Machine Learning

5. modelevaluatie en hyperparameter tuning

ML Actueel

TechCrunch

AMD to supply 6GW of compute capacity to OpenAI in chip deal worth tens of billions

AMD has signed a multi-year chip supply deal with OpenAI that could generate tens of billions in revenue for the chipmaker, helping accelerate its momentum in the AI industry.

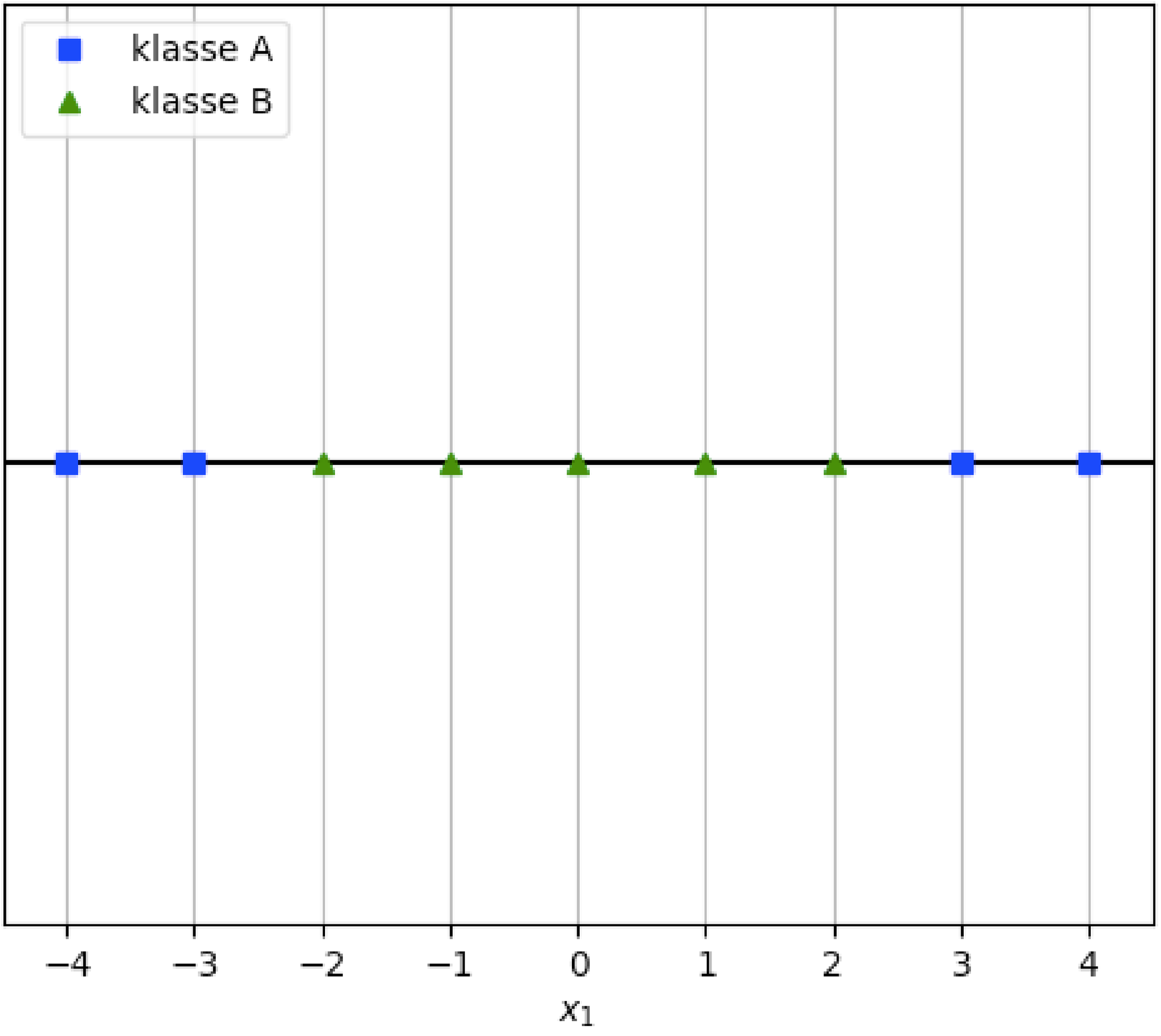
AMD has agreed to supply 6 gigawatts of compute capacity to OpenAI — enough to power up to 4.5 million homes — across multiple generations of its Instinct GPUs, starting with the Instinct MI450 GPU. OpenAI will receive the first gigawatt of capacity in the second half of 2026, when the new chip is scheduled for deployment.

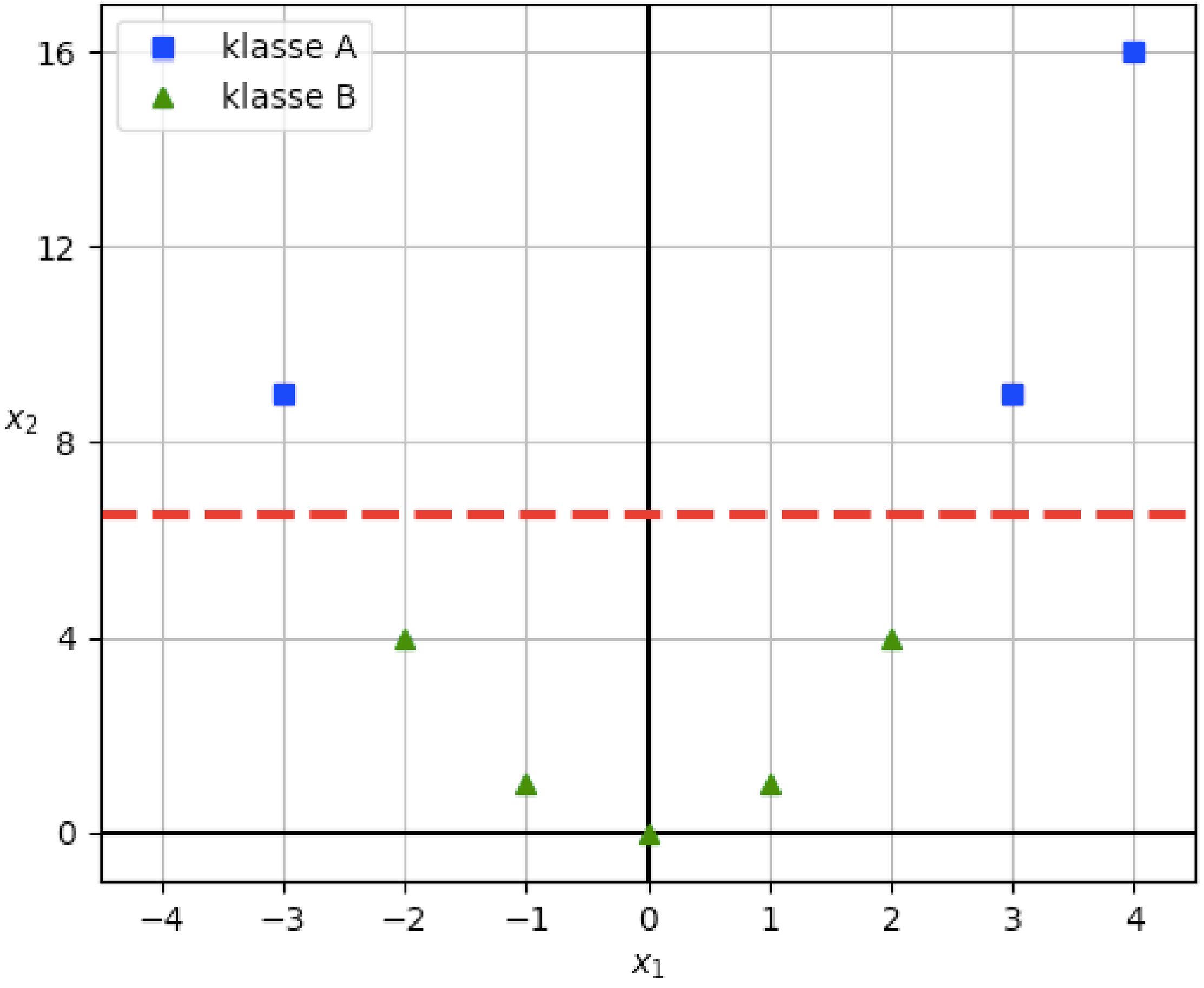
AMD claims the MI450 series will outperform Nvidia's comparable offerings (Nvidia Rubin CPX) through hardware and software improvements, many of which will be made with OpenAI's input. Its current MI355X and MI300X series GPUs, currently used in some workloads for OpenAI, are already strong for AI inference in large language models due to their large memory capacity and bandwidth.

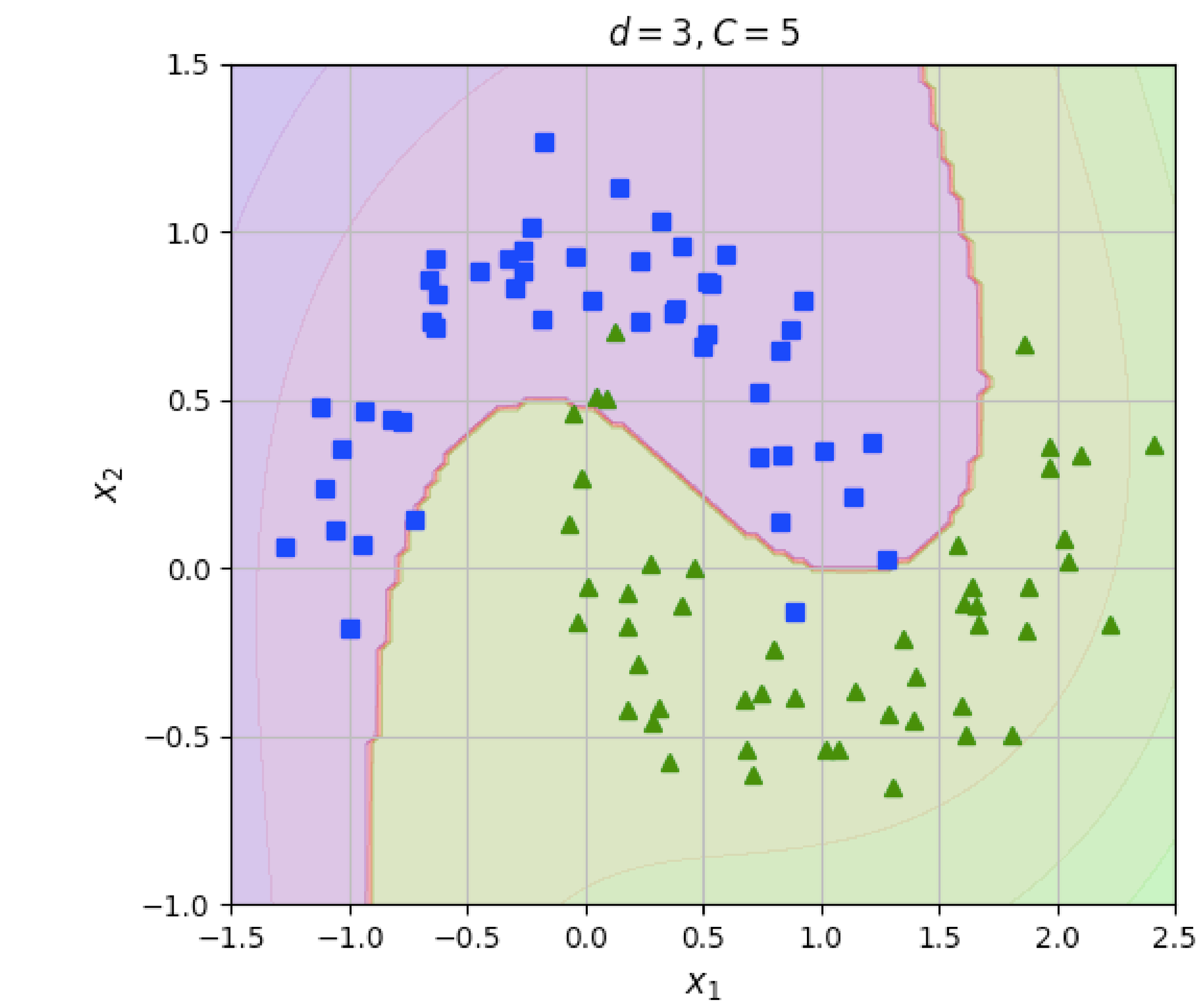
Het is gebruikelijk om bij zulke miljardendeals strategische verbintenissen aan te gaan in de vorm van deelnames. Minder gebruikelijk is de financiële constructie. OpenAI, dat dit jaar 13 miljard dollar omzet hoopt te halen, lijdt de komende jaren nog verlies en lijkt op een risicovolle manier geld rond te pompen; Nvidia wil 100 miljard dollar steken in OpenAI. OpenAI steekt nu geld in Nvidia's voornaamste concurrent, AMD, en bestelt voor 300 miljard dollar aan datacenters bij Oracle (4,5 gigawatt). Oracle bestelt weer voor 40 miljard dollar chips bij Nvidia.

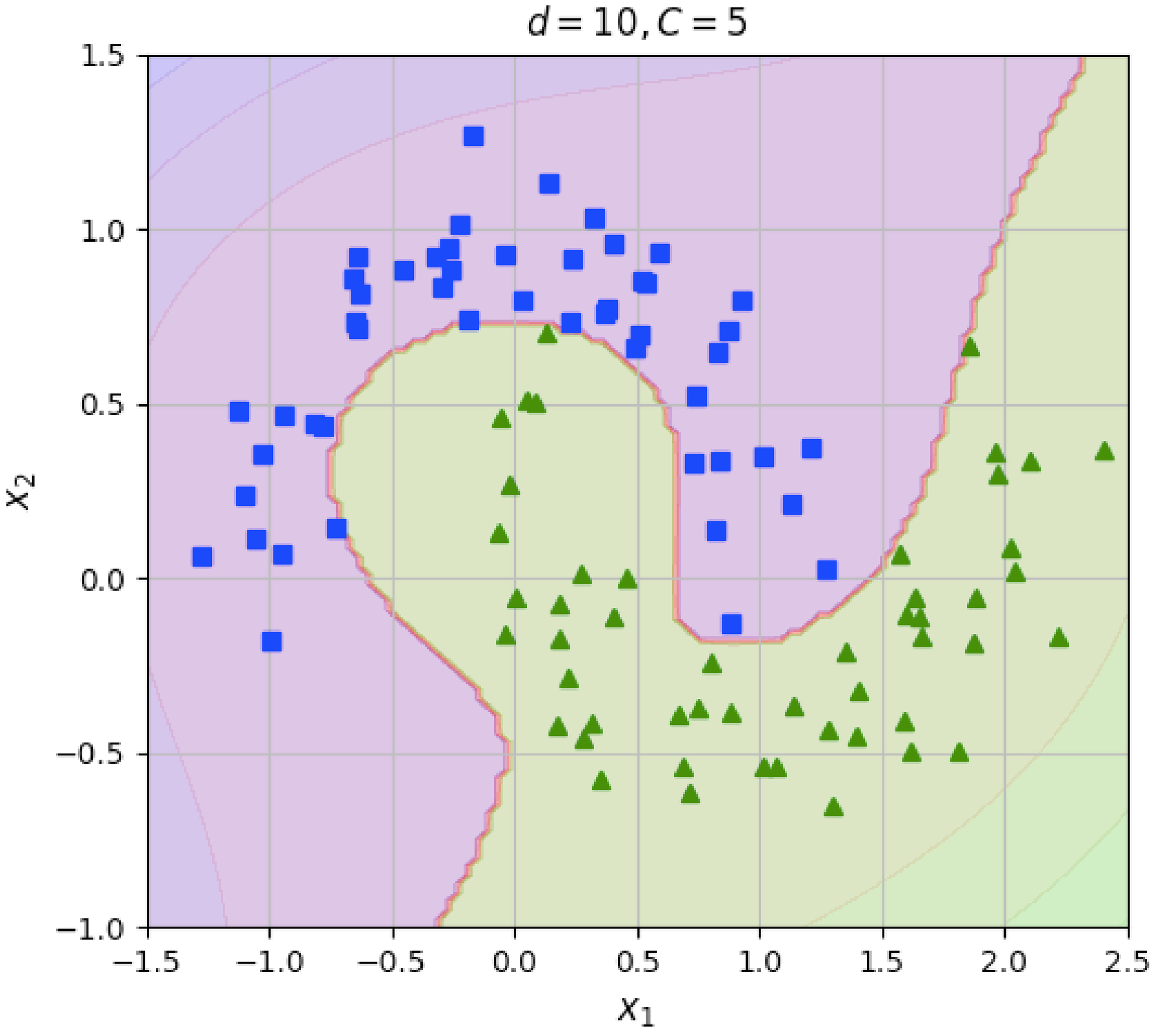
https://www.nrc.nl/nieuws/2025/10/06/weer-een-mega-chipdeal-openai-koopt-voor-tientallen-miljarden-aan-amd-chips-a4908613

ml:polynomial

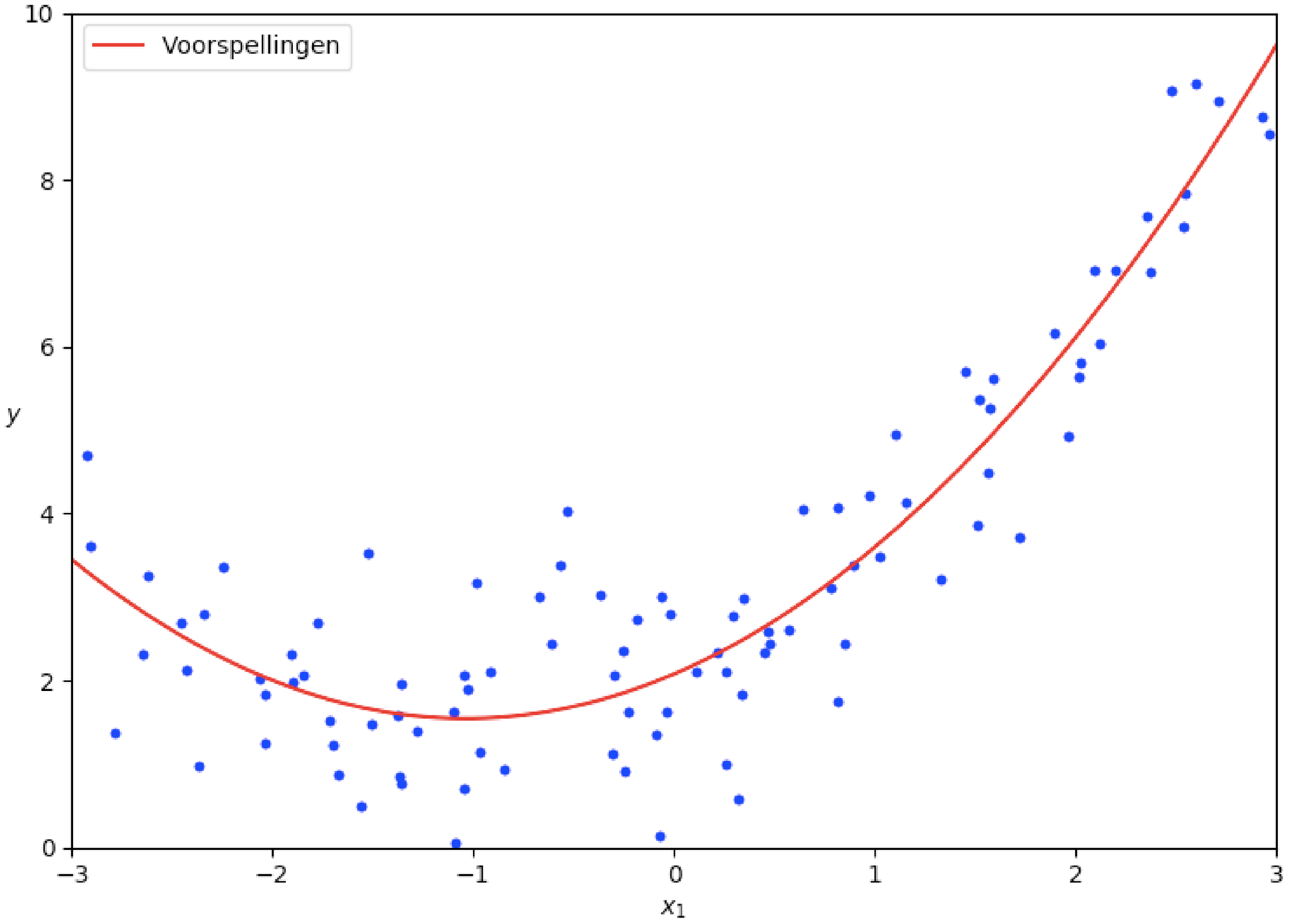

klasse A
klasse B
−4
−3
−2
−1
0
1
2
3
4
x_1

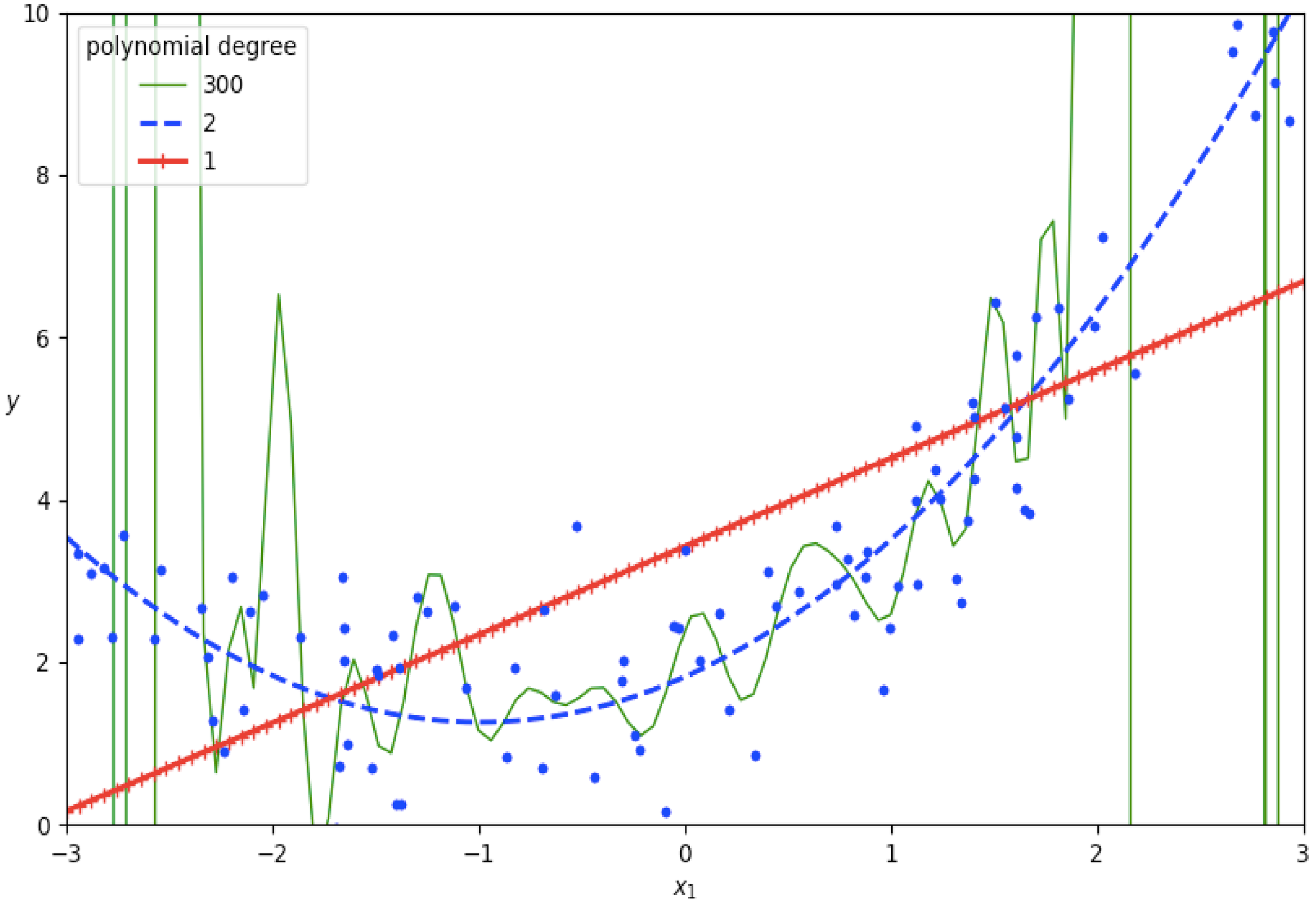

klasse A
klasse B
16
12
8
4
0
x_2
−4
−3
−2
−1
0
1
2
3
4
x_1


$d = 3, C = 5$
1.5
1.0
0.5
0.0
−0.5
−1.0
x_2
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
x_1


$d = 10, C = 5$
1.5
1.0
0.5
0.0
−0.5
−1.0
x_2
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
x_1

ml:overfitting and underfitting


Voorspellingen
10
8
6
4
2
0
y
−3
−2
−1
0
1
2
3
x_1


polynomial degree
300
2
1
10
8
6
4
2
0
y
−3
−2
−1
0
1
2
3
x_1

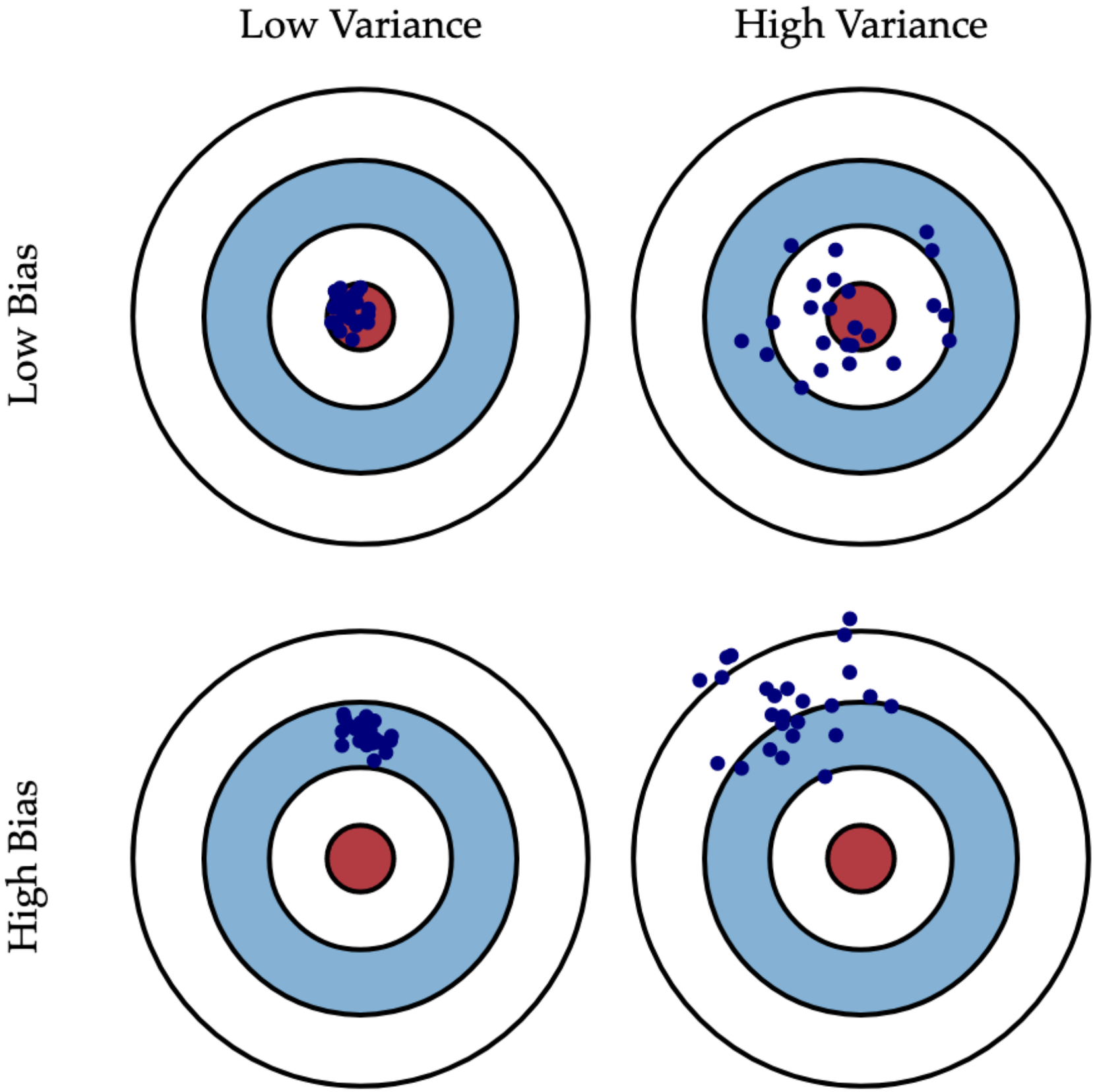
Voorkomen van overfitting

Verminderen van het aantal *features*

- Welke *features* wel en niet te gebruiken?
- Welk algoritme past het best?

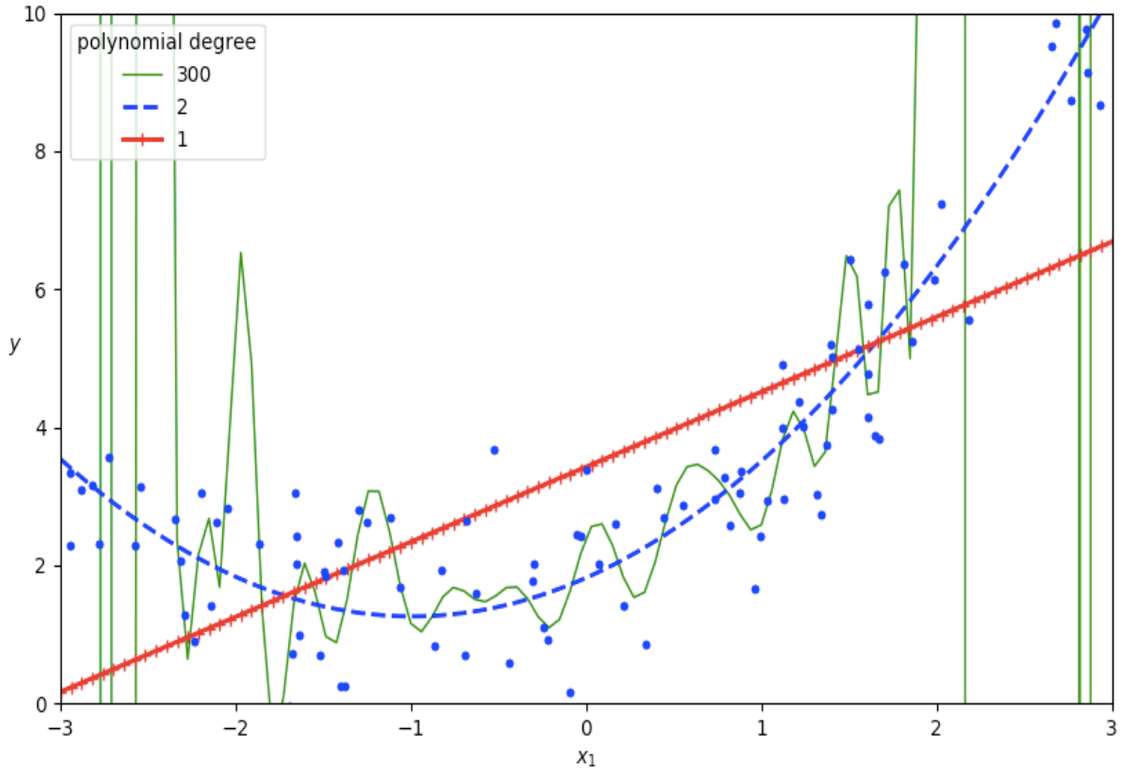
Regularisatie

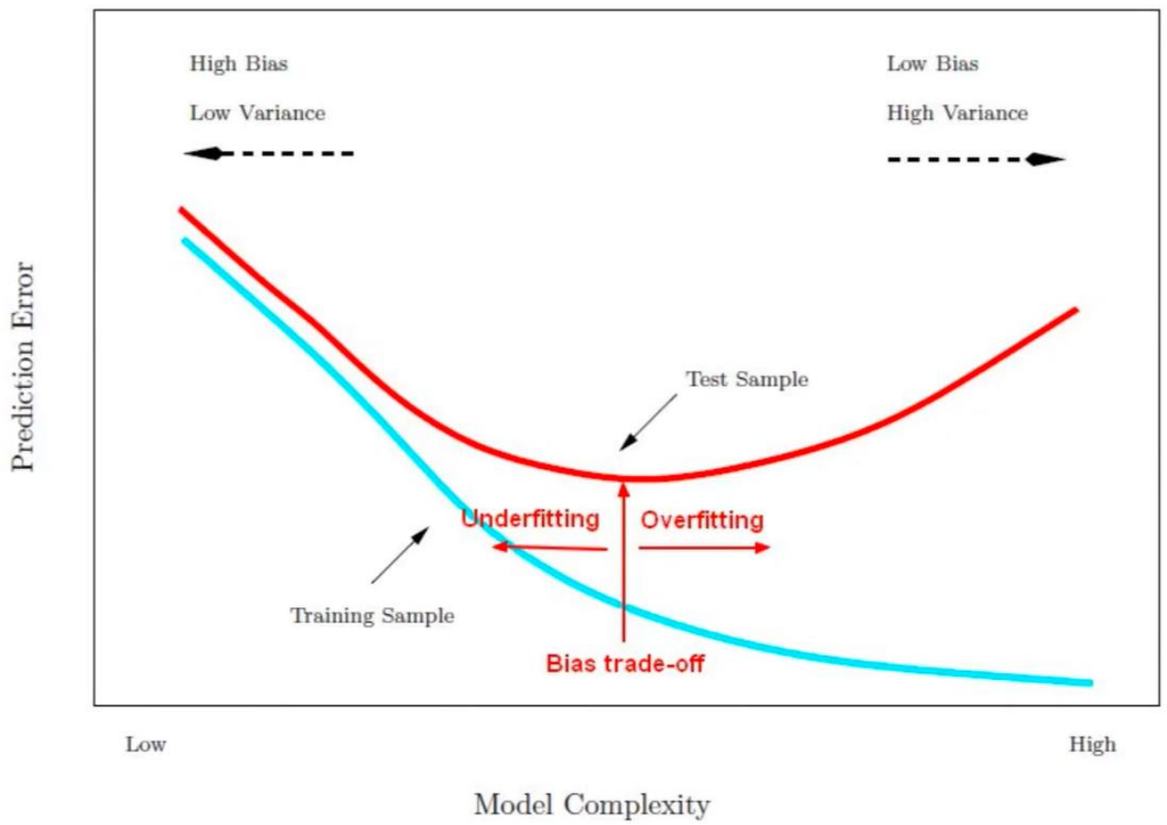
- Verminderen van de grootte en complexiteit van θ



http://scott.fortmann-roe.com/docs/BiasVariance.html

bias variance tradeoff



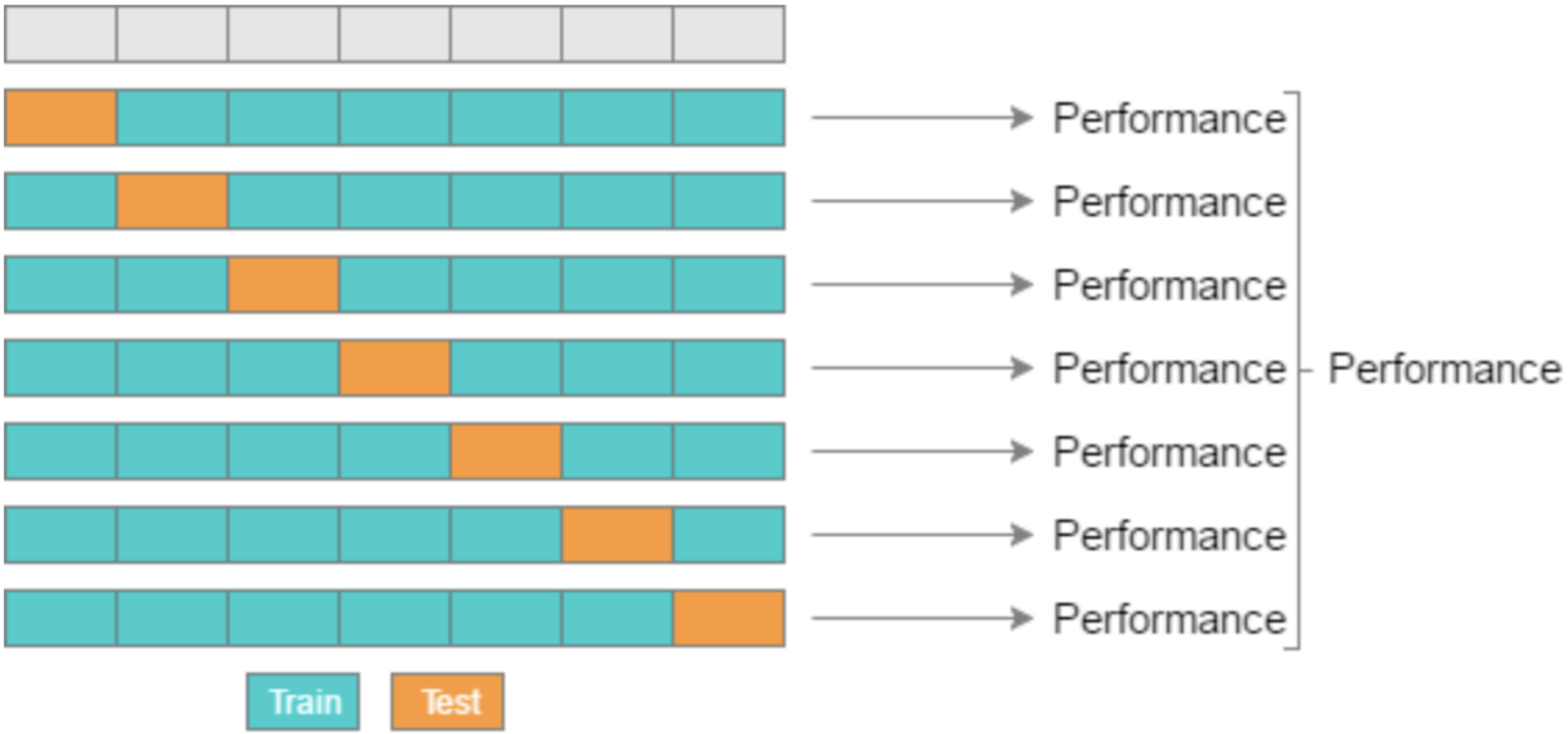


Bron: https://datalab.com.ng/wp-content/uploads/2020/05/Bias-Variance-Tradeoff.png

ml:trainings-set, cv-set en test-set

$$X = \begin{bmatrix} — & (x^{(1)})^T & — \\ — & (x^{(2)})^T & — \\ — & (x^{(3)})^T & — \\ & \vdots & \\ — & (x^{(m)})^T & — \end{bmatrix}$$

Cross-validation



https://towardsdatascience.com/cross-validation-a-beginners-guide-5b8ca04962cd

Maak verschillende hypothesen voor $\theta^{(d)}$

Bepaal $J_{cv}(\theta)$voor elk van deze hypothesen

Selecteer de laagste $J_{cv}(\theta)$

Bepaal de generalisatie van deze hypothese
met behulp van $J_{test}(\theta^{(sd)})$

Verkoopprijs huizen Groningen

trainings-data
cv-data

verkoopprijs (k€)

250
200
150
100
50

10
110
210
310
410

grootte (m^2)

ml:confusion matrix

de *confusion matrix*

– voorspeld –

– werkelijk –

True Negative (TN)	False Positive (FP)
False Negative (FN)	True Positive (TP)

– voorspeld –

– werkelijk –

TN:
voorspeld als niet-spam, is ook geen spam

FP:
voorspeld als spam, is geen spam

FN:
voorspeld als niet-spam, is wel spam

TP:
voorspeld als spam, is ook spam

de *confusion matrix*

– voorspeld –

– werkelijk –

voorspeld als niet-5 is ook geen 5	voorspeld als 5 is geen 5
voorspeld als niet-5 is wel een 5	voorspeld als 5 is ook een 5

Beoordelingsmaten

$$Accuracy = \frac{n_{TP} + n_{TN}}{n_{total}}$$

Wanneer werkt deze maat wel en wanneer niet?

$$recall(TPR) = \frac{TP}{TP + FN}$$

$$precision(PPV) = \frac{TP}{TP + FP}$$

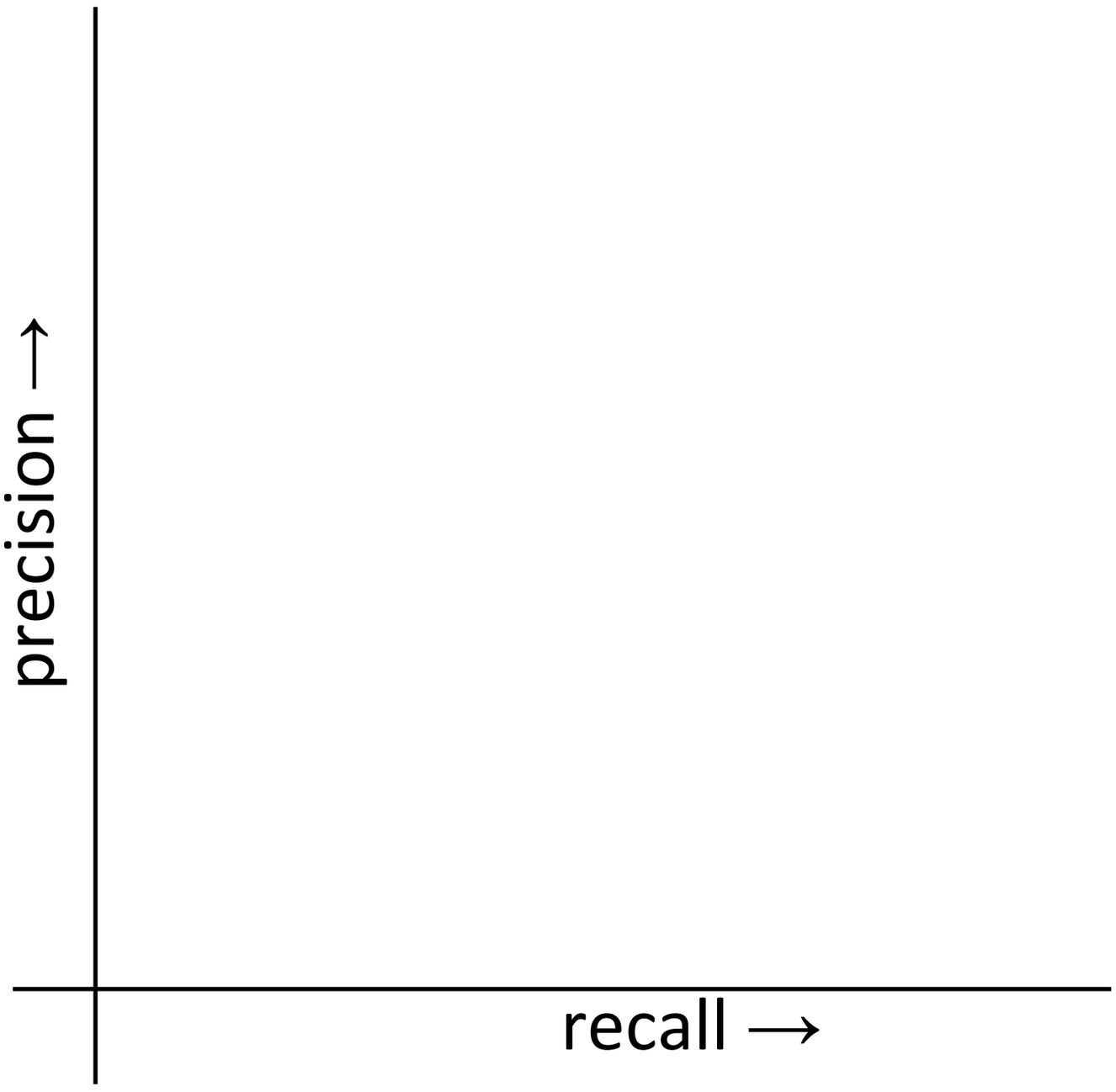
$$specificity(TNR) = \frac{TN}{TN + FP}$$

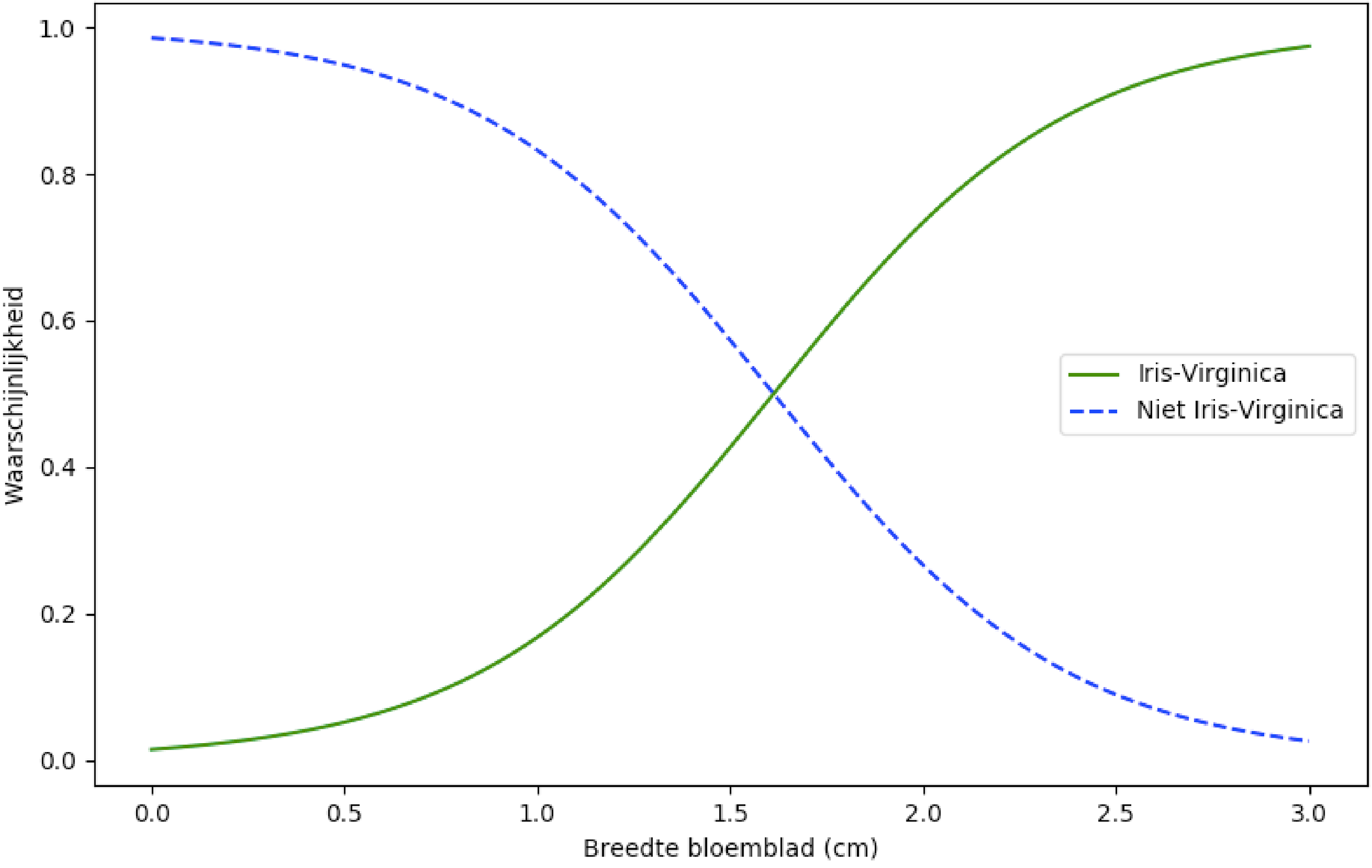
$$fall - out(FPR) = \frac{FP}{FP + TN}$$

$$F1\text{-}score = 2 \times \frac{P \times R}{P + R}$$

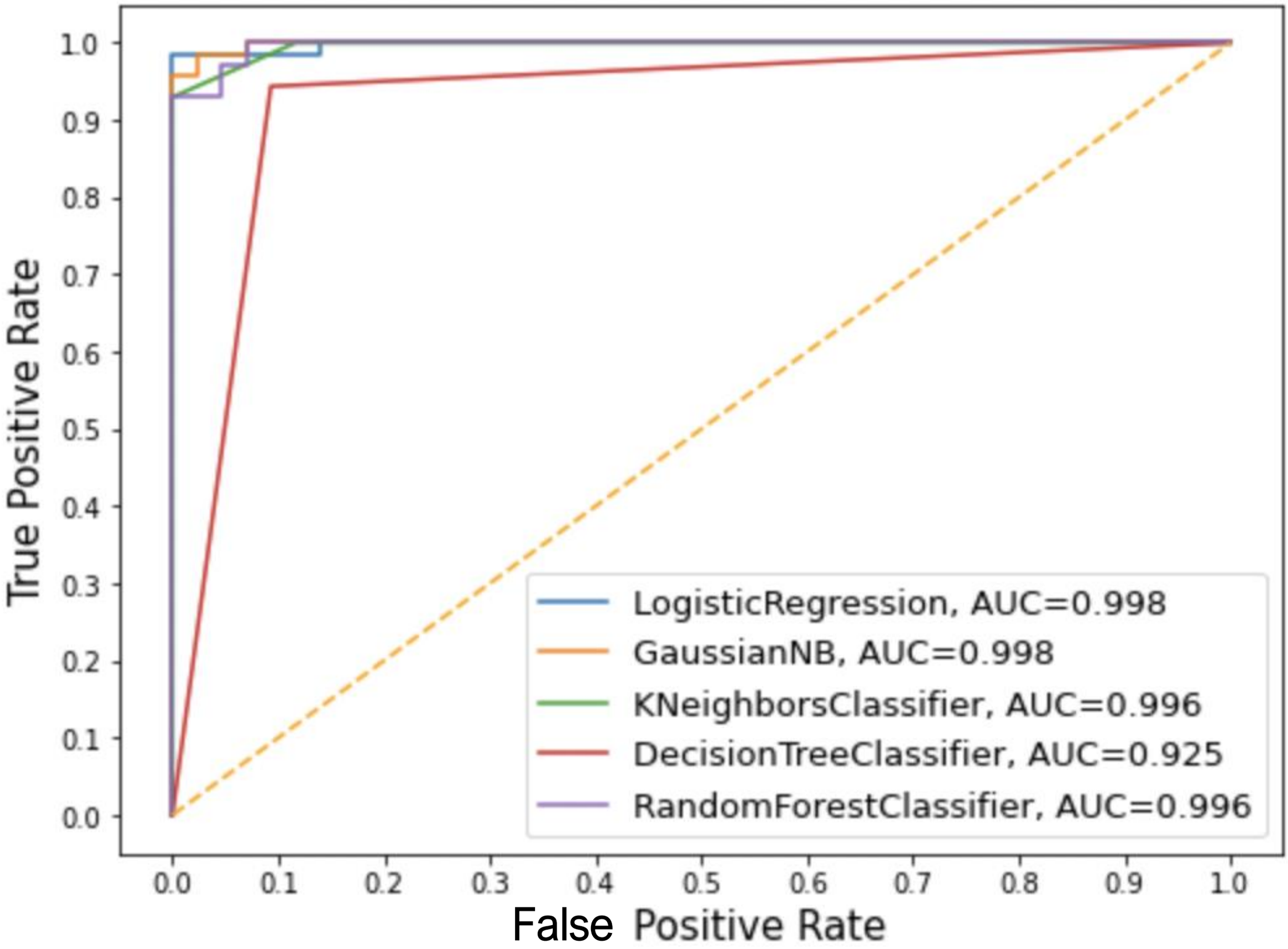
Precision	Recall	Gemiddelde	F1-score
0,5	0,4	0,45	0,44
0,7	0,1	0,4	0,175
0,02	1,0	0,51	0,039

Precision en *recall* gaan over de *zekerheid*
waarmee we een voorspelling kunnen doen.




1.0
0.8
0.6
0.4
0.2
0.0
Waarschijnlijkheid
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Breedte bloemblad (cm)
Iris-Virginica
Niet Iris-Virginica


TPR →
FPR →


ROC Curve Analysis
True Positive Rate
False Positive Rate
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
LogisticRegression, AUC=0.998
GaussianNB, AUC=0.998
KNeighborsClassifier, AUC=0.996
DecisionTreeClassifier, AUC=0.925
RandomForestClassifier, AUC=0.996

Intermezzo: live coding

Notebook over Modelevaluatie

ml:hyperparameter tuning

Hoe nu een ML-probleem adresseren?

Welk algoritme past het best bij dit probleem?

Wat zijn de beste waarden van θ ?

Wat is de beste graad van de polynoom?

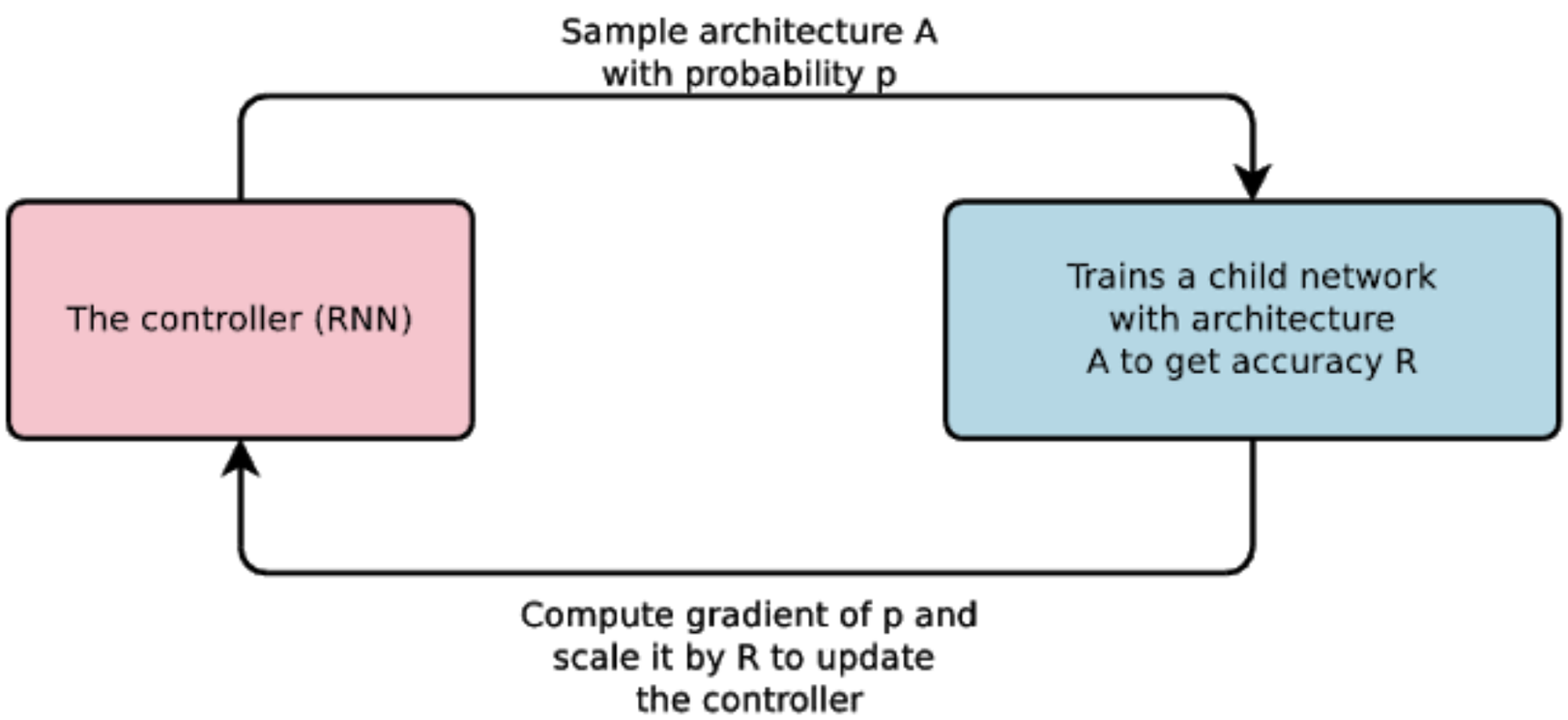
Wat is de beste kernel die ik kan gebruiken?

Wat is de beste architectuur van het neurale netwerk?

Wat zijn de beste waarden van de initiële parameters?

Hyperparameter tuning

- **Hyper**parameters
  - Parameters van het model, niet van de data
  - Bijv. SVC: kernel, degree, C, gamma… (volgende week meer hierover)
- Vraag: welke combinatie van hyperparameters is de beste voor mijn ML-model?
- Oplossing: gebruik ML!



https://ai.googleblog.com/2017/05/using-machine-learning-to-explore.html

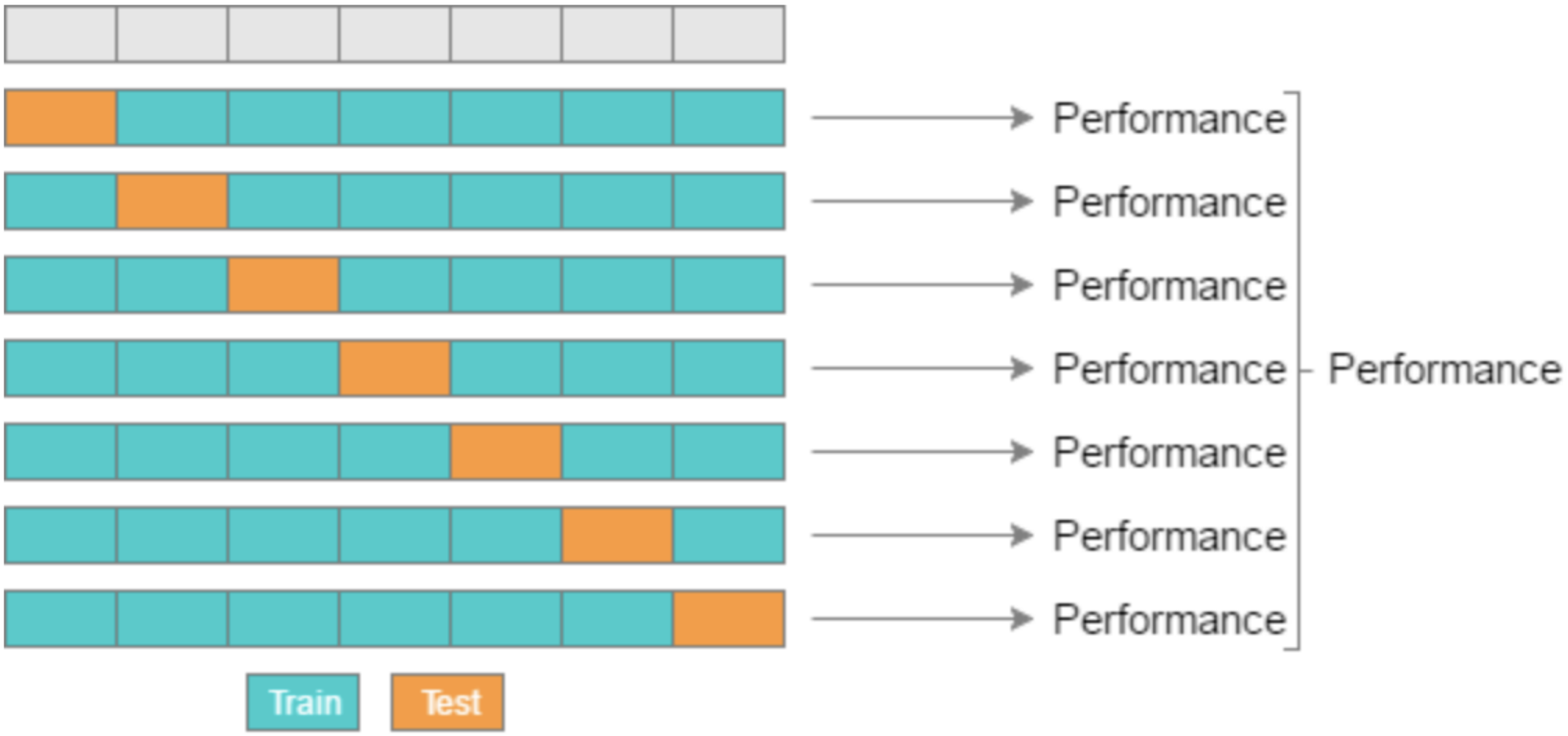
Hyperparameter tuning

- Bijvoorbeeld in de DBSCAN-opdracht (set 3, opgave 3)

```
min_samples = [3, 4, 5, 7, 9]
eps = [.1, .4, .55, .6, 1, 2]
```

```
for s in min_samples:
    for e in eps:
        # Make and train model
        dbs = DBSCAN(eps=e, min_samples=s)
        dbs.fit(X, y)
        # Evaluate
        pct_core = len(dbs.core_sample_indices_) / m
        n_clusters = len(np.unique(dbs.labels_[dbs.labels_ > -1]))
        pct_noise = len(dbs.labels_[dbs.labels_ == -1]) / m
        results.append([s,e,pct_core,n_clusters, pct_noise])
```

CV = Cross-validation (herhaling)



https://towardsdatascience.com/cross-validation-a-beginners-guide-5b8ca04962cd

GridSearchCV

- Probeert alle combinaties van de opgegeven parameters uit
  - param_grid
- Per combinatie Cross Validation met *cv* folds
- Kan erg lang bezig zijn
  - Toevoegen van een parameter met *n* waardes maakt doorlooptijd *n* x langer

GridSearchCV in code

sklearn.model_selection.GridSearchCV

class **sklearn.model_selection.GridSearchCV**(*estimator, param_grid, *, scoring=None, n_jobs=None, refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', error_score=nan, return_train_score=False*) [source]

- Na het fitten zijn o.a. op te vragen:
  - cv_results_
  - best_score_
  - best_params_
- Voorbeeld zometeen in Notebook

RandomizedSearchCV

- Performt veel beter dan GridSearchCV
  - Toevoegen van parameters maakt niet uit voor de doorlooptijd
- Werkt soms zelfs beter dan GridSearchCV
  - Van continue variabelen *kan* hiermee elke waarde aan bod komen
- Parameter *n_iter* bepaalt hoeveel iteraties je wilt doen
  - In elke iteratie wordt een random combinatie geprobeerd
  - *param_grid* => ***param_distributions***

RandomizedSearchCV in code

sklearn.model_selection.RandomizedSearchCV

class **sklearn.model_selection.RandomizedSearchCV**(*estimator, param_distributions, *, n_iter=10, scoring=None, n_jobs=None, refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', random_state=None, error_score=nan, return_train_score=False*) [source]

- Na het fitten zijn o.a. op te vragen:
  - cv_results_
  - best_score_
  - best_params_
- Voorbeeld in Notebook

Halving

- HalvingGridSearchCV en HalvingRandomSearchCV
- Werken in rondes:
  - In de eerste ronde wordt een groot aantal combinaties van parameters gegenereerd
  - Van elke combinatie wordt een model gemaakt
  - Modellen worden getraind op een kleine **subset** van de trainingsdata => minder resources nodig
  - Alleen de beste modellen (1/factor) gaan door naar de volgende ronde
  - Elke ronde minder modellen…
  - …en dus een grotere trainingsset mogelijk…
  - …want meer resources beschikbaar

Halving in code

sklearn.model_selection.HalvingGridSearchCV

class **sklearn.model_selection.HalvingGridSearchCV**(*estimator*, *param_grid*, ***, *factor=3*, *resource='n_samples'*, *max_resources='auto'*, *min_resources='exhaust'*, *aggressive_elimination=False*, *cv=5*, *scoring=None*, *refit=True*, *error_score=nan*, *return_train_score=True*, *random_state=None*, *n_jobs=None*, *verbose=0*) [source]

sklearn.model_selection.HalvingRandomSearchCV

class **sklearn.model_selection.HalvingRandomSearchCV**(*estimator*, *param_distributions*, ***, *n_candidates='exhaust'*, *factor=3*, *resource='n_samples'*, *max_resources='auto'*, *min_resources='smallest'*, *aggressive_elimination=False*, *cv=5*, *scoring=None*, *refit=True*, *error_score=nan*, *return_train_score=True*, *random_state=None*, *n_jobs=None*, *verbose=0*) [source]

Afsluiting: live coding

- Notebook over (Halving) Grid & Randomized Search