mikrocentrum
opleiden • ontmoeten • ondernemen

Simcha van Helvoort

Dag 2

AI voor Engineers

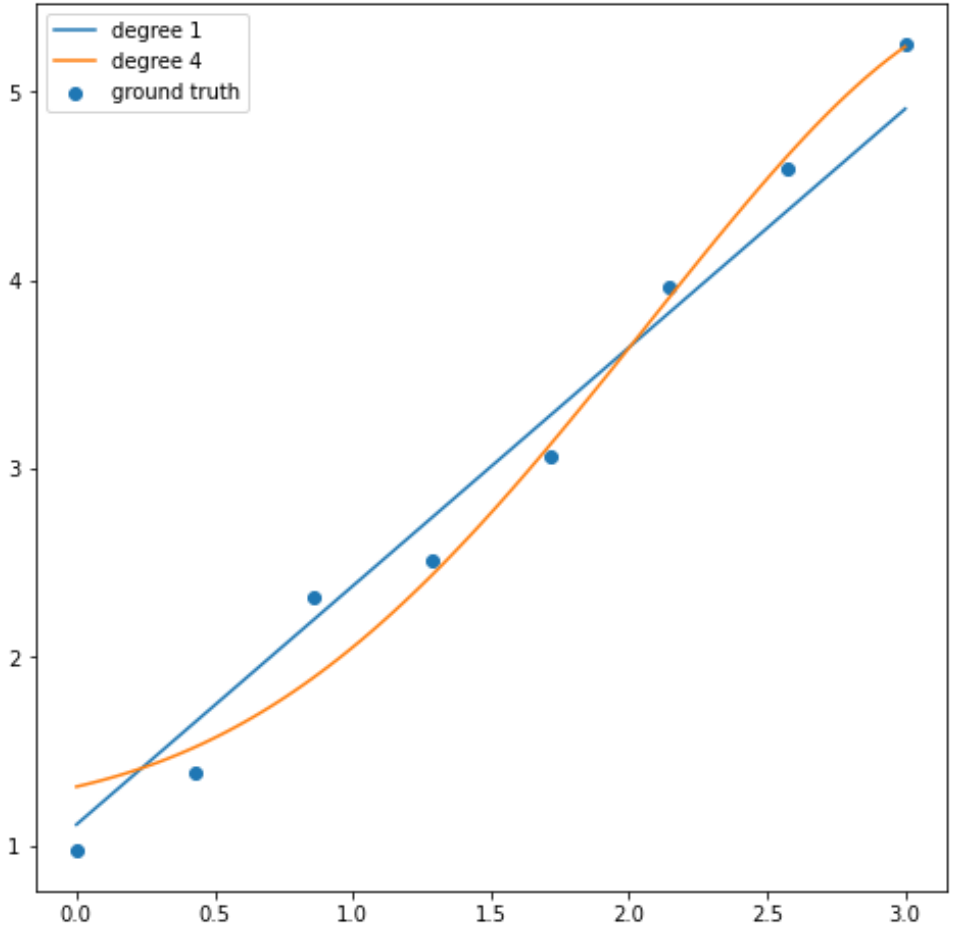
Leerdoelen

- De cursist snapt het verschil tussen ruwe data en input data
- De cursist kan van ruwe data, input data maken in Python
- De cursist kan omgaan met feature scaling, uitschieters en andere data cleaning technieken
- De cursist snapt wat overfitting is en hoe dit tegengegaan moet worden
- De cursist kan met ruwe data een geschikt model maken om specifieke analyses te doen
- De cursist weet hoe een model geëvalueerd moet worden en kan op basis daarvan de volgende stappen voor het model bepalen

Terugblik

Wat is overfitting?

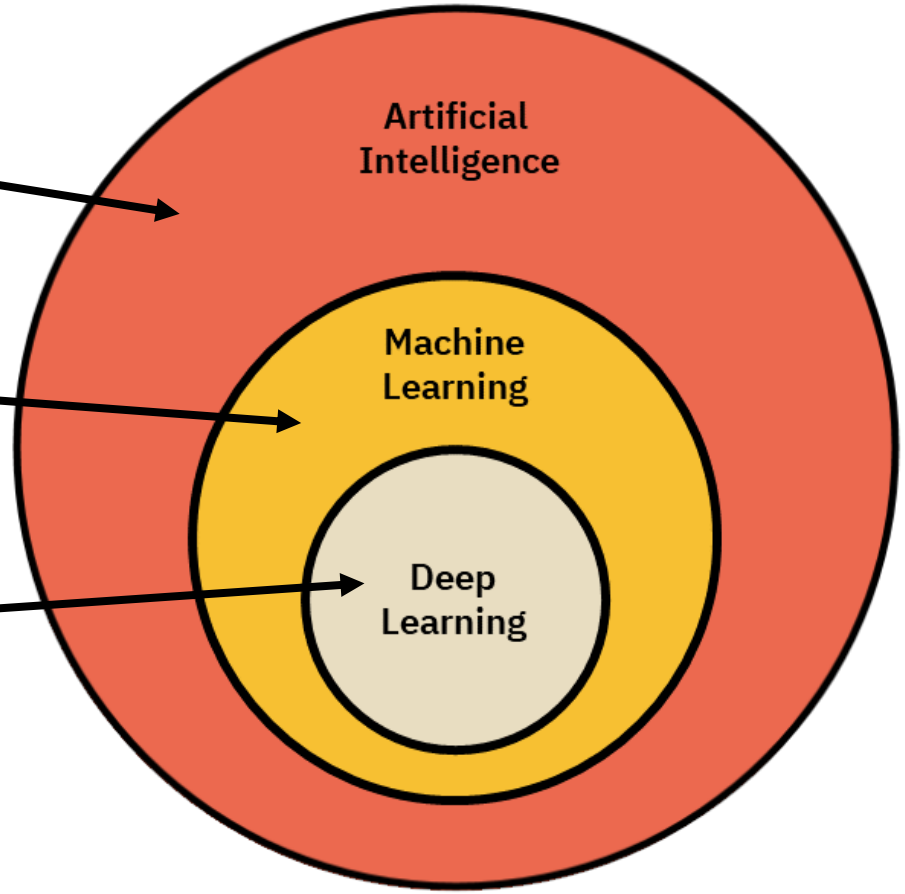
Wat kunnen we er tegen doen?



Het namaken van intelligentie

Het namaken van leren

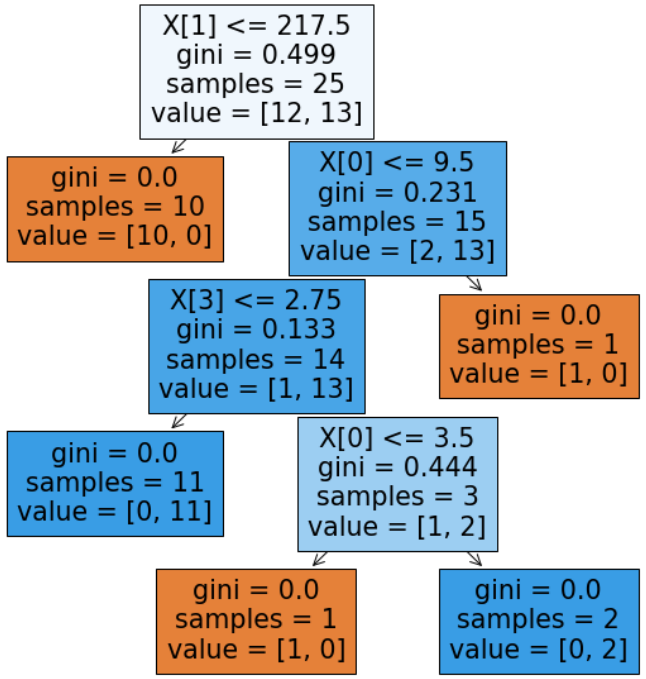
Het namaken van hersens

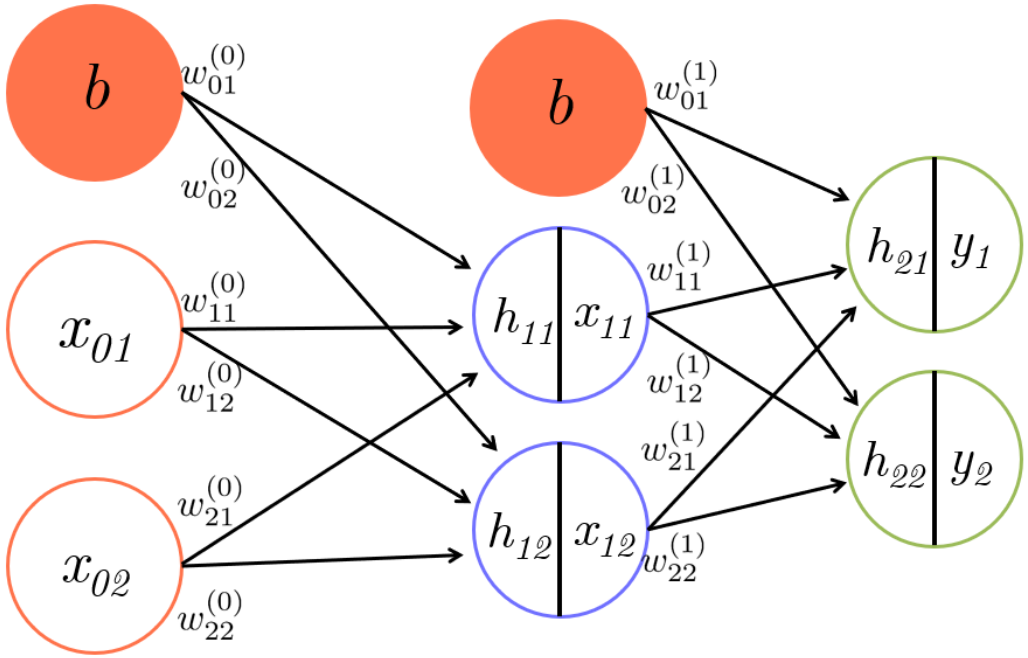


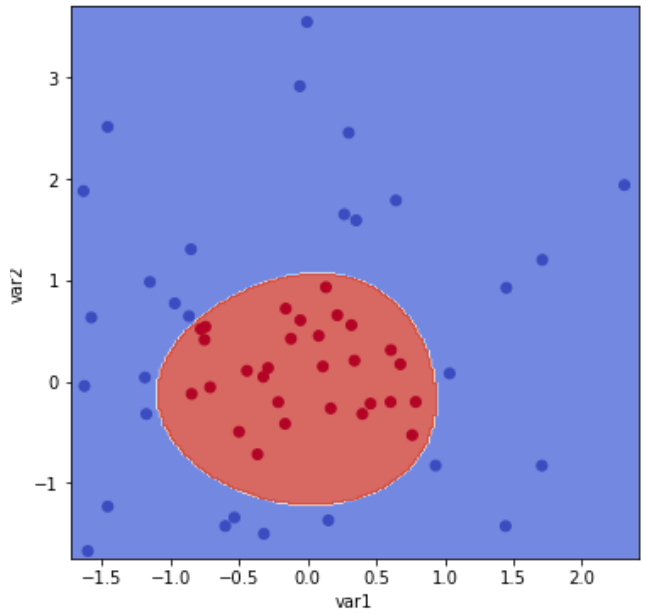
Terugblik

Makkelijk/Simpel

Bij welke methode hoort dit voordeel?

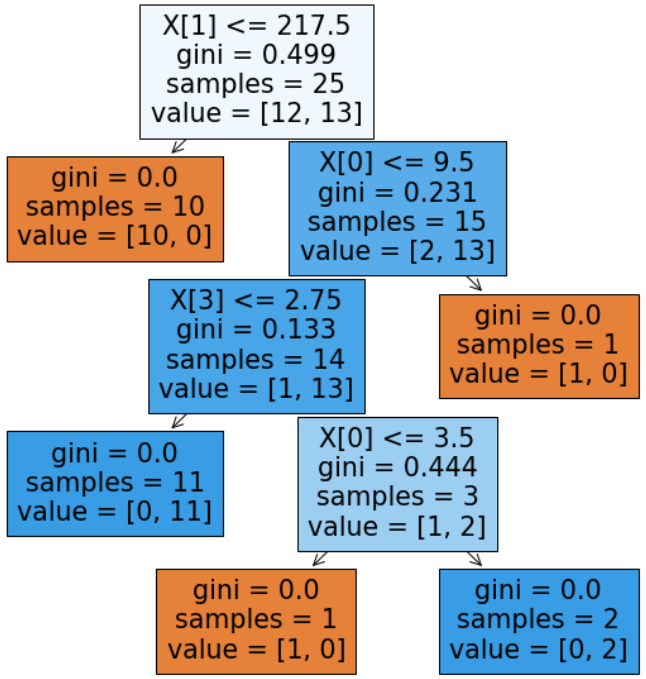


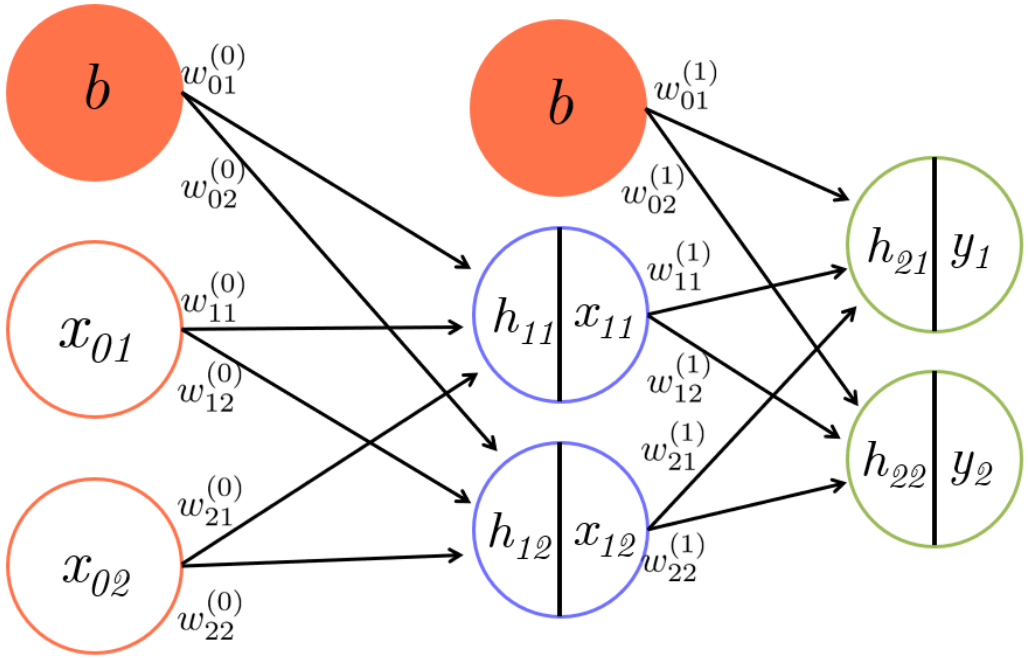


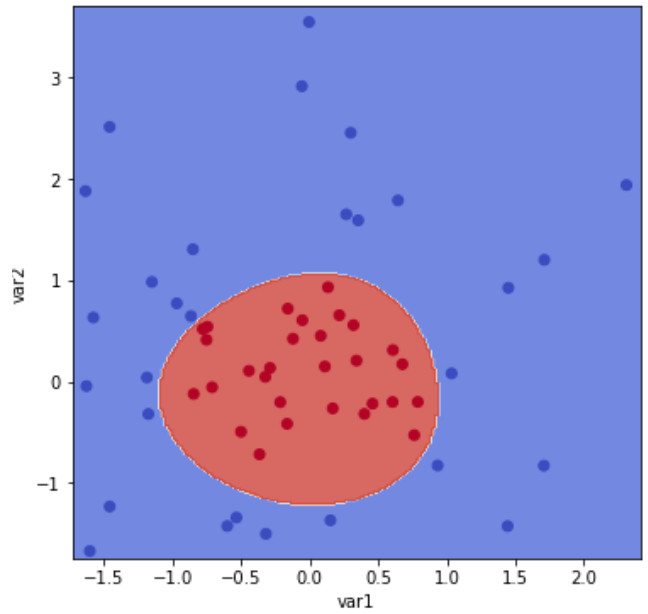


Kan meerdere variabelen in 1x vergelijken

Bij welke methode hoort dit voordeel?

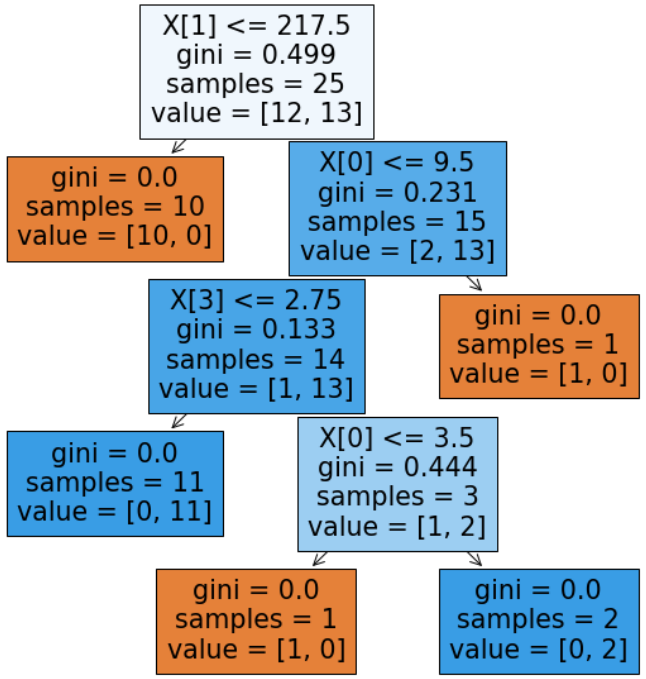


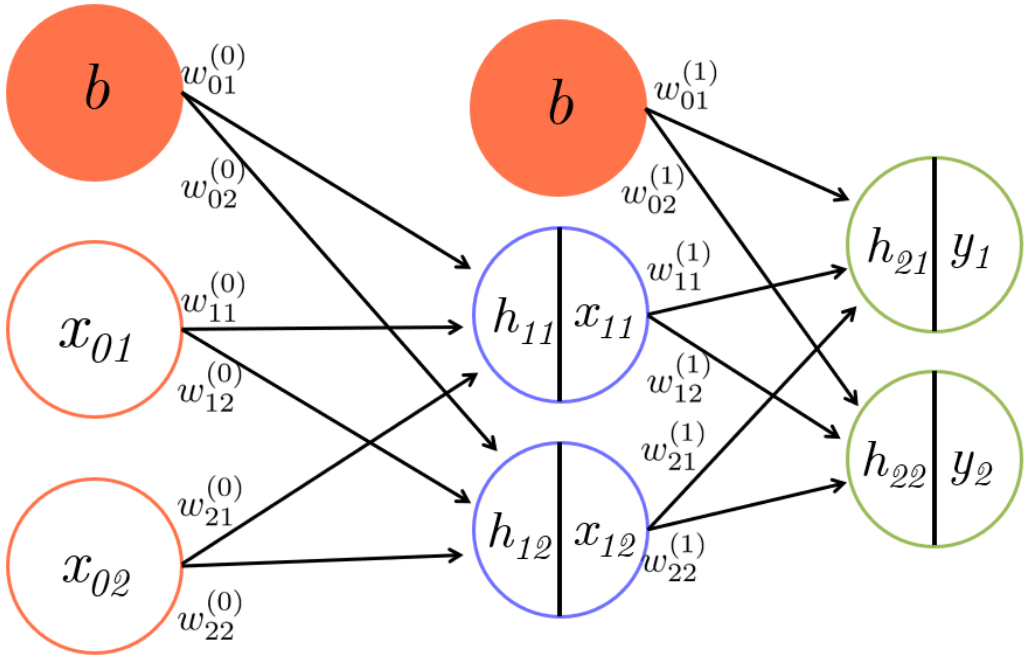


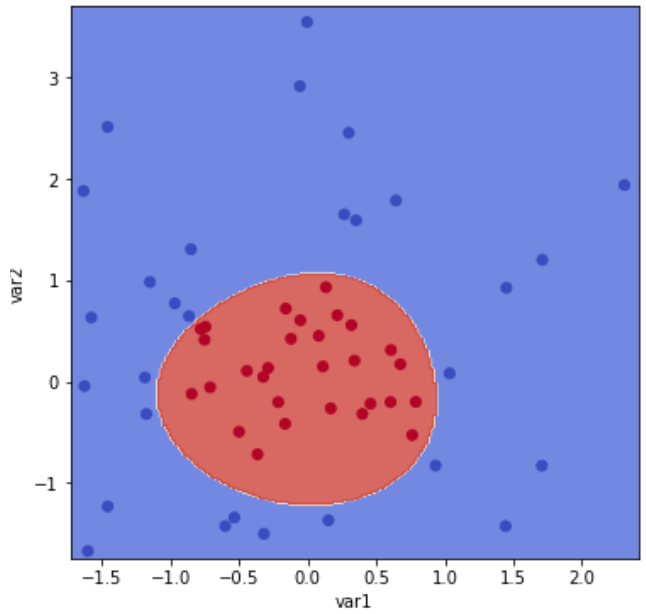


Lastig om hyperparameters te bepalen

Bij welke methode hoort dit nadeel?

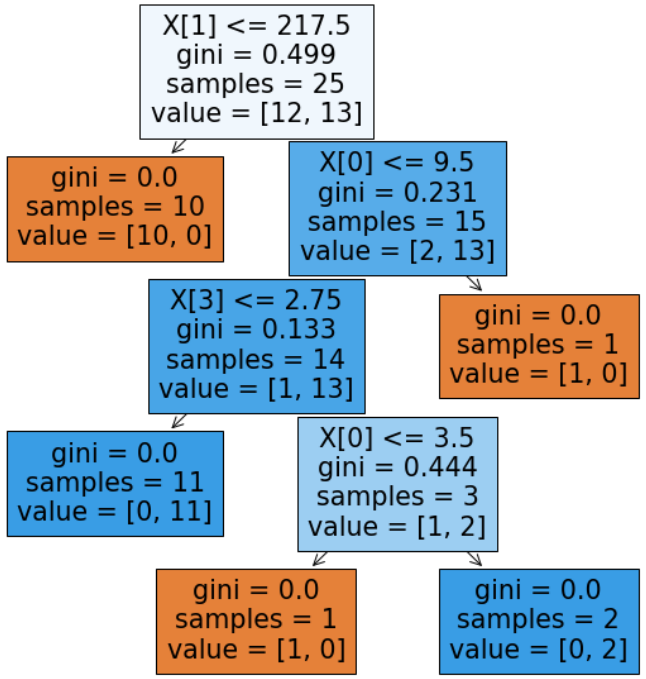


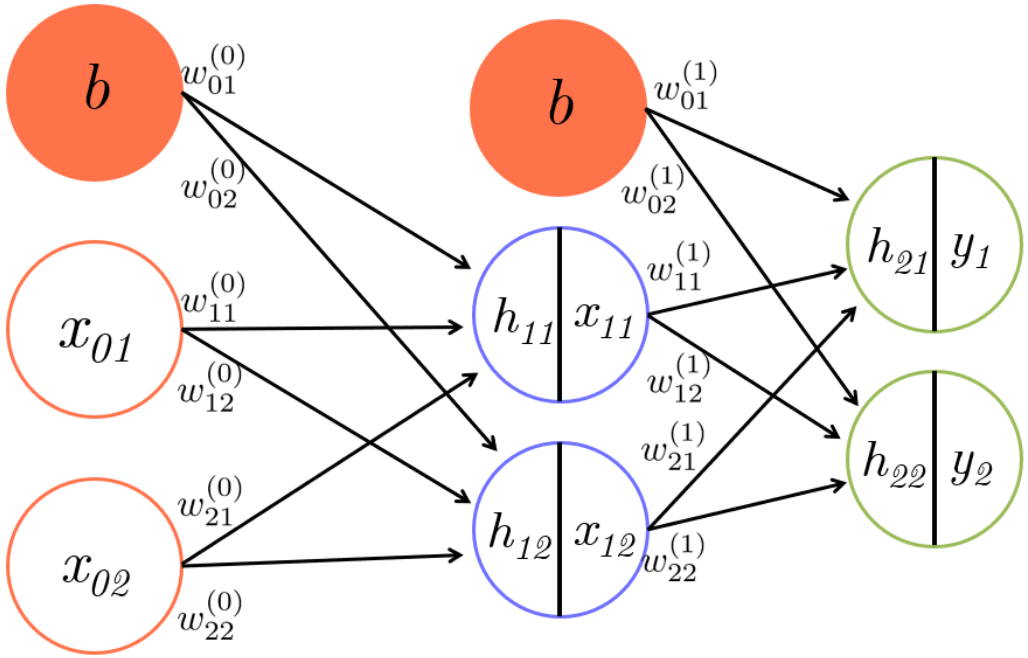


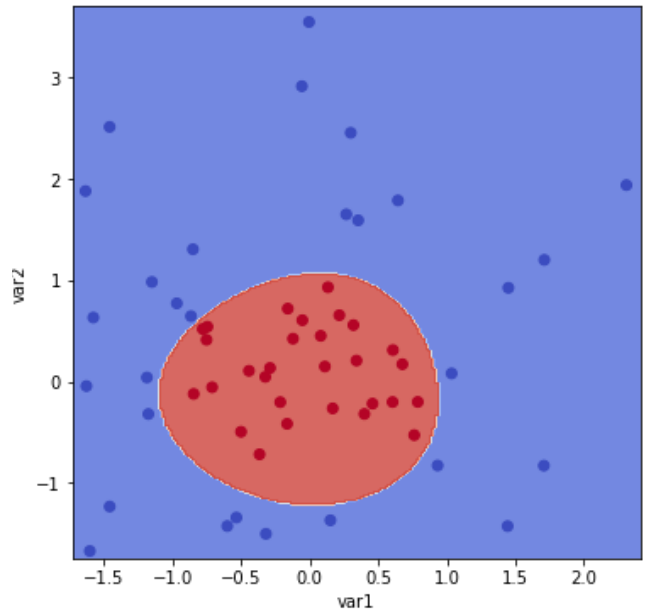


Gevoelig voor ruis

Bij welke methode hoort dit nadeel?

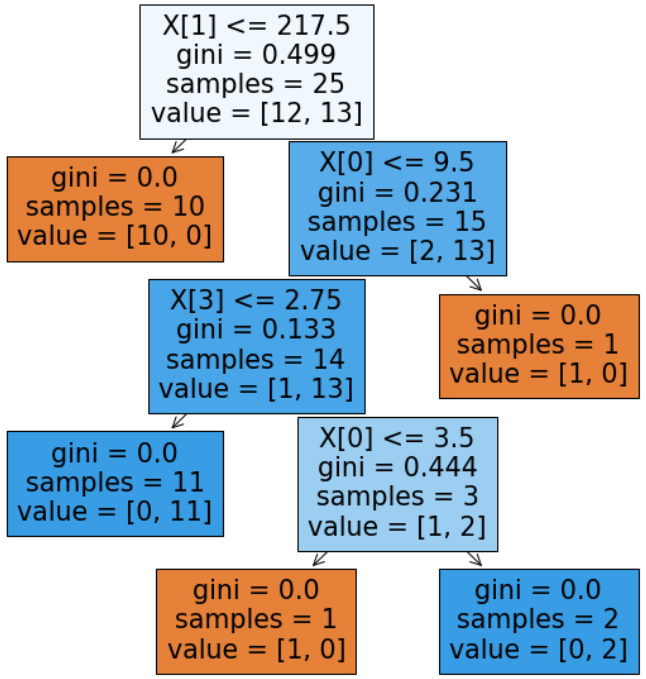


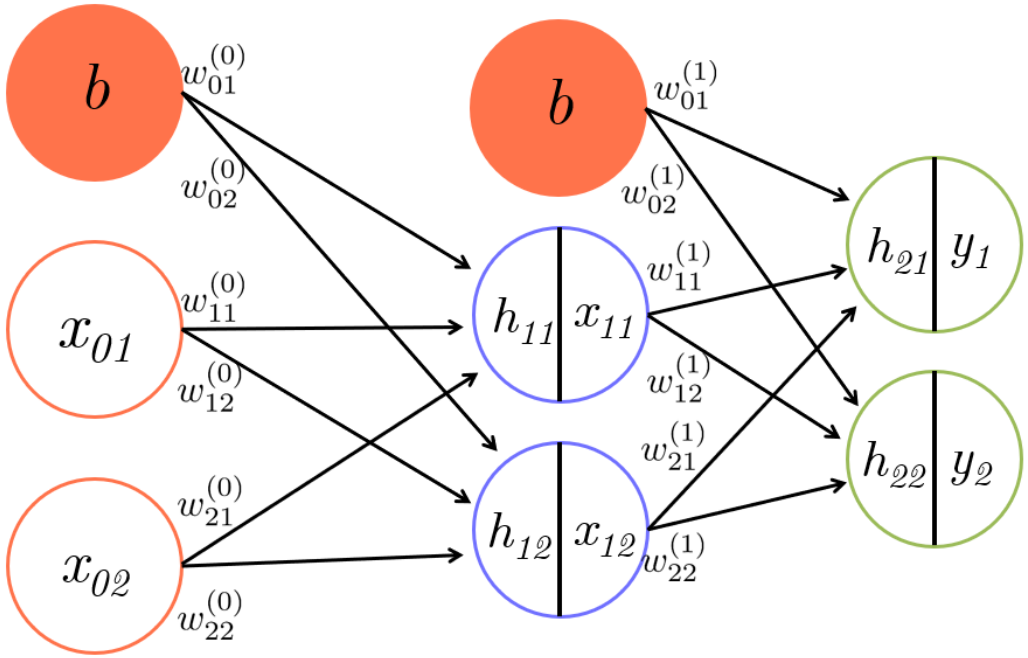


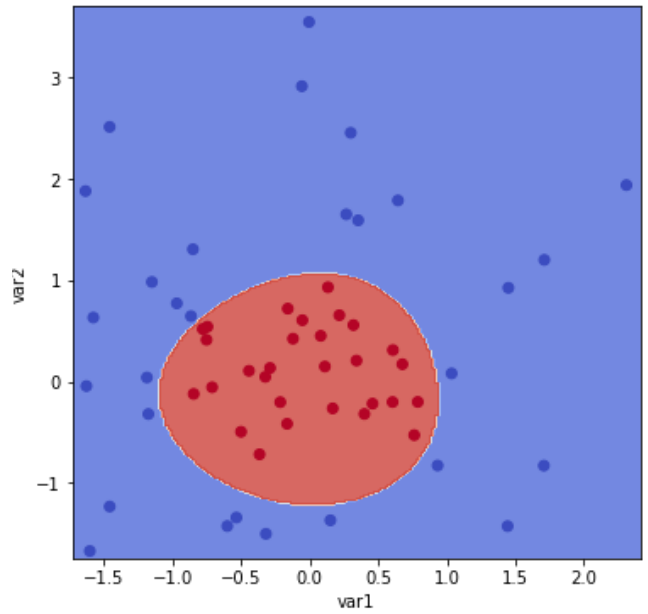


Werkt goed met weinig data

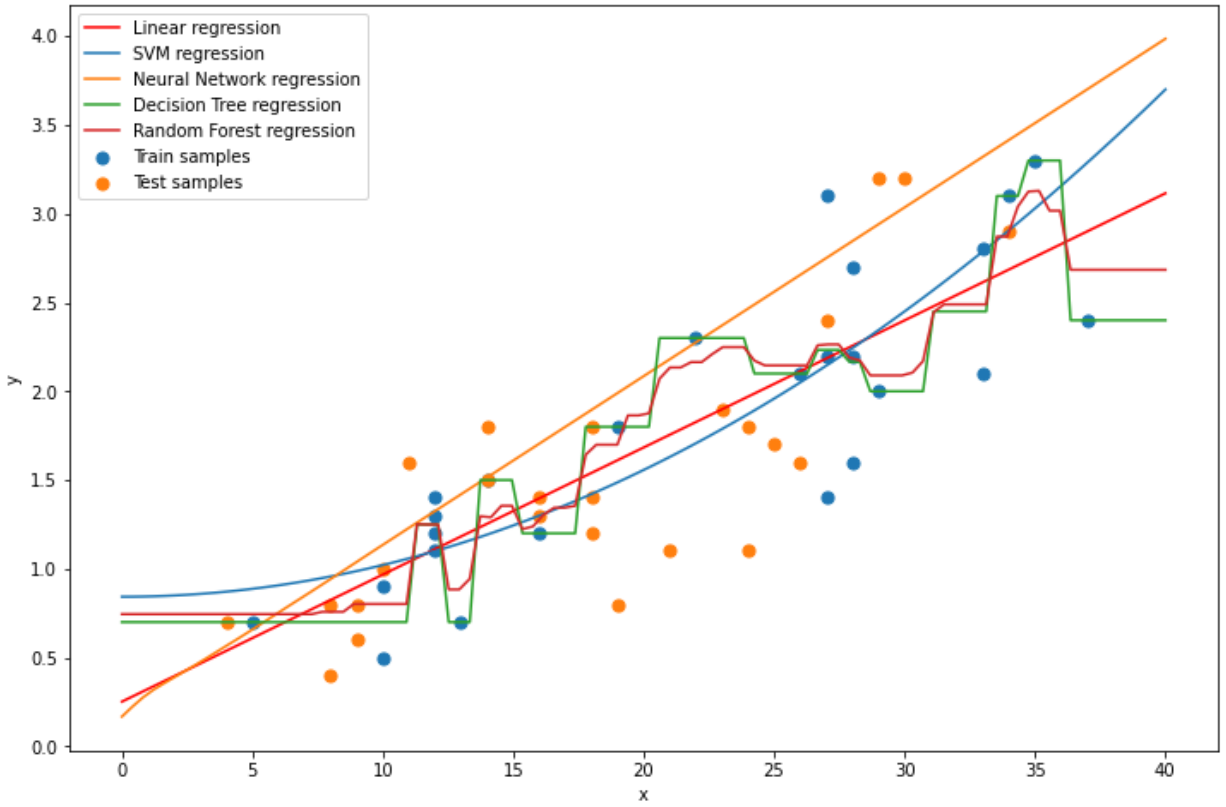
Bij welke methode hoort dit voordeel?



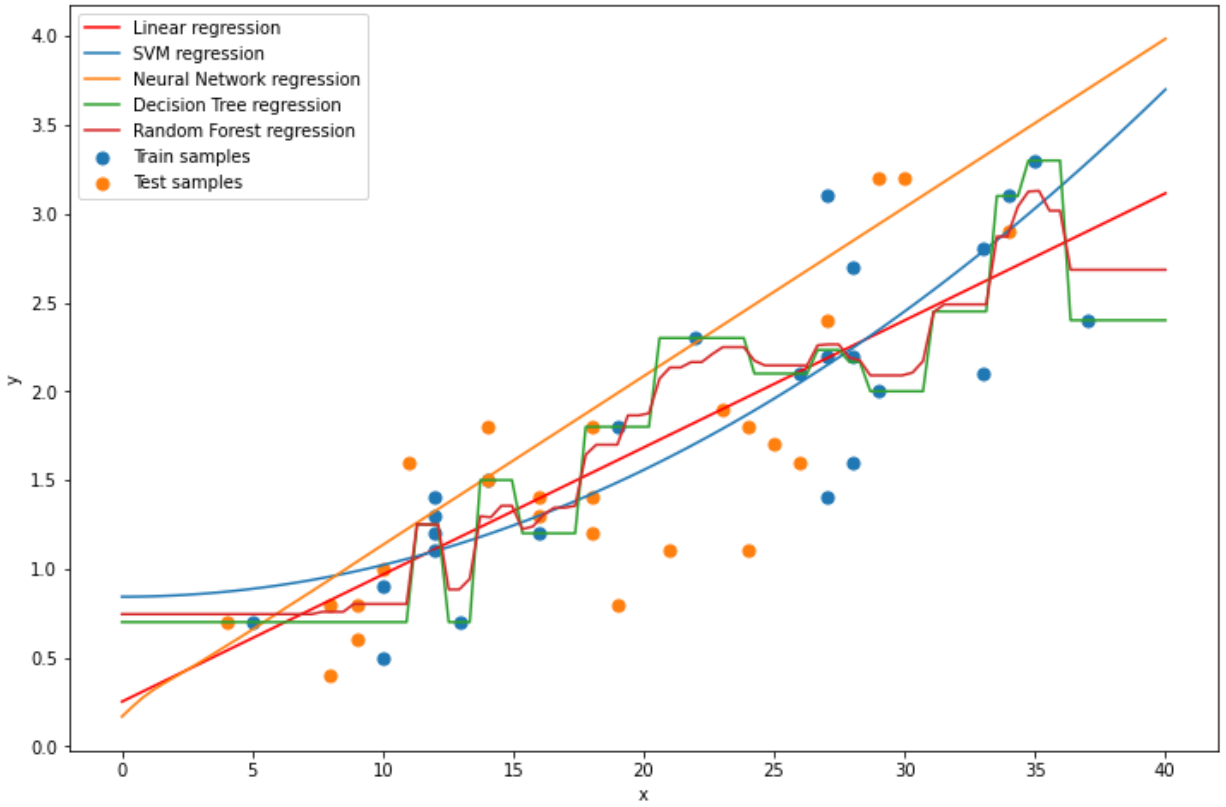




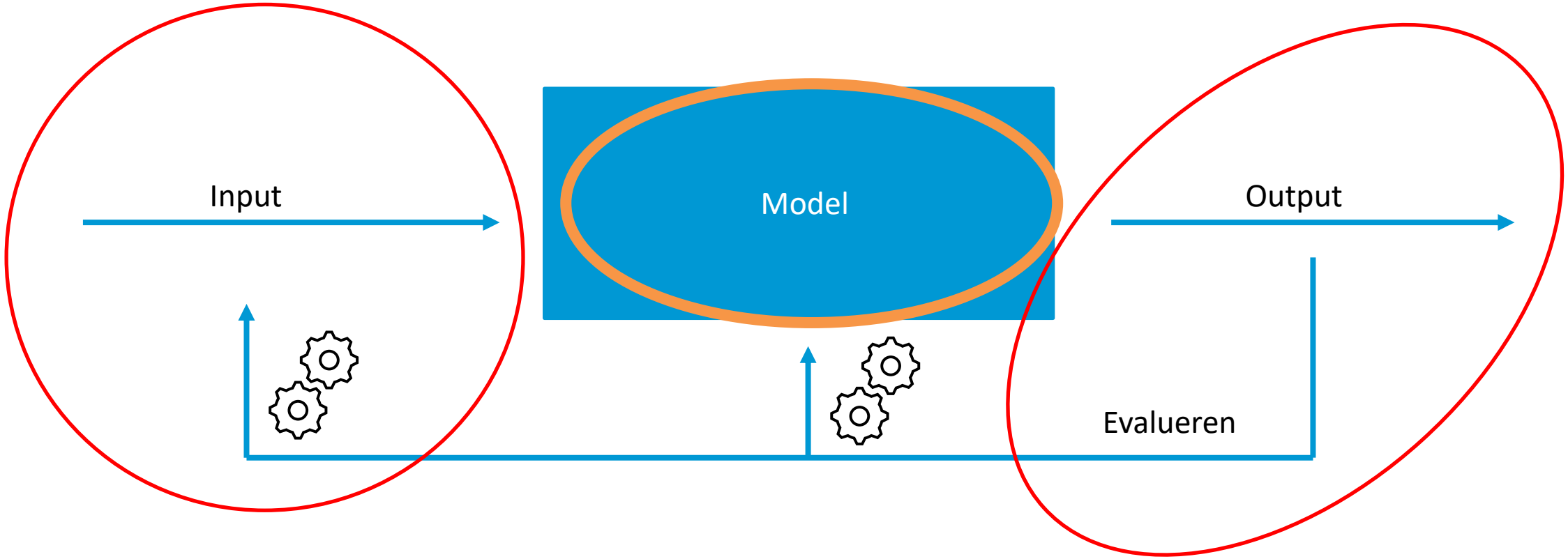
Wat is het voordeel van SVM regressie t.o.v. lineaire regressie?



Wat voegt Ridge regressie toe aan lineaire regressie?



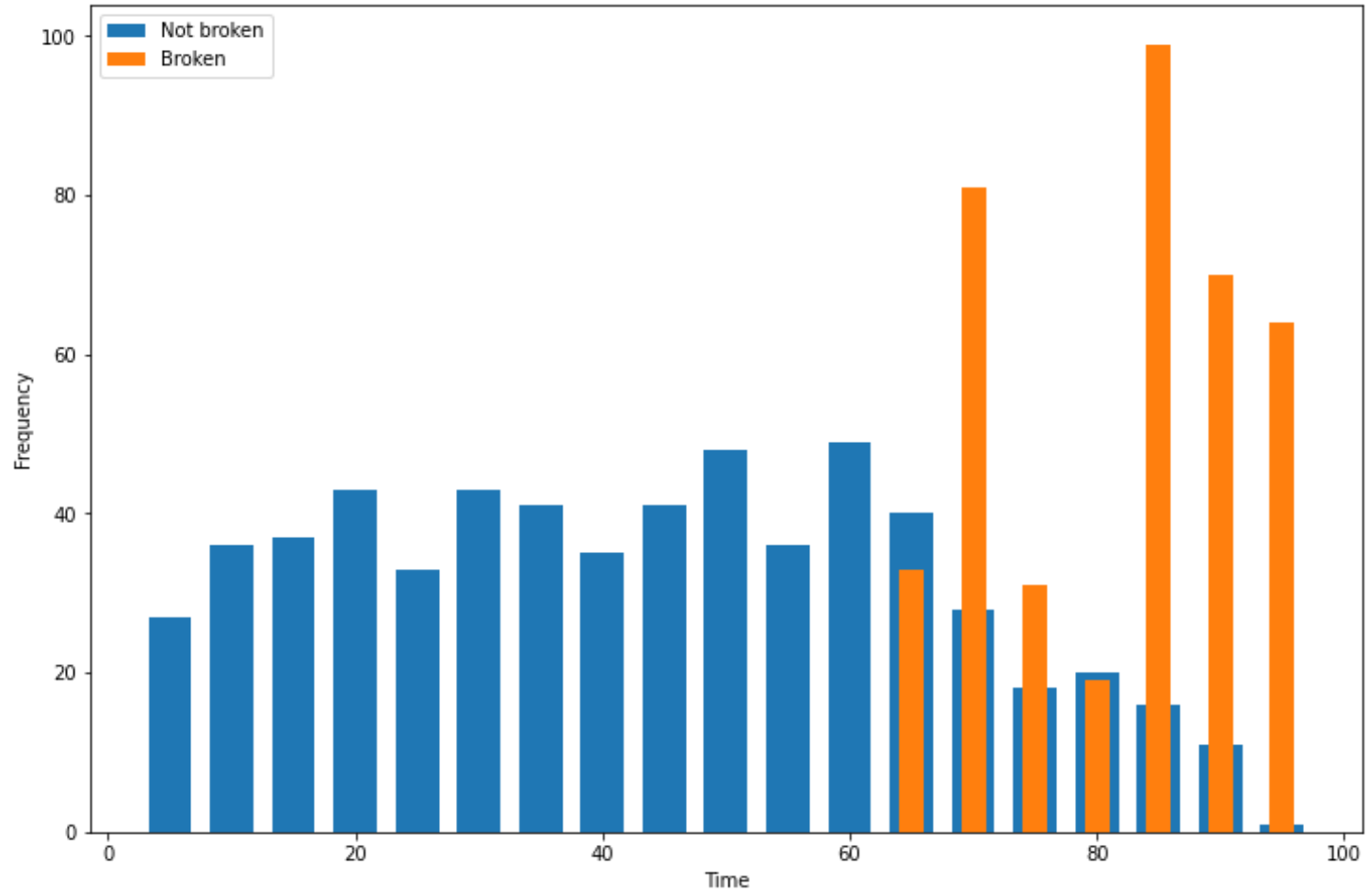
Vandaag



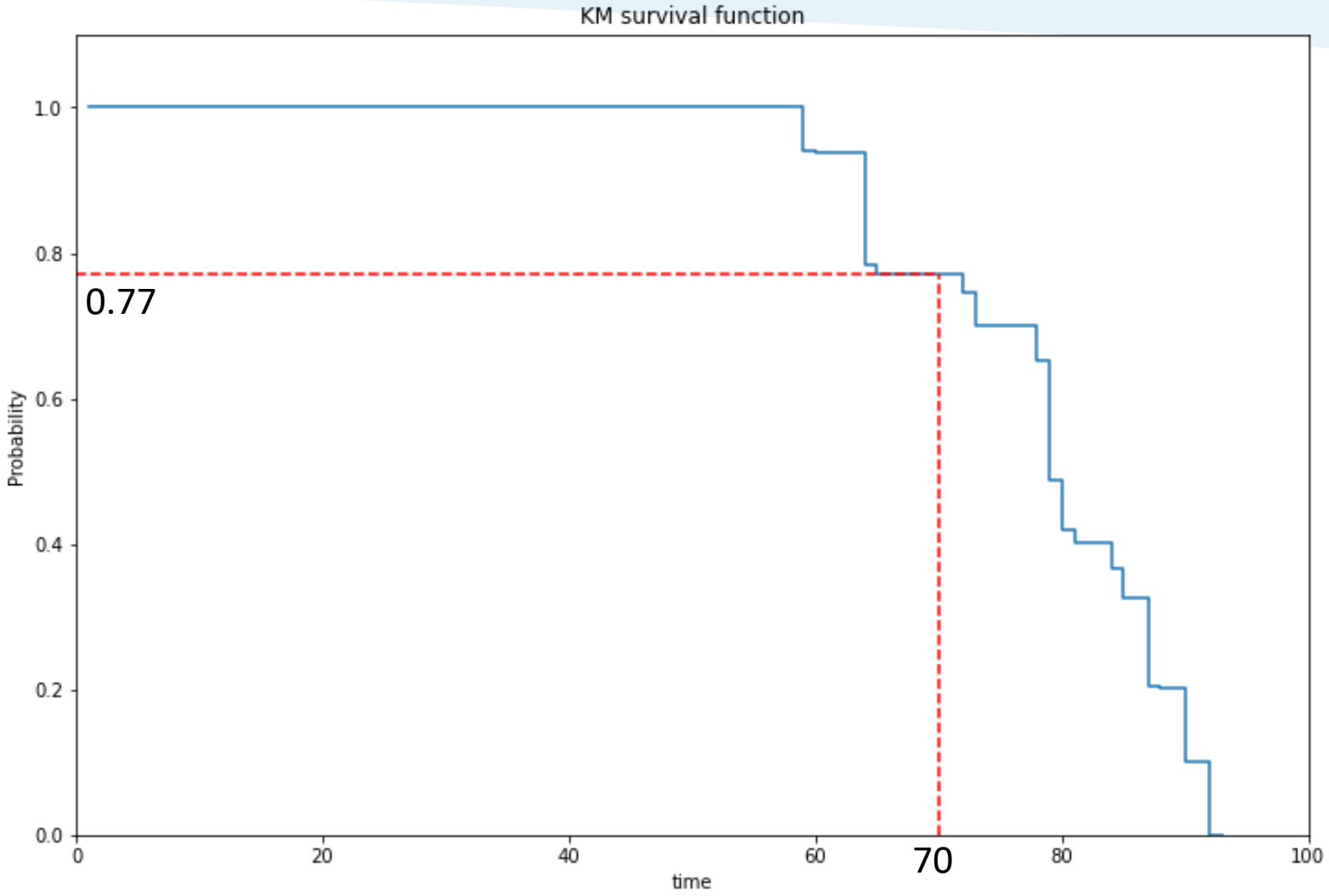
SURVIVAL ANALYSIS

Survival analysis

Het analyseren van “time to event” data. Oftewel: voorspellen wanneer *iets* gebeurd.



Survival Analysis



Dataset – Broken machinery

Hoelang is elk element gebruikt tot nu toe of totdat het kapot ging (in weken)

~40% is kapot

	lifetime	broken	pressureInd	moistureInd	temperatureInd	team	provider	censored
0	56	0	92.178854	104.230204	96.517159	TeamA	Provider4	1
1	81	1	72.075938	103.065701	87.271062	TeamC	Provider4	0
2	60	0	96.272254	77.801376	112.196170	TeamA	Provider1	1
3	86	1	94.406461	108.493608	72.025374	TeamC	Provider2	0
4	34	0	97.752899	99.413492	103.756271	TeamB	Provider1	1

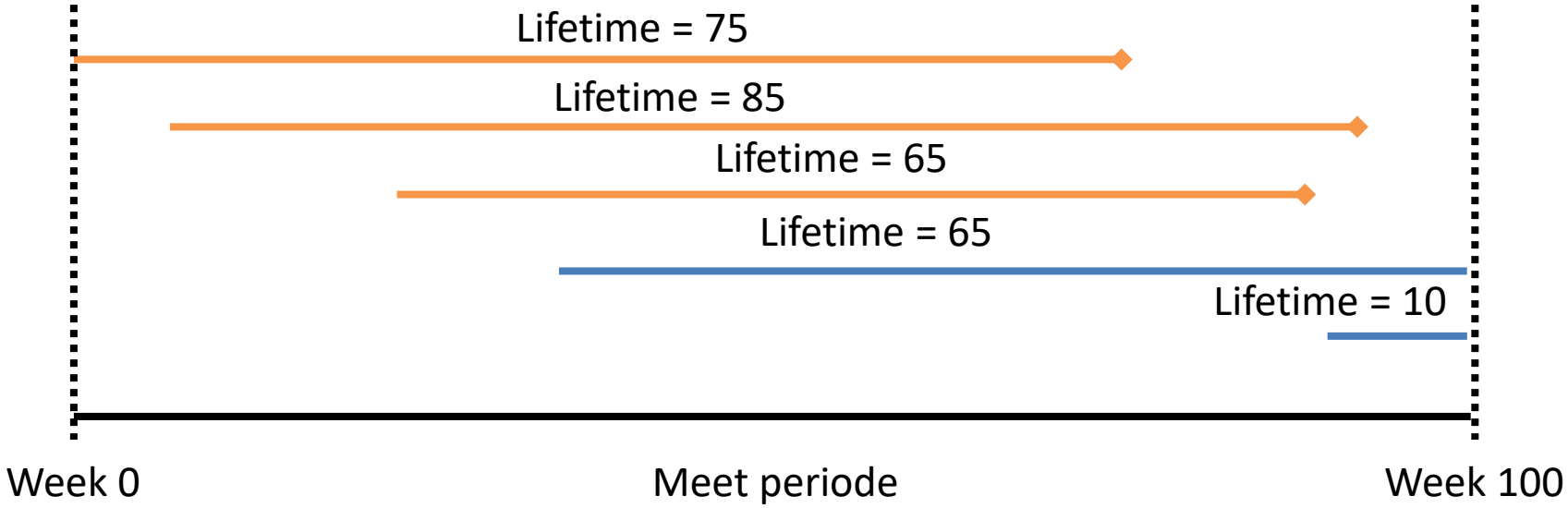
Machine is 34 weken oud en is nog niet kapot

Machine is op week 86 kapot gegaan

Waarom niet regressie?

Waarom kunnen we niet zeggen dat we met regressie de *Remaining Usefull Lifetime (RUL)* gaan voorspellen?

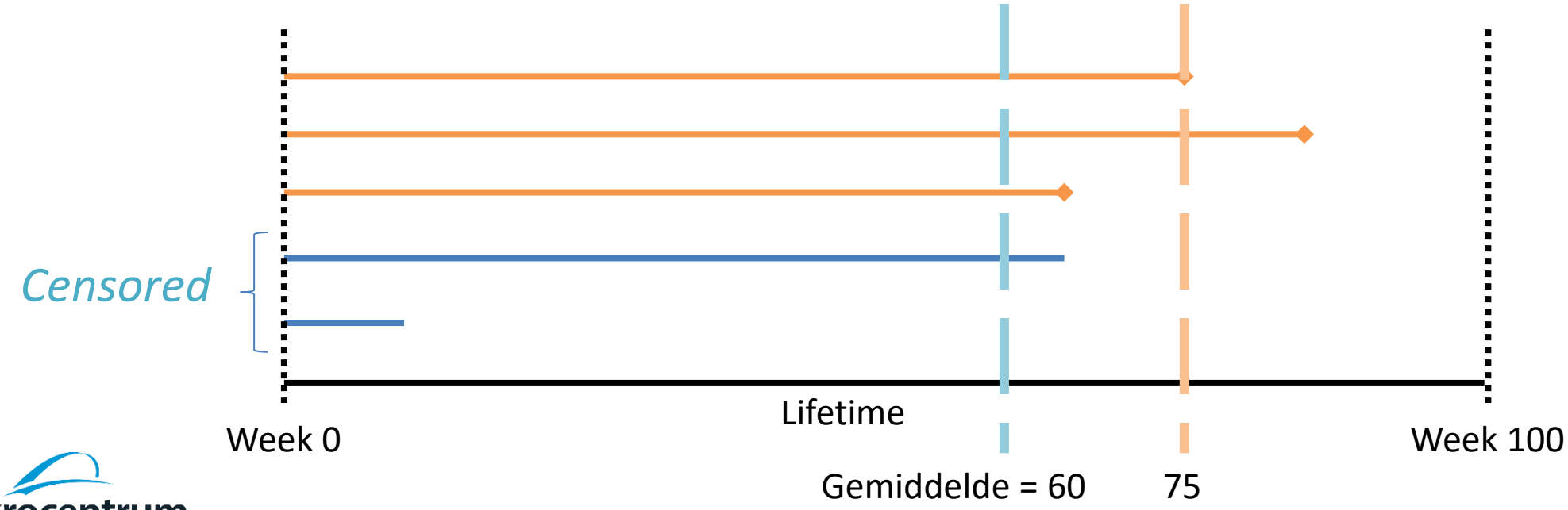
$$\text{lifetime} = \beta_0 + \beta_1 \cdot \text{pressureInd} + \beta_2 \cdot \text{moistureInd} + \beta_3 \cdot \text{temperatureInd}$$



Waarom niet regressie?

Waarom kunnen we niet zeggen dat we met regressie de *Remaining Usefull Lifetime (RUL)* gaan voorspellen?

$$\text{lifetime} = \beta_0 + \beta_1 \cdot \text{pressureInd} + \beta_2 \cdot \text{moistureInd} + \beta_3 \cdot \text{temperatureInd}$$



Kaplan Meier Estimator

- 1000 datapunten (*n*)
- Data:
  Broken: *Hoeveel zijn er kapot gegaan op dag X?*
  Censored: *Van de machines die niet kapot zijn gegaan, hoe oud zijn die?*

- Aantal Samples = n – Broken – Censored
- HF = Broken/Aantal Samples
- $cHF_{new} = cHF_{old} + HF$
- $Survival_{new} = (1\text{-}HF) \times Survival_{old}$

Gegeven | Berekend

Lifetime	Broken	Censored	Aantal Samples	HF (Hazard function)	cHF (cumulative hazard function)	Survival
1	0	5	1000	0	0	1
2	0	11	995	0	0	1

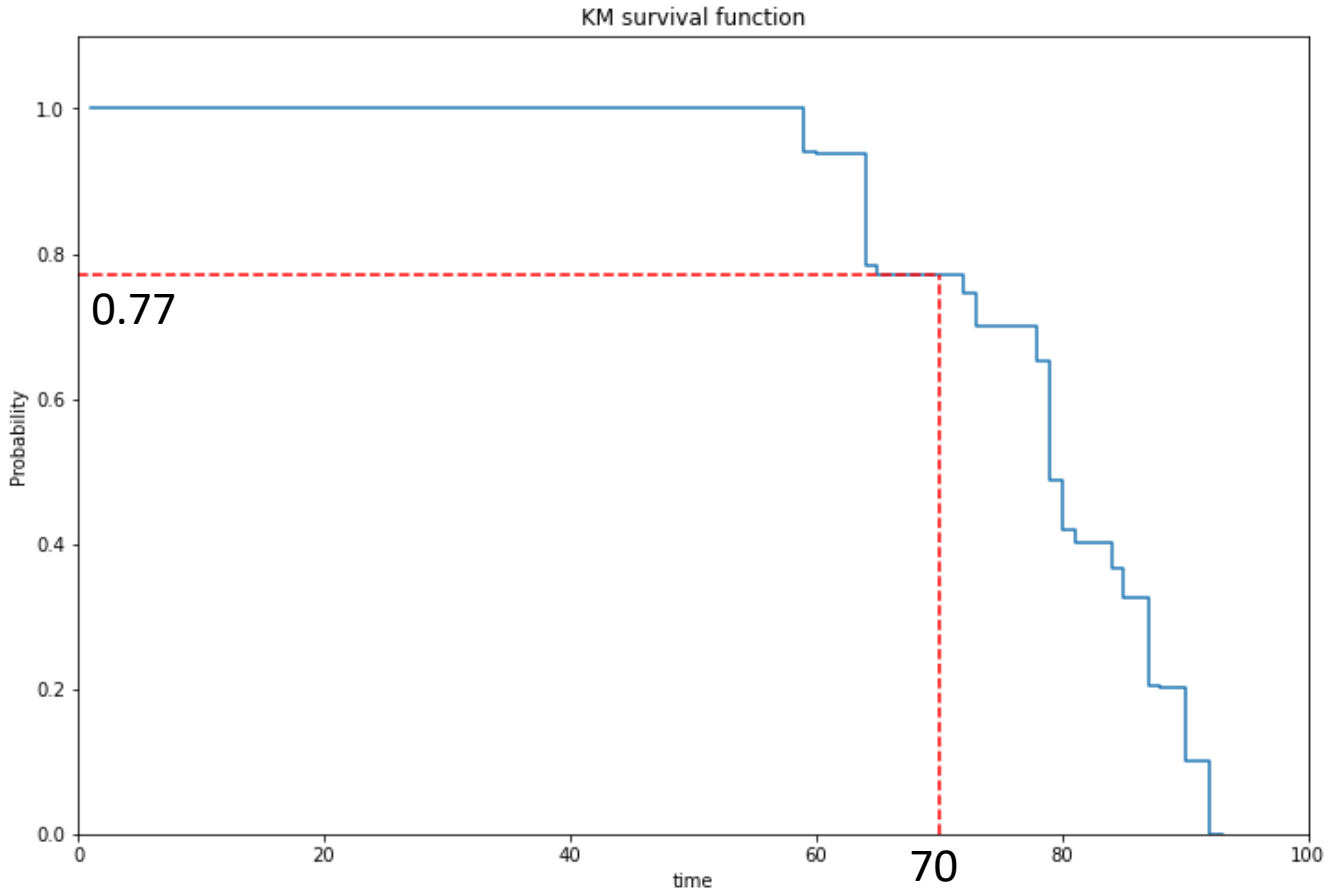
Kaplan Meier Estimator

- Aantal Samples = n – Broken – Censored
- HF = Broken/Aantal Samples
- $cHF_{new} = cHF_{old} + HF$
- $Survival_{new} = (1\text{-}HF) \times Survival_{old}$

Lifetime	Broken	Censored	Aantal Samples	HF (Hazard function)	cHF (cumulative hazard function)	Survival
1	0	5	1000	0	0	1
2	0	11	995			1
⋮	⋮	⋮	⋮	⋮		⋮
59	0	5	536	0.00	0	1
60	32	7	531	0.060	0.060	0.940
61	1	10	492	0.002	0.062	0.938
62	0	9	481	0.000	0.062	0.938
63	0	11	472	0.000	0.062	0.938
64	0	3	461	0.000	0.062	0.938
65	75	7	458	0.160	0.222	0.778

Kans dat een machine kapot gaat op week 60

Kans dat een machine week 61 overleeft



Lifetime	Survival
1	1
2	1
59	1
60	0.940
61	0.938
62	0.938
63	0.938
64	0.938
65	0.778

Oefening

- Programmeer zelf de Kaplan-Meier tabel
  - (stappenplan staat op de volgende slide)
  - Gebruik de notebook survival_opdracht.ipynb
- Plot de Kaplan Meier survival kans

~15 minuten

- Stappen:
  - Laad de data in en voeg de kolom “censored” toe aan de dataset
  - Verwijder de onnodige kolommen (pressureInd, moistureInd, etc.)
  - Sorteer de dataset op “lifetime”
  - Groepeer de dataset zo dat alle machines met dezelfde “lifetime” samengevoegd worden.
  - Maak de kolommen aantal samples, hazard function, cumulative hazard function en survival probability aan
  - Vul deze kolommen in voor “lifetime” = 1
  - Loop over alle lifetimes en vul de kolommen in met de correcte formules zoals gegeven op slide 36
  - Print het resultaat. Deze zou er ongeveer uit moeten komen te zien als die op slide 37

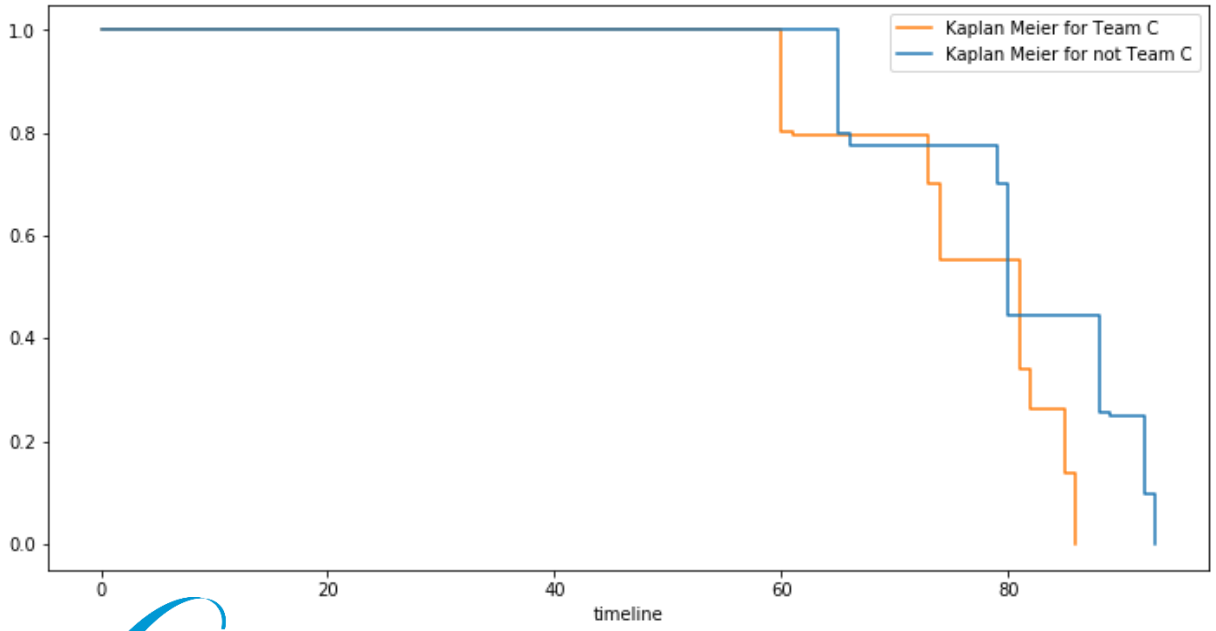
Oefening

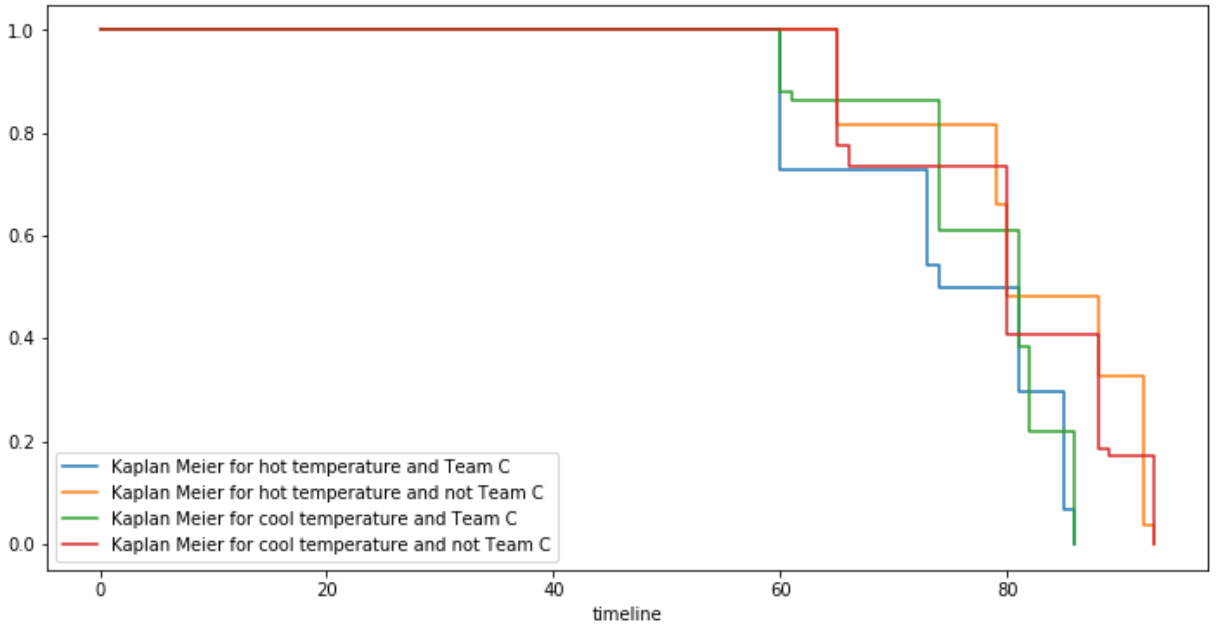
- Gebruik de package `lifelines` om een kaplan meier estimator te maken met `KaplanMeierFitter`
- Controller je eigen antwoorden met die van `lifelines`

~5 min

Meer variabelen?

	lifetime	broken	pressureInd	moistureInd	temperatureInd	team	provider	censored
0	56	0	92.178854	104.230204	96.517159	TeamA	Provider4	1
1	81	1	72.075938	103.065701	87.271062	TeamC	Provider4	0
2	60	0	96.272254	77.801376	112.196170	TeamA	Provider1	1
3	86	1	94.406461	108.493608	72.025374	TeamC	Provider2	0
4	34	0	97.752899	99.413492	103.756271	TeamB	Provider1	1






```
Hot = temperatureInd >= 1
Cool = temperatureInd < 1
```

Cox proportional hazard model (Cox Regression)

Kaplan Meier

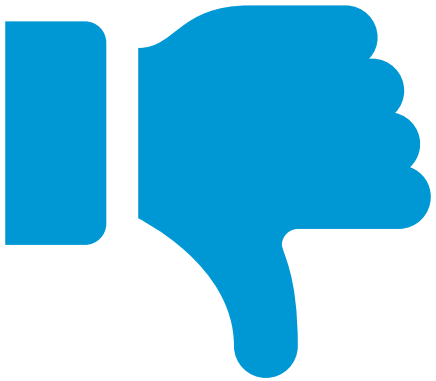
$$\mathrm{HF} = \frac{\text{Broken}}{\text{Aantal Samples}}$$

Cox Proportional Hazard Model

$$\mathrm{HF} = f(X) = \beta_2 x_2 + \beta_1 x_1 + \beta_0$$

Categorische variabelen

Cijfers



Letters

Categorische data

```
pd.get_dummies(df)
```

	temperatureInd	team
0	96.517159	TeamA
1	87.271062	TeamC
2	112.196170	TeamA
3	72.025374	TeamC
4	103.756271	TeamB
5	89.792105	TeamA
6	142.827001	TeamB
7	98.316190	TeamB
8	96.028822	TeamB
9	95.492965	TeamC

	temperatureInd	TeamA	TeamB	TeamC
0	96.517159	1	0	0
1	87.271062	0	0	1
2	112.196170	1	0	0
3	72.025374	0	0	1
4	103.756271	0	1	0
5	89.792105	1	0	0
6	142.827001	0	1	0
7	98.316190	0	1	0
8	96.028822	0	1	0
9	95.492965	0	0	1

Oefening

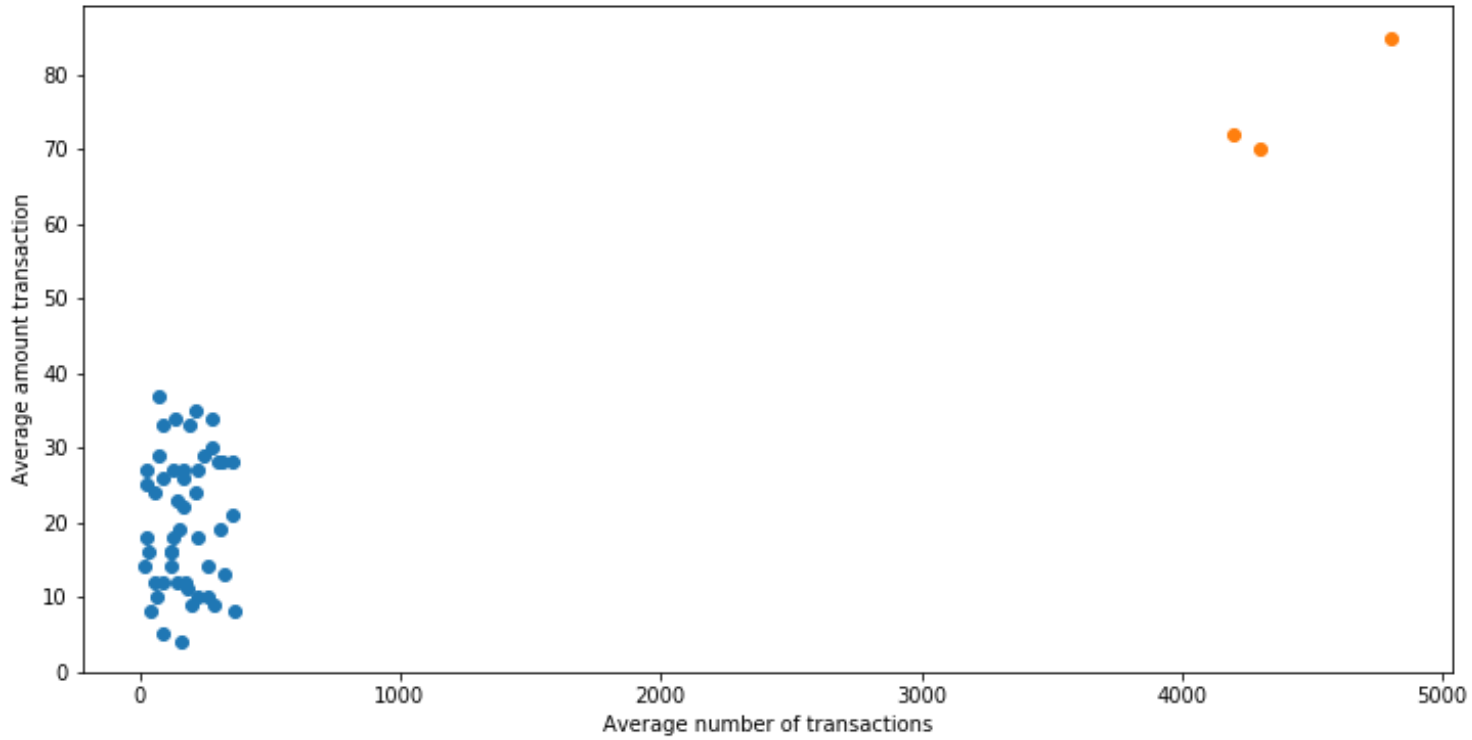
- Maak een cox regression model
- Na het trainen: gebruik `.print_summary()` functie
- Gebruik `.predict_expectation()` als voorspelling en evalueer het model
- Maak een visualisatie

~15 minuten

ANOMALY DETECTION

Anomaly detection/outlier detection

- Kan erg handig zijn (unsupervised)
- Heeft hele specifieke toepassingen (fraude detecte, maar ook predictive maintenance in sommige gevallen)
- Gaan we niet behandelen, maar hier wat leesmateriaal: Novelty and Outlier Detection



Anomaly Detection

Outlier detection

Data met uitschieters zijn gegeven. Kan je de uitschieters van de normale data onderscheiden?

Novelty Detection

Alleen de normale data is bekend. Kan je detecteren wanneer er iets afwijkt?

DATA PRE-PROCESSING

Wat gaan we behandelen?

- Features maken
  - Gemiddelde
  - Standaarddeviatie
  - Kurtosis/Skewness
  - Energy
  - Periodicity (Autoregression/Fourier)
  - Wavelet
  - Dummies
- Feature scaling
- Uitschieters/NaN
- Class imbalance
- Smoothing (moving average)
- Collineariteit

Features maken

- Werken met sensoren → time data
- Feature engineering → 80% van je werk
- Een feature zegt iets zinvols over het gene wat je wil voorspellen

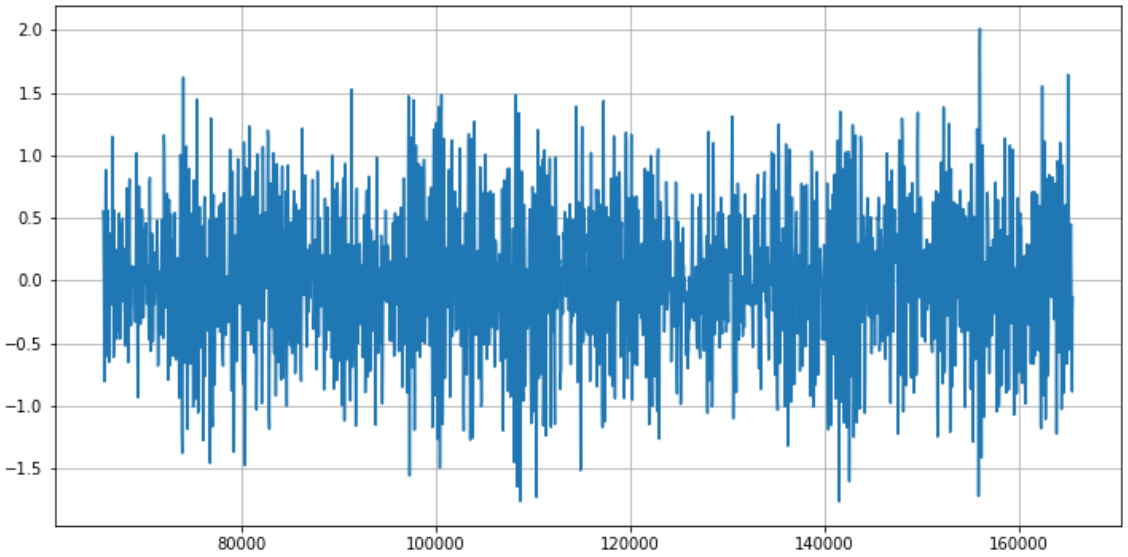
	lifetime	broken	pressureInd	moistureInd	temperatureInd	team	provider	censored
0	56	0	92.178854	104.230204	96.517159	TeamA	Provider4	1
1	81	1	72.075938	103.065701	87.271062	TeamC	Provider4	0
2	60	0	96.272254	77.801376	112.196170	TeamA	Provider1	1
3	86	1	94.406461	108.493608	72.025374	TeamC	Provider2	0
4	34	0	97.752899	99.413492	103.756271	TeamB	Provider1	1

Engineered features

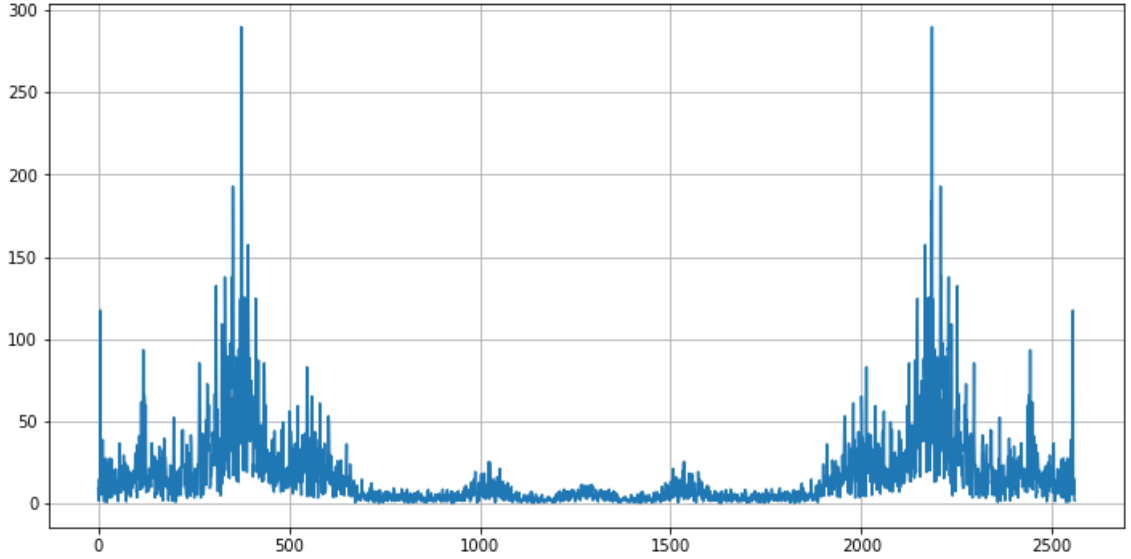
(deze zijn overigens ook genormaliseerd, dit komt later)

Timeseries data

Tijd domein



Frequentie domein



Tijdsdomein

- Gemiddelde

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

```
x.mean()
```

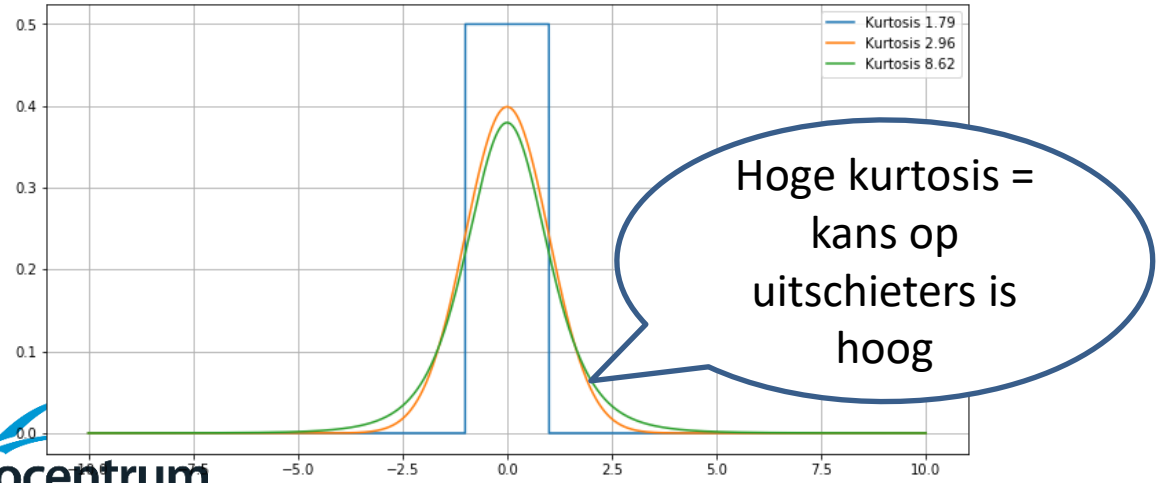
- Standaarddeviatie

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

```
x.std()
```

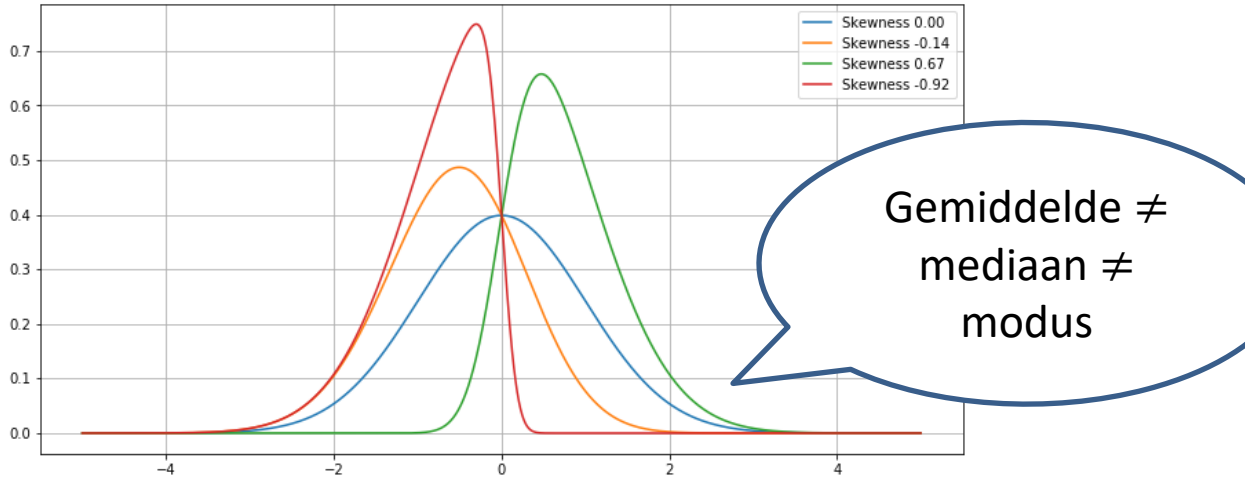
- Kurtosis (tailedness)

```
scipy.stats.kurtosis(x, fisher=False)
```



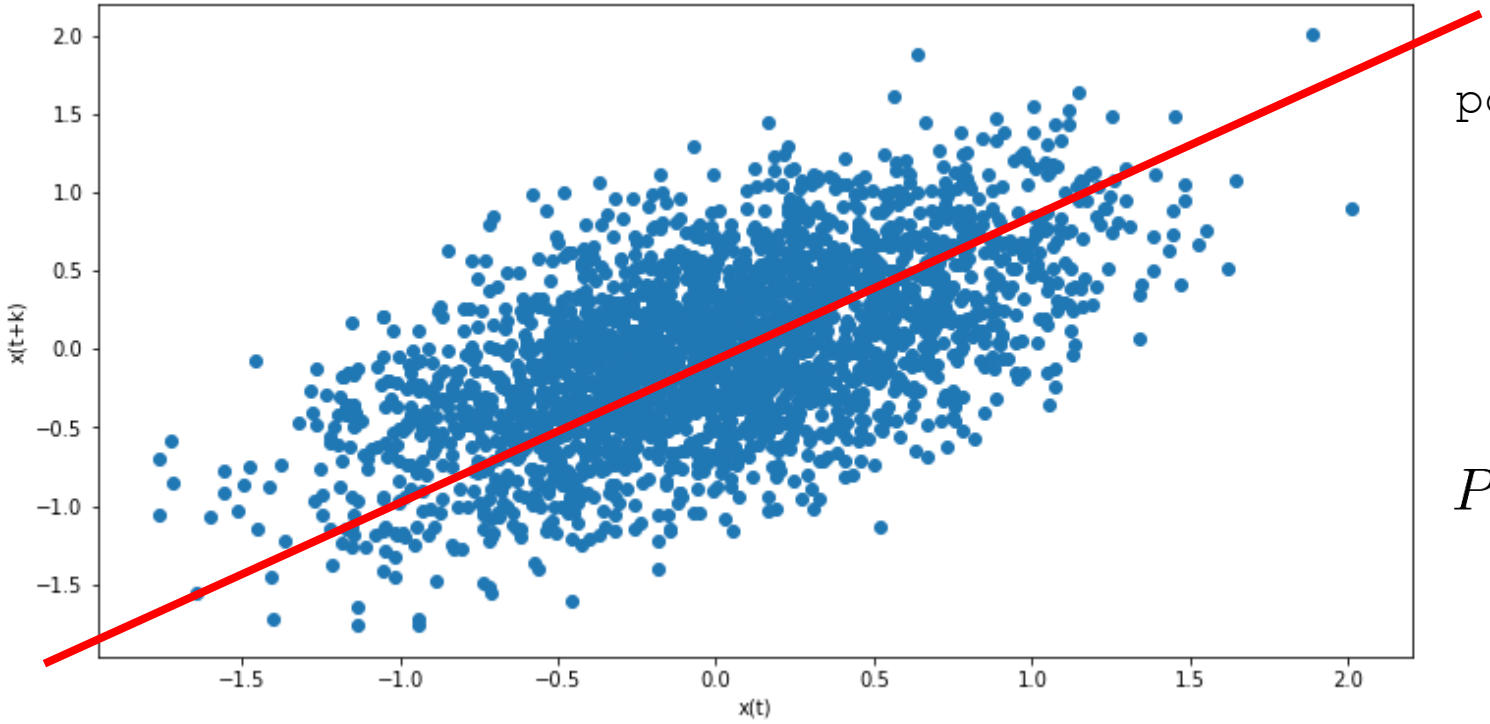
- Skewness

```
scipy.stats.skew(x)
```



Autoregressie

- Grieks **Auto-** = zelf- --> zelf-regressie
- Verband zoeken tussen variabele en de variabele vertraagd




```
pd.plotting.lag_plot(x, lag = 1)
```

$$P_k = \frac{1}{n}\sum_{i=1}^{n}\frac{(x_i - \mu_{x_i})(x_{i+k} - \mu_{x_{i+k}})}{\sigma_{x_i}\sigma_{x_{i+k}}}$$

```
pk = lambda x, k: sum(
                    (x - x.mean())*(np.roll(x, k)-np.roll(x, k).mean())/
                    (x.std()*np.roll(x, k).std())
                    )/len(x)
```

```
pk = lambda x, k: sum(
                      (x - x.mean())*(np.roll(x, k)-np.roll(x, k).mean())/
                      (x.std()*np.roll(x, k).std())
                      )/len(x)
```

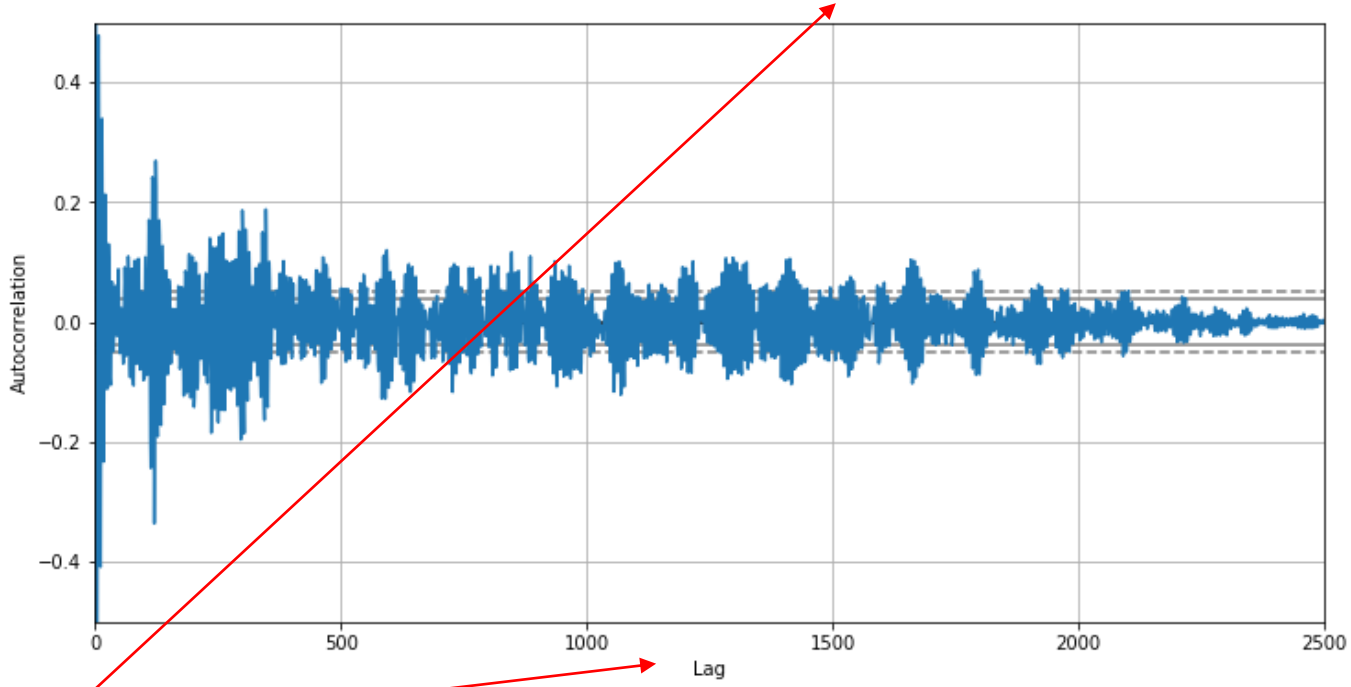
=

```
def pk(x, k):
    return sum(
               (x - x.mean()) * (np.roll(x, k) - np.roll(x, k).mean())/
               (x.std() * np.roll(x, k).std())
               )/len(x)
```


0
/ AI voor Engineers / dag 2
..
Dagcasus
regression.ipynb
Survival analysis.ipynb
survival_opdracht.ipynb
autocorrelation.py
signal_data.csv
survival_data.csv

Autoregressie

```
pd.plotting.autocorrelation_plot(x)
```




```
pk = lambda x, k: sum(
                    (x - x.mean())*(np.roll(x, k)-np.roll(x, k).mean())/
                    (x.std()*np.roll(x, k).std())
                    )/len(x)
```

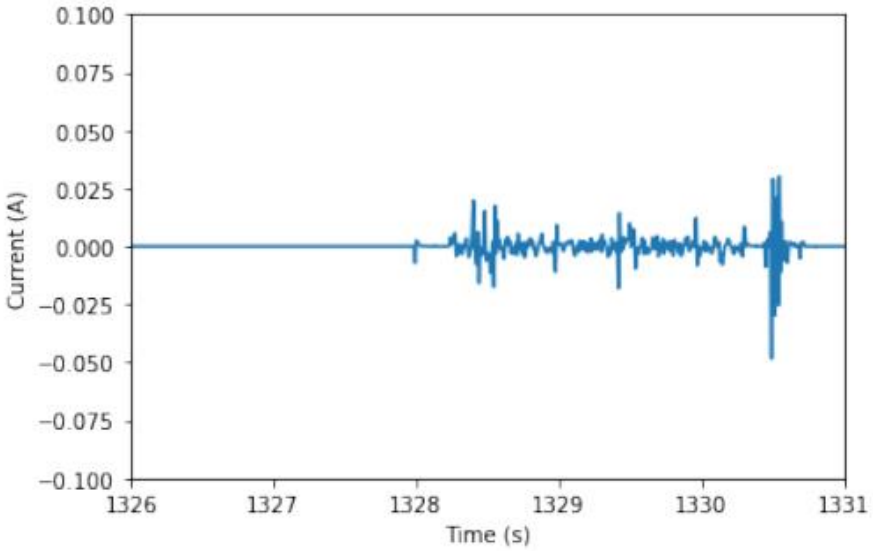
Oefening (1/3)

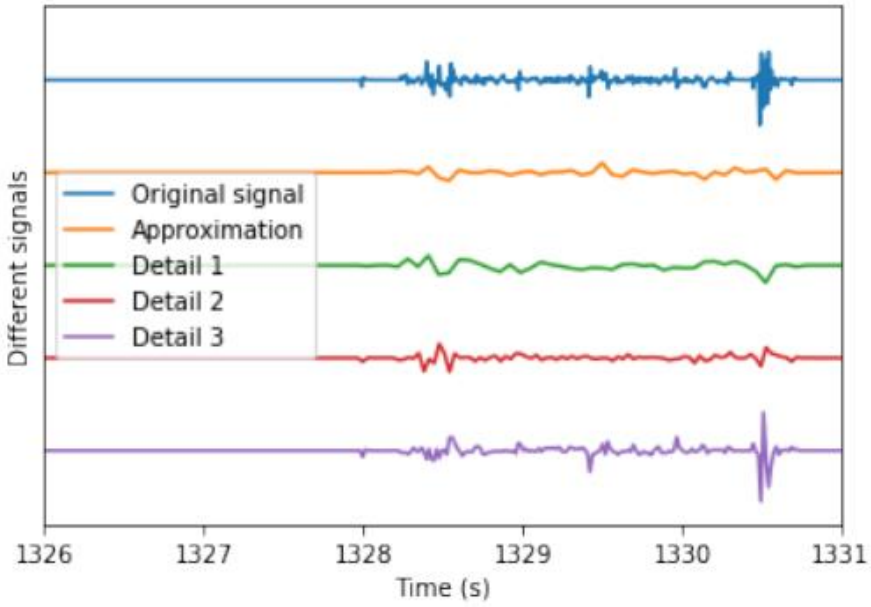
- Bereken de volgende features van `signal_data.csv`. Specifiek van ‘V2’:
  - Gemiddelde
  - Standaard deviatie
  - Kurtosis
  - Skewness
  - Autoregressie met een vertraging van 100

  ~ 20 minuten

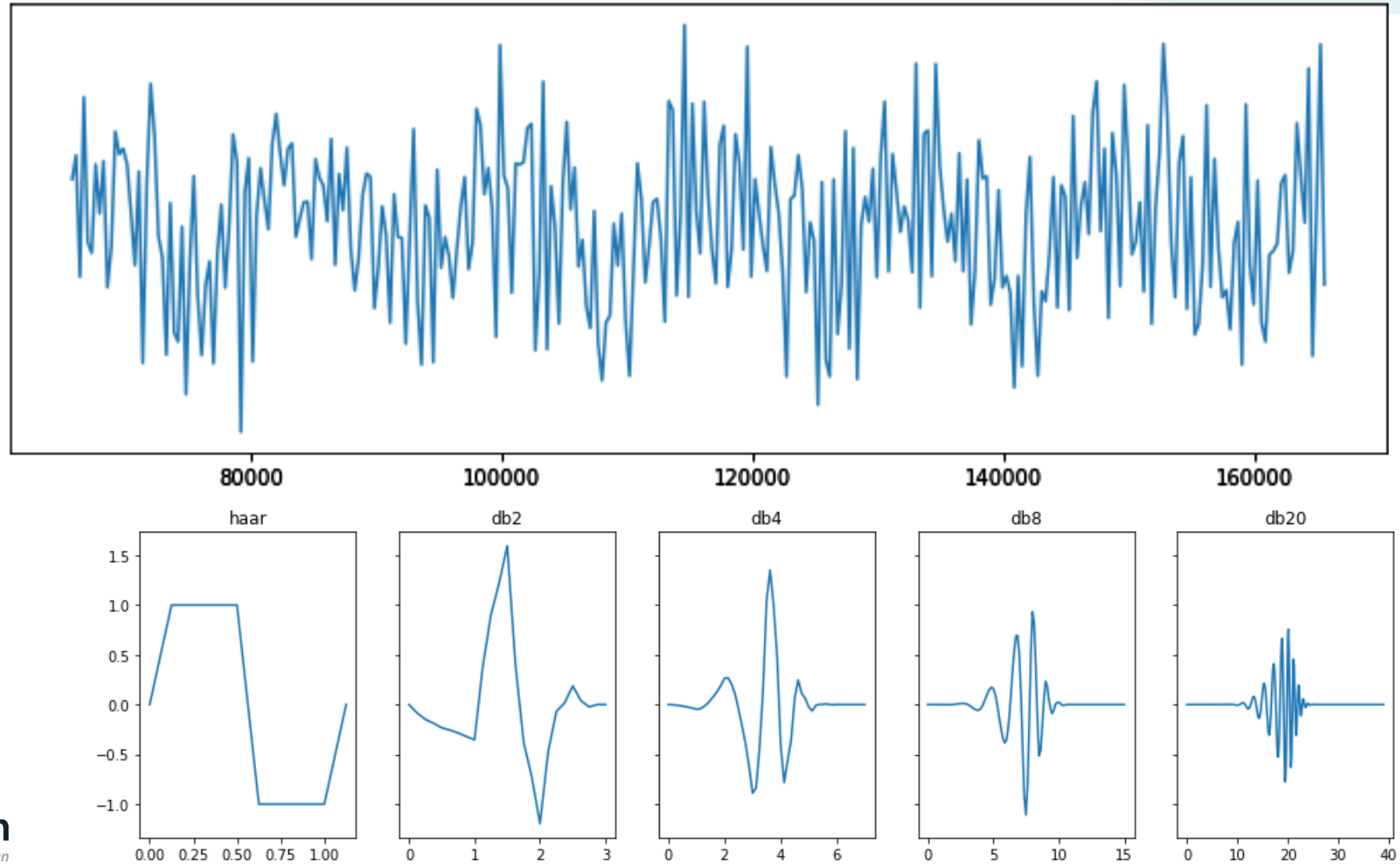
Wavelet transform

- Wavelet transformaties decompenseert het originele signaal
- Acceleratie data
- Detecteert veranderingen over verschillende frequenties

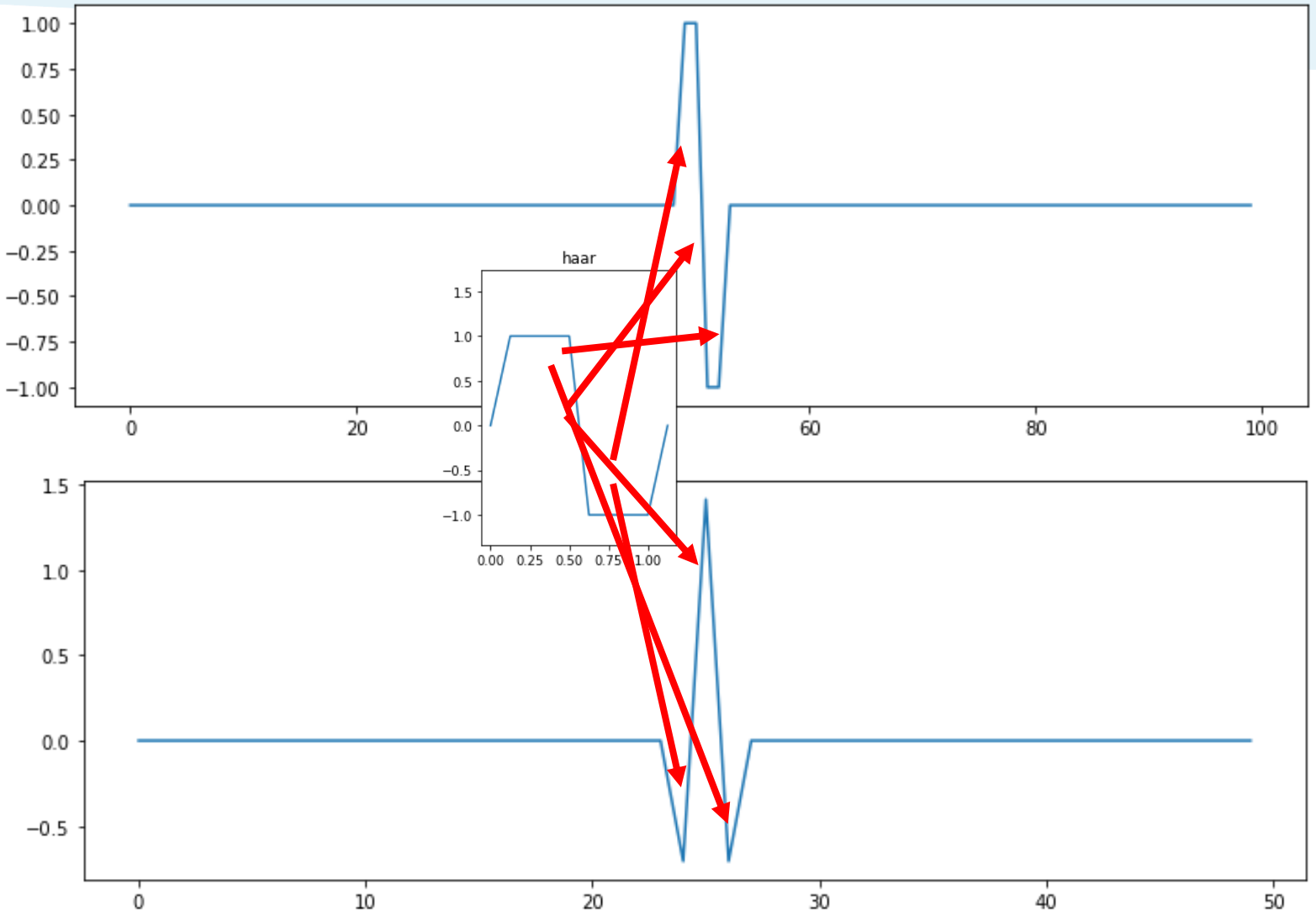




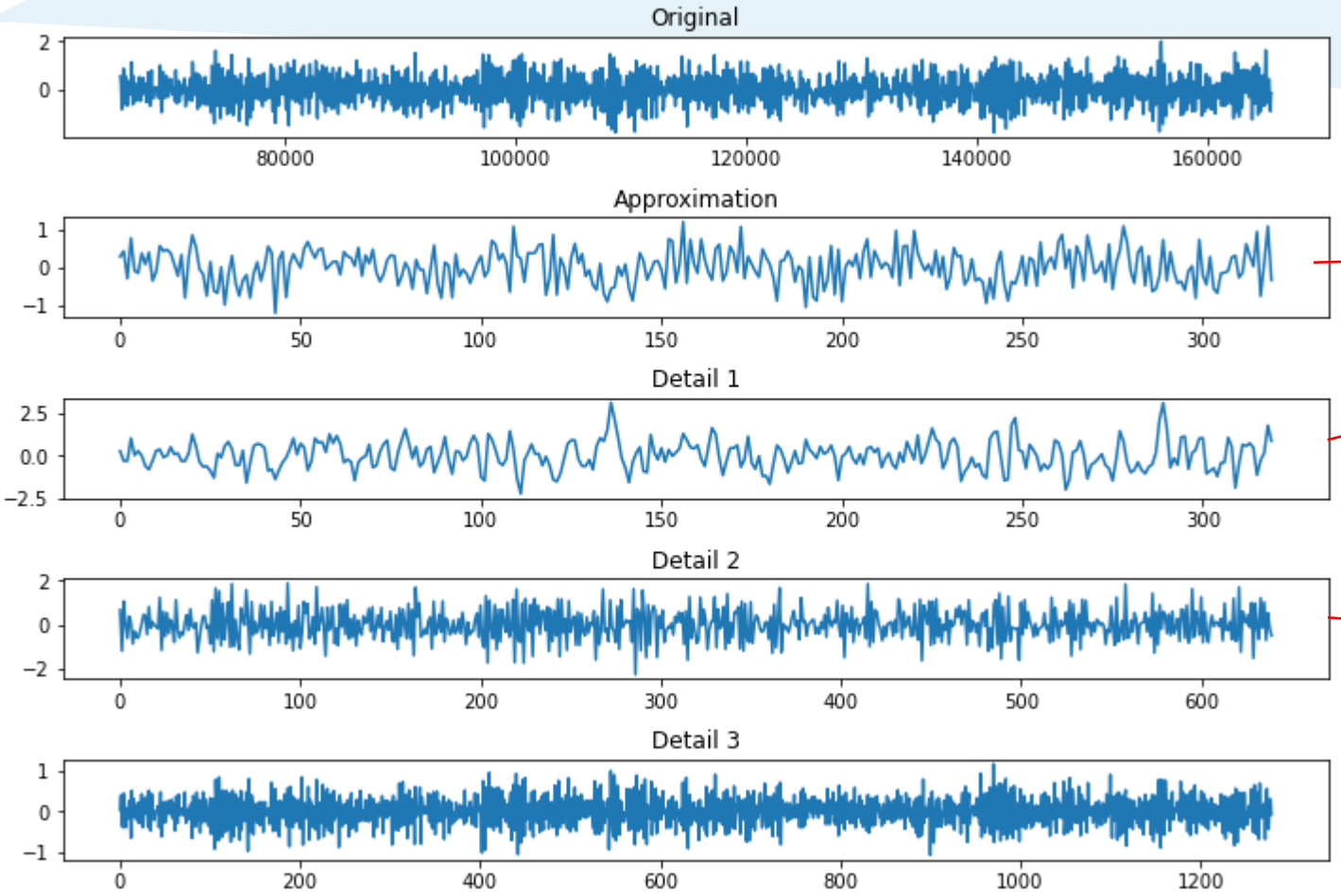
Wavelet transfer

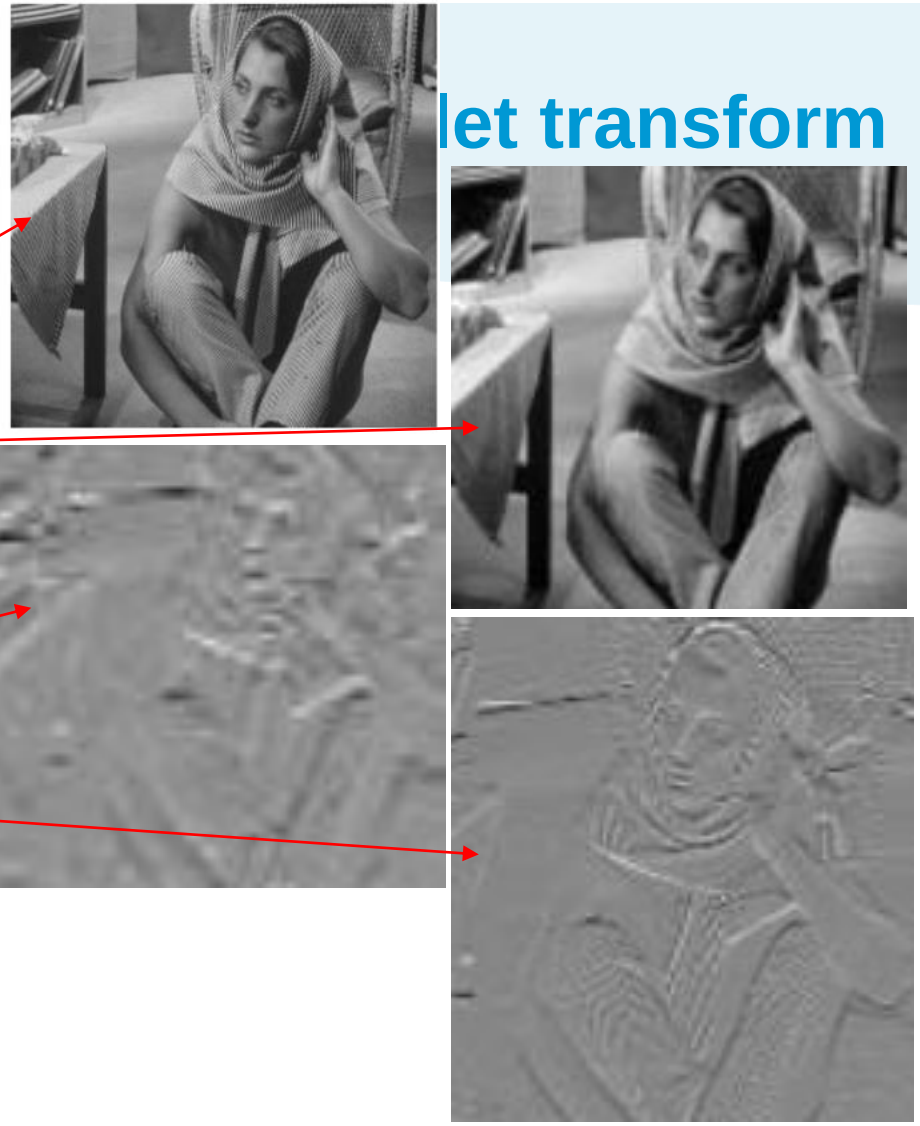


Signaal met wavelet erin



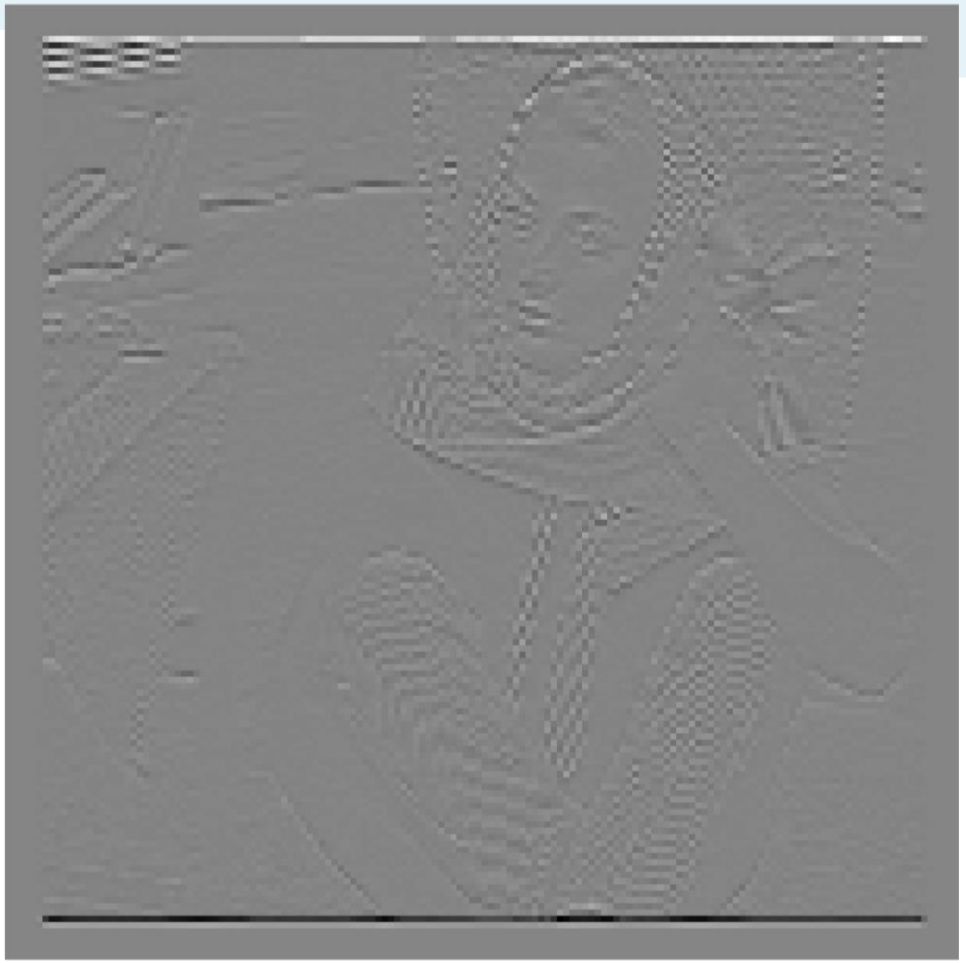
let transform





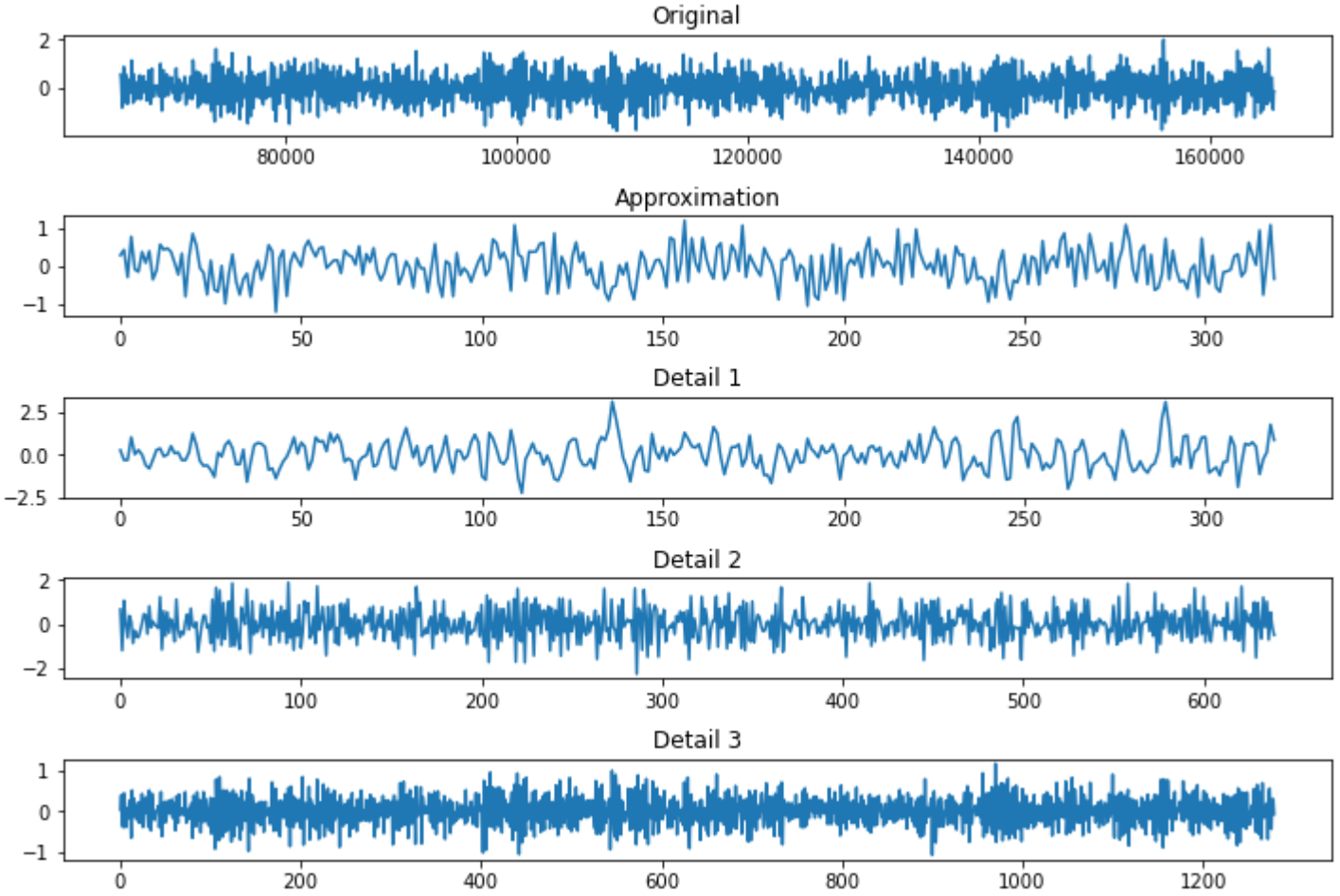
```
cA, cD1, cD2, cD3 = pywt.wavedec(signal, wavelet, level = 3)
```

Approximation	Detail 1	Detail 2	
			Haar
			Db1
			Db2



$$\text{Energy} = \sum_{i=1}^{n} |W_i|^2$$

Wavelet transform (“haar”)



Energy = 807.8

Energy = 68.8

Energy = 228.7

Energy = 340.89

Energy = 169.43

\+ 807.8

Oefening (2/3)

- Bereken de energy van de approximation en details wavelet met een niveau 4 decompositie en de “haar” wavelet
- Visualiseer de wavelet transformaties
- Gebruik de features die we hiervoor behandeld hebben op de getransformeerde wavelets

15 minuten

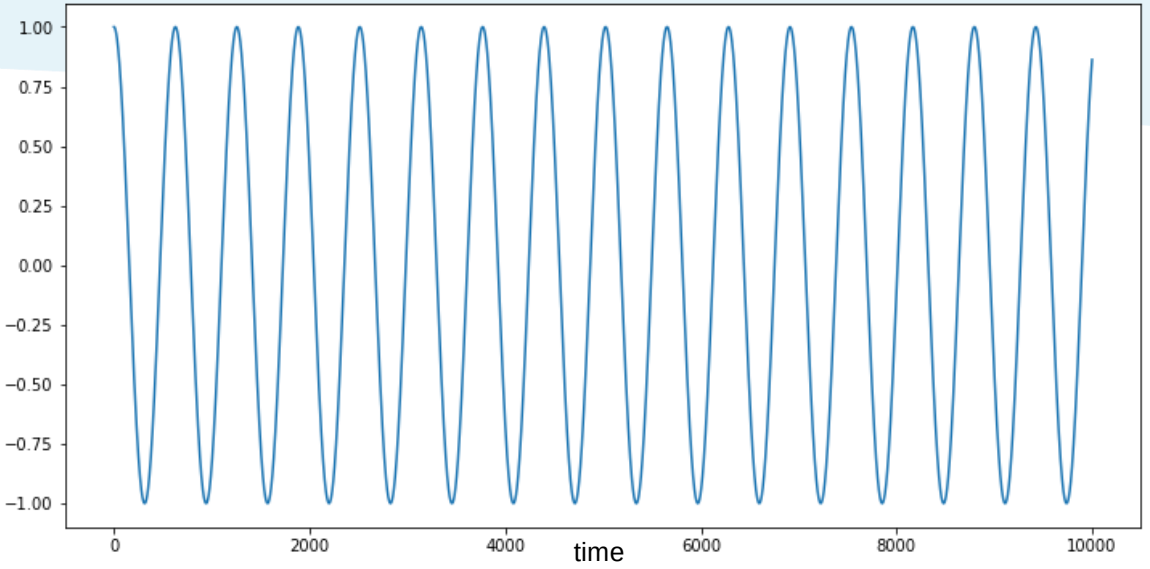
```
import pywt
pywt.wavedec
```

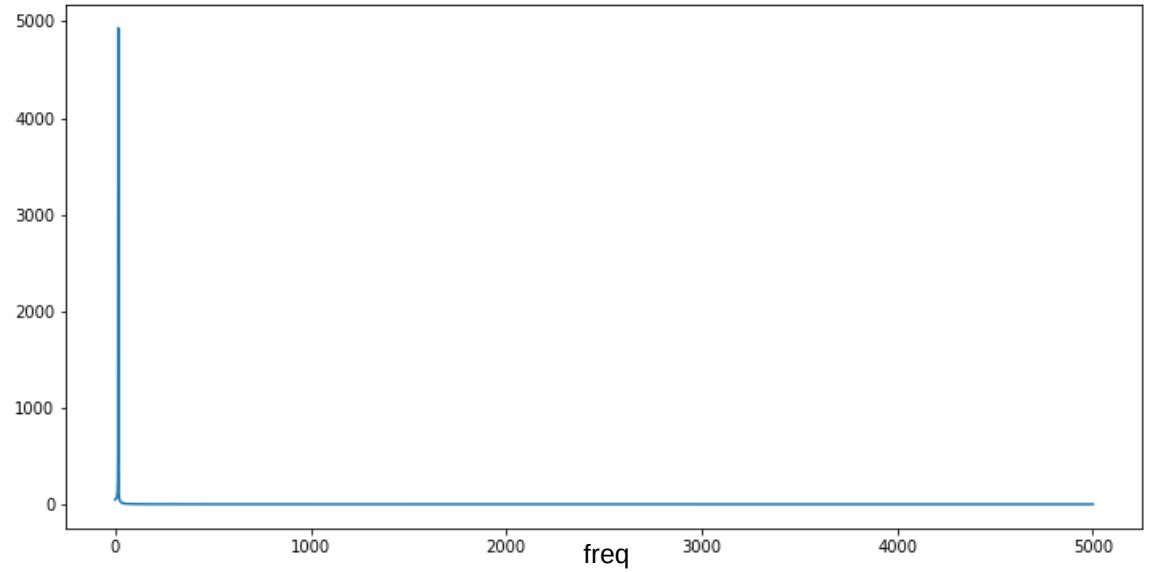
Frequentie domein

Fast Fourier Transform

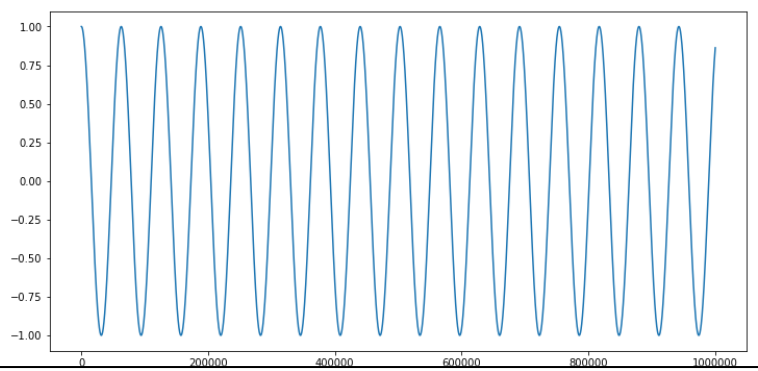
```
F = np.fft.rfft(f)
```

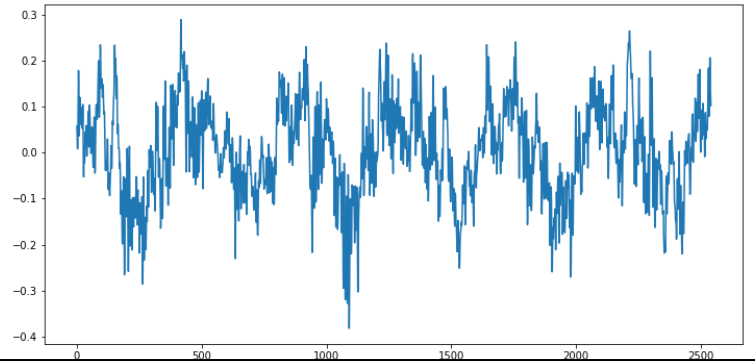
$$\mathcal{F}(\omega) = \mathrm{FFT}(f(x))$$

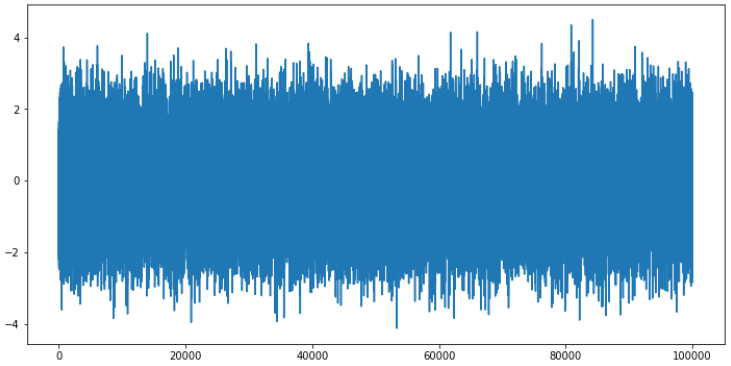


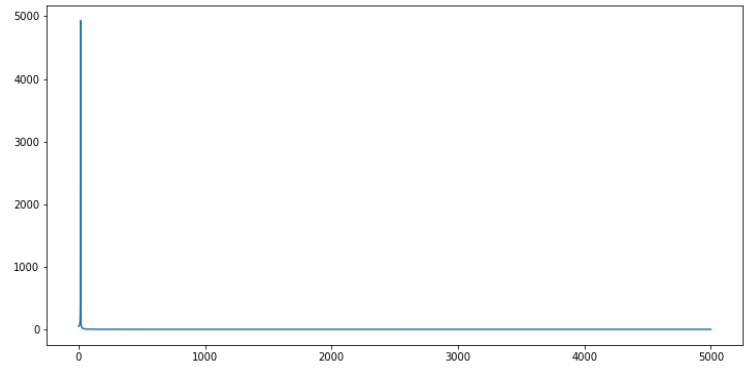


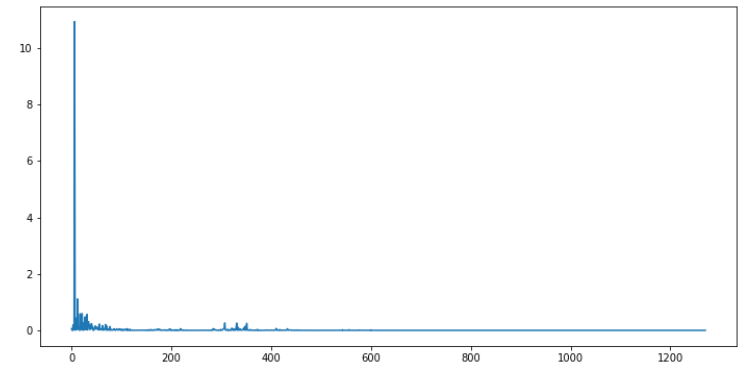
Spectral Flatness

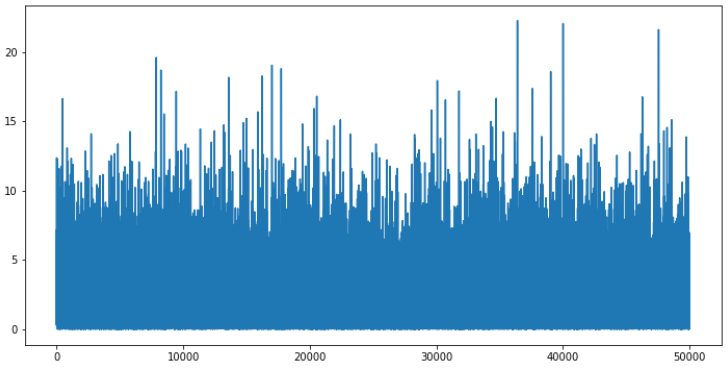












Power spectrum

Spectral Flatness = 0	Spectral Flatness = 0.02	Spectral Flatness = 0.58

mikrocentrum
opleiden • ontmoeten • ondernemen

Spectral Flatness

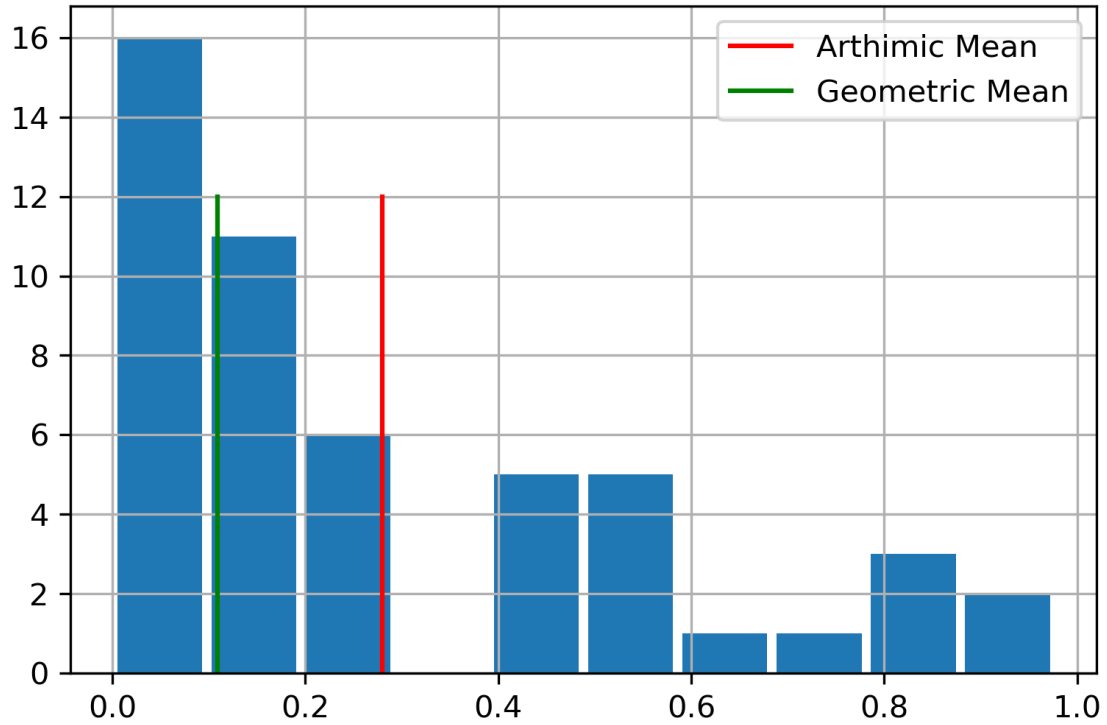
Arithmic mean $$\mu_A = \frac{1}{n}\sum_{i=1}^{n} S(\omega)$$

Geometric mean $$\mu_G = \exp\left(\frac{1}{n}\sum_{i=1}^{n} \ln(S(\omega))\right)$$

$$S(\omega) = |\mathcal{F}(\omega)|^2$$

$$F = \frac{\mu_G}{\mu_A}$$

$$\exp(x) = e^x$$



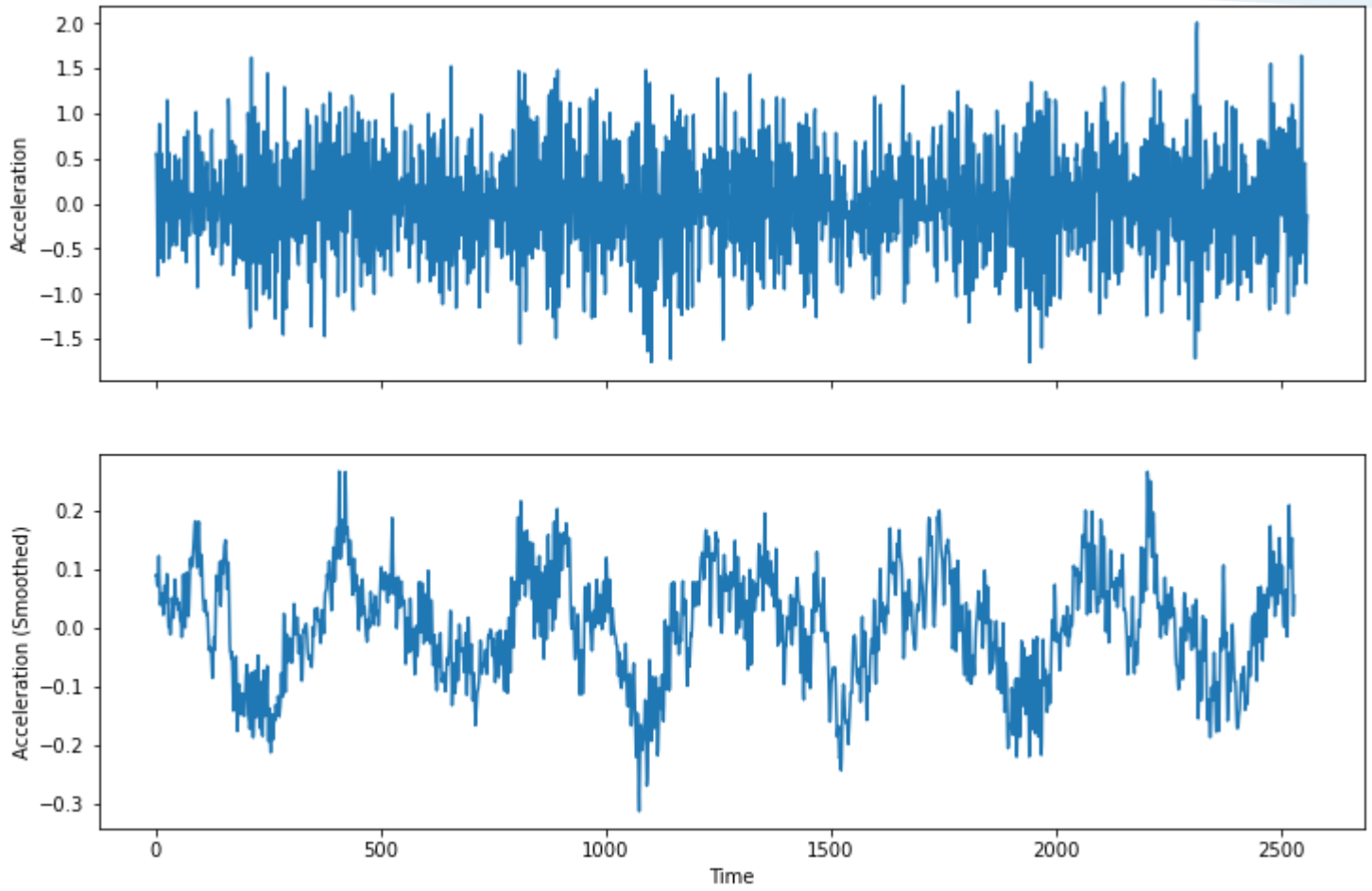
mikrocentrum
opleiden • ontmoeten • ondernemen

Oefening (3/3)

- Bereken de spectral flatness van het gegeven signaal
- Bereken voorgaande features over het frequentie domein (gemiddelde, standaard deviatie, skewness en kurtosis)

- Huiswerk:
  - Vind 3 nieuwe features in literatuur die gebruikt kan worden op signaaldata

Smoothing



$$x'_k = \frac{1}{M} \sum_{i=\,\mathrm{k}}^{M} x_{i-k}$$

x = signaal

x’ = smoothed signaal

M = window width

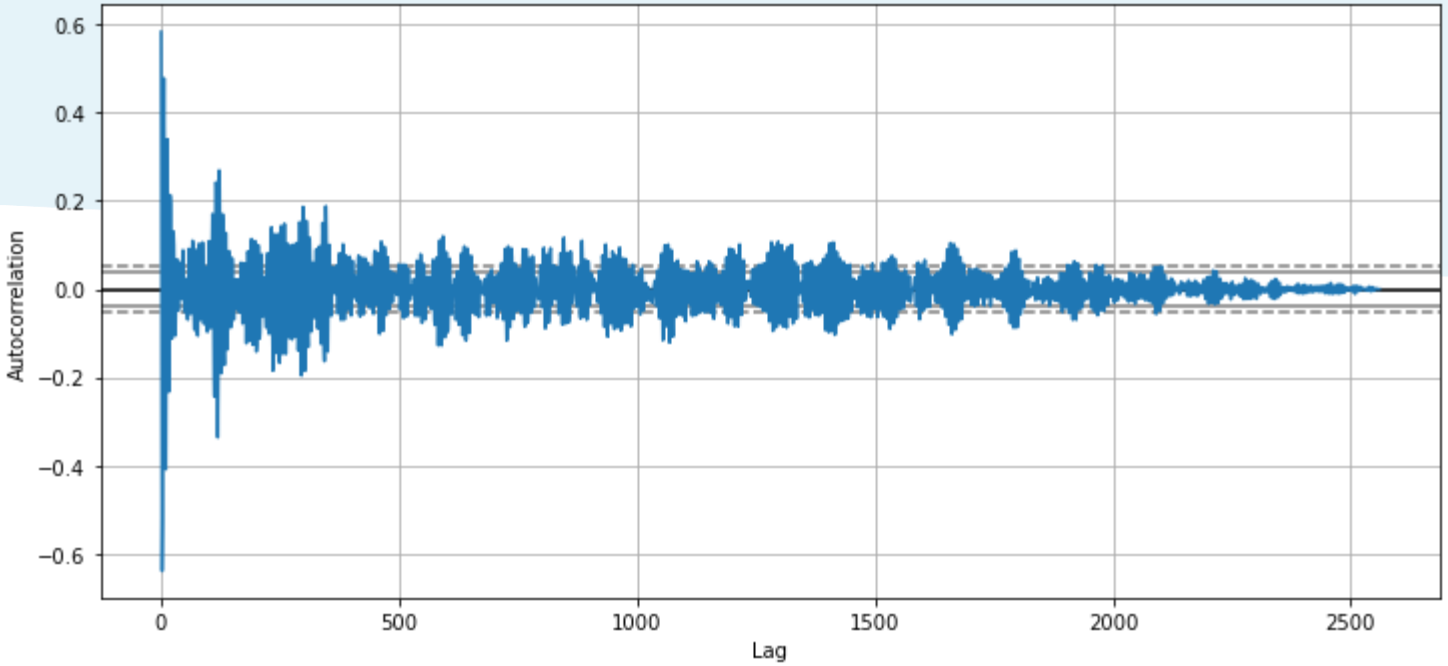
K = window position

Smoothing

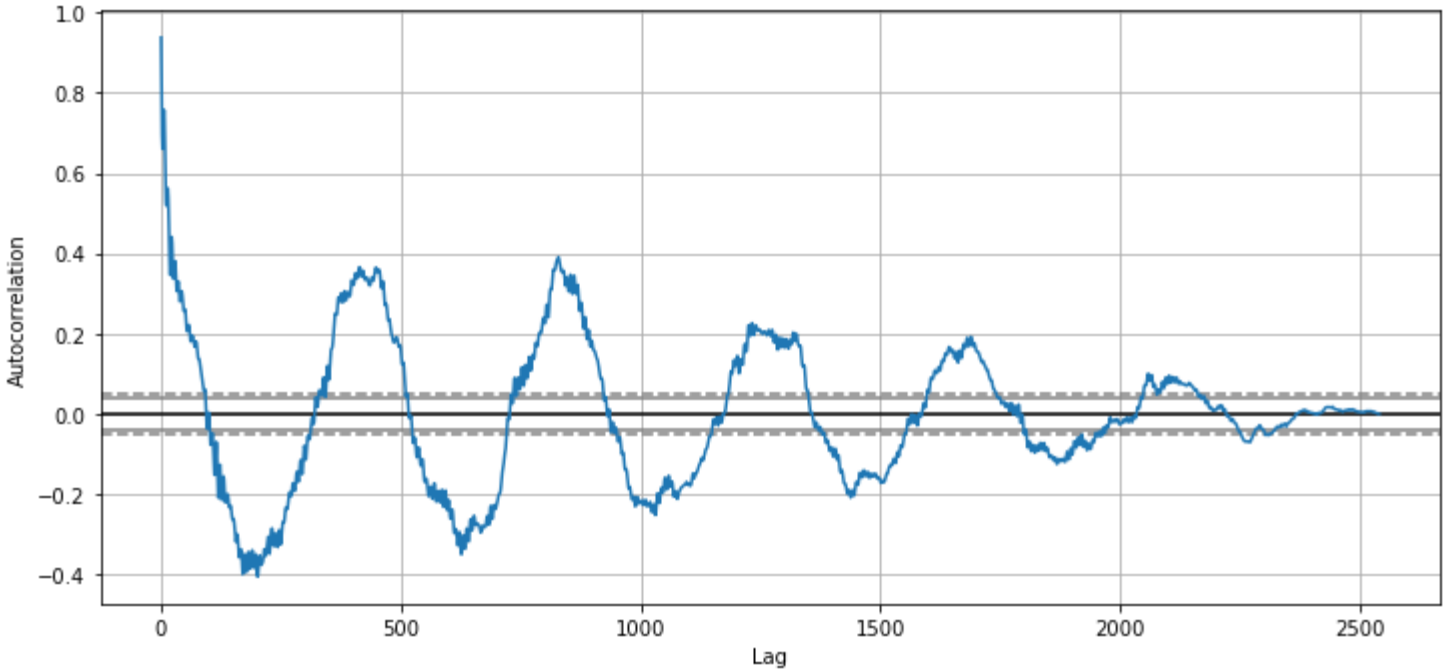
- Niet altijd handig voor wavelet transformaties
- Handig voor autoregression

Waarom is dit niet altijd handig voor wavelet transformaties, maar wel voor autoregression?

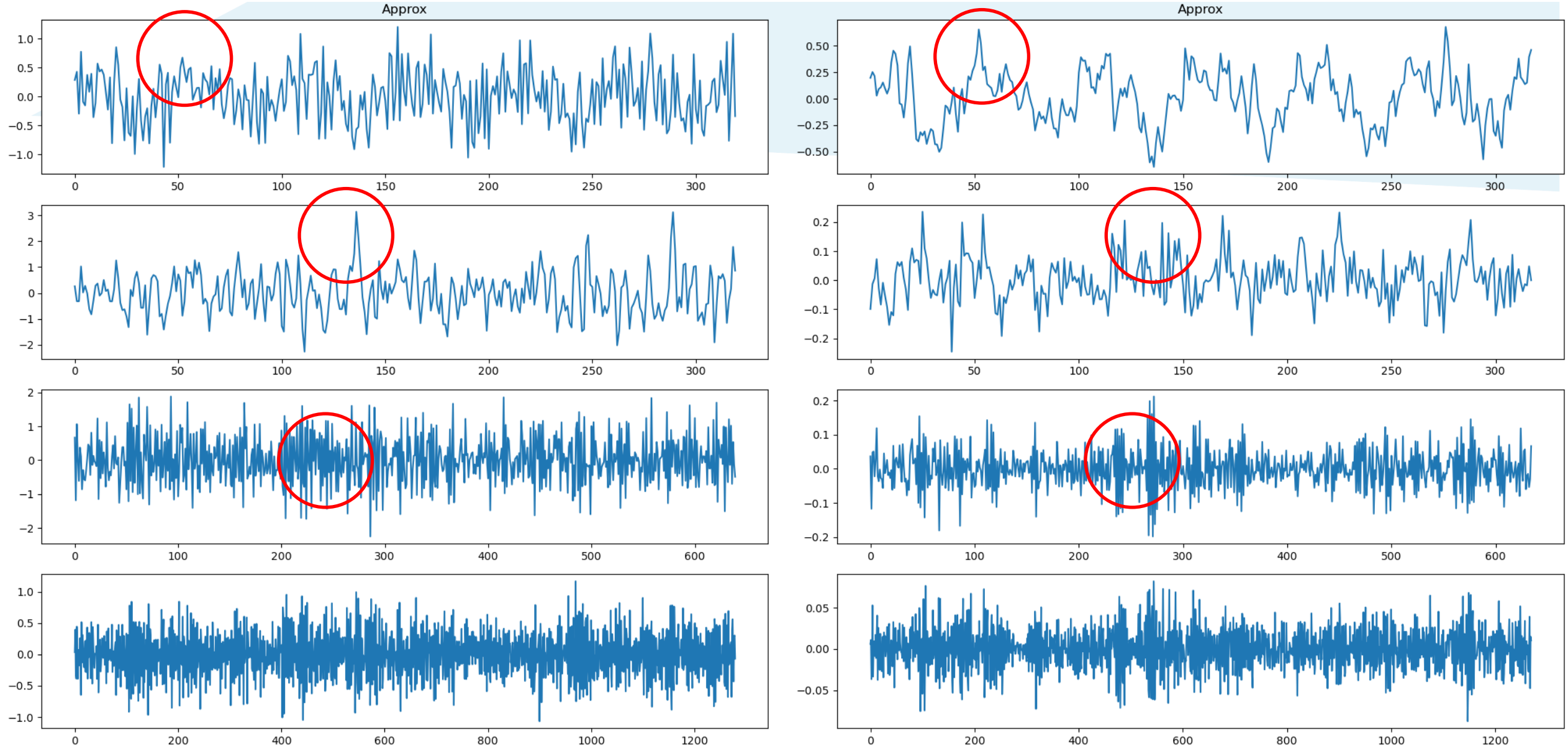
Geen smoothing



Smoothing



mikrocentrum
opleiden • ontmoeten • ondernemen



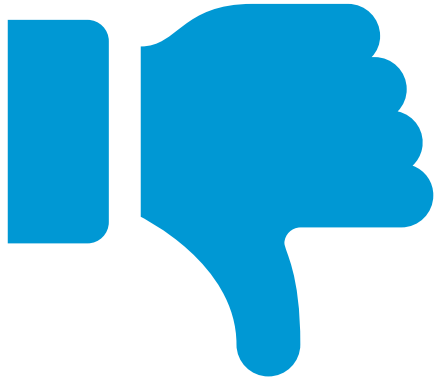
Geen smoothing

Smoothing

mikrocentrum
opleiden • ontmoeten • ondernemen

Categorische variabelen

Cijfers



Letters

```
pd.get_dummies(df)
```

	temperatureInd	team
0	96.517159	TeamA
1	87.271062	TeamC
2	112.196170	TeamA
3	72.025374	TeamC
4	103.756271	TeamB
5	89.792105	TeamA
6	142.827001	TeamB
7	98.316190	TeamB
8	96.028822	TeamB
9	95.492965	TeamC

	temperatureInd	TeamA	TeamB	TeamC
0	96.517159	1	0	0
1	87.271062	0	0	1
2	112.196170	1	0	0
3	72.025374	0	0	1
4	103.756271	0	1	0
5	89.792105	1	0	0
6	142.827001	0	1	0
7	98.316190	0	1	0
8	96.028822	0	1	0
9	95.492965	0	0	1

Curse of dimensionality

Hoe meer features je gaat gebruiken, hoe meer data je nodig hebt

DATA CLEANING

mikrocentrum
opleiden • ontmoeten • ondernemen

Uitschieters/NaN

Uitschieters

- Wat zijn uitschieters?
- Vervangen
  - Door wat? Kan ik het goed schatten?
- Verwijderen
  - Hoeveel datapunten houd ik over?
- Behouden
  - Bevat het informatie?

NaN/Missing values

- Vervangen (gemiddelde? Schatten met regressie?)
  - Doet het meer kwaad dan goed?
- Verwijder het datapunt

Wat zeggen domein experts?

mikrocentrum
opleiden • ontmoeten • ondernemen

Feature scaling

- Wat is feature scaling? Waarom?

$$\text{Energy} = \sum_{i=1}^{n} |W_i|^2$$ ➡ Onbegrenst van 0 tot ∞

$$\text{Spectral Flatness} = \frac{\mu_G}{\mu_A}$$ ➡ Begrenst tussen 0 en 1

$$\hat{y} = \beta_2 x_2 + \beta_1 x_1 + \beta_0$$

$$\beta_2 = 0.000001, \qquad \beta_1 = 0.1$$

Feature scaling

Op welk percentiel zit x?

Normaliseren

- Min max → $x = \frac{x_i - \min(x))}{\max(x) - \min(x)}$

```
normalize = lambda x: (x-x.min())/(x.max()-x.min())
```

- Mean → $x = \frac{x_i - \mu_x}{\max(x) - \min(x)}$

```
normalize = lambda x: (x-x.mean())/(x.max()-x.min())
```

Op hoeveel procent zit x van het gemiddelde?

- Z-score → $\hat{x} = \frac{x_i - \mu_x}{\sigma_x}$

Hoeveel standaarddeviaties zit x van het gemiddelde?

Standaardisatie

```
normalize = lambda x: (x-x.mean())/x.std()
```

Oefening

- Zoek naar uitschieters in je featureset
- Zijn er features die je moet normaliseren? Welke features moet je absoluut niet normaliseren?

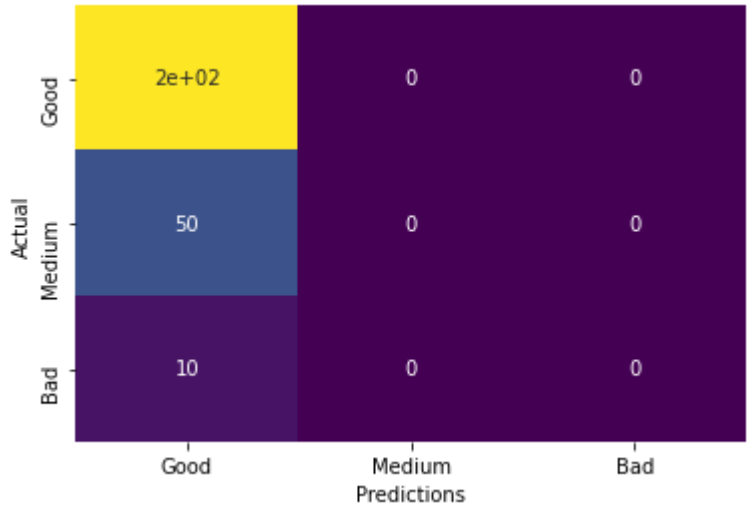
10 minuten

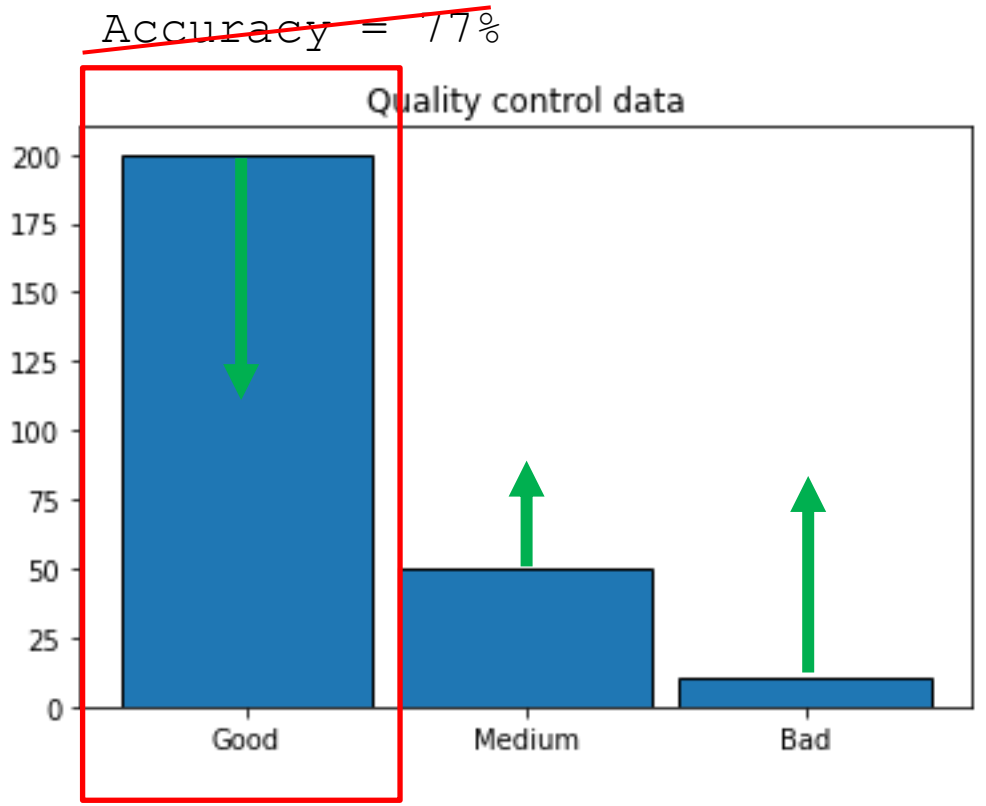
Class imbalance

- Geen goede verdeling tussen classes (vooral probleem bij classificatie)
  - Biased model
  - Slechte evaluatie
- Oplossing
  - Over/down sampeling

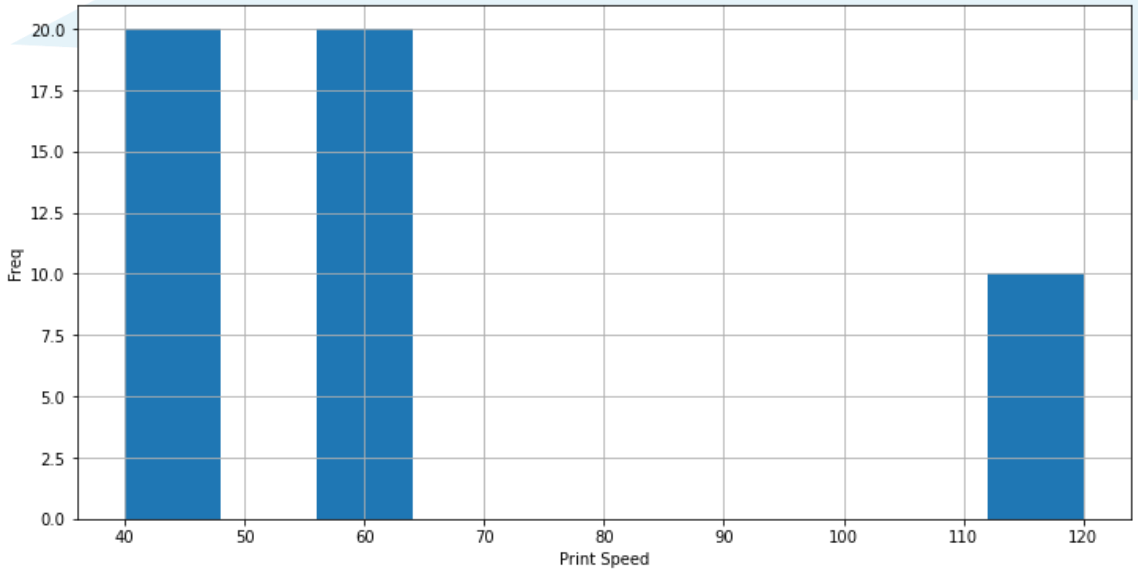
```
from sklearn.utils import resample
```

  - Kies de juiste evaluatie manier
  - Wees kritisch (duh…)

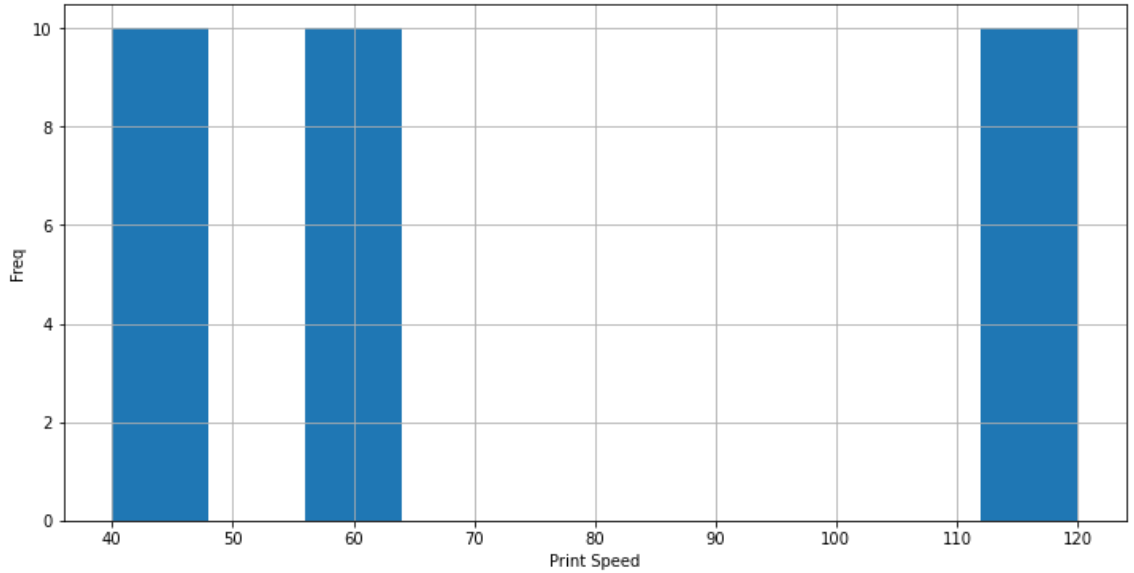




3D printer dataset. Histogram van Print speed




```
from sklearn.utils import resample
data40, data60 = resample(
        df[df.print_speed == 40],
        df[df.print_speed == 60],
        n_samples = sum(df.print_speed == 120),
        replace = True
)
pd.concat([
        data40,
        data60,
        df[df.print_speed == 120]
]).hist()
```



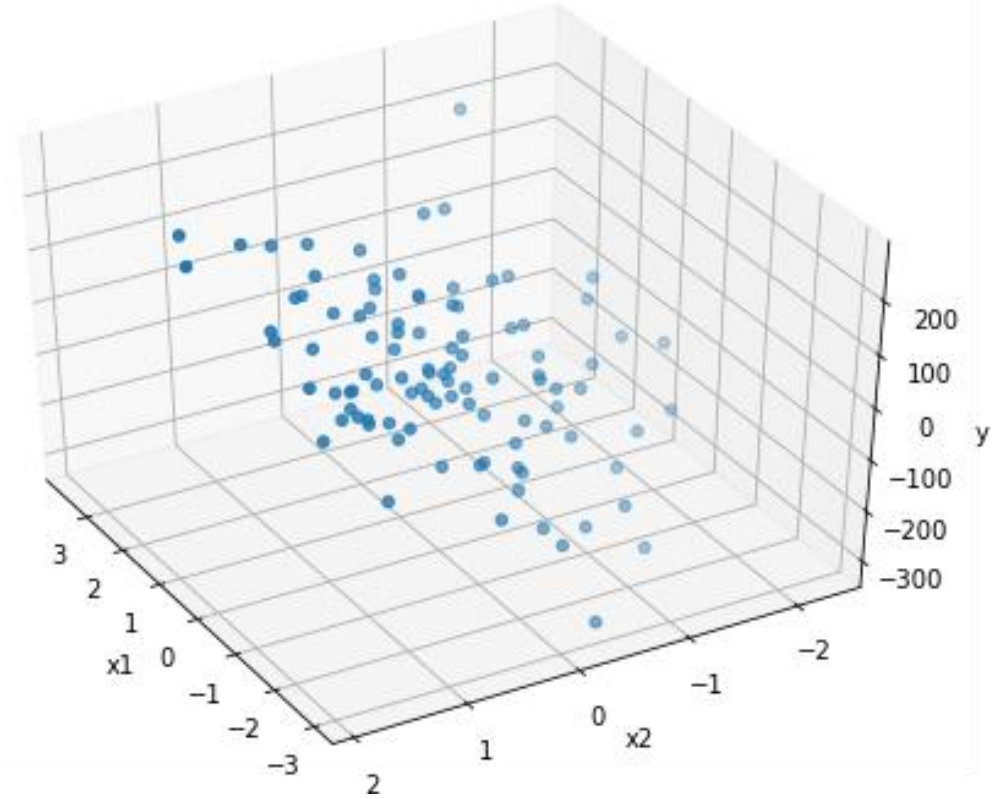
Collineariteit

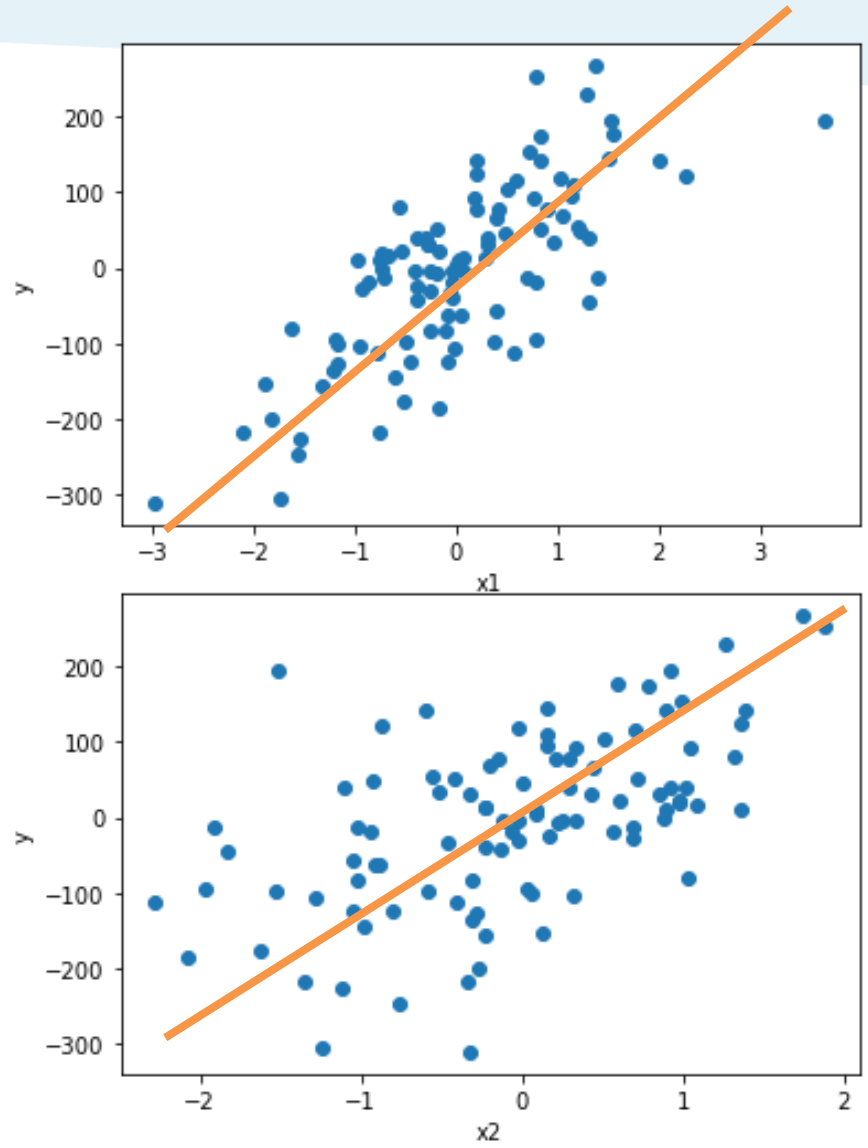
Als $$y = \beta_2 x_2 + \beta_1 x_1 + \beta_0$$

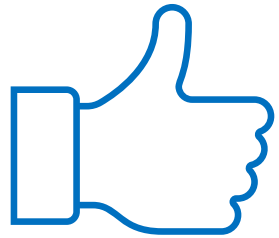
Dan $$x_2 \neq \alpha_1 x_1 + \alpha_0$$

Collineariteit

$$y = \beta_2 x_2 + \beta_1 x_1 + \beta_0$$

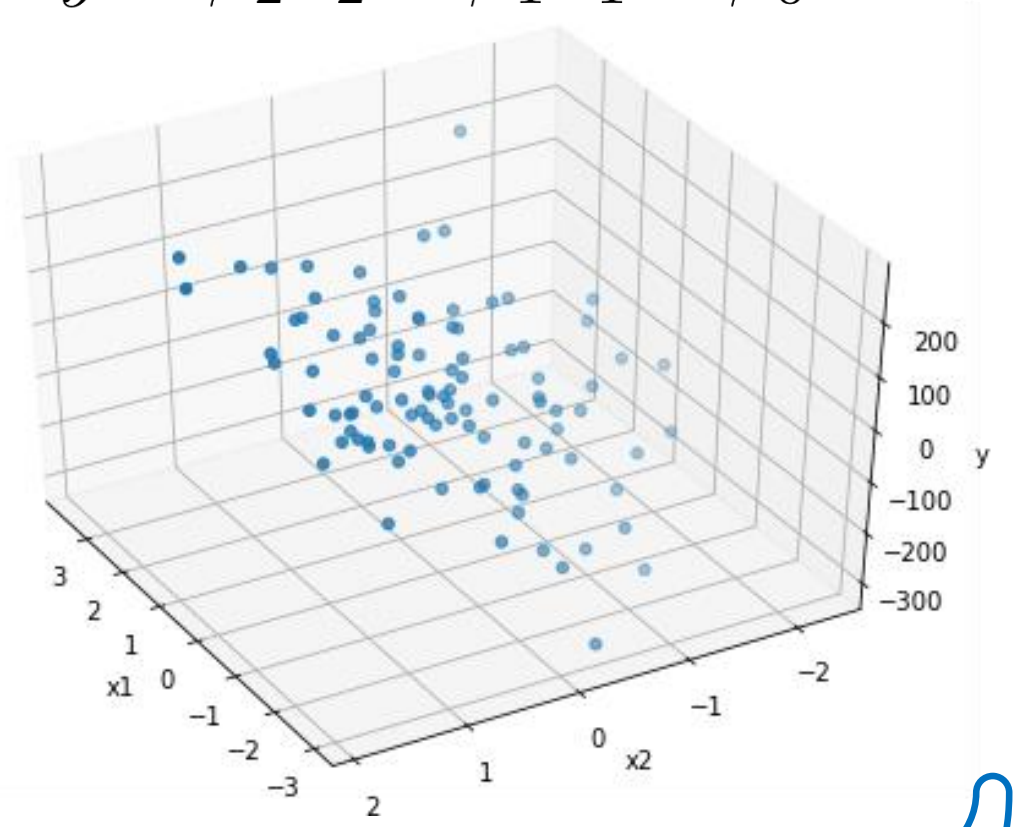


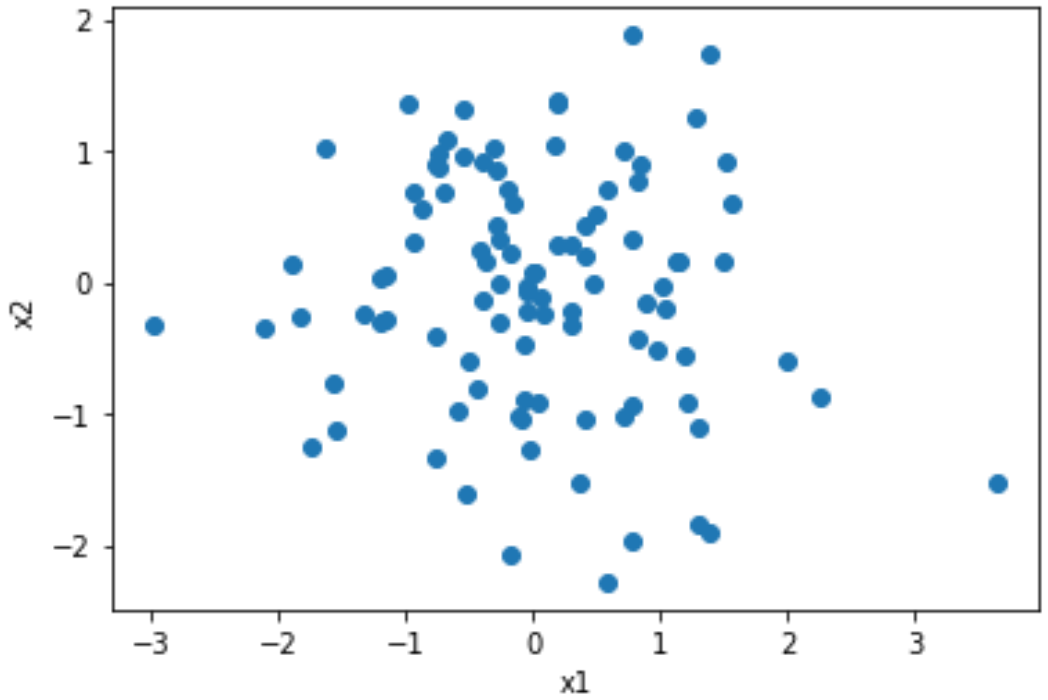


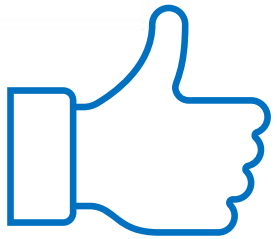


Collineariteit

$$y = \beta_2 x_2 + \beta_1 x_1 + \beta_0$$





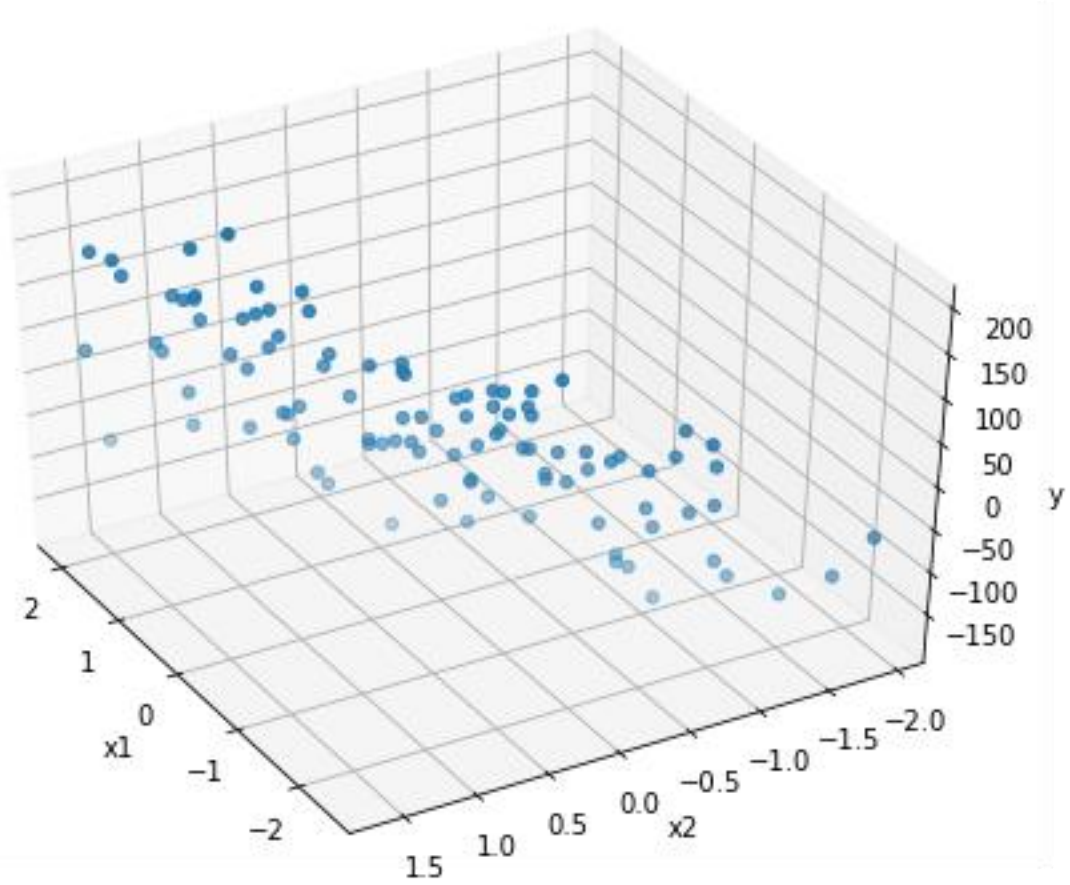


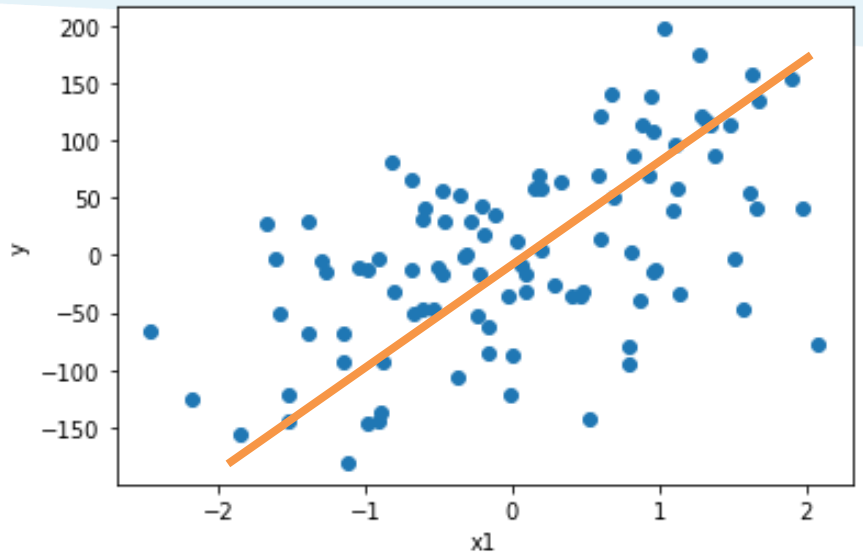
$$x_2 \neq \alpha_1 x_1 + \alpha_0$$

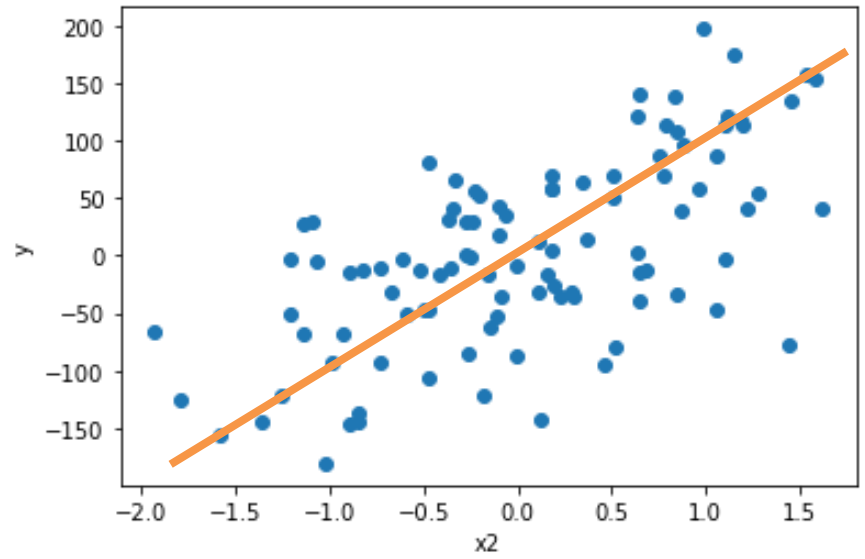
mikrocentrum
opleiden • ontmoeten • ondernemen

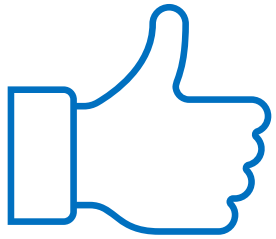
Collineariteit

$$y = \beta_2 x_2 + \beta_1 x_1 + \beta_0$$



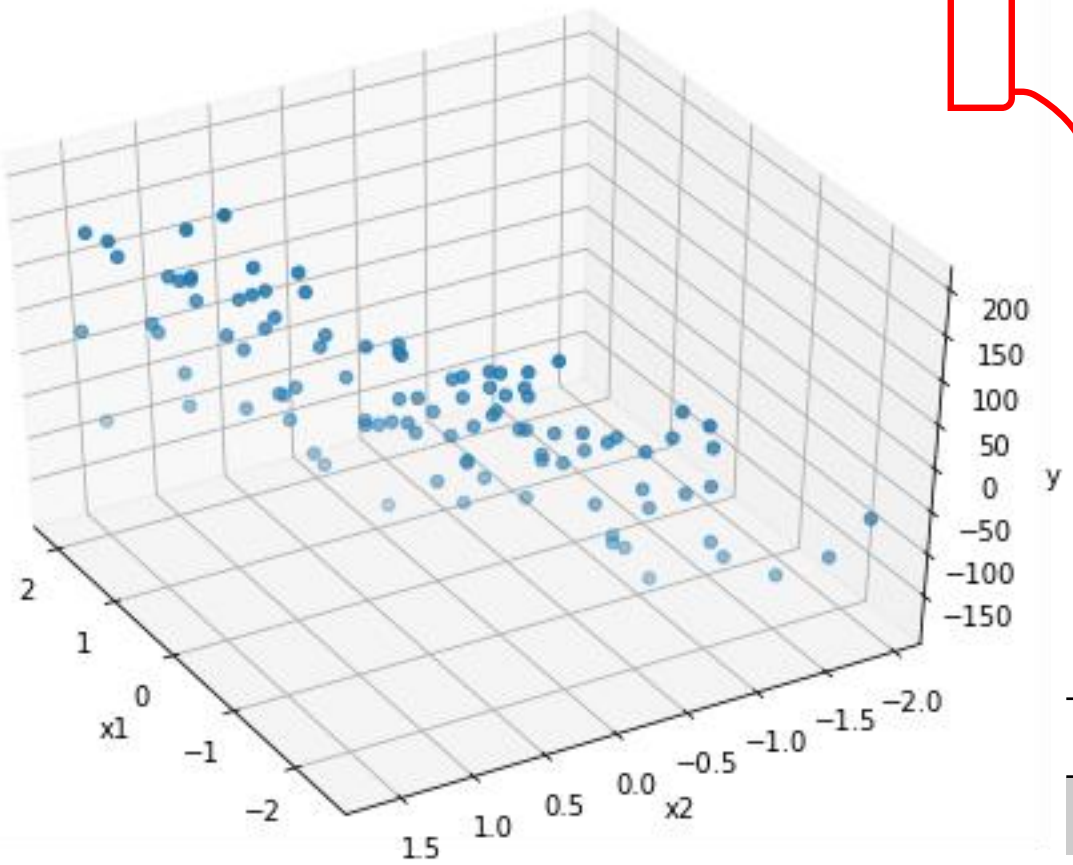




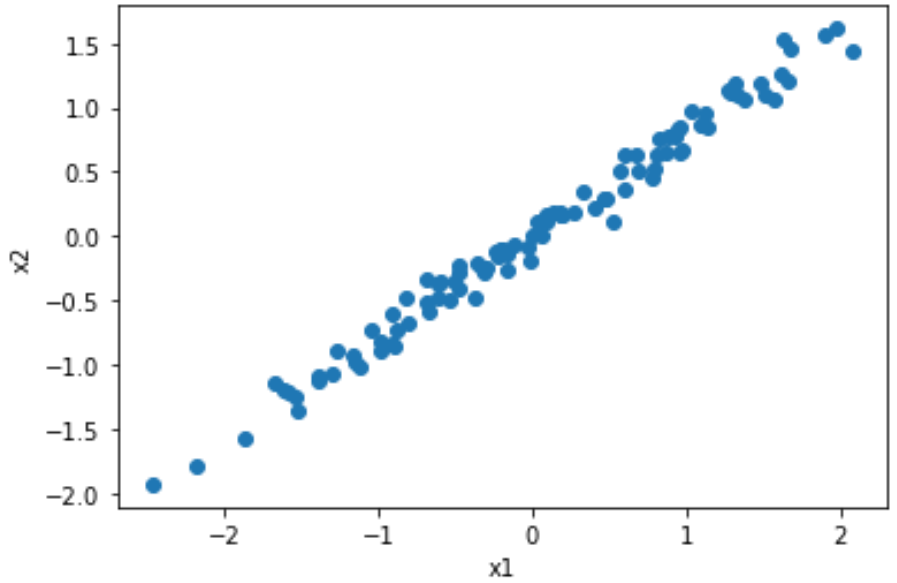


Collineariteit

$$y = \beta_2 x_2 + \beta_1 x_1 + \beta_0$$



$$x_2 \neq \alpha_1 x_1 + \alpha_0$$



Cross correlatie matrix

	y	x1	x2
y	1.000000	0.554169	0.637916
x1	0.554169	1.000000	0.992601
x2	0.637916	0.992601	1.000000

	lifetime	broken	pressureInd	moistureInd	temperatureInd	team	provider	censored
0	56	0	92.178854	104.230204	96.517159	TeamA	Provider4	1
1	81	1	72.075938	103.065701	87.271062	TeamC	Provider4	0
2	60	0	96.272254	77.801376	112.196170	TeamA	Provider1	1
3	86	1	94.406461	108.493608	72.025374	TeamC	Provider2	0
4	34	0	97.752899	99.413492	103.756271	TeamB	Provider1	1

Collineariteit

$$y = \beta_2 x_2 + \beta_1 x_1 + \beta_0 \qquad x_2 \neq \alpha_1 x_1 + \alpha_0$$

```
df.corr()
```

	pressureInd	moistureInd	temperatureInd	TeamA	TeamB	TeamC	Provider1	Provider2	Provider3	Provider4
pressureInd	1.000000	0.002836	-0.020603	0.097637	-0.063948	-0.033603	0.039749	0.049319	-0.072836	-0.018493
moistureInd	0.002836	1.000000	0.002280	-0.025368	0.013754	0.011686	-0.019676	-0.004281	-0.037703	0.063316
temperatureInd	-0.020603	0.002280	1.000000	-0.001231	0.006713	-0.005678	-0.008319	0.018860	0.001831	-0.012931
TeamA	0.097637	-0.025368	-0.001231	1.000000	-0.525821	-0.477549	0.005260	0.003023	0.007715	-0.016478
TeamB	-0.063948	0.013754	0.006713	-0.525821	1.000000	-0.496232	0.034737	-0.012403	0.012879	-0.036213
TeamC	-0.033603	0.011686	-0.005678	-0.477549	-0.496232	1.000000	-0.041250	0.009726	-0.021177	0.054224
Provider1	0.039749	-0.019676	-0.008319	0.005260	0.034737	-0.041250	1.000000	-0.353774	-0.336794	-0.324659
Provider2	0.049319	-0.004281	0.018860	0.003023	-0.012403	0.009726	-0.353774	1.000000	-0.341479	-0.329175
Provider3	-0.072836	-0.037703	0.001831	0.007715	0.012879	-0.021177	-0.336794	-0.341479	1.000000	-0.313375
Provider4	-0.018493	0.063316	-0.012931	-0.016478	-0.036213	0.054224	-0.324659	-0.329175	-0.313375	1.000000

Collineariteit (vif-test)
Variance Inflation Factor

1	broken
2	pressureInd
3	moistureInd
4	temperatureInd
5	TeamA
6	TeamB
7	TeamC
8	Provider1
9	Provider2
10	Provider3
11	Provider4

$$x_1 = \alpha_2 x_2 + \alpha_3 x_3 + \ldots + \alpha_{11} x_{11}$$
$$x_2 = \alpha_1 x_1 + \alpha_3 x_3 + \ldots + \alpha_{11} x_{11}$$

$$\vdots$$

$$\vdots$$

$$x_{11} = \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_{10} x_{10}$$

Collineariteit (vif-test)
Variance Inflation Factor

$$\mathrm{VIF}_i = \frac{1}{1 - R_i^2}$$

	variable	VIF
1	broken	1.014712
2	pressureInd	1.016746
3	moistureInd	1.007758
4	temperatureInd	1.006792
5	TeamA	inf
6	TeamB	inf
7	TeamC	inf
8	Provider1	inf
9	Provider2	inf
10	Provider3	inf
11	Provider4	inf

$$x_1 = \alpha_2 x_2 + \alpha_3 x_3 + \ldots + \alpha_{11} x_{11}$$
$$x_2 = \alpha_1 x_1 + \alpha_3 x_3 + \ldots + \alpha_{11} x_{11}$$

$$\vdots$$

$$\vdots$$

$$x_{11} = \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_{10} x_{10}$$

Collineariteit (vif-test)
Variance Inflation Factor

	variable	VIF
0	Intercept	0.000000
1	broken	1.014712
2	pressureInd	1.016746
3	moistureInd	1.007758
4	temperatureInd	1.006792
5	TeamA	inf
6	TeamB	inf
7	TeamC	inf
8	Provider1	inf
9	Provider2	inf
10	Provider3	inf
11	Provider4	inf

	variable	VIF
0	Intercept	150.813964
1	broken	1.014712
2	pressureInd	1.016746
3	moistureInd	1.007758
4	temperatureInd	1.006792
5	TeamB	1.324964
6	TeamC	1.316672
7	Provider2	1.496661
8	Provider3	1.472065
9	Provider4	1.491600

> 5 ✗

mikrocentrum
opleiden • ontmoeten • ondernemen

VIF-test uitvoeren

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from patsy import dmatrices

y, X = dmatrices("""lifetime ~ broken + pressureInd + moistureInd +
       temperatureInd + TeamB + TeamC + Provider2 + Provider3 + Provider4""",
       data=df_dummied, return_type='dataframe')

vif = pd.DataFrame()
vif['variable'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
```

Oefening

- Van de 3d printer dataset. Welke variabelen verwacht je dat collineair zijn?

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
filename = '../3dprinter/data.csv'
df = pd.read_csv(filename)
```

In [3]:
```
df.head()
```

Out[3]:

	layer_height	wall_thickness	infill_density	infill_pattern	nozzle_temperature	bed_temperature	print_speed	material	fan_speed	roughness	tension_strenght
0	0.02	8	90	grid	220	60	40	abs	0	25	18
1	0.02	7	90	honeycomb	225	65	40	abs	25	32	16
2	0.02	1	80	grid	230	70	40	abs	50	40	8
3	0.02	4	70	honeycomb	240	75	40	abs	75	68	10
4	0.02	6	90	grid	250	80	40	abs	100	92	5

Extra leesmateriaal

- Make a pipeline: https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html#sklearn.pipeline.Pipeline
- Lifelines talk: https://www.youtube.com/watch?v=XQfxndJH4UA
- Anomaly detection: https://scikit-learn.org/stable/modules/outlier_detection.html
- Dealing with imbalance: https://www.youtube.com/watch?v=6M2d2n-QXCc

Dagcasus

- In de map Dagcasus staan 2 notebooks: "regressie_opdracht" en "feature_opdracht". Dit is huiswerk voor de volgende keer.
- regressie_opdracht: maak een regressie model om de kwaliteit van een product te voorspellen aan de hand van onbekende features
- "feature_opdracht": maak features op basis van de time_series.csv dataset. Deze dataset heeft meer dan 200 cycles gedraaid met elke cycle tussen de ~250 datapunten. Maak van elke cycle een set features op de kolom "sensor".

Dagcasus

- In de map Dagcasus staan 2 notebooks: "regressie_opdracht" en "feature_opdracht". Dit is huiswerk voor de volgende keer.
- regressie_opdracht: maak een regressie model om de kwaliteit van een product te voorspellen aan de hand van onbekende features
- "feature_opdracht": maak features op basis van de time_series.csv dataset. Deze dataset heeft meer dan 200 cycles gedraaid met elke cycle tussen de ~250 datapunten. Maak van elke cycle een set features op de kolom "sensor".
  - Zoek naar uitschieters in je featureset
  - Zijn er features die je moet normaliseren? Welke features moet je absoluut niet normaliseren?