

TIME SERIES : INTRODUCTION, TASKS, APPROACHES & ISSUES

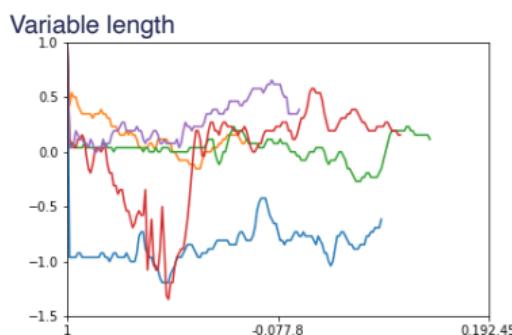
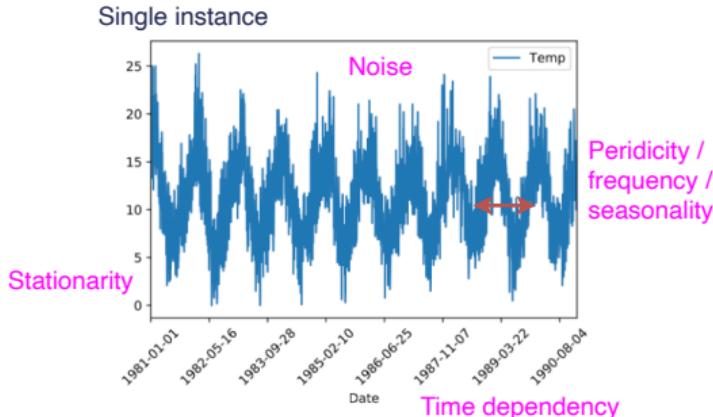
Vincent Guigue



INTRODUCTION

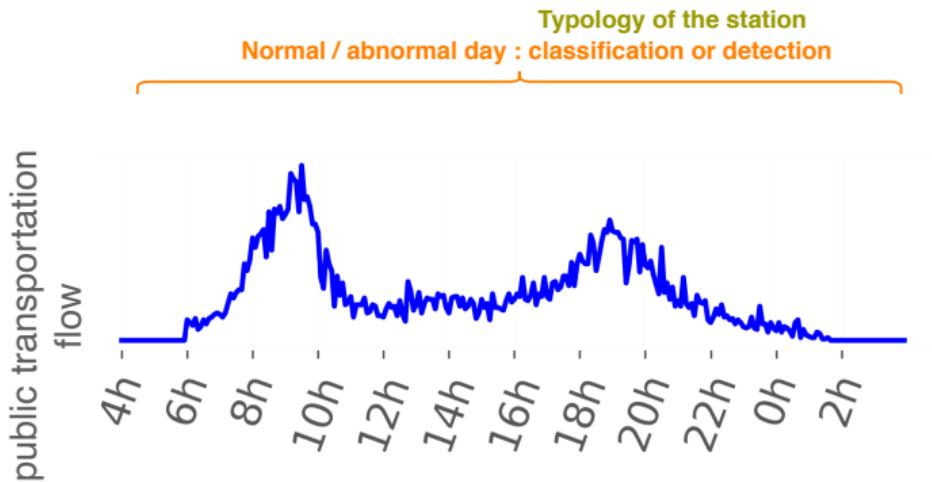
General schedule

- 1 Signal specificities
- 2 Very different applications
- 3 Different points of view
- 4 Benefits/pitfalls
of ML approaches
- 5 Benefits of deep learning
- 6 New opportunities



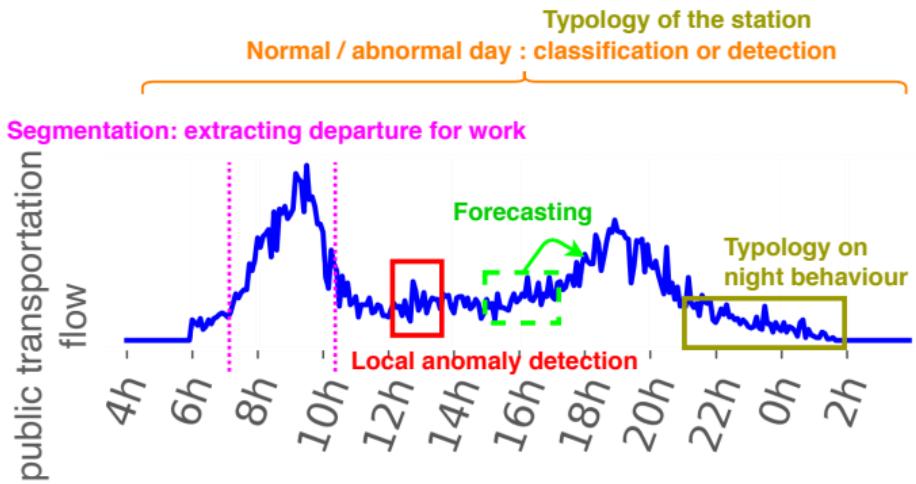
General schedule

- 1 Signal specificities
- 2 Very different applications
- 3 Different points of view
- 4 Benefits/pitfalls
of ML approaches
- 5 Benefits of deep learning
- 6 New opportunities



General schedule

- 1 Signal specificities
- 2 Very different applications
- 3 Different points of view
- 4 Benefits/pitfalls
of ML approaches
- 5 Benefits of deep learning
- 6 New opportunities



General schedule

- 1 Signal specificities
- 2 Very different applications
- 3 Different points of view
- 4 Benefits/pitfalls
of ML approaches
- 5 Benefits of deep learning
- 6 New opportunities



Statistics

Data Science



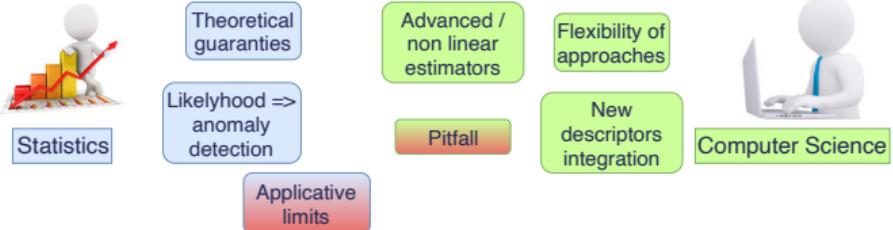
Sales history



Computer Science

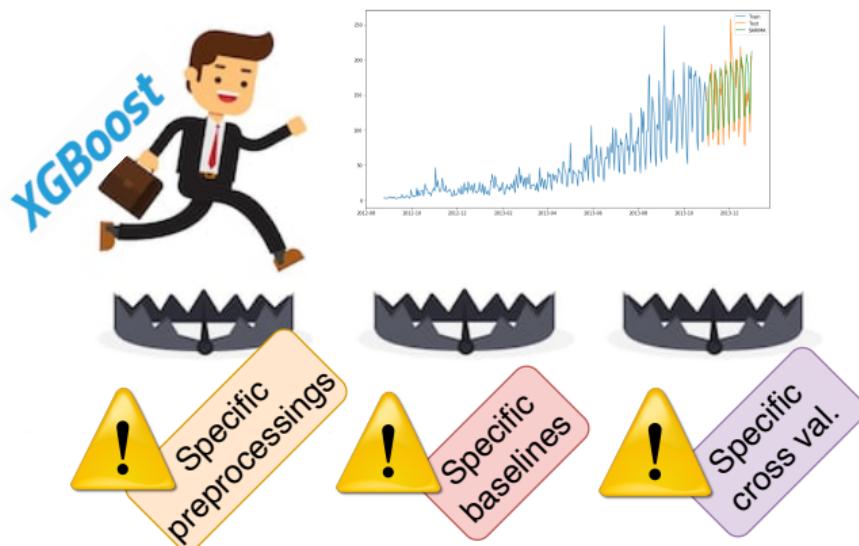
General schedule

- 1 Signal specificities
- 2 Very different applications
- 3 Different points of view
- 4 Benefits/pitfalls
of ML approaches
- 5 Benefits of deep learning
- 6 New opportunities



General schedule

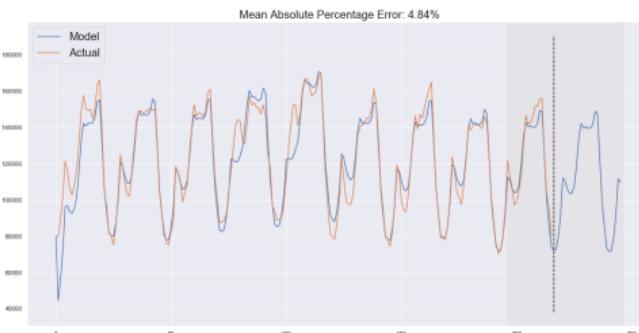
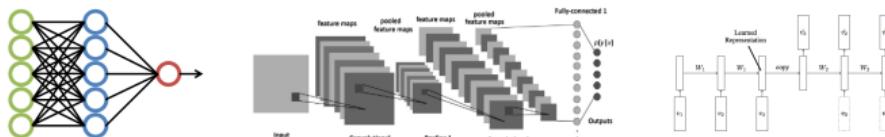
- 1 Signal specificities
- 2 Very different applications
- 3 Different points of view
- 4 Benefits/pitfalls
of ML approaches
- 5 Benefits of deep learning
- 6 New opportunities



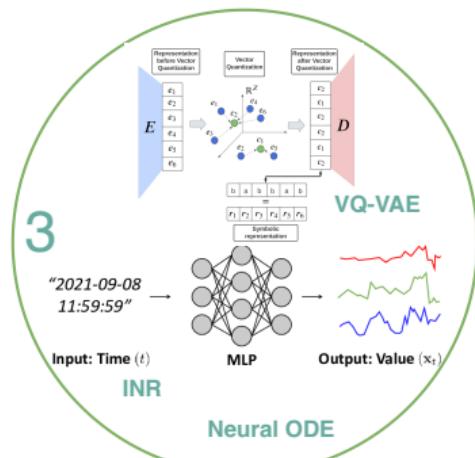
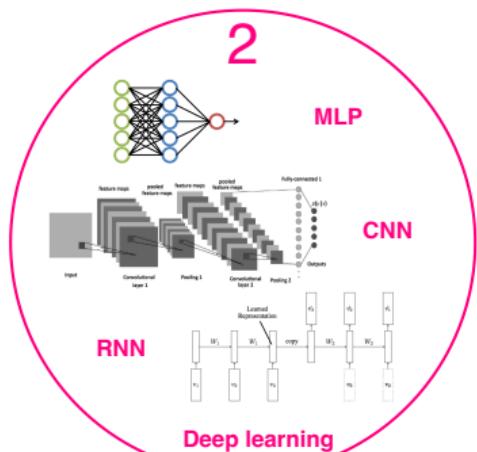
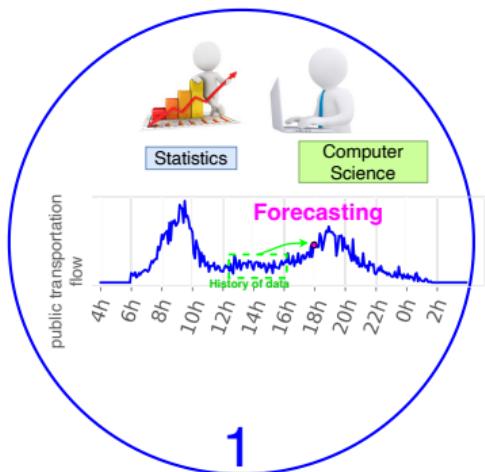
General schedule

- 1 Signal specificities
- 2 Very different applications
- 3 Different points of view
- 4 Benefits/pitfalls of ML approaches
- 5 Benefits of deep learning
- 6 New opportunities

Which architecture for which application?
... And which benefits?

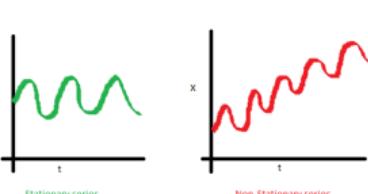


Courses organization



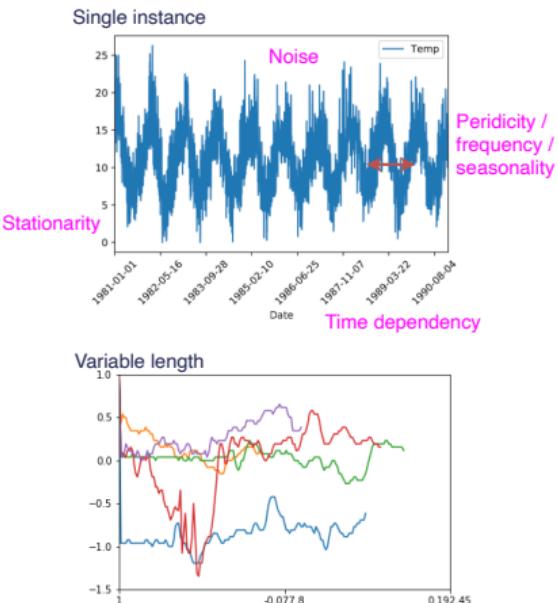
Time series = specific object

- ▶ Variable length
 - ▶ An issue... Or not, depending on the application
- ▶ Noise & de-noising
 - ▶ Specific de-noising strategy based on temporal dependency
- ▶ Stationarity / Periodicity
 - ▶ Constant statistical properties over time (mean, std. dev.)



Variable
covariance

- ▶ Number of instances
 - ▶ May be one !



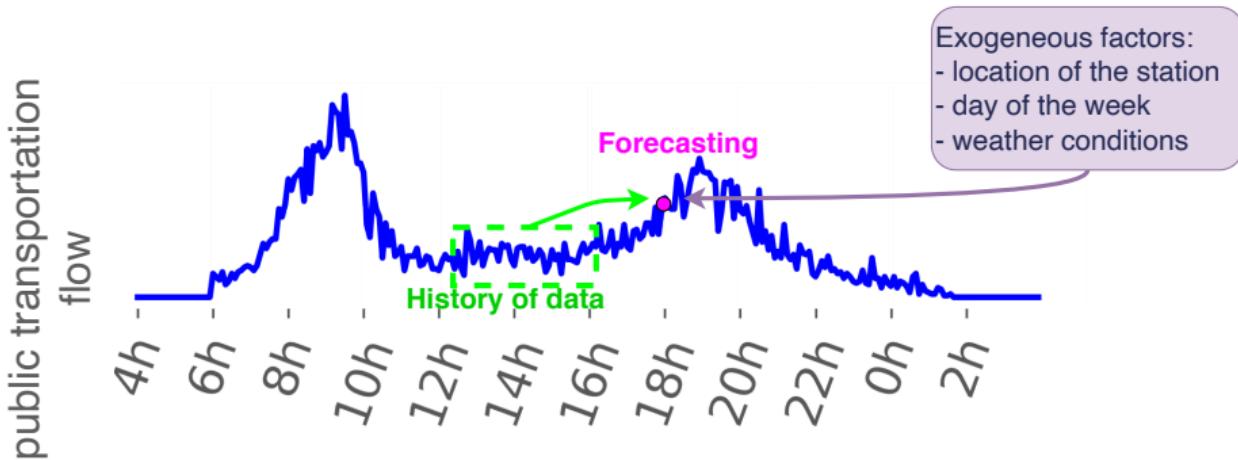
STATISTICAL FORECASTING OF NEXT VALUES

Forecasting task

- 1 Modeling prediction from history only
- 2 Adding context = exogenous factors

Prediction is very difficult, especially about the future

Niels Bohr



Can be applied on single/multiple instance problem.

Auto-Regressive approaches

AR/ARMA

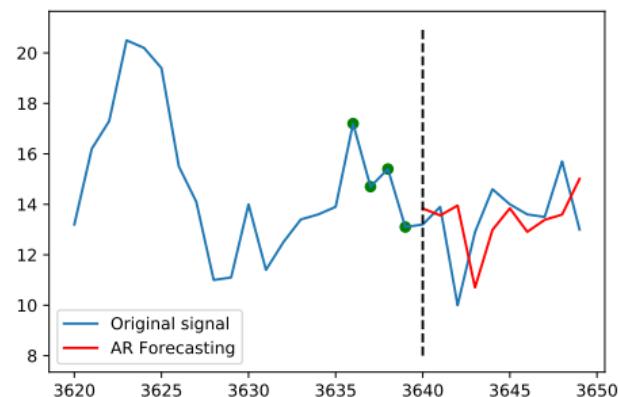
The historical answer to time-series forecasting (from statisticians)

AR : Auto-Regressive modeling:

$$Y_t = \alpha + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \dots + \alpha_p Y_{t-p} + \varepsilon_t \quad (1)$$



Complete Guide to Time Series Forecasting in Python,
<https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/>



AR (order 4): 4 last measures are weighted by α to predict T

Auto-Regressive approaches

AR/ARMA

The historical answer to time-series forecasting (from statisticians)

AR : Auto-Regressive modeling:

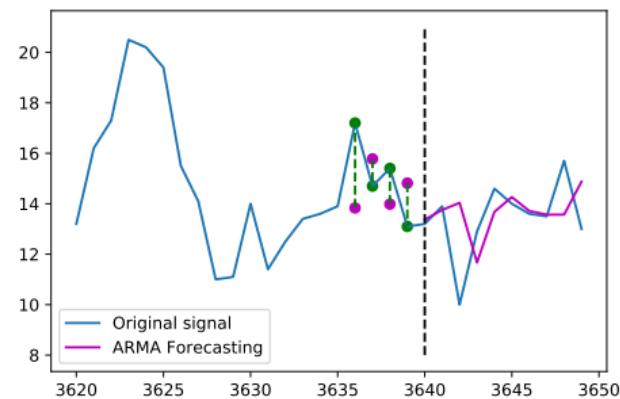
$$Y_t = \alpha + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \dots + \alpha_p Y_{t-p} + \varepsilon_t \quad (1)$$

ARMA : Auto-Regressive Moving Average modeling:

$$Y_t = \alpha + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \dots + \alpha_p Y_{t-p} + \beta_1 \varepsilon_{t-1} + \beta_2 \varepsilon_{t-2} + \dots + \beta_q \varepsilon_{t-q} \quad (2)$$



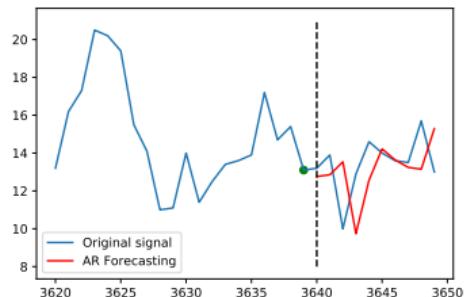
Complete Guide to Time Series Forecasting in Python,
<https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/>



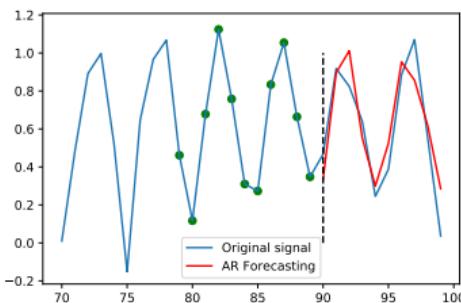
ARMA (order 4,4): 4 last measures are weighted by α & 4 errors ε are weighted by β to predict T

AR basic examples & Periodicity discussion

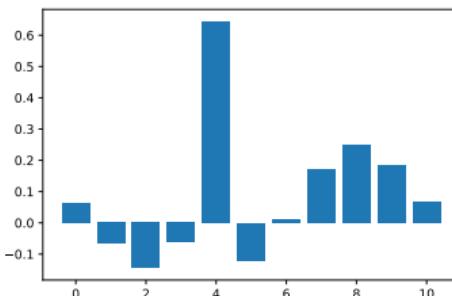
AR (order = 1): a very intuitive model:



AR on periodic time series (order = 10):



- ▶ Order 1 = delayed version of the original signal
- ▶ Periodic signals: high coefficients on the period



AR inference

AR:

- ▶ Prediction at t :

$$\hat{y}_t = \alpha + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \dots + \alpha_p y_{t-p}$$

- ▶ Dynamic Prediction at t (from $t-2$):

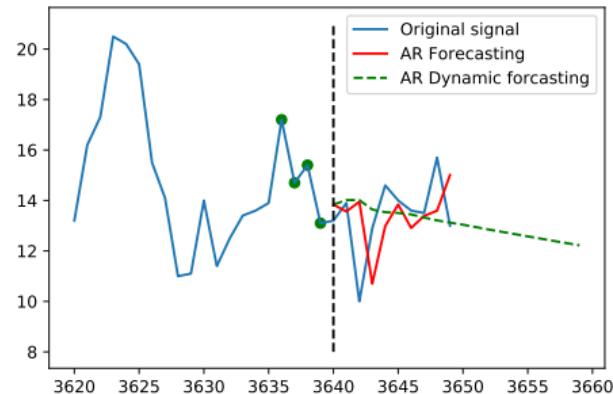
$$\hat{y}_t = \alpha + \alpha_1 \hat{y}_{t-1} + \alpha_2 y_{t-2} + \dots + \alpha_p y_{t-p}$$

ARMA:

- ▶ Prediction at t :

$$y_t = \alpha + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \dots + \alpha_p y_{t-p} + \beta_1 \varepsilon_{t-1} + \beta_2 \varepsilon_{t-2} + \dots + \beta_q \varepsilon_{t-q}$$

- ▶ Dynamic Prediction at t (from $t-2$): ε_{t-1} can no longer be computed



ε_t that we can't compute are set to 0

Parameter optimization

Problem formulation (MSE):

$$\mathcal{L} = \sum_t (y_t - \hat{y}_t)^2, \quad \arg \min_{\alpha, (\beta)} \mathcal{L}$$

- ▶ AR problem admits a closed form solution (Yule Walker)
- ▶ ARMA is a convex problem that is solved by gradient descent

During training, \hat{y}_t is estimated from real y_{t-p} values...

Our model is not dedicated to long term prediction.



Wikipedia,
https://en.wikipedia.org/wiki/Autoregressive_model

Finding AR optimal hyper-parameters

Statistician

- ▶ Information Criterion : AIC (/ BIC)
- ▶ Akaike information criterion:

$$AIC = 2k - 2 \ln(\mathcal{L})$$

k = nb estimated parameters

- ▶ Maximizing likelihood
- ▶ while penalizing model complexity

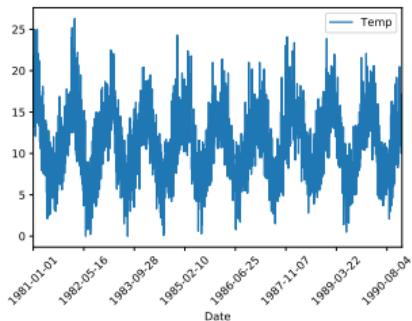
In practice: always very **low orders**

Computer scientist

- ▶ Cross validation (always)
 - ▶ Reconstruction criterion: MSE
- ▶ Estimating the generalization error on unseen data

ARIMA

- ▶ AR / ARMA approaches are dedicated to **stationary signals**
- ▶ Issue 1: **measuring** stationarity
- ▶ Issue 2: **improving** stationarity



Results of Dickey-Fuller Test:

Test Statistic	-4.444805
p-value	0.000247
#Lags Used	20.000000
Number of Observations Used	3629.000000
Critical Value (1%)	-3.432153
Critical Value (5%)	-2.862337
Critical Value (10%)	-2.567194

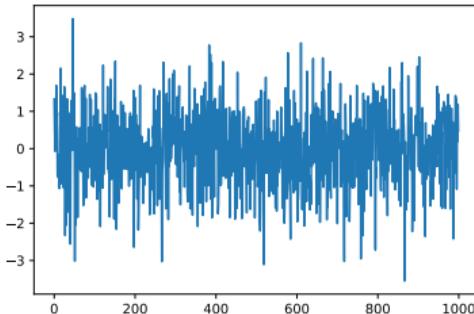
Hypothesis testing (e.g. Dicky Fuller Test):



mind the definition of the null hypothesis
(Dickey-Fuller : null = non-stationary)

Stationarity & ARIMA model

- AR / ARMA approaches are dedicated to stationary signals
- Issue 1: **measuring** stationarity
- Issue 2: **improving** stationarity



Hypothesis testing (e.g. Dicky Fuller Test):

Results of Dickey-Fuller Test:

Test Statistic	-31.448939
p-value	0.000000
#Lags Used	0.000000
Number of Observations Used	999.000000
Critical Value (1%)	-3.436913
Critical Value (5%)	-2.864437
Critical Value (10%)	-2.568313



mind the definition of the null hypothesis
(Dickey-Fuller : null = non-stationary)

Improving stationarity = signal differencing

Differencing the signal:

Order 1 :

$\delta_t = y_t - y_{t-1}$ instead of y_t

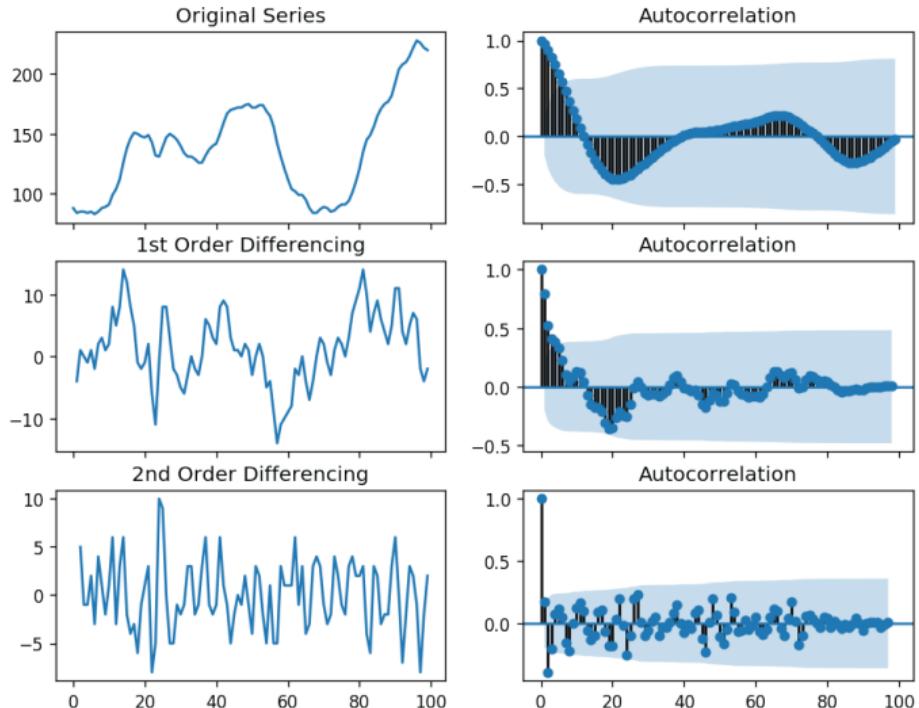
Order 2 :

$\delta_t^{(2)} = \delta_t - \delta_{t-1}$ instead of y_t

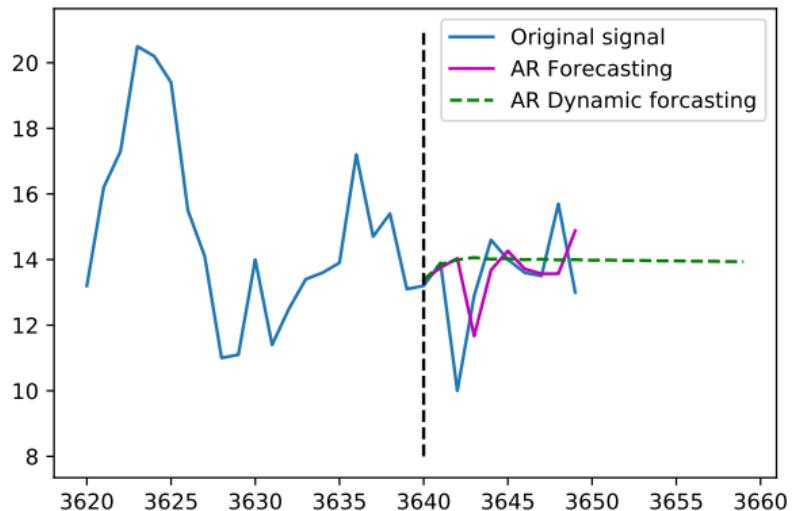
ARIMA :

Adding a (I)ntegrating parameter
= order of differencing

No autocorrelation =
stationary signal



What can we expect from AR modeling in practice?



In practice:

ARMA with low order = good local prediction

- ▶ Interesting modeling at $t + 1$
- ▶ Flat prediction at $t + N$

Main issue:

no **seasonality** is taken into account
(\approx a way to model long term dependency)

Extracting trends & seasonality

1 Working at different scales: year, month, week, day, ...
depending on the dataset

2 Seasonality extraction is done by convolution

- ▶ denoting a trend t , a season s and a residue ε
 - ▶ period p must be provided

▶ Additive model

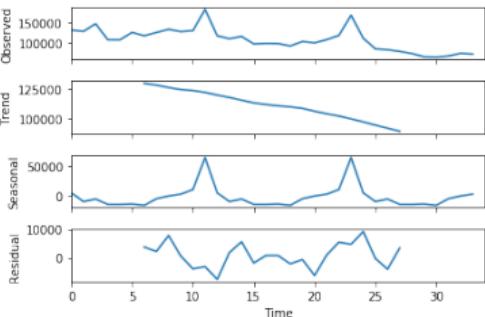
$$y = t + s + \varepsilon$$

▶ Multiplicative model

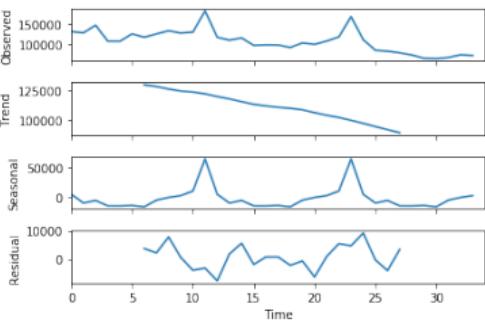
$$y = t \times s \times \varepsilon$$

3 ARMA is performed on the residue

Additive model:



Multiplicative model:

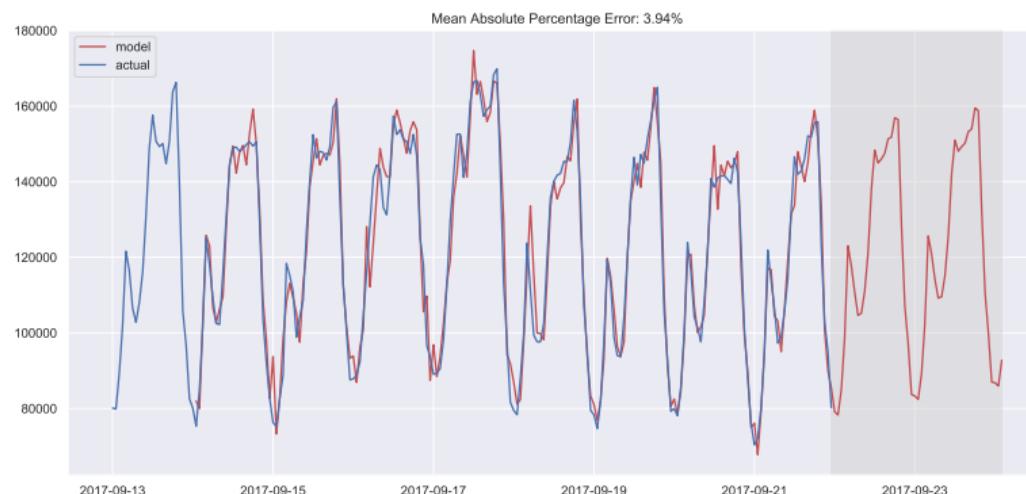


Time series Basics : Exploring traditional TS, G. Jagan
[https://www.kaggle.com/jagangupta/
time-series-basics-exploring-traditional-ts](https://www.kaggle.com/jagangupta/time-series-basics-exploring-traditional-ts)

SARIMA

Extension: Seasonality ARIMA

- ▶ Add a season length parameter
- ▶ Order(s) + Integration inside season
- ▶ + at the season level : $s_t = \alpha s_{t-1} + \dots$



Adding seasonality enables long term better predictions.

ARMA tends to 0.
Seasonality & trend remain reasonable.

AR/ARMA/SARIMA & exogenous factors

Give side informations about:

- ▶ the day, the hour,
- ▶ the weather condition...

AR:

$$\hat{y}_t = \alpha + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \dots + \alpha_p y_{t-p}$$

AR + exogenous factors e_1, e_2, \dots :

$$\hat{y}_t = \alpha + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \dots + \alpha_p y_{t-p} + \beta_1 e_{t,1} + \beta_2 e_{t,2} + \dots$$

⇒ you must provide exogenous factor for the inference on the test set

Exponential Filtering (Holt-Winters predictor)

A well known alternative to SARIMA:

- ▶ Order 1:

$$\hat{y}_t = \alpha \cdot y_t + (1 - \alpha) \cdot \hat{y}_{t-1}$$

- ▶ Seasonality triple exponential filtering (=Holt-Winters)

Prediction at horizon m , season = s , season length = L

1st order recursive model: $\ell_t = \alpha (y_t - s_{t-L}) + (1 - \alpha) (\ell_{t-1} + b_{t-1})$

Differencing: $b_t = \beta (\ell_t - \ell_{t-1}) + (1 - \beta) b_{t-1}$

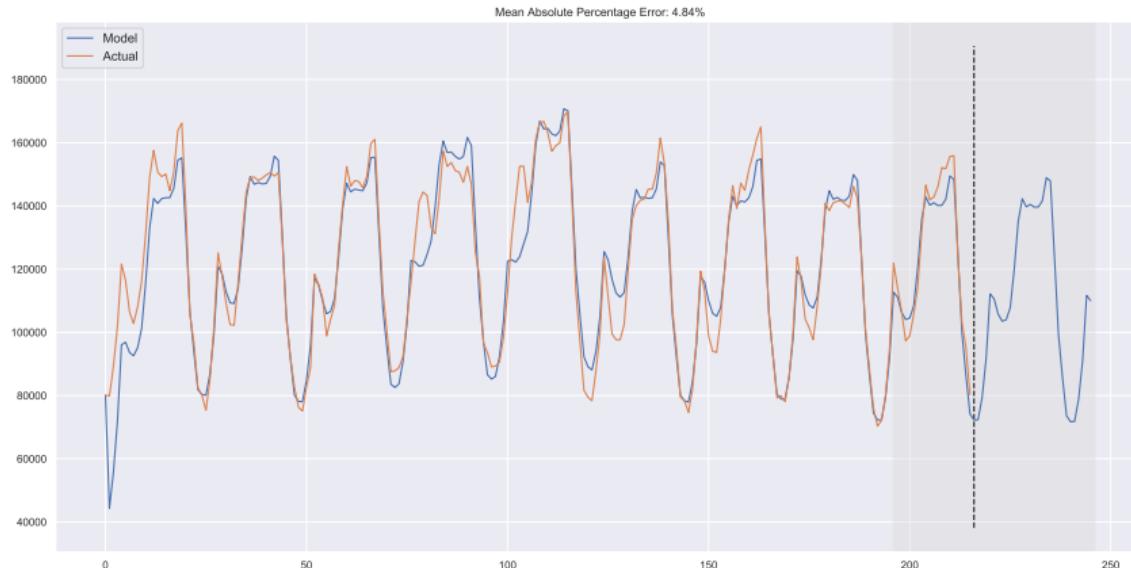
Seasonality: $s_t = \gamma (y_t - \ell_t) + (1 - \gamma) s_{t-L}$

Combination: $\hat{y}_{t+m} = \ell_t + mb_t + s_{t-L+1+(m-1)\%L}$

NB: the way to build this estimator is close to the gradient computation in ADAM

Exponential Filtering : results are close to SARIMA

Seasonality becomes more important than local modeling...



Greater horizon, simpler model

To look away an averaged season + trend is enough

Facebook Prophet model

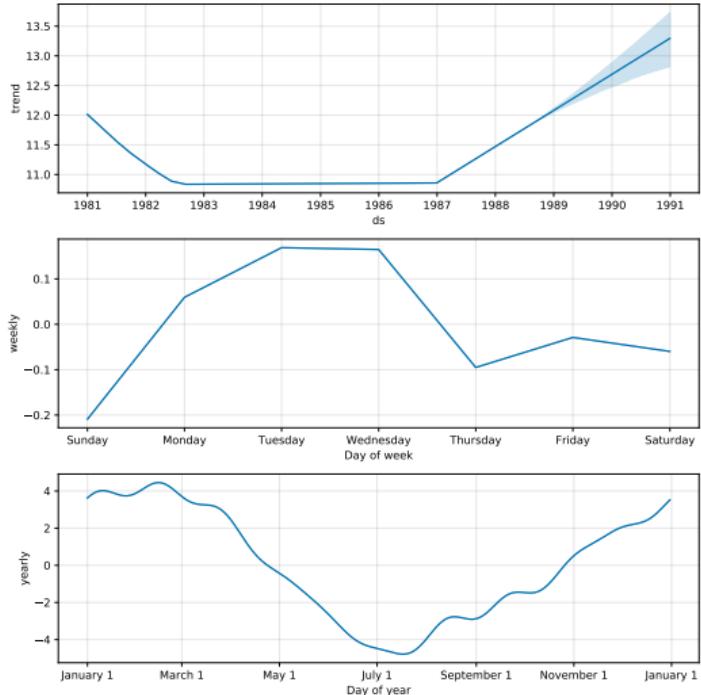
General formulation:

$$y(t) = g(t) + s(t) + h(t) + e(t)$$

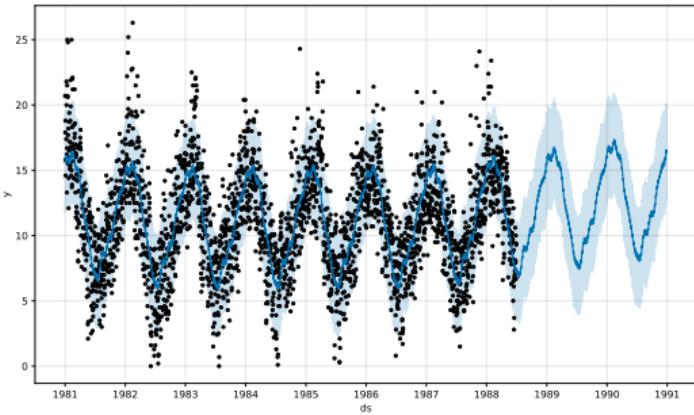
- ▶ $g(t)$: trends (for non periodic changes)
- ▶ $s(t)$: seasonality. In fact seasonality is multi-scale:
 - ▶ $s_h(t)$ hour, $s_d(t)$ day, $s_w(t)$ week, $s_m(t)$ month
- ▶ $h(t)$: holidays = prophet denomination for exogenous factors
- ▶ $e(t)$: residue

⇒ from statistics... Prophet \approx SARIMA

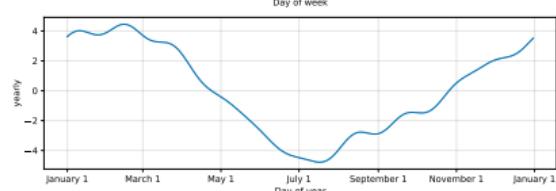
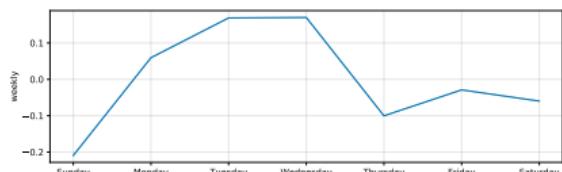
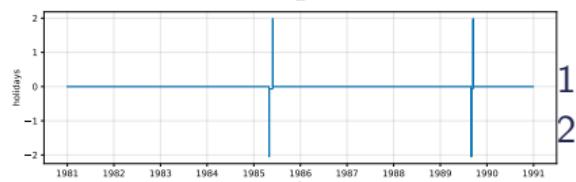
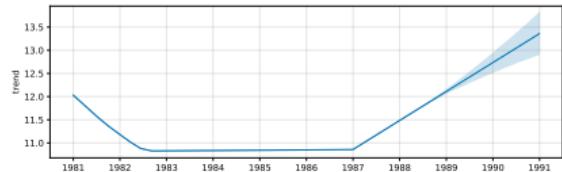
Prophet output



Australian daily minimum temperature



Prophet exogenous factor encoding



Simply define a DataFrame for your event & add it...

```
mod = Prophet(holidays=special_events)  
#mod = Prophet()
```

Additional refinement:

- ▶ definition of overlapping special events
- ▶ possibilities to define additive or subtractive behaviour.

MACHINE LEARNING FORECASTING

From AR to standard ML chains

- ▶ Linear (& non linear) regression: not only an interpolator but also a predictor
- ▶ feature engineering
 - ▶ Statistical features (tsfresh) + time freq
 - ▶ Sales prediction case
 - ▶ example of features
- ▶ SVM, XGboost, ... Or neural networks
- ▶ **Easy & cheap**

some models will never be *production ready* as they demand too much time for the data preparation (for example, SARIMA),

or require frequent re-training on new data (again, SARIMA),

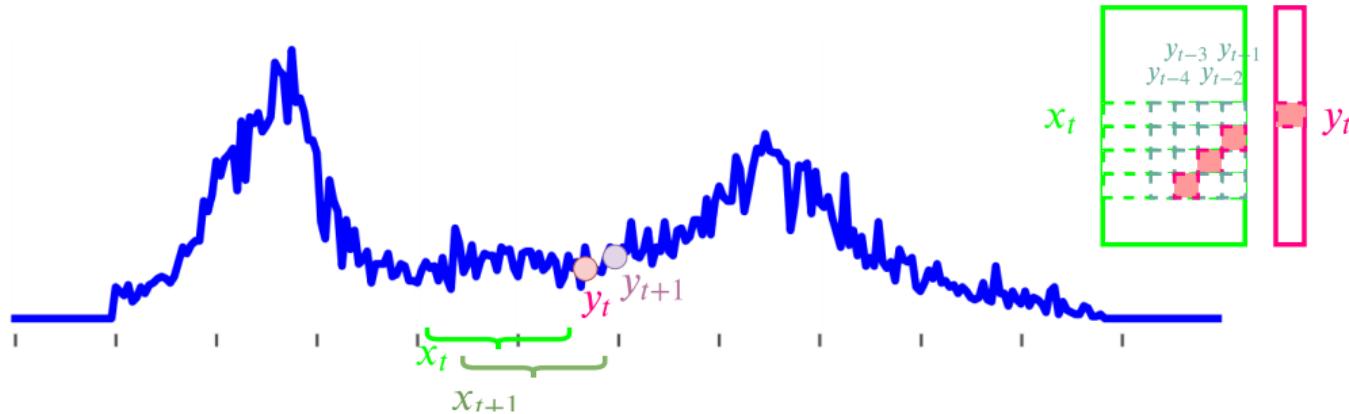
or are difficult to tune (good example - SARIMA),

so it's very often much easier to select a couple of features from the existing time series and build a simple linear regression or, say, a random forest. Good and cheap.

Dmitry Sergeyev

Rolling approaches

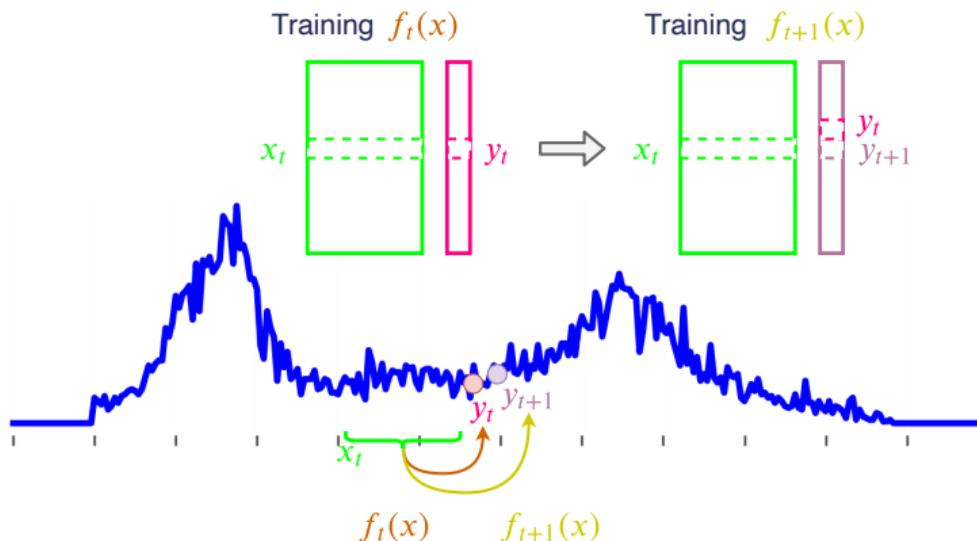
Introducing AR features in an ML environment:



Time	y	lag_6	lag_7	lag_8	lag_9	lag_10	lag_11	lag_12
2017-09-21 17:00:00	151790	132335.0	114380.0	105635.0	98860.0	97290.0	106495.0	113950.0
2017-09-21 18:00:00	155665	146630.0	132335.0	114380.0	105635.0	98860.0	97290.0	106495.0
2017-09-21 19:00:00	155890	141995.0	146630.0	132335.0	114380.0	105635.0	98860.0	97290.0

Predict further in time

- ▶ how to do it with ARMA?
 - ▶ Train your model at $t + 1$ (always)
 - ▶ Apply learnt coefficient α on prediction \hat{y}
 - ▶ Expect poor results (without seasonality)
- ▶ how to do it with ML chain
 - ▶ Learning to predict further directly

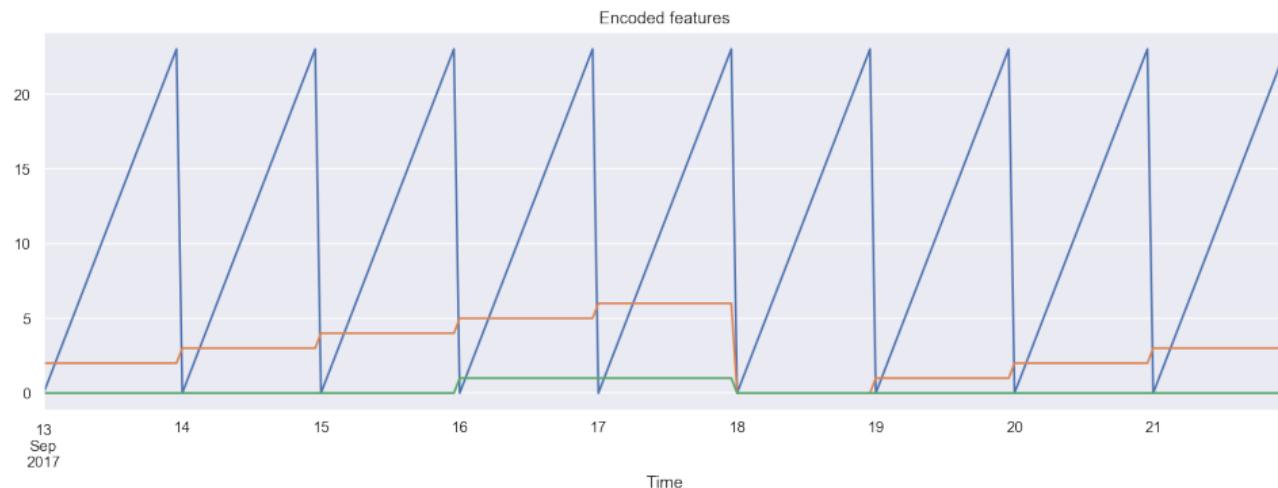


Exogenous factors / Feature engineering

Exogenous factors is straightforward in ML: just add features in the dataset

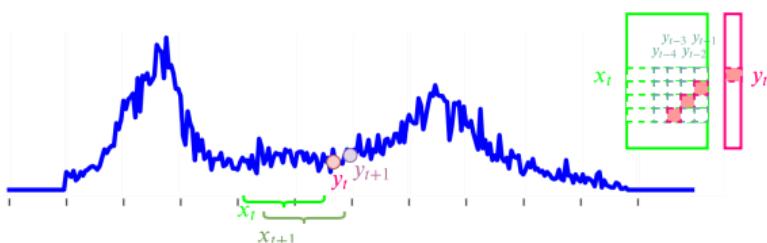
- ▶ Another way to encode seasonality
- ▶ Using pandas to catch up prophet functions
 - ▶ Types of days
 - ▶ Hours...

Example of time encoding: hour, day, week-end:



Feature engineering

- ▶ Depending on the application ⇒ discussions with experts
[Often] more expert features ⇒ more performance
- ▶ Local statistics computations
- ▶ Statistic moment
- ▶ Frequency / power spectral density...
- ▶ tsfresh
 - ▶ Hundreds of features...
 - ▶ ... & test of relevant ones
 - ▶ ! to be used only on time series
(=lag variables)
- ▶ Classical feature engineering
 - ▶ Feature clustering, ...



Feature engineering

- ▶ Depending on the application ⇒ discussions with experts
[Often] more expert features ⇒ more performance
- ▶ Local statistics computations
- ▶ Statistic moment
- ▶ Frequency / power spectral density...
- ▶ tsfresh
 - ▶ Hundreds of features...
 - ▶ ... & test of relevant ones
 - ▶  to be used only on time series
(=lag variables)
- ▶ Classical feature engineering
 - ▶ Feature clustering, ...

<code>abs_energy (x)</code>	Returns the absolute energy of the time series.
<code>absolute_sum_of_changes (x)</code>	Returns the sum over the absolute value of differences of consecutive elements.
<code>agg_autocorrelation (x, param)</code>	Calculates the value of an aggregation function over the autocorrelation.
<code>agg_linear_trend (x, param)</code>	Calculates a linear least-squares regression over the aggregation function.
<code>approximate_entropy (x, m, r)</code>	Implements a vectorized Approximate Entropy calculator.
<code>ar_coefficient (x, param)</code>	This feature calculator fits the unconditioned autoregression.
<code>augmented_dickey_fuller (x, param)</code>	The Augmented Dickey-Fuller test is a hypothesis test for unit roots.
<code>autocorrelation (x, lag)</code>	Calculates the autocorrelation of the specified lags.
<code>binned_entropy (x, max_bins)</code>	First bins the values of x into max_bins equidistant bins.
<code>c3 (x, lag)</code>	This function calculates the value of the C3 statistic.
<code>change_quantiles (x, ql, qh, isabs, f_agg)</code>	First fixes a corridor given by the quantiles ql and qh.
<code>cid_ce (x, normalize)</code>	This function calculator is an estimate for a confidence interval.

Sales prediction use case

- ▶ Nature of the data
 - ▶ Shops
 - ▶ Items
- ▶ Target:
 - ▶ Fine grain: predicting the amount of each items in each Shops
 - ▶ General: sales revenue

Idea: extracting features for all objects



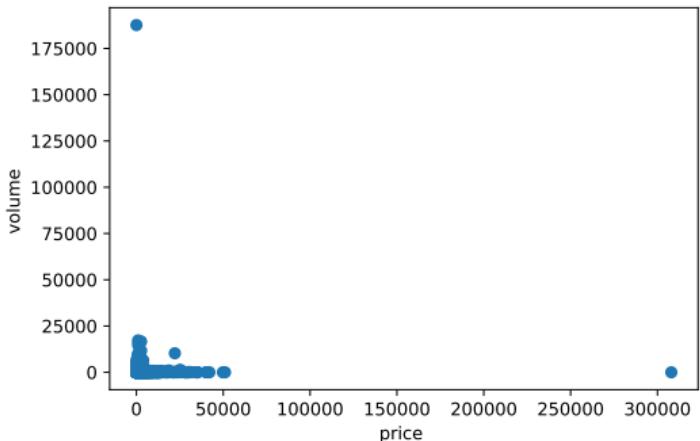
K. Yacovlev, Kaggle notebook, 2019

<https://www.kaggle.com/kyakovlev/1st-place-solution-part-1-hands-on-data>

Sales prediction use case : classical features

- ▶ **Items** (aggregated over all shops)
 - ▶ Item category (from expert, from name, ...)
 - ▶ Item general trends
 - ▶ Binary: Is deprecated / Is new
 - ▶ Price/volume categorization :
 - ▶ removing (or separating outliers)
 - ▶ linspace separation
 - ▶ histogram
 - ▶ clustering
- ▶ **Shops**
 - ▶ Same features + co-clustering with items
- ▶ **Time**
 - ▶ Black Friday, holidays, ...

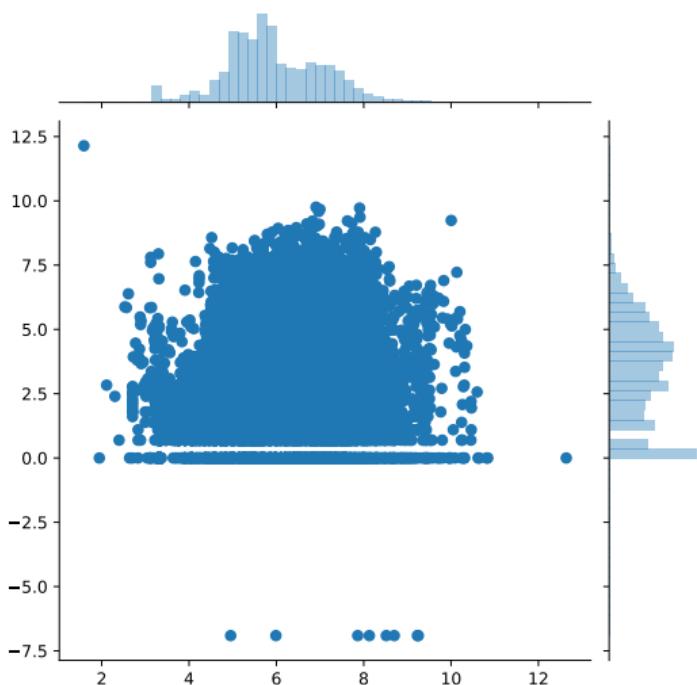
Items prices & volumes



Sales prediction use case : classical features

- ▶ **Items** (aggregated over all shops)
 - ▶ Item category (from expert, from name, ...)
 - ▶ Item general trends
 - ▶ Binary: Is deprecated / Is new
 - ▶ Price/volume categorization :
 - ▶ removing (or separating outliers)
 - ▶ linspace separation
 - ▶ histogram
 - ▶ clustering
- ▶ **Shops**
 - ▶ Same features + co-clustering with items
- ▶ **Time**
 - ▶ Black Friday, holidays, ...

Items **log** prices & volumes

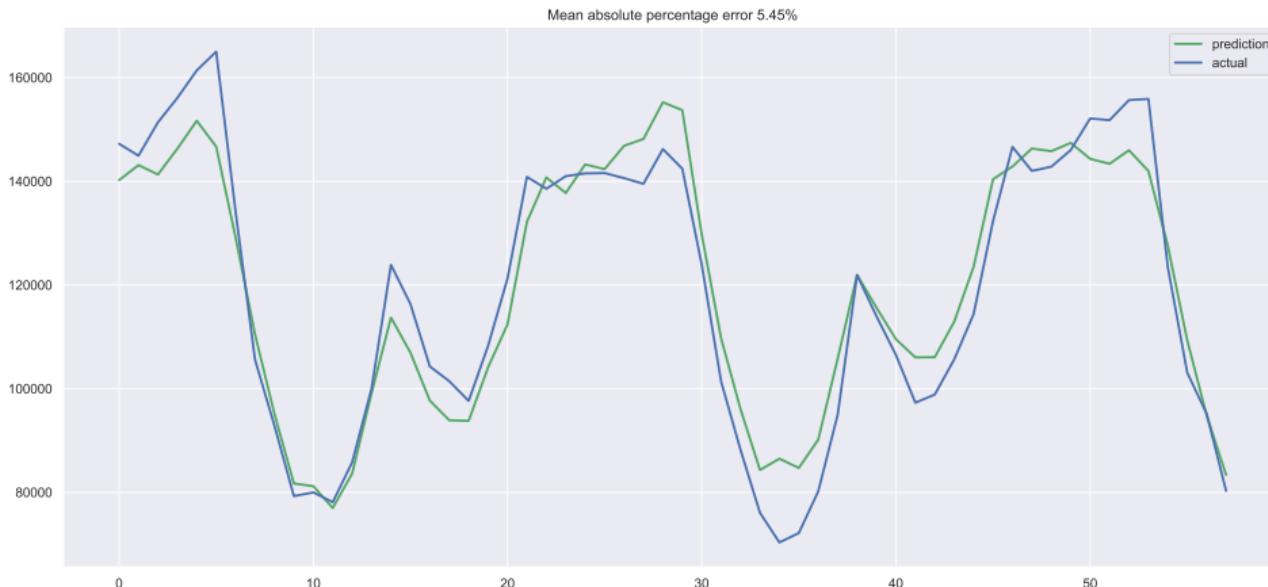


Sales prediction use case : mono/multivariate approach

- ▶ Single shop prediction
 - ▶ with some features computed on multiple shops
 - ▶ Multiple shop prediction
 - ▶ Predicting all item per shop sales
-
- ▶ Feature engineering makes monovariate prediction very strong
 - ▶ Deep learning (try to) tackles multivariate prediction
 - ▶ To extract relevant feature automatically
 - ▶ To find fine correlation between shop/item dynamics

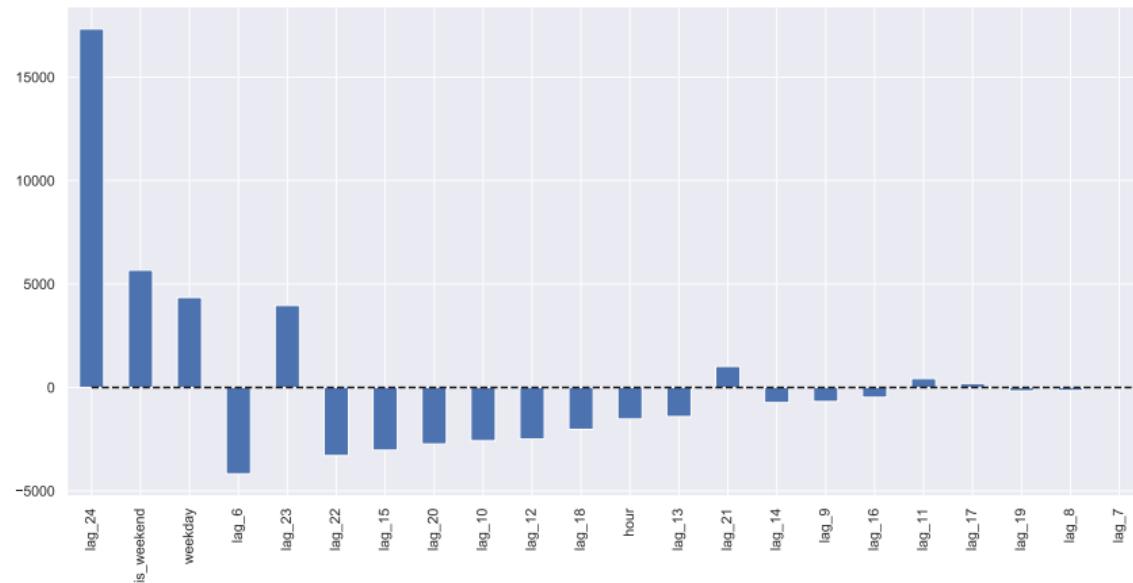
Variable weights normalization & interpretations

- ▶ Do not forget to normalize your data
 - ▶ Always in ML... But really mandatory when dealing with heterogenous variables
- ▶ Model introspection is always a good Idea



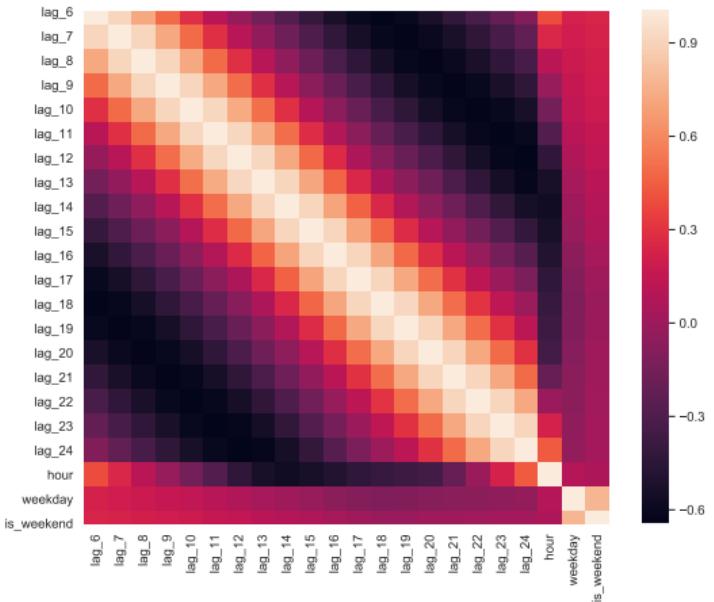
Variable weights normalization & interpretations

- ▶ Do not forget to normalize your data
 - ▶ Always in ML... But really mandatory when dealing with heterogenous variables
- ▶ Model introspection is always a good Idea



Regularization & variable selection

Extracting many features (or even several) lead to this kind of data shape:



- ▶ Ridge/regularized logistic regression
 - ▶ at least, to reduce the bad impact of correlated variables
- ▶ LASSO/Variable selection procedure
 - ▶ Reducing the number of features
 - ▶ scikit-learn...

XGBoost (everything?)



A nice algorithm...

And a **wonderful** implementation!

- ▶ like libSVM, word2vec, ...

Unsurprisingly \Rightarrow Yes, it is a good idea

Model introspection: let's exploit all available functions



Amjad Abu-Rmileh, The Multiple faces of Feature importance in XGBoost

<https://towardsdatascience.com/>

[be-careful-when-interpreting-your-features-importance](#)

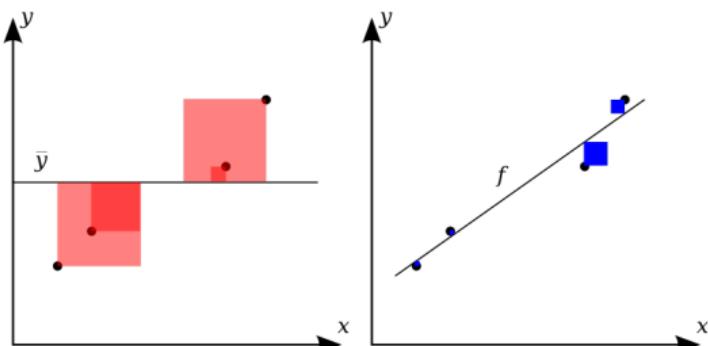
METRICS

Metrics – How to evaluate if our predictions are relevant?

R squared

coefficient of determination (in econometrics it can be interpreted as a percentage of variance explained by the model)

$$R^2 = 1 - \frac{SS_{residue}}{SS_{total}}$$



- 1 : all variance of the data explained \Rightarrow best results
- 0 : worst model
- < 0 : how did you make it?

Classical metrics

- ▶ Mean Absolute error

$$MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n}$$

- ▶ Median Absolute Error (robust to outliers)

$$MedAE = \text{median}(|y_1 - \hat{y}_1|, \dots, |y_n - \hat{y}_n|)$$

- ▶ Mean Absolute Percentage Error

$$MAPE = \frac{100}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{y_i}$$

Choose the metric adapted to your data to evaluate... Even if you (often) use MSE as a learning criterion

Side effects with MAPE, definition of SMAPE

Working on sparse data (e.g. validations of a single user in public transportation)

Lots of 0 in the ground truth:

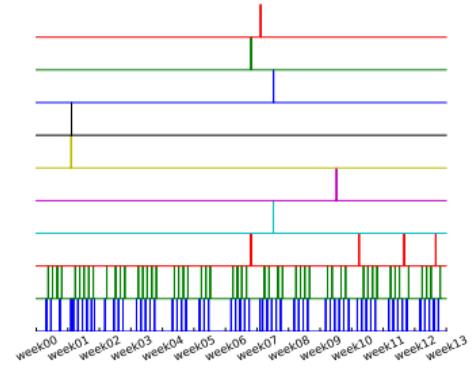
$$\blacktriangleright \text{MAPE} = \frac{100}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{y_i} \text{ diverges}$$

Solutions:

- ▶ Rough aggregation over time to reduce sparseness
- ▶ Dedicated metrics
- ▶ ... SMAPE:

$$\text{SMAPE}_1 = \frac{100\%}{n} \sum_{t=1}^n \frac{|y_t - \hat{y}_t|}{(|\hat{y}_t| + |y_t|)/2} \quad \text{or}$$

$$\text{SMAPE}_2 = \frac{\sum_{t=1}^n |y_t - \hat{y}_t|}{\sum_{t=1}^n (\hat{y}_t + y_t)}$$



Not magic... but sometimes robust enough to build an operational system

ANOMALIES

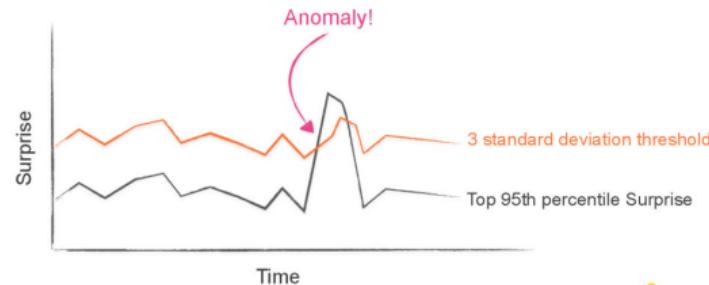
Anomaly definition

Something that is not supposed to append. Different cases:

- ▶ Outlier / error of measurement
- ▶ Distance between observations and predictions
- ▶ Anomaly tag labeled in the dataset

⇒ We focus on distances

⇒ Something that surprises the model: compute statistics on errors made by the model



Required for **in-depth evaluation** & for model **monitoring** in production



Anomaly detection: proposed implementation

- ▶ Computing bounds very easily:

```
1 mae = mean_absolute_error(series[window:], prediction[window:])
2 deviation = np.std(series[window:] - prediction[window:])
3 lower_bound = prediction - (mae + scale * deviation)
4 upper_bound = prediction + (mae + scale * deviation)
```

scale = 1.96 (often)

- ▶ A more robust approach (still easy to implements)

```
1 cv = cross_val_score(model, series[window:], target[window:],
2 scoring="neg_mean_absolute_error")
3 mae = cv.mean() * (-1)
4 deviation = cv.std()
5 lower_bound = prediction - (mae + scale * deviation)
6 upper_bound = prediction + (mae + scale * deviation)
```

scale = 1.96 (often)

Expected results

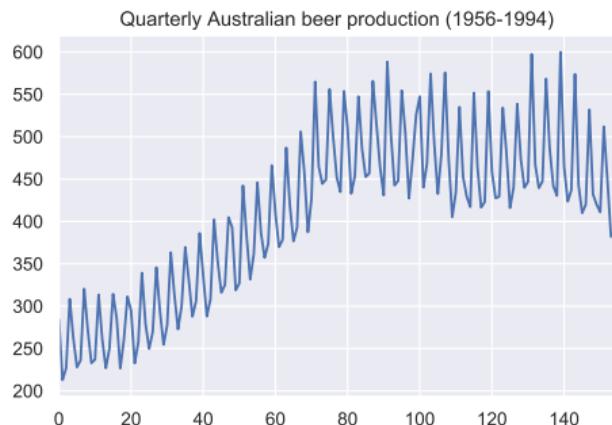


Anomalies are prediction outside the confidence bounds

PITFALLS IN FORECASTING

Case 1: trend change

Difficult case: what happen when the test distribution diverges from the training one?



Predicting on long term basis with
SARIMA:
trend + season

Australian beer production: typical series
with abrupt change

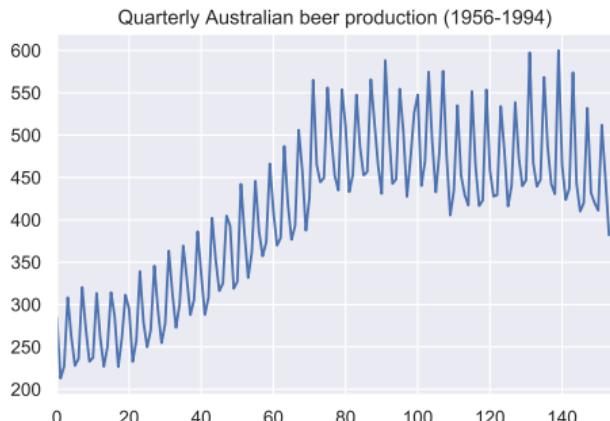


Time Series Nested Cross-Validation, Courtney Cochrane

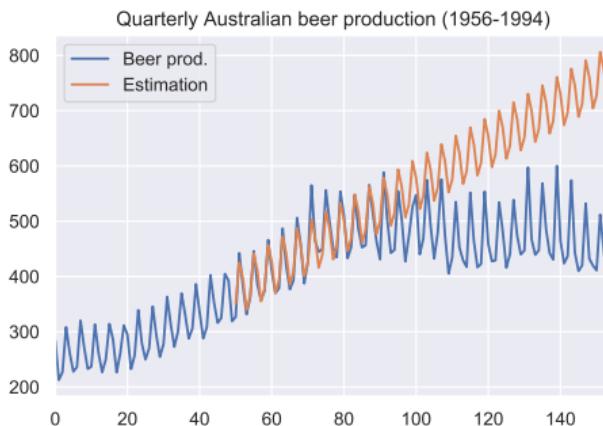
<https://towardsdatascience.com/time-series-nested-cross-validation-76adba623eb9>

Case 1: trend change

Difficult case: what happen when the test distribution diverges from the training one?



Australian beer production: typical series with **abrupt change**



Predicting on long term basis with SARIMA:
trend + season



Time Series Nested Cross-Validation, Courtney Cochrane

<https://towardsdatascience.com/time-series-nested-cross-validation-76adba623eb9>

Case 1: Detection & Resolution

- + A good example: trend may change abruptly in many situations
 - Who has ever used the same (SARIMA) model 10 years long without:
 - ▶ retraining
 - ▶ feeding the model with real data regularly

Detection

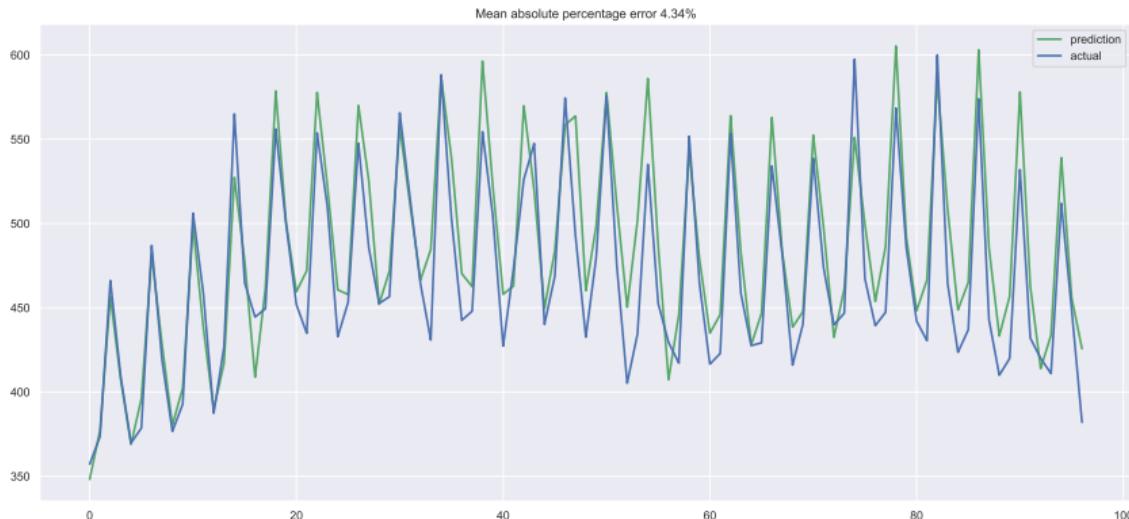
- ▶ Human monitoring
 - ▶ Anomaly detection
 - ▶ Change detection

Resolution

- ▶ Re-train models regularly
 - ▶ Evaluate your model on recent Data (even in production)
 - ▶ Perform anomaly detection, raise alarms

Case 1: getting closer to real case

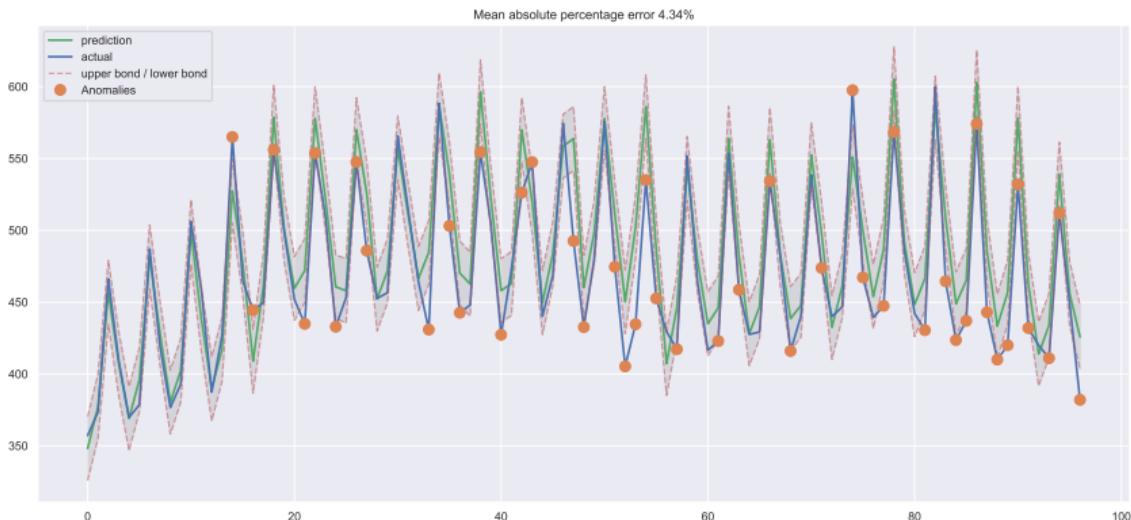
- ▶ Keep the same model on a long term basis...
 - ▶ ... But feed it with new data at each time step



Difficult to estimate the quality of the prediction

Case 1: getting closer to real case

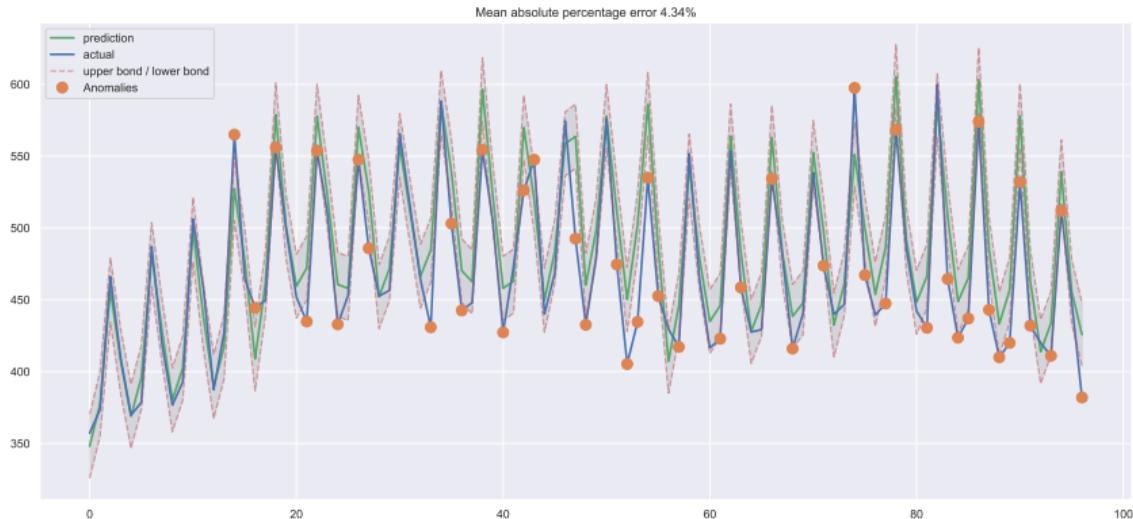
- ▶ Keep the same model on a long term basis...
- ▶ ... But feed it with new data at each time step



⇒ Adding confidence bounds & alarm detection

Case 1: getting closer to real case

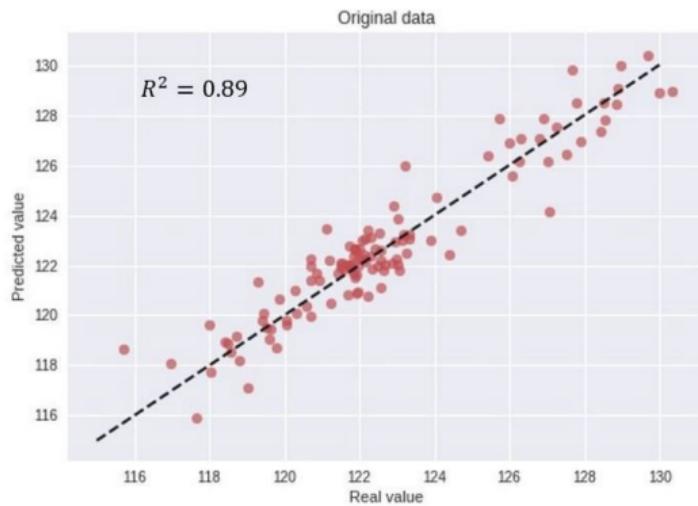
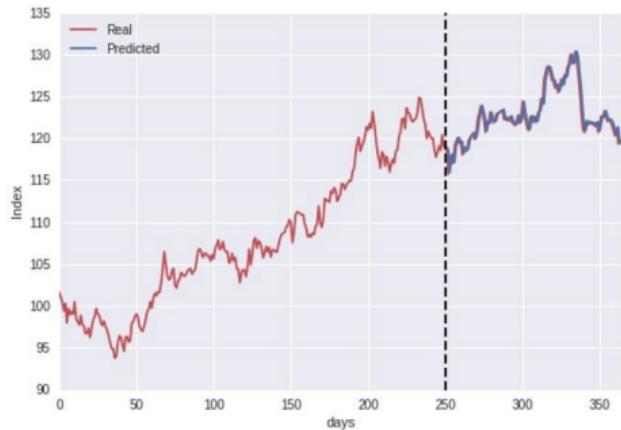
- ▶ Keep the same model on a long term basis...
- ▶ ... But feed it with new data at each time step



Alternative: Monitoring the results of a strong baseline

⇒ Results will converge between advanced model & baseline

Case 2 : a pretty good forecast



When you look at it from afar



How (not) to use Machine Learning for time series forecasting: Avoiding the pitfalls, Vegard Flovik
<https://towardsdatascience.com/how-not-to-use-machine-learning-for-time-series-forecasting-avoiding-the-pitfalls-10f3a2a2a2d1>

Case 2 : a pretty good forecast

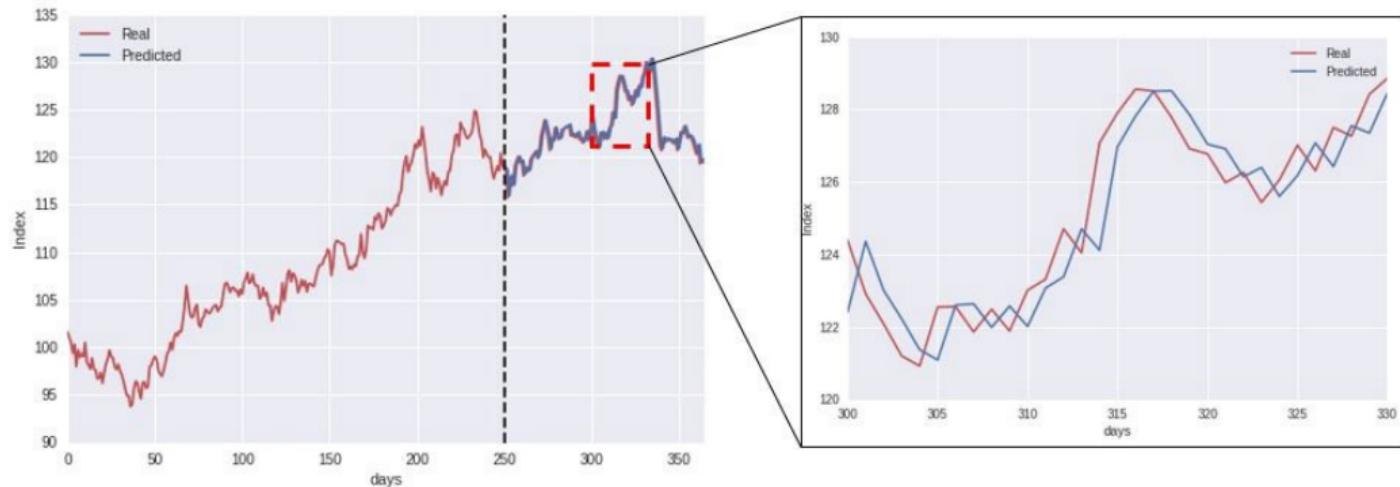
But actually:

- Data are generated from a random walk...
⇒ it can't be predicted !



How (not) to use Machine Learning for time series forecasting: Avoiding the pitfalls, Vegard Flovik
<https://towardsdatascience.com/how-not-to-use-machine-learning-for-time-series-forecasting-avoiding-the-pitfalls-10f3a2a2a2d>

Case 2 : look at the details

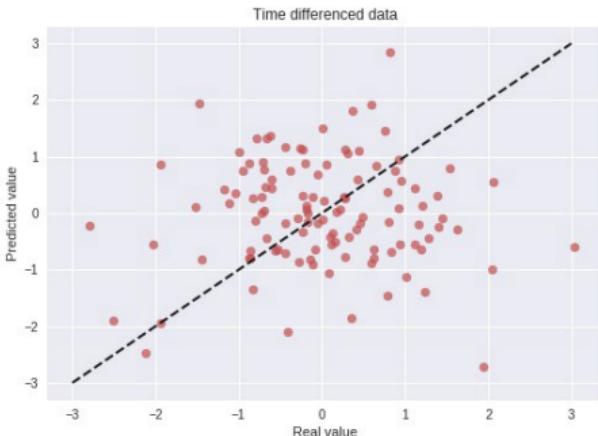
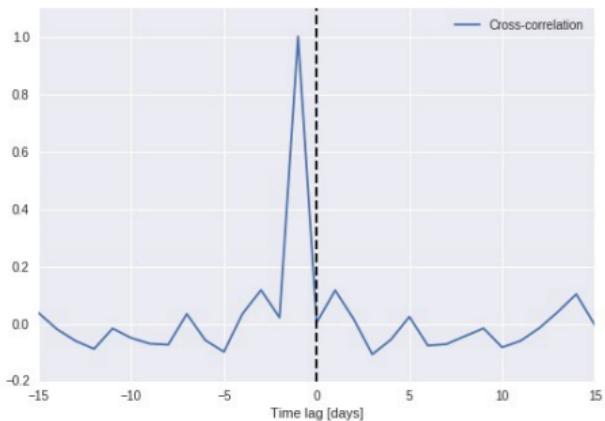
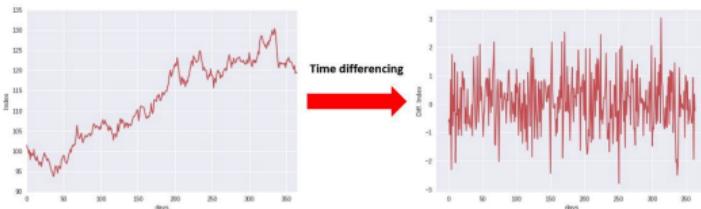


A very basic solution: $\hat{y}_t = y_{t-1}$

Case 2 : what was wrong? How to detect this case?

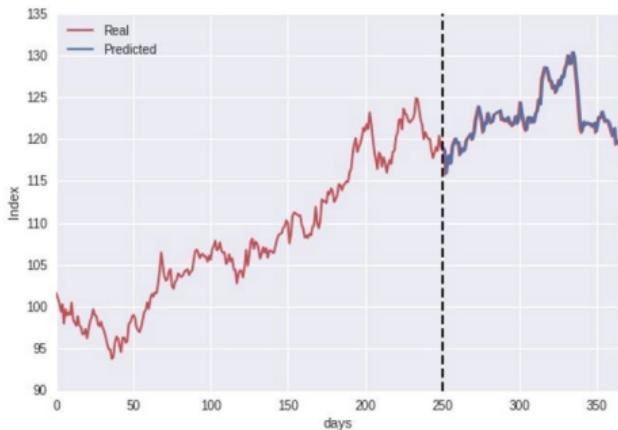
[Statistics]

- ▶ Do not use ARMA on non stationary signals
 - ▶ Measure cross-correlation between y and \hat{y} .



Case 2 : what was wrong? How to detect this case? [Computer Sc.]

Always compare a predictor to a strong baseline



- ▶ Choose the good baselines
 - ▶ Linear? \Rightarrow not sufficient
 - ▶ Time series \Rightarrow at least
 - ▶ Previous value: $\hat{y}_t = y_{t-1}$
 - ▶ Moving average: $\hat{y}_t = \frac{1}{T} \sum_i y_{t-i}$

- 1 Compare the results in CV...
- 2 And the standard deviation on the results.

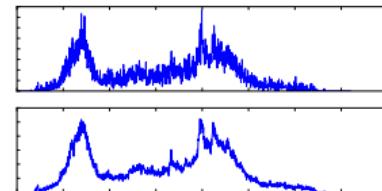
Baselines: a more comprehensive list

General baseline for time series:

- ▶ Previous value: $\hat{y}_t = y_{t-1}$
- ▶ Moving average: $\hat{y}_t = \frac{1}{T} \sum_i y_{t-i}$
- ▶ Predict always the most frequent value (rarely efficient on time series)

Seasonal data:

- ▶ Take one averaged season
 - ▶ e.g. in public transportation:
average weekday or average Monday
= prediction for every Monday
 - ▶ Strong baseline on specific dataset
 - ▶ **Very** strong baseline beyond $T + 1$



One Wednesday / averaged Wednesday on a particular subway station

ML classical baseline

- ▶ Linear model
- ▶ K-nn : all $y_{t-1} = \alpha$ lead to $y_t = \beta$

Case 3 : In-sample evaluation

Classical block cross-validation = seeing the future

- ▶ Cheating on any unpredictable (/ not modeled) **trends**
 - ▶ Difficult **prediction problem** turns into a simpler **interpolation problem**
- ▶ Side effect at the beginning/end of each test split (cf next slide)
- ▶ Also an issue for **normalization**

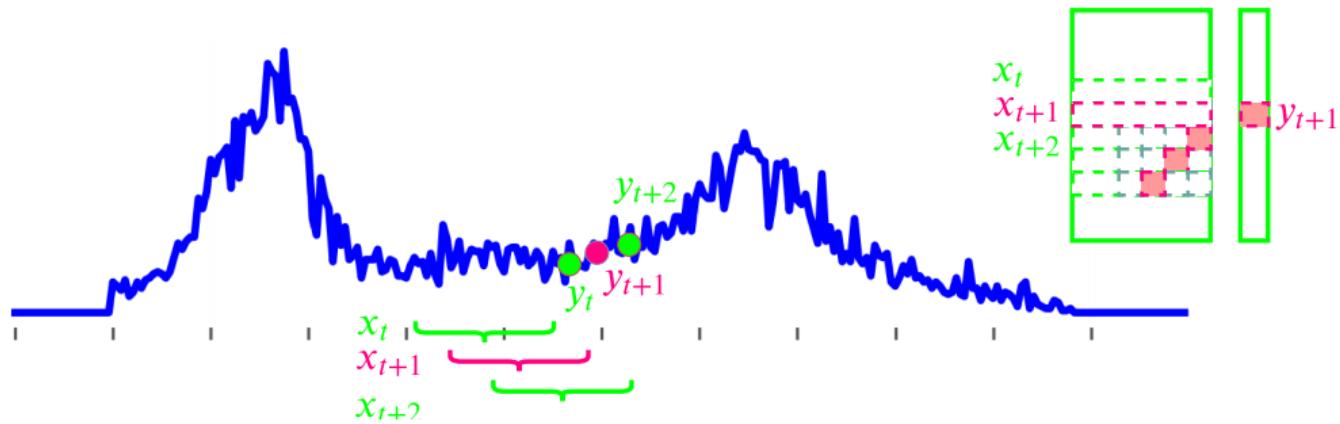


Leonard J. Tashman, Int. Jour. of Forecasting, 2000

Out-of-sample tests of forecasting accuracy: an analysis and review

Case 3 : In-sample evaluation & information leak

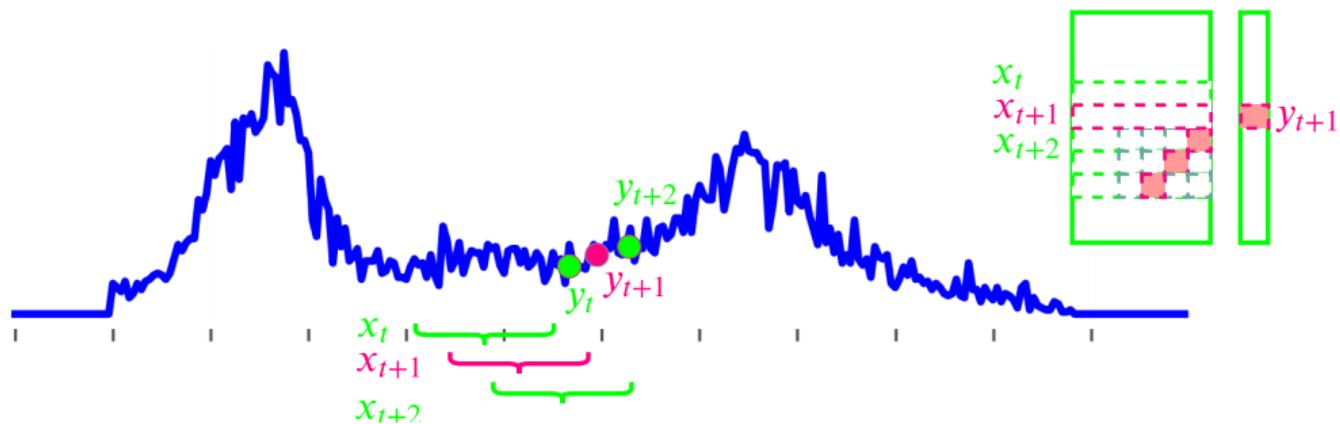
Shuffled cross validation: test samples are surrounded by training ones



- ▶ Target x_{t+1} : find the most correlated points $\{x_c\}$ in the training set
 - ▶ Add criterion to detect the better one regarding temporal evolution
 - ▶ Predict $x_c^*[end]$
- ⇒ it can even work to predict at $T + N$!

Case 3 : In-sample evaluation & information leak

Shuffled cross validation: test samples are surrounded by training ones



The only thing evaluated in this procedure is:

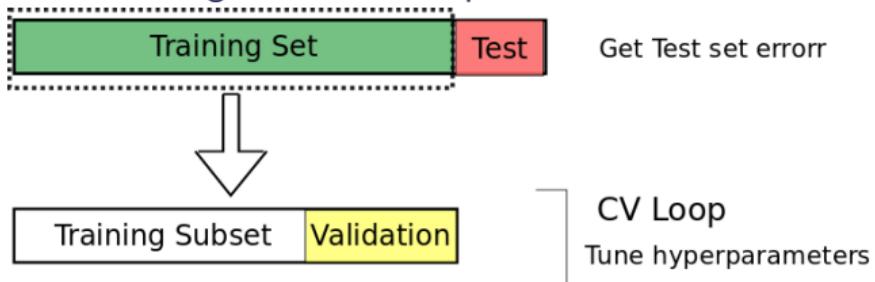
Is y_{t+1} close to y_t most of the time?
... And it is often the case !

⇒ You may obtain a prediction accuracy better than the intrinsic noise level!

Case 3 : In-sample / Out-of-sample evaluation

[single series]

Switching to out-of-sample evaluation:



Note: to optimize your parameters, you have to use a validation set at the end of the training set.



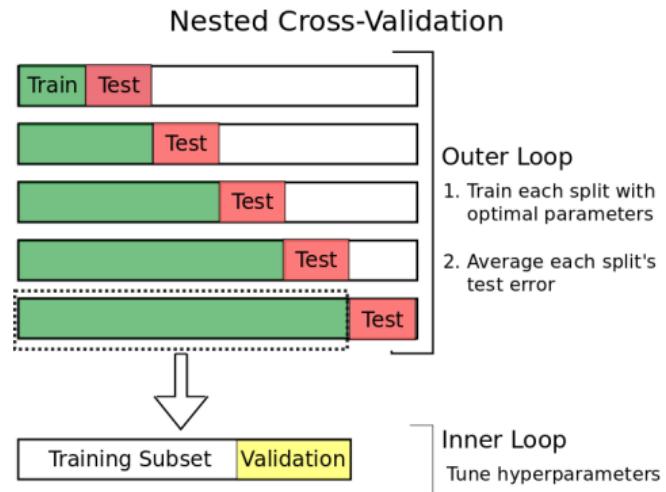
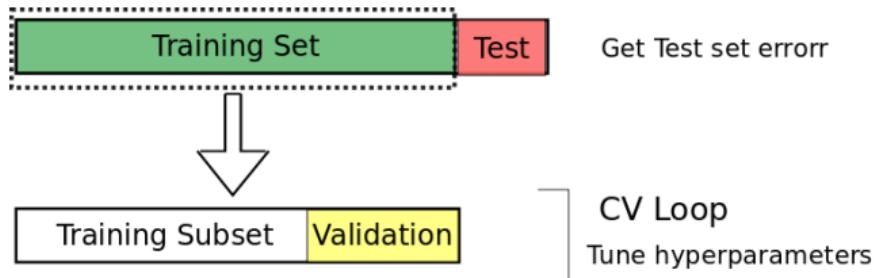
Leonard J. Tashman, Int. Jour. of Forecasting, 2000

Out-of-sample tests of forecasting accuracy: an analysis and review

Case 3 : In-sample / Out-of-sample evaluation

[single series]

Switching to out-of-sample evaluation:



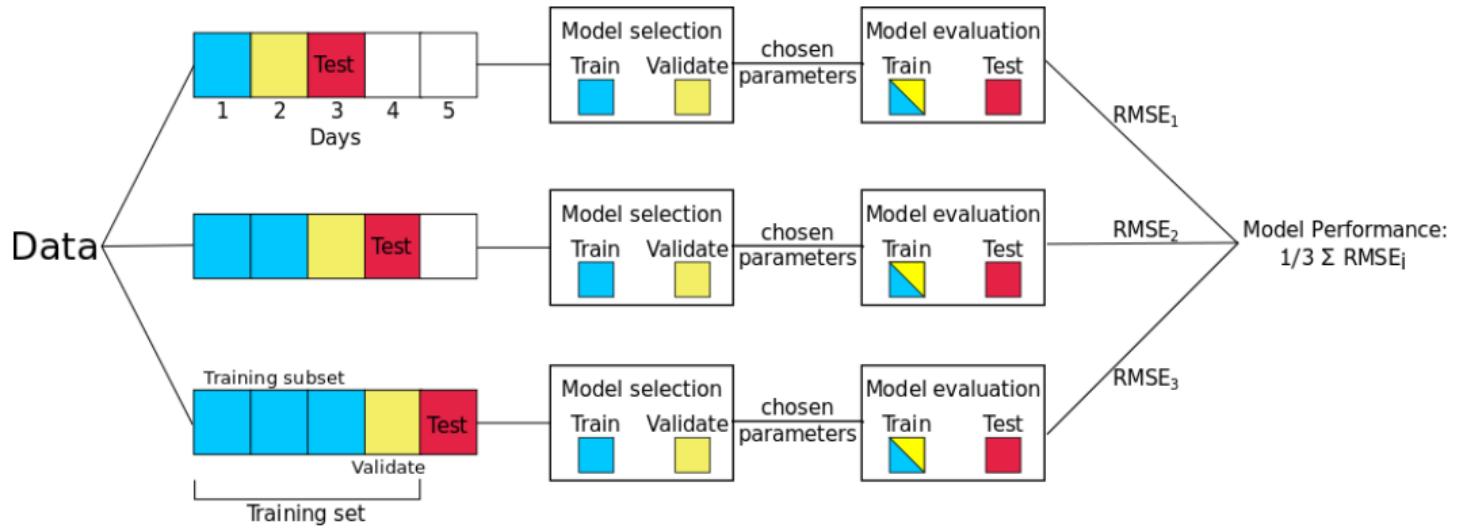
Note: to optimize your parameters, you have to use a validation set at the end of the training set.



Leonard J. Tashman, Int. Jour. of Forecasting, 2000

Out-of-sample tests of forecasting accuracy: an analysis and review

Case 3 : Complete Nested Cross-Validation procedure [single series]



Good news: it is already implemented in scikit-learn. ⇒ just use it!

```
from sklearn.model_selection import TimeSeriesSplit
```

Case 3 : Nested Cross Validation over a population

Interdependent samples

e.g. evolution of the temperature in multiple cities / sales in different shops

Independent samples

e.g. Patient response to a treatment

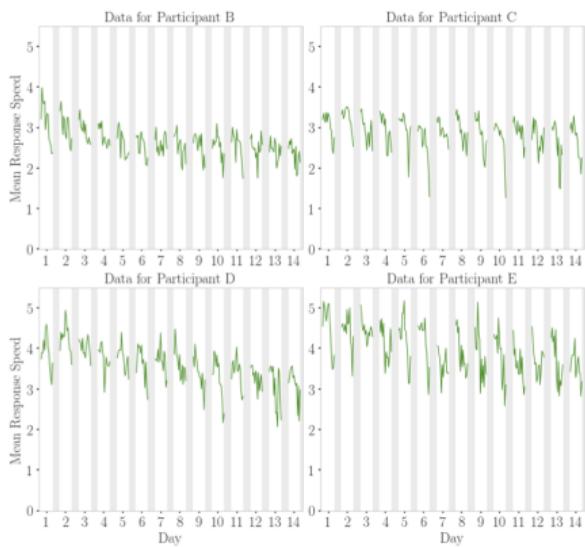
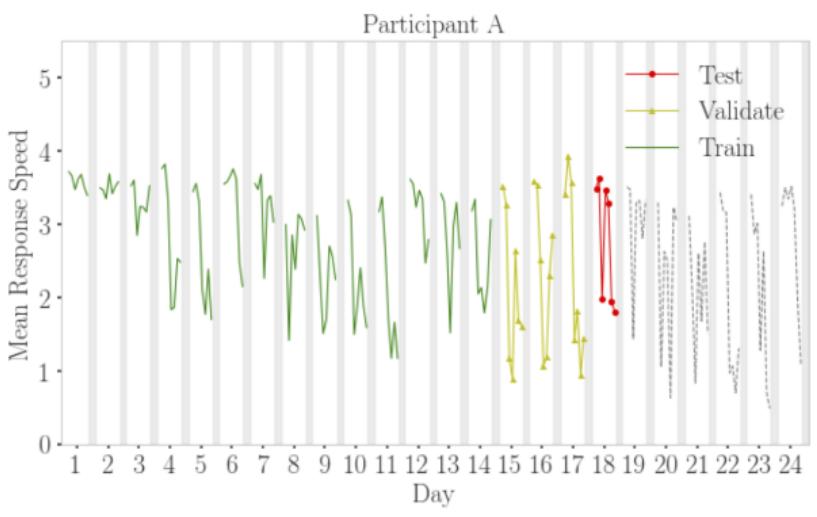
- ▶ Apply the Nested CV on all samples
- ▶ Make sure that Train/test frontier corresponds to an absolute time-stamp

Case 3 : Nested Cross Validation over a population

Interdependent samples

Independent samples

e.g. Patient response to a treatment



Case 3 : Detect temporal information leak

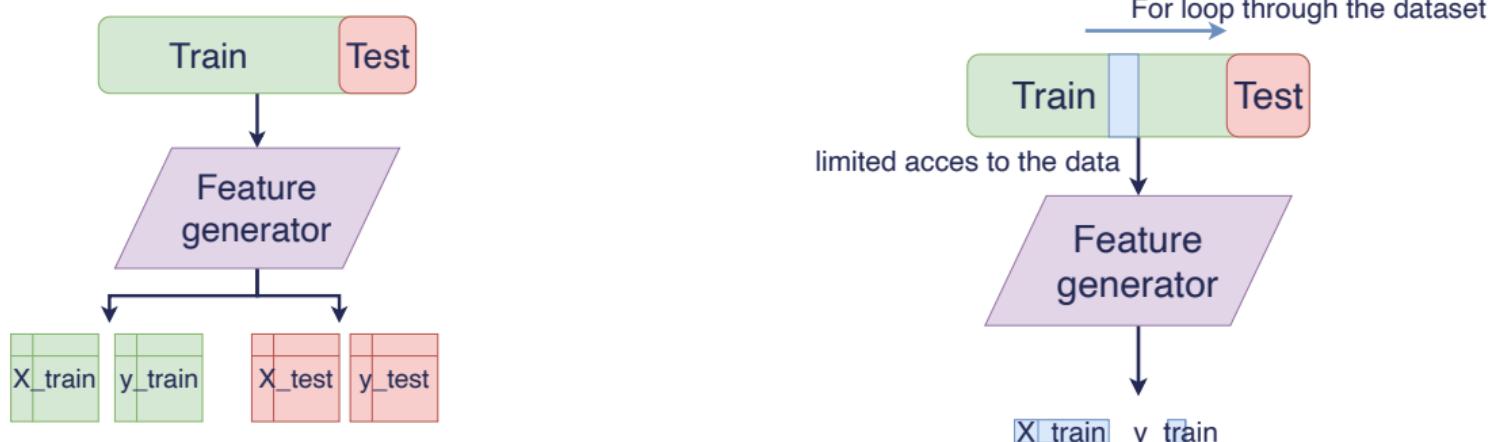
Nested cross validation **is supposed to** enable you to detect information leak...

One case remains critical

Creating a **leaking feature** may be difficult to detect.

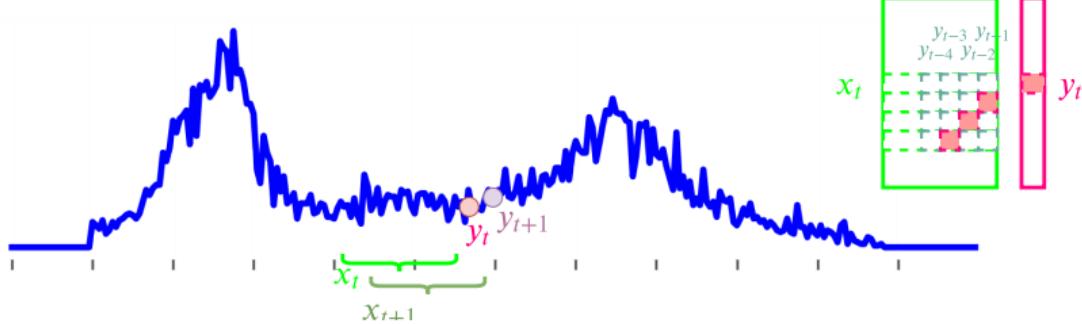
e.g.

- ▶ An aggregation of values overflowing in the future of the sample
- ▶ Uncontrolled feature (from tsfresh)



⇒ Even if we don't like for loop, even if it is less convenient...

Case 4 : Normalization of the lag variables (not the contextual ones)



As in any machine learning use case, dealing with time series requires normalization:

- ▶ **Normalization by columns**
 - ⇒ Great impact of future measurements on the data
 - ⇒ Destruction of the temporal dependencies
- ▶ **Normalization by line**
 - ▶ Impact of the future measurements on the data
 - ▶ ... But limited impact
 - ▶ Normalizing by the max
 - ▶ Even better: normalizing by the 99% percentile
 - ▶ Very stable information that could have been given by an expert

CONCLUSION

Specific issues in time series

- ▶ Noise level
 - ▶ Discuss with expert
 - ▶ Filter
- ▶ Sampling
 - ▶ Subsampling (at least at the beginning, to reduce computations)
 - ▶ Average to reduce the noise (day \Rightarrow week \Rightarrow month)
 - ▶ How to deal with irregular Sampling?
 - ▶ Subsample to a regular & available rate
 - ▶ Fill missing values (EM / interpolation)

Statistician & computer scientist use different tools: make sure to exploit the best of each community proposals

Ideal combination:

(1) Naive baseline

(2) Prophet

(3) Xgboost on engineered features