Faktion

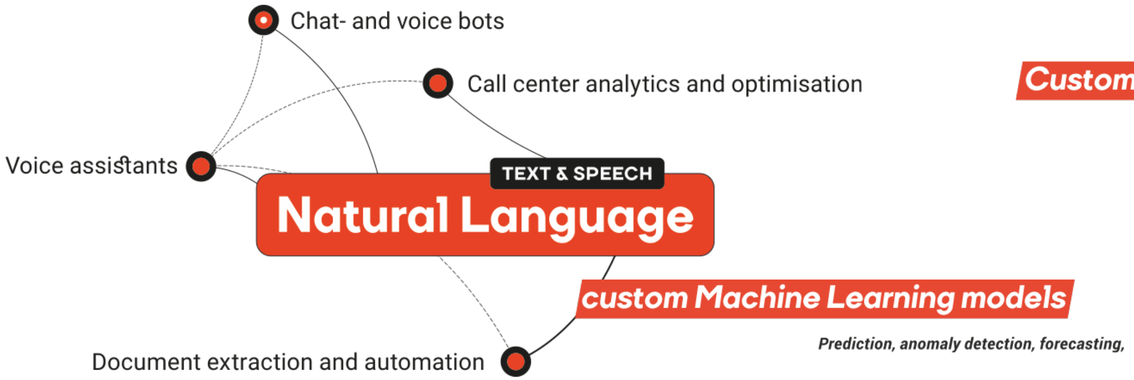
Jos Polfliet, VP of Applied AI

FAKTION — we put thought in everything

WE BUILD DEEP LEARNING, MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE SOLUTIONS

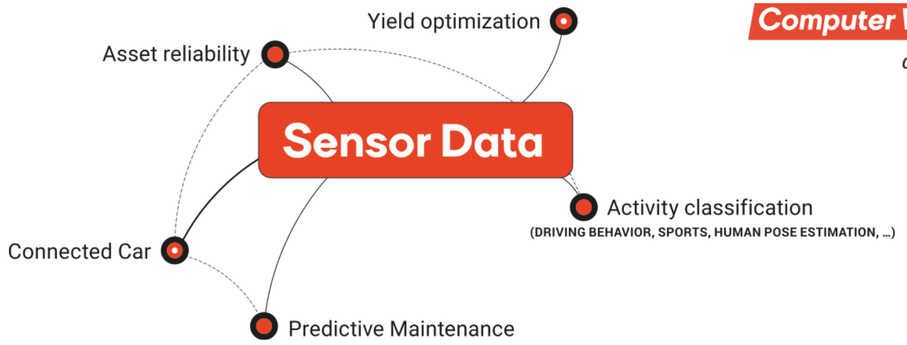
Artificial Intelligence is confusing. We know. Truth is, there is money to be made by selling hype. Contact us when you need a partner that delivers results instead.

FAKTION — “we put thought in everything”



Faktion is an applied AI service provider.
We build custom software applications
with Machine and Deep Learning technology.

Due to the strong extent of our experience,
we share that by means of strategic consulting.



We build reusable and scalable AI sensor data platforms that can easily be extended to dozens of other use cases

HOW

CAN WE BUILD THIS

Let's use Machine Learning for sensor data.

THE FUNDAMENTAL PROBLEM

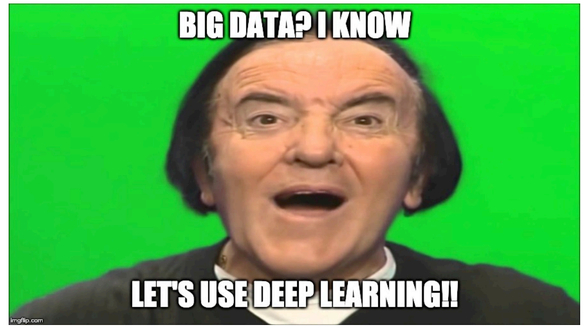
A journey of discovery

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
…	…	…



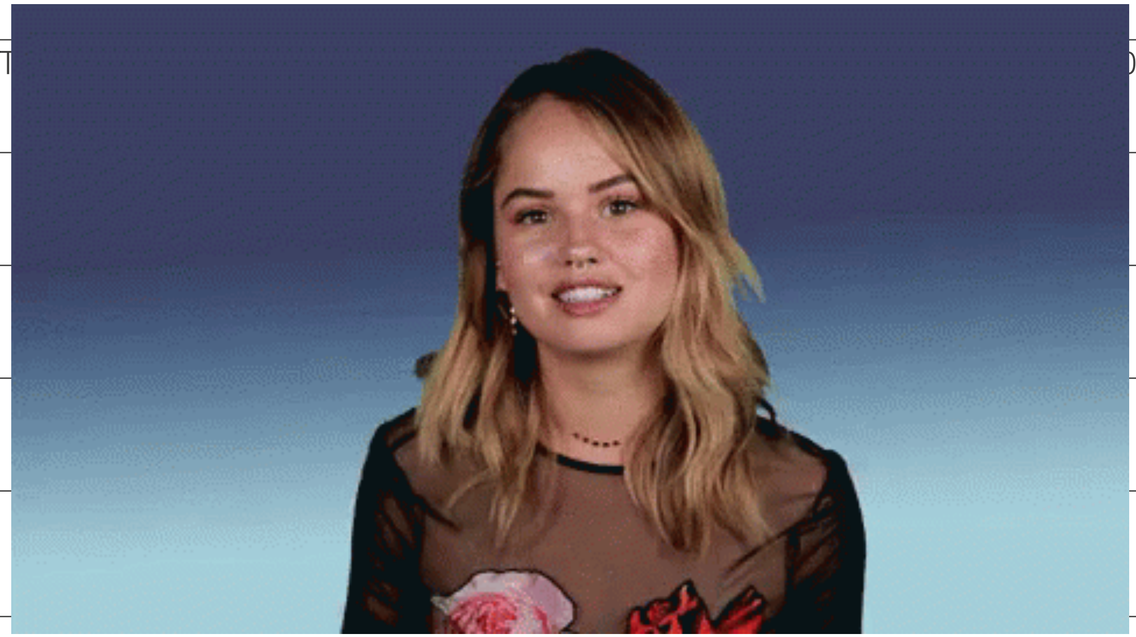
What not to do

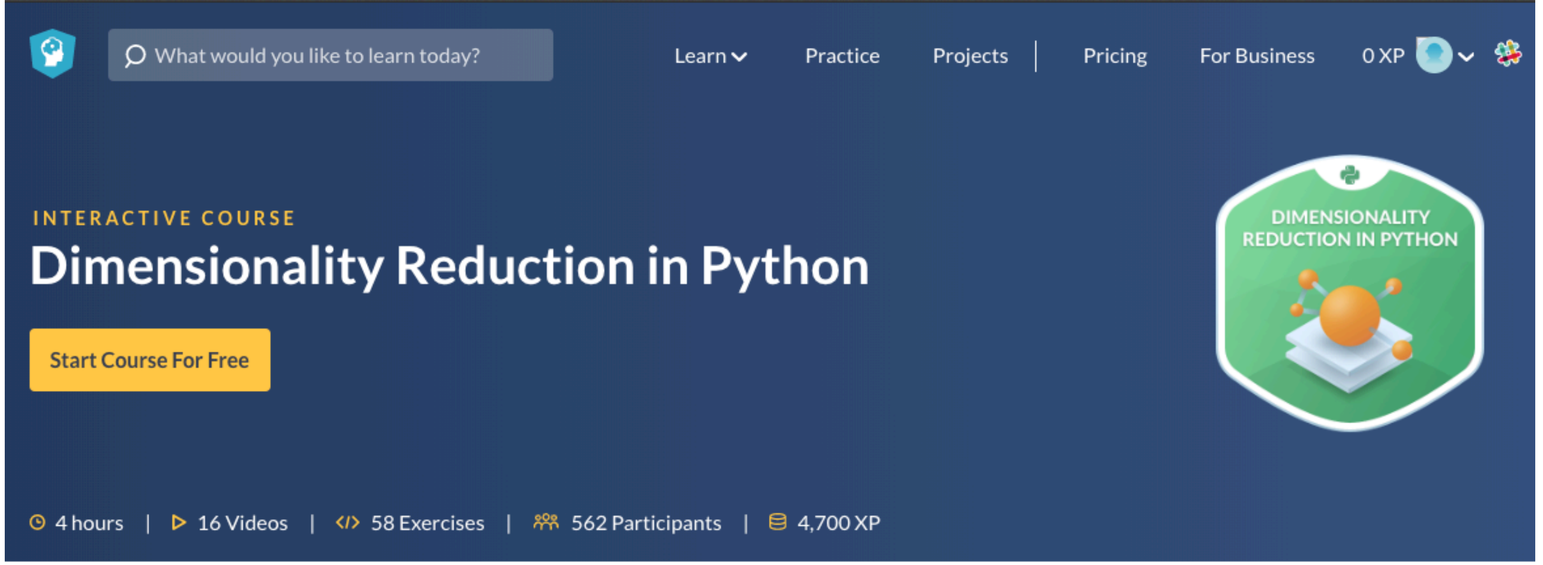
25x Temperature, 25x pressure, 25x flow at 100Hz for 60s

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

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

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





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

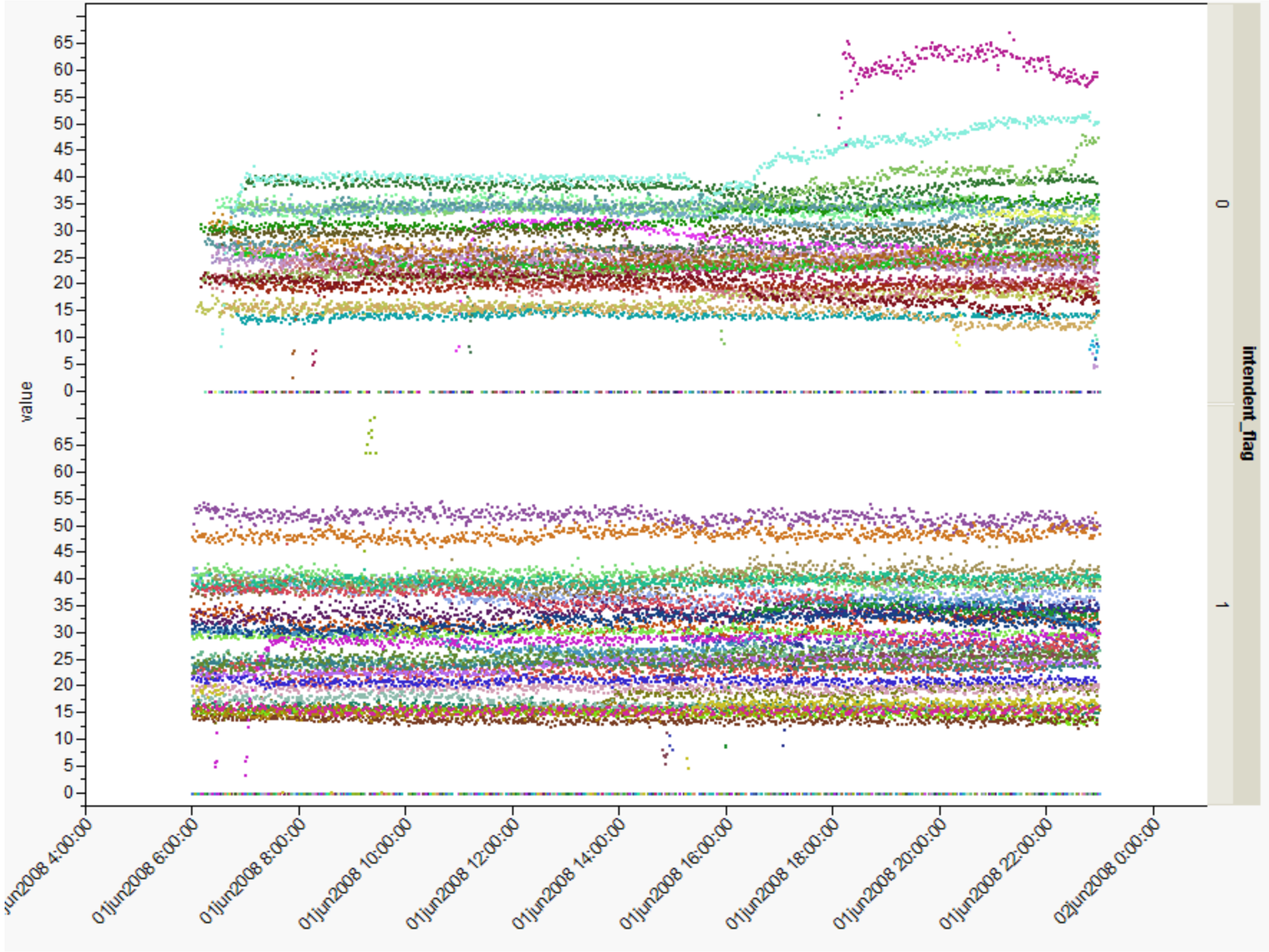
Jeroen is a machine learning engineer working at Faktion, an AI company from Belgium. He uses both R and Python for his analyses and has a PhD background in computational biology. His experience mostly lies in working with structured data, produced by sensors or digital processes.

See More

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

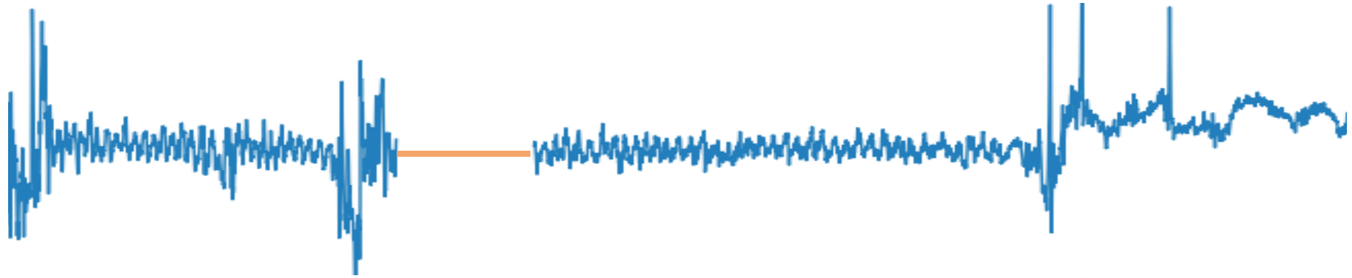
Time series data mining

Step 1: Prepare timeseries



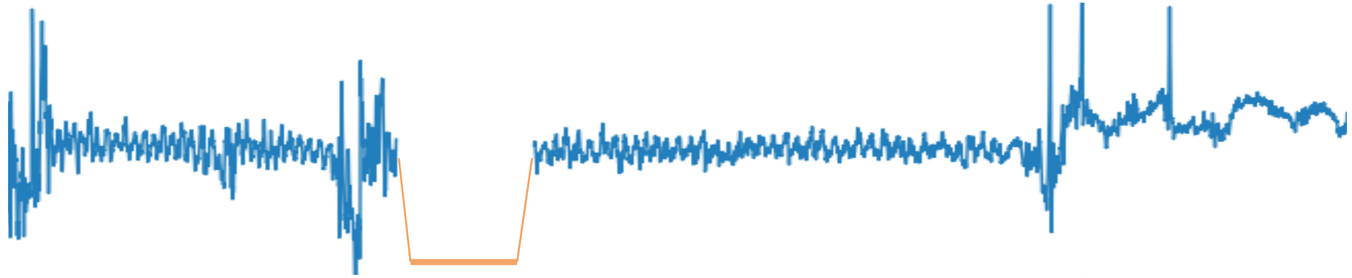
Pre-processing - Interpolation

TEMPERATURE



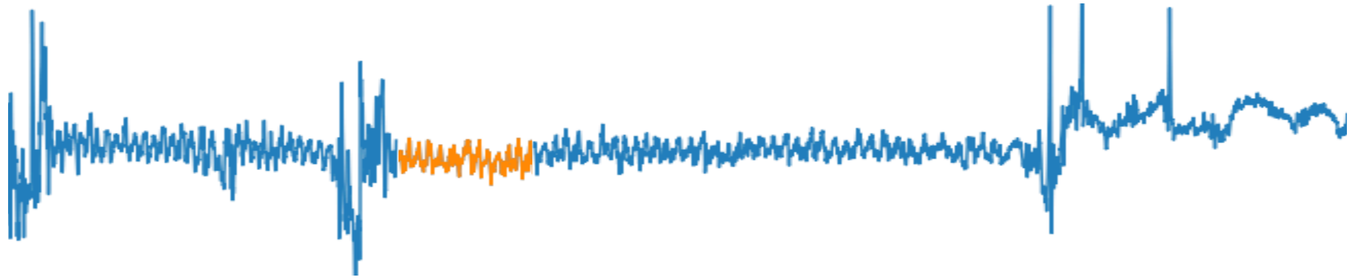
Interpolate linearly, with last value, first new know value, smooth,

PRODUCTION VOLUME



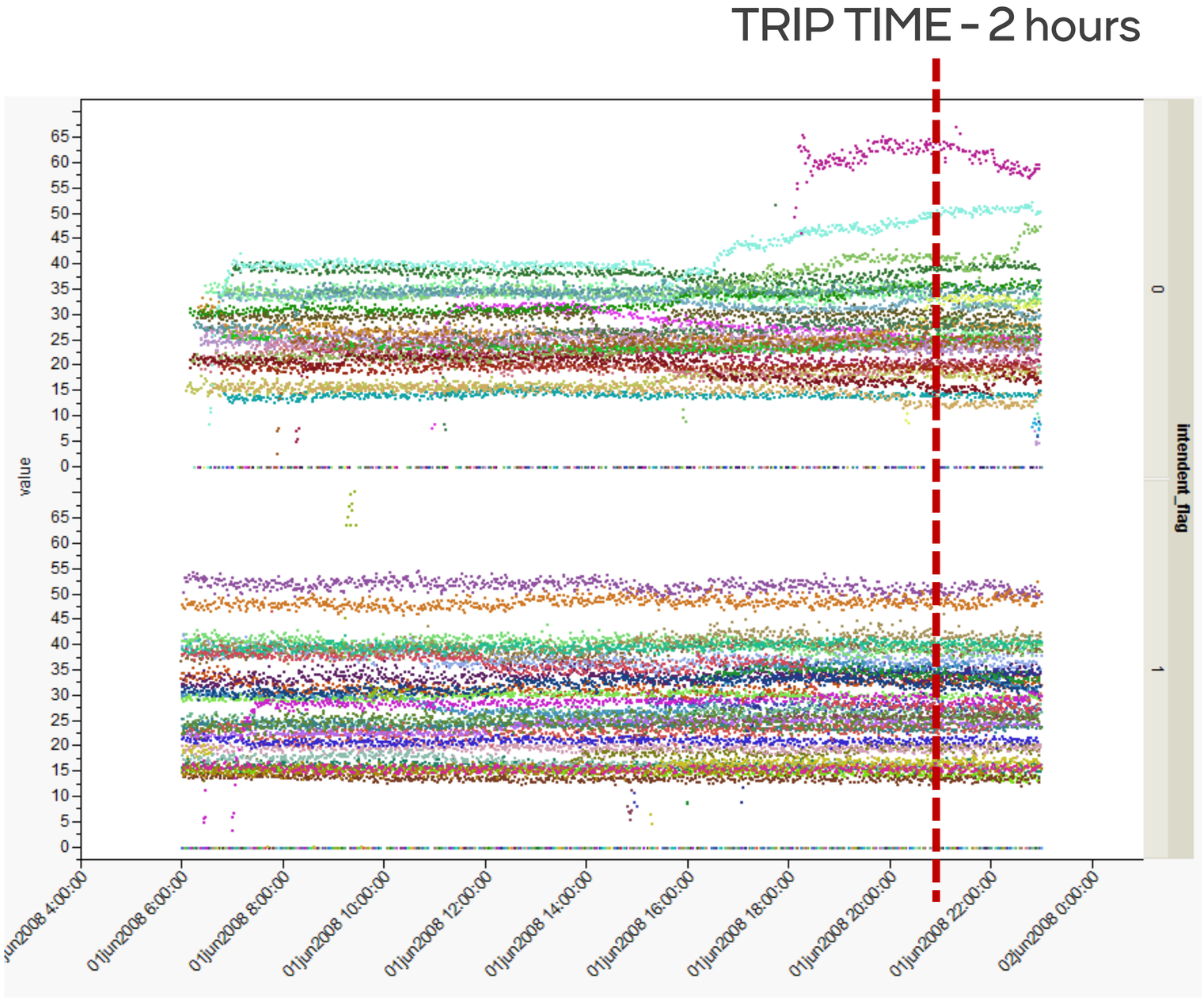
Impute with zeroes

YOU HAVE TO MUCH TIME



Decompose, interpolate trend with splines and re-add seasonality

Bucketization



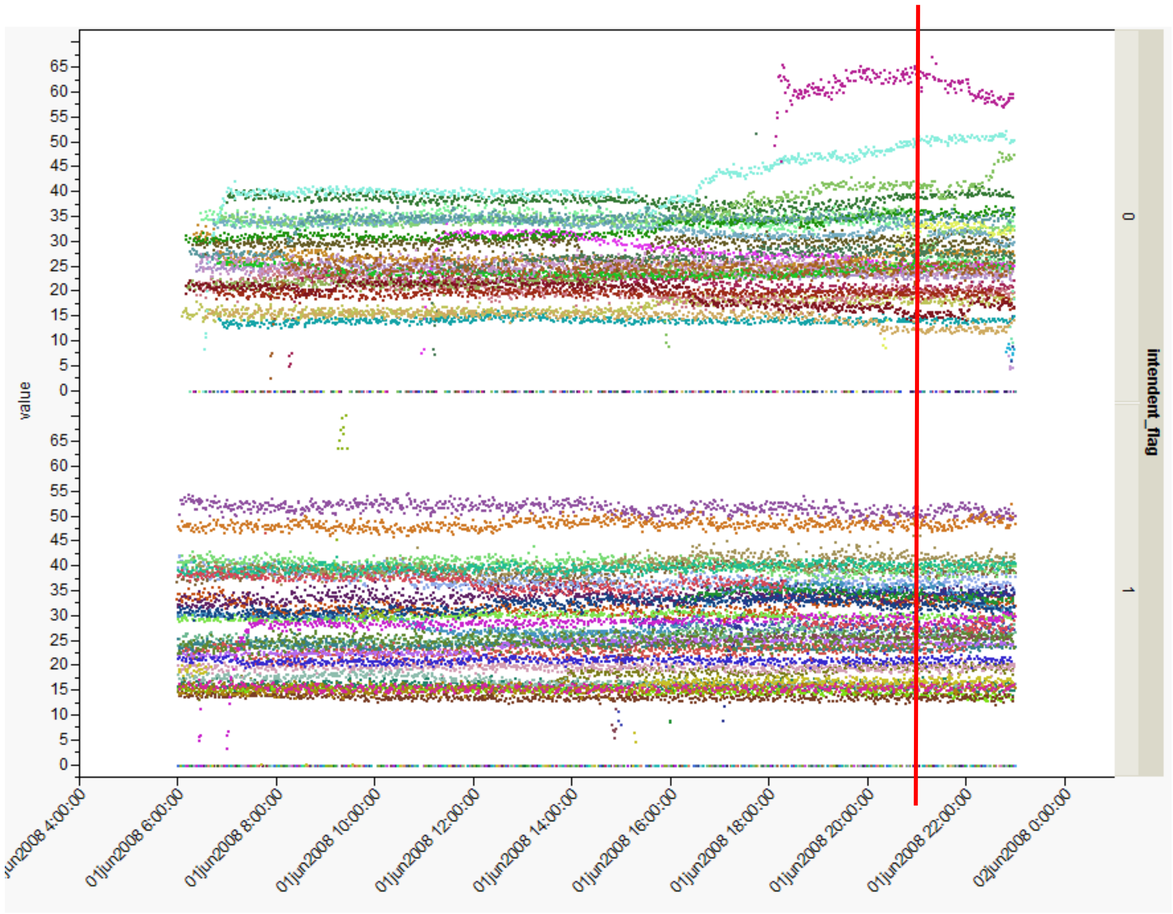
Cleaning

SOME OTHER MAGIC

- Removing outliers
- Changing reference systems
- Kalman Filter smoothing
- Combining multiple readings into 1
- ...

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,  ...]),

            ]
```

Series.abs()	Return a Series/DataFrame with absolute numeric value of each element.
Series.all([axis, bool_only, skipna, level])	Return whether all elements are True, potentially over an axis.
Series.any([axis, bool_only, skipna, level])	Return whether any element is True, potentially over an axis.
Series.autocorr([lag])	Compute the lag-N autocorrelation.
Series.between(left, right[, inclusive])	Return boolean Series equivalent to left <= series <= right.
Series.clip([lower, upper, axis, inplace])	Trim values at input threshold(s).
Series.clip_lower(threshold[, axis, inplace])	(DEPRECATED) Trim values below a given threshold.
Series.clip_upper(threshold[, axis, inplace])	(DEPRECATED) Trim values above a given threshold.
Series.corr(other[, method, min_periods])	Compute correlation with *other* Series, excluding missing values.
Series.count([level])	Return number of non-NA/null observations in the Series.
Series.cov(other[, min_periods])	Compute covariance with Series, excluding missing values.
Series.cummax([axis, skipna])	Return cumulative maximum over a DataFrame or Series axis.
Series.cummin([axis, skipna])	Return cumulative minimum over a DataFrame or Series axis.
Series.cumprod([axis, skipna])	Return cumulative product over a DataFrame or Series axis.
Series.cumsum([axis, skipna])	Return cumulative sum over a DataFrame or Series axis.
Series.describe([percentiles, include, exclude])	Generate descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding `NaN` values.
Series.diff([periods])	First discrete difference of element.
Series.factorize([sort, na_sentinel])	Encode the object as an enumerated type or categorical variable.
Series.kurt([axis, skipna, level, numeric_only])	Return unbiased kurtosis over requested axis using Fisher's definition of kurtosis (kurtosis of normal == 0.0).
Series.mad([axis, skipna, level])	Return the mean absolute deviation of the values for the requested axis.
Series.max([axis, skipna, level, numeric_only])	Return the maximum of the values for the requested axis.
Series.mean([axis, skipna, level, numeric_only])	Return the mean of the values for the requested axis.
Series.median([axis, skipna, level, …])	Return the median of the values for the requested axis.
Series.min([axis, skipna, level, numeric_only])	Return the minimum of the values for the requested axis.
Series.mode([dropna])	Return the mode(s) of the dataset.
Series.nlargest([n, keep])	Return the largest *n* elements.
Series.nsmallest([n, keep])	Return the smallest *n* elements.
Series.pct_change([periods, fill_method, …])	Percentage change between the current and a prior element.
Series.prod([axis, skipna, level, …])	Return the product of the values for the requested axis.
Series.quantile([q, interpolation])	Return value at the given quantile.
Series.rank([axis, method, numeric_only, …])	Compute numerical data ranks (1 through n) along axis.
Series.sem([axis, skipna, level, ddof, …])	Return unbiased standard error of the mean over requested axis.
Series.skew([axis, skipna, level, numeric_only])	Return unbiased skew over requested axis Normalized by N-1.
Series.std([axis, skipna, level, ddof, …])	Return sample standard deviation over requested axis.
Series.sum([axis, skipna, level, …])	Return the sum of the values for the requested axis.
Series.var([axis, skipna, level, ddof, …])	Return unbiased variance over requested axis.
Series.kurtosis([axis, skipna, level, …])	Return unbiased kurtosis over requested axis using Fisher's definition of kurtosis (kurtosis of normal == 0.0).
Series.unique()	Return unique values of Series object.
Series.nunique([dropna])	Return number of unique elements in the object.
Series.is_unique	Return boolean if values in the object are unique.
Series.is_monotonic	Return boolean if values in the object are monotonic_increasing.
Series.is_monotonic_increasing	Return boolean if values in the object are monotonic_increasing.
Series.is_monotonic_decreasing	Return boolean if values in the object are monotonic_decreasing.
Series.value_counts([normalize, sort, …])	Return a Series containing counts of unique values.
Series.compound([axis, skipna, level])	Return the compound percentage of the values for the requested axis.

Time series data mining

Step 1: Prepare timeseries

Step 2: Calculate characteristics

Step 3: Train model

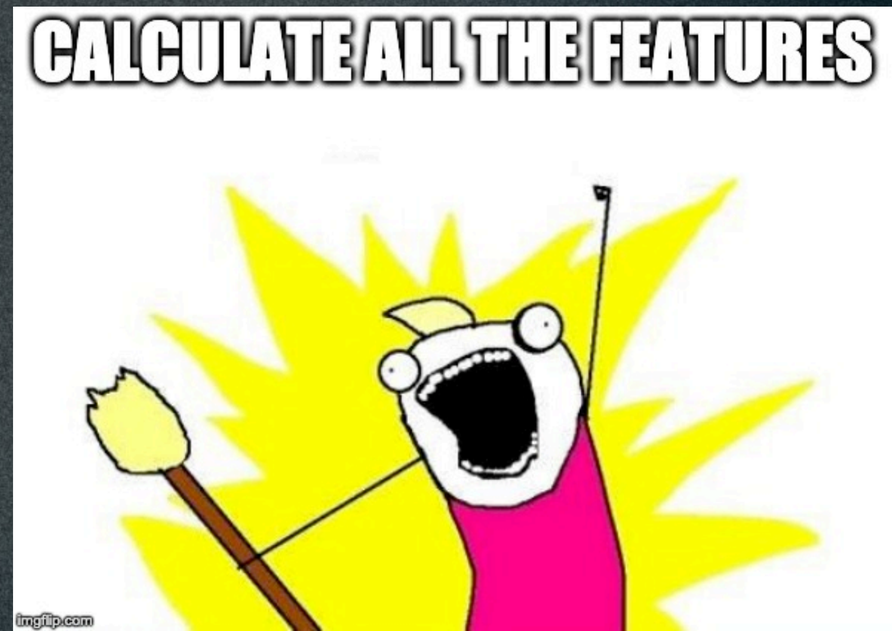
Characteristics become features (+- 12000)
160 observations

Random forests, XGBoost, Linear Discriminant Analysis, ... or another high dimensional classification technique

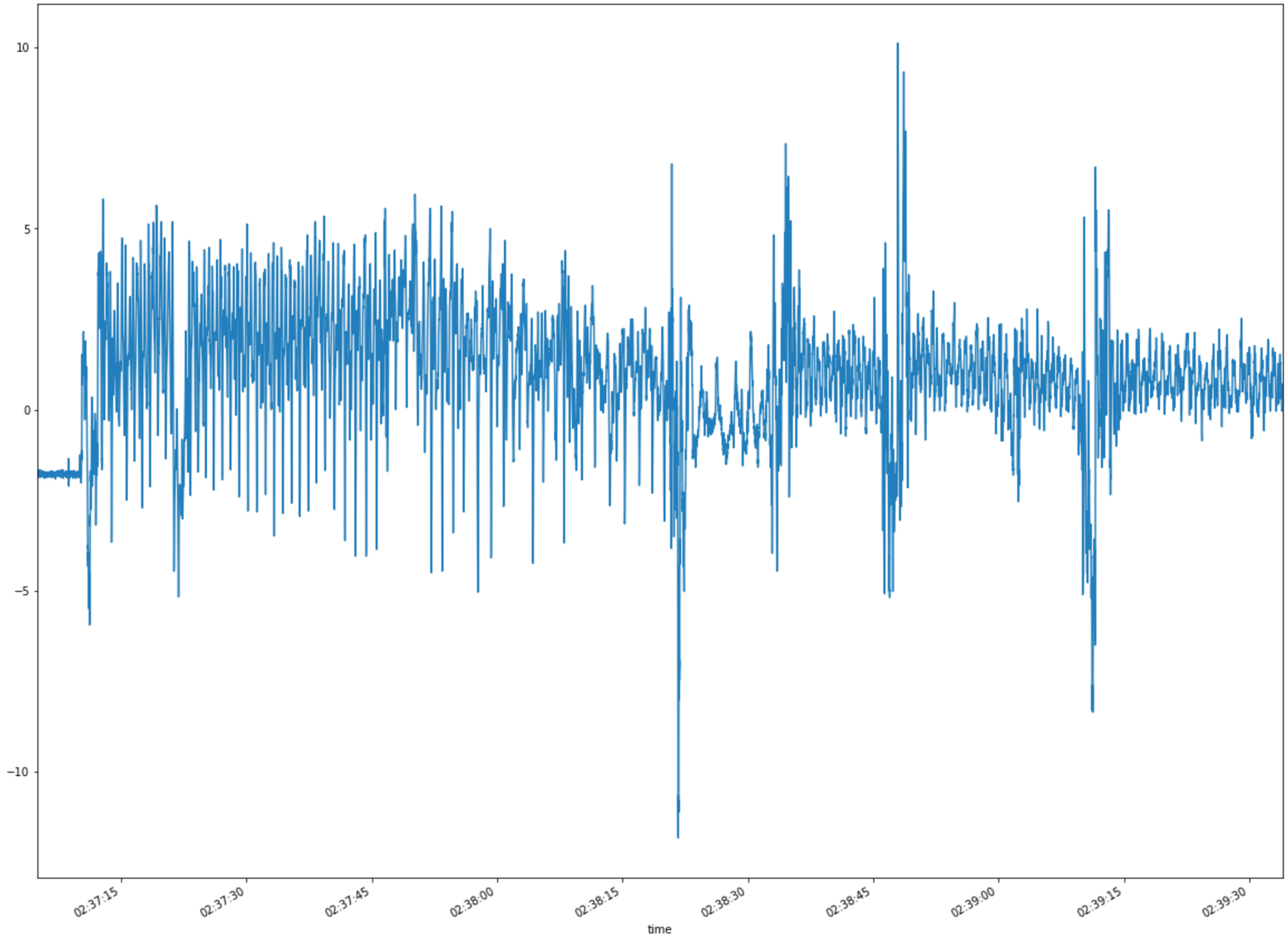
Pro-tip: https://www.datacamp.com/courses/dimensionality-reduction-in-python

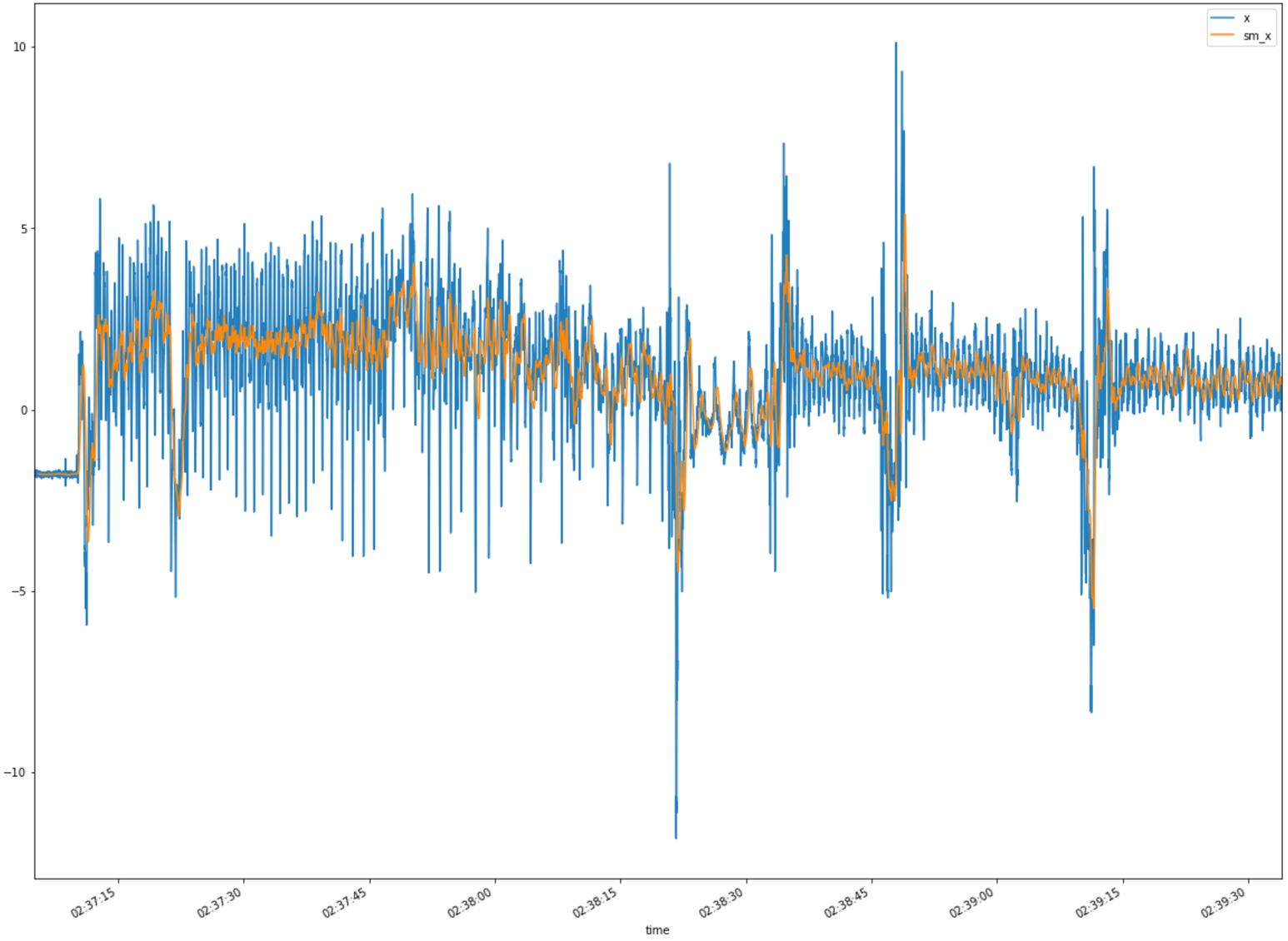
MORE FEATURES

Specific applications require specific models.

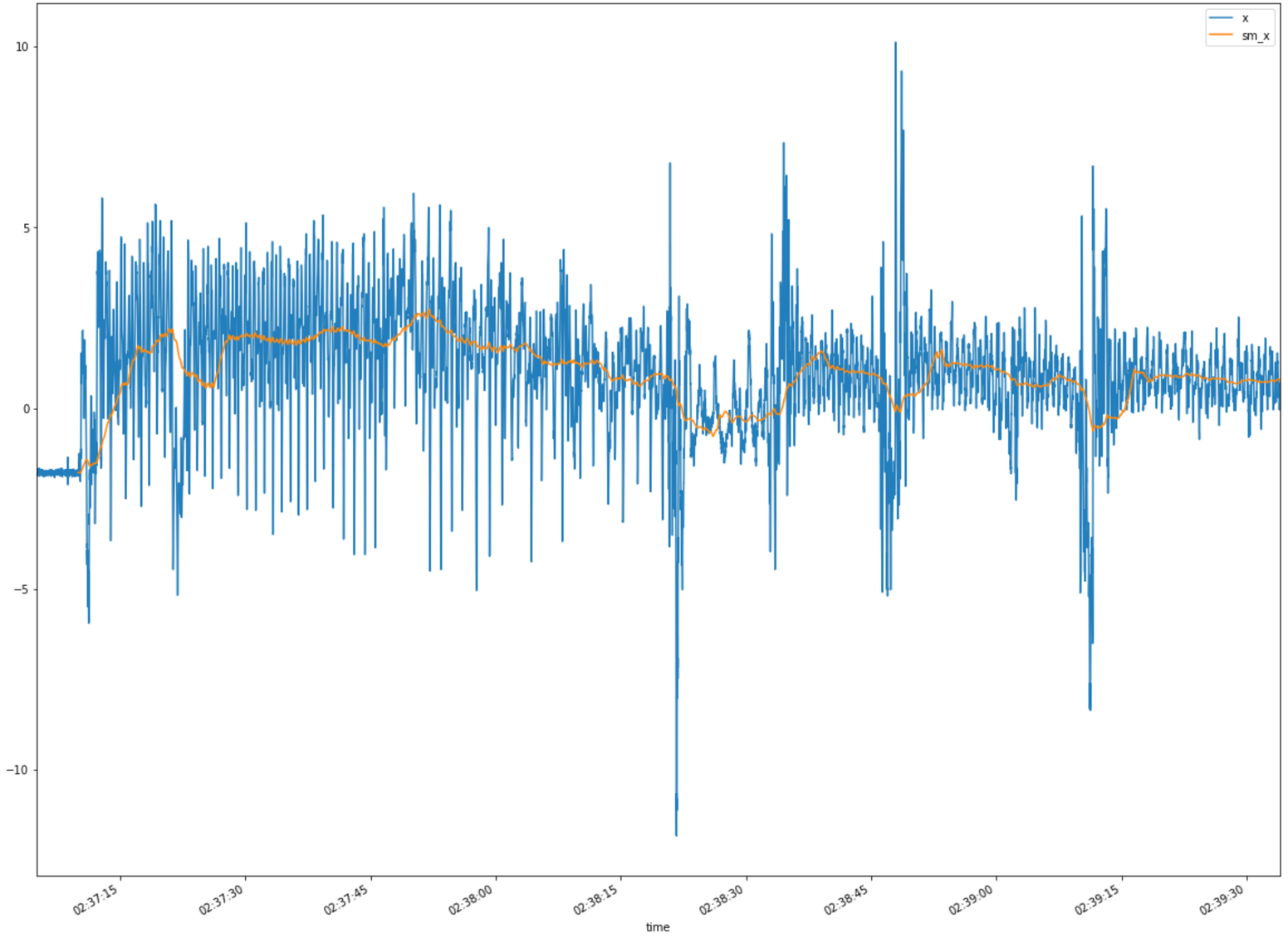


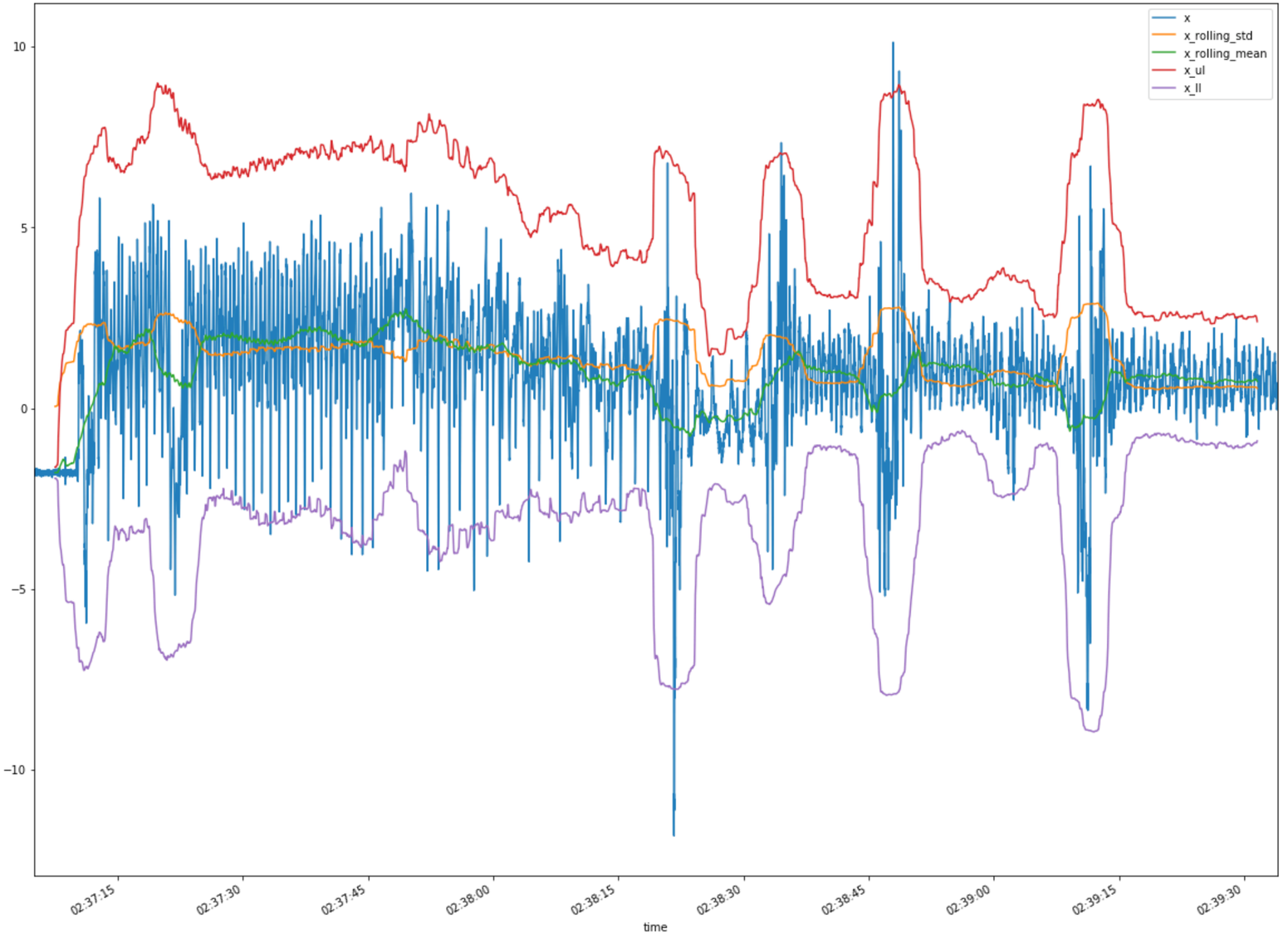
Peak detection algorithms





Rolling mean with window=50


x
sm_x
10
5
0
−5
−10
02:37:15
02:37:30
02:37:45
02:38:00
02:38:15
02:38:30
02:38:45
02:39:00
02:39:15
02:39:30
time



http://localhost:8888/notebooks/plot%20sensor%20data%20example.ipynb

DFT

JOSEPH FOURIER

Joseph Fourier was a French mathematician and physicst who lived from 1768 and 1830 invented Fourier analysis, which is an important component of analyzing sensor data

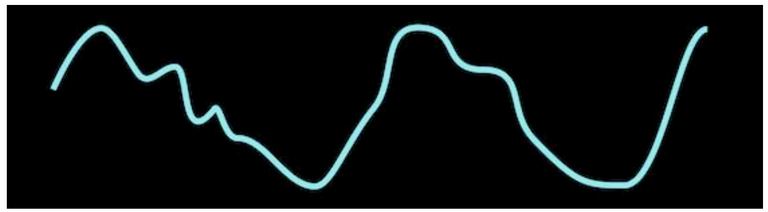
BEFORE

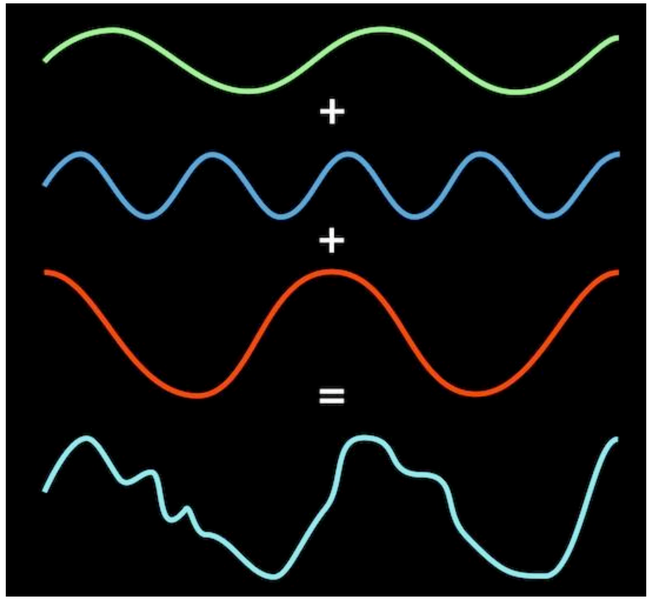
DFT

AFTER

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N} kn}$$

FUNCTION





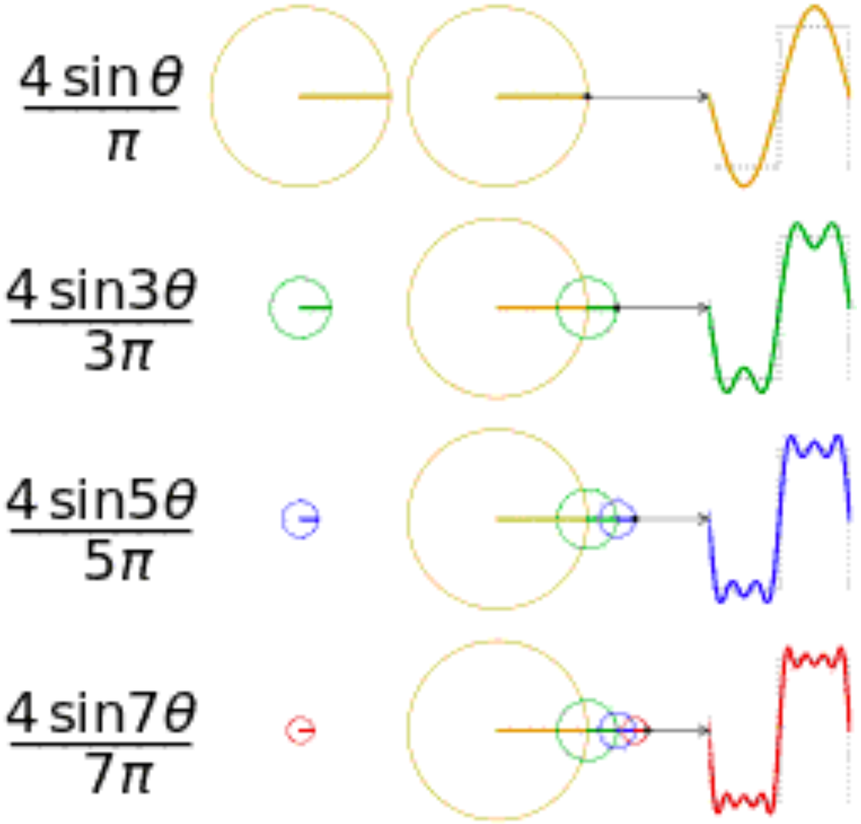
DATA

1000 times and values

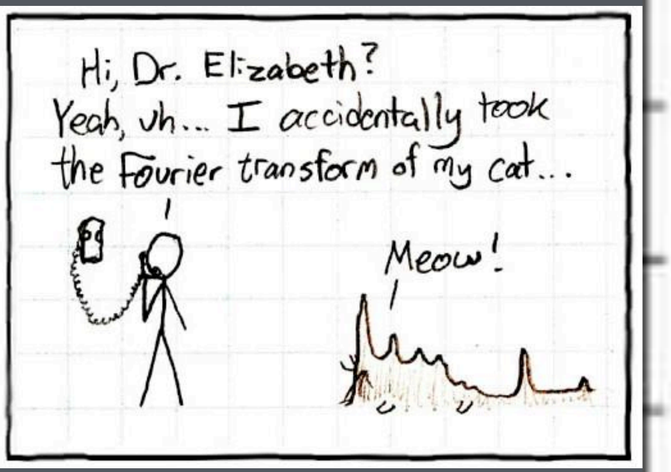
1000 frequencies and amplitudes

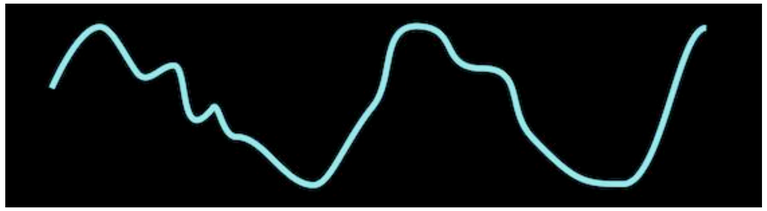
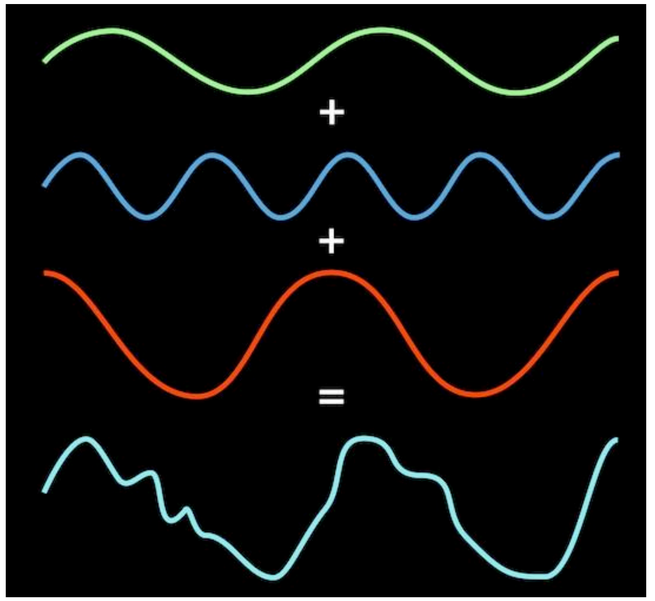
Recap or Intro to FOURIER analysis

SINE WAVE



Decomposition

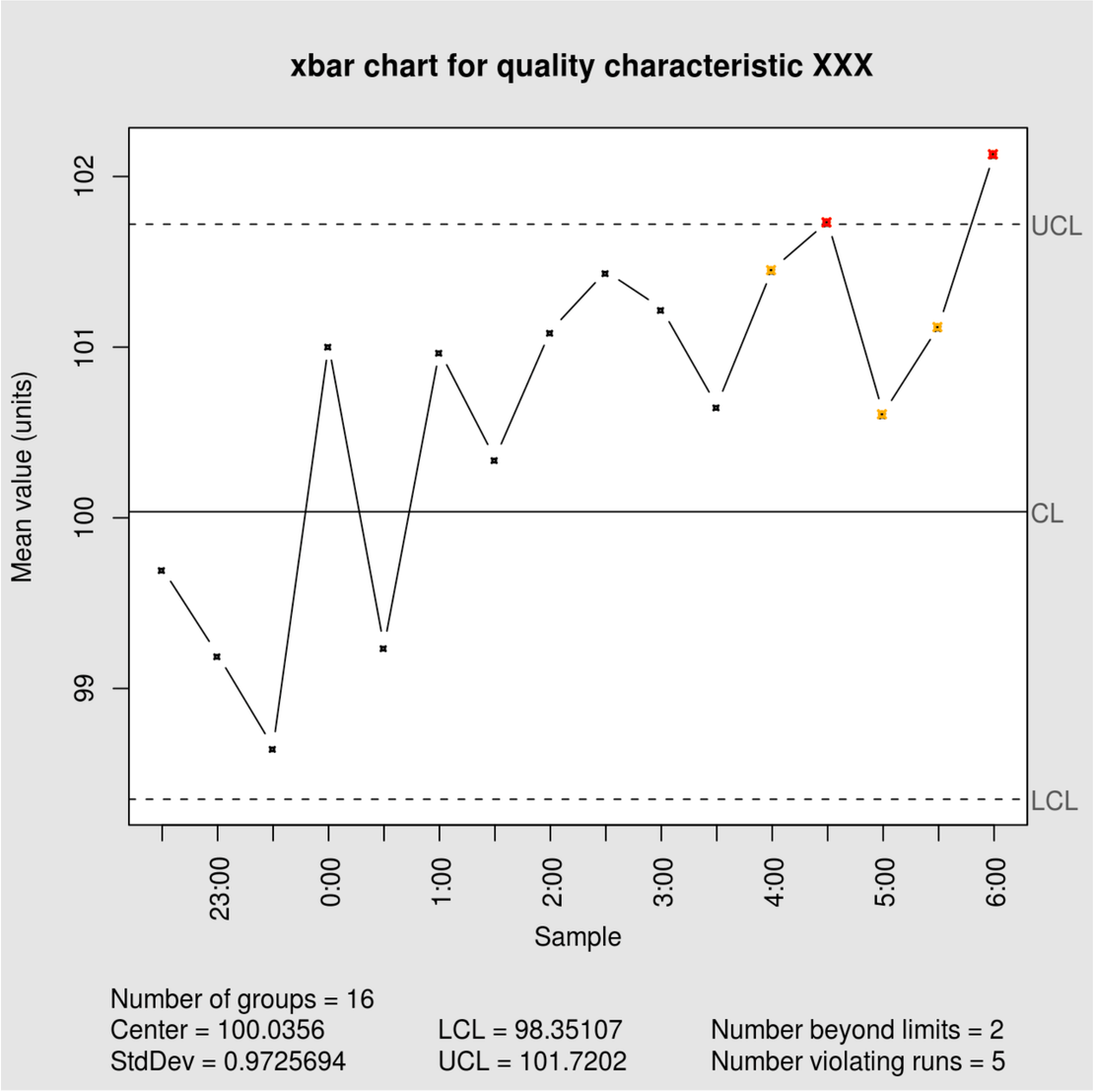

Hi, Dr. Elizabeth?
Yeah, uh... I accidentally took the Fourier transform of my cat...
Meow!

	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

http://localhost:8888/notebooks/plot%20fourier.ipynb#10.1.-Analyzing-the-frequency-components-of-a-signal-with-a-Fast-Fourier-Transform

SHEWHART CONTROL CHARTS

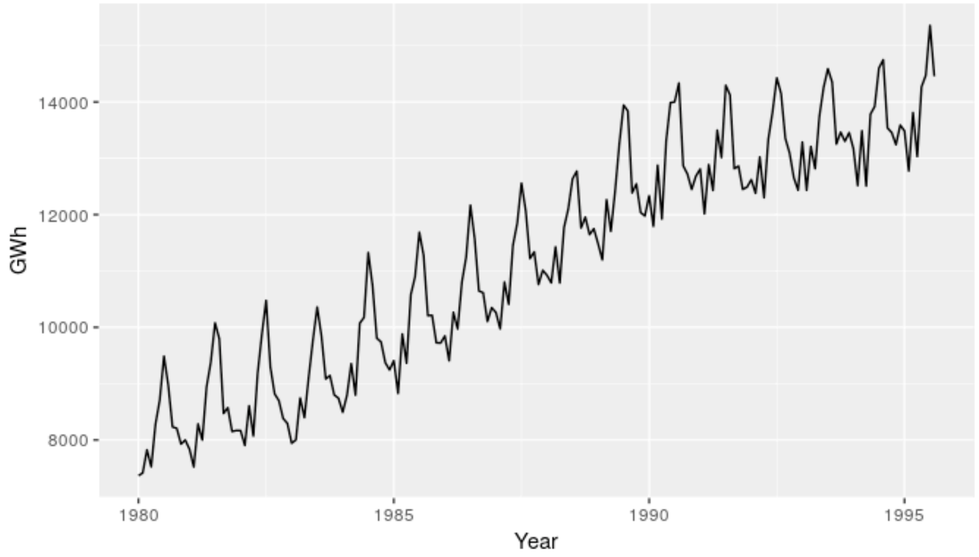


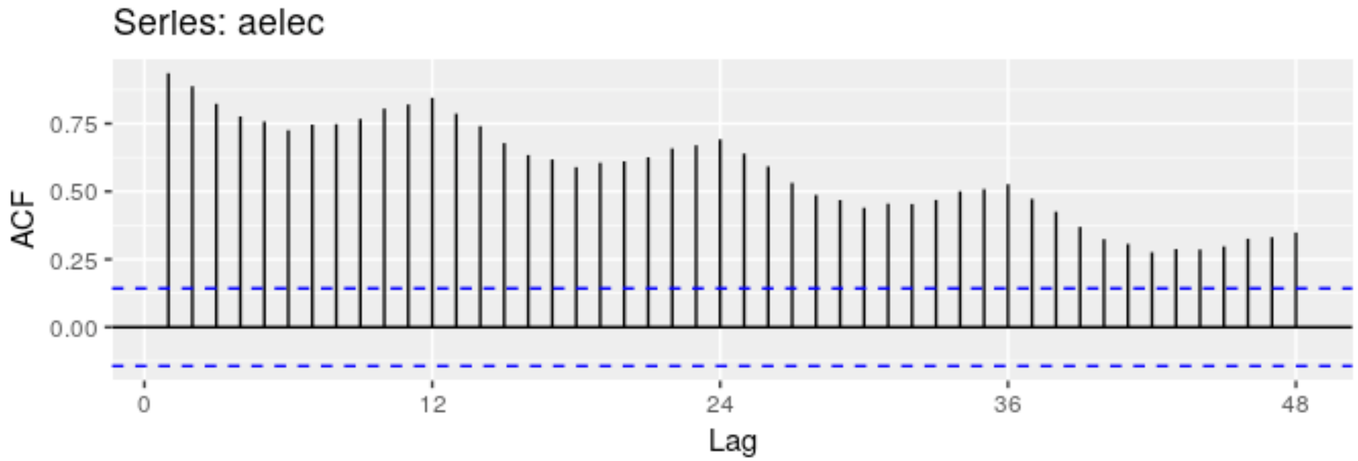
Feature	Value
7 consecutive increasing	0
# > σ	1
# > 3*σ	1
5 consecutive above μ	1

https://en.wikipedia.org/wiki/Control_chart

AUTOCORRELATION

$$r_k = \frac{\sum_{t=k+1}^{T} (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T} (y_t - \bar{y})^2},$$





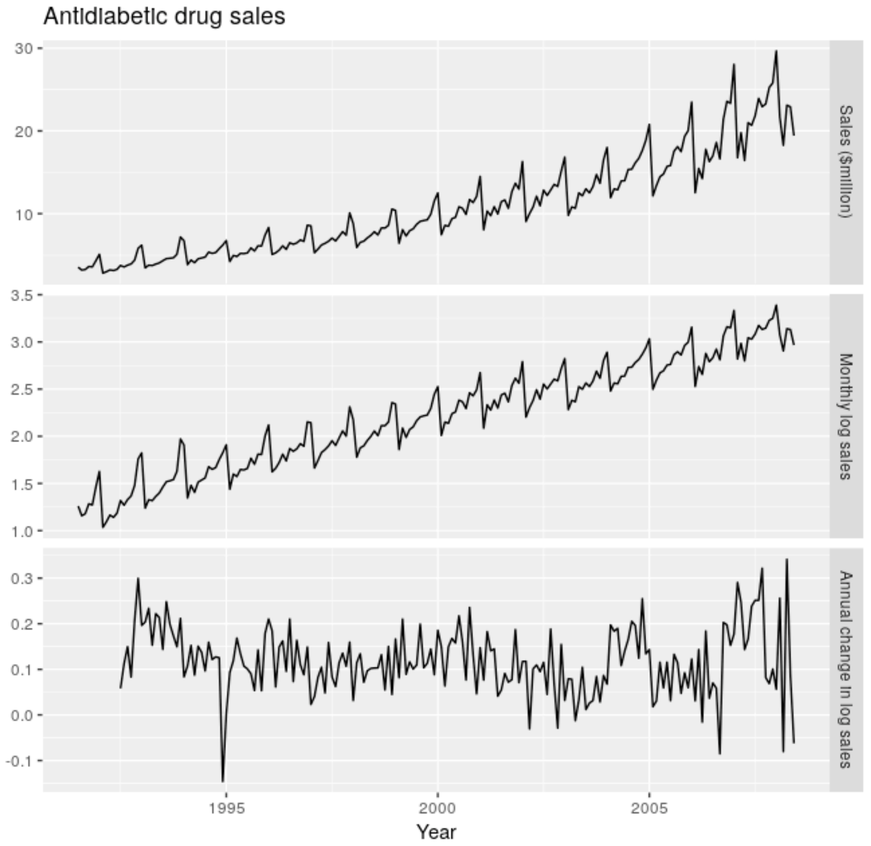
Forecasting using ARIMA

AutoRegressive Integrated Moving Average.

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t,$$

- **AR**: *Autoregression*. A model that uses the dependent relationship between an observation and some number of lagged observations.
- **I**: *Integrated*. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
- **MA**: *Moving Average*. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Learn more here: https://otexts.com/fpp2/index.html

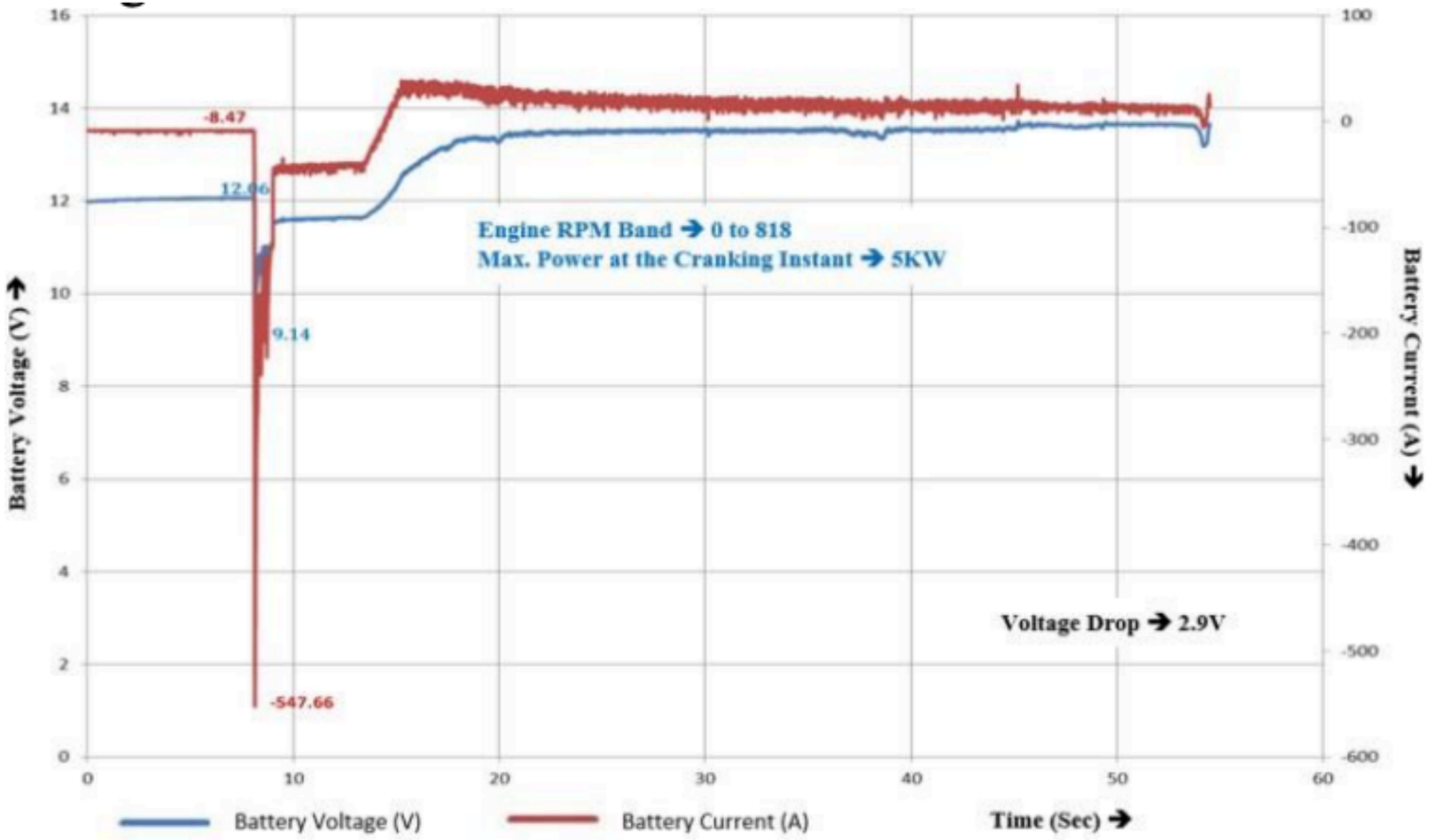


Do you like features?

```
from tsfresh 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 l the sequence from (
approximate_entropy(x, m, r)	Implements a vecto
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 esti valleys etc.).
count_above_mean(x)	Returns the number of values in x
count_below_mean(x)	Returns the number of values in x
cwt_coefficients(x, param)	Calculates a Continuous wavelet transform for the Ricker wavelet, also known as the “Mexican hat wavelet” which is
energy_ratio_by_chunks(x, param)	Calculates the sum of squares of chunk i out of N chunks expressed as a ratio with the sum of squares over the whole series.
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
	quence in x that is smaller than the mean of x
	x)=0}` estimated from polynomial ,
	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.
	correlation function at the given lag.
	es, that are present in the time series more than once.
percentage_of_reoccurring_values_to_all_values(x)	Returns the ratio of unique values, that are present in the time series more than once.
quantile(x, q)	Calculates the q quantile of x.
range_count(x, min, max)	Count observed values within the interval [min, max).
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.

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

MORE MODELS

Specific applications require specific models.

Kaplan Meier estimator (product limit)

The estimator of the survival function $S(t)$ (the probability that life is longer than t) is given by:

$$\widehat{S}(t) = \prod_{i:\ t_i \leq t} \left(1 - \frac{d_i}{n_i}\right),$$

with t_i a time when at least one event happened, d_i the *number of events* (i.e., deaths) that happened at time t_i and n_i the *individuals known to have survived* (have not yet had an event or been censored) up to time t_i.

https://en.wikipedia.org/wiki/Kaplan%E2%80%93Meier_estimator

Cox Proportional Hazards model

Hazard function

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t}$$

Baseline hazard

$$h_0(t)$$

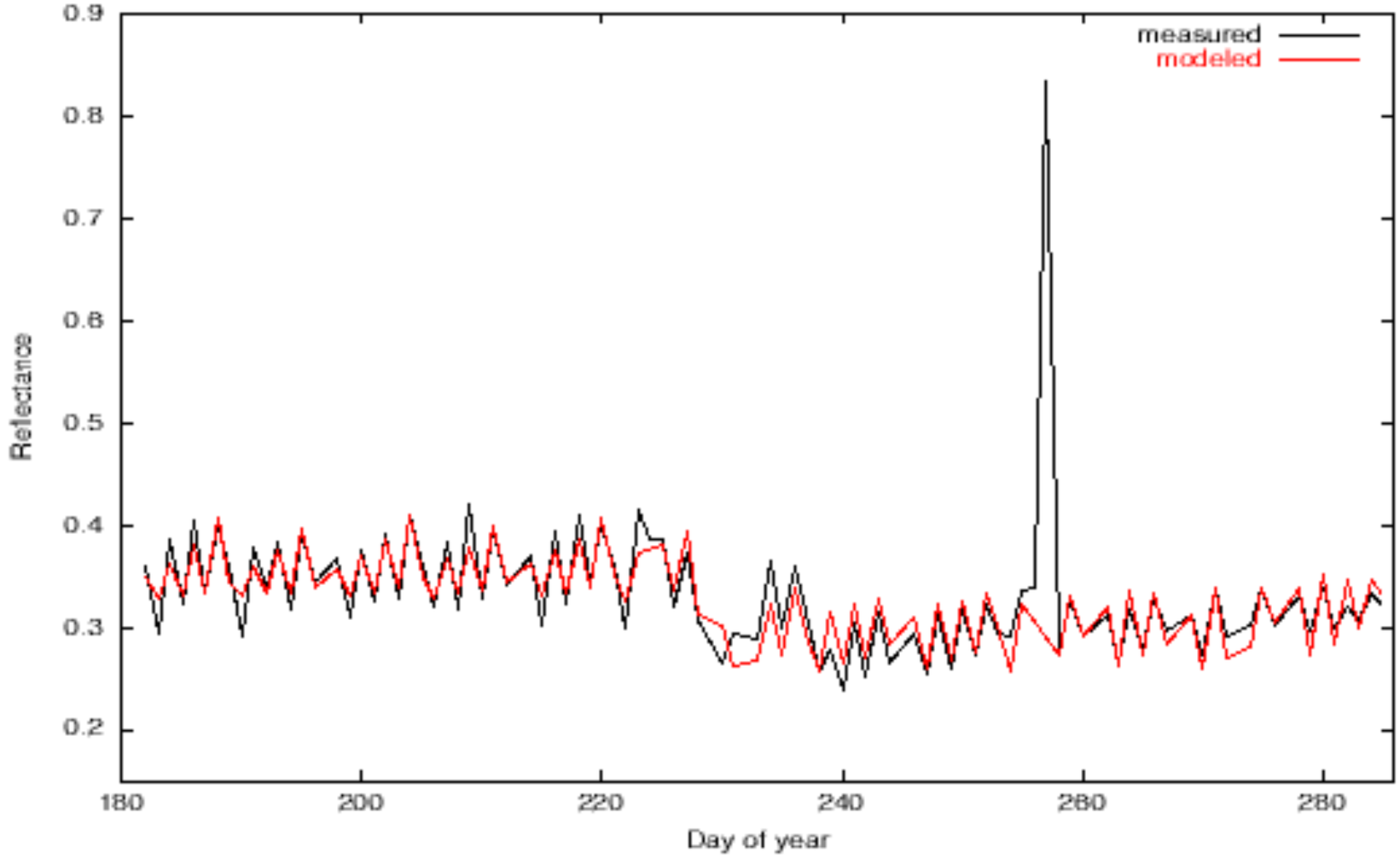
Hazard of individual i

$$h_i(t) = h_0(t) e^{\beta_1 X_{i1} + \cdots + \beta_p X_{ip}}$$

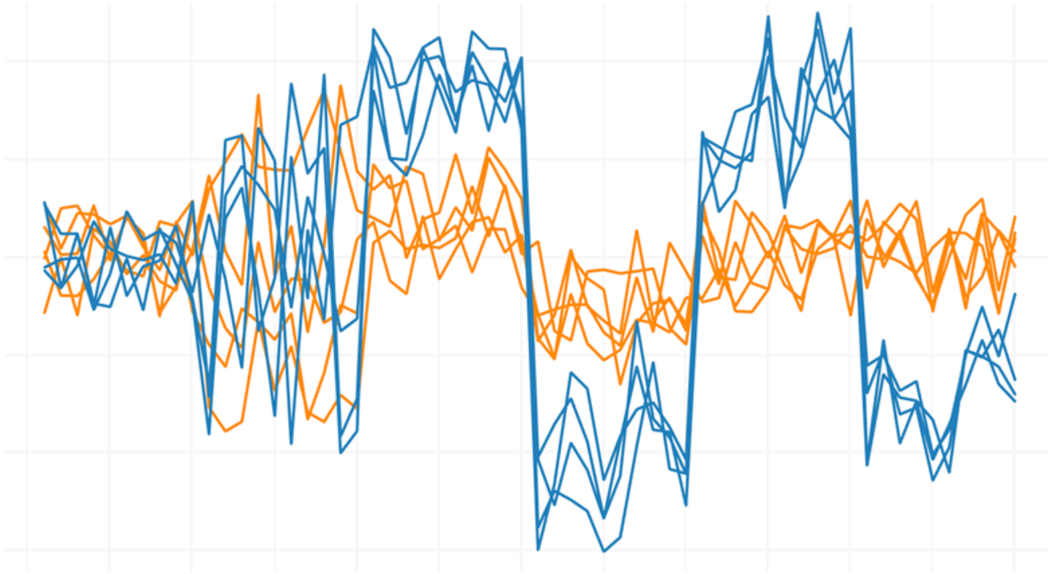
Time independent covariates

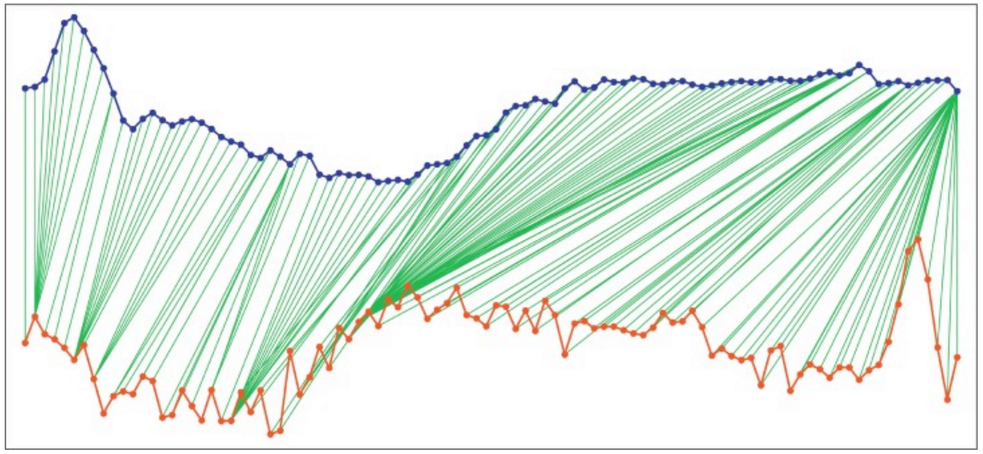
Advanced anomaly detection

It's just forecasting! But with an extra step



Time Series Clustering and Regression

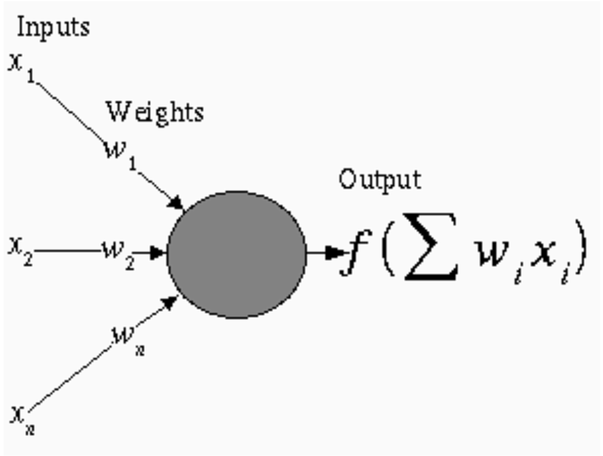




- k-means but with an L2 norm as the distance metric!
- Dynamic Time Warping
- Kernel prediction methods
- ...

DEEP LEARNING BASED APPROACHES

Basic Neuron Model In A Feedforward Network



» **Inputs x_i** arrive through pre-synaptic connections

» Synaptic efficacy is modeled using real **weights w_i**

» The response of the neuron is a **nonlinear function *f*** of its weighted inputs

$$f(x) = \frac{1}{1+e^{-\lambda x}}$$

$$TSSE = \frac{1}{2} \sum_{patterns} \sum_{outputs} (desired - actual)^2$$

Loss function: $RMSE = \sqrt{\frac{2 * TSSE}{\#\ patterns * \#\ outputs}}$

Neuron error $\delta_{pj}=(T_{pj}-O_{pj})\ O_{pj}\ (1-O_{pj})$

T_{pj} is the target value of output neuron j for pattern p

O_{pj} is the actual output value of output neuron j for pattern p

High level algorithm

1. Calculate loss with current weights Wji
2. Compute weight adjustments ΔW_{ji} at time t by

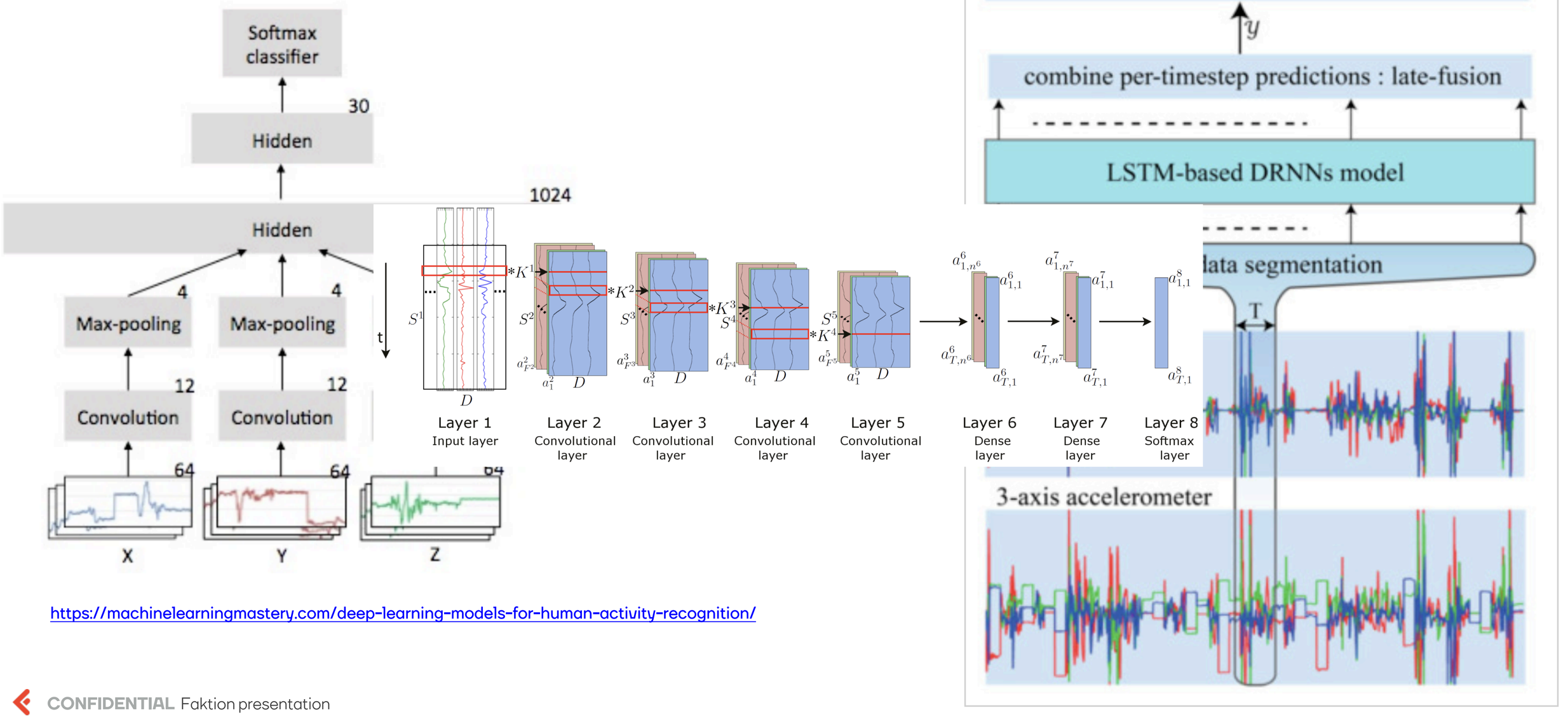
   $\Delta W_{ji}(t) = \eta\ \delta_{pj}\ O_{pi}$

3. Apply weight adjustments according to

   $W_{ji}(t+1) = W_{ji}(t) + \Delta W_{ji}(t)$

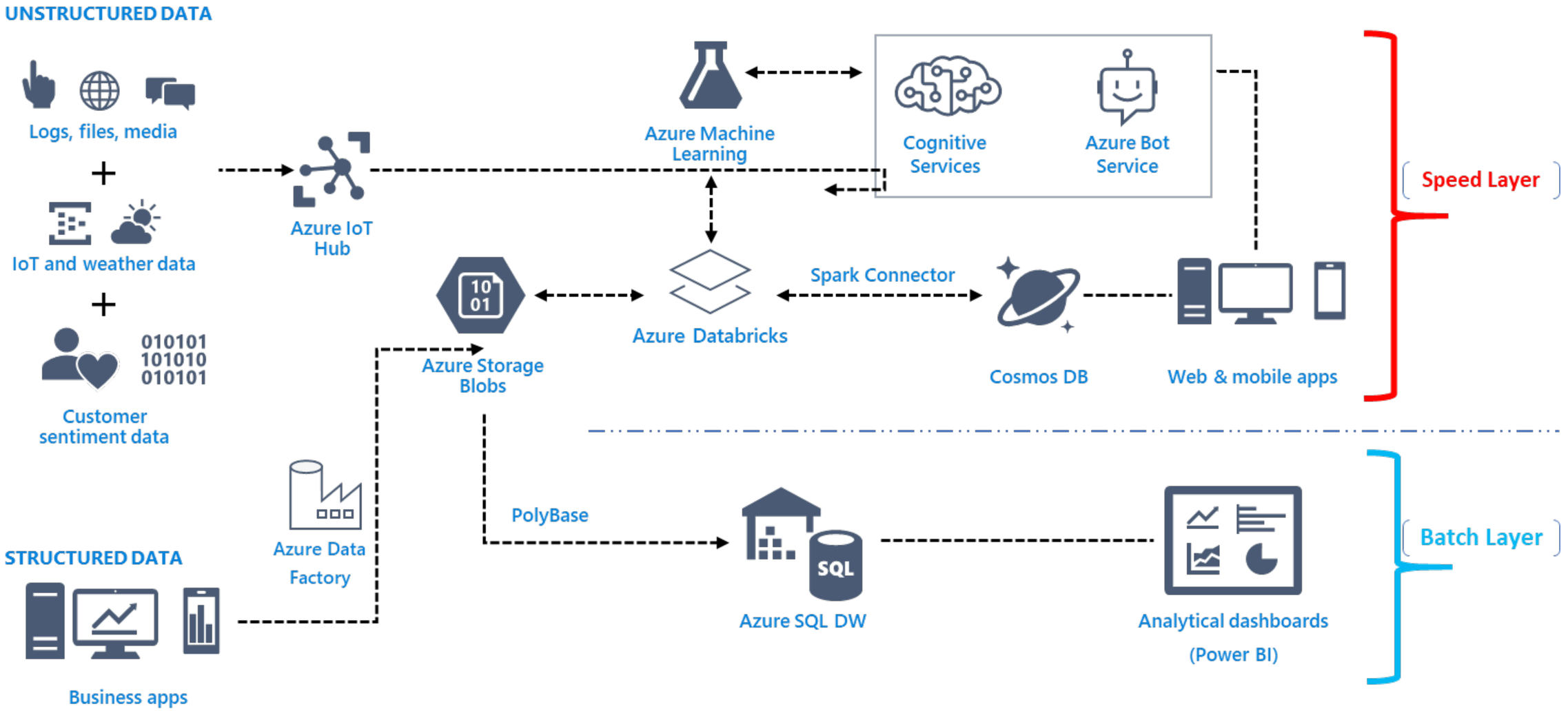
4. Iterate until loss is acceptable or convergence

CONVOLUTIONAL AND RECURRENT



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

Technical Architecture



THIS IS ONLY TOP OF THE ICEBERG.
ARE YOU GETTING MORE

CURIOUS ?

CAN YOU CODE?

START AN AMAZING CAREER AT FAKTION

Get in touch!

FAKTION
ARTIFICIAL INTELLIGENCE. APPLIED.

Oudeleeuwenrui 39
2000 Antwerpen
www.faktion.com

ARE YOU GRADUATING
AND DO YOU LIKE MACHINE LEARNING?

Get in touch! Seriously, we can't imagine why you would miss the opportunity to get to know us better in a face-to-face conversation. It's the only way to see how we could fit. And for sure, at least we'll have an interesting chat.

ARE YOU A STUDENT
AND HAVE AN INTEREST IN MACHINE LEARNING?

Work with real-world data and gain experience on how the industry uses Machine Learning right now. This means we won't let you do academic research and we go beyond the typical applications of the big tech players. Instead, you'll work with experts and help implement Machine Learning models, projects and products that will help real people and companies in Belgium and beyond.

- **MASTER/BACHELOR THESIS**
- **INTERNSHIP**
- **PAID SUMMER JOB**

THANK YOU

FAKTION
May 14, 2019