Machine Learning and A.I.
for the Smart City and the smart building

Chapter 3

Sensors and Time Series analysis

Gilles MOREL – UTC / GSU / Avenues – 2020

Sections of this chapter

1. Sensors and Time Series (TS) in the smart building
2. TS fundamental properties
3. TS basic analysis and pre-processing
4. TS dimensionality reduction and feature extraction
5. Machine learning models applied to TS
6. Predicting the future from the past with ARIMA models
7. Complete examples of ML applied to sensors and TS in the smart building
8. TS special issues and conclusions

Timeseries (TS) are forever and paramount …

- Timeseries capture the change of data in time, not only the state at time t
- Machine learning with TS : understanding the past to predict the future
- They participate to the « numerical twin » of the world :
  - GIS (Geographic Information System) + TS : numerical twin of the city
  - BIM (Building Information Model) + TS : numerical twin of the building
- TS are everywhere in data science, especially for the smart city and the smart building
  - But also in medecine, finance, industry, environment, automobile …

SENSORS AND TIME SERIES IN THE SMART BUILDING

Sensors and IoT in the smart building

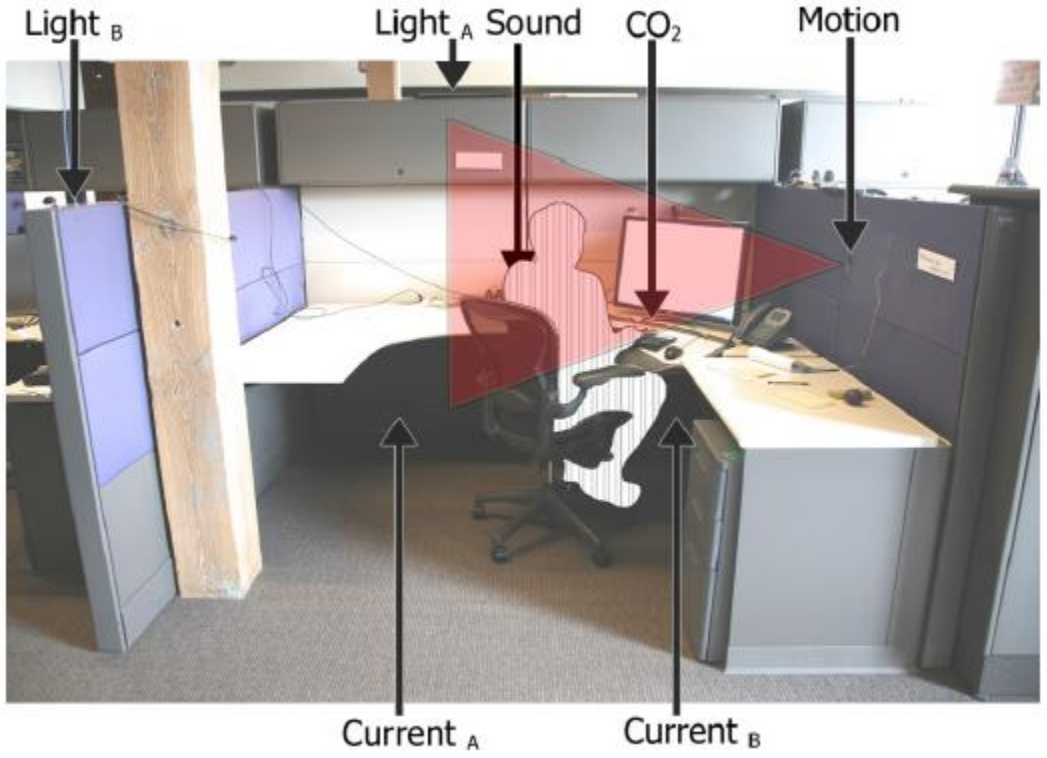
- The majority of the data in the smart building (and the smart city) come from sensors that produce time series (technical data acquired at regular intervals)
- Traditional data (coming from BMS – building management system) or new sensors are all integrated in a local IoT and connected to a common gateway that collects all information
- The role of the gateway or the BOS (building operating system) is to homogeneize the data to the same format (into a DB) to facilitate data processing
- Ideally, at the end we get a database with timeseries at the same timestamps for all the device of the building

Sensors in the smart building produce timeseries

Ref / copyright : Autodesk – « project Dasher » - Kean Walmsley

Autodesk building in Toronto is a smart building (here the numerical twin)

Sensors in the smart building



Ref / copyright : Autodesk
« project Dasher » - Kean Walmsley

Example of sensors installation in an office (Autodesk Toronto)
Light, CO2, temperature, occupancy, sound and energy use

Examples of (timeseries) database format in smart buildings

Smart building with 2000 sensors
1 data every 15 mn for each (here temp.)

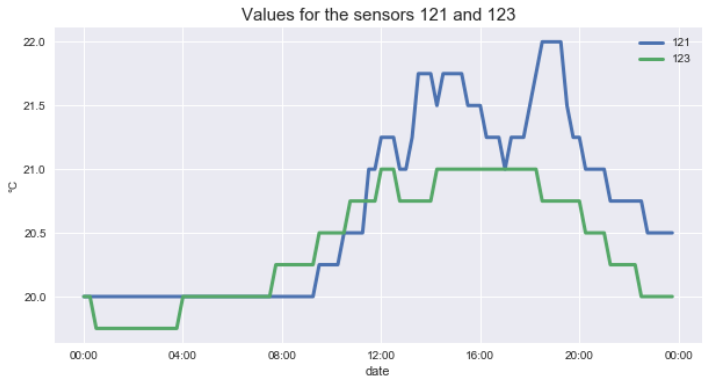
date_debut	point_mes	valeur	qualite
2012-08-01 00:00:00+02	1	24	0
2012-08-01 00:15:00+02	1	24	0
2012-08-01 00:30:00+02	1	24	0
2012-08-01 00:45:00+02	1	24	0
2012-08-01 01:00:00+02	1	24	0
2012-08-01 01:15:00+02	1	24	0
2012-08-01 01:30:00+02	1	24	0
2012-08-01 01:45:00+02	1	24	0
2012-08-01 02:00:00+02	1	24	0
2012-08-01 02:15:00+02	1	23.75	0
2012-08-01 02:30:00+02	1	23.75	0
2012-08-01 02:45:00+02	1	23.75	0

...

2012-08-01 00:00:00+02	122	22.25	0
2012-08-01 00:15:00+02	122	22.25	0
2012-08-01 00:30:00+02	122	22.25	0
2012-08-01 00:45:00+02	122	22.25	0
2012-08-01 01:00:00+02	122	22.25	0
2012-08-01 01:15:00+02	122	22.25	0
2012-08-01 01:30:00+02	122	22.25	0

Multisensor AEOTEC 6 data (UTC room)
Temp., humidity, light, presence … (every 15 mn)

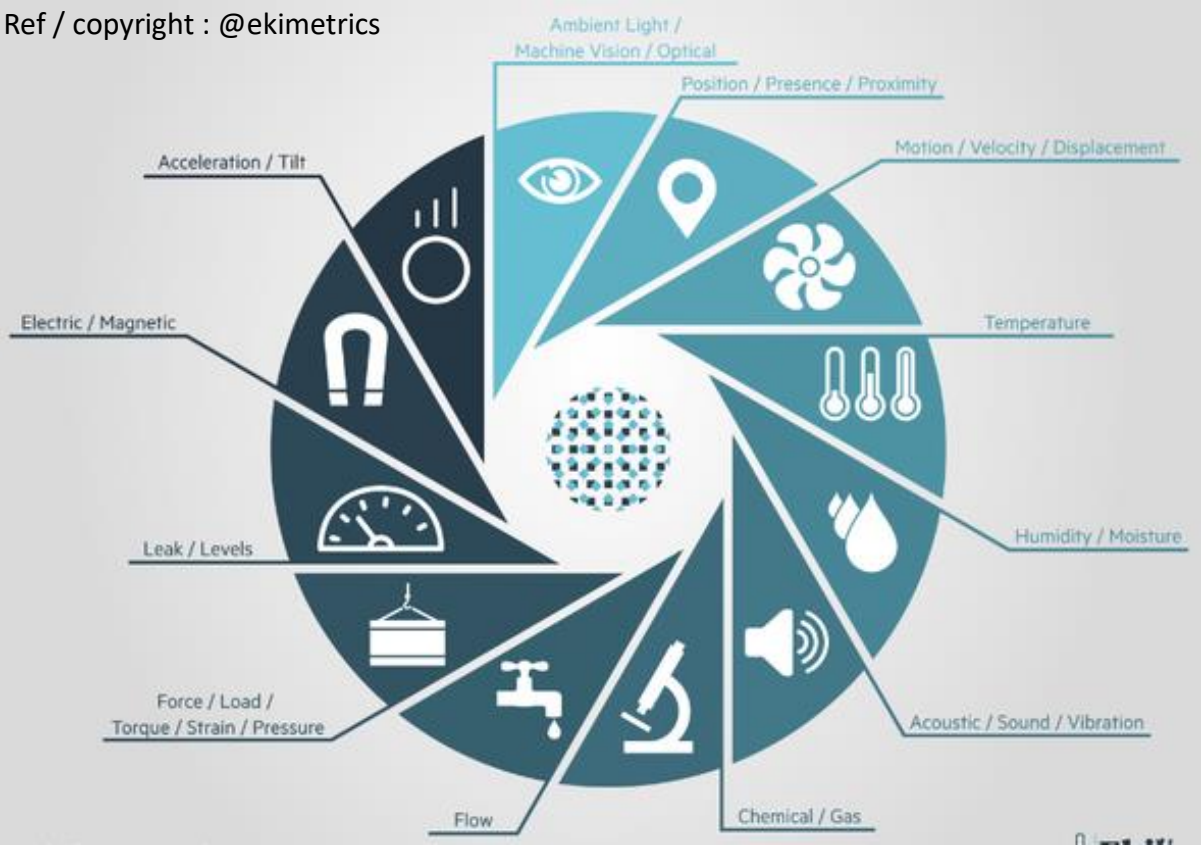
,entity_id,state,last_changed
0,sensor.aeon_labs_zw100_multisensor_6_temperature,18.8,2020-04-16 12:15:00+00:00
1,sensor.aeon_labs_zw100_multisensor_6_relative_humidity,45.0,2020-04-16 12:15:00+0
2,sensor.aeon_labs_zw100_multisensor_6_ultraviolet,0.0,2020-04-16 12:15:00+00:00
3,sensor.aeon_labs_zw100_multisensor_6_luminance,84.0,2020-04-16 12:15:00+00:00
4,sensor.aeon_labs_zw100_multisensor_6_burglar,7.0,2020-04-16 12:15:00+00:00
5,sensor.aeon_labs_zw100_multisensor_6_temperature,18.9,2020-04-16 12:30:00+00:00
6,sensor.aeon_labs_zw100_multisensor_6_luminance,92.0,2020-04-16 12:30:00+00:00
7,sensor.aeon_labs_zw100_multisensor_6_ultraviolet,0.0,2020-04-16 12:30:00+00:00
8,sensor.aeon_labs_zw100_multisensor_6_burglar,6.0,2020-04-16 12:30:00+00:00
9,sensor.aeon_labs_zw100_multisensor_6_relative_humidity,46.0,2020-04-16 12:30:00+0
10,sensor.aeon_labs_zw100_multisensor_6_relative_humidity,46.0,2020-04-16 12:45:00+
11,sensor.aeon_labs_zw100_multisensor_6_temperature,19.0,2020-04-16 12:45:00+00:00
nultisensor_6_luminance,92.0,2020-04-16 12:45:00+00:00
nultisensor_6_ultraviolet,0.0,2020-04-16 12:45:00+00:00



Typology of sensors in a smart building

The view of device providers and data scientist

Ref / copyright : @ekimetrics



Ref / copyright : @neothings

1. **Lighting**
   Lights control to provide the right luminosity where and when it is needed
2. **Control Panel**
   Access control panel via interactive touch screen devices or from mobile
3. **Occupancy Detection**
   Occupancy and motion sensors for a comfortable space
4. **Heating, ventilation and air-conditioning**
   Optimum climate, temperature and air control
5. **Power Supply**
   Stable bus voltage and safe access to power network data
6. **Management Station**
   Improved maintenance management and energy performance
7. **Energy Efficiency**
   Increase energy savings and reduce building operating costs

Typology of main sensors (left) and their use (right) in the smart building according to private compagnies (Ekymetrics and Neothings)

Example of sensors useful in a smart building (research review)

Sensor	Measurement	Category
Infrared sensor	User presence in a room	Environmental sensors
Video cameras	Human actions	Environmental sensors
RFID	Object identification	Environmental sensors
Motion sensor	Object/User presence/ location	Environmental sensors
Contact switch	Detect users' interaction with the object	Environmental sensors
Pressure sensor	Tracking movements and location of the user	Environmental sensors
Light sensor	Intensity of light	Environmental sensors
Temperature sensor	Temperature of surrounding environment	Environmental sensors
Humidity sensor	Detect the air humidity in a specific area	Environmental sensors
Power sensor	Detect the usage of electric devices	Environmental sensors
Accelerometer	The rate of acceleration accompanying a sensitive axis	Wearable inertial sensors
Gyroscope	Angular velocity and maintain orientation	Wearable inertial sensors
Electroencephalography	observing electrical brain activity	Wearable vital sign sensors
Electrooculography	observing eye movement of ocular activity	Wearable vital sign sensors
Electromyography	observing muscle activity	Wearable vital sign sensors
Electrocardiography	observing cardiac activity, pressure sensors for observing blood pressure	Wearable vital sign sensors
CO2 gas sensors	observing respiration	Wearable vital sign sensors
Thermal sensors	observing body temperature	Wearable vital sign sensors
galvanic skin response	observing skin sweating	Wearable vital sign sensors

Ref: B. Qolomany, A. Al-Fuqaha, A. Gupta, D. Benhaddou, S. Alwajidi, J. Qadir, A. C. Fong, "Leveraging machine learning and big data for smart buildings: A comprehensive survey", *CoRR*, vol. abs/1904.01460, pp. 1-39, May 2019.

Rem : the majority of the sensors actually produce timeseries

Doing machine learning (ML) with timeseries

- In chapter 2, we saw the principles, methods and models of ML in the general case (all kinds of data and format)

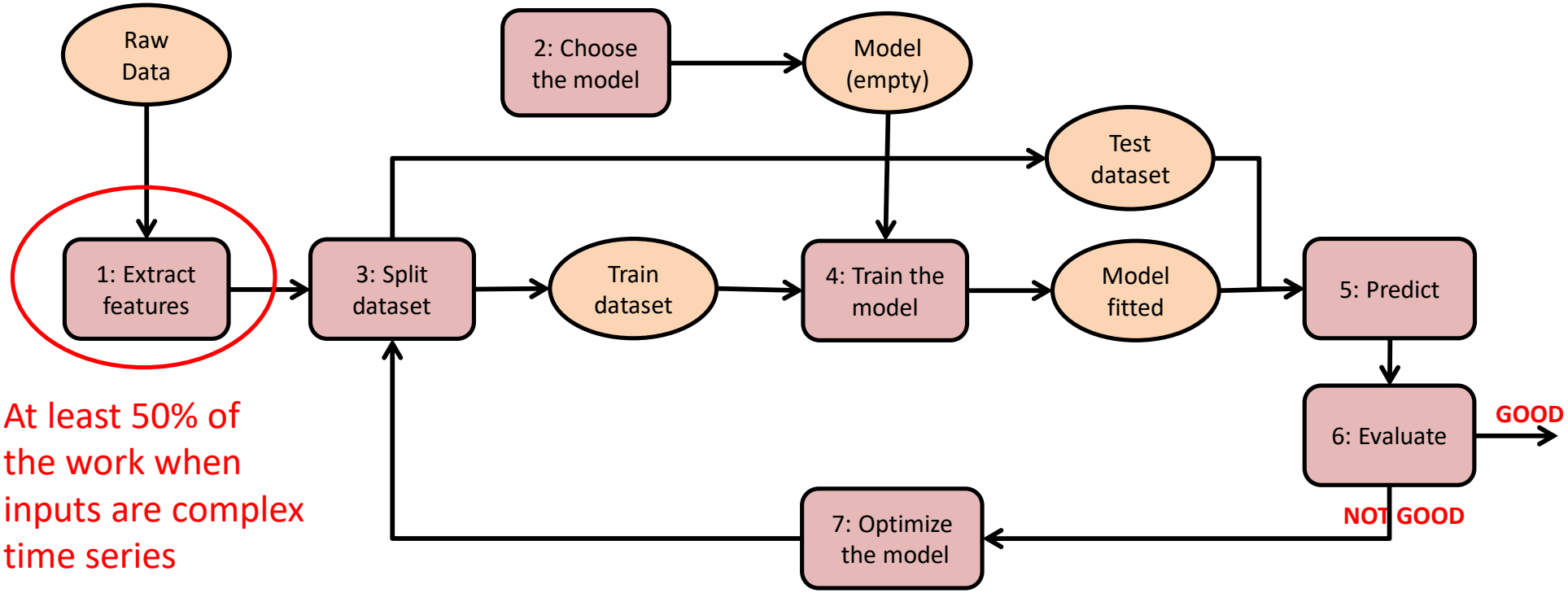
- Timeseries are a special case of data with time as (X) index
  - They need special processing

- The main difference (with other data) is that the first step - data preparation for ML modeling - is more complex and more important (see next sections)

- That's why 2/3 of this chapter is about timeseries analysis that is made before applying an ML model and predicting

Adapting the ML process to TS

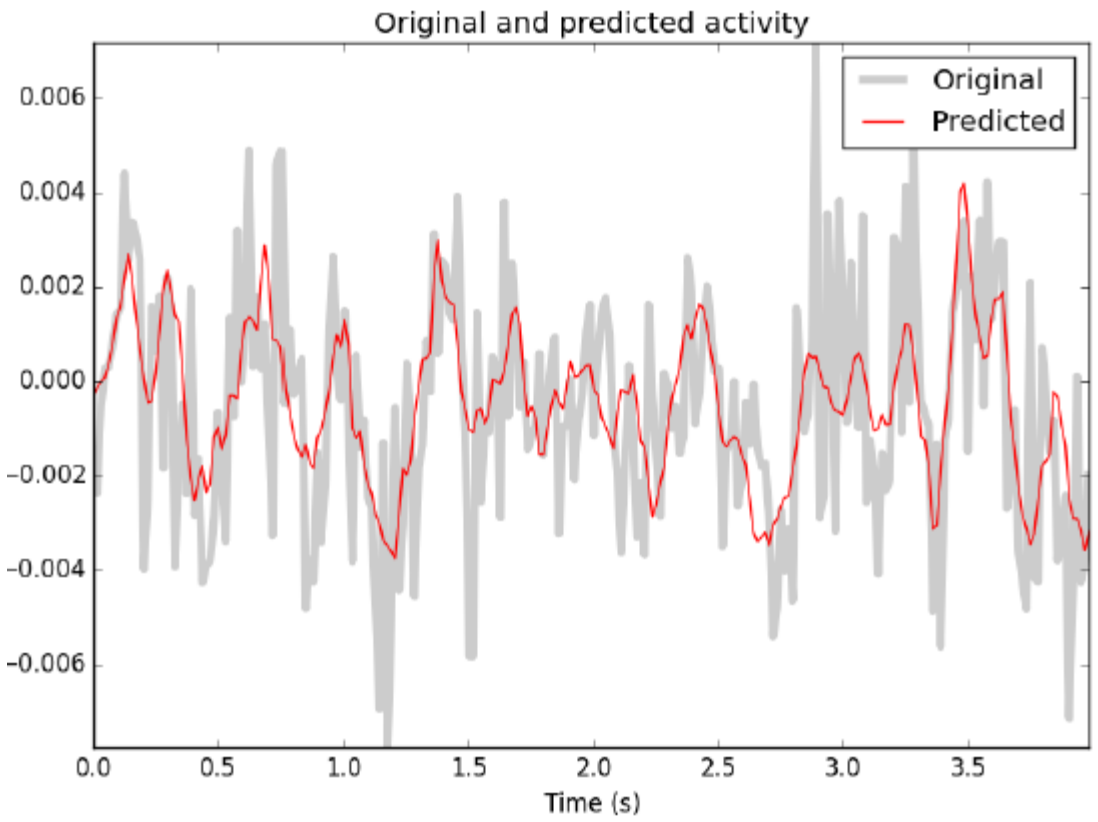
The importance of preparing the data before ML modeling



The general process of ML (chap. 2) is valid for timeseries BUT step 1 « feature extraction » is more important

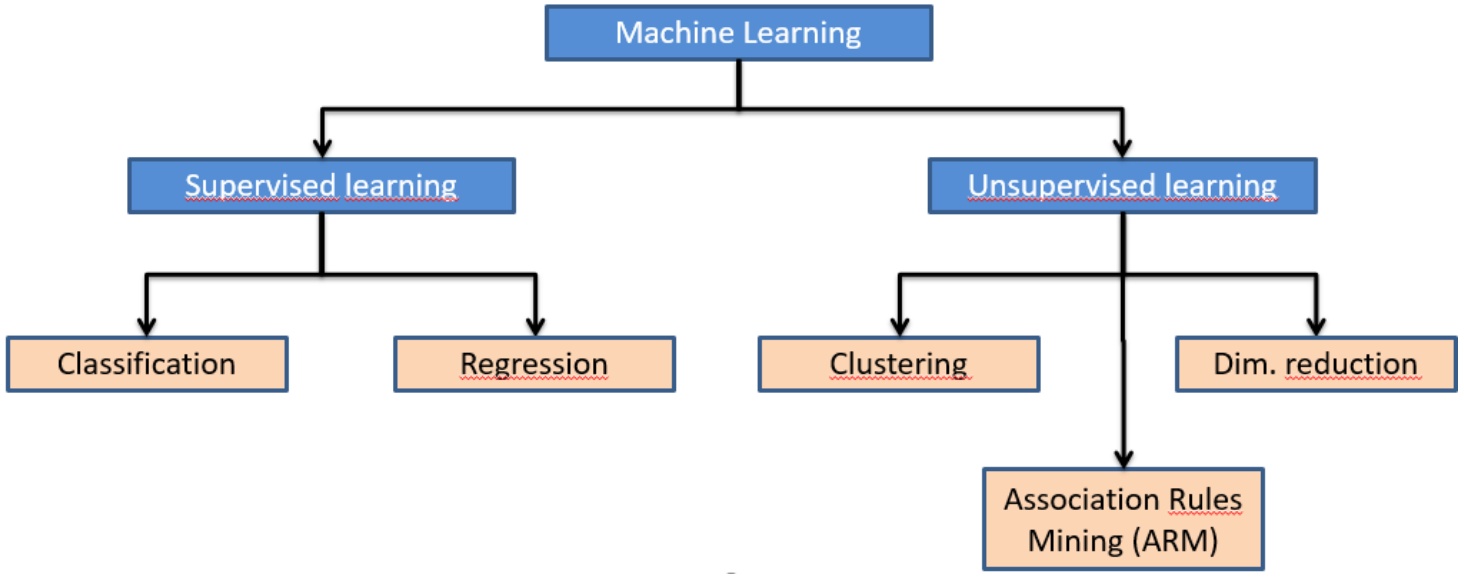
Why applying ML to sensors time series ?

- Detect patterns and abnormal events
  - Preventive maintenance of device
- Predict the future
  - Anticipate the users' needs
- Optimize the systems
  - Energy savings



Which ML models can apply to TS ?

- After feature extractions (step 1), all ML models (see chap. 2) can be possibly applied to TS - Remember the ML method toplevel typology :



- There are some specific models that take into account the time dimension and can predict the future (data) from the past (data)

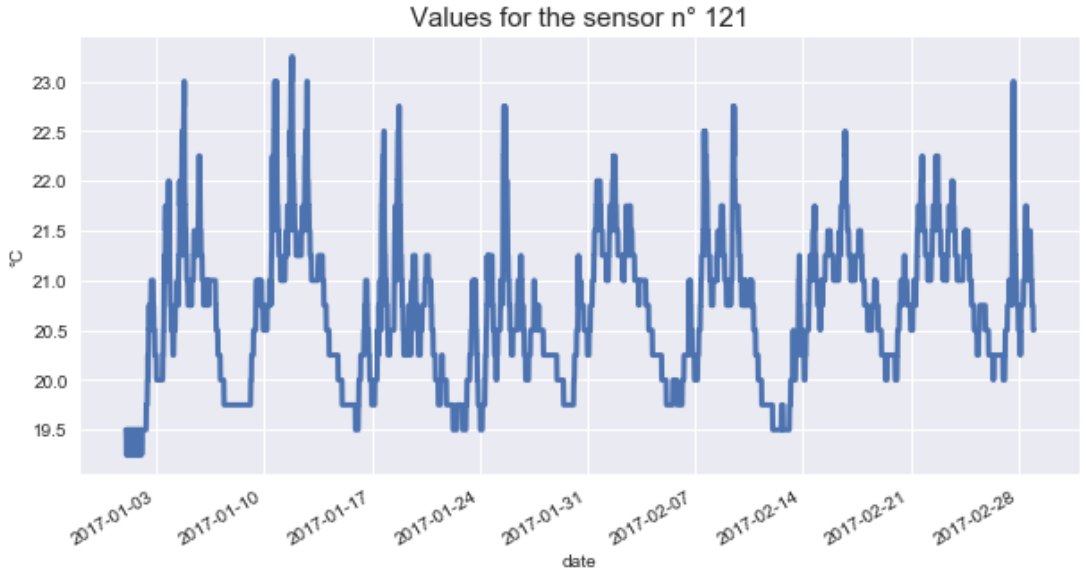
  ➢ See the last § with ARIMA models (these models come from statistics)

Sensors and Time Series Analysis

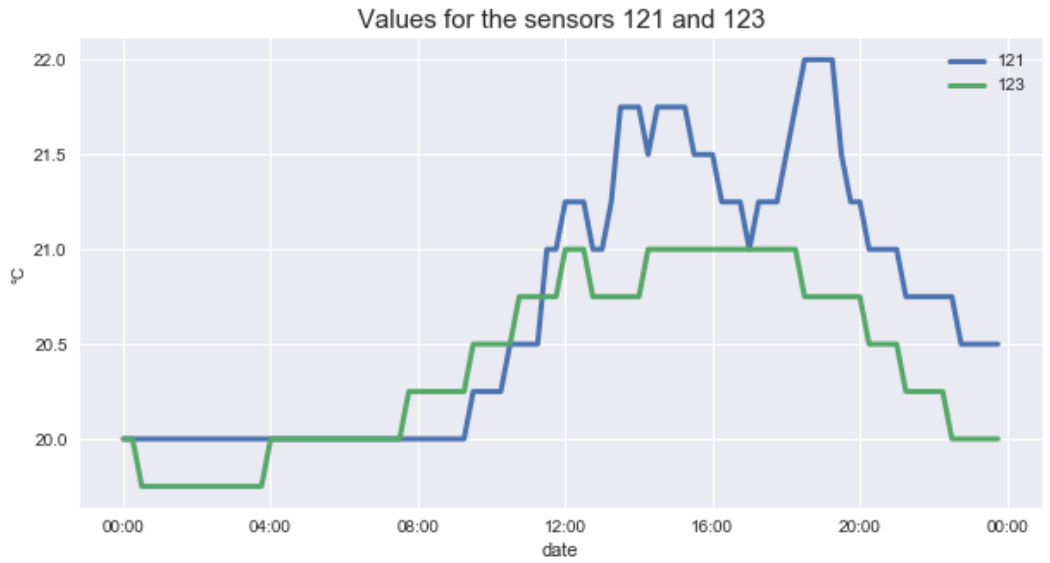
TIME SERIES FUNDAMENTAL PROPERTIES

Examples of time series - Visualizing

Temperatures in a building



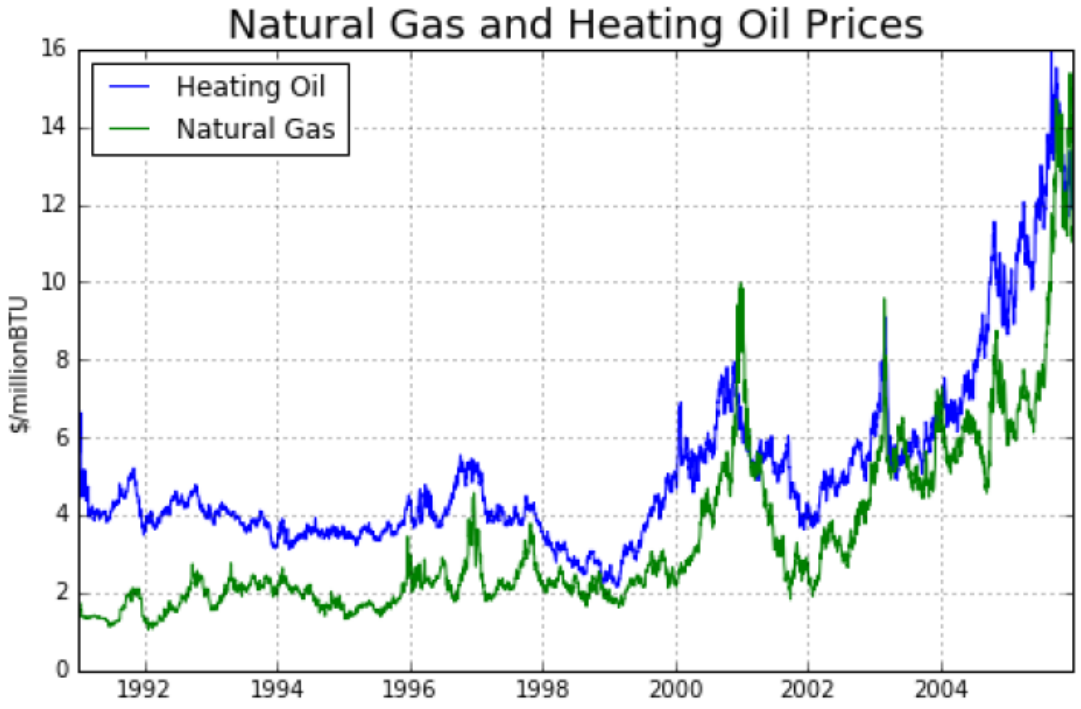
1 sensor data (temperature) for 2 months

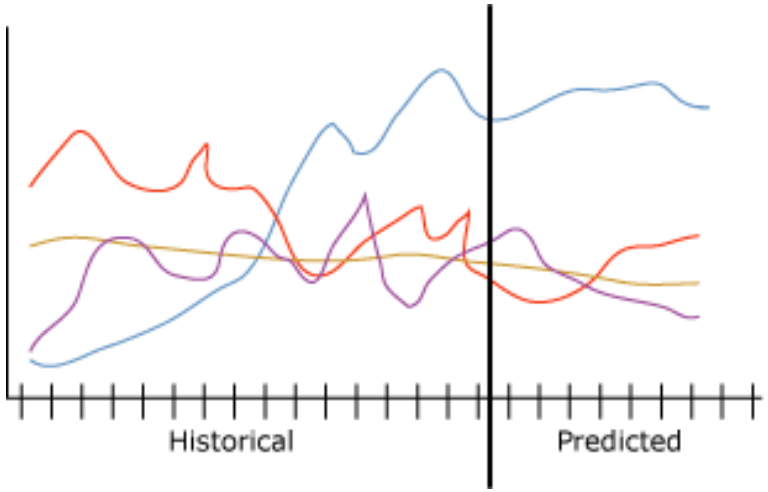


2 sensors for 1 day

Time serie y = f(t) is often the result of data acquisition by a sensor at regular intervals (here 15 mn in a building)

Other examples and first applications





One interest of timeseries is to predict the (short time) future from past data : it has been first applied to the financial domain (lots of money to earn !), and more recently to smart city and smart building issues

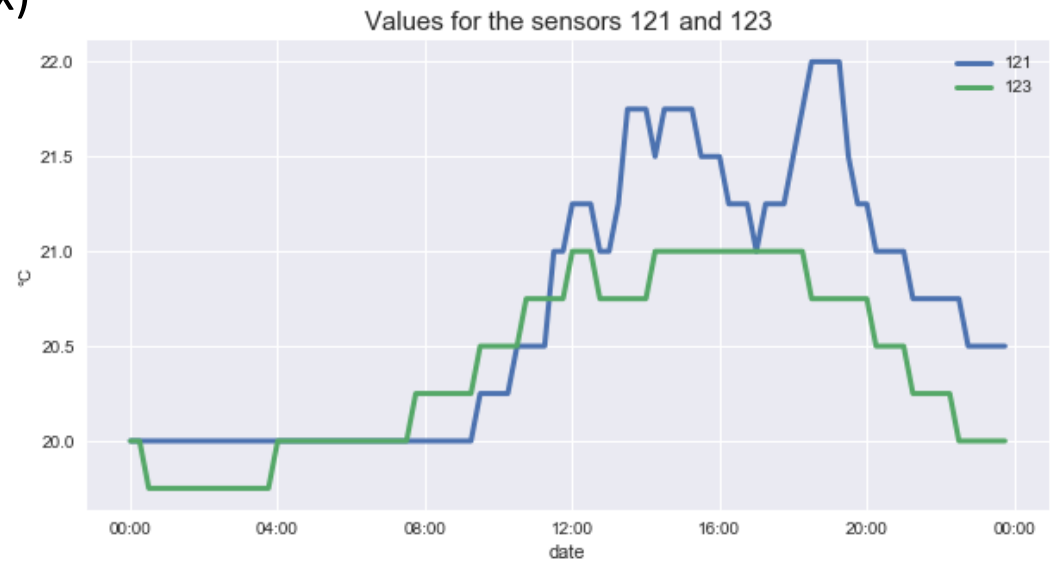
Definition and basic characteristics

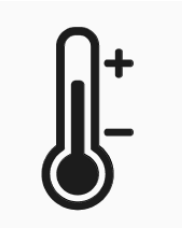
- A time serie is an ordered sequence (serie) of values (y), with the time as index (x index)

- Standard time serie (or sample) shape :

X Time index Y value (temp) = f(t)

date	
2017-01-01 00:00:00+01:00	19.50
2017-01-01 00:15:00+01:00	19.50
2017-01-01 00:30:00+01:00	19.50
2017-01-01 00:45:00+01:00	19.25
2017-01-01 01:00:00+01:00	19.25
2017-01-01 01:15:00+01:00	19.50
2017-01-01 01:30:00+01:00	19.50
2017-01-01 01:45:00+01:00	19.25
2017-01-01 02:00:00+01:00	19.25
2017-01-01 02:15:00+01:00	19.25
2017-01-01 02:30:00+01:00	19.50
2017-01-01 02:45:00+01:00	19.50
2017-01-01 03:00:00+01:00	19.25
2017-01-01 03:15:00+01:00	19.25
2017-01-01 03:30:00+01:00	19.50
2017-01-01 03:45:00+01:00	19.50
2017-01-01 04:00:00+01:00	19.25
2017-01-01 04:15:00+01:00	19.25
2017-01-01 04:30:00+01:00	19.25
2017-01-01 04:45:00+01:00	19.25
2017-01-01 05:00:00+01:00	19.50





X Time index = ordered sequence of date and/or time
The X time points are called timestamps
The intervall between two timestamps (supposed fixed) is called the sampling rate (here one point every 15 mn)

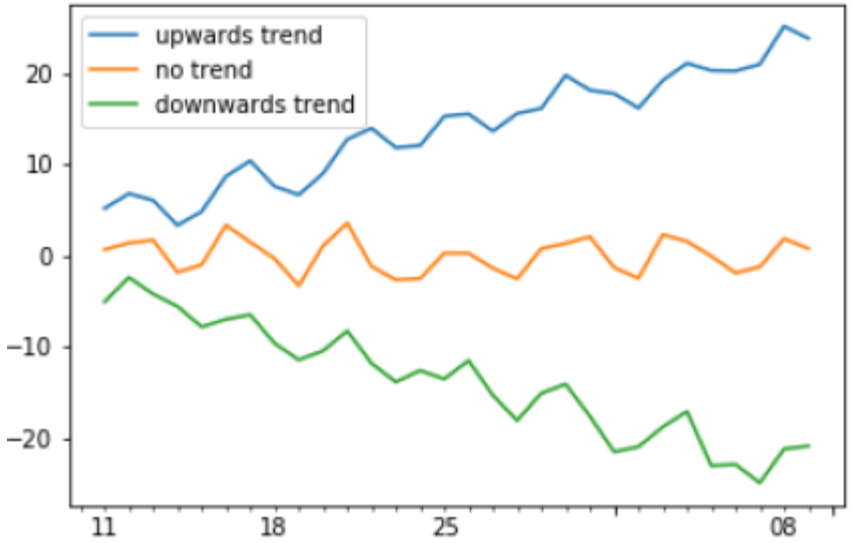
It can vary from 1 ms to 1 year !

Recall of the basic statistics features on a serie

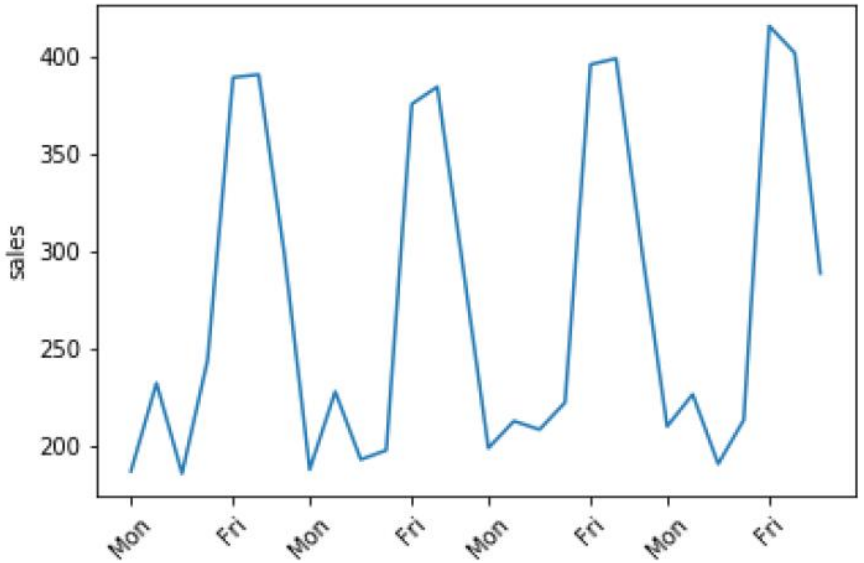
Ref : Wikipedia

- Mean
  - the arithmetic mean, also called the average, is the central value of a discrete set of numbers: specifically, the sum of the values divided by the number of values
- Standard deviation
  - measures the amount of variation or dispersion of a set of values. A low standard deviation indicates that the values tend to be close to the mean
  - standard_deviation = sqrt(sum((x - mean)^2) / count(x))
- Variance (link to standard deviation)
  - the expectation of the squared deviation of a variable from its mean. Informally, it measures how far a set of numbers are spread out from their average value
- Percentile
  - For a given dataset, the Nth percentile is the value where N% of the data is below that datapoint, and 100 - N% of the data is above that datapoint.
- Autocorrelation (see further in this §)
  - also known as **serial correlation**, is the correlation of a signal (or a time serie) with a delayed copy of itself as a function of delay.

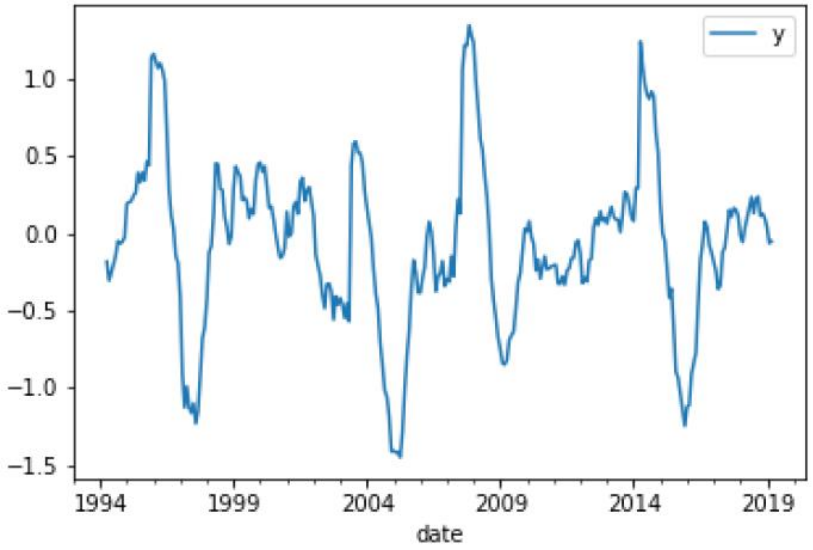
First general properties of timeseries



The notion of « trend »



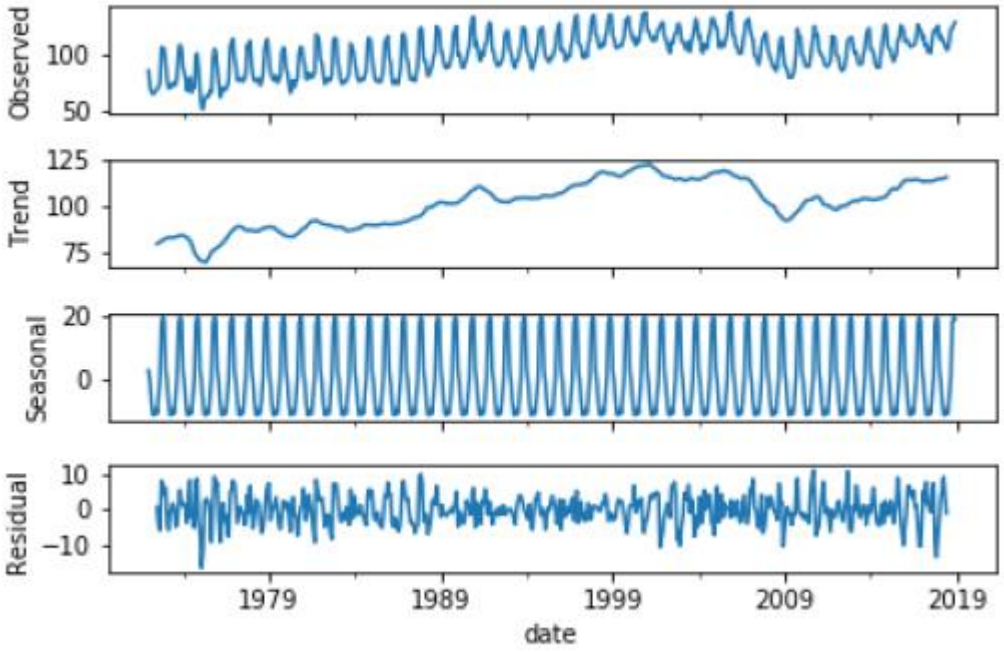
The notion of « seasonality » (here weekly)



The notion of periodicity or cyclicality

Decomposition of a time serie

- A timeserie is often a sum of different components
  - Trend + seasonal + residual
- … more or less visible by plotting
- A decomposition + plot can help distinguish and understand



Decomposition of a timeserie (up) into 3 parts :
- Trend
- Seasonal
- Residual

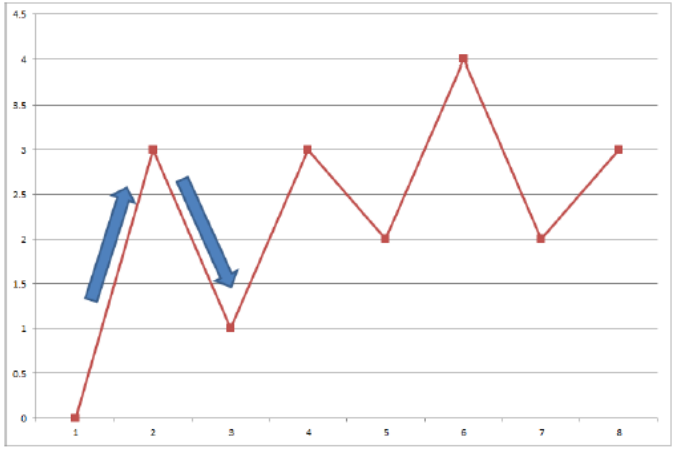
Autocorrelation

- Autocorrelation for a TS is the relation (or the correlation) between y(t) and y(t-1), …, y(t-n)
- It is also called serial correlation
- It is like making a correlation between the serie and a lagged version of itself (t - n)

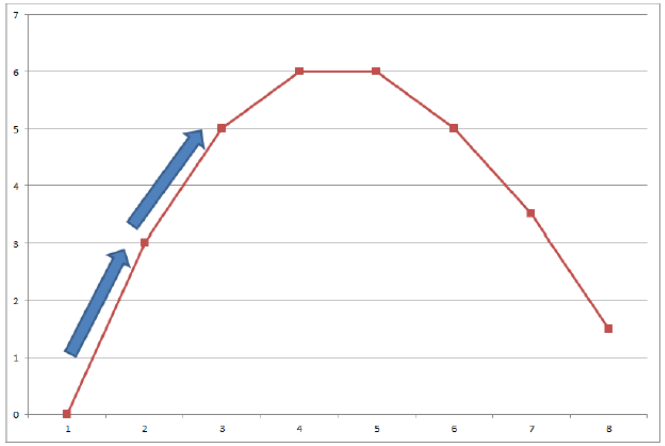
Series	Lagged Series
5	
10	5
15	10
20	15
25	20
⋮	⋮

AutoRegression model (AR) can predict (short term) future values from past : see next § ARIMA models

(ex : financial trading)



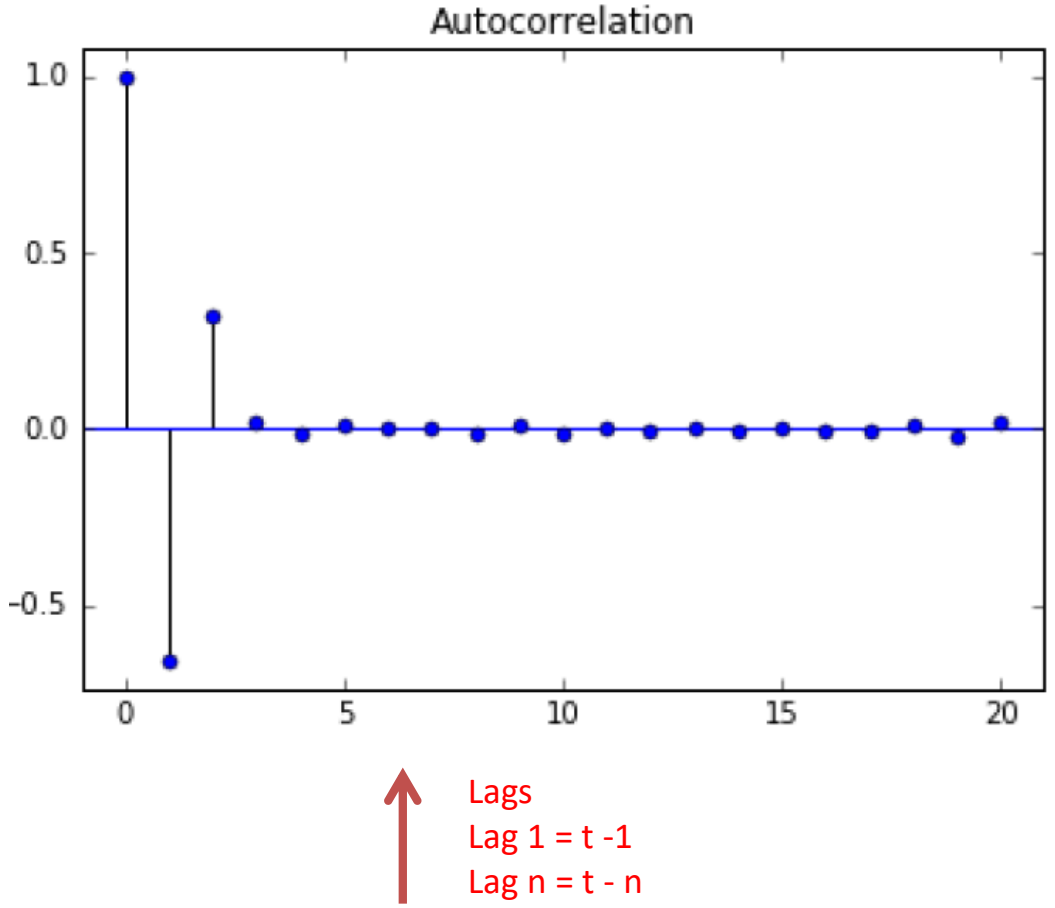
Negative correlation
or « mean reversion »



Positive correlation
or « trend following »

Representing and analysing autocorrelation

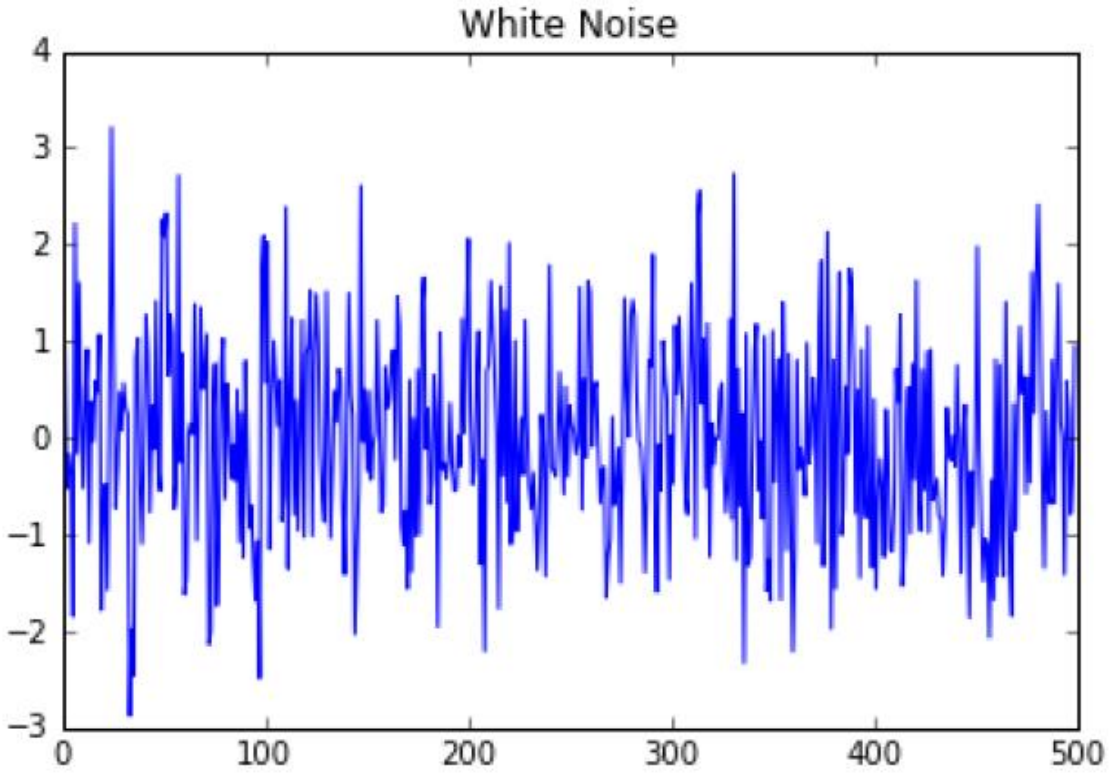
- ACF : is the autocorrelation function with the « lag » as the x axis
  - The value AC is in [0 – 1]
  - The x represents the lag (goes in the past, not in the future)
- AC equals 1 at lag 0 ! And generally decreases with lags far in the past
- 1 lag (sampling rate) can be a second or a mn or 10 mn …
- ACF can help choose and calibrate autoregression models (see the section on ARIMA models)
  - In the example on the right, the two previous values y(t-1) and y(t-2) can be used to predict y(t), the influence of other values in the past seems neglectable
  - Ex : estimate the value of a stock exchange for toomorrow depending on today and yesterday



Rem : it is logic that ACF = 1 at lag 0 (a single data is autocorrelatd with Itself !) and decrease with time (ancient data)

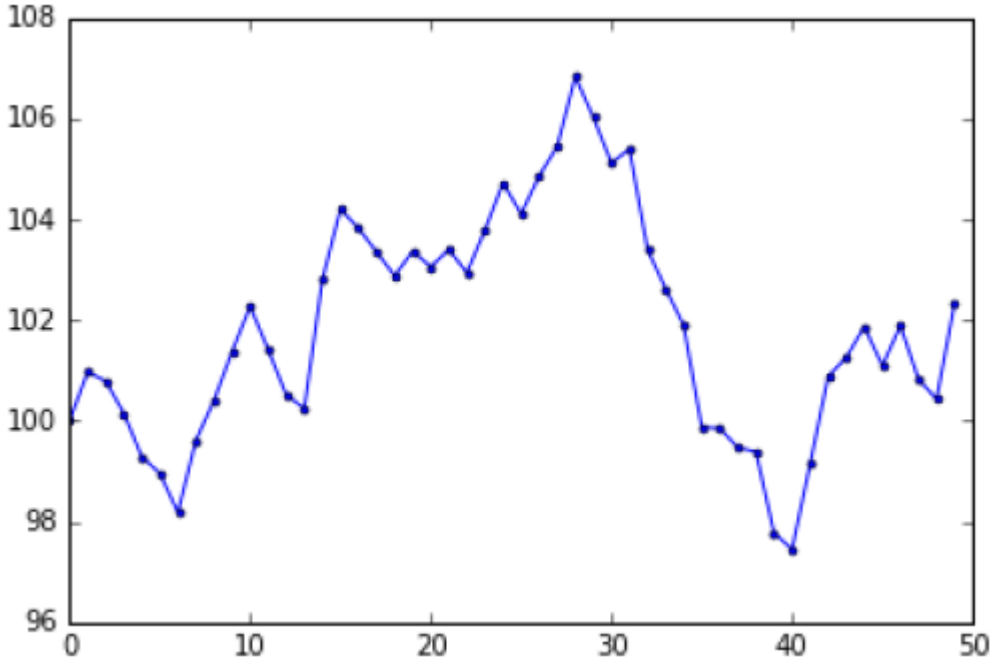
White noise series

- It is a serie where the (next) points are completly unpredictable
- The statistics of a white noise serie are
  - Constant mean (along time)
  - Constant variance
  - (near) Zero correlation (at all lags)
- Conclusion : not interesting for applying ML and make prediction
  - We expect that our sensor series are the contrary of white noise !



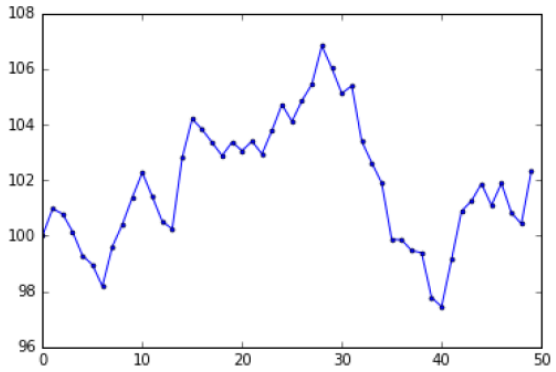
Random walk series

- $y(t) = y(t-1) + \varepsilon$ where ε is a (random) small variation
- Like white noise, random walk series can't be predicted either
  - Ex : The best price for toomorrow prediction is today's price !
  - Fortunately, $y(t) - y(t-1)$ (the diff) is sometime not random !
- Statistical tests :
  - Dickey-Fuller : see the stationarity property

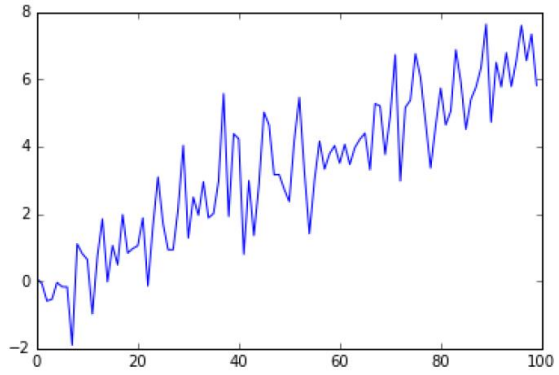


Stationarity

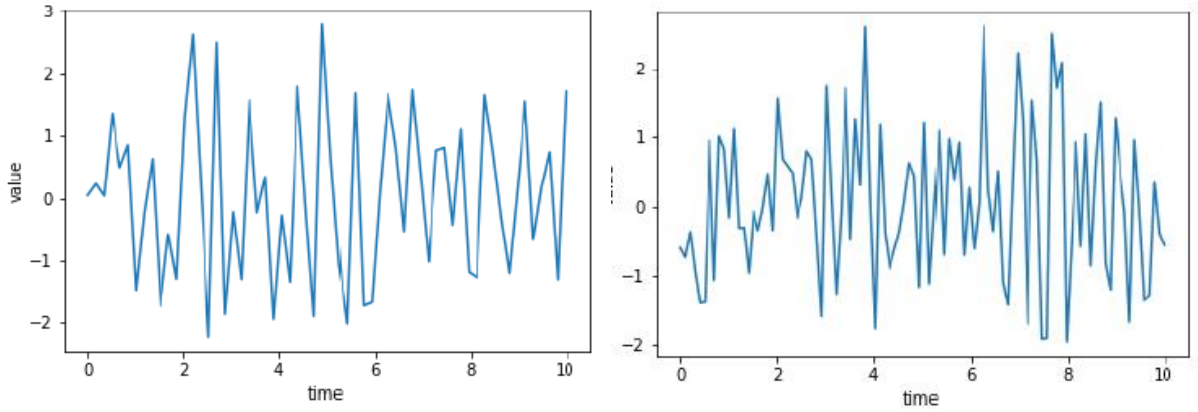
- Stationarity : the statistical properties of the TS don't change over time (mean, std deviation, autocorrelation …)
- This property is necessary to model a TS and make prediction on it
  - Ensures that a model of a TS is stable along time (see the section on the ML model)
- You can measure it with the Augmented Dicky-Fuller test (not developped here)
- Rem :
  - A TS can be « more or less » stationary
  - A TS can be not stationary globally but some parts of them yes
  - It is better to make a serie stationary before applying prediction models (next slide)



Ex 1 : A random walk is not stationary because the autocorrelation is not stable



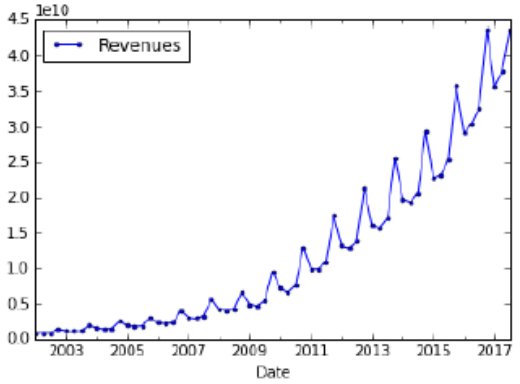
Ex 2 : not stationary because the mean (and deviation) is changing over time



Ex 3/4 : stationary TS – stats don't change over time
Trend is zero, variance and autocorrelation are (quite) constant

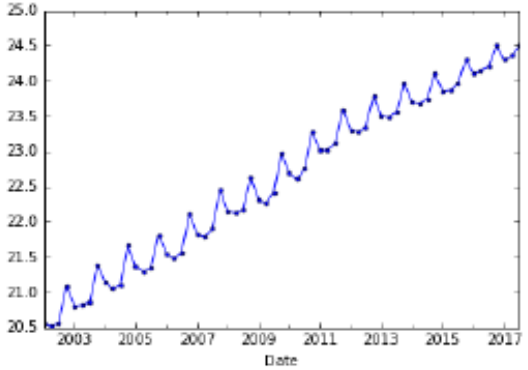
How to make a TS stationary ?

- Real TS from sensors are generally not stationary (null hypothesis), so what to do for modeling and predicting ???
- 1 : test it with the AD-Fuller test
- 2 : If not good, you can transform the TS with some function(s) to make it stationary before modeling
  - Ex: First apply the difference (most common)
  - Other transformations (log, sqrt, dy/dt …)
  - Once applied the model, you have to apply the reverse transformation to come back to original data but :
  - Some models do it for you ! (see ARIMA)



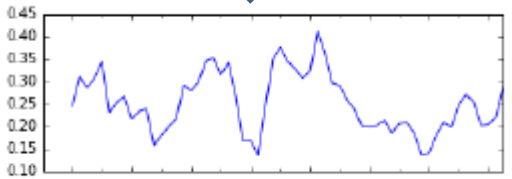
Original data y = f(t)
Not stationary at all

Log(f)



First transfomation with Log
No more exponential but still not stationary (trend is positive)

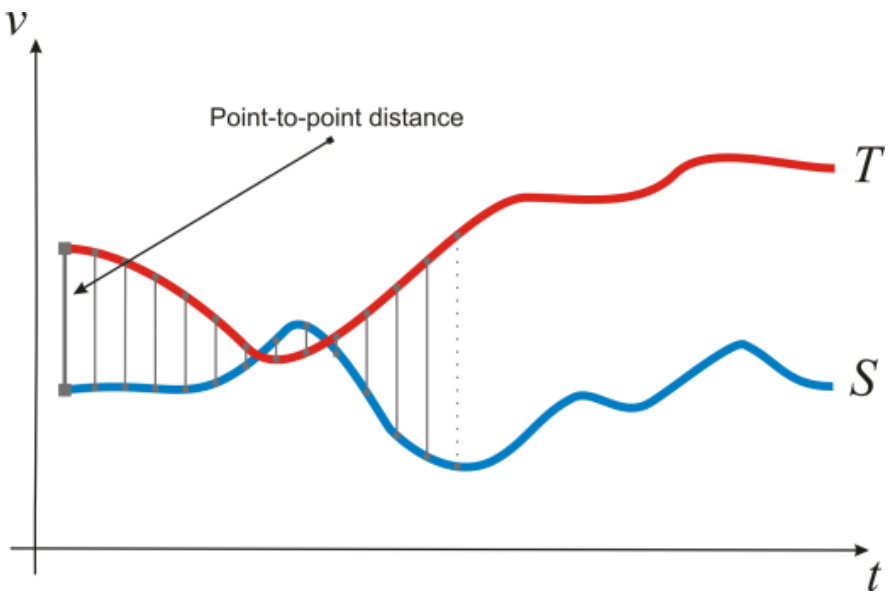
Diff(Log(f))



2nd transformation Diff :
Y(t) = y(t) – y(t-1)
More or less stationary

Similarity measure (distance between TS)

- Given two series y1(t) and y2(t) with t at the same timestamps, we can define a « distance » or a « similarity measure » between them
- This a measure used to compare series
  1. 2 series on the same whole length t
  2. Or one sliding serie S2 as a subsequence of S1
- Ex (most simple see on the right) : $d = \sum |y2 - y1|$
  - This is the Euclidian distance between the 2 curves
  - Lots of distance exist in the litterature
- It can be defined directly on the raw TS (shape) or on tranformed version or on some limited features (ex : on the TFT – see further)
- Then this distance can be used in some models of classification or clustering, like KNN (k nearest neighbours)



Basic Euclidian similarity measure with point-to-point distance

Sensors and Time Series Analysis

TIME SERIES BASIC ANALYSIS, PRE-PROCESSING (AND DIMENSIONALITY REDUCTION)

Why raw TS has to be pre-processed before modeling ?

- Raw data are generaly too « messy », too complex, too « dirty », too numerous, too heterogeneous to be injected directly into a model
  - A lot of reasons for cleaning, reducing, selecting, extracting …
  - Sensors are particularly subject to noise and outliers

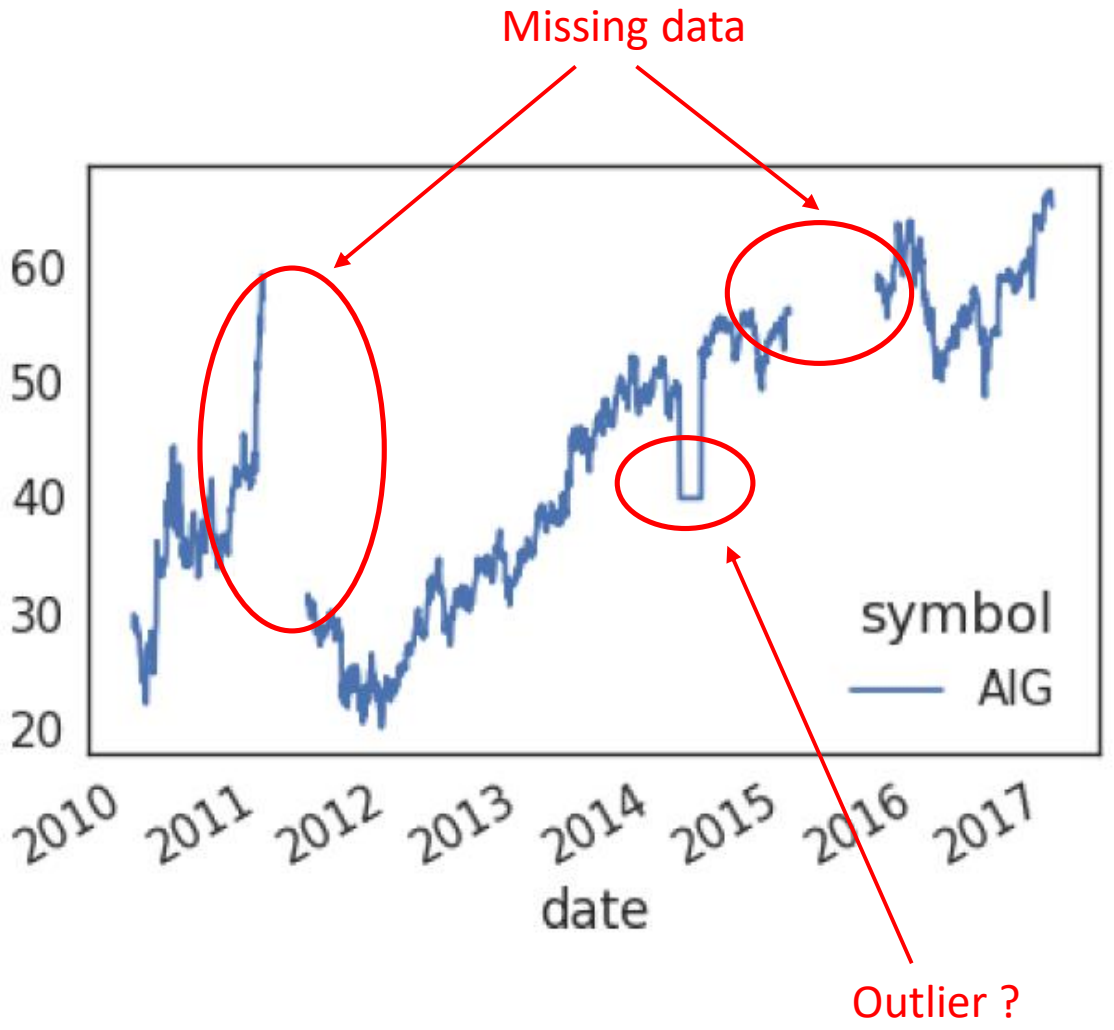
- There can be several steps of preparation and transformation (see chap 2 on ML) :
  1 – basic cleaning and preparation
  2 – Transformation / dimensionality reduction
  3 – Features selection and extraction (next §)

- For a timeserie, dimensionality reduction means reducing the number of data points

First level of cleaning

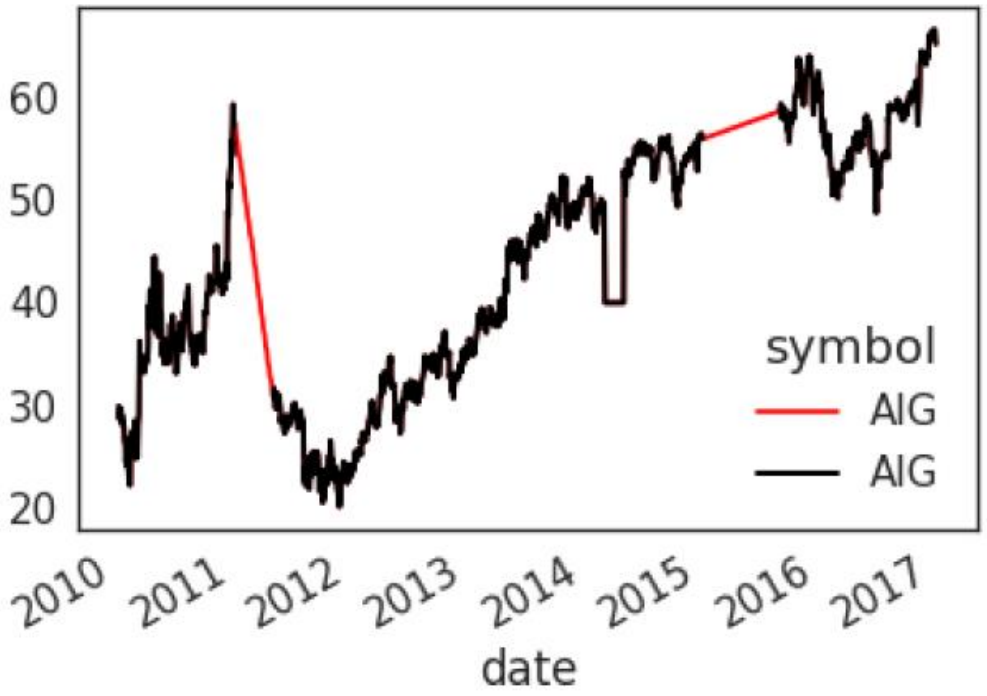
Missing data and outliers

- The TS must be coherent for the future model to be relevant and performing (for prediction)
- One must first avoid missing data and « false » measures (called outliers)
- The challenge is first to detect them and then to replace them



Detecting and replacing missing data and outliers

(linear) Interpolation is the simplest way to replace missing values :



Detecting outliers by thresholds

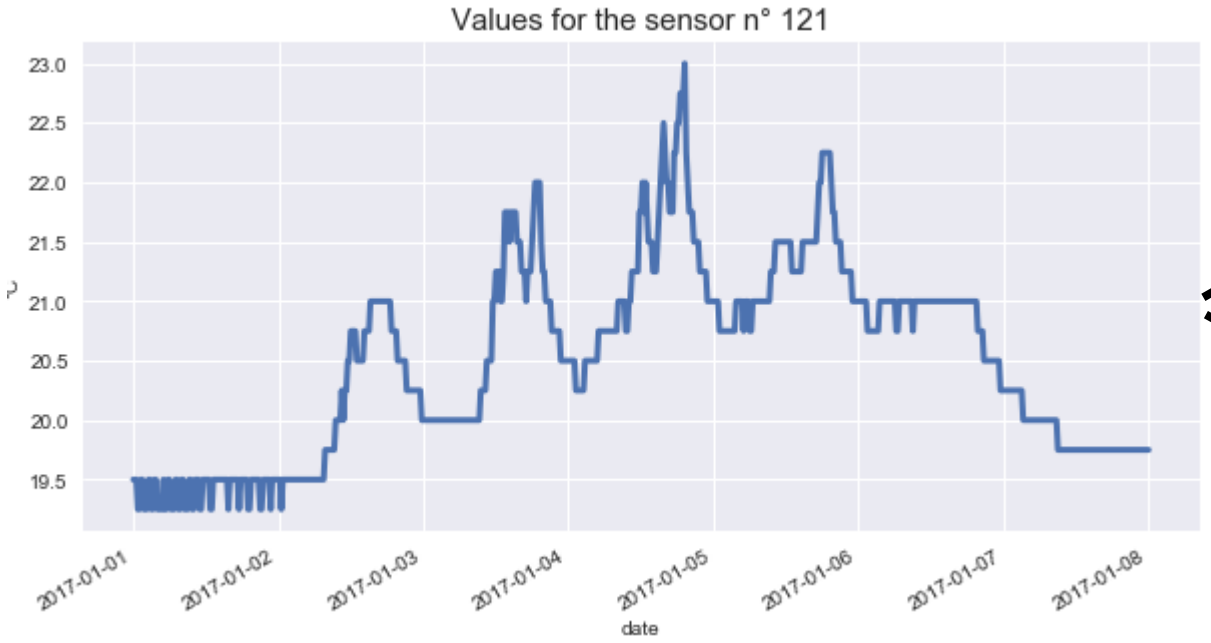
Real outliers can biase a model (overfitting risk)

But detecting them can be difficult and replacing them dangerous (can represent a real anomaly !)

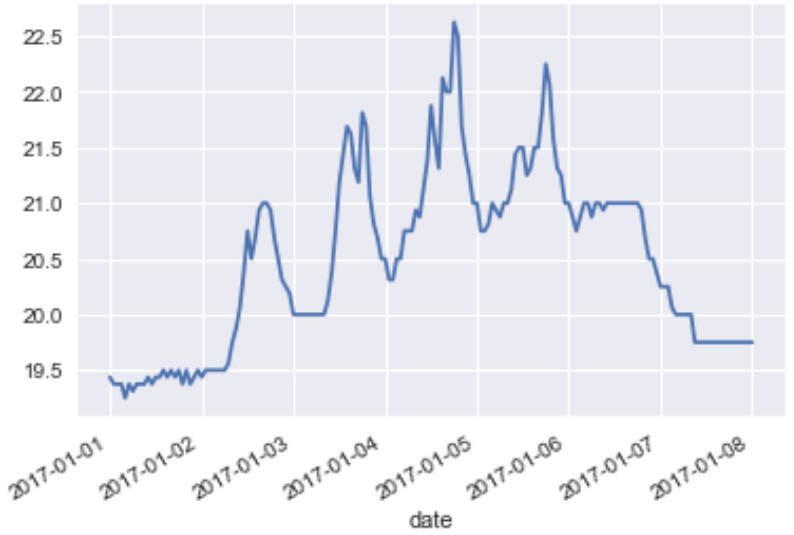
Simplify a TS by « resampling » (smoothing)

Temperature sensor

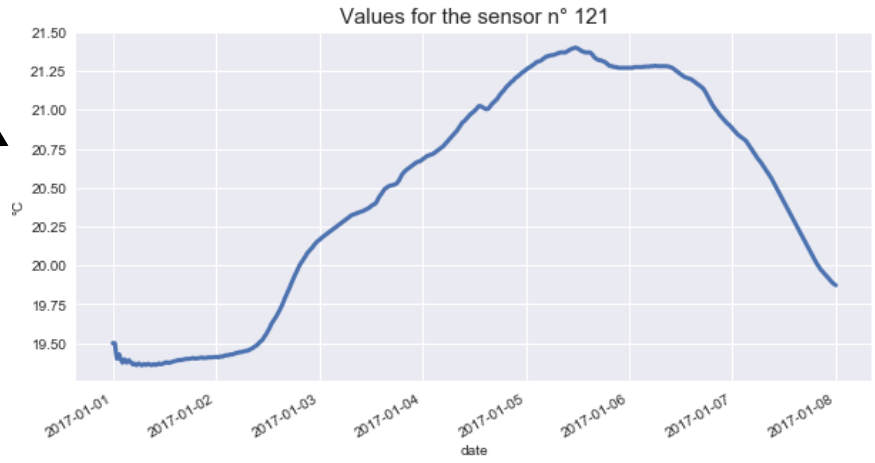
Resampling means changing the time frequency (here downsampling)



Resample by HOUR
24 points instead of 96 by day



Resample by DAY
Only 1 point by day (the mean)



Original TS - Raw data :
8 days – 1 measure each 15 mn : 96 points by day

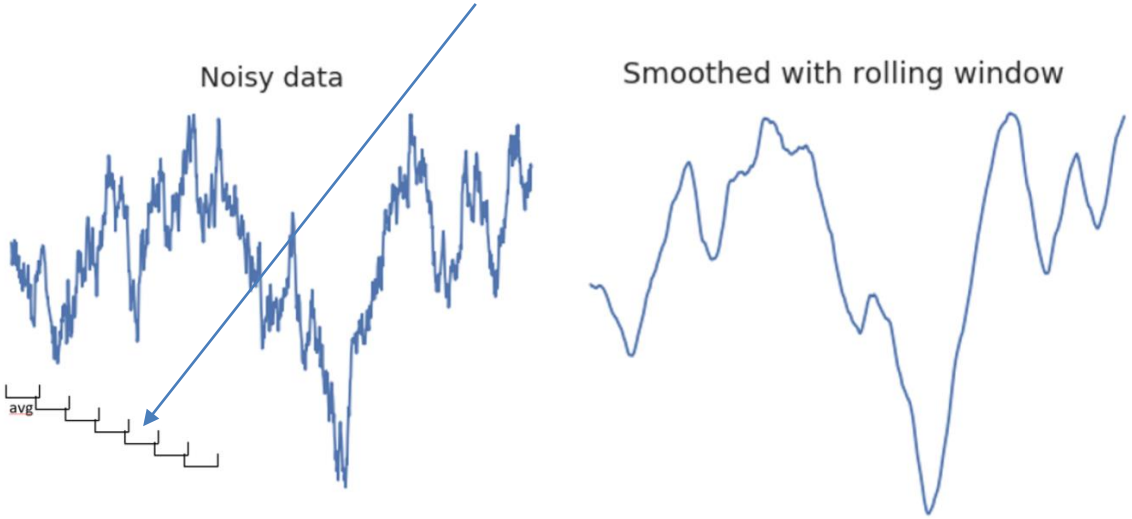
Rem : here the reduction of points (this is a dim. reduction) is calculated by applying the mean function to the window set

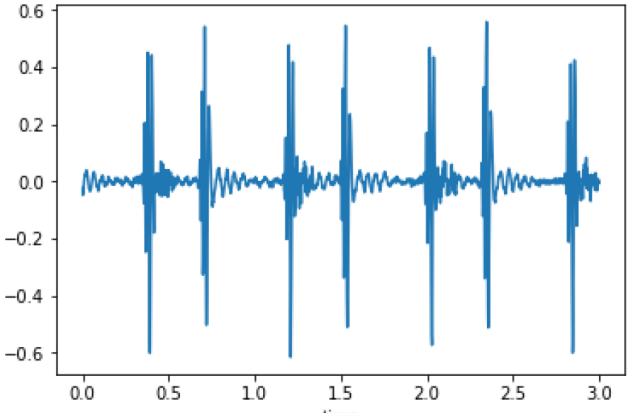
SMOOTHING removes short-term noise and retains the general pattern : the enveloppe

Generalization of the smoothing technique

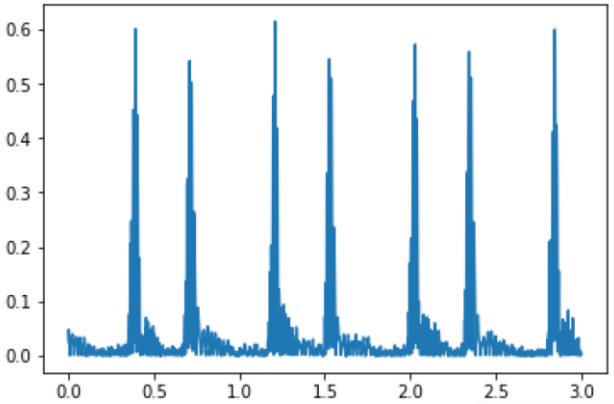
Applying a function to a « rolling window »

- Smoothing consists in taking the mean for n local points on a « rolling window »

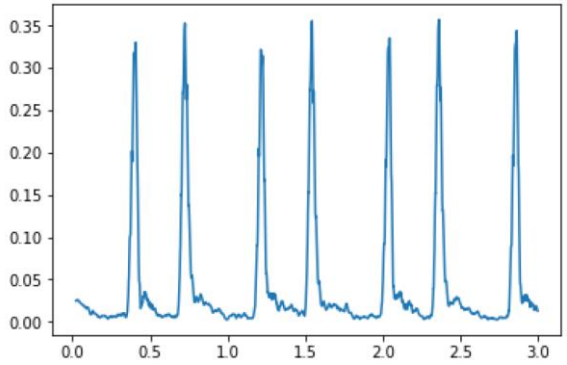




ABS



MEAN



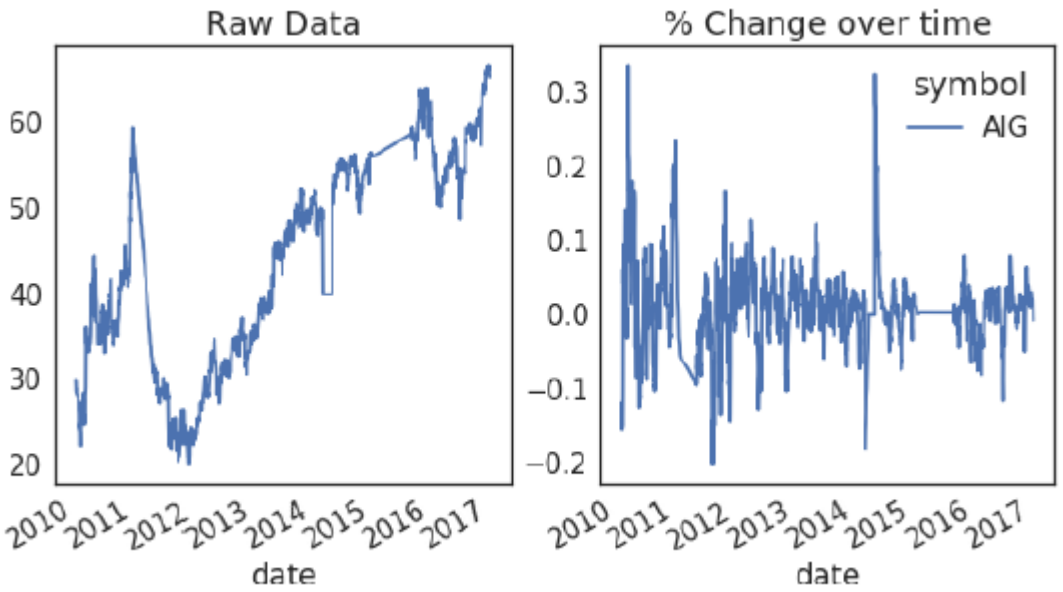
- The same principle can be generalized with any other function to transform the TS
  - The diff, the log, the abs, the max, the std deviation …
  - See also Fourier transform next slides
  - Several fct can be applied successively
  - You have to specify how many points you put in the local window to apply function f

Other transform functions

Percent change

The original data (left) is replaced by the average % change of the n previous points (right)

It permits to standardize the serie (its variance) if the absolute value changes a lot over time (see further standardization)



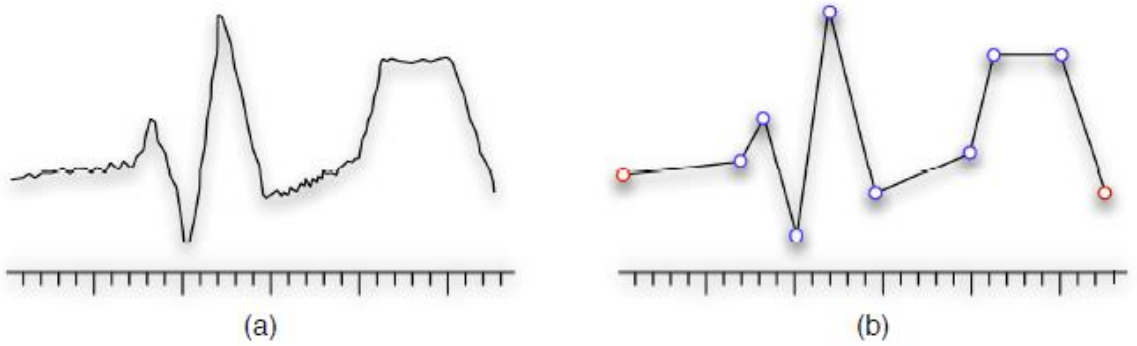
Other tranform functions :

- Diff : y(t) is replaced by y(t) – y(t-1)
- Diff2 : applies diff twice (ex : acceleration)
- Standard deviation
- Fourier transform : see next
- Wavelet transform
- More generally, signal processing techniques can be applied

- Advise : better choose transform functions that keep the variance of the TS and the similarity measure between 2 TS

Dimensionality reduction by segmentation

- A TS can be « segmented » in several non overlapping parts
- The TS is approximated by straight lines, linking selected points
  - Points can be local extrema or « key points » - other points are eliminated
- This can be made manually (visually) or automatically
  - Lots of techniques exist (ex : clustering with a sliding window)
  - Domain knowledge can help you detect the sequence



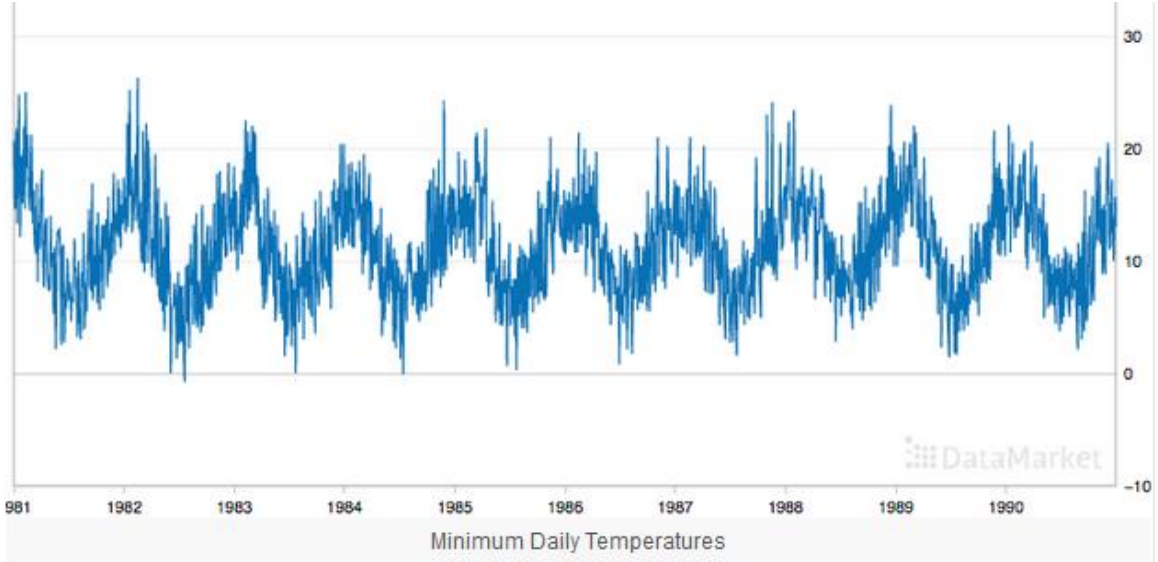
Dimensionaly reduction (nb of points) by segmentation using PLA method (Piecewise Linear Approximation)

Normalizing and Standardizing input data
(in the case of time series)

- A model can be performing (good scoring) if the input data (features) are « homogeneous » : must have a consistent scale and distribution
- When do you have to rescale your data ?
  - When the variance of one feature is very high and/or several variances are very different among feature
  - When the features are continuous (ex time series) and at different scale
  - When you want to use a linear model
- There are two main techniques :
  - ➢ Normalization
  - ➢ Standardization

Normalization

- Normalization is a rescaling of the data from the original range so that all values are within the range of 0 and 1 (if positive values)
- It can be required when
  - your TS have data at very different scales
  - You want to use some models like linear regression, KNN (that uses a distance between TS)
- The basic normalisation is calculated knowing the min and the max
  - y(norm) = (y - min) / (max - min)
  - Other normalization : the log, see next slide
- You can reverse the normalization applied to the output (prediction) to have it at the original scale



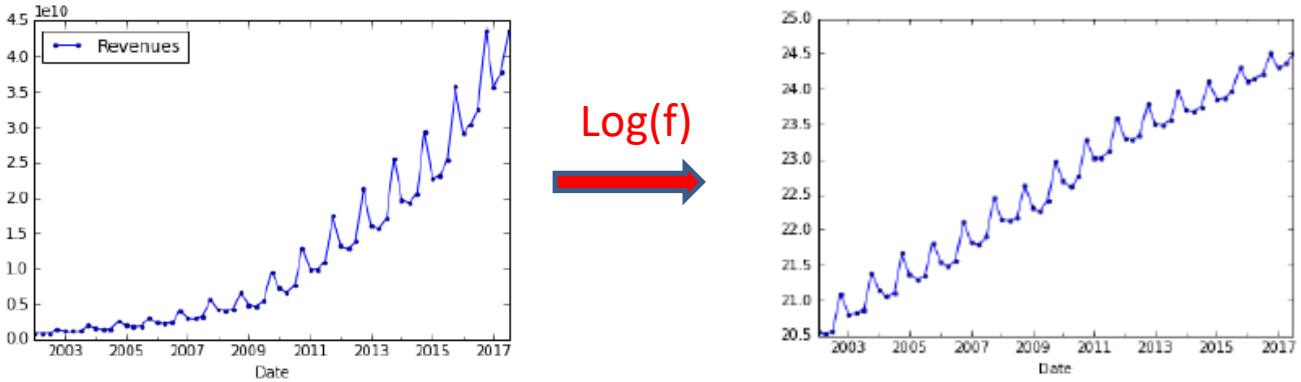

```
1981-01-01 20.7
1981-01-02 17.9
1981-01-03 18.8
1981-01-04 14.6
1981-01-05 15.8
Name: Temp, dtype: float64
Min: 0.000000, Max: 26.300000
[ 0.78707224]
[ 0.68060837]
[ 0.7148289]
[ 0.55513308]
[ 0.60076046]
[ 20.7]
[ 17.9]
[ 18.8]
[ 14.6]
[ 15.8]
```

Example of normalizing
a timeserie of temperatures
The original range [0 – 27] °C
becomes [0 1]

Log normalization

- If you have an exponentiel evolution data (variance very high), ML model won't like it !
- you need to make it linear by applying a log function
- We already saw that with the criteria of the stationarity of timeseries

	col1	col2	log_2
0	1.00	3.0	1.098612
1	1.20	45.5	3.817712
2	0.75	28.0	3.332205
3	1.60	100.0	4.605170



The variance of this feature 2 is too high ~ 1691 ! (too big variations !)

After applying the log, variation is much smaller and variance is near 1 !

Standardization

à revoir

- Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1.
  This can be thought of as subtracting the mean value or centering the data.

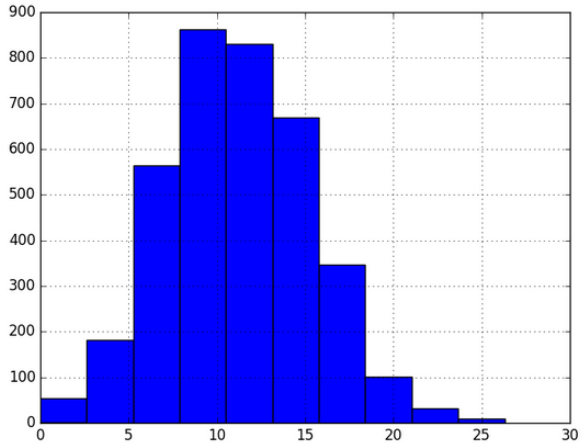
- y(stand) = (y - mean) / standard_deviation

- Standardization assumes that your observations fit a Gaussian distribution (bell curve) with a well behaved mean and standard deviation.

- Can be needed for some models which perform better with Gaussian distribution (SVM, Logistic Regression …)

```
1  Date
2  1981-01-01 20.7
3  1981-01-02 17.9
4  1981-01-03 18.8
5  1981-01-04 14.6
6  1981-01-05 15.8
7  Name: Temp, dtype: float64
8  Mean: 11.177753, StandardDeviation: 4.071279
9  [ 2.33888328]
10 [ 1.65113873]
11 [ 1.87219948]
12 [ 0.84058266]
13 [ 1.13533032]
14 [ 20.7]
15 [ 17.9]
16 [ 18.8]
17 [ 14.6]
18 [ 15.8]
```

Example of standardization with the same set of temperatures

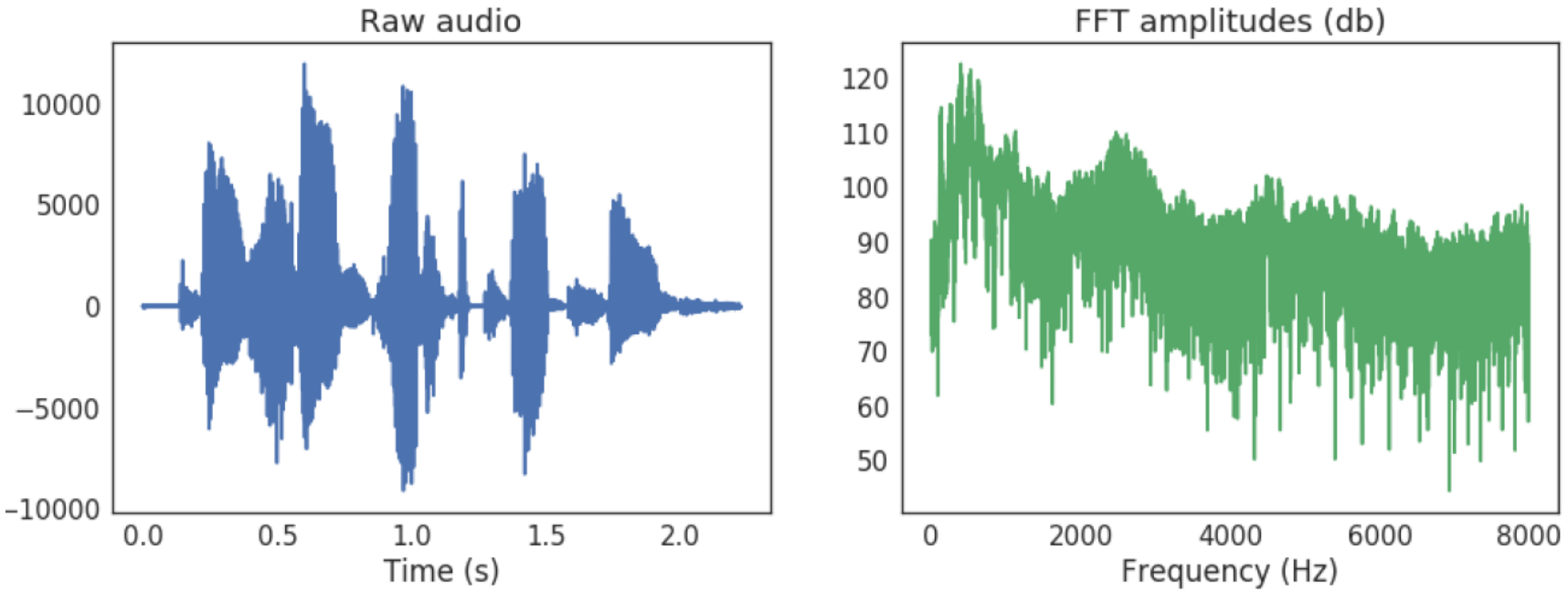


The values of the TS actually follow a Gaussian distribution

Spectral decomposition

Fourier transform (1)

- A Fourier Transform (FFT) describes the « oscillation » of a TS, as combination of fast and slow changes

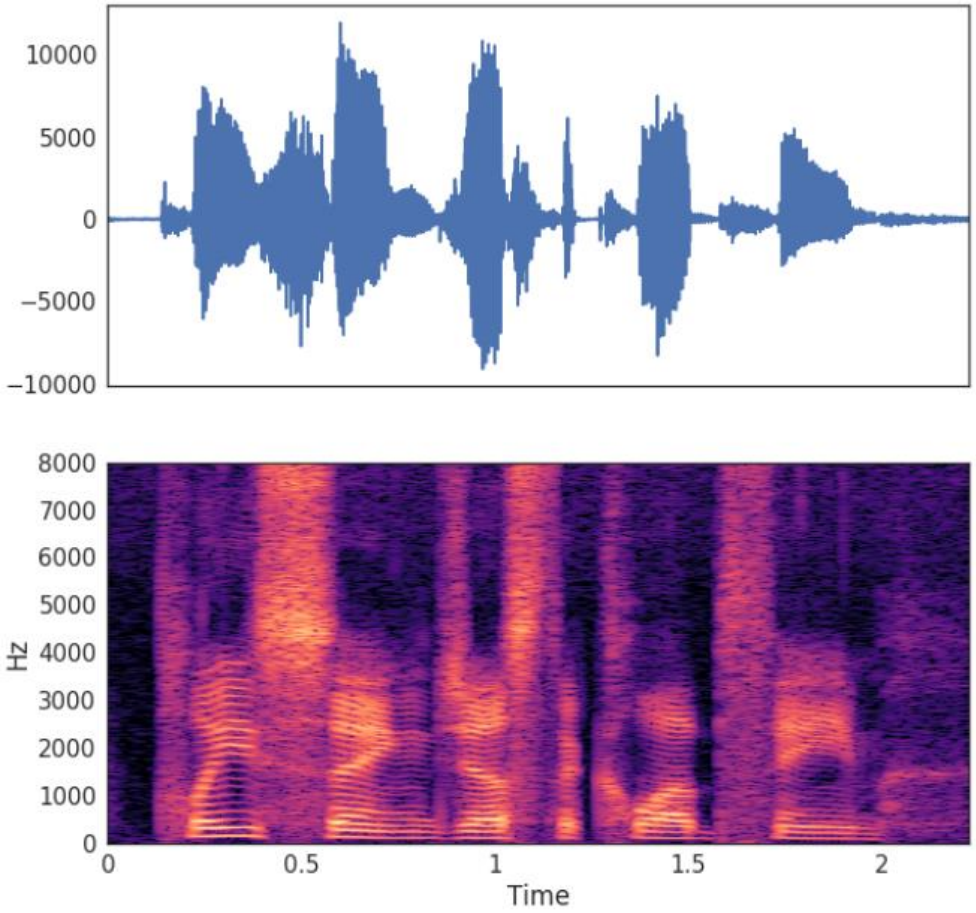


Example of a FFT applied to an audio signal

Spectral decomposition

Short Time Fourier Transform - STFT

- If we apply the rolling window principle with TFT (calculate the local FT at each point of a TS), we obtain a spectrogram
  - « short time Fourier transform » = STFT
- Each TS has a specific spectral pattern
- Interesting features can be extracted (see next section) from a spectrogram and used as input of a ML model
  - Spectral bandwith, spectral centroïds …
  - Then use for a KNN classification for instance
- See also: Wavelet Transform



Sensors and Time Series Analysis

DIMENSIONALITY REDUCTION (2) AND FEATURE EXTRACTION

Why feature extraction is needed for ML ?

- Raw timeseries are generally too complex, too noisy to apply directly a model on it

- One must use techniques to reduce the complexity and/or extract a limited number of features that will « represent » the TS
  - ➢ Without loosing to many interesting information on the TS

- 3 main options (that can be combined or not) :
  1. Keep the TS shape (and the time index) but simplify it (we already saw the « resampling » or « smooting » technique)
  2. Transform the TS into another shape (ex of spectrogram – X is not time)
  3. Replace the TS by discrete features (stats or other) and eliminate the time / X index : case of classification, clustering
  - ➢ One can make a simplication, then a transformation, then a feature extraction

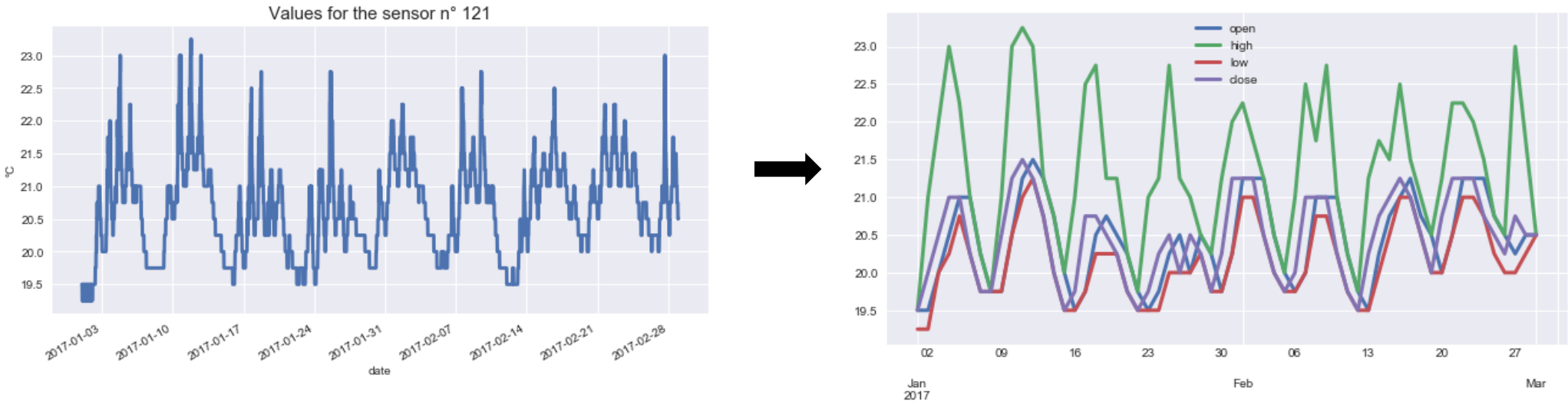
Which features can be extracted from TS ?

- Statistical features :
  - See the list in previous slide (mean, std deviation …) and examples in next slide
  - Rem : time can be eliminated (1 value for the whole TS) or not (rolling window – 1 value for n points)
- Spectral features
  - See further extracting features from STFT (or DWT)
- Datetime (« seasonal ») features
  - Some features depend on time, like the day of the week (see further example of classification), month of the year
- Segmentation and/or Pattern recognition
  - Some techniques consists in identifying recurrent patterns or segments
- Above options can be combined

Example of statistical feature extraction

Dimensionality reduction from TS

Example 1 : an audio sound (lots of points) is replaced by three basic features (here time eliminated)



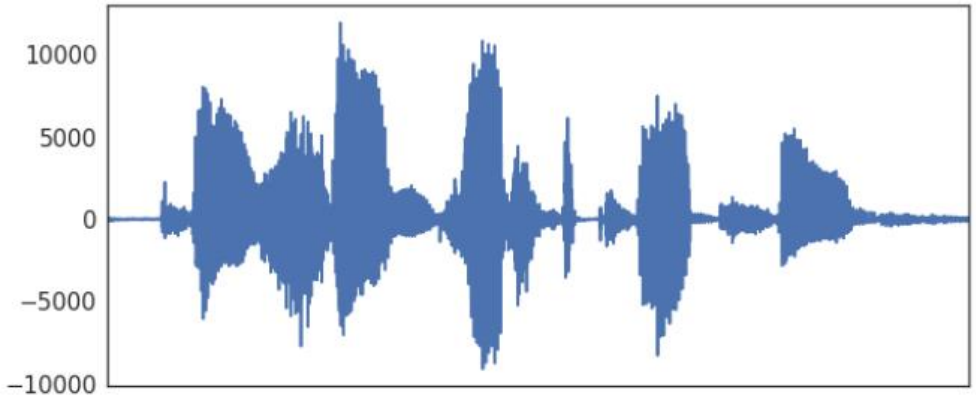
Example 2 : for a temperature sensor, each daily TS (96 points) is replaced by only four values (ohlc)

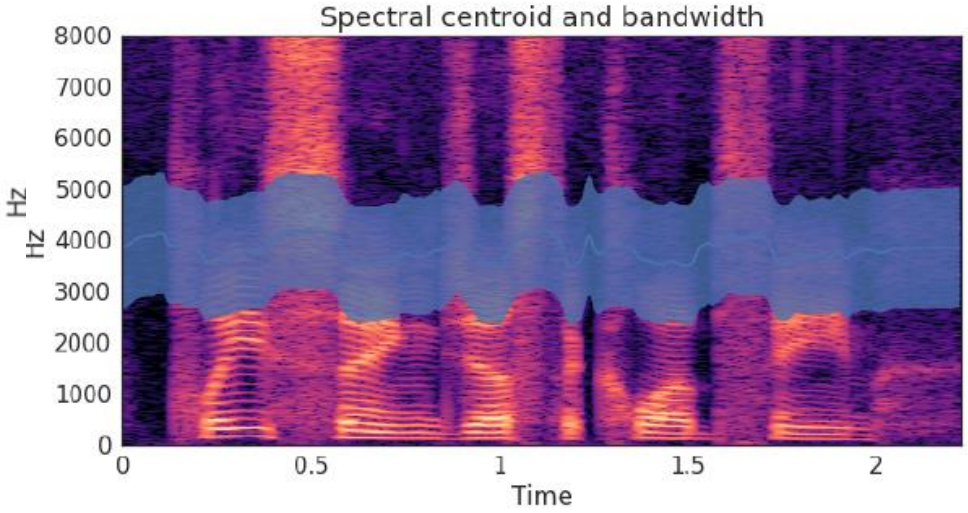
Extracting features from the spectrogram STFT

- We saw in the the previous § that :
  - We can calculate a spectrogram (« short time Fourier transform » = STFT) for a TS
  - Each TS has a specific spectral pattern
  - Interesting features can be extracted from a spectrogram (mean of the spectral bandwith, spectral centroïds …)

- Spectral features can be used alone or in combination with temporal features (min, average …) as input to a ML model (ex : classification)

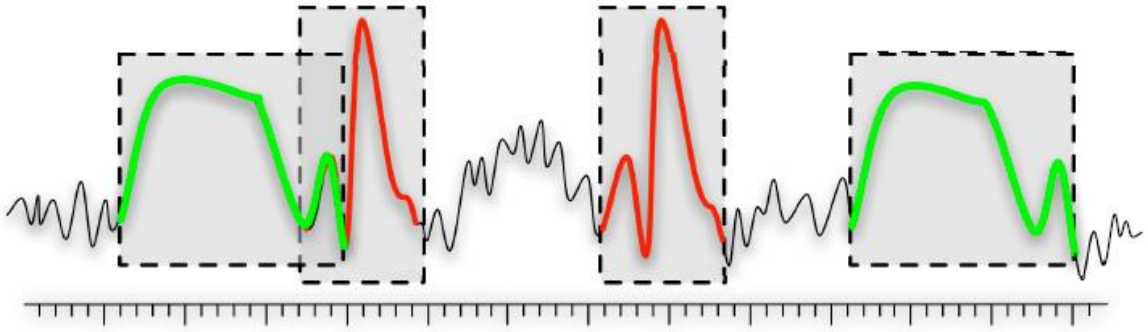
- TFT analysis can be used to extract periodical structure and patterns (see next slide)



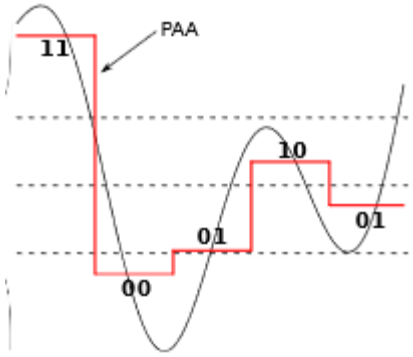


Segment indexing - Pattern recognition

- From the orginal or segmented TS, recurrent patterns can be identified in subsequences
  - Similarity measure with a sliding subsequence serie can be used
- Patterns can be used as features input of a classification model
- It is also called « motif discovery »
- The segmented TS can also be transformed to a symbolic string (or a binary code serie)
  - Each segment can be categorized / clustered
  - It is replaced by a a word or a binary code in the sequence (see the SAX model)



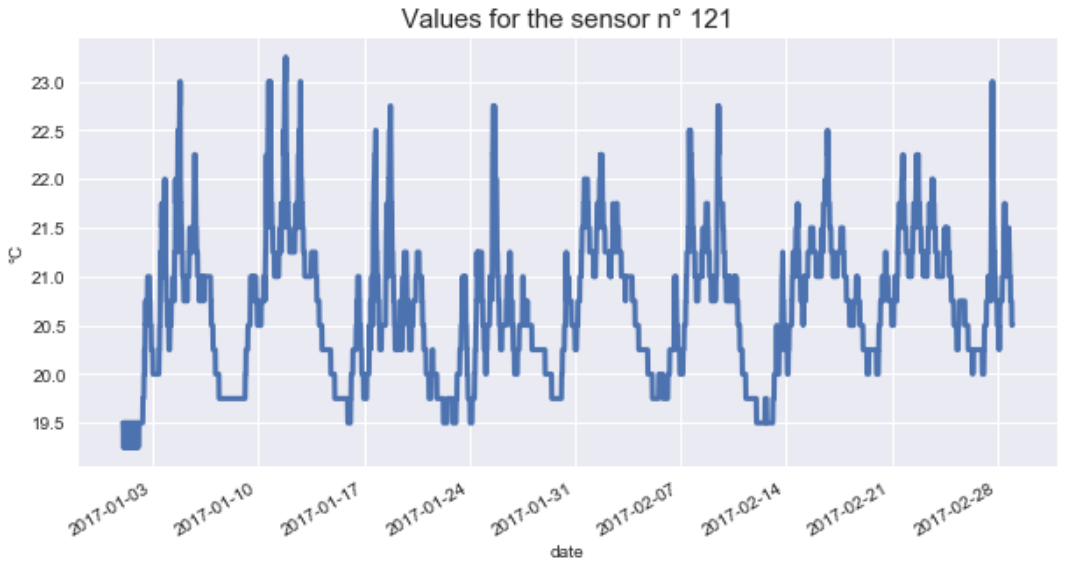
Pattern recognition (or motif discovery) in a TS : extracting recurrent subsequences



Transformation of a TS into a binary sequence (first segmented with a PAA method)

Datetime reduction or extraction

- Periodicity is very important for sensor data, to make the model perform well
- The principle of segmentation or pattern recognition can be applied to time periodic patterns
- Some techniques can detect it (like pattern recognition) but generally you know it :
  - The period of a temperature in a building follows day or/and week periods
- Some models can integrate the « seasonality » of TS (see SARIMAX)



A 2 months temperature sensor in a building
One can check there is a week AND daily periodicity

Sensors and Time Series Analysis

DATA MINING AND MACHINE LEARNING APPLIED TO TIME SERIES

Introduction – adapting the ML process to TS

- The boundaries between pre-processing, tranforming, extracting features and applying a model are not strict in the case of TS

- Preparing and transforming the data before applying the final model is generally heavier, due to the complexity of the raw TS

- The final goal is to discover hidden and high-level information and knowledge inside a TS or a set of TS, in order to make or help prediction, diagnosis, explanation, decision

Main methods and models applied to TS

- Pattern discovery and clustering (discovering categories)
  - Ex of application : Anomaly detection, activity recognition
  - Rem : patterns are often periodic in TS
- Classification (models SVM, HMM, KNN … - see chapter 2 on ML)
  - Transformation, feature extraction and/or clustering must done be before
    - Ex : Fourier transform and then apply KNN on spectral features
  - Ex : activity recognition, failure identification (preventive maintenance)
  - Rem : clustering and classification can apply both on a set of TS, or on a set of sub-sequences (patterns) inside a TS !
- Rule discovery – decision tree
- Regression
  - For time delayed prediction : see next section dedicated to ARIMA models

Applying the ML process to TS : 2 main cases

1 – Time is not important for the ouput of the model

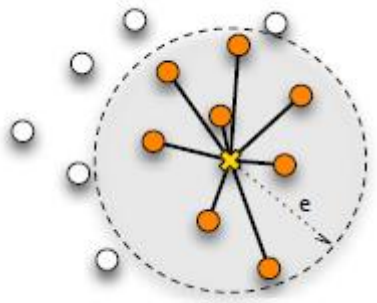
- ex : classification of TS in categories – activity recognition
- first extract features independant of time from TS (see previous section)
- Then apply an ML model like for any other data type (see chapter 2 – Introduction of ML for smart building)

2 – Time is important for the output of the model

- Ex : predicting the future data depending of past data
- You can first apply normalization/scaling and/or dimensionality reduction to simplify the TS (but keep the time index)
- See the section on AR and ARIMA models for prediction

Illustration with the case of KNN classification

- KNN (k nearest neigbours) runs with a distance on samples/points
  - 2 input points (or TS) of the sample belong to the same output category (Y) if they are closed (distance < R)

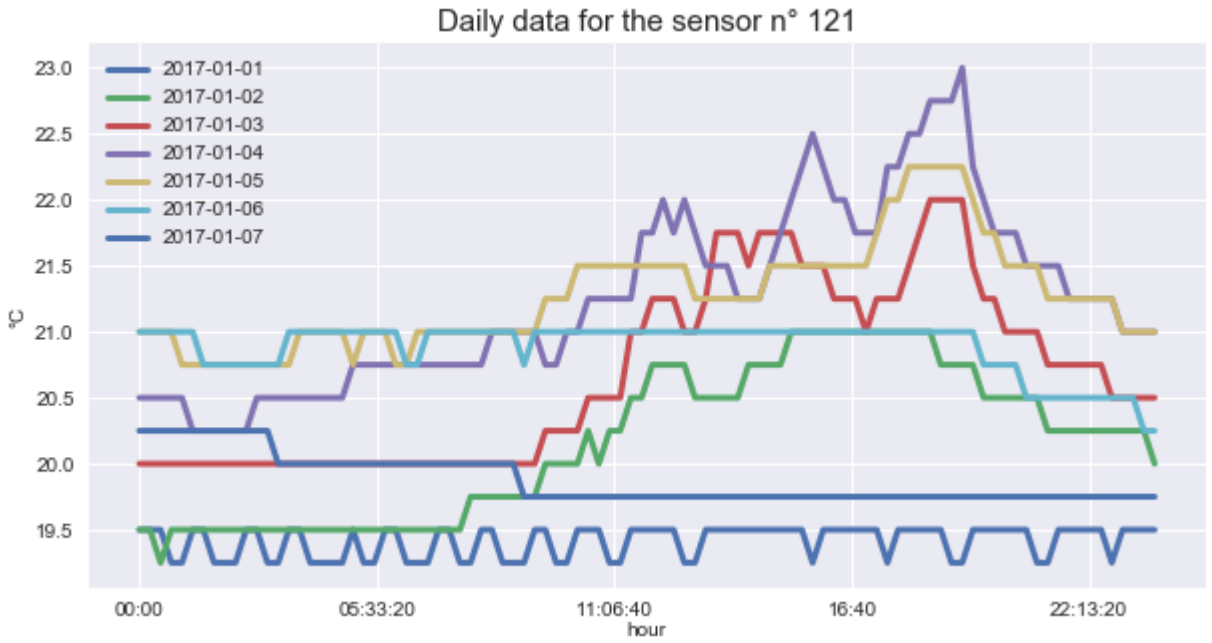


- 2 options (with or without the time)
  1. One apply a similarity measure between original TS as a distance (distance y2(t) – y1(t))
  2. One apply a distance between extracted non temporal features (statistics, spectral STFT, …)

Here a point can either represent an original TS with Y=f(t) OR a vector of extracted features (ex : mean, std, max, min)

Example of classification (in a building) after stat features extraction (1)

The problem : classify daily TS with the corresponding day of the week, after 4 features extraction



Extracting daily TS from a weely TS

feature extraction : each daily TS is replaced by four values « ohlc »

	open	high	low	close
2017-01-01	19.50	19.50	19.25	19.50
2017-01-02	19.50	21.00	19.25	20.00
2017-01-03	20.00	22.00	20.00	20.50
2017-01-04	20.50	23.00	20.25	21.00
2017-01-05	21.00	22.25	20.75	21.00
2017-01-06	21.00	21.00	20.25	20.25
2017-01-07	20.25	20.25	19.75	19.75
2017-01-08	19.75	19.75	19.75	19.75

X input features (4)

```
In [12]: print(y)
2017-01-01       Sunday
2017-01-02       Monday
2017-01-03      Tuesday
2017-01-04    Wednesday
2017-01-05     Thursday
2017-01-06       Friday
2017-01-07     Saturday
2017-01-08       Sunday
2017-01-09       Monday
2017-01-10      Tuesday
2017-01-11    Wednesday
2017-01-12     Thursday
```

Y output labels (1)

Each line of X represents a day of temperature :

- Initially a TS with 96 points : replaced by 4 values
- This is both dimensionality reduction and feature extraction
- The challenge : can a classification model (ex : KNN) predict the day of the week (Y) from the 4 values X ?

Example of classification after feature extraction (2)

- Main steps of the ML process
  1. Daily TS are extracted from a single long TS
  2. OHLC features are extracted, and Y too (name of the day) - see previous slide
  3. The 2 months set of daily TS is split into train (80%) and test (20%)
  4. The classification model is chosen (KNN, SVC …)
  5. The model is fitted with the train data
  6. Model prediction is made on the test dataset
  7. Predictions can be compared to actual Y data tes (see right) and the score calculated
  8. Cross validation can be used to have a better accurate score (from several train/test datasets)
  9. Come back to previous steps to improve the model !!! Scaled data give better scores (50%)

```
In [21]: print(y_compare)
                test predict
2017-02-26     Sunday  Sunday
2017-02-20     Monday  Sunday
2017-01-01     Sunday  Sunday
2017-01-20     Friday  Sunday
2017-01-14   Saturday  Sunday
2017-01-17    Tuesday  Monday
2017-02-28    Tuesday  Sunday
2017-02-09   Thursday  Monday
2017-01-25  Wednesday  Monday
2017-02-16   Thursday  Sunday
```

```
In [24]: print(score)
0.16666666666666666
```

Comparing prediction with actual Y data test and evaluating the score (only 16,6 % !)

```
In [27]: print("Average 5-fold CV Score: {}".form
Average 5-fold CV Score: 0.19142857142857142
```

The average score calculated with 5 train/fit tests thanks to cross-validation : 19,14 %

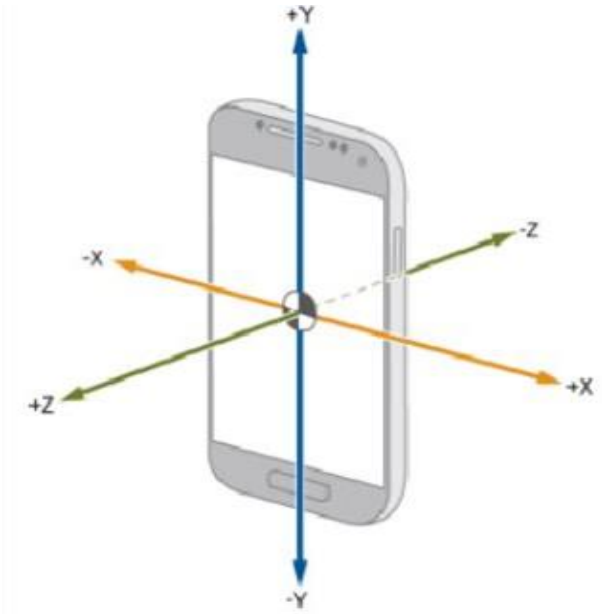
Provisional conclusions :

- Less than 20% is a bad score, not reliable
- 4 features OHLC are not representative of daily TS to predict the day name
- One must try to improve the process : change the features, the model …

Other classification example
Real time man fall detection for security (1)

The challenge

- A smartphone app send the movement of a person (can be at home or outside) thanks to the 3 axis acceleration data
- For security reason, we want to detect accidental fall in real time
- An alert is send if a fall is suspected
- The user can confirm or not
- Difficulty : a rapid movement is not always a failure ! The classifier must distinguish both with 99% of success !

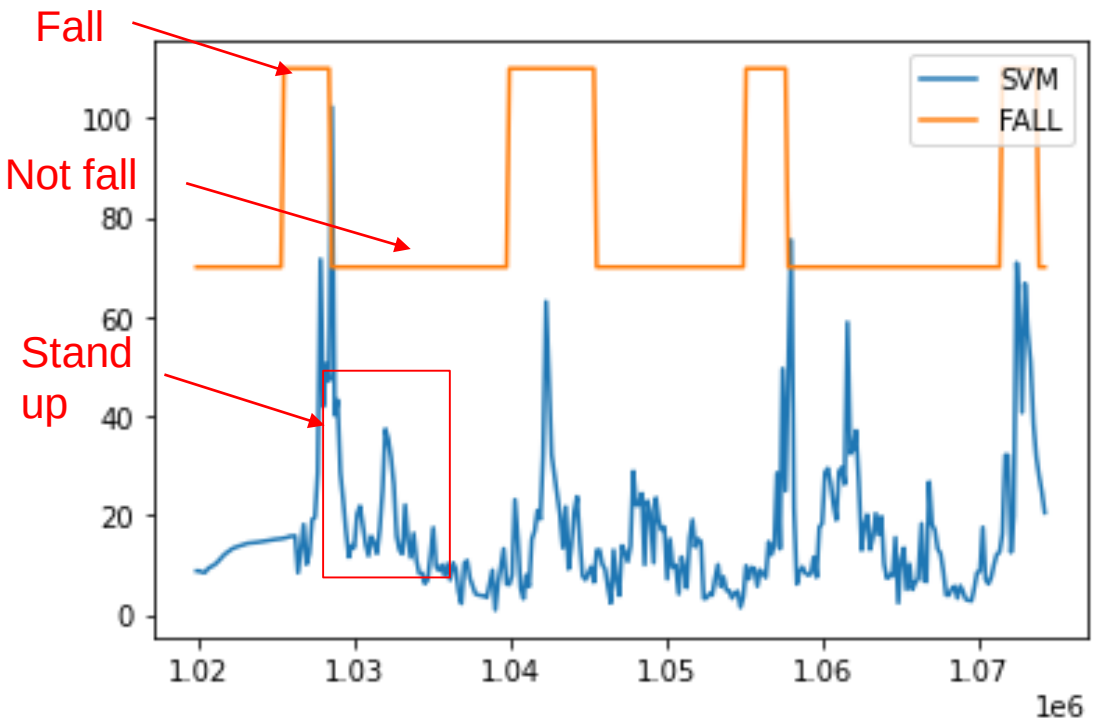


Real time man failure detection – Security (2)

Signal processing and sequence classification

- The data of the time serie is the signal magnitude vector $svm^* = \sqrt{A_x^2 + A_y^2 + A_z^2}$
- The raw signal is transformed in a fall / not fall signal (sliding window of 80 ms in real time)
- ML classification models are trained with real or not real falls
- Different kinds of movements and falls are referenced and classified (standing, stumbling …)

Relation between svm* and fall :



Real time man fall detection for Security (3)

Comparison of ML classification models

	accuracy	F1 score
Logistic Regression	0.800309	0.790195
Naïve Bayes	0.719657	0.640257
K nearesr neighbours	0.801322	0.802311
Binary tree	0.812908	0.818572
Random Forest	0.810124	0.812381
SVM	0.816029	0.820062

- Confusion matrix is used to extract best relevant scores
- Provisional conclusions :
  - Best score around 80% to detect falls
  - Good but not sufficient for vital security
- How to improve the process :
  - Take into account other data : voice, gyroscope ..
  - Try other models : deep learning, Arima ?

Correlation and regression for TS

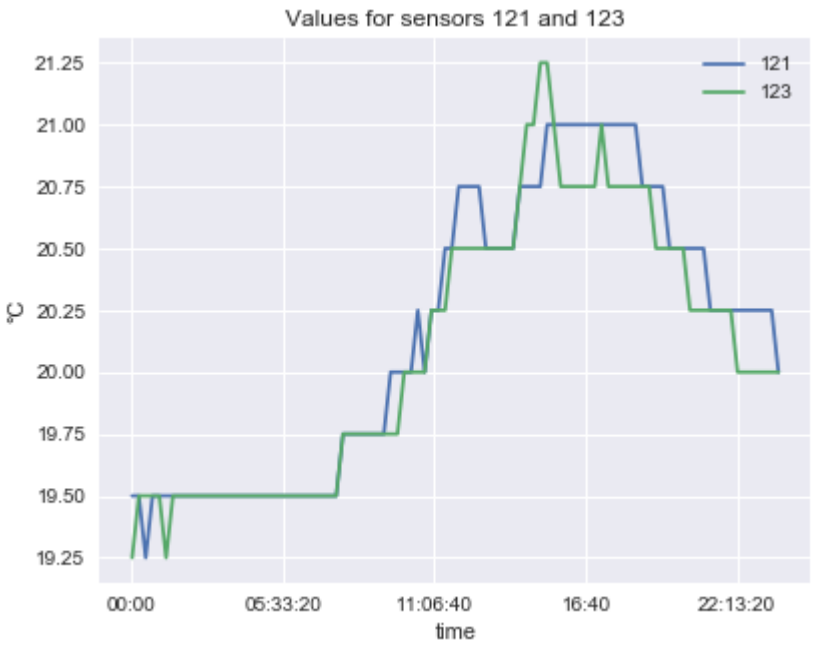
Autocorrelation and time delayed prediction

- The correlation (in [0 1]) between 2 TS measures the level of relation between the 2 curves (at all timestamps)
- You can estimate it by visualising or by calculating (see next slide)
- If the correlation is good (near 1), you can envisage to fit a regression model (next slide)
- If not, you can test it on some transformations on the TS (diff, log …) as seen before
- Warning : correlation can vary over time ! (2 TS can be correlated at one period and not at another)
- AutoRegression (AR) of a TS is the modeling of the relation with itself in time (y(t) with y(t-i))
- AR is the base for making prediction over time (in the future, from past data – also called time delayed prediction)
- It is like calculating a correlation and a regression between the TS (at t) and a lagged version of if (TS at t-1, t-2, …., t-n)
- See the different options of AR models in the next § on ARIMA models

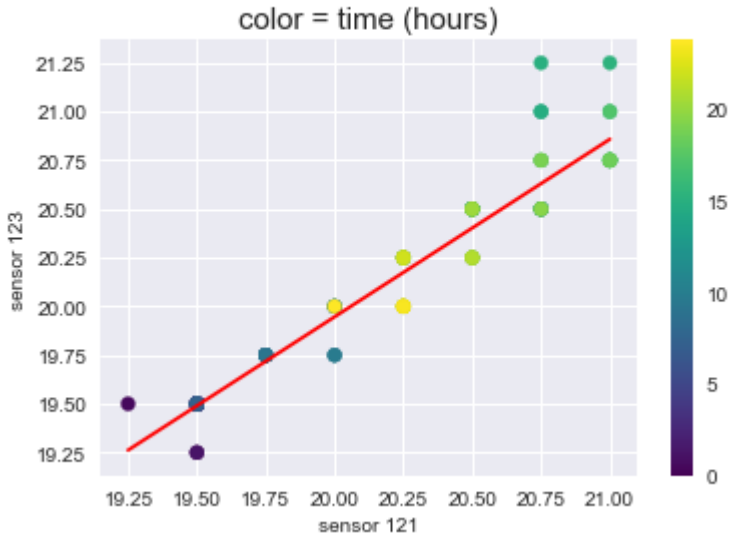
Simple linear regression (OLS) between 2 TS (time eliminated)

Simple linear regression: $y_t = \alpha + \beta x_t + \epsilon_t$

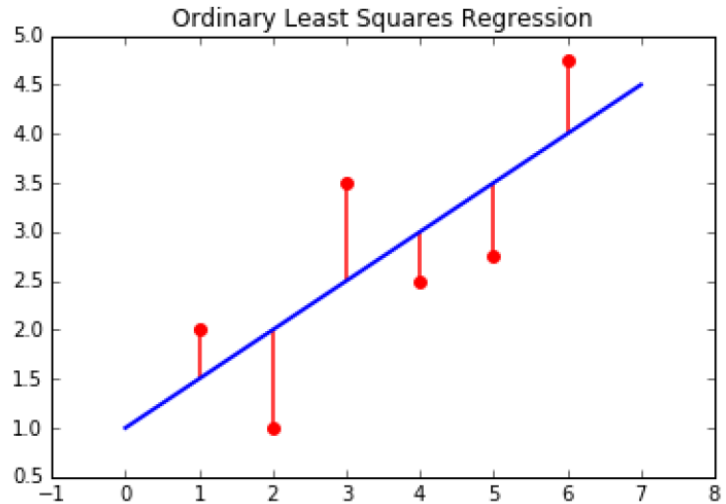
Correlation = 0,88 : quite good : regression is possible !




```
Model slope (coeff a) :  0.9120281577338925
Model intercept (coeff b)  :  1.7057032056821306
```



- Ordinary Least Squares (OLS)

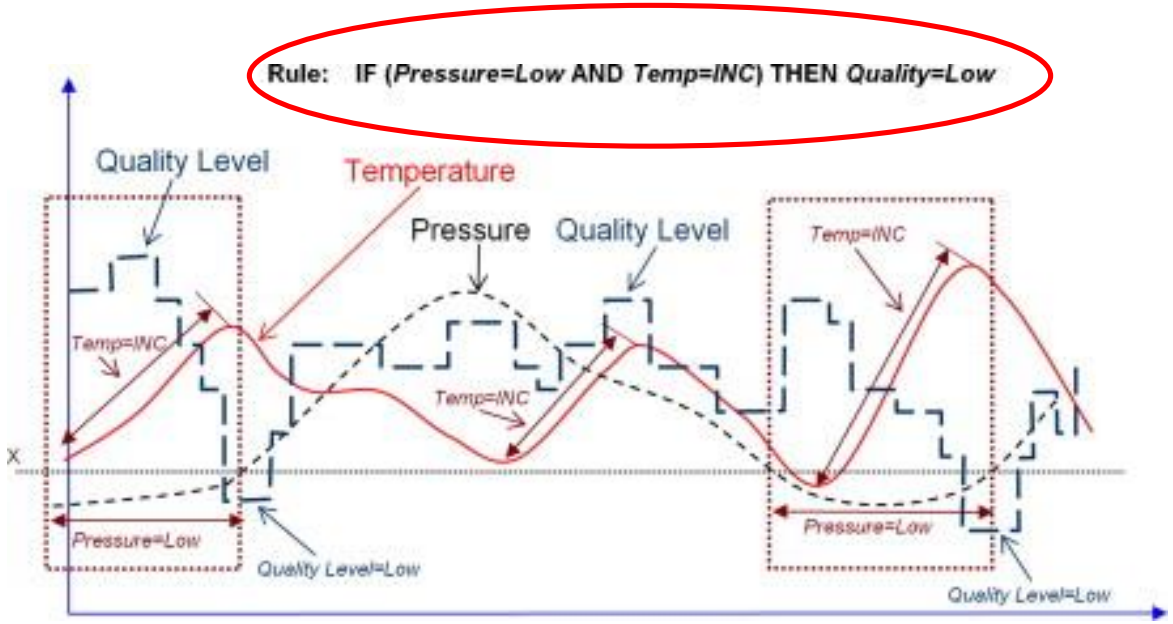


Scoring : $[\mathrm{corr}(x,y)]^2 = R^2$ (or R-squared)

In this case, the OLS regression (minimize the distance) « compare » the y at fixed time t and gives y2 = f(y1),
so the final result « eliminate » the time componant (not so interesting)
See further the Auto Regression models (AR) that can predict future y(t) from past time y(t-i)

Rule discovery / association rules

- The goal is to find hidden rules inside a TS or between TS
- Like for classification, feature extraction must be led before
  - Segmentation/clustering for pattern recognition …
  - Then find the rules between recurrent patterns
  - Ex : pattern A → pattern B after 20 lags
- Main methods for rule discovery :
  - Association rule mining
    - Ex : find some links/association between different sensors
  - Decision trees / random forest

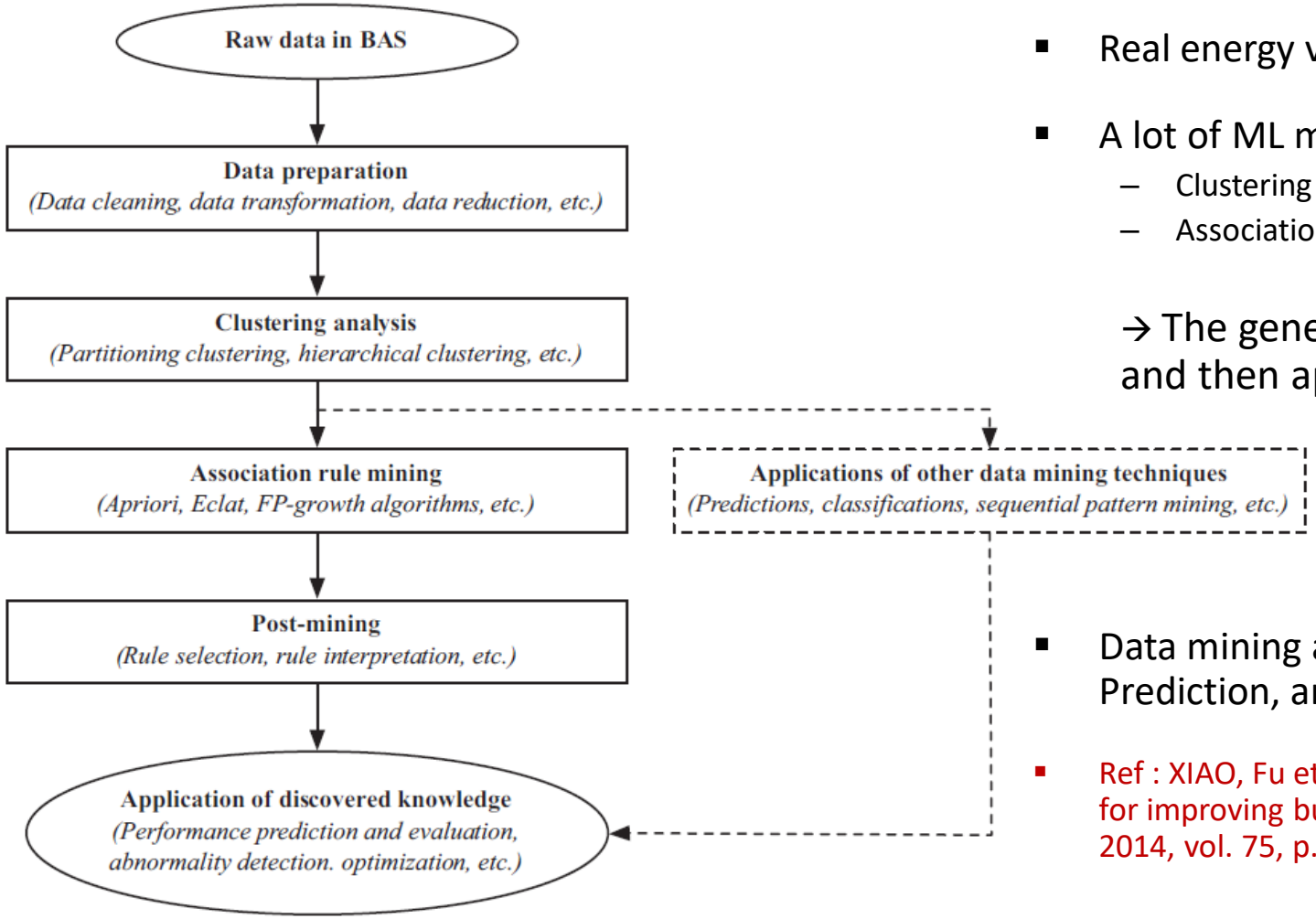


Finding an « association rule » between temperature, pressure and quality level (of a process) from timeseries (here for detecting failure in an industrial process)

Ref : Francisco Javier Martínez-de-Pisón, Andrés Sanz, Eduardo Martínez-de-Pisón, Emilio Jiménez, Dante Conti, Mining association rules from time series to explain failures in a hot-dip galvanizing steel line, Computers & Industrial Engineering, Volume 63, Issue 1, 2012, Pages 22-36, ISSN 0360-8352,

Example of clustering applied to SB

Energy optimisation with association rules and other models



- Real energy values in the highest Hong Kong building
- A lot of ML models are applied and compared
  - Clustering for pattern recognition in TS
  - Association rule mining : interpretation with domain knowledge

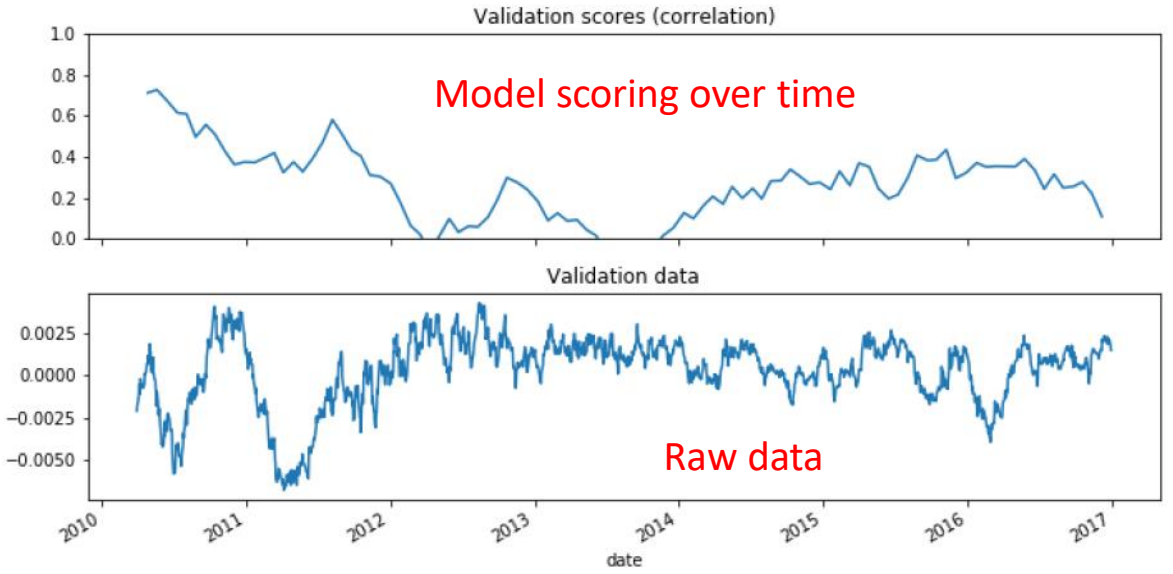
→ The general goal is to fine hidden knowledge in TS and then apply this knowledge for decision support

- Data mining and machine learning techniques applied For Prediction, anomaly detection, optimization …
- Ref : XIAO, Fu et FAN, Cheng. Data mining in building automation system for improving building operational performance. *Energy and buildings*, 2014, vol. 75, p. 109-118.

Model stability (over time)

Cross validation and scoring over time

- If the TS is not stationary, the model won't be stable over time
  - That means that its parameters vary over time (the contrary of stability is variability)
- One can use cross validation to test the stability of the model parameters along the different split of the data
- One can also visualize model performance / scoring over time as a time serie

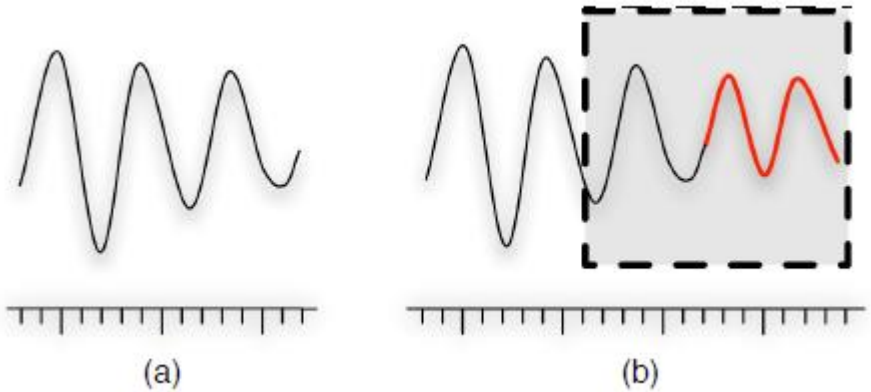


Sensors and Time Series Analysis

TIME DELAYED PREDICTION WITH ARIMA MODELS

The principles of time prediction for TS

- A timeserie is considered as « smooth » (continuous) : if not a « white noise », we can expect to (model it and) predict a value at t from the previous values at t-1, t-2 … , t-n
- The autocorrelation is an indicator to apply a time prediction model to a TS
- We can predict shor term value but long term prediction is far more difficult
- The most common used model are statistics models like the AR / ARIMA family seen in this §

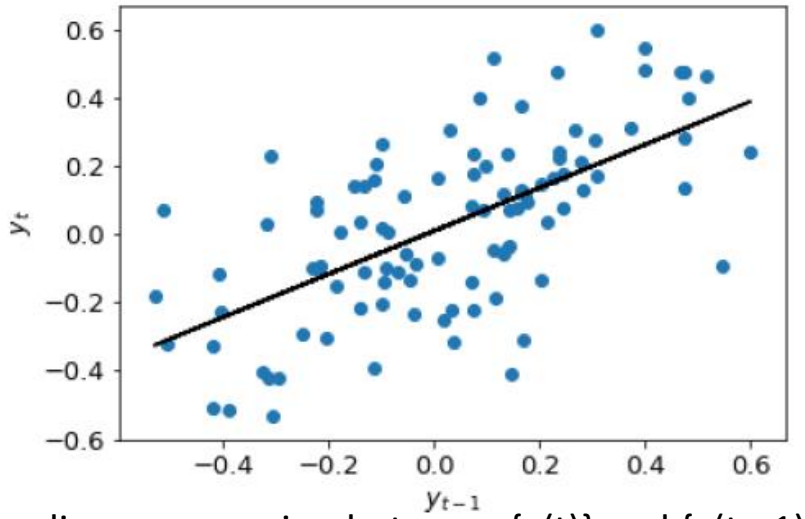


Short term prediction (red) using a model and the most recent values

- Some models can take into account external (exogeneous) data outside the TS
  - Ex : sensor 1 (temp) used itself + sensor2 (humidity)

Autoregressive models (AR)

- Aims at approximating y(t) linearly from y(t-1) and possibly y(t-2) … y(t-p)
- AR(1) : $y_t = a_1 y_{t-1} + \epsilon_t$ (AR order 1)
  - a1 is the parameter (in [-1, 1] if TS stationary)
  - ϵ_t is the uncertainty (on the prediction)
- AR(2) : $y_t = a_1 y_{t-1} + a_2 y_{t-2} + \epsilon_t$
- AR(p) : $y_t = a_1 y_{t-1} + a_2 y_{t-2} + \ldots + a_p y_{t-p} + \epsilon_t$
  - p is the order of AR model (a challenge is to find the best optimized « p » parameter)



AR(1) is a linear regression between {y(t)} and {y(t - 1)}

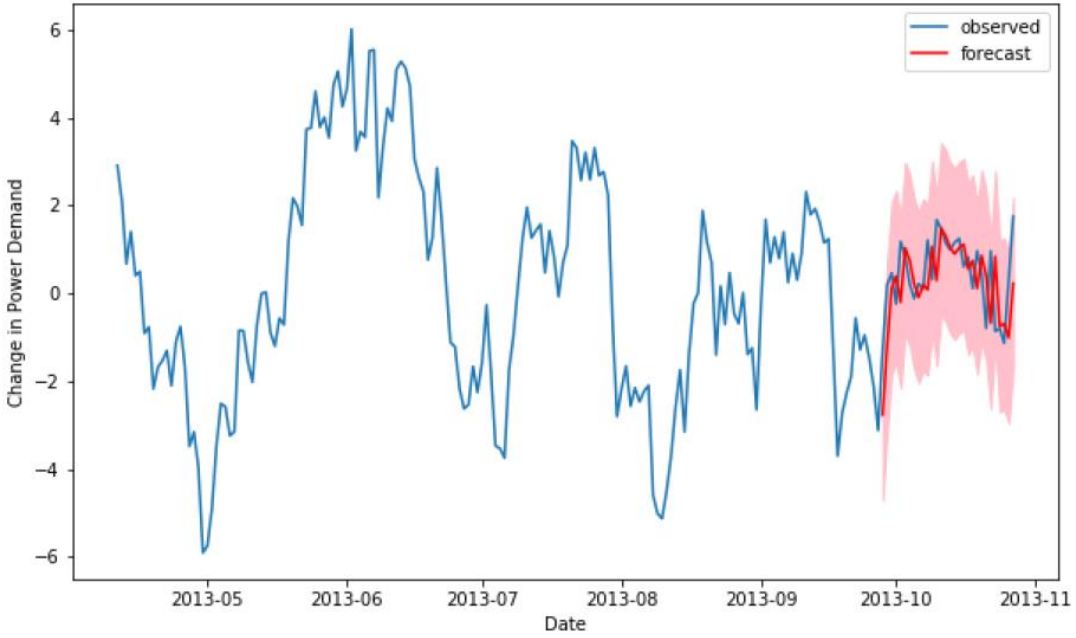
Moving Average (MA) and ARMA models

- MA : in this case, y(t) depends on a small linear variation from y(t-1) but not directly from y(t-1)
  - MA(1) : $y_t = m_1 \epsilon_{t-1} + \epsilon_t$
  - MA(q) : $y_t = m_1 \epsilon_{t-1} + m_2 \epsilon_{t-2} + \ldots + m_q \epsilon_{t-q} + \epsilon_t$
  - q is the order of the MA model
- ARMA models are a combination of AR and MA linear regressions
  - ARMA(1,1) : $y_t = a_1 y_{t-1} + m_1 \epsilon_{t-1} + \epsilon_t$
  - ARMA(p , q) : p is the order of the AR part, q is the order of the MA part
    - One challenge is to optimized and find the best order for p and q
    - See different techniques further

Predicting with an ARMA model

Case 1 : « one step ahead » prediction

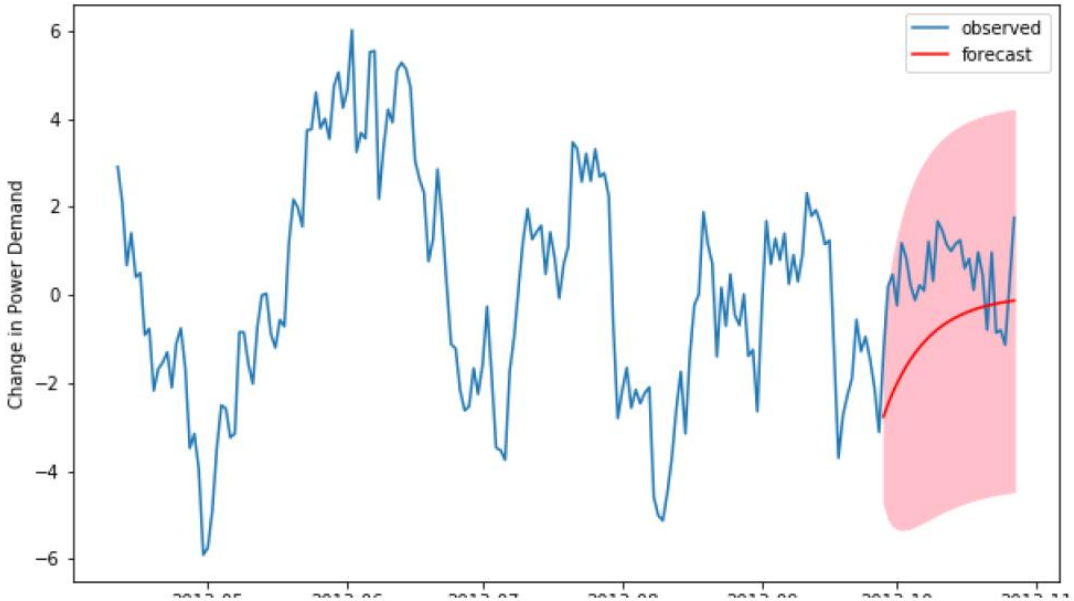
- y(t) is predicted with previous known value(s) – at each timestep t (very short term)



Comparison between forcasted and real data with a t lag 1
(in pink is the uncertainty zone – [min, max] possible

Case 2 : prediction on a longer future interval (dynamic prediction)

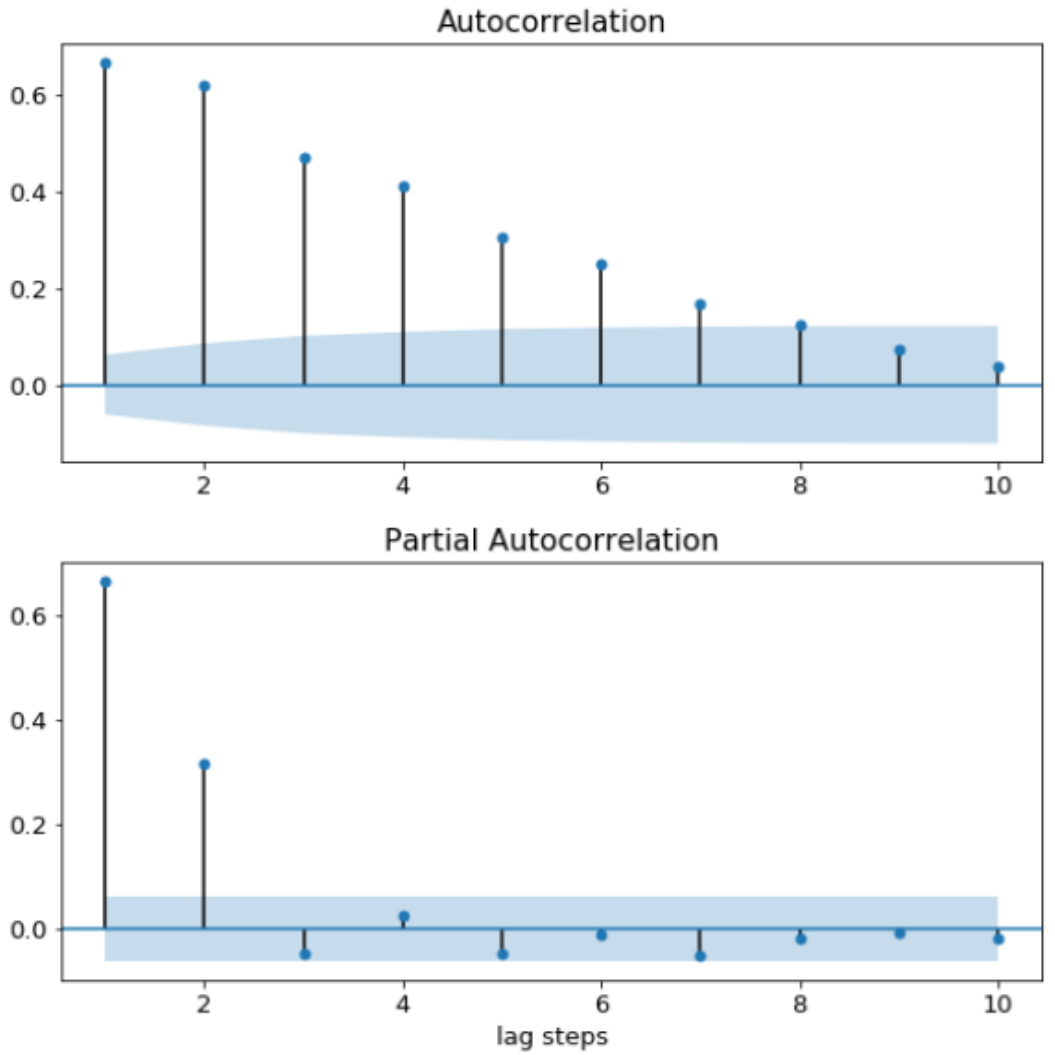
- Y is predicted at t, t+1, …, t+n but with much more uncertainty (always with t-1, ….., t-p)



Comparison between dynamic prediction and real data
In a future period : we notice that the model only predicts the Trend

The choice of p and q values with ACF and PACF

- How to chose p and q order when applying ARMA(p, q) ?
  - Rem : p or q can be = 0
- The AutoCorrelation Function (ACF – see upon) and the partial ACP (PACF) diagram shapes can help
- Example (on the right)
  - AR(p) : if ACF tails off AND PACF cuts after lag 2 → p=2
  - All the rules are not developped here (see complete chapter on ARIMA models)



ARMAX : adding external information

- With ARMA, you can predict the future data from a TS with past data of the same TS (ex: sensor temperature S1)

- Sometimes, you would like to take into account additional data outside de TS (ex : another sensor S2, temp, humidity …)
  - ➢ Rem : You first have to study the correlation with other sensors/TS before selecting X

- ARMAX : means eXogenous ARM
  - = ARMA + linear regression on external data
  - ARMAX(1,1): $y_t = x_1 z_t + a_1 y_{t-1} + m_1 \epsilon_{t-1} + \epsilon_t$
  - Rem : you can do and ARIMAX

- Generalization : making AR models on several TS is VAR models : vector autogression models (not developped here)

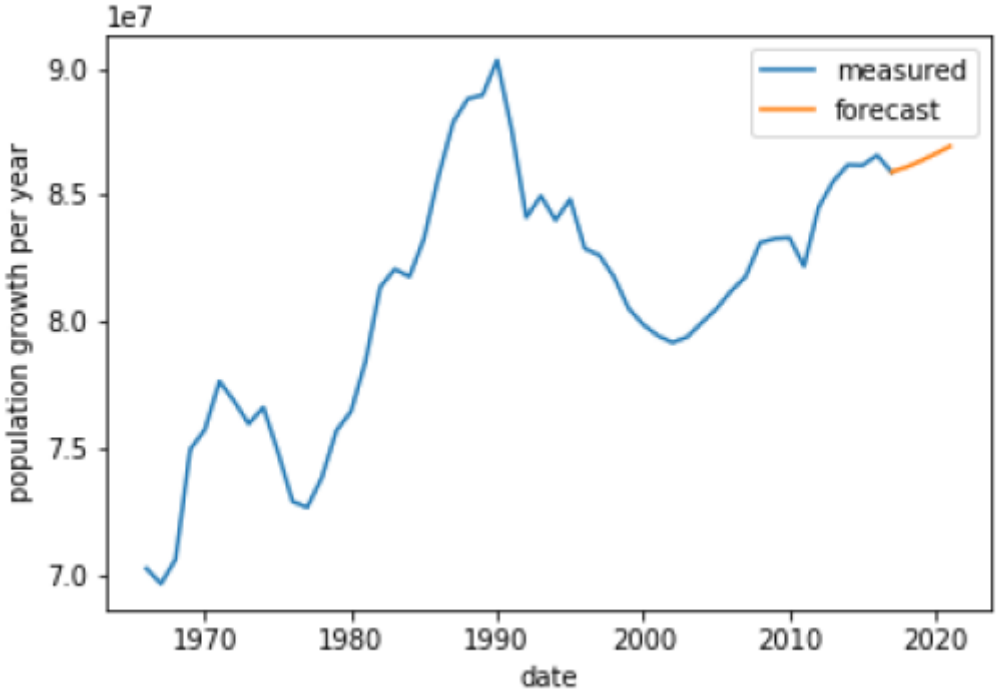
ARIMA model

AutogRessive Integrated Moving Average

- Remember that models don't work (predict) well with non stationary TS (model not stable over time)
- Most TS from sensors are not stationary – So what ?
- But the diff (y(t) – y(t-1)) is more likely to be stationary (check with the AD-Fuller test)
- In this case, you can apply an ARMA to the diff (prediction), but then you have to « reconstruct » the original TS (complicated !)
- ARIMA do the process for your : it is an ARMA models which works with the original TS even it is not stationary but if the diff is stationary (or the diff of the diff !)

ARIMA application and example

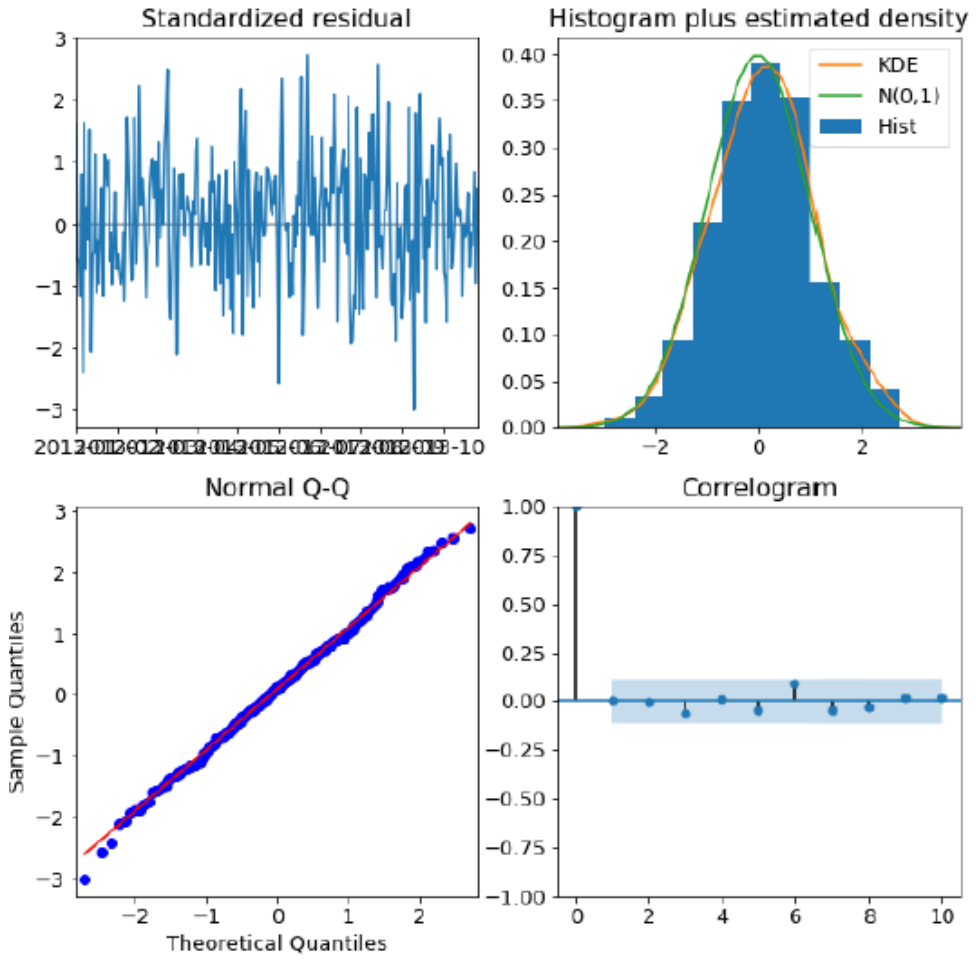
- ARIMA (p, d, q)
  - p is the AR order (like in AR)
  - q is the MA order
  - d is the order of differencing : you can apply diff several times to get a stationary TS ! (not reasonable over 2 may be)
    - Ex : acceleration in the case of a TS to identify human falls in an apparment
    - You can also use ACF and PCF to estimate the right order d



Example of the application of ARIMA for prediction on a non stationary TS (population growth)
In this cas the TS.diff is stationary (d = 1)

Evaluate an ARIMA model

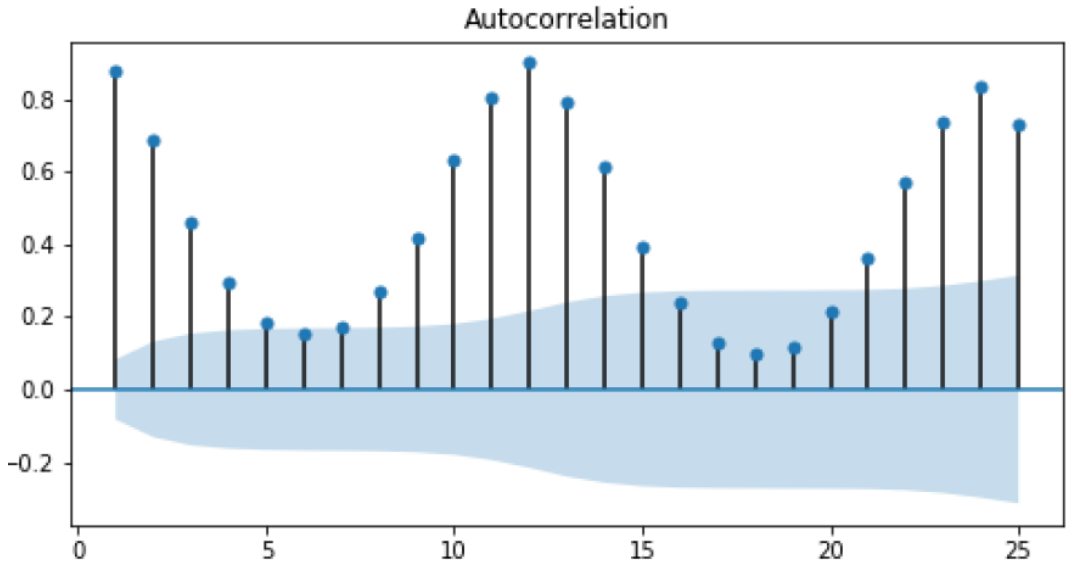
- 2 criteria (similar) to evaluate an ARIMA model :
  - AIC : Akaike information criteria
  - BIC : Bayesian information criteria
  - The lower criterion value, the better the model
  - You can try different values for p, d and q orders and compare the AIC/BIC and choose the better orders
    - An alternative method to ACF/PACF
- There are other tools and indicators to estimate the model (on the right)



4 diagnosis indicators and their visualization
For a SARIMA model

SARIMA : seasonal ARIMA

- SARIMA can take into account a seasonal period
- One have to identify the number of lags (timesteps) corresponding to a period
  - Plots the ACF (see on the right)
  - Ex : on value/month, periodicity of 12 lags (one year)
  - This frequency is a parameter of the SARIMA model
- Very frequent, useful for sensors :
  - Ex : internal temperature : daily period
  - Sometime the period is directly visible on the TS
  - There can be several cumulative periods (ex : day + week)



This ACF clearly identifies a period of retun of 12 months = one year

Rem : SARIMA is complex to calibrate (lots of parameters – use ACF and PACF / AIC and BIC) always check the quality of your model

Resuming ARIMA models and process

The model(s)

S - seasonal

A - autoregressive

R

I - integrated

M - moving average

A

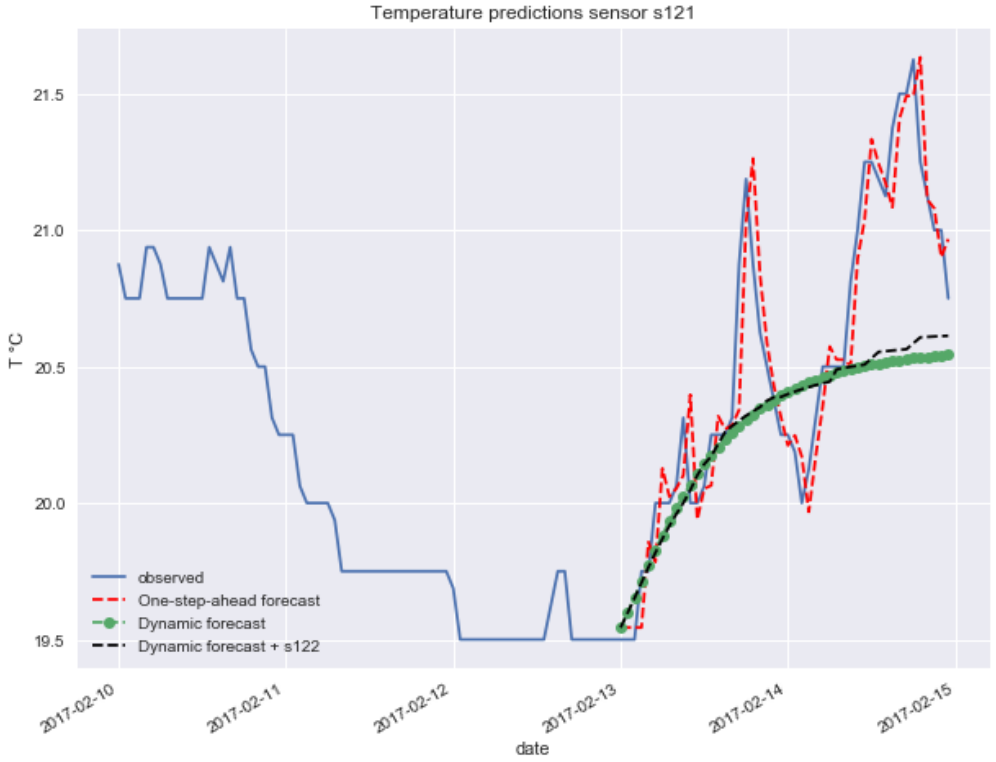
X - exogenous

The (main step of the) process

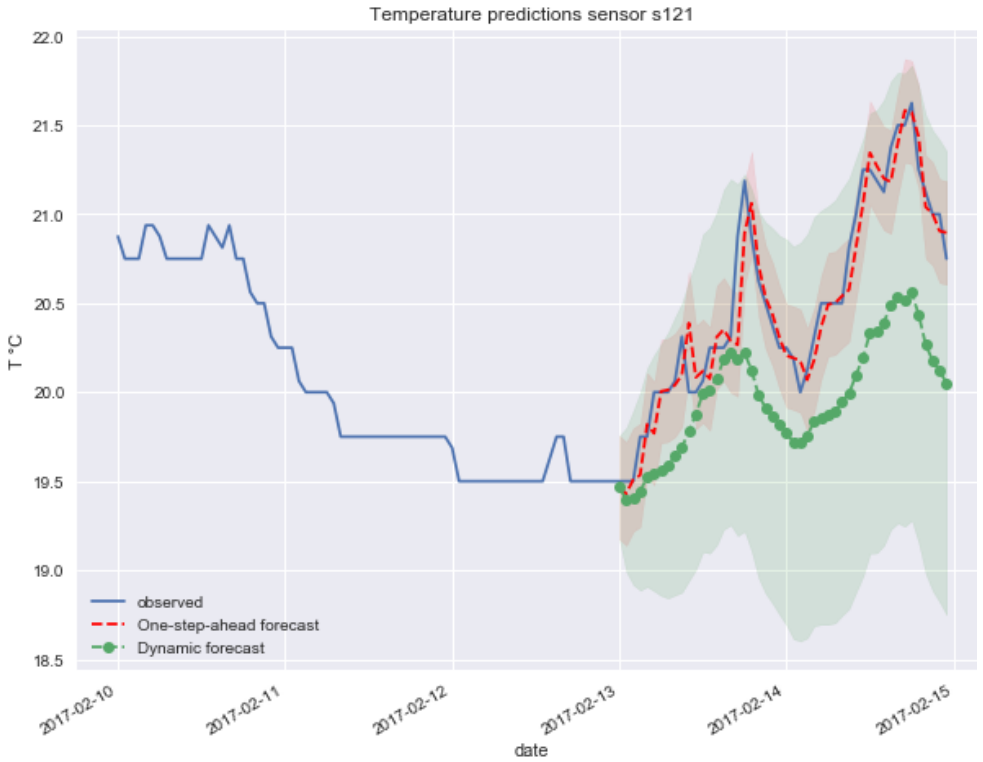
- Test for stationarity and seasonality
- Find promising model orders using ACF and PACF
- Fit models and narrow selection with AIC/BIC
- Perform model diagnostics tests
- Make forecasts

Example of (S)ARIMAX in a building (UTC)

Temperature prediction with a multisensor AEOTEC



Forecast of temperature with ARIMAX without seasonality



Forecast with a daily seasonality (SARIMAX)

Other models that can be used for time prediction

Statistical models

- Croston
- Prophet
- XgBoost
- Neural Net
- LSTN

- Drawback : blackbox, not explainable

Non statistical models

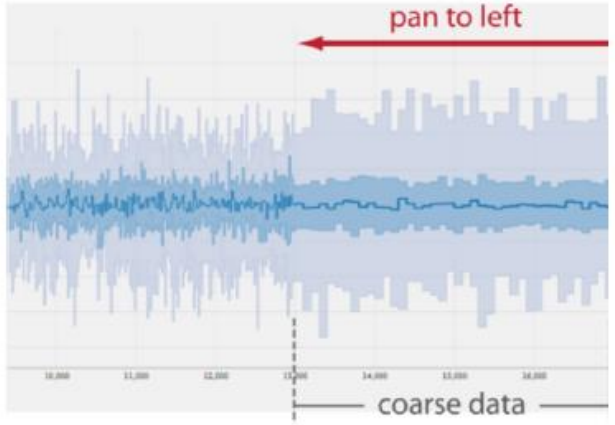
- Genetic programming
- Profile modelling

Sensors and Time Series Analysis

TIME SERIES OTHER SPECIAL ISSUES AND CONCLUSIONS

Streaming and TS indexing

- To store timeseries data of a smart building, one can use either a classical SQL DB or one specialised in TS processing
- Specialized DB for TS are supposed to deal better :
  - Big data storage and indexing
  - Real time streaming data indexing, processing and visualizing
- Example of special DB :
  - Data 360 and Splash (Autodesk)



Visualizing data streaming with Autodesk Splash

Provisional conclusions

- There are no strict boundaries between :
  - pre-processing
  - dimensionality reduction
  - feature selection and extraction
  - Modeling
  - Optimizing

- This is a sequenced (pipeline) process of data transformation from raw data to added-values data that can be interpreted for business goals and human tasks : prediction, diagnoses, decision support …

What we have learned in this course

1. Why timeserie analysis is so important for the smart building/city
2. The main properties and characteristics of TS
3. The main methods and techniques to prepare and transform TS for machine learning : preprocessing, dim. reduction, feature engineering …
4. Examples of application of classical machine learning models (seen in chapter 2) to TS (classification, clustering, regression ….)
5. Time delayed modeling and prediction with the family of ARIMA models (AutoRegression)
6. Real examples and application with data coming from sensors of the smart building

Bibliography

- ESLING, Philippe et AGON, Carlos. Time-series data mining. *ACM Computing Surveys (CSUR)*, 2012, vol. 45, no 1, p. 1-34.