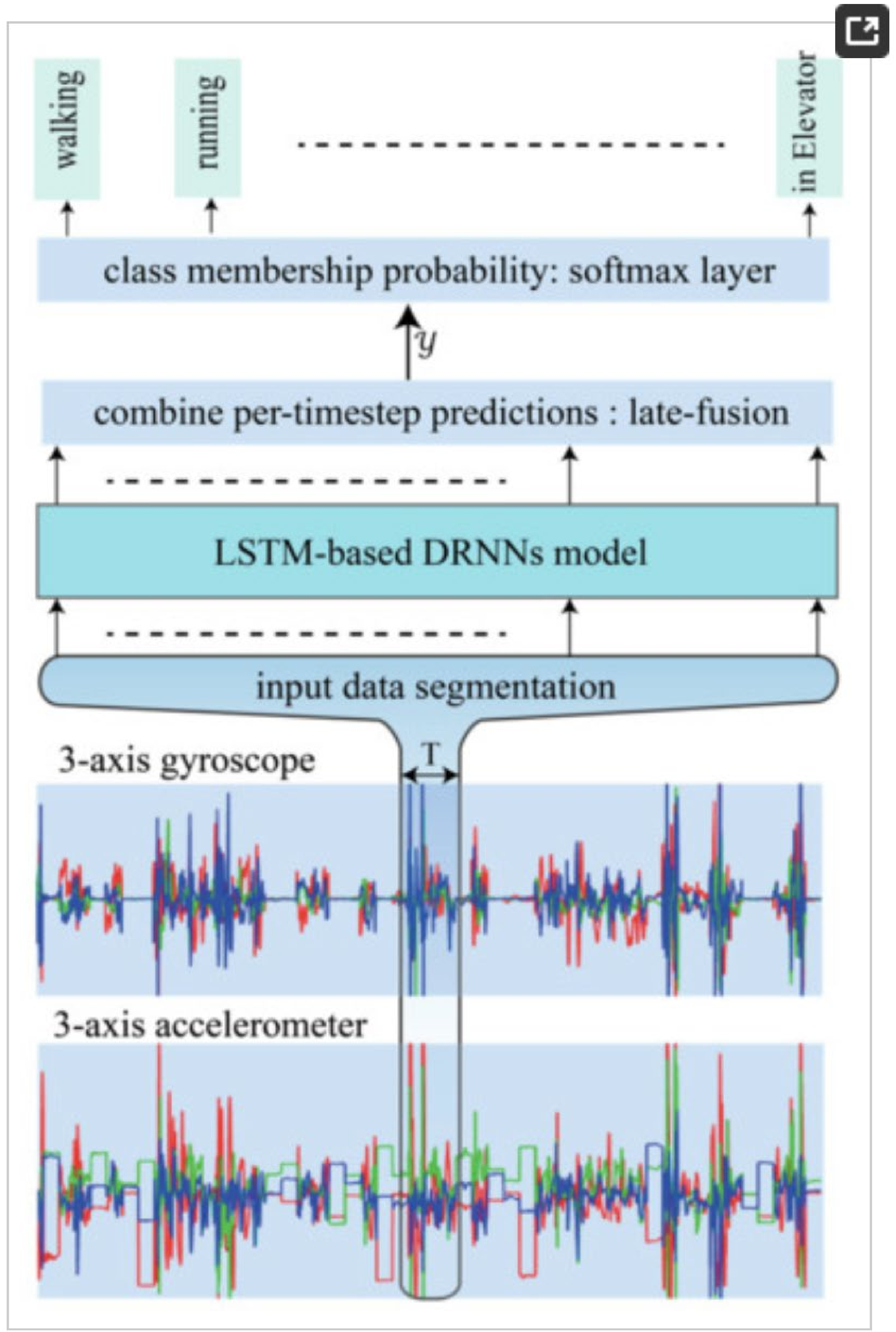
Sources of features

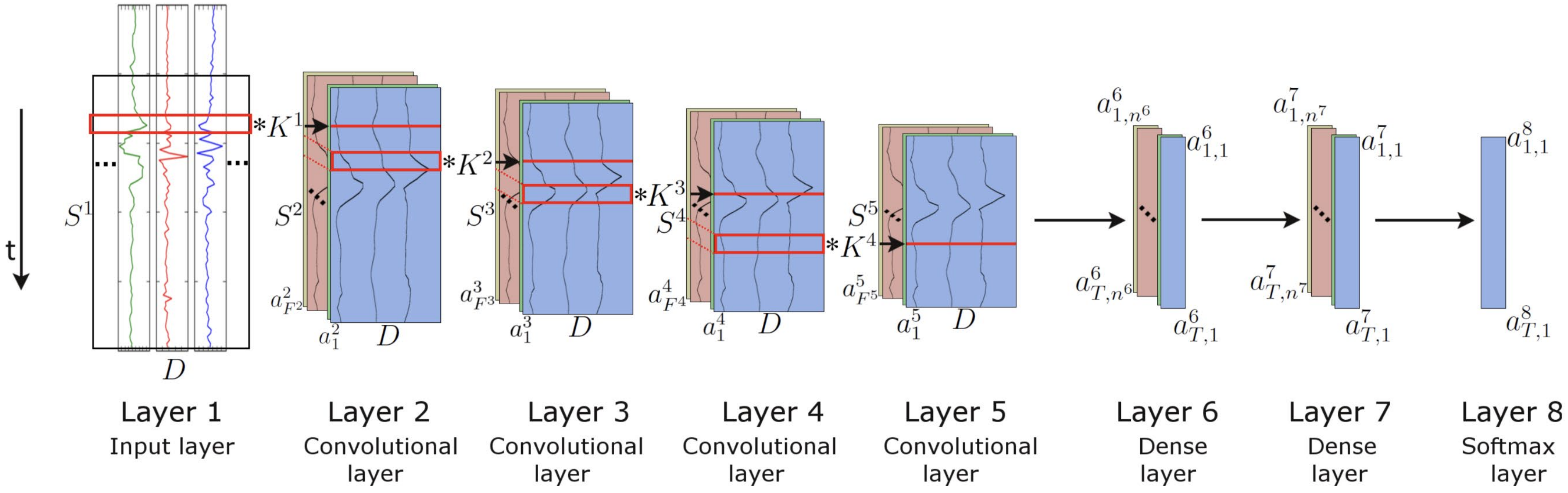
1. Univariate features
2. Think about the problem and translate to math
3. Fast Fourier Transform
4. Timeseries features
5. Subject matter experts

Sometimes… you can circumvent feature calculation by using Deep Learning

Convolutional and recurrent neural networks







https://machinelearningmastery.com/deep-learning-models-for-human-activity-recognition/

WE ARE CONSULTANTS WHO BUILD DEEP LEARNING, MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE PRODUCTS & SOLUTIONS

Artificial Intelligence is confusing. We know. Truth is, there is money to be made by selling hype. Contact us when you need a partner that delivers results instead.

THANK YOU

AI4Growth
May 27, 2019

WANT MORE?

- Full presentation including examples and exercises https://github.com/JosPolfliet/ml-sensor-data-minicourse-data
- The best book on time series analysis ever (in R though, not Python) https://otexts.com/fpp3/
- Intro –level filter theory https://ipython-books.github.io/102-applying-a-linear-filter-to-a-digital-signal/
- Modeling Survival Data: Extending the Cox Model https://www.springer.com/gp/book/9780387987842
- Reinforcement Learning and Optimal Control https://web.mit.edu/dimitrib/www/RLbook.html
- Azure IoT cloud architecture https://azure.microsoft.com/en-us/overview/iot/

- Still want more? Public training “Machine Learning for Sensor Data” on 8-9/9/2020 and 19-20/11/2020. Contact training@faktion.com