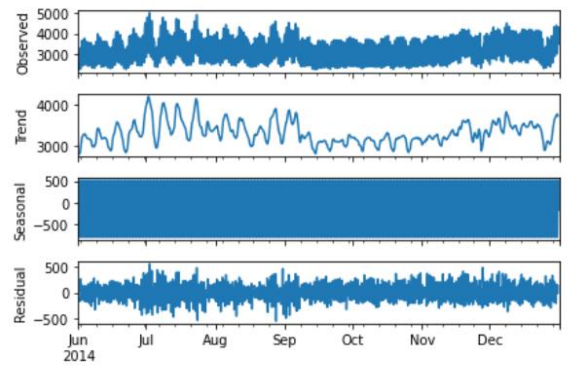
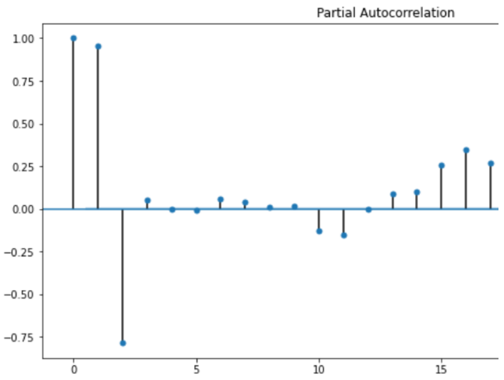
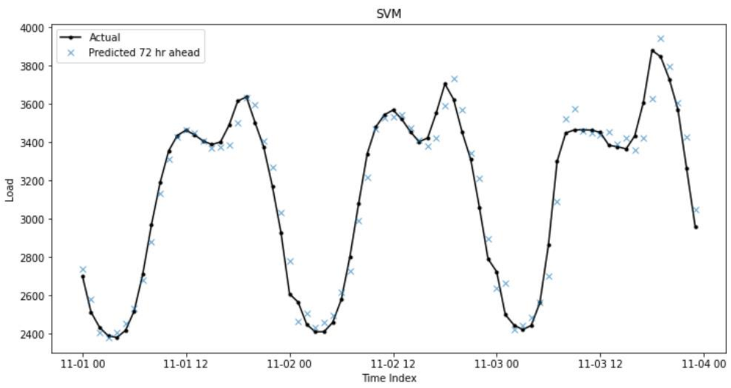
Data Driven Engineering I: Machine Learning for Dynamical Systems

Analysis of Dynamical Datasets I: Time Series

Institute of Thermal Turbomachinery
Prof. Dr.-Ing. Hans-Jörg Bauer







Dynamical Datasets I: Time Series

Karlsruhe Institute of Technology

Outline

* Time Series: Overview
* Statistical Models for time series
* State space models ⇒ DDE II
* Machine Learning Part I
* Machine Learning Part II

Time Series Analysis

* Time Series
  - Analysis
    - identify patterns
  - Forecasting
    - Modelling

Train on past → Predict future

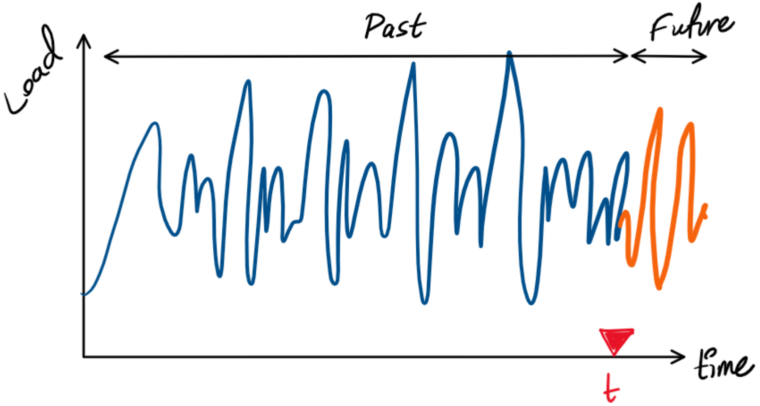
Relatively new field:

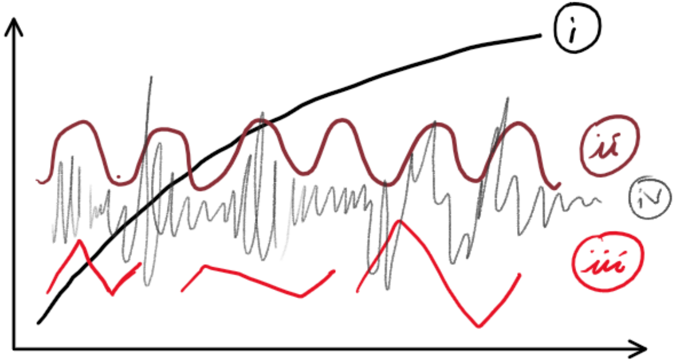
- Forecasting, ~old as humankind
- Autoregressive model ~ 1920s
- Box – Jenkins Model ~ 1970

"All models are wrong, but some are useful."
G. Box

Time Series Analysis

* Components of time series





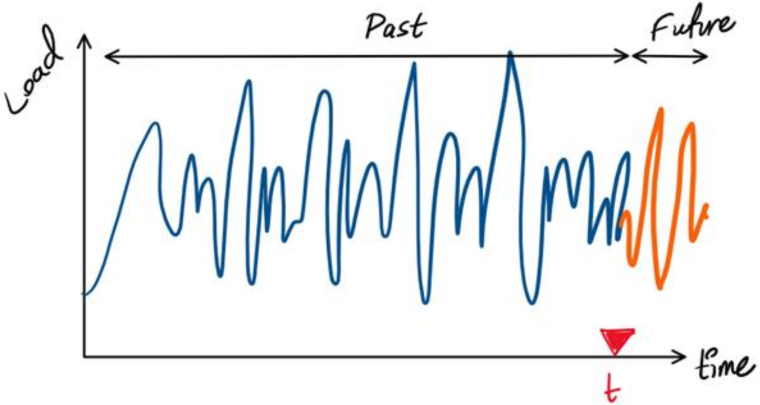
i) Long term trends

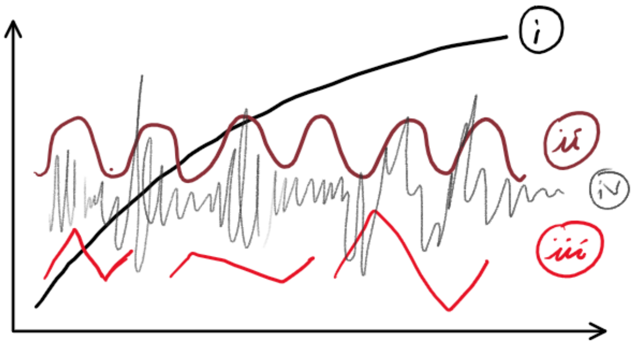
ii) ST seasonal variations

iii) Cyclic variations

iv) Random fluctuations

Time Series Analysis





* Components analysis

⇩

" Stationarity "

[$\bar{X}$ & $\sigma \neq f(t)$]

~ Strong ~

[$\bar{X}$ & auto covariance $\neq f(t)$]

~ weak ~

* Forecasting :

☑ There is an ordered relationship between observations

$t_{i-j} \rightarrow t_i$

Before we begin :

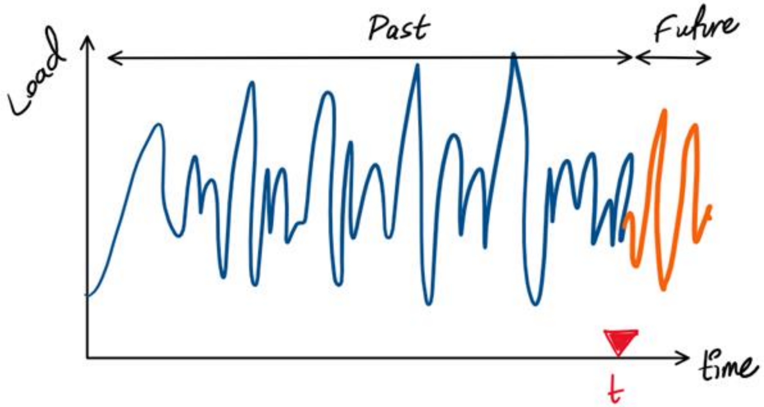
* horizon of your model (short term vs. long term)
* level of granularity you need (Δt_i)
* Univariate or multivariate models

$P_1, P_2, P_3 \ldots$

$\begin{pmatrix} P_1 \\ T_1 \end{pmatrix}, \begin{pmatrix} P_2 \\ T_2 \end{pmatrix}, \begin{pmatrix} P_3 \\ T_3 \end{pmatrix}, \begin{pmatrix} P_4 \\ T_4 \end{pmatrix} \ldots$

Before we begin :

* Sliding Window



time	Load
0	321
1	316
2	314
3	
4	318
...	

$\}\ y(t)$

$$y=f(x)$$

Before we begin :

* Sliding Window

$$y = \begin{bmatrix} 101 \\ 14 \\ 46 \\ 84 \\ 72 \\ 13 \end{bmatrix}$$

W = 1

x	y
NaN	101
101	14
14	46
46	84
84	72
72	13

W=2

X_1	X_2	y
NaN	NaN	101
NaN	101	14
101	14	46
14	46	84

...

Before we begin :

* Single / multistep forecasting

(1) Direct multi-step :

N models $y = f(x)$

(2) Recursive multi-step :

1 model

$1 \to 2, \; :2 \to 3, \; :3 \to 4, \ldots \; :N-1 \to N$

$1 \to 2, \; 2 \to 3, \; 3 \to 4, \ldots \; N-1 \to N$

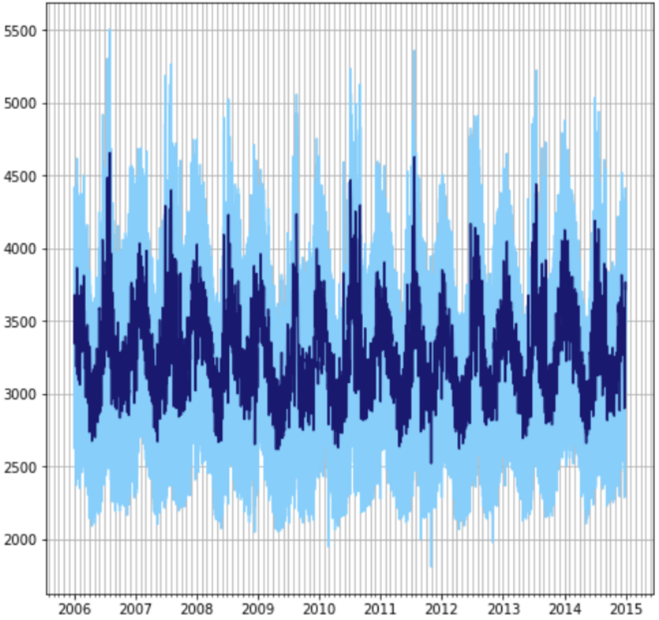
(3) Multiple output : [history] → [future]

Time Series Analysis

Workflow template:

1) Understand the problem/business
2) Data exploration
3) Data preprocessing // feature eng.
4) Short list the models/algorithms
5) Train your model
6) Evaluation phase

Case: Energy Demand Forecasting



* 8 years data of Temp & Load ($\Delta t = hr$)

? Power Demand forecasting

Short Term

Long Term

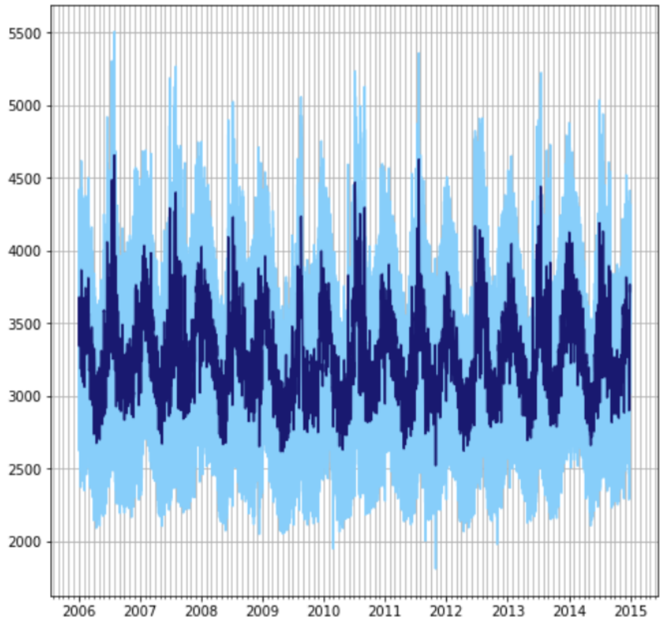
Case: Energy Demand Forecasting

* Short term load forecasting : ~ 1 hr to 24 hr ⇐ near past is used
  ~ demand / supply ⇐ Temperature is an important feature

* Long term LF : ~ 1 week to months } Planning & investment ⇐ Seasonal patterns
  ~ years ⇐ Long term trends
  ⇐ Climate Models

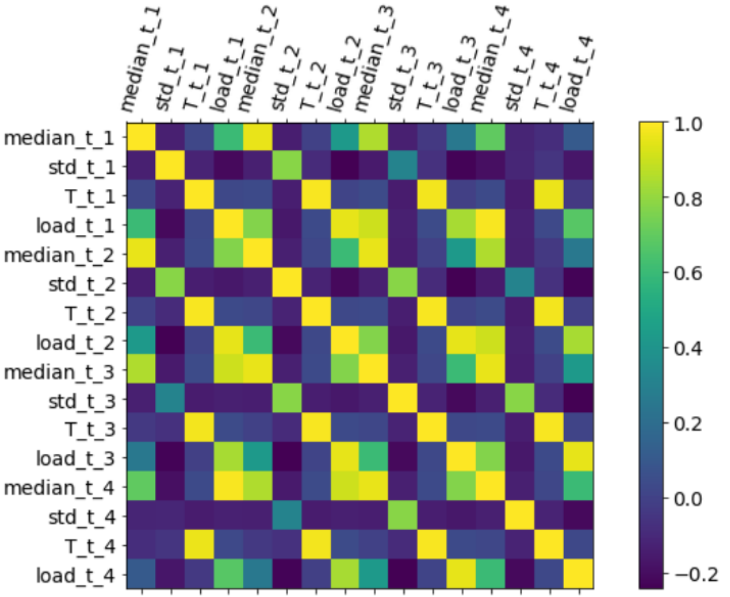
Case: Energy Demand Forecasting

Typical	STLF	LTLF
Horizon	1 hr – 2 days	≥ 1 months
Granularity	~ hr	~ hr – day
History Range	~ 2 years	~≥ 5 years
Accuracy	≤ 5% error	≤ 25% error
Forecasting freq.	~ hr to day	≥ month



Data Exploration: What we already know

- ☑ Basic statistics (mean, median, STD...)
- ☑ Plots ⇒ 1D : Temporal data
  - ⇒ 2D : Scatter plots
  - ⇒ Histograms
  - ⇒ Box plots, violin plots
- ☑ Correlation matrix



time: 314 ms (started: 2020-12-29 18:29:17 +00:00)

Data Exploration: Temporal Nature of data

① How to handle "time stamps"

	Date	Hour	load	T
0	01/01/2004	1	NaN	37.33
1	01/01/2004	2	NaN	37.67
2	01/01/2004	3	NaN	37.00
3	01/01/2004	4	NaN	36.33
4	01/01/2004	5	NaN	36.00

⇨

	load	T
2012-01-05 00:00:00	3167.0	19.00
2012-01-05 01:00:00	3014.0	22.33
2012-01-05 02:00:00	2921.0	22.33
2012-01-05 03:00:00	2874.0	22.00
2012-01-05 04:00:00	2876.0	21.67

Ateliers & Saveurs in Montreal

colab

Data Exploration: Temporal Nature of data

② Temporal data decomposition → Trend
→ Seasonality
→ Noise

Stationarity

⇓

how 'stable' your system
- [x] Intuition
- [ ] Tests

⇓

how much we should expect the past reflects itself on future ? ⇒ "Self Correlations"

colab

Data Exploration: Temporal Nature of data

(3) Feature Eng. for Time Series

- [ ] Date / time information

Data
- hour → f_1
- business hour / day → f_2
- holidays → f_3
- seasonal trends → f_4
- ...

Multivariate Model

Data Exploration: Temporal Nature of data

③ Feature Eng. for Time Series

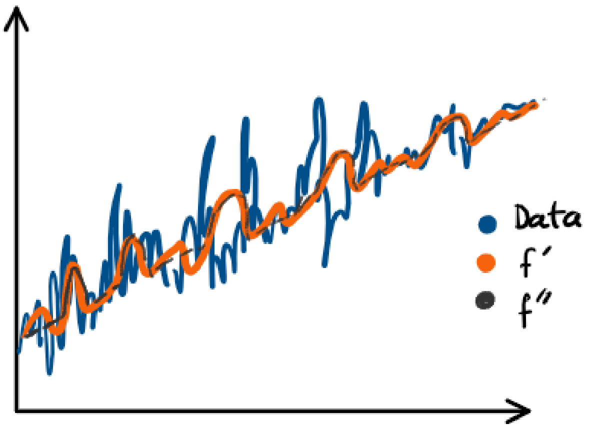
- ☐ Date / time information
- ☐ Window functions

Data → Sliding window → Mean → f_5

Sliding window → Median → f_6

Sliding window → min/max → f_7

...



colab

Data Exploration: Temporal Nature of data

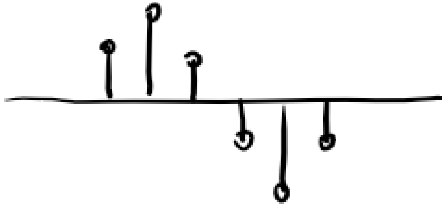
④ Self / Auto Correlations in temporal data

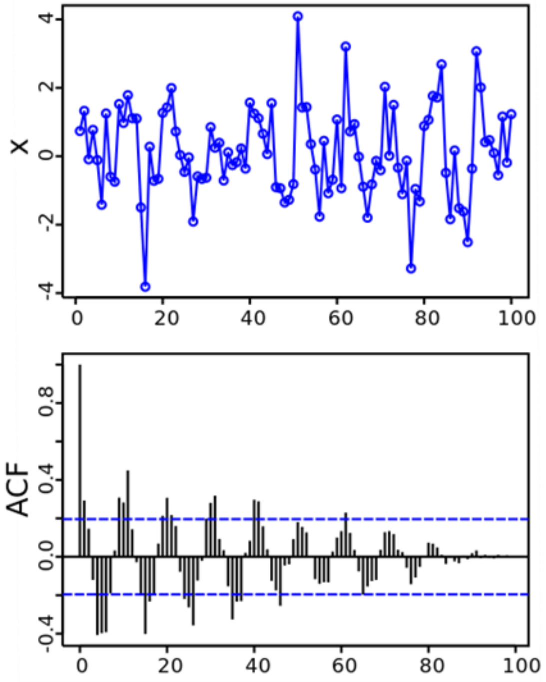
- ☐ Autocorrelation function (acf)
- ☐ Partial ACF (pacf)

☑ How data points are linearly related as a function of time difference.

Data Exploration: Temporal Nature of data

* ACF ⇒ it preserves the periodicity
* ACF = 1 @ lag = Ø [self correlated]
* ACF (white noise) → Ø
* ACF is symmetric

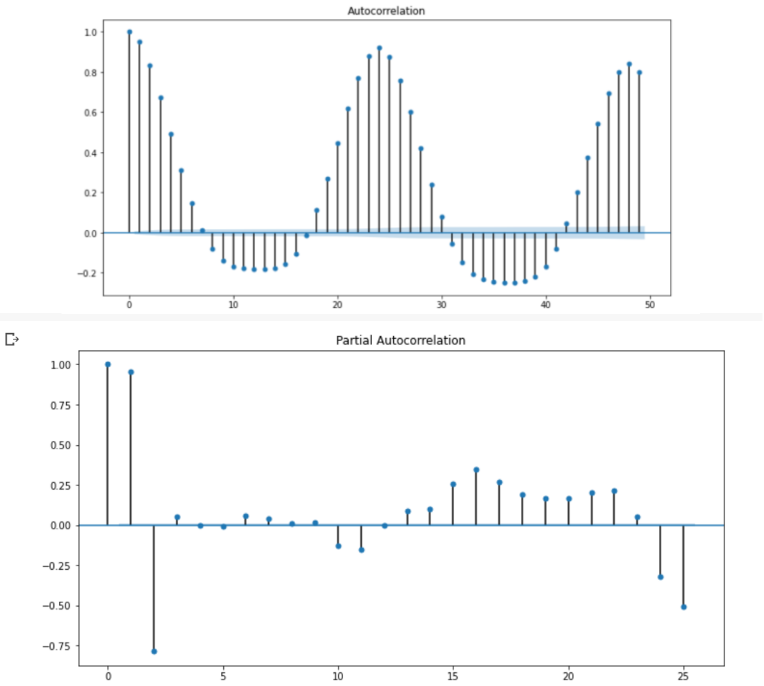




Data Exploration: Temporal Nature of data

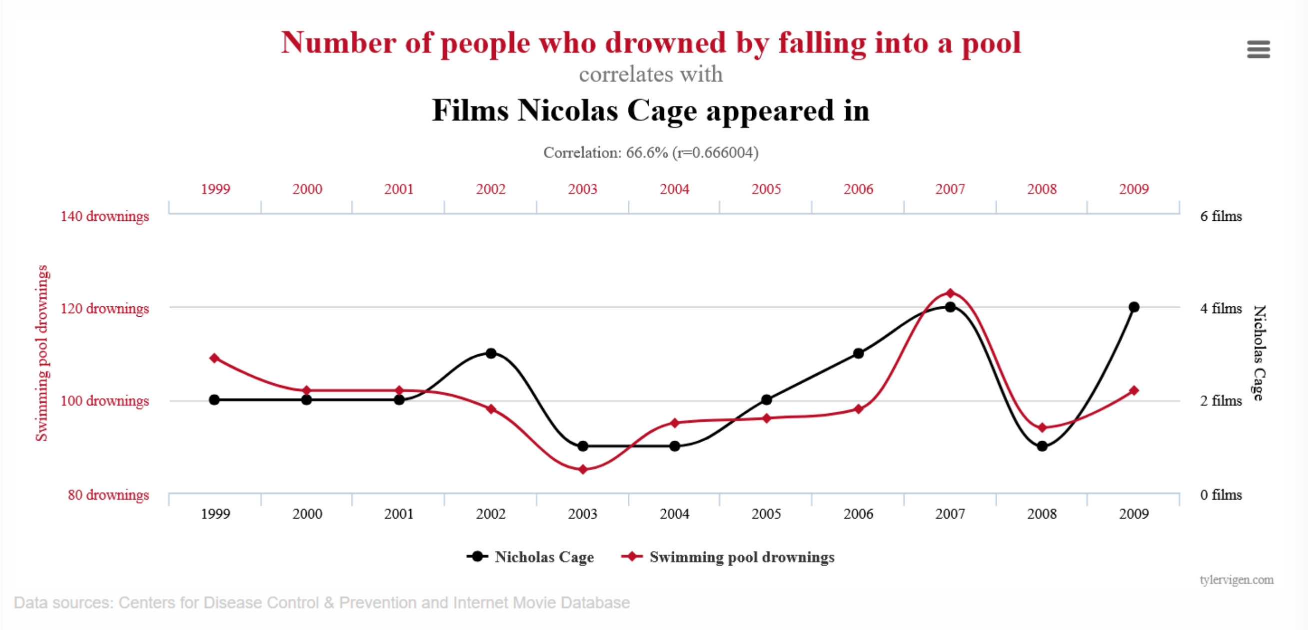
* pACF → which time lag is "informative"
  ~ filters periodic behavior

* pACF → determine the "order" of a model



time: 366 ms (started: 2020-12-29 14:54:10 +00:00)

Spurious Correlations



colab

Overview of Statistical Models

* **Obj:** (i) find time-related trends
  (ii) find seasonality
  (iii) find auto-corr. (corr. wrt. time)

AR Model: Auto Regressive

* $y_t = a_0 + a_1 y_{t-1} + Err \Big\}$ history := 1 lag

* Order (p) := history info; $p=2$

  $y_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + Err$

* Order $\leftarrow$ "pacf"

Overview of Statistical Models

AR-I-MA : AR - Integrated-MA

* add differencing ⇒ Remove trends
  " baseline correction "

* ARIMA $= f(p,d,q)$
  $(0,0,0) \rightarrow$ white noise
  $(0,1,0) \rightarrow$ random walk
  $(0,1,1) \rightarrow$ exp. smoothing

* SARIMA := Seasonal ARIMA
  ▷ Adjacent points in time can have influence on one another

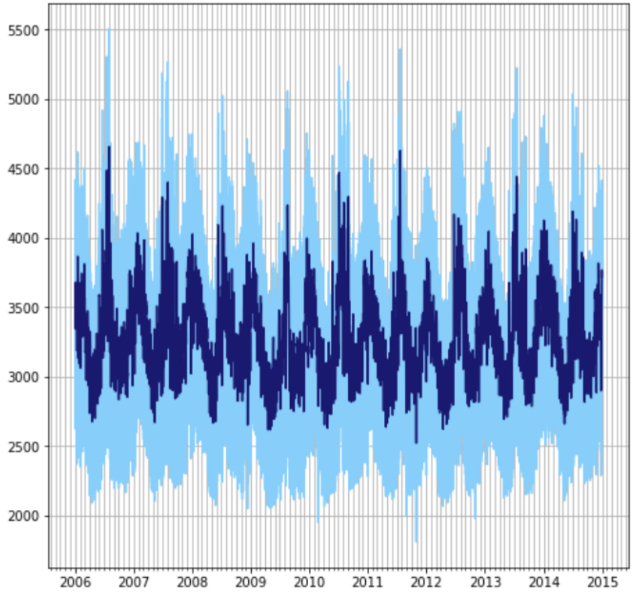
MA : Moving Average

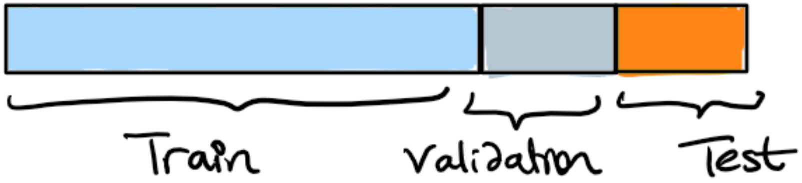
* $y_t = a_0 + E_t + a_1 E_{t-1} + a_2 E_{t-2} + \cdots + a_q E_{t-q}$
  ↓
  Errors dissipate in time

* $q \leftarrow$ order

* $q \leftarrow$ ACF
  " stop error propagation sharply "

Model Training





Train → model fitting

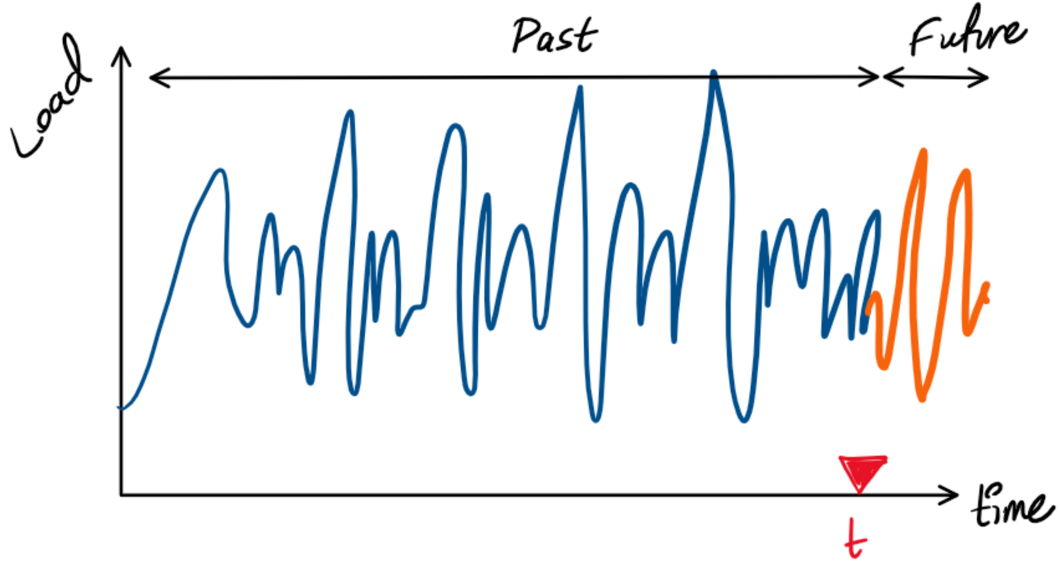
Valid → hyper-parameter tuning

Test → Performance evaluation

Karlsruhe Institute of Technology

colab

how can we use ML algorithms?



In M.L.:

* [x] → [y]

⇨

time	Load
0	321
1	316
2	314
3	318
4	
...	

$\}$ y(t)

how can we use ML algorithms ?

* Time Series ⇒ "Supervised learning task" [batch // real-time]

$$y = \begin{bmatrix} 101 \\ 14 \\ 46 \\ 84 \\ 72 \\ 13 \end{bmatrix} \rightarrow$$

x	y
NaN	101
101	14
14	46
46	84
84	72
72	13

W = 1

→

x_1	x_2	y
NaN	NaN	101
NaN	101	14
101	14	46
14	46	84

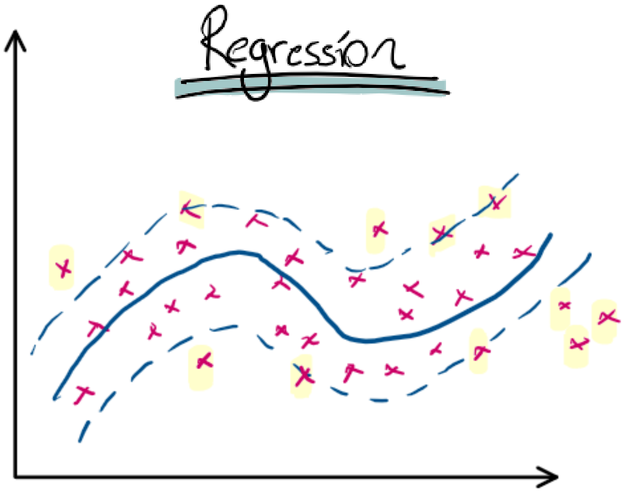
W=2

...

Use any regression model !

Model Selection: SVM for Regression

– SVM –



* Fit as many instance as possible
* "Street" width is controlled by margin ϵ.
* Convex optimization problem:

- [x] C
- [x] ϵ
- [x] Kernel

colab

Additional Notes