Décomposition et prévision des TS : de la théorie à la pratique

Syrine Ben Salah
Paul Péton

AVANADE

38,000
Professionnels

1000+
Consultants en France
(incluant Azeo)

85%
Certifiés

avanade

Créé en 2000 par Accenture et Microsoft, Avanade associe les meilleurs talents stratégiques et technologiques pour aider ses clients à libérer le potentiel de leurs systèmes informatiques et de leur activité.

accenture avanade Microsoft

Applications & Infrastructure

- Modern Application Transformation
- Microsoft Azure Platform Services

Modern Workplace

- Workplace Platform Modernization
- Workplace Value Realization

Data & AI

- Industrial IoT
- Intelligent Automation
- Data Platform Modernization
- Artificial intelligence

Business applications

- Finance and Operating Services
- Digital Sales and Service
- Digital Marketing

École IA Microsoft

Agenda

Principes de la décomposition

Forecasting

- Méthode naïve
- Exponential Smoothing

Quelques packages :

- fbprophet
- Neural Prophet
- Kats, PyCaret, AutoML de Databricks…

Questions pratiques

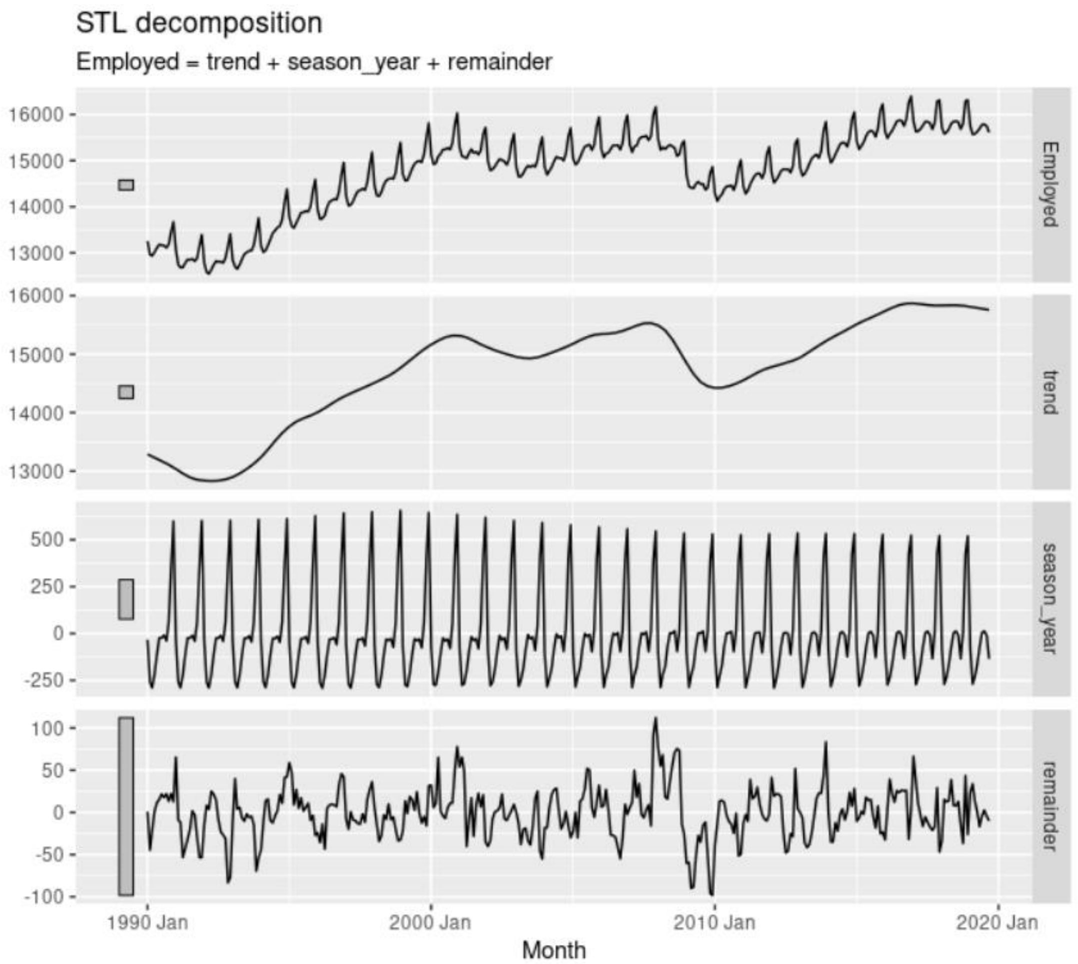
Time serie decomposition

Identifier les éléments composant la série :
- **Tendance** (pas forcément linéaire, pas forcément constante…)
- Un **cycle** (par exemple, macro-économique)
- Une (ou plusieurs) **saisonnalité**(s)
- Du **bruit** que l'on ne pourra jamais prévoir

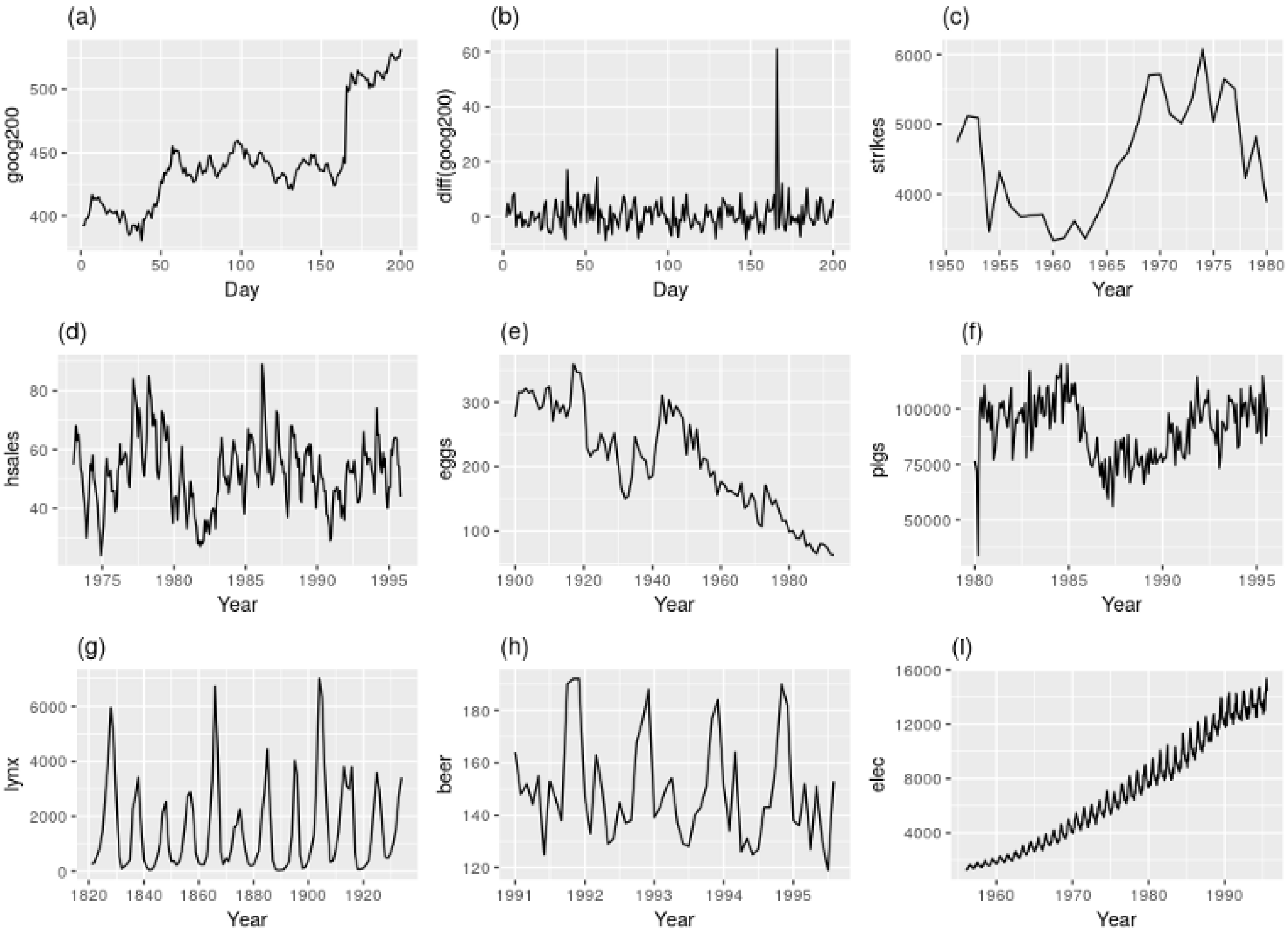
Ces éléments peuvent s'associer de manière **additive** ou **multiplicative**.

Méthode :
- **Isoler** chaque composant
- Les **analyser** individuellement
- Les modéliser individuellement pour le "**prolonger**"
- **Réassocier** toutes les parties dans un même modèle

Total monthly number of persons in thousands employed in the retail sector across the US since 1990



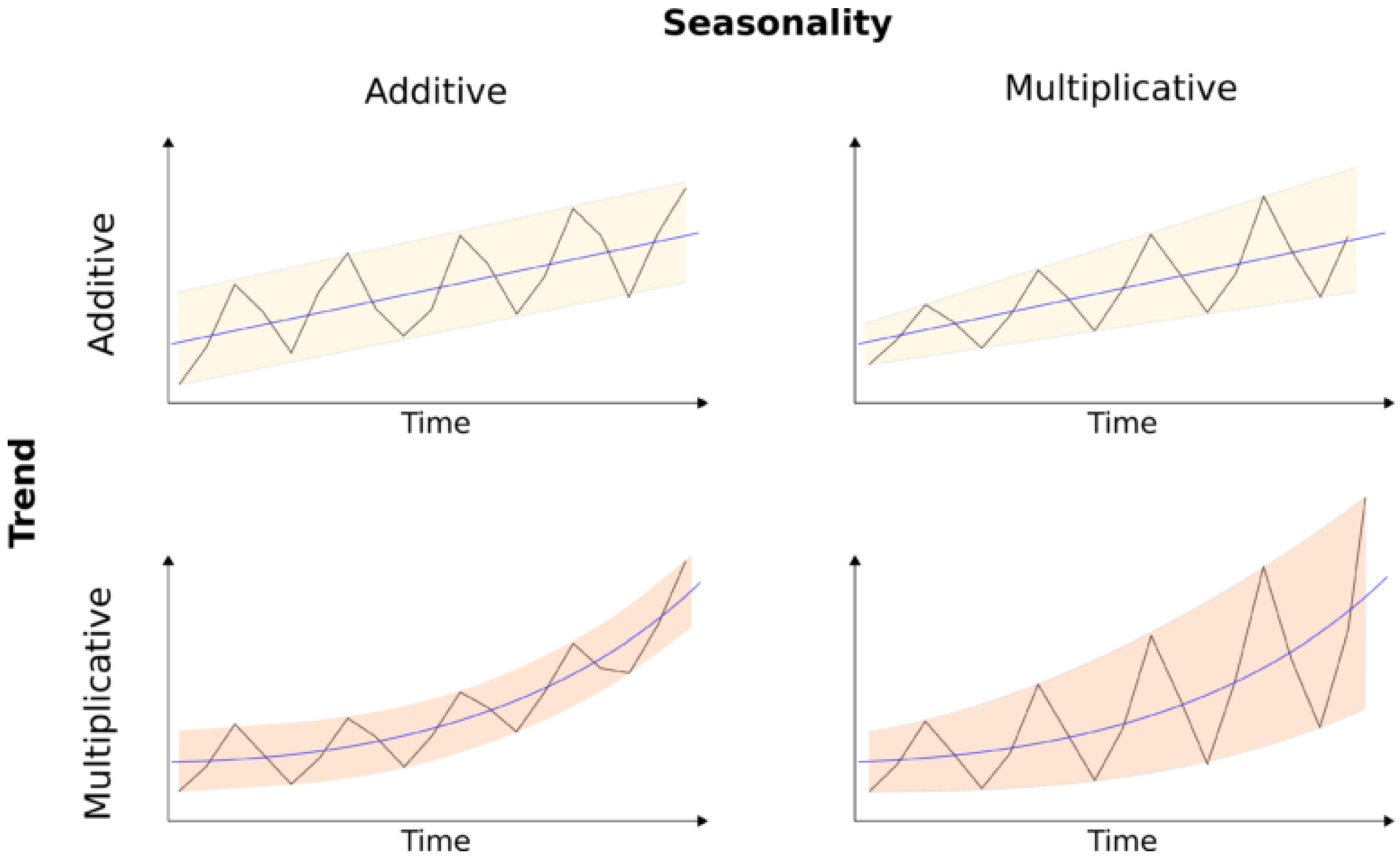
Tendance, saisonnalité ou bien… stationnaire ?



(a) Google stock price for 200 consecutive days
(b) Daily change in the Google stock price for 200 consecutive days
(c) Annual number of strikes in the US
(d) Monthly sales of new one-family houses sold in the US
(e) Annual price of a dozen eggs in the US (constant dollars)
(f) Monthly total of pigs slaughtered in Victoria, Australia
(g) Annual total of lynx trapped in the McKenzie River district of north-west Canada
(h) Monthly Australian beer production
(i) Monthly Australian electricity production

Seasonality : (d), (h), (i)
Trend : (a), (c), (e), (f), (i)
Stationary : (b), (g)

Additive versus multiplicative models

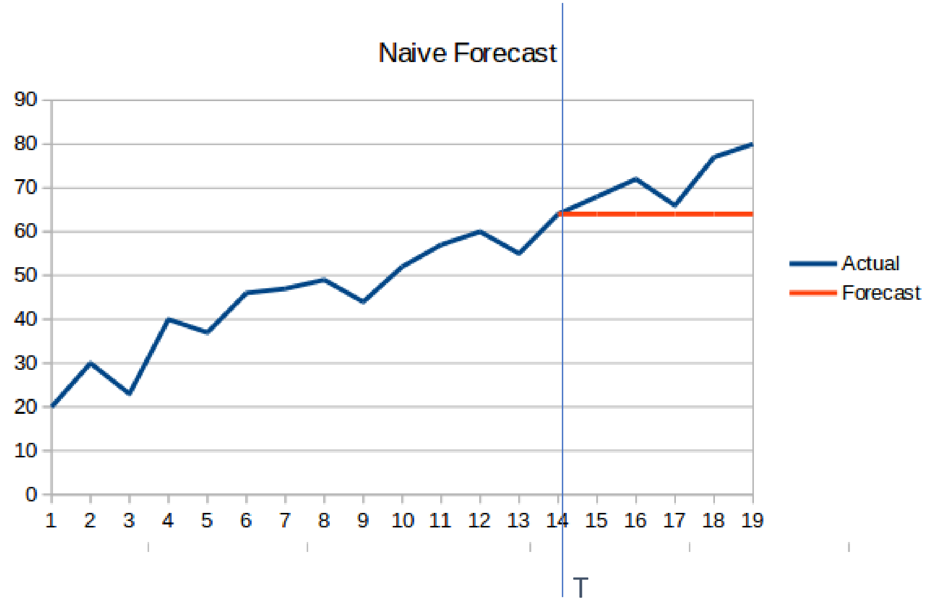


- For an additive decomposition, the *seasonally adjusted data* are
- For a multiplicative decomposition, the *seasonally adjusted data* are

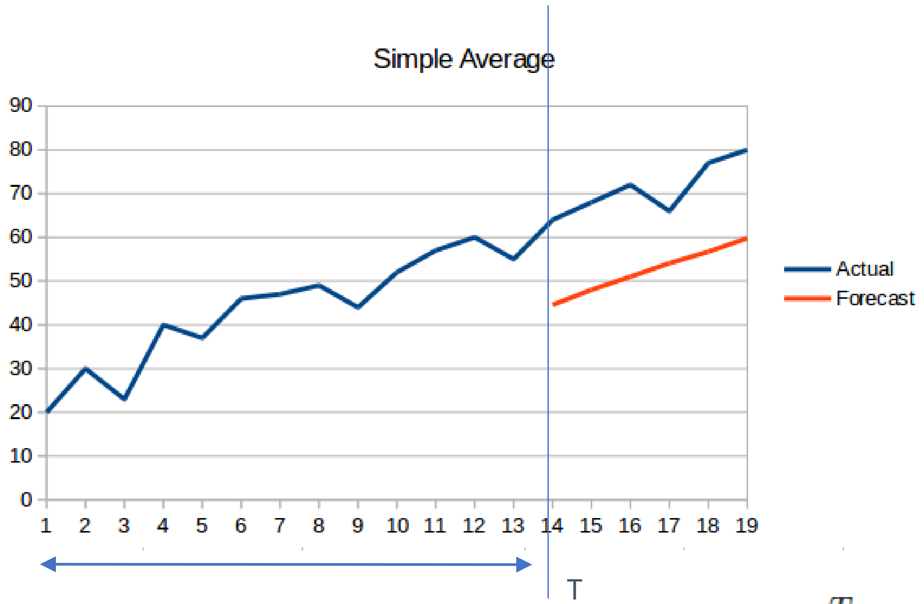
Source : https://www.daitan.com/innovation/exponential-smoothing-methods-for-time-series-forecasting/

Time series forecasting

Simplistic approach



$$\hat{y}_{T+h|T} = y_T$$



$$\hat{y}_{T+h|T} = \frac{1}{T}\sum_{t=1}^{T} y_t$$

Assume that the **most recent observation is the only important one**, and all previous observations provide no information for the future.

Assumes **that all observations are of equal importance** and gives them equal weights when generating forecasts.

Exponential smoothing

an approach in-between

Attach larger weights to more recent observations than to observations from the distant past.

Forecast at time T+1

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha) y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots$$

$$0 \le \alpha \le 1$$

α : is the smoothing parameter

	α=0.2	α=0.4	α=0.6	α=0.8
yT	0.2000	0.4000	0.6000	0.8000
yT−1	0.1600	0.2400	0.2400	0.1600
yT−2	0.1280	0.1440	0.0960	0.0320
yT−3	0.1024	0.0864	0.0384	0.0064
yT−4	0.0819	0.0518	0.0154	0.0013
yT−5	0.0655	0.0311	0.0061	0.0003

Exponential smoothing

3 types

- Simple exponential smoothing
- Double exponential smoothing (Holt’s trend method)
- Triple exponential smoothing (Holt-winters)

Simple exponential smoothing

Weighted average form

Forecast at time T+1

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots$$

$$0 \le \alpha \le 1$$

α : is the smoothing parameter

Fitted value (one-step forecast)

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

$$\text{for } t = 1, \ldots, T$$

$$0 \le \alpha \le 1$$

For t=1; $\hat{y}_{t|t-1}$ = ℓ_0

2 parameters

Weighted average form

$$\hat{y}_{T+1|T} = \alpha y_T + (1-\alpha)\hat{y}_{T|T-1}$$

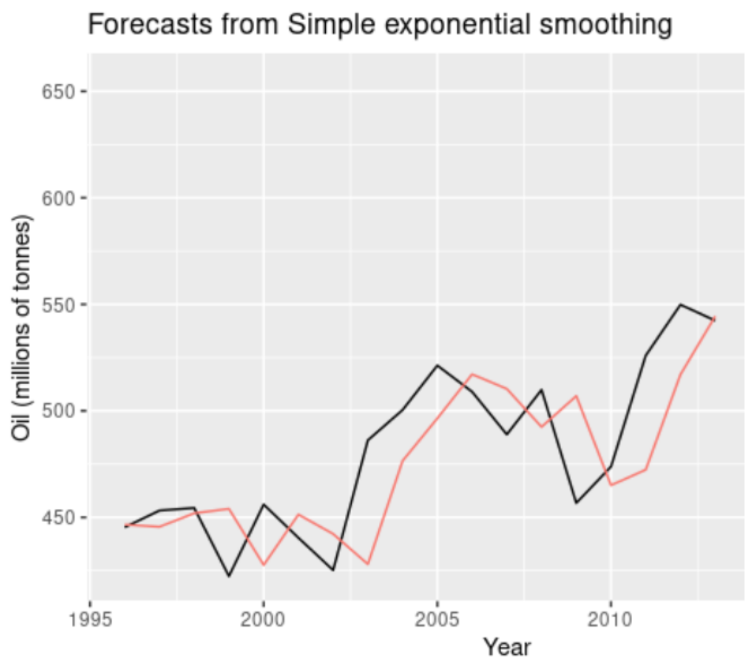
$$\hat{y}_{T|T-1} = \alpha y_{T-1} + (1-\alpha)\hat{y}_{T-1|T-2}$$

$$\vdots$$

$$\hat{y}_{4|3} = \alpha y_3 + (1-\alpha)\hat{y}_{3|2}$$

$$\hat{y}_{3|2} = \alpha y_2 + (1-\alpha)\hat{y}_{2|1}$$

$$\hat{y}_{2|1} = \alpha y_1 + (1-\alpha)\ell_0$$



Simple exponential smoothing

Component form

Weighted average form

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

$$0 \le \alpha \le 1$$

for $t = 1, \ldots, T$

Component form

Forecast equation	$\hat{y}_{t+h|t} = \ell_t$
Smoothing equation	$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}$

- ℓ_t is the level (or the smoothed value) of the series at time t.
- The smoothing equation for the level gives the estimated level of the series at each period t.
- The forecast equation shows that the forecast value at time t+1 is the estimated level at time t.
- The forecast is independent from h.

Simple exponential smoothing

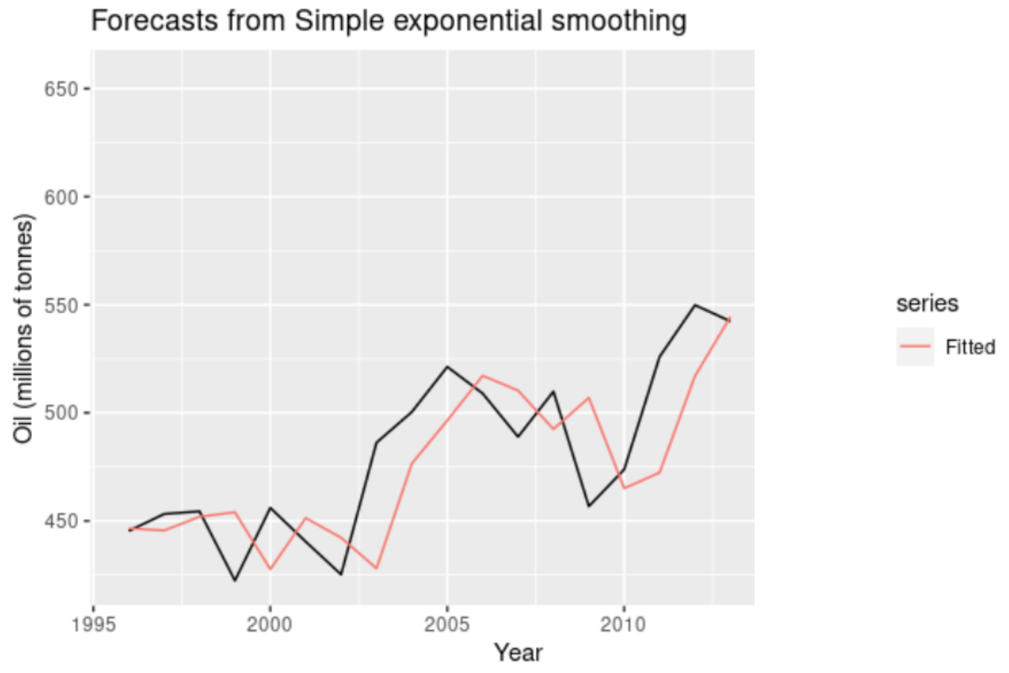


Figure 7.2: Simple exponential smoothing applied to oil production in Saudi Arabia (1996–2013).

On training data (fitted values = one-step forecast):

- Learn best α and ℓ_0, that minimize RSS (residual sum of squares)
- At each time t, calculate the level l_t (based on observed data and l_{t-1})
- Forecast at t+1 is equal to l_t

Simple exponential smoothing

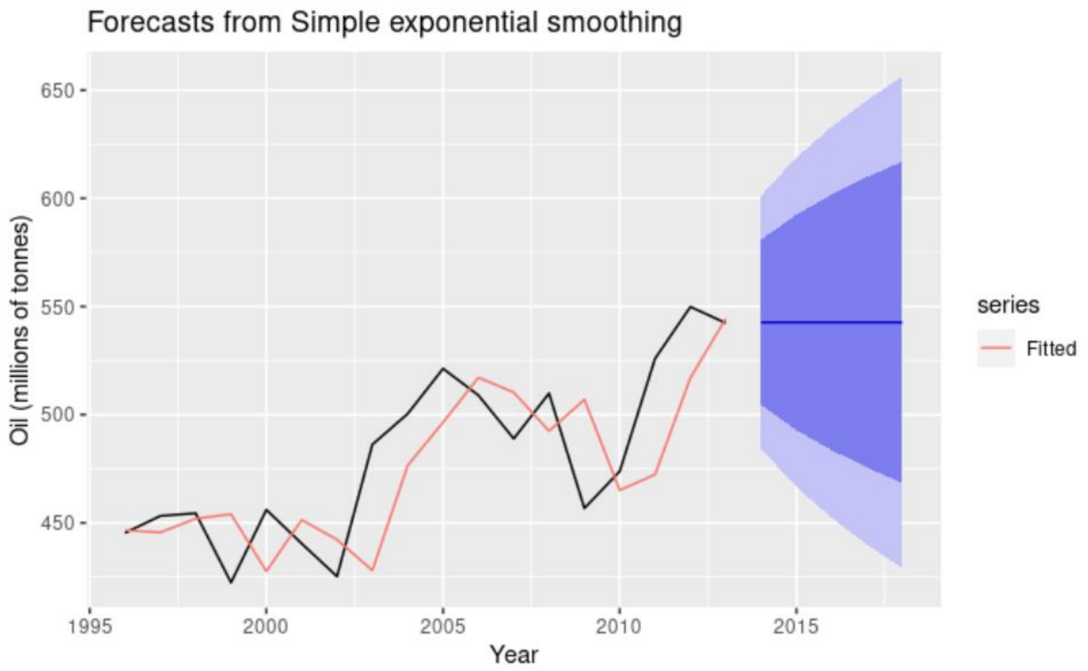


Figure 7.2: Simple exponential smoothing applied to oil production in Saudi Arabia (1996–2013).

On training data (fitted values = one-step forecast):

- Learn best α and ℓ_0, that minimize RSS (residual sum of squares)
- At each time t, calculate the level l_t (based on observed data and l_{t-1})
- Forecast at t+1 is equal to l_t

On testing data $\hat{y}_{t+h|t} = \ell_t$

Flat forecasts
Simple exponential smoothing will only be suitable if the time series has no trend or seasonal component.

Double exponential smoothing (Holt)

- Extend Simple exponential smoothing to **allow forecasting series with a trend**

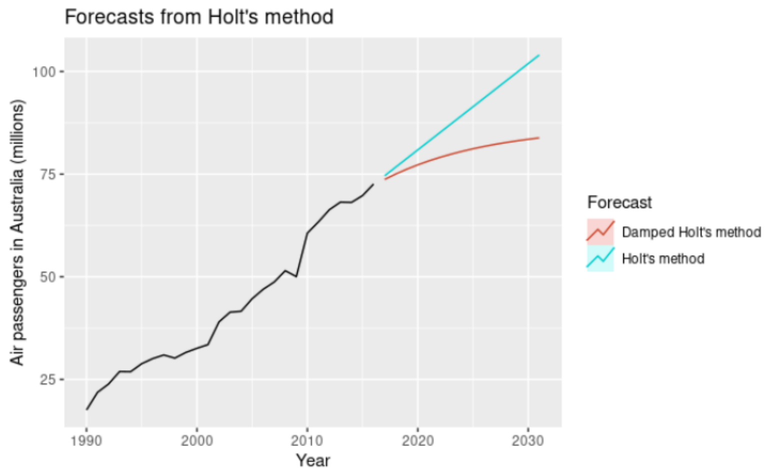


Figure 7.3: Forecasting total annual passengers of air carriers registered in Australia (millions of passengers, 1990–2016). For the damped trend method, $\phi = 0.90$.

Simple exponential smoothing

Forecast equation $\quad \hat{y}_{t+h|t} = \ell_t$

Smoothing equation $\quad \ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}$

$0 \leq \alpha \leq 1$

Double exponential smoothing

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

Forecast equation $\quad \hat{y}_{t+h|t} = \ell_t + hb_t$

Level equation $\quad \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1})$

Trend equation $\quad b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1},$

$0 \leq \alpha \leq 1$
$0 \leq \beta^* \leq 1.$

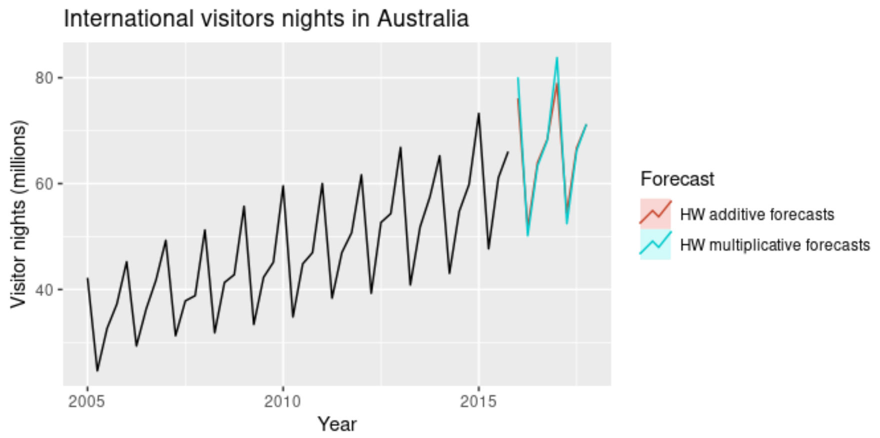
Estimated trend at time t

Estimated trend at time t-1

Triple exponential smoothing (Holt-winter)

Additive seasonality

- Extend double exponential smoothing **to consider serie with seasonality (and trend)**



Double exponential smoothing

$$\begin{aligned} \text{Forecast equation} \quad & \hat{y}_{t+h|t} = \ell_t + hb_t \\ \text{Level equation} \quad & \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ \text{Trend equation} \quad & b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}, \end{aligned}$$

$$0 \le \alpha \le 1$$
$$0 \le \beta^* \le 1$$

Triple exponential smoothing

$$\begin{aligned} \hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t+h-m(k+1)} \\ \ell_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \\ s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}, \end{aligned}$$

k is the integer part of $(h-1)/m$

Sum up

Simple exponential smoothing

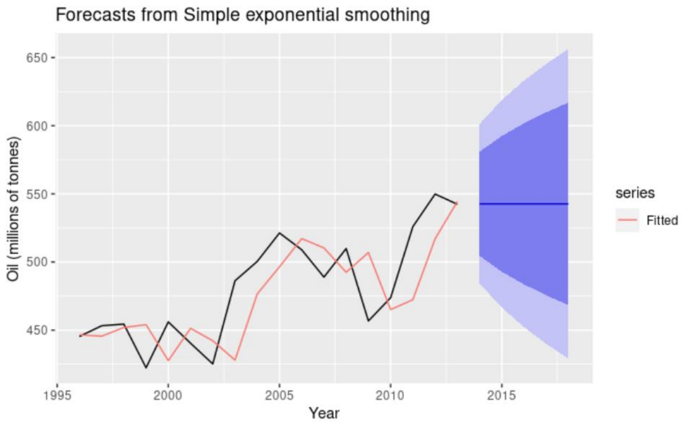


Figure 7.2: Simple exponential smoothing applied to oil production in Saudi Arabia (1996–2013).

No trend, no seasonality

Double exponential smoothing (Holt)

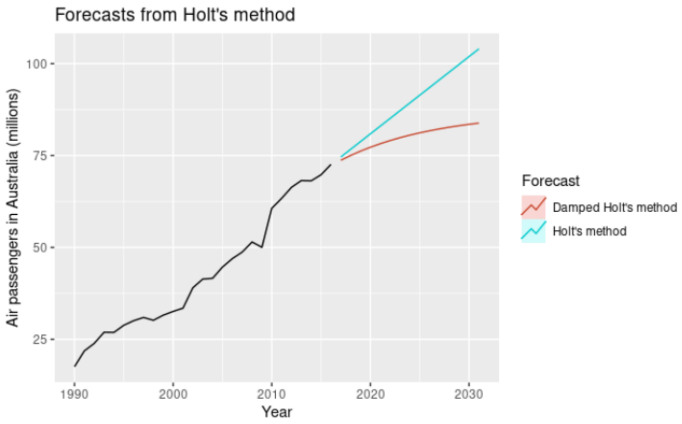
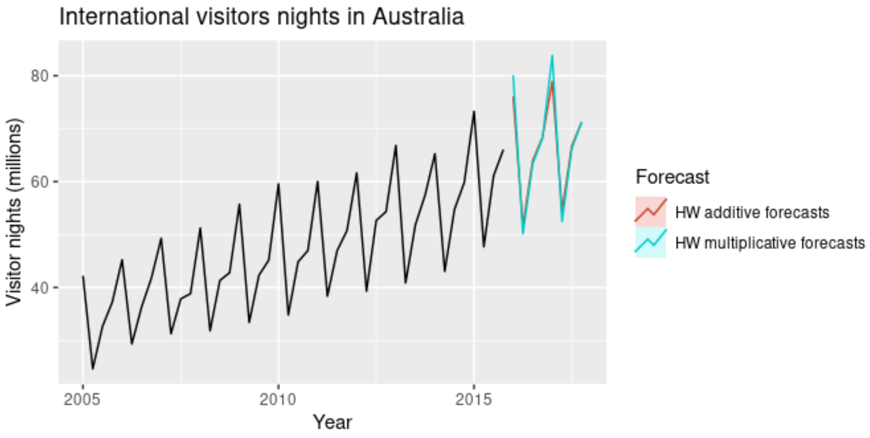


Figure 7.3: Forecasting total annual passengers of air carriers registered in Australia (millions of passengers, 1990–2016). For the damped trend method, $\phi = 0.90$.

Trend

Triple exponential smoothing (Holt-winters)

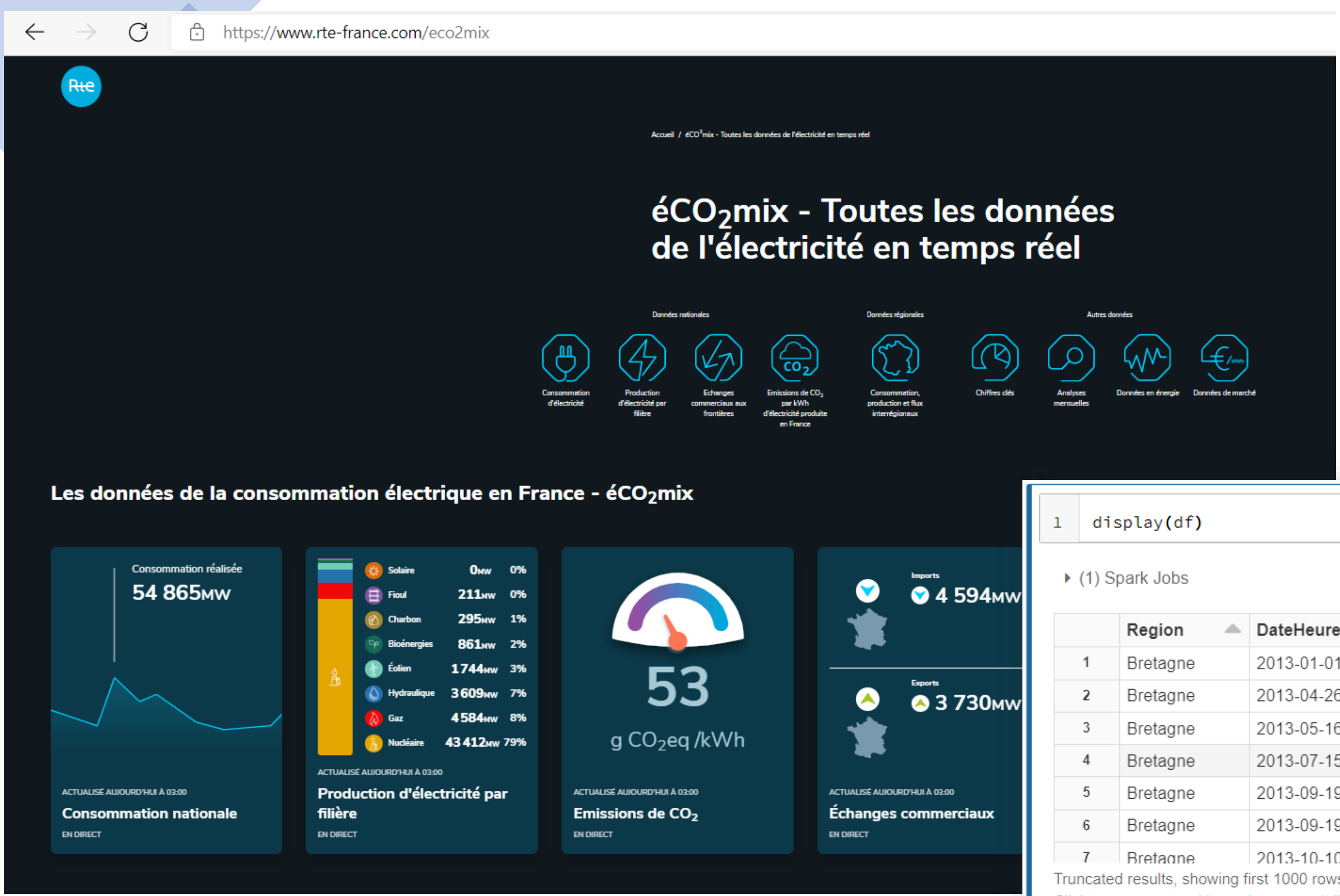


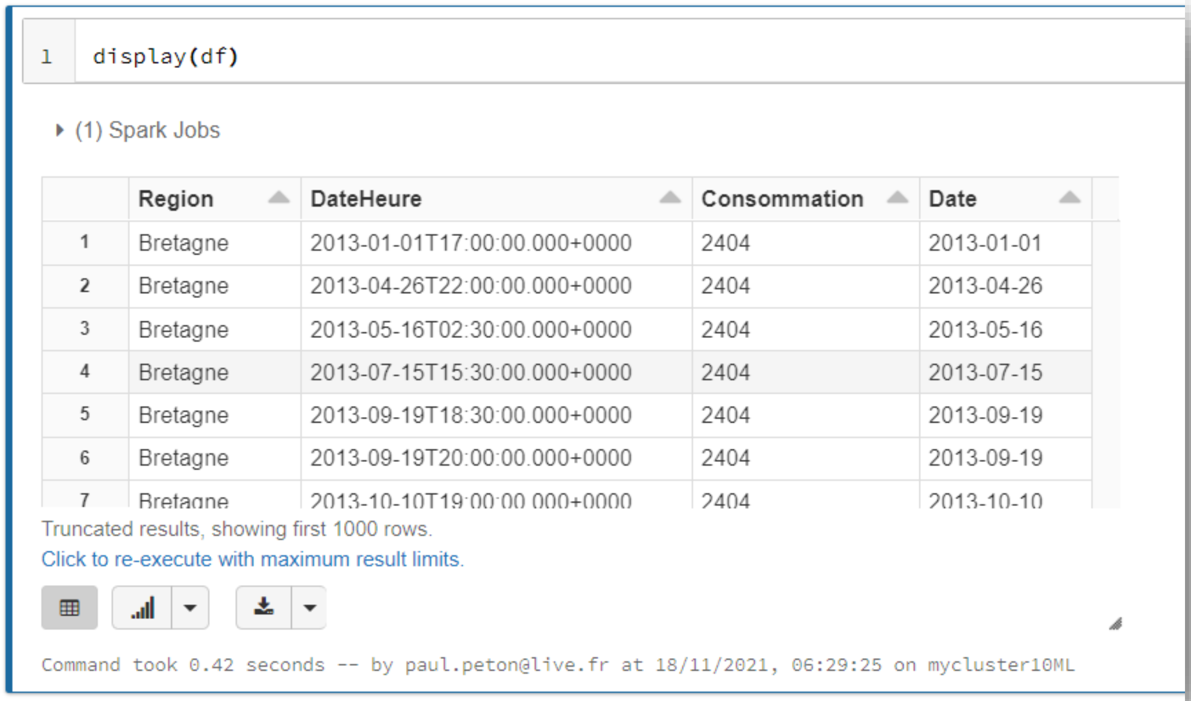
Seasonality

- It is easy to learn and apply.
- More suitable for short term forecast since it gives more importance to recent values
- Fast computation time

- Only univariate time series prediction
- Not for mid/long term forecast : as it assumes future patterns and trends will look like current patterns and trends (cf. lag behind actual)

notre dataset





```
1  display(df)
```

(1) Spark Jobs

	Region	DateHeure	Consommation	Date
1	Bretagne	2013-01-01T17:00:00.000+0000	2404	2013-01-01
2	Bretagne	2013-04-26T22:00:00.000+0000	2404	2013-04-26
3	Bretagne	2013-05-16T02:30:00.000+0000	2404	2013-05-16
4	Bretagne	2013-07-15T15:30:00.000+0000	2404	2013-07-15
5	Bretagne	2013-09-19T18:30:00.000+0000	2404	2013-09-19
6	Bretagne	2013-09-19T20:00:00.000+0000	2404	2013-09-19
7	Bretagne	2013-10-10T19:00:00.000+0000	2404	2013-10-10

Truncated results, showing first 1000 rows.

Click to re-execute with maximum result limits.

Command took 0.42 seconds -- by paul.peton@live.fr at 18/11/2021, 06:29:25 on mycluster10ML

PROPHET

Docs GitHub

Forecasting at scale.

Prophet is a forecasting procedure implemented in R and Python. It is fast and provides completely automated forecasts that can be tuned by hand by data scientists and analysts.

INSTALL PROPHET | GET STARTED IN R | GET STARTED IN PYTHON | READ THE PAPER

https://facebook.github.io/prophet/

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

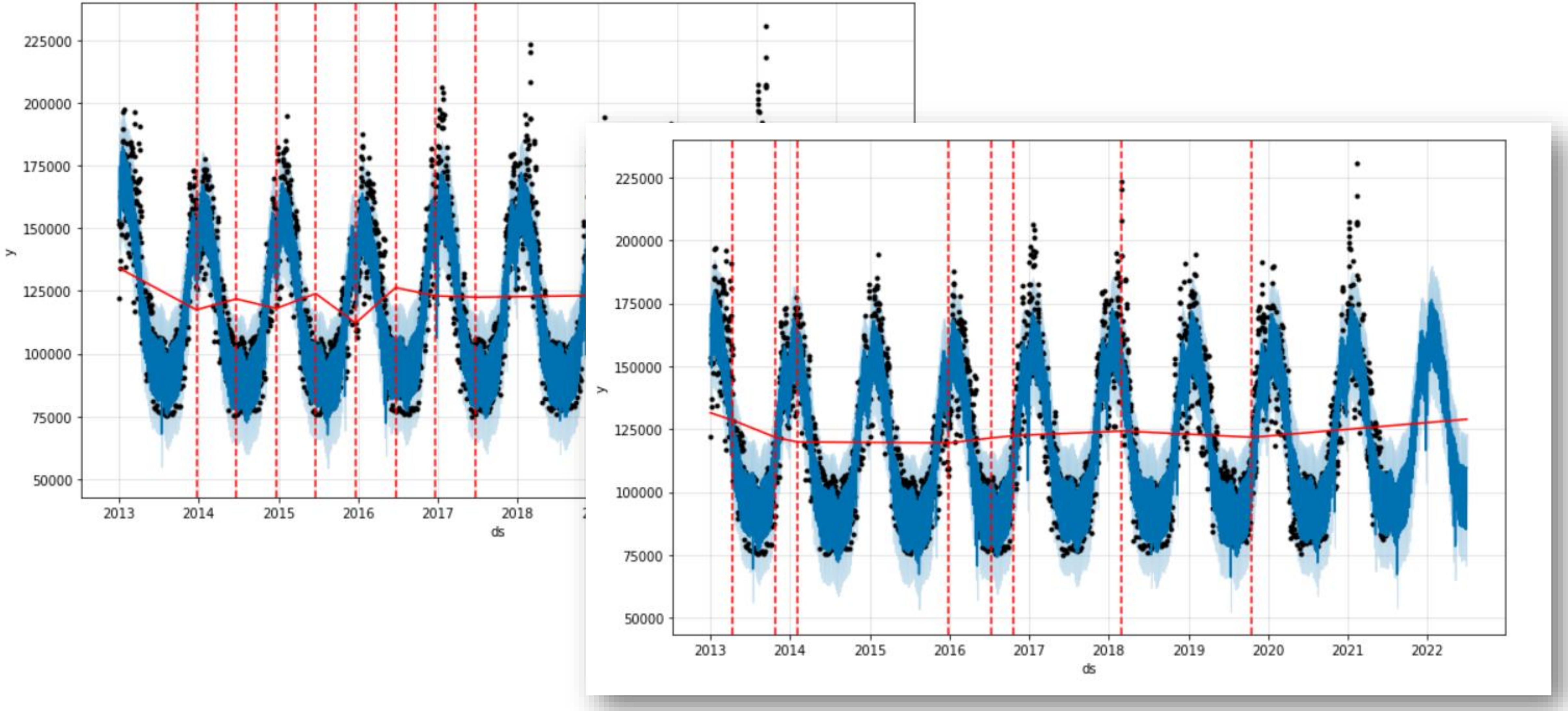
avec respectivement :

- g(t) : la **tendance** (linéaire ou logistique)
- s(s) : une ou plusieurs composantes **saisonnières** (annuelle, hebdomadaire ou quotidienne)
- h(t) : l'effet des **vacances** ou de jours spécifiques qui pourront être paramétrés
- e(t) : **l'erreur**, bruit aléatoire qui mesure l'écart entre le modèle et les données réelles

Quelques recommandations et astuces :

- Disposer d'années complètes
- Réaliser une CV pour déterminer les meilleures HP puis ré-entrainer avec les dernières données
- Tester l'ajout de « special events »

prophet changepoints



prophet prediction

```
df_test = df_day[(df_day['ds'] >= train_test_limit) & (df_day['ds'] < train_test_limit + timedelta(days = 365))]
print(df_test.shape)

range_test = m.make_future_dataframe(periods=365, freq='d', include_history=False)
fc_test = m.predict(range_test)
```

```
1  display(fc_test)
```

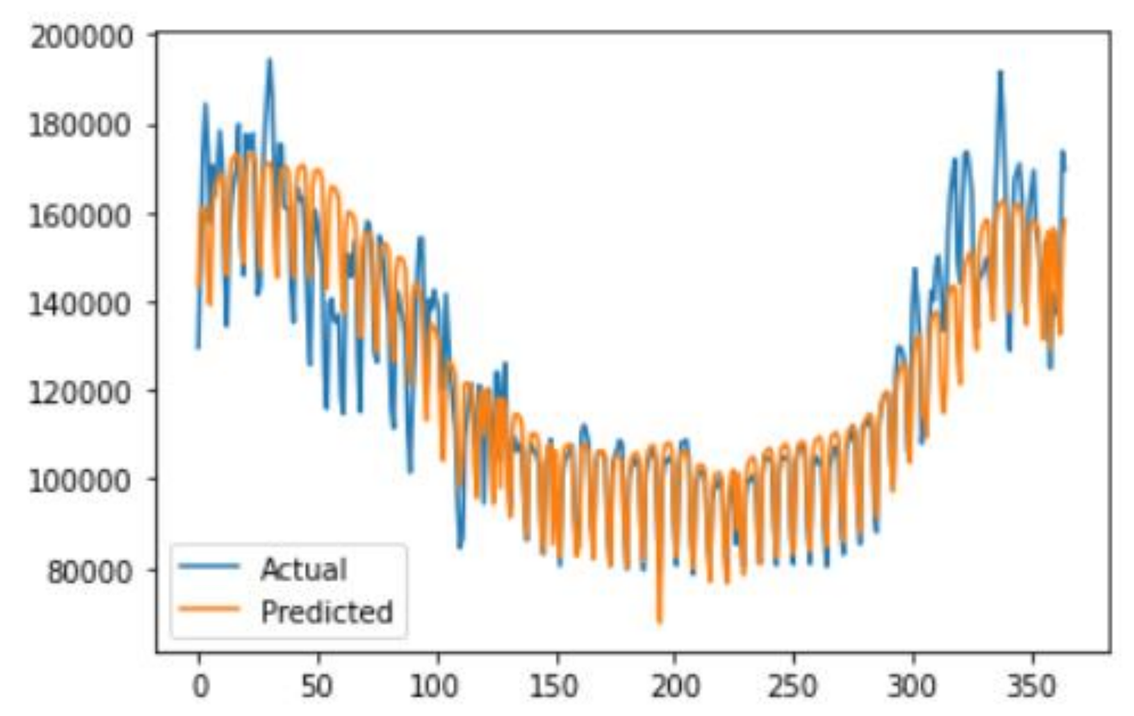
(2) Spark Jobs

	ds	trend	yhat_lower
1	2019-01-01T00:00:00.000+0000	124686.9710973223	131912.7056416217
2	2019-01-02T00:00:00.000+0000	124689.67056501805	147565.95364895702
3	2019-01-03T00:00:00.000+0000	124692.37003271384	149389.80123006777
4	2019-01-04T00:00:00.000+0000	124695.0695004096	149255.73971672292
5	2019-01-05T00:00:00.000+0000	124697.76896810539	133139.73953936002
6	2019-01-06T00:00:00.000+0000	124700.46843580117	127016.37604019756
7	2019-01-07T00:00:00.000+0000	124703.16790349693	148515.66385174484

Showing all 365 rows.

Command took 3.18 seconds -- by paul.peton@live.fr at 18/11/2021, 06:27:18 on mycluster

MAE: 6953

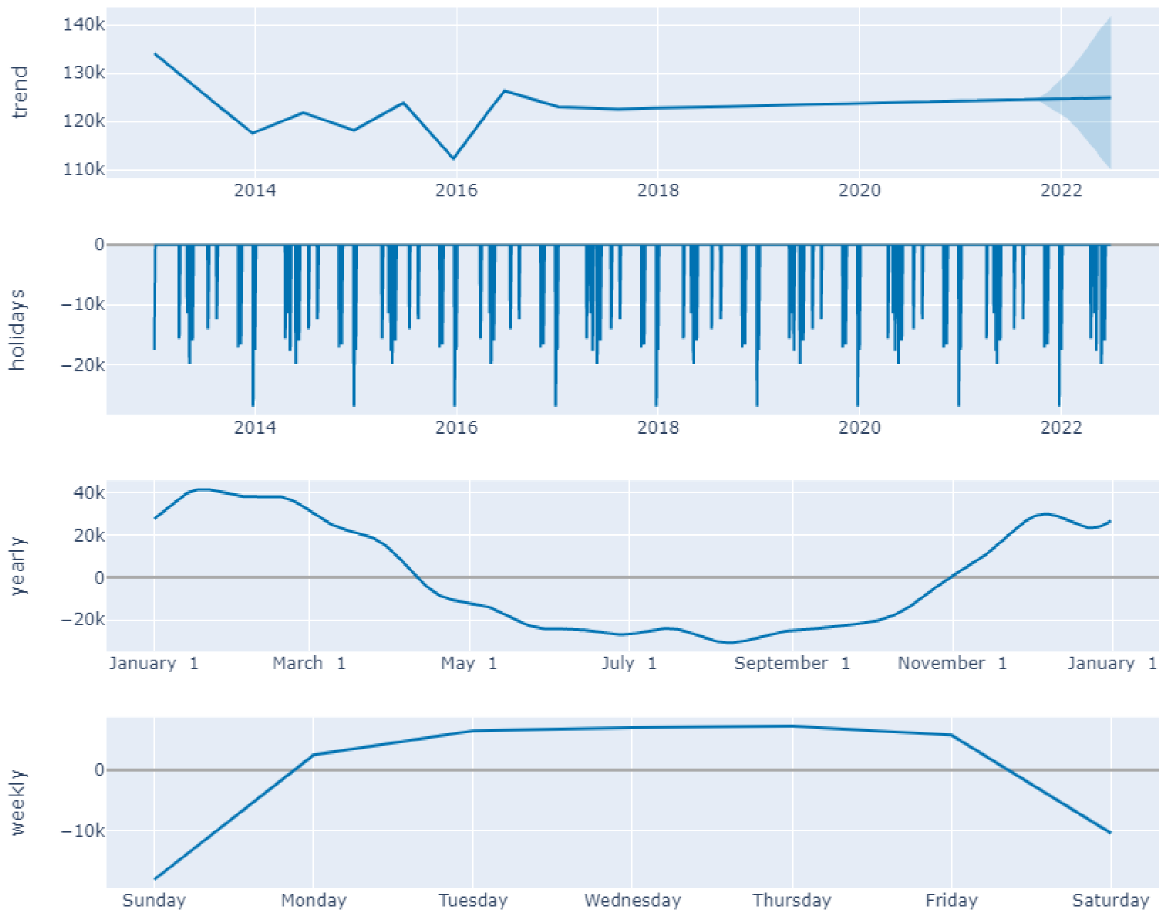


prophet decomposition

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

```
1  plot_components_plotly(m, forecast)
```

Out[50]:



release v0.2.8 | pypi v0.2.8 | python 3.6+ | code style black | license MIT

Neural Prophet

is — an open-source forecasting library.

tl;dr — Prophet in PyTorch + AR + Covar + NN + multistep + ...

Task: Forecasting.

Data: 1E+2 to 1E+6 of samples. Unidistant, real-valued.

Dynamics: Future values must depend on past observations.
e.g. Seasonal, trended, events, correlated variables.

Applications: Human behavior, energy, traffic, sales, environment, server load, ...

https://github.com/ourownstory/neural_prophet
https://neuralprophet.com/html/index.html

NeuralProphet is more than the Neural evolution of Prophet.

facebook

PROPHET

Stanford University

Neural Prophet

facebook TotalEnergies MONASH University Skoltech

Prophet has three major shortcomings:

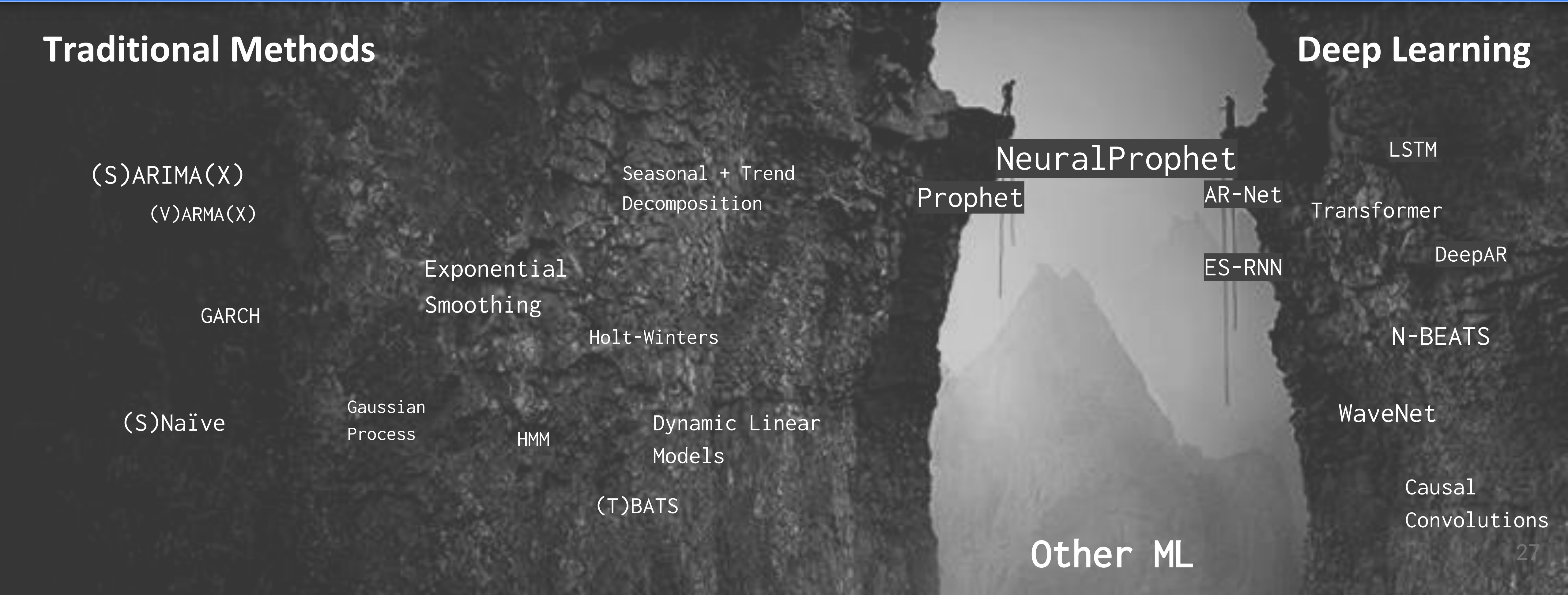
1. Missing local context for predictions
2. Acceptable forecast accuracy
3. Framework is difficult to extend (Stan)

NeuralProphet solves these:

1. Support for auto-regression and covariates.
2. Hybrid model (linear <> Neural Network)
3. Python package based on PyTorch using standard deep learning methods.

Stan → PyTorch

Time series forecasting is messy.
We need hybrid models to bridge the gap.



Neural Prophet

Model components

$$y(t) = g(t) + s(t) + h(t) + \underbrace{\mathbf{AR(t) + LR(t)}}_{\textit{Fully connected neural networks}} + \varepsilon(t)$$

- g(t): piecewise linear or logistic growth curve for modelling non-periodic changes in time series
- s(t): periodic changes (e.g. weekly/yearly seasonality)
- h(t): effects of holidays (user provided) with irregular schedules
- AR(t) : to model Auto-Regression
- LR(t) : to model covariates (Lagged regression)
- ε_t: error term accounts for any unusual changes not accommodated by the model

Model Auto-regression and covariates as AR-Nets

Auto-Regression (AR)

AR refers to the process of regressing a variable's future value against its past values.

time	Target
0	y_0
1	y_1
…	…
p	y_p
…	…
t-2	y_{t-2}
t-1	y_{t-1}
t	y_t

The number of past values included is usually referred to as the order p of the AR(p) model.
In classic AR model

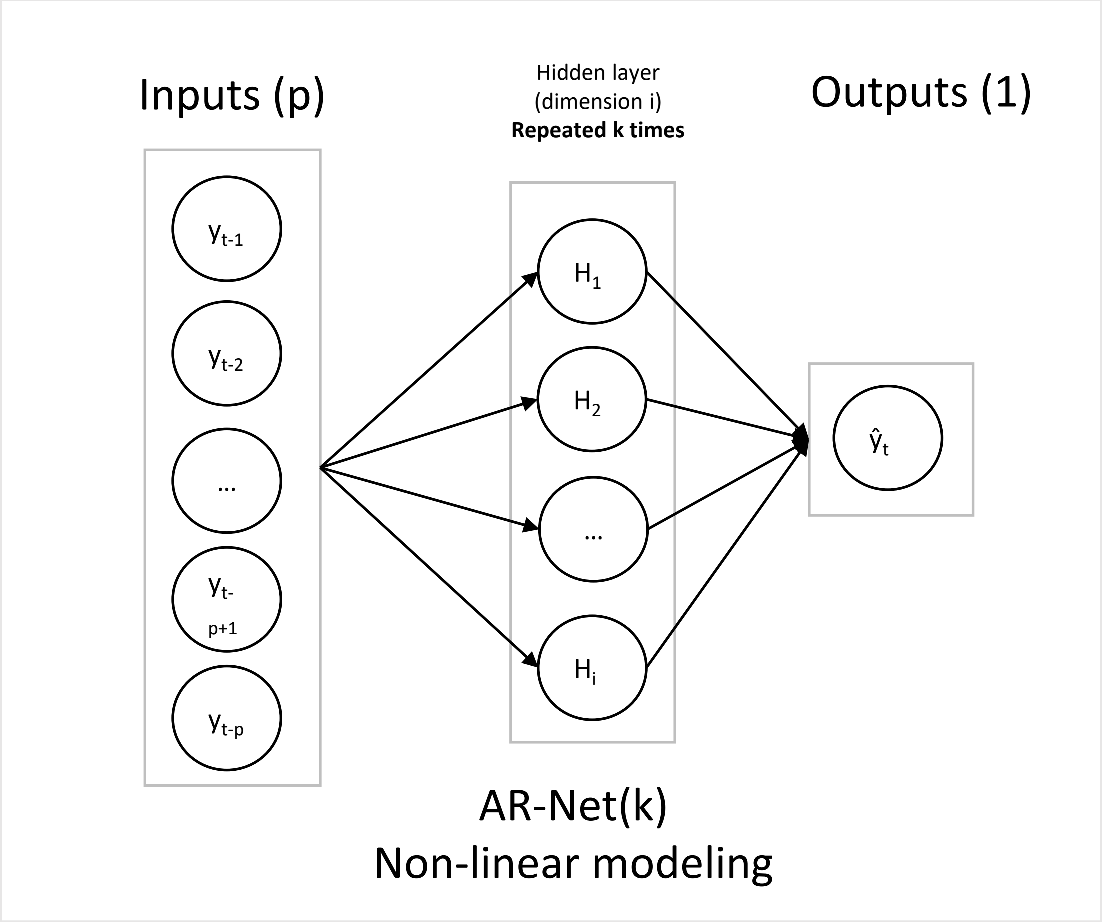
$$y_t = c + \sum_{i=1}^{i=p} \theta_i \cdot y_{t-i} + e_t$$

Neural Prophet

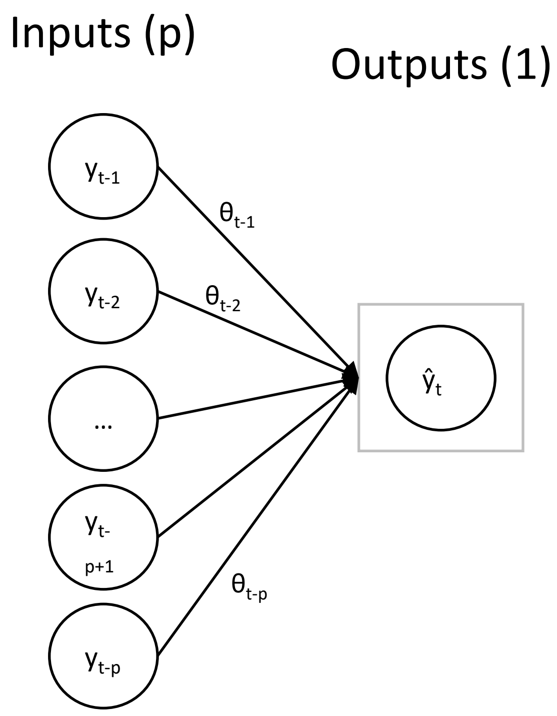
Model Auto-Regression

Neural prophet params :

- n_lags = p
- n_forecasts = 1
- num_hidden_layer = k
- d_hidden = i

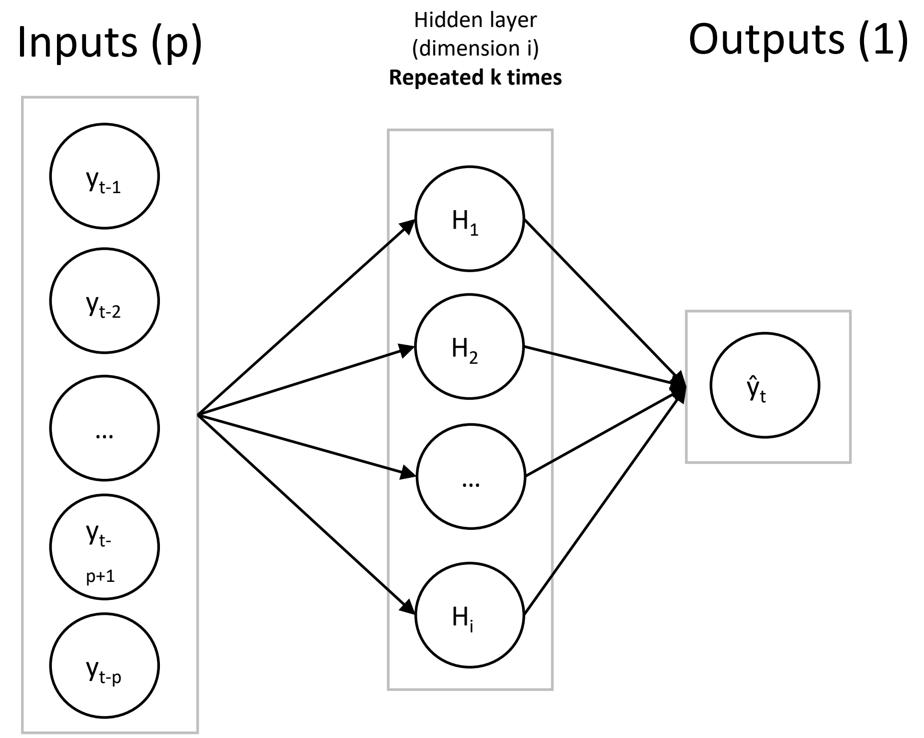


NeuralProphet



n_lags = p
n_forecasts =1
num_hidden_layer = 0

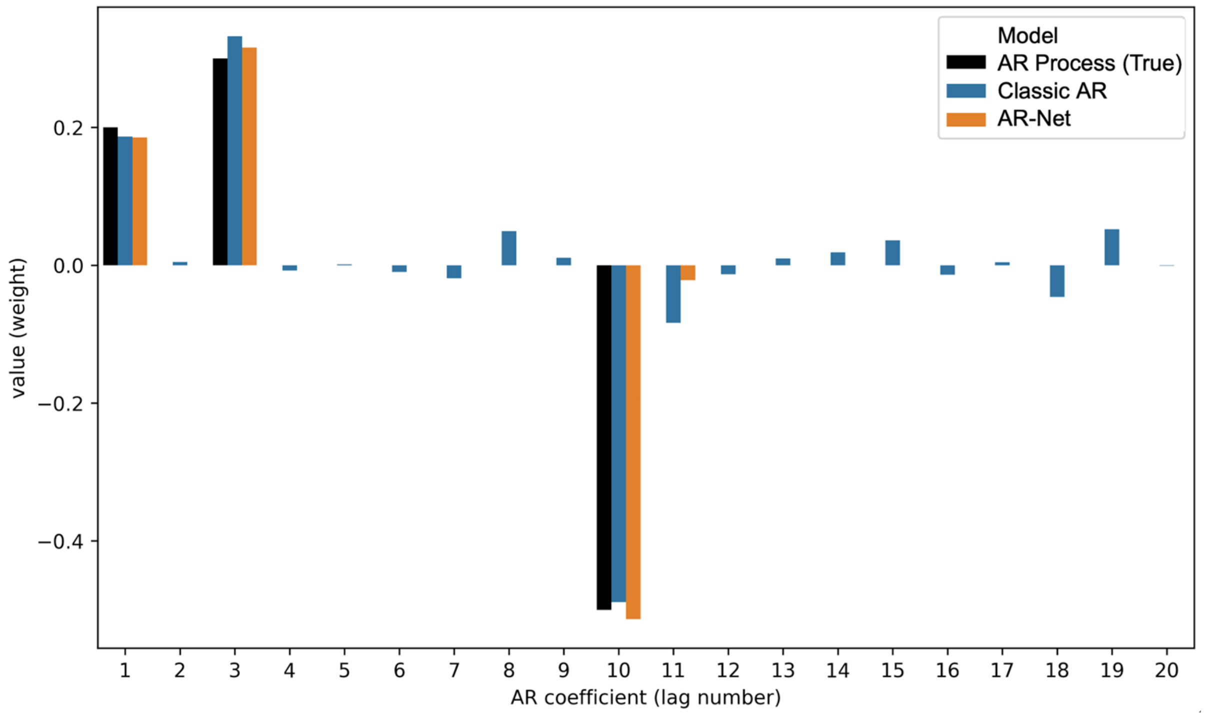
AR-Net(0)
Interpretable



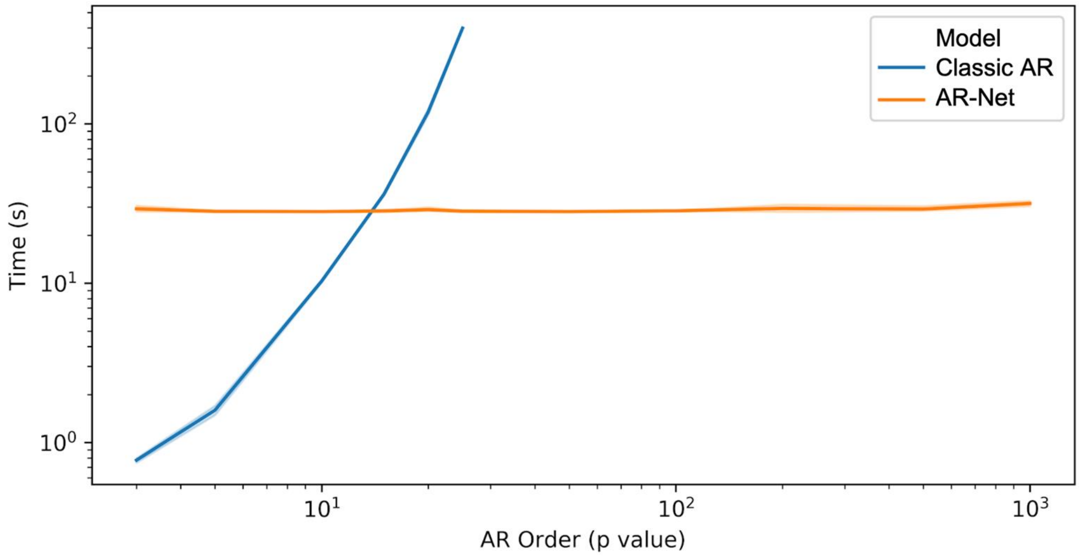
n_lags = p
n_forecasts = 1
num_hidden_layer = k
d_hidden = i

AR-Net(k)
Non-linear modeling

Automatic AR-lag selection, yet faster.



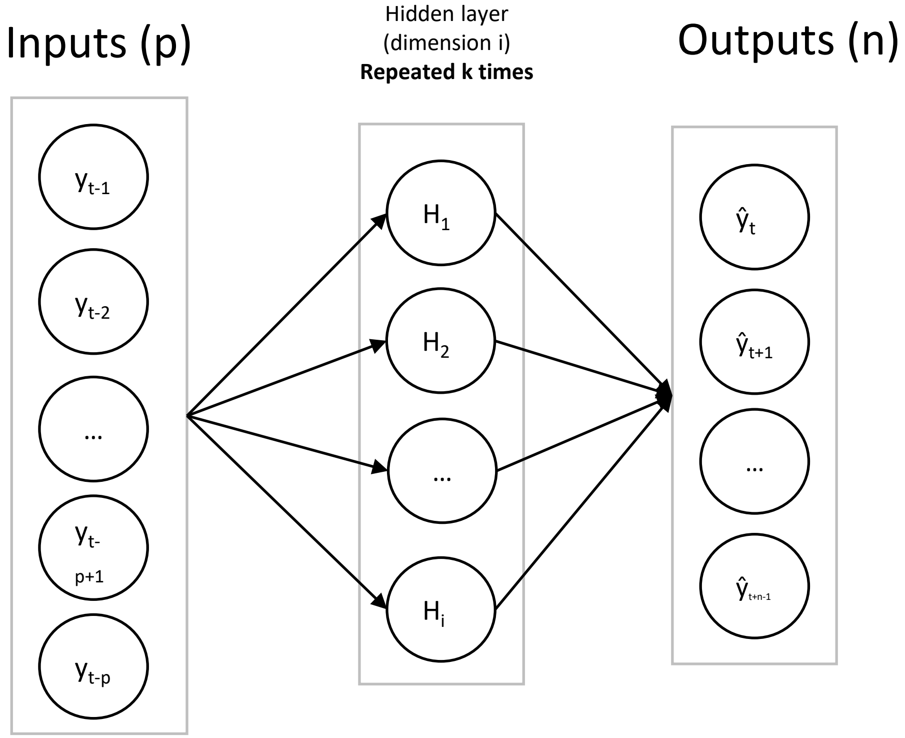
Automatic Sparsity



Quadratically faster

NeuralProphet

Forecast horizon > 1



n_lags = p
n_forecasts = n
n_hidden_layer = k
d_hidden = i

AR-Net(k)
Non-linear modeling

A user-friendly Python package

Gentle learning curve.

Get results first. Learn. Improve.

Powerful, customizable, extendable.

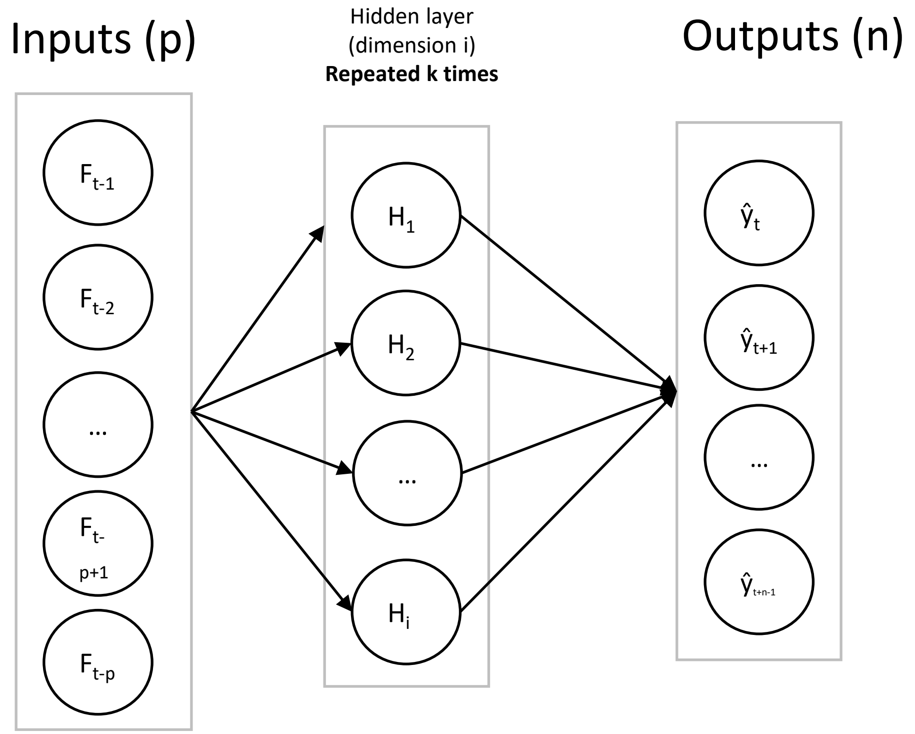
```
m = NeuralProphet()
metrics = m.fit(df, freq='D')
forecast = m.predict(df)
m.plot(forecast)
```

NeuralProphet

Model covariates

Lagged Regression

time	feature	Target
0	F_0	y_0
1	F_1	y_1
…	…	…
t-2	F_{t-2}	y_{t-2}
t-1	F_{t-1}	y_{t-1}
		y_t



Fully connected Neural network

Upcomings

STAY TUNED

Extensions [upcoming]

- Hierarchical Forecasting
  & Global Modelling
- Quantifiable and Explainable Uncertainty
- Anomaly Prediction
  & Semi-Supervised Learning
- Attention: Automatic Multimodality
  & Dynamic Feature Importance

Improvements [upcoming]

- Improved NN
- Faster Training Time
  & GPU support
- Improved UI
- Diagnostic Tools for Deep Dives

Anything trainable by gradient descent can be added as module

Kats

One stop shop for time series analysis in Python

Get Started

Star 3,217

https://facebookresearch.github.io/Kats/

Kats is a toolkit to analyze time series data, a lightweight, easy-to-use, and generalizable framework to perform time series analysis. Time series analysis is an essential component of Data Science and Engineering work at industry, from understanding the key statistics and characteristics, detecting regressions and anomalies, to forecasting future trends. Kats aims to provide the one-stop shop for time series analysis, including detection, forecasting, feature extraction/embedding, multivariate analysis, etc. Kats is released by Facebook's Infrastructure Data Science team. It is available for download on PyPI.

Forecasting

Detection

TSFeatures

Utilities

- kats.models.metalearner package
  - kats.models.metalearner.get_metadata module
  - kats.models.metalearner.metalearner_hpt module
  - kats.models.metalearner.metalearner_modelselect module
  - kats.models.metalearner.metalearner_predictability module
  - kats.models.metalearner module
- kats.models.model module
- kats.models.nowcasting package
  - kats.models.nowcasting.feature_extraction module
  - kats.models.nowcasting.model_io module
  - kats.models.nowcasting.nowcasting module
  - kats.models.nowcasting module
- kats.models.prophet module
- kats.models.quadratic_model module
- kats.models.reconciliation package

- kats.models package
  - kats.models.arima module
  - kats.models.bayesian_var module
  - kats.models.ensemble package
    - kats.models.ensemble.ensemble module
    - kats.models.ensemble.kats_ensemble module
    - kats.models.ensemble.median_ensemble module
    - kats.models.ensemble.weighted_avg_ensemble module
    - kats.models.ensemble module
  - kats.models.harmonic_regression module
  - kats.models.holtwinters module
  - kats.models.linear_model module
  - kats.models.lstm module

- kats.models.reconciliation package
  - kats.models.reconciliation.base_models module
  - kats.models.reconciliation.thm module
  - kats.models.reconciliation module
- kats.models.sarima module
- kats.models.stlf module
- kats.models.theta module
- kats.models.var module
- kats.models module

Time series features

Cmd 58

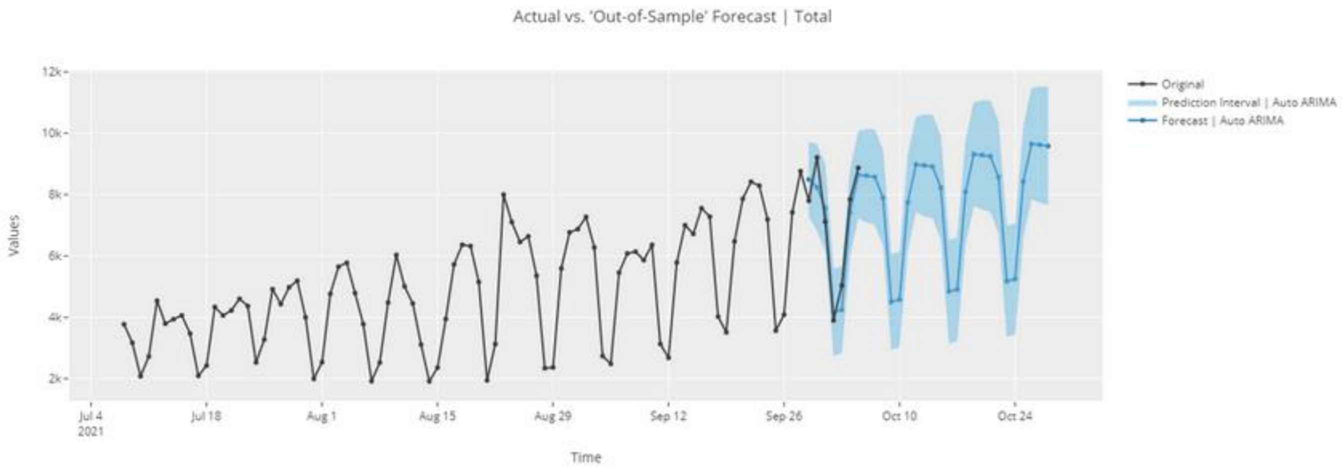
```python
# Initiate feature extraction class
from kats.tsfeatures.tsfeatures import TsFeatures

features = TsFeatures().transform(kats_ts)
features
```

```
INFO:numba.core.transforms:finding looplift candidates
INFO:numba.core.transforms:finding looplift candidates
Out[48]: {'length': 3103,
 'mean': 122553.42732839187,
 'var': 841434913.5969608,
 'entropy': 0.3974353904243942,
 'lumpiness': 1.2709757190432784e+16,
 'stability': 675695093.5617276,
 'flat_spots': 1,
 'hurst': 0.09971712749496514,
 'std1st_der': 8749.786788183632,
 'crossing_points': 199,
 'binarize_mean': 0.43474057363841445,
 'unitroot_kpss': 0.649565748617368,
 'heterogeneity': 2846.417272383972,
 'histogram_mode': 90592.2,
 'linearity': 0.0001424275838226213,
 'trend_strength': 0.9746746652418055,
 'seasonality_strength': 0.8319643021683869,
 'spikiness': 188902088.4567415,
```

Announcing PyCaret's New Time Series Module

Moez Ali 1 day ago · 6 min read



(Image by Author) PyCaret's New Time Series Module

Statistical testing, model training and selection (30+ algorithms), model analysis, automated hyperparameter tuning, experiment logging, deployment on cloud, and more.

compare_models *function trains and evaluates 30+ algorithms from ARIMA to XGboost (TBATS, FBProphet, ETS, and more).*

	Model	MAE	RMSE	MAPE	S
auto_arima	Auto ARIMA	531.918	626.211	0.0911	0
arima	ARIMA	568.497	687.251	0.0957	0
theta	Theta Forecaster	526.386	672.942	0.0921	0
lr_cds_dt	Linear w/ Cond. Deseasonalize & Detrending	607.773	752.876	0.0972	0
en_cds_dt	Elastic Net w/ Cond. Deseasonalize & Detrending	607.774	752.875	0.0972	0
ridge_cds_dt	Ridge w/ Cond. Deseasonalize & Detrending	607.773	752.876	0.0972	0
lasso_cds_dt	Lasso w/ Cond. Deseasonalize & Detrending	607.774	752.875	0.0972	0
lar_cds_dt	Least Angular Regressor w/ Cond. Deseasonalize...	607.773	752.876	0.0972	0
rf_cds_dt	Random Forest w/ Cond. Deseasonalize & Detrending	605.303	691.655	0.0963	0
llar_cds_dt	Lasso Least Angular Regressor w/ Cond. Deseaso...	622.354	757.409	0.1005	0
omp_cds_dt	Orthogonal Matching Pursuit w/ Cond. Deseasona...	637.073	779.951	0.1038	0
et_cds_dt	Extra Trees w/ Cond. Deseasonalize & Detrending	633.58	719.312	0.1022	0
lightgbm_cds_dt	Light Gradient Boosting w/ Cond. Deseasonalize...	623.385	713.959	0.1086	0
br_cds_dt	Bayesian Ridge w/ Cond. Deseasonalize & Detren...	649.424	764.901	0.1061	
ada_cds_dt	AdaBoost w/ Cond. Deseasonalize & Detrending	639.408	729.451	0.1057	0
exp_smooth	Exponential Smoothing	554.193	638.494	0.1026	0
gbr_cds_dt	Gradient Boosting w/ Cond. Deseasonalize & Det...	685.786	798.831	0.1099	0
knn_cds_dt	K Neighbors w/ Cond. Deseasonalize & Detrending	749.455	857.95	0.124	0
dt_cds_dt	Decision Tree w/ Cond. Deseasonalize & Detrending	778.955	906.237	0.1244	0
snaive	Seasonal Naive Forecaster	758.619	868.809	0.1273	0
huber_cds_dt	Huber w/ Cond. Deseasonalize & Detrending	823.94	953.952	0.141	0
par_cds_dt	Passive Aggressive w/ Cond. Deseasonalize & De...	879.72	1070.64	0.1433	0
naive	Naive Forecaster	1340.62	1921.11	0.3362	0
polytrend	Polynomial Trend Forecaster	1636.96	1793.5	0.3259	0
grand_means	Grand Means Forecaster	2193.97	2395.24	0.3474	0

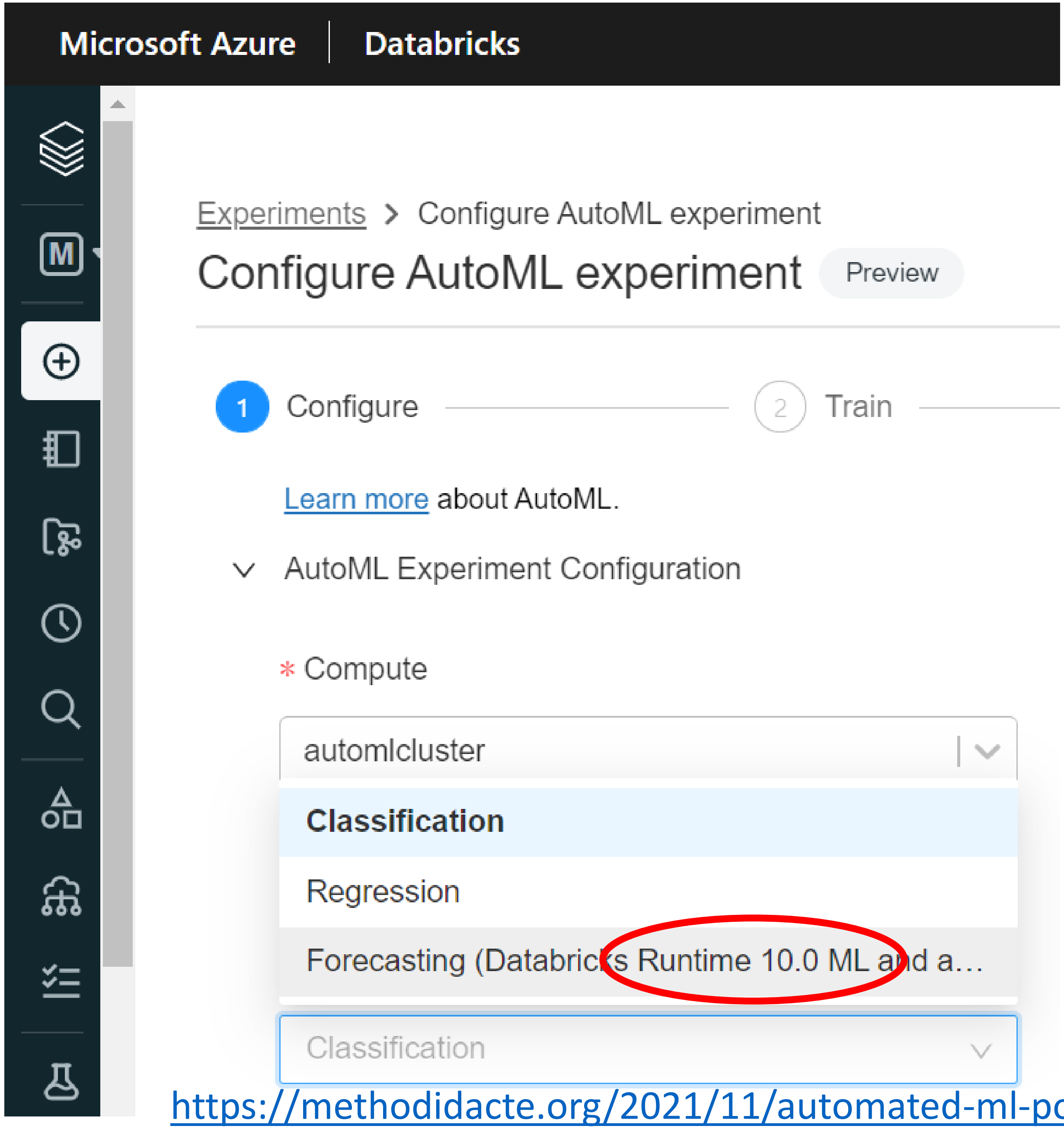
Tests statistiques automatisés

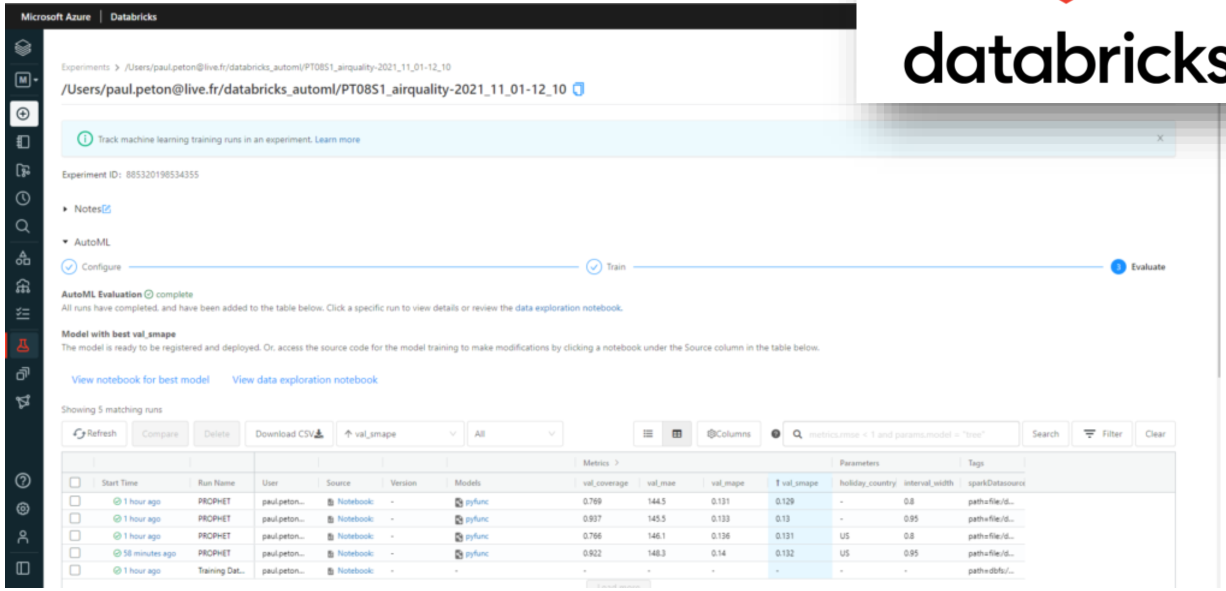
	Test	Test Name	Property	Setting	Value
0	Summary	Statistics	Length		3103.0
1	Summary	Statistics	Mean		122553.427328
2	Summary	Statistics	Median		114517.0
3	Summary	Statistics	Standard Deviation		29012.172776
4	Summary	Statistics	Variance		841706169.210626
5	Summary	Statistics	Kurtosis		-0.550774
6	Summary	Statistics	Skewness		0.523174
7	Summary	Statistics	# Distinct Values		3051.0
8	White Noise	Ljung-Box	Test Statictic	{'alpha': 0.05, 'K': 24}	43400.733667
9	White Noise	Ljung-Box	Test Statictic	{'alpha': 0.05, 'K': 48}	69519.004581
10	White Noise	Ljung-Box	p-value	{'alpha': 0.05, 'K': 24}	0.0
11	White Noise	Ljung-Box	p-value	{'alpha': 0.05, 'K': 48}	0.0
12	White Noise	Ljung-Box	White Noise	{'alpha': 0.05, 'K': 24}	False
13	White Noise	Ljung-Box	White Noise	{'alpha': 0.05, 'K': 48}	False
14	Stationarity	ADF	Stationarity	{'alpha': 0.05}	True
15	Stationarity	ADF	p-value	{'alpha': 0.05}	0.000976
16	Stationarity	ADF	Test Statistic	{'alpha': 0.05}	-4.098078
17	Stationarity	ADF	Critical Value 1%	{'alpha': 0.05}	-3.43248
18	Stationarity	ADF	Critical Value 5%	{'alpha': 0.05}	-2.862481
19	Stationarity	ADF	Critical Value 10%	{'alpha': 0.05}	-2.567271
20	Stationarity	KPSS	Trend Stationarity	{'alpha': 0.05}	True

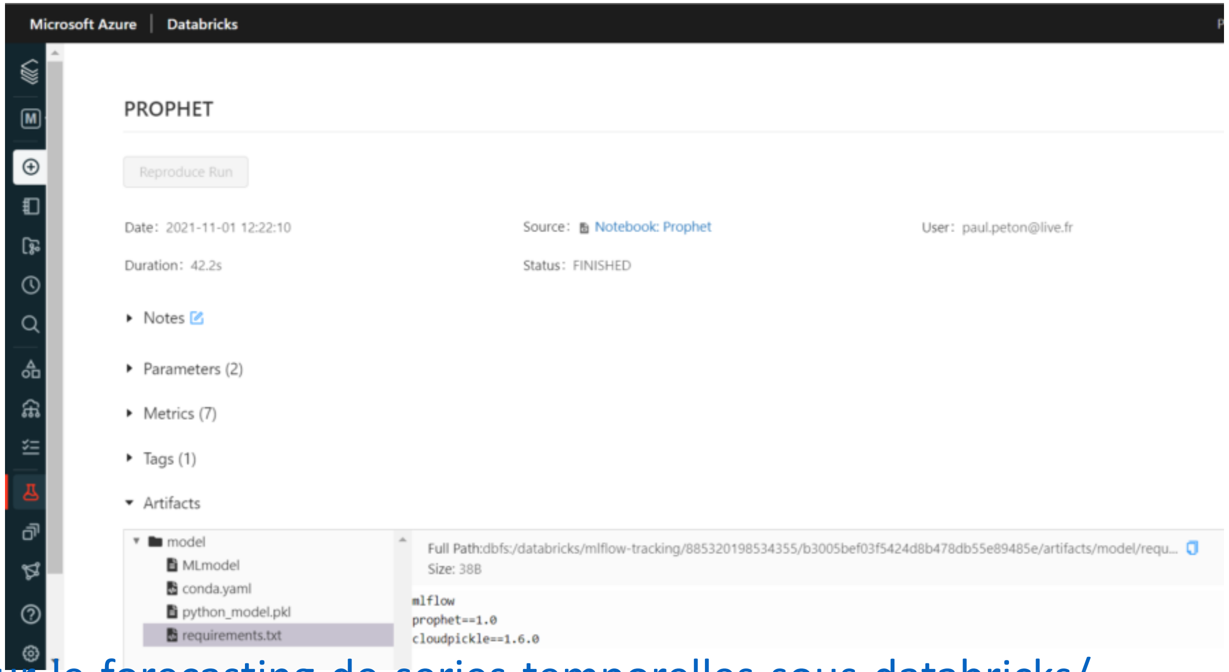
Ljung-Box : L'hypothèse nulle (H0) stipule qu'il n'y a pas auto-corrélation des erreurs d'ordre 1 à r. L'hypothèse de recherche (H1) stipule qu'il y a auto-corrélation des erreurs d'ordre 1 à r.

ADF : Le test augmenté de Dickey-Fuller ou test ADF est un test statistique qui vise à savoir si une série temporelle est stationnaire c'est-à-dire si ses propriétés statistiques (espérance, variance, auto-corrélation) varient ou pas dans le temps.

KPSS : vise à savoir si une série temporelle est stationnaire c'est-à-dire si ses propriétés statistiques (espérance, variance, auto-corrélation) varient ou pas dans le temps.







https://methodidacte.org/2021/11/automated-ml-pour-le-forecasting-de-series-temporelles-sous-databricks/

Questions pratiques ET fondamentales

- Ne pas se donner un horizon trop lointain
  - *Au tiers de l'historique disponible*
- Disposer de périodes complètes pour analyser les saisonnalités
  - *Avoir plusieurs occurrences complètes des périodes*
- Gestion du calendrier :
  - *Supprimer les 29 février ?*
  - *Comment gérer les semaines incomplètes (0 ou 53 ?)*

Questions pratiques ET fondamentales

- Comment séparer les datasets train et test ?
  - *Sur une date précise ?*
  - *Définir des rolling windows ?*
- Inflexion de tendances : difficile à prédire
  - *On reste sur la dernière tendance modélisée*
- La météo est rarement est un bon régresseur
  - *On a du mal à dire quel temps il fera dans une semaine !*
- Comment prendre en compte l'effet COVID / confinement ?
  - *Ce sujet mériterait un meetup complet !*

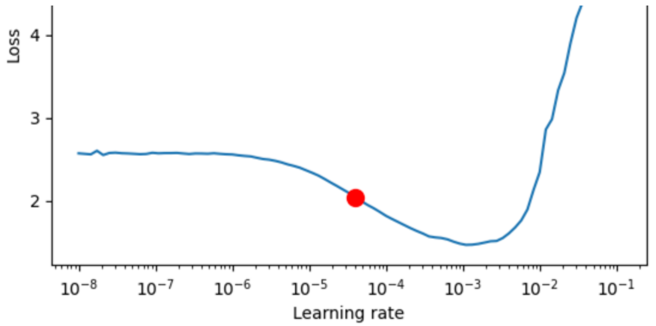
Annexes

Hyperparameters have smart defaults.

Loss Function is **Huber loss**,
unless user-defined.

$$L_{huber}(y, \hat{y}) = \begin{cases} \frac{1}{2\beta}(y - \hat{y})^2, & \text{for } |y - \hat{y}| < \beta \\ |y - \hat{y}| - \frac{\beta}{2}, & \text{otherwise} \end{cases}$$

The learning rate is approximated
with a **learning-rate range test**.



Batch size and epochs are approximated
from the dataset size.

We use **one-cycle policy**
with AdamW as optimizer for simplicity.