TIME-SERIES ANALYSIS & DEEP-LEARNING

June 17^{th}, 2022

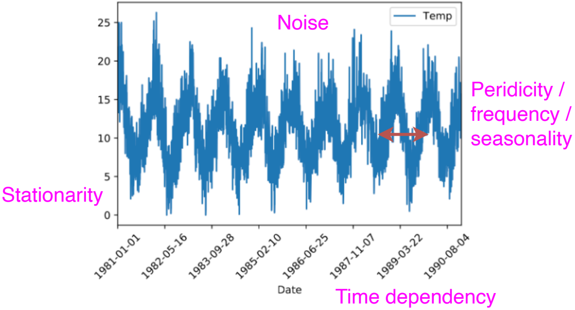
Vincent Guigue

Introduction

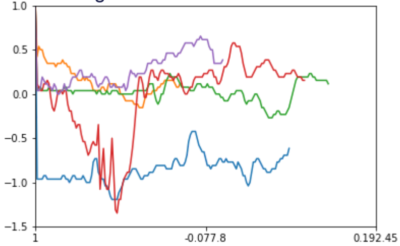
General schedule

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

Single instance

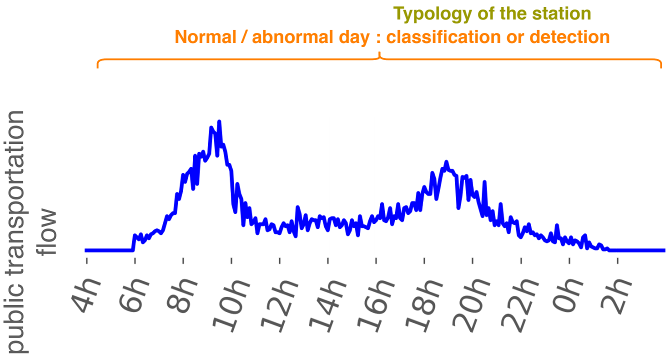


Variable length



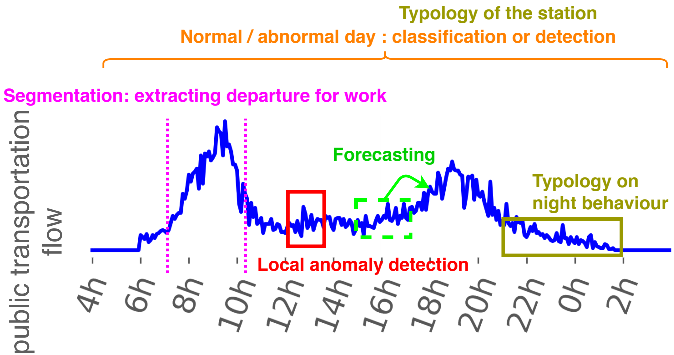
General schedule

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



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



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

Data Science

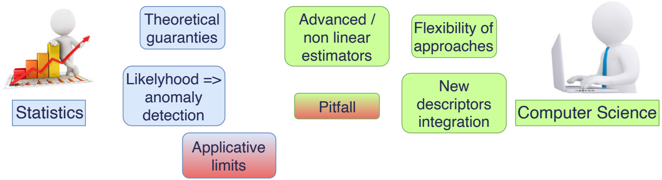
Statistics

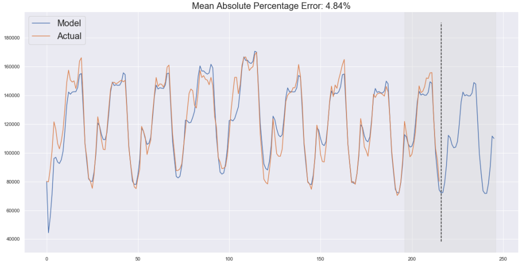
Computer Science

Sales history

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





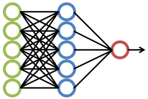
General schedule

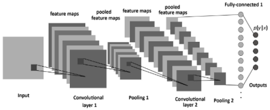
1. Signal specificities
2. Very different applications
3. Different points of view
4. Benefits/pitfalls
   of ML approaches
5. Benefits of deep learning

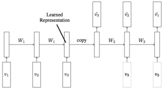
General schedule

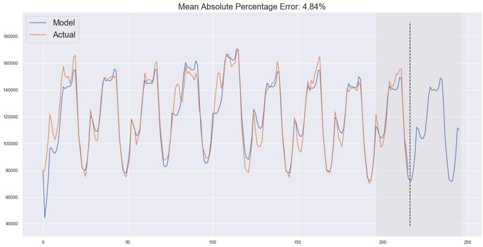
1. Signal specificities
2. Very different applications
3. Different points of view
4. Benefits/pitfalls of ML approaches
5. Benefits of deep learning

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



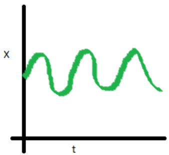




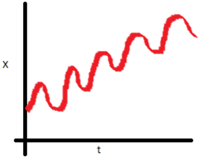


Time series = specific object

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

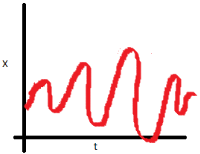


Stationary series



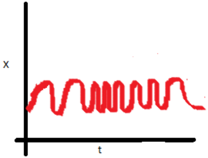
Non-Stationary series

Variable mean



Non-Stationary series

Variable std. dev.



Non-Stationary series

Variable covariance

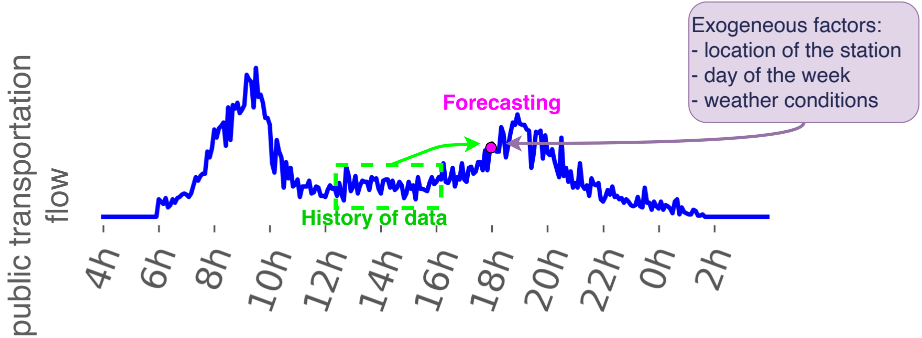
- Number of instances
  - May be one !

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

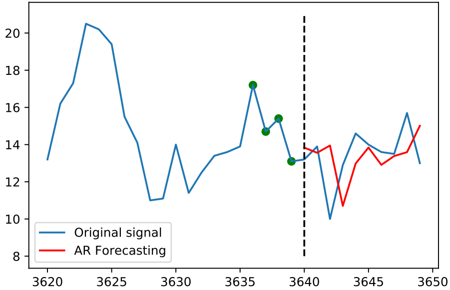
AR/ARMA

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

AR : Auto-Regressive modeling :

$$Y_t = \alpha + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \ldots + \alpha_p Y_{t-p} + \varepsilon_1 \quad (1)$$

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

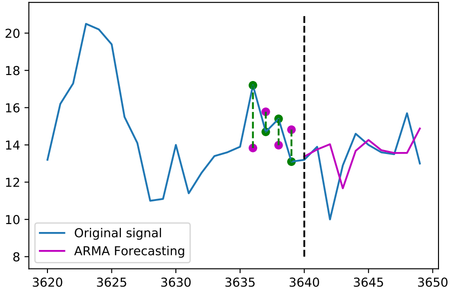
The historical answer to time-series forecasting (from statiticians)

AR : Auto-Regressive modeling :

$$Y_t = \alpha + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \ldots + \alpha_p Y_{t-p} + \varepsilon_1 \quad (1)$$

ARMA : Auto-Regressive Moving Average modeling :

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

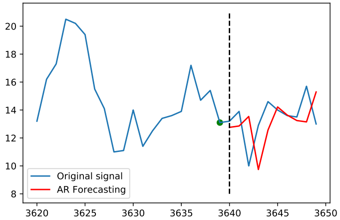


Complete Guide to Time Series Forecasting in Python, https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/

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

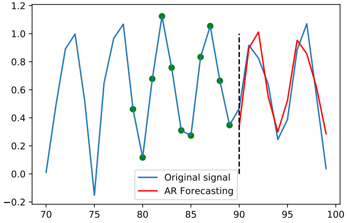
AR basic exemples & Periodicity discussion

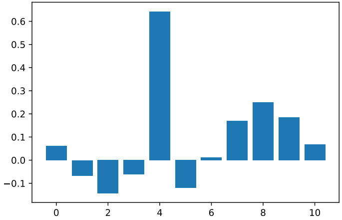
AR (order = 1) : a very intuitive model :



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

AR on periodic time series (order = 10) :





AR inference

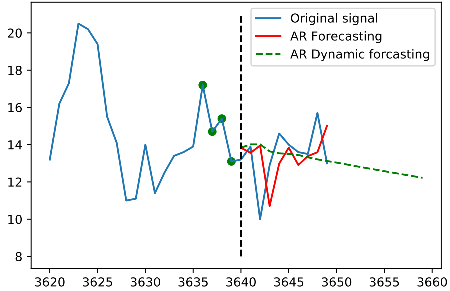
AR :

- Prediction at t :

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

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

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



ARMA :

- Prediction at t :

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

- 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) :

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

- 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

- Model selection : AR, ARMA, ARIMA
- Temporal window

Statistician

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

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

k = nb estimated parameters

- Maximizing likelyhood
- while penalizing model complexity

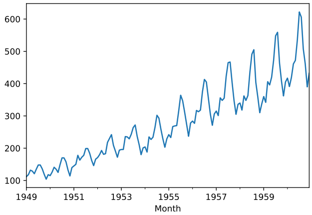
In practice : always very **low orders**

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



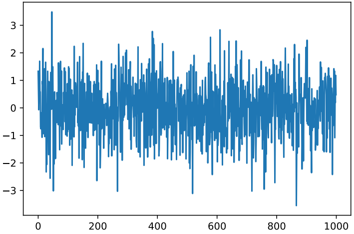
Hypothesis testing
(e.g. Dicky Fuller Test) :

```
Results of Dickey-Fuller Test:
Test Statistic                  0.815369
p-value                         0.991880
Critical Value (1%)            -3.482
Critical Value (5%)            -2.884
Critical Value (10%)           -2.579
```

⚠ 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) :

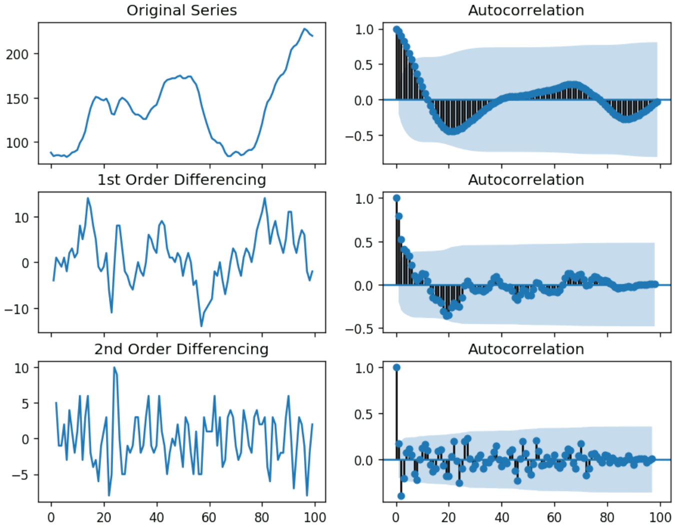
```
Results of Dickey-Fuller Test:
Test Statistic                -31.448939
p-value                         0.000000
Critical Value (1%)            -3.436
Critical Value (5%)            -2.864
Critical Value (10%)           -2.568
```

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

Improving stationarity = signal differencing

Differencing the signal :

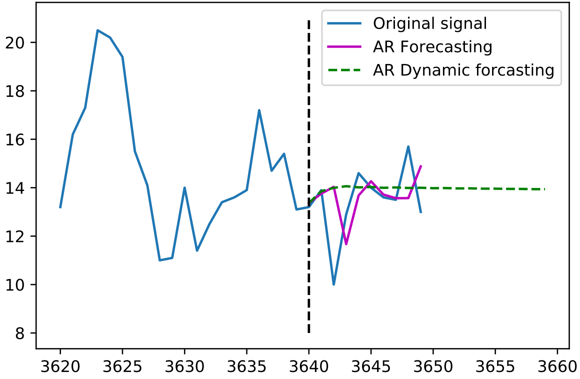
Order 1 :
$\delta_t = y_t - y_{t-1}$ instead of y_t
Order 2 :
$\delta_t^{(2)} = \delta_t - \delta_{t-1}$ instead of y_t



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

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

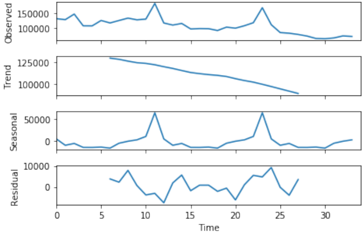
   - Multiplicative model

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

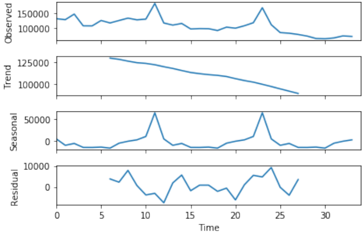
3. ARMA is performed on the residue

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

Additive model :



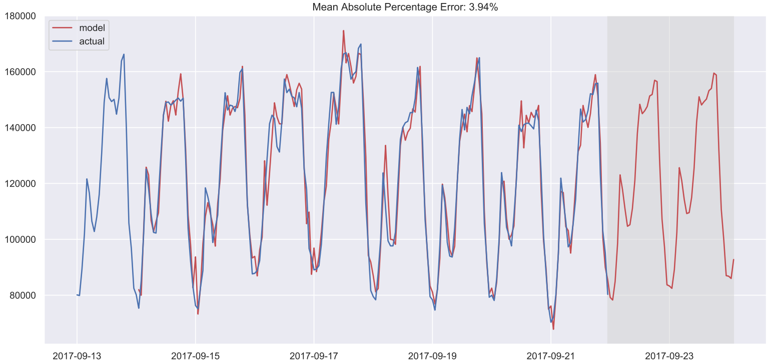
Multiplicative model :



SARIMA

Extension : Seasonality ARIMA

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



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 :

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

AR + exogenous factors $e_1, e_2, \ldots$:

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

$\Rightarrow$ 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 :

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

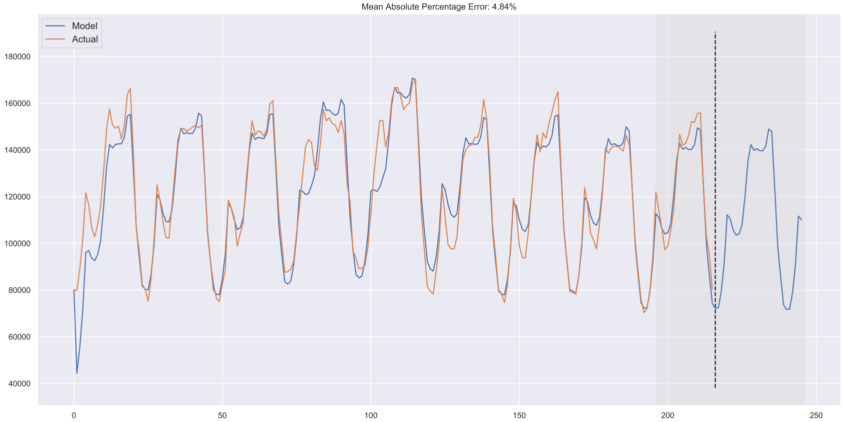
- Seasonality triple exponential filtering (=Holt-Winters)
  Prediction at horizon m, season $= s$, season length$=L$

$$\begin{aligned}
\text{1st order recursive model} : \ell_t &= \alpha\,(y_t - s_{t-L}) + (1-\alpha)\,(\ell_{t-1} + b_{t-1}) \\
\text{Differencing} : b_t &= \beta\,(\ell_t - \ell_{t-1}) + (1-\beta) b_{t-1} \\
\text{Seasonality} : s_t &= \gamma\,(y_t - \ell_t) + (1-\gamma) s_{t-L} \\
\text{Combination} : \hat{y}_{t+m} &= \ell_t + m b_t + s_{t-L+1+(m-1)\%L}
\end{aligned}$$

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

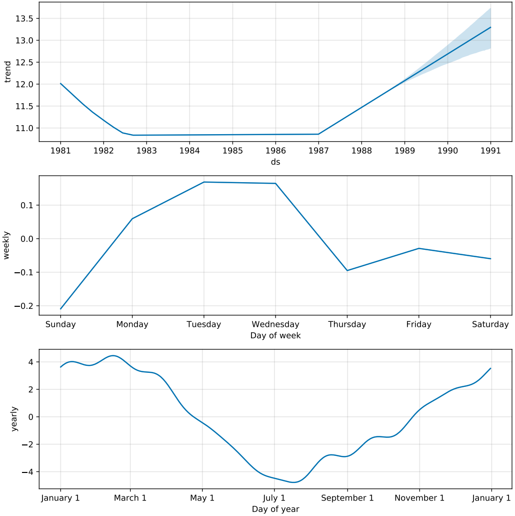
General formulation (Additive model) :

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

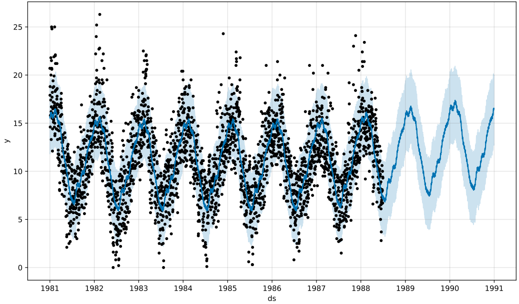
- $g(t)$: trends (for non periodic changes)
- $s(t)$: seasonality. In fact seasonality is multi-scale :
  - $s_h(t)$ hour, $s_d(t)$ day, $s_w(t)$ week, $s_m(t)$ month
- $h(t)$: holidays = prophet denomination for exogenous factors
- $e(t)$: residue

$\Rightarrow$ from statistics... Prophet $\approx$ SARIMA

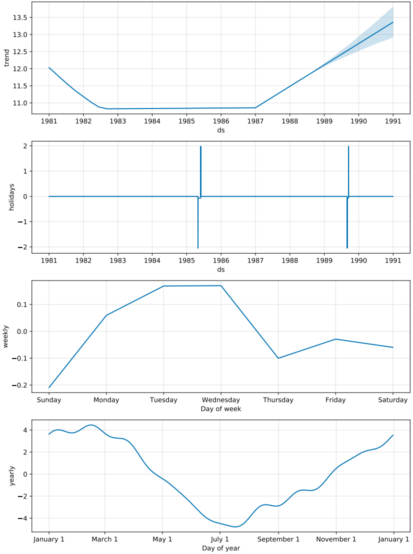
Prophet output



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.

Prophet *vs* SARIMA

Prophet = a statistician tool in a computer science package

- more efficient (faster)
- more convenient ((almost) no parameter to set)
  - great integration with `pandas`
  - auto seasonality determination (relying on the calendar)
  - obvious counterpart : pandas is required
- better ML integration (scikit-learn / cross validation)

⇒ The statistical baseline to challenge ML approaches

Machine Learning Forecasting

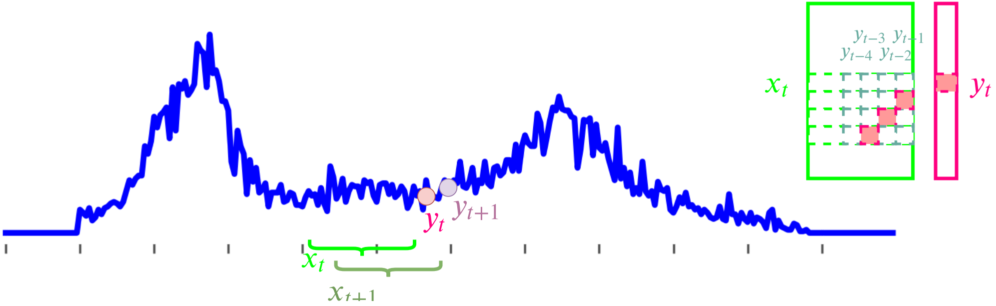
From AR to standard ML chains

- 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

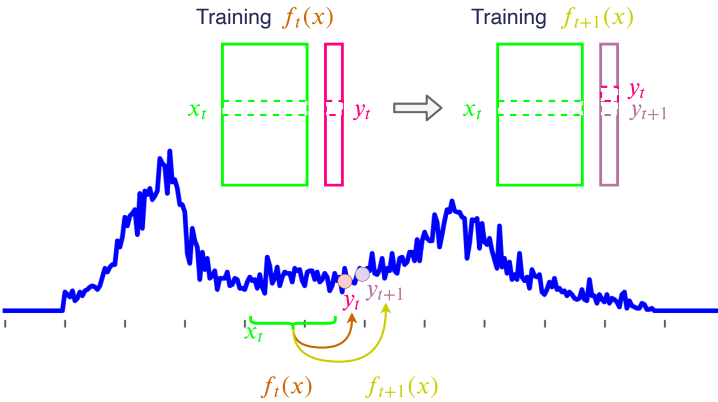
Introducing AR features in an ML environment :



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

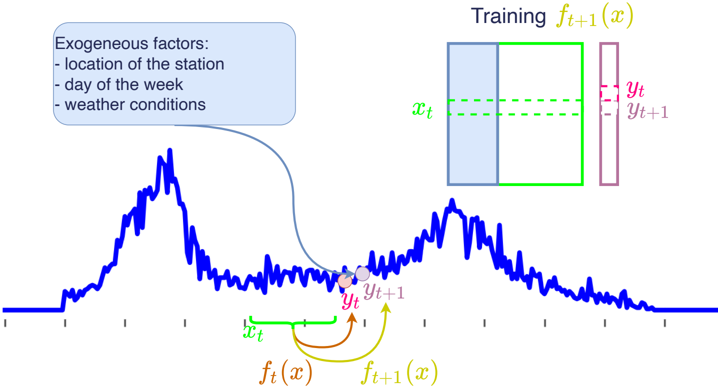
Predict further in time

- 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
  - Intrinsically compatible with exogenous variables



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
  - Intrinsically compatible with exogenous variables

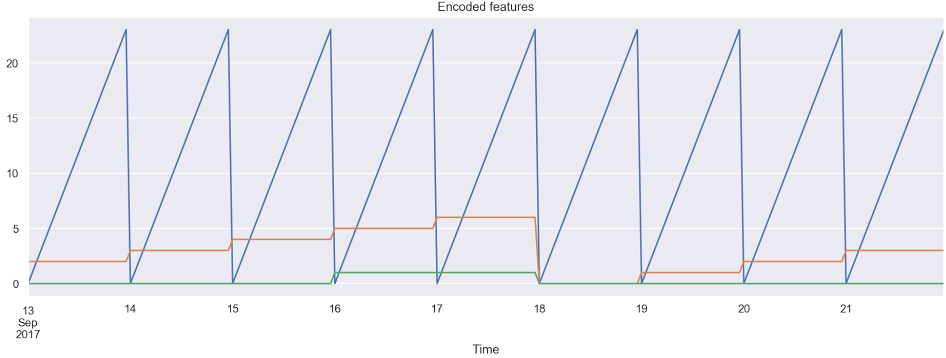


Exogenous factors / Feature engineering

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

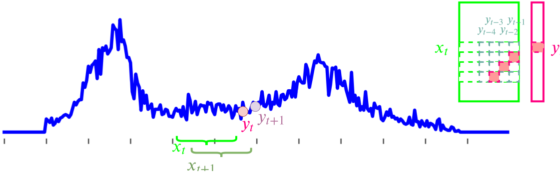
- Another way to encode seasonality
- Using pandas to catch up prophet functions
  - Types of days
  - Hours...

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
- Classical feature engineering
  - Feature clustering, ...
- Target encoding
  - e.g. Average value on Monday / weekday / ...



⇒ `pandas` is required for rolling, date reading & target encoding...
`pandas` is essential for timeseries !

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
- Classical feature engineering
  - Feature clustering, ...
- Target encoding
  - e.g. Average value on Monday / weekday / ...

`abs_energy` (x)	Returns the absolute energy of the time se
`absolute_sum_of_changes` (x)	Returns the sum over the absolute value of
`agg_autocorrelation` (x, param)	Calculates the value of an aggregation func
`agg_linear_trend` (x, param)	Calculates a linear least-squares regression
`approximate_entropy` (x, m, r)	Implements a vectorized Approximate entr
`ar_coefficient` (x, param)	This feature calculator fits the unconditiona
`augmented_dickey_fuller` (x, param)	The Augmented Dickey-Fuller test is a hyp
`autocorrelation` (x, lag)	Calculates the autocorrelation of the specif
`binned_entropy` (x, max_bins)	First bins the values of x into max_bins equ
`c3` (x, lag)	This function calculates the value of
`change_quantiles` (x, ql, qh, isabs, f_agg)	First fixes a corridor given by the quantiles
`cid_ce` (x, normalize)	This function calculator is an estimate for a

⇒ `pandas` is required for rolling, date reading & target encoding...
`pandas` is essential for timeseries !

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

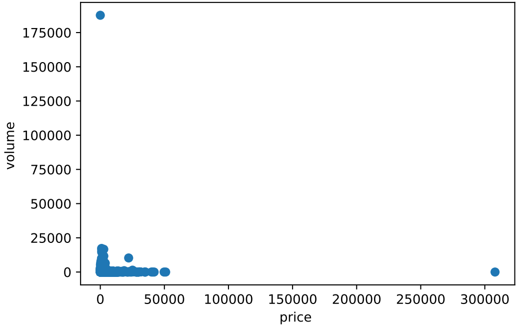
$\Rightarrow$ Exploit ML plasticity $\Leftrightarrow$ hard to model with AR

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, ...

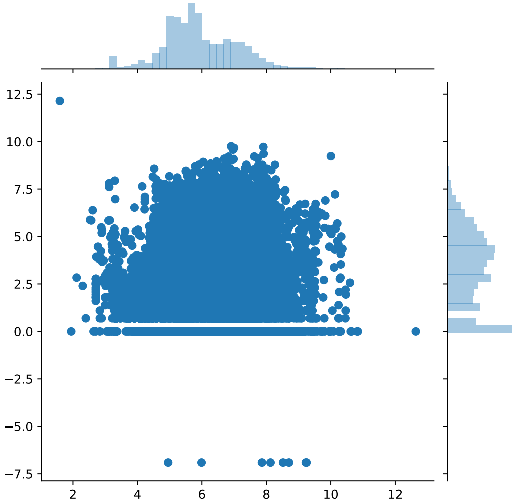
Items prices & volumes



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

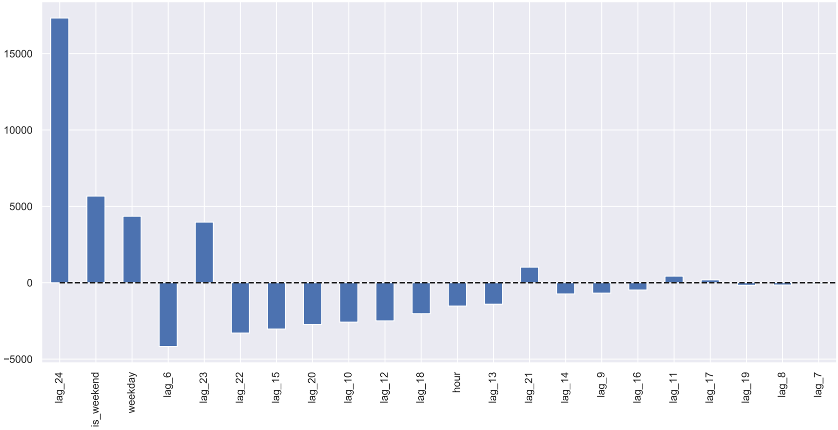
- Feature engineering makes monovariate prediction very strong
- Deep learning (try to) tackles multivariate prediction
  - To extract relevant feature automatically
  - To find fine correlation between shop/item dynamics

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

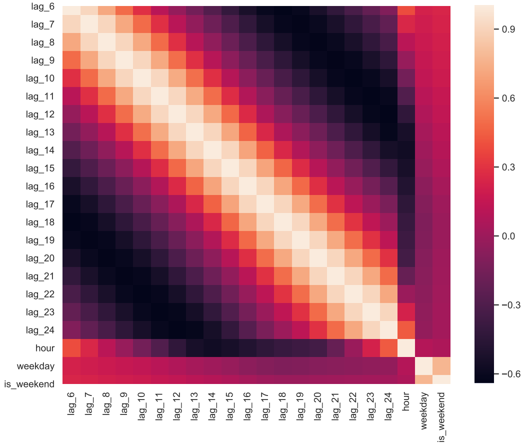
Variable weights normalization & interpretations

- 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

- Unmatched performance/development-time ratio
  - Surprising resistance to overfitting
  - Efficiency of the default setting
- 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-impor`

Metrics & Anomalies

Metrics – How to evaluate if our predictions are relevant ?

R squared

s_y^2 = Empirical variance of Y :

$$s_y^2 = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2 + \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

s_y^2 = explained variance + residual variance

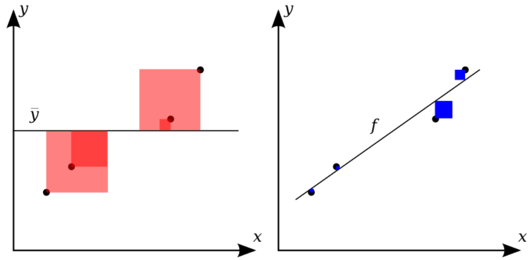
$$R^2 = \frac{\sum_i(\hat{y}_i - \overline{y})^2}{\sum_i(y_i - \hat{y}_i)^2} = \frac{\text{explained variance}}{\text{residual variance}}$$

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

Classical metrics

- Mean Absolute error

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

- Median Absolute Error (robust to outliers)

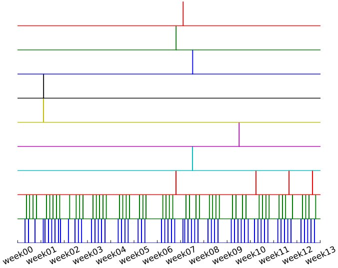
$$MedAE = \text{median}(|y_1 - \hat{y}_1|, \ldots, |y_n - \hat{y}_n|)$$

- 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 :

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

Solutions :

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

$$\mathrm{SMAPE}_1 = \frac{100\%}{n}\sum_{t=1}^{n}\frac{|y_t - \hat{y}_t|}{(|\hat{y}_t| + |y_t|)/2} \quad \text{or} \quad \mathrm{SMAPE}_2 = \frac{\sum_{t=1}^{n}|y_t - \hat{y}_t|}{\sum_{t=1}^{n}(\hat{y}_t + y_t)}$$

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

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

$\Rightarrow$ we focus on distances

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

Anomaly detection : proposed implementation

- Computing bounds very easily :

```
mae = mean_absolute_error(series[window:], prediction[window:])
deviation = np.std(series[window:] - prediction[window:])
lower_bond = prediction - (mae + scale * deviation)
upper_bond = prediction + (mae + scale * deviation)
```

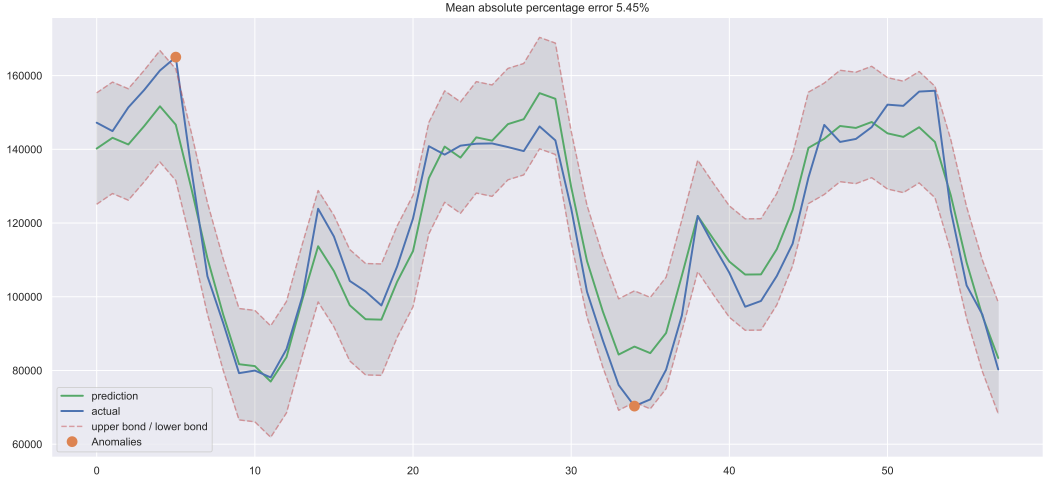
`scale = 1.96` (often)

- A more robust approach (still easy to implements)

```
cv = cross_val_score(model, series[window:], target[window:],
                     scoring="neg_mean_absolute_error")
mae = cv.mean() * (-1)
deviation = cv.std()
lower_bond = prediction - (mae + scale * deviation)
upper_bond = prediction + (mae + scale * deviation)
```

`scale = 1.96` (often)

Expected results

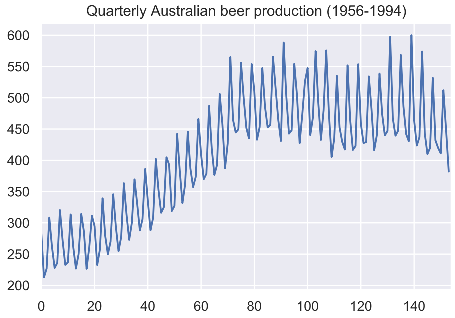


Anomalies are prediction outside the confidence bounds

Pitfalls in ML for time series

Case 1 : trend change

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



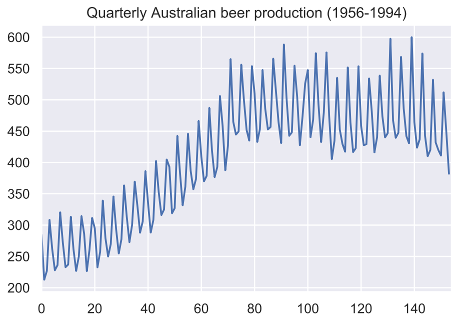
Predicting on long term basis with SARIMA :
trend + season

Autralian beer production : typical series with abrupt change

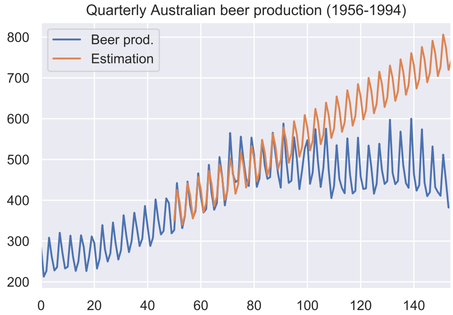
Time Series Nested Cross-Validation, Courtney Cochrane
https://towardsdatascience.com/time-series-nested-cross-validation-76adba623eb9

Case 1 : trend change

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



Autralian 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 use the same (SARIMA) model 10 years long without :
    - retraining
    - feeding the model with real data regularly
- A particular instance of iid hypothesis failure : very classical in ML

Detection

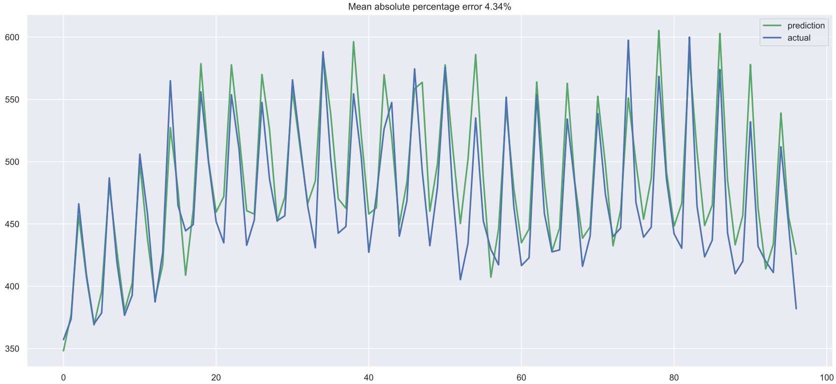
- Human monitoring
- Anomaly detection
- Change detection

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

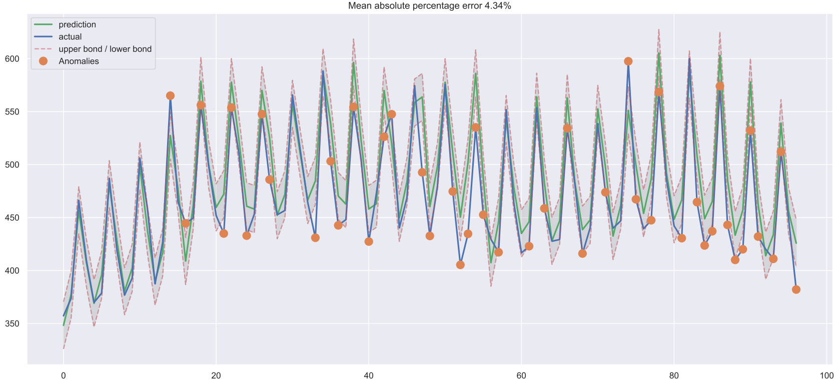
- Keep the same model on a long term basis...
- ... But feed it with new data at each time step



Difficult to estimate the quality of the prediction

Case 1 : getting closer to real case

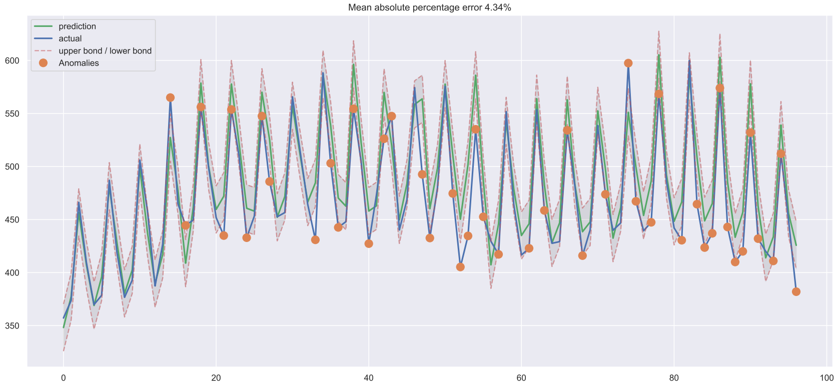
- Keep the same model on a long term basis...
- ... But feed it with new data at each time step



⇒ Adding confidence bounds & alarm detection

Case 1 : getting closer to real case

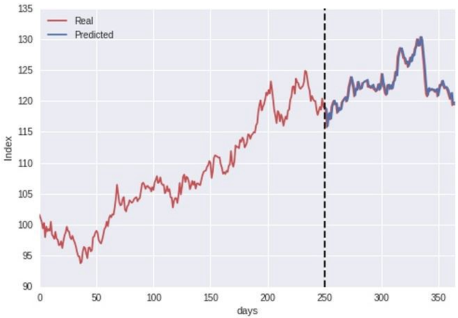
- Keep the same model on a long term basis...
- ... But feed it with new data at each time step

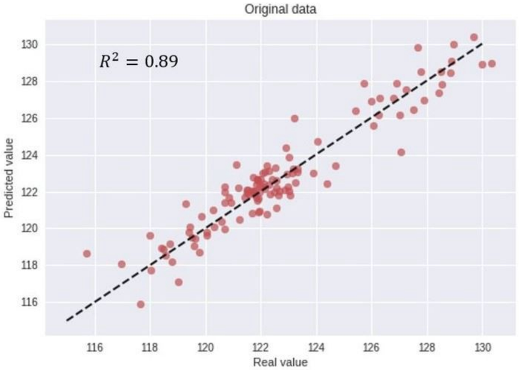


Alterative : Monitoring the results of a strong baseline

$\Rightarrow$ 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-avo

Case 2 : a pretty good forecast

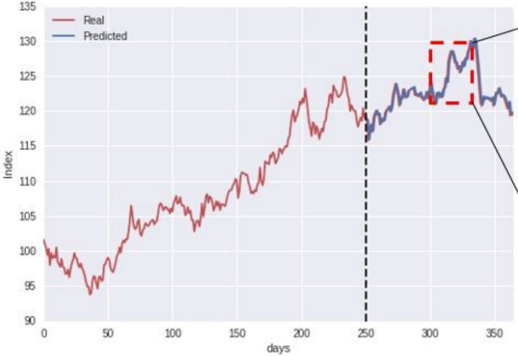
But actually :

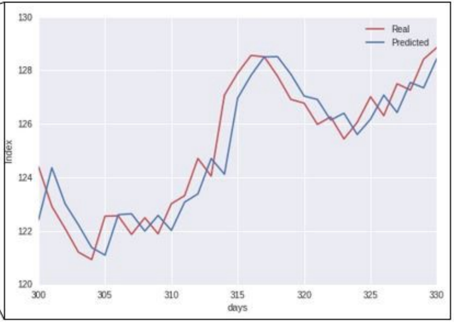
- Data are generated from a random walk...

⇒ 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-avo

Case 2 : look at the details

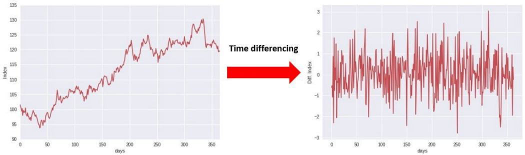


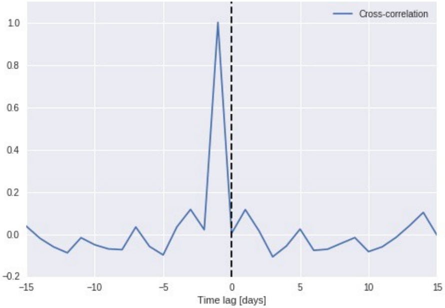


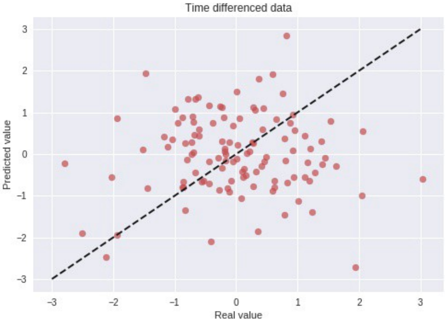
A very basic solution : $\hat{y}_t = y_{t-1}$

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 $\hat{y}$.

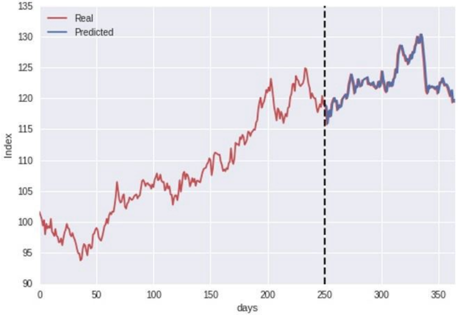






Case 2 : what was wrong ? How to detect this case ? [Computer science]

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.

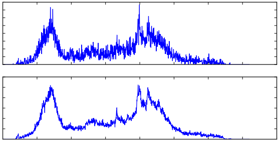
Baselines : a more comprehensive list

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

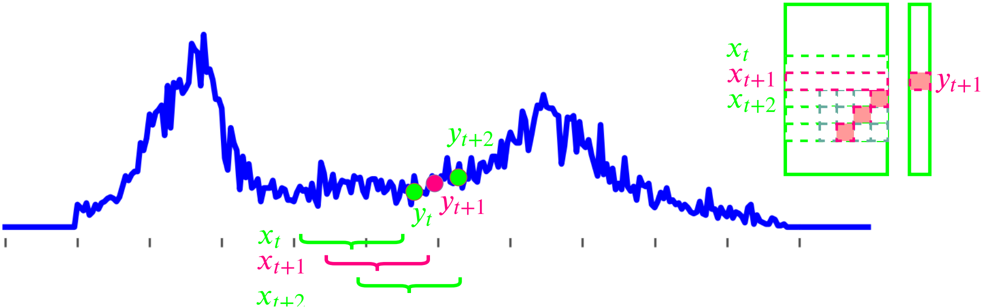
- cheating on any unpredictable (/ not modeled) trends
  - Difficult prediction problem turns into to a simpler interpolation problem
- side effect at the beginning/end of each test split (cf next slide)

Iteration 1	Test	Train	Train	Train	Train
Iteration 2	Train	Test	Train	Train	Train
Iteration 3	Train	Train	Test	Train	Train
Iteration 4	Train	Train	Train	Test	Train
Iteration 5	Train	Train	Train	Train	Test

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

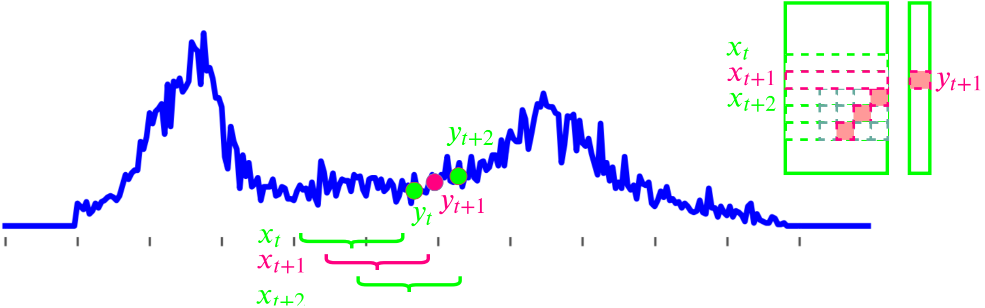


- Target x_{t+1} : find the most correlated points $\{x_c\}$ in the training set
- Add criterion to detect the better one regarding temporal evolution
- Predict $x_c^{\star}[end]$

$\Rightarrow$ 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 :

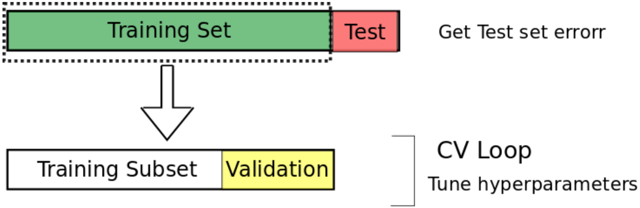
Is y_{t+1} close to y_t most of the time ?

... And it is often the case !

$\Rightarrow$ You may obtain a prediction accuracy better that the intrinsic noise level !

Case 3 : In-sample / Out-of-sample evaluation [single series]

Switching to out-of-sample evaluation :

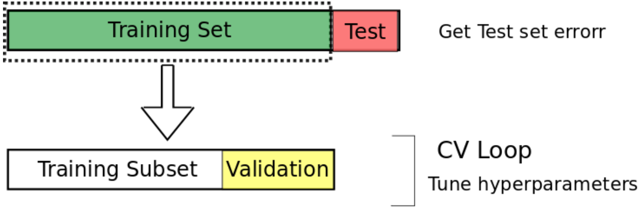


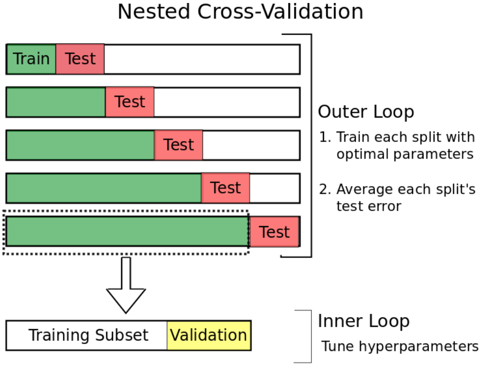
Note : to optimize your parameters, you have to use a validation set at the end of the training set.

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 :

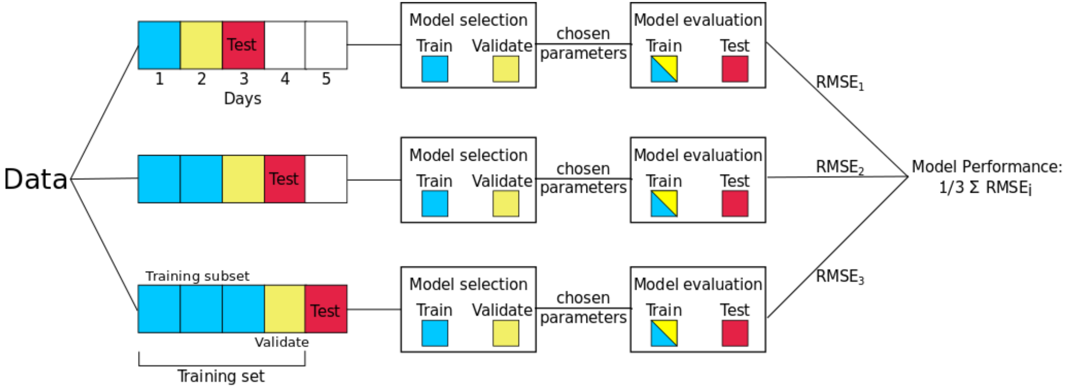




Note : to optimize your parameters, you have to use a validation set at the end of the training set.

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

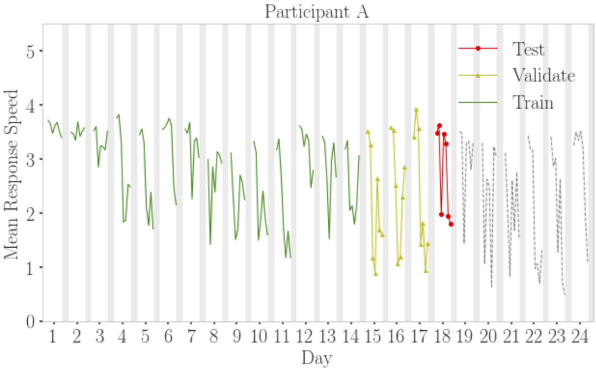
- Apply the Nested CV on all samples
- Make sure that Train/test frontier corresponds to an absolute time-stamp

Independent samples

e.g. Patient response to a treatment

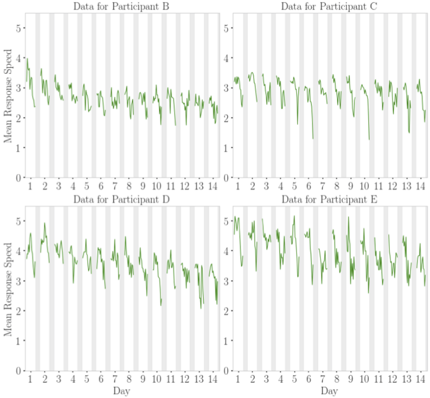
Case 3 : Nested Cross Validation over a population

Interdependent samples



Independent samples

e.g. Patient response to a treatment



Case 3 : Detect temporal information leak

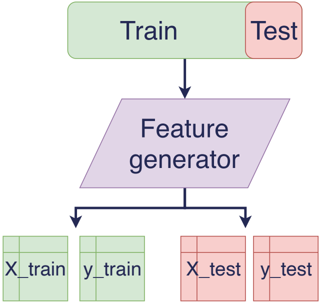
Nested cross validation **is supposed to** enable you to detect information leak...

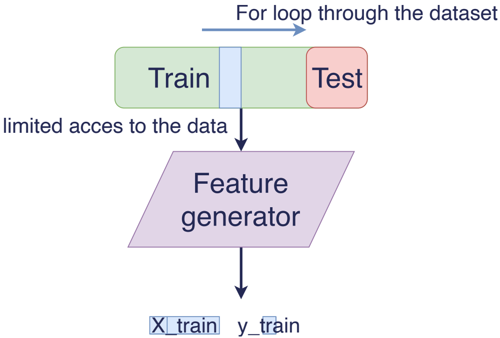
One case remains critical

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

e.g.

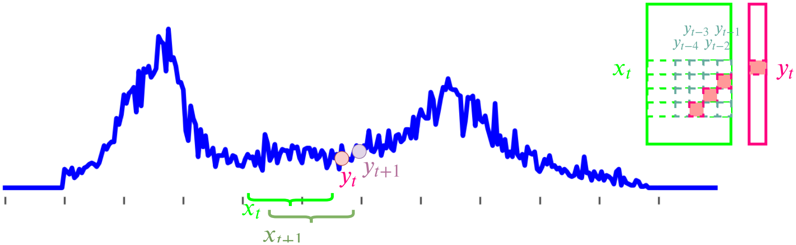
- An aggregation of values overflowing in the future of the sample
  - Target encoding
- Uncontrolled feature (from tsfresh)





⇒ 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

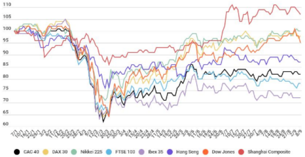
Conclusion

Important factors in time series prediction

- Analysing local dynamics = Auto Regressive models
  - $\Rightarrow$ Useful for close prediction
  - $\Rightarrow$ Not sufficient for mid/long term prediction
- Distinguish : past values ; trends & seasonality ; exogenous factors...
  - $\Rightarrow$ Impact of specific pattern ?
  - $\Rightarrow$ Bridge between time series prediction $\Leftrightarrow$ source separation
- Pitfalls are numerous...
  - $\Rightarrow$ Don't forget your statistical references
  - $\Rightarrow$ Always consider a XGBoost/Random Forest option... and compare it to the right baselines
  - $\Rightarrow$ Beware of magical features / unrealistic good prediction (!)

Global picture

python



Modèles AR — PROPHET, statsmodels, NumPy, pandas, matplotlib

Modèles ML : SVM, XGB — scikit learn, XGBoost

Modèles DL : CNN, RNN — PyTorch, NVIDIA CUDA

Incremental

- Different approaches, different paradigms/syntaxes
- Different costs, different expectations
- Different hardware supports
- Great tools... But remind the time frame