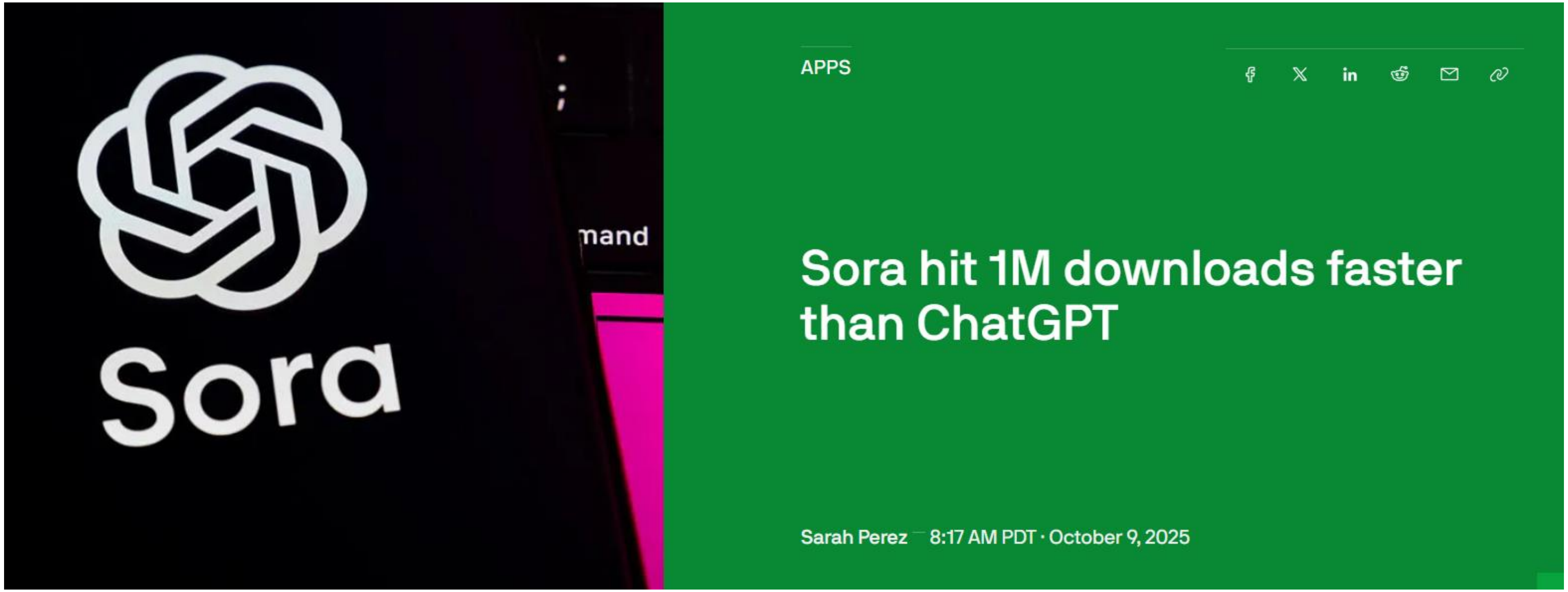
Machine Learning
6. andere modellen en ensemble learning

ML Actueel



https://openai.com/nl-NL/index/sora-2/

Classificatie van classificatie-algoritmen

- Linear classifiers
    - Fisher's linear discriminant
    - Logistic regression
    - **Naive Bayes classifier**
    - Perceptron
- **Support vector machines**
    - Least squares support vector machines
- Quadratic classifiers
- Kernel estimation
    - k-nearest neighbor
- **Boosting (meta-algorithm)**
- **Decision trees**
    - Random forests
- Neural networks
- Learning vector quantization
- Unsupervised clustering
    - **k-means**
    - **DBSCAN**

Onderwerp 1: andere (classificatie)modellen

- Naive Bayes
- Support Vector Classifiers/Machines
- Clustering
  - k-means
  - DBSCAN
- Decision trees

Bayes

$$P(A \mid B) = \frac{P(A)P(B \mid A)}{P(B)}$$

Posterior
(P to compute)

Prior
(Estimation)

Likelihood
(counted)

$$P(H \mid D) = \frac{P(H)P(D \mid H)}{P(D)}$$

Normalising constant

Naive Bayes

Aanname: alle features zijn even belangrijk en zijn onafhankelijk van elkaar

$$P(H \mid D) = \frac{P(H)P(D \mid H)}{P(D)}$$

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

Voorbeeld: kans dat iemand lenzen nodig heeft (y) gegeven een aantal eigenschappen (x1 t/m x4)

AGE			SPECTACLE PRESCRIPTION			ASTIGMATISM			TEAR PROD RATE			LENSES RECOMMENDED	
	YES	NO		YES	NO		YES	NO		YES	NO	YES	NO
YOUNG	2	3	MYOPE	2	2	WAAR	3	4	REDUCED	6	2	9	5
PREPRESBYOPIC	4	0	HYPERMETROPE	4	2	ONWAAR	6	1	NORMAL	3	3		
PRESBYOPIC	3	2	NORMAL	3	1								
YOUNG	2/9	3/5	MYOPE	2/9	2/5	WAAR	3/9	4/5	REDUCED	6/9	2/5		
PREPRESBYOPIC	4/9	0/5	HYPERMETROPE	4/9	2/5	ONWAAR	0/9	1/5	NORMAL	3/9	3/5		
PRESBYOPIC	3/9	2/5	NORMAL	3/9	1/5								

AGE	SPECTACLE PRESCRIPTION	ASTIGMATISM	TEAR PROD RATE
YOUNG	NORMAL	WAAR	NORMAL
2/9	3/9	3/9	3/9

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

$$p(Yes) = 9/14 \times 2/9 \times 3/9 \times 3/9 \times 3/9 = 0{,}0053$$
$$p(No) = 5/14 \times 3/5 \times 1/5 \times 4/5 \times 3/5 = 0{,}0206$$

$$p(Yes) = 0.0053/(0.0053 + 0.0206) = 0.205$$
$$p(No) = 0.0206/(0.0053 + 0.0206) = 0.795$$

Naive Bayes in Code

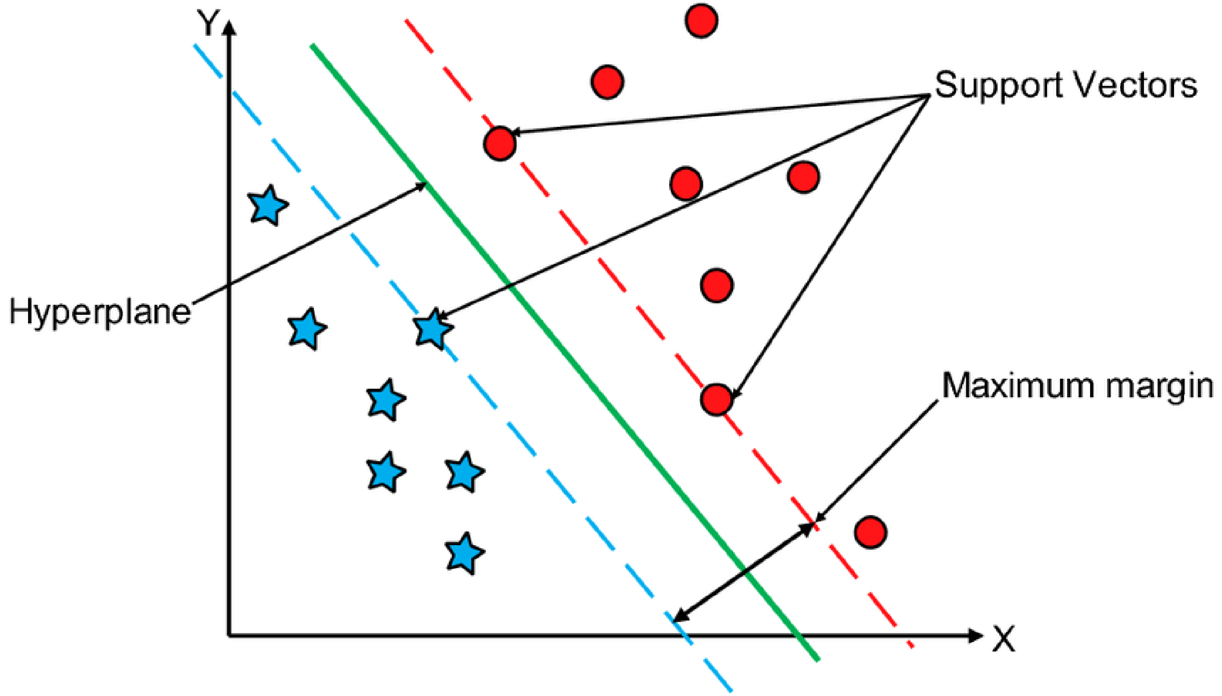
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.5,
                                                    random_state=0)
gnb = GaussianNB()
y_pred = gnb.fit(X_train, y_train).predict(X_test)
```

https://scikit-learn.org/stable/modules/naive_bayes.html

NB: Gaussian wordt gebruikt als de features (ongeveer) normaal verdeeld zijn

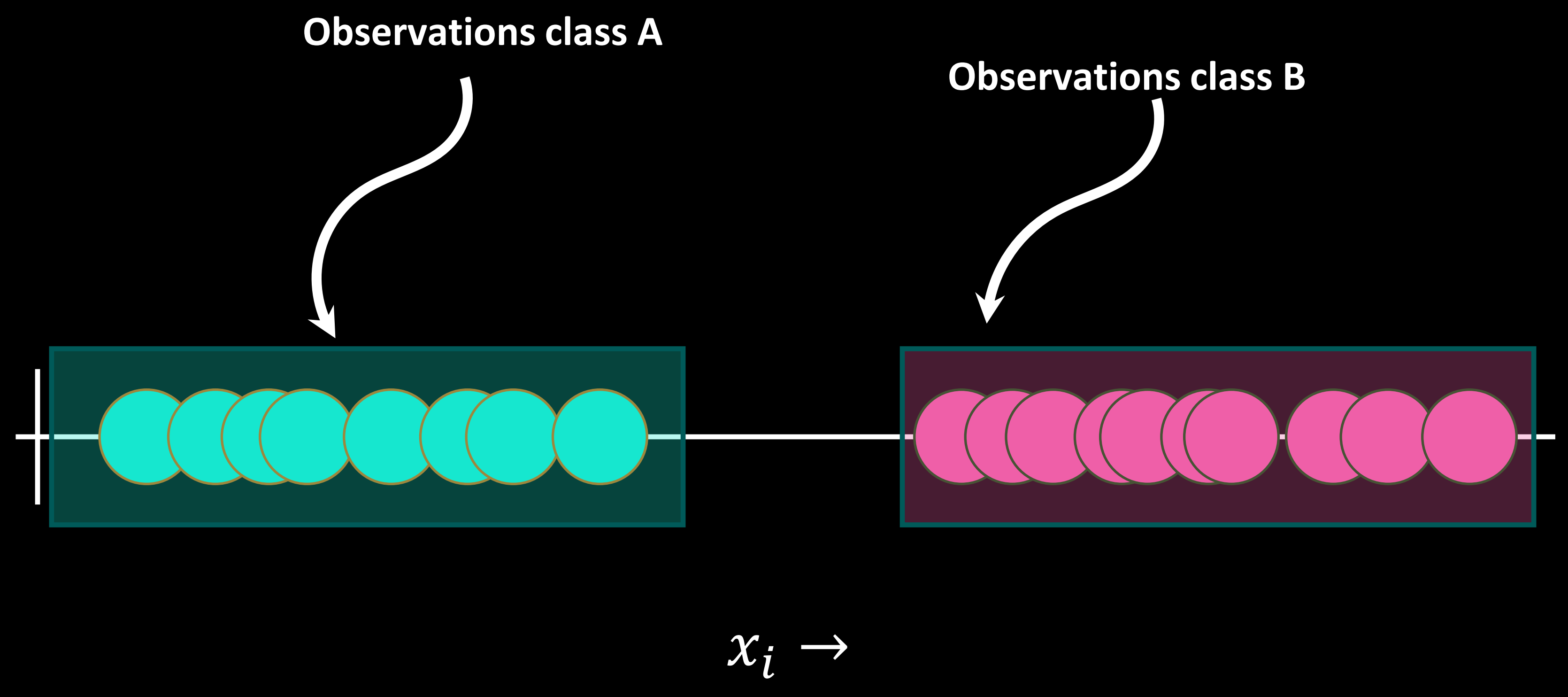
Support Vector Classifiers (SVC)

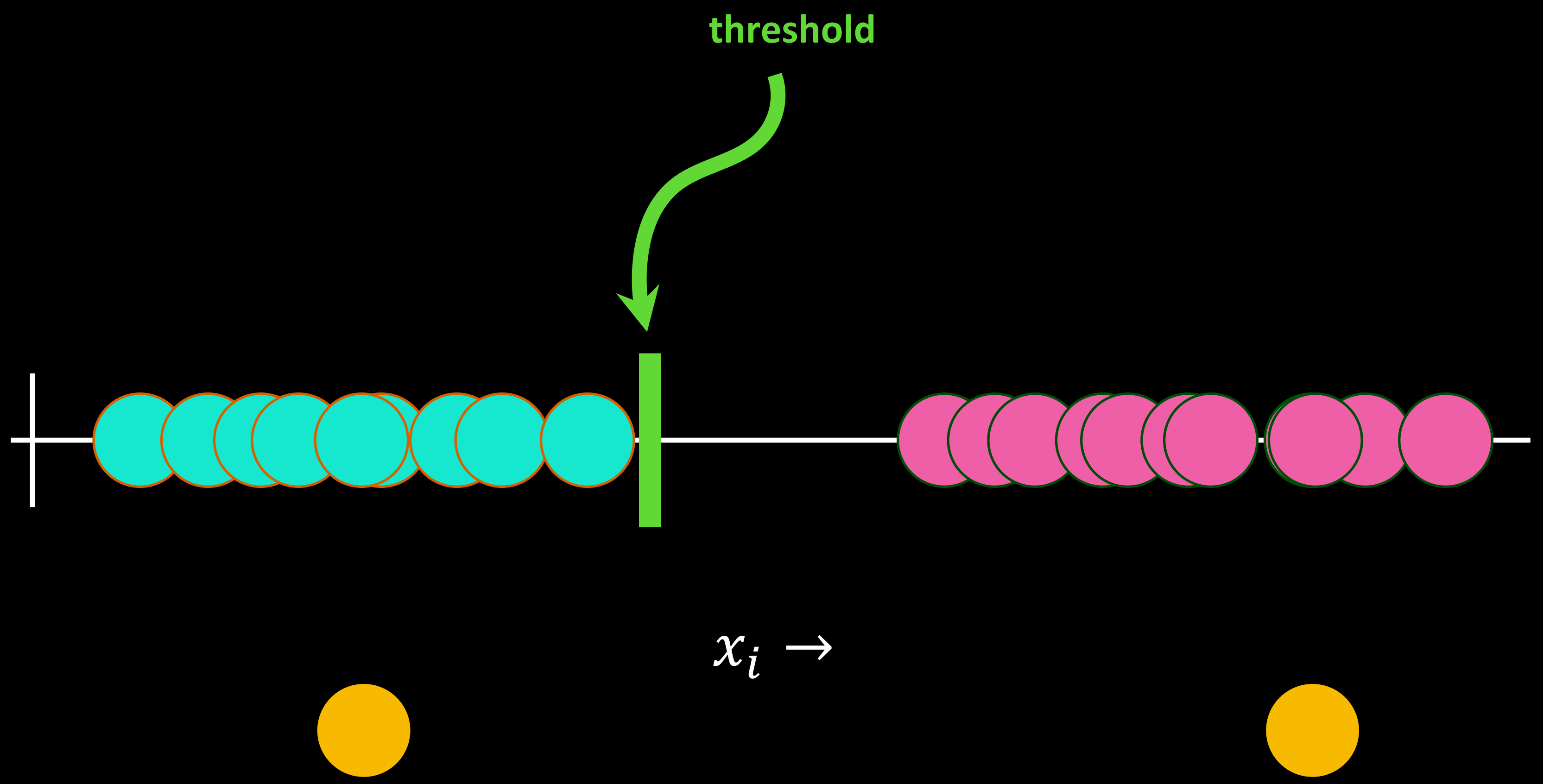
- Doel: vinden van een optimale *decision boundary* tussen waarnemingen van verschillende klassen
  - Lijn (2D)
  - Vlak (3D)
  - Hyperplane (nD)
- Kunnen goed omgaan met outliers (uitbijters)

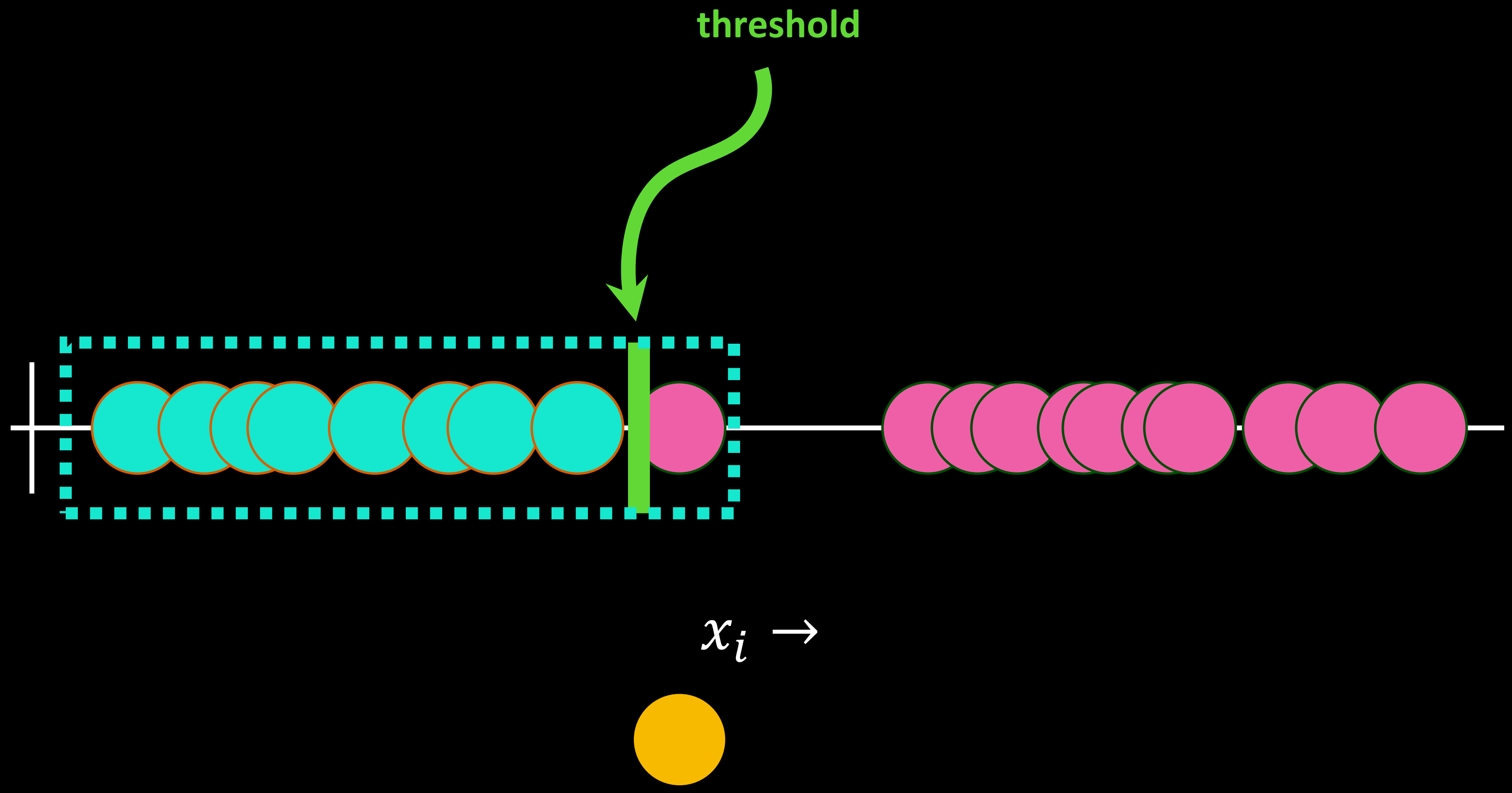


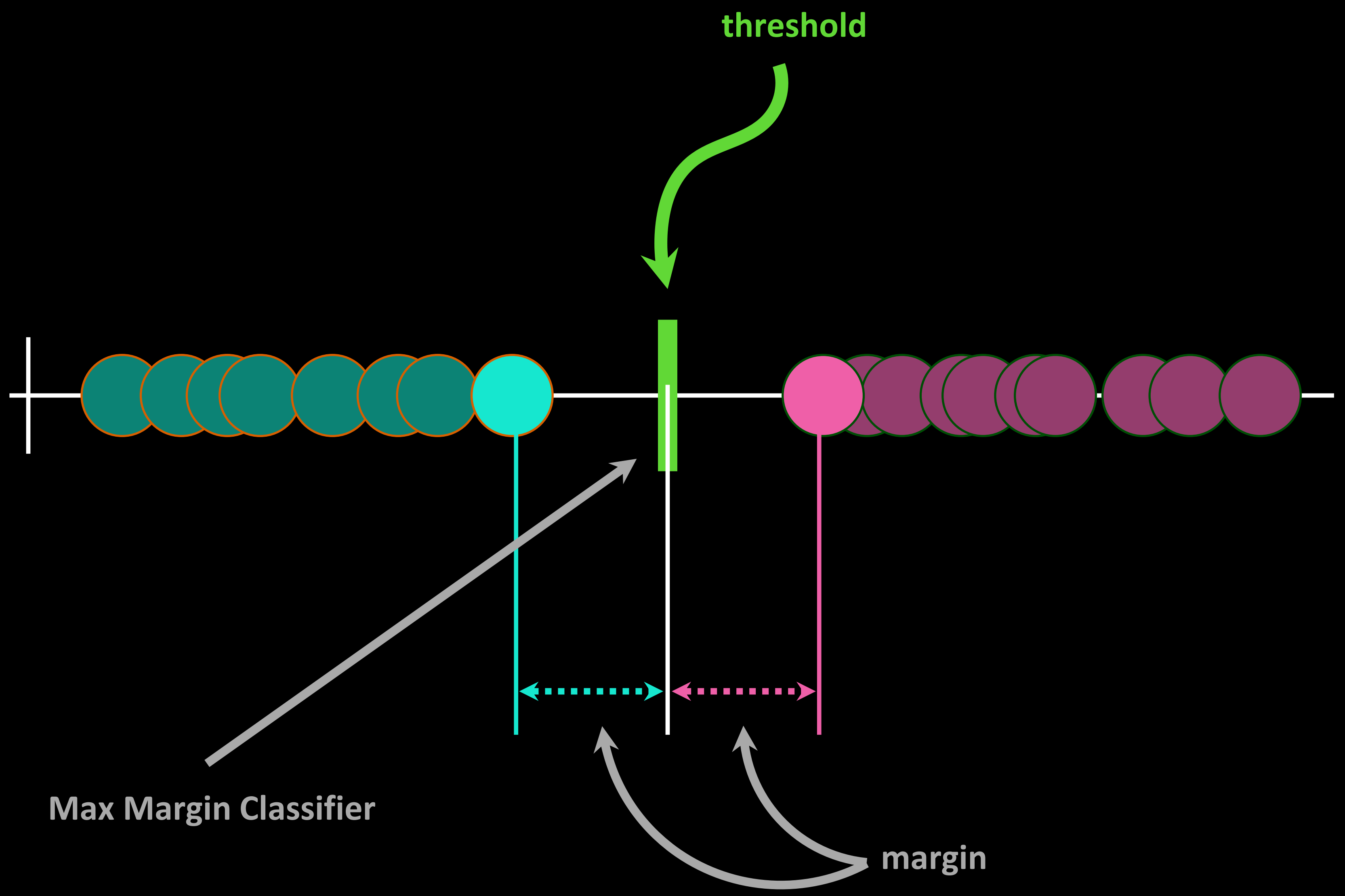
Bron: researchgate.net

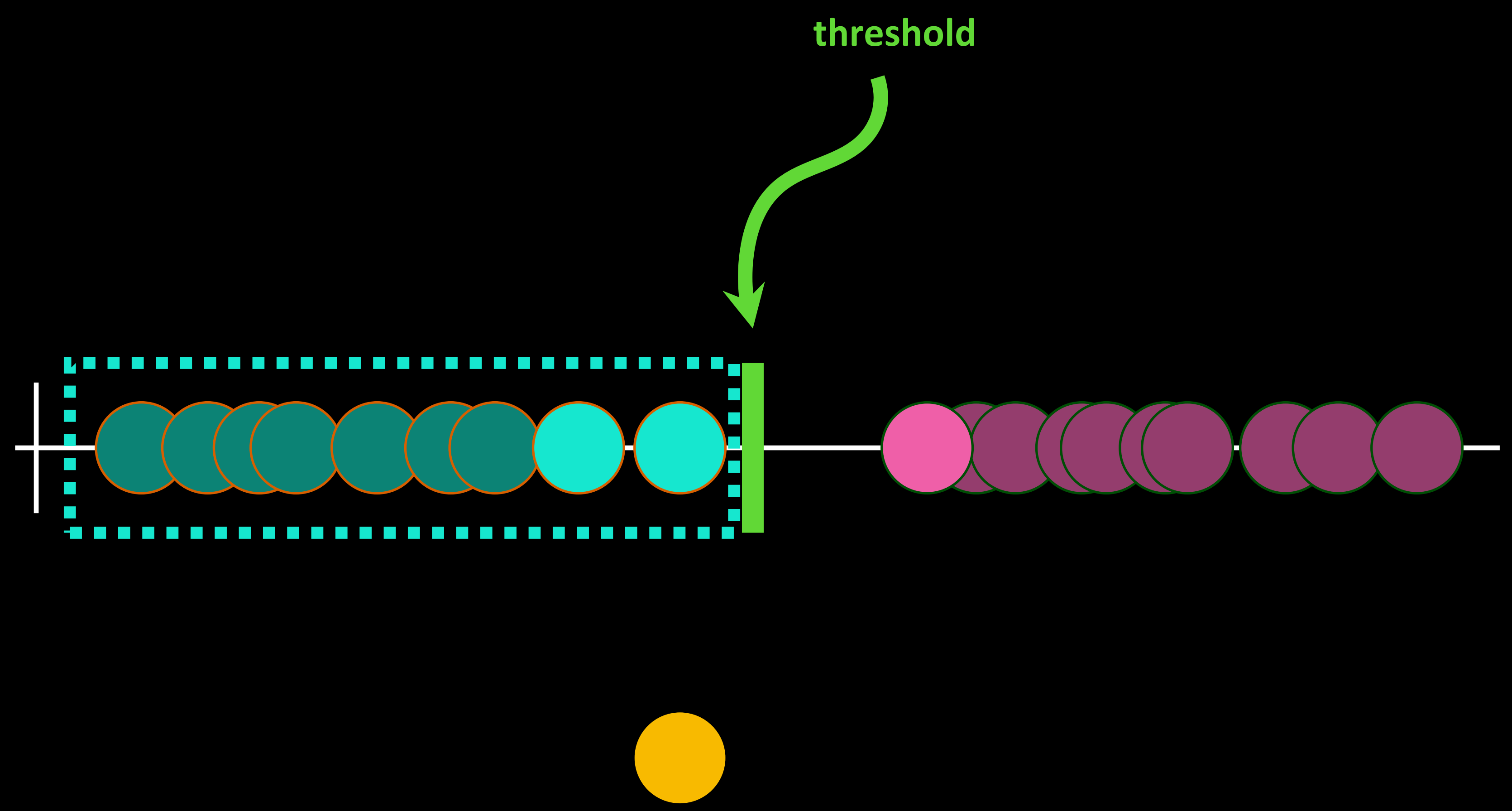
svm:Large Margin Classifier

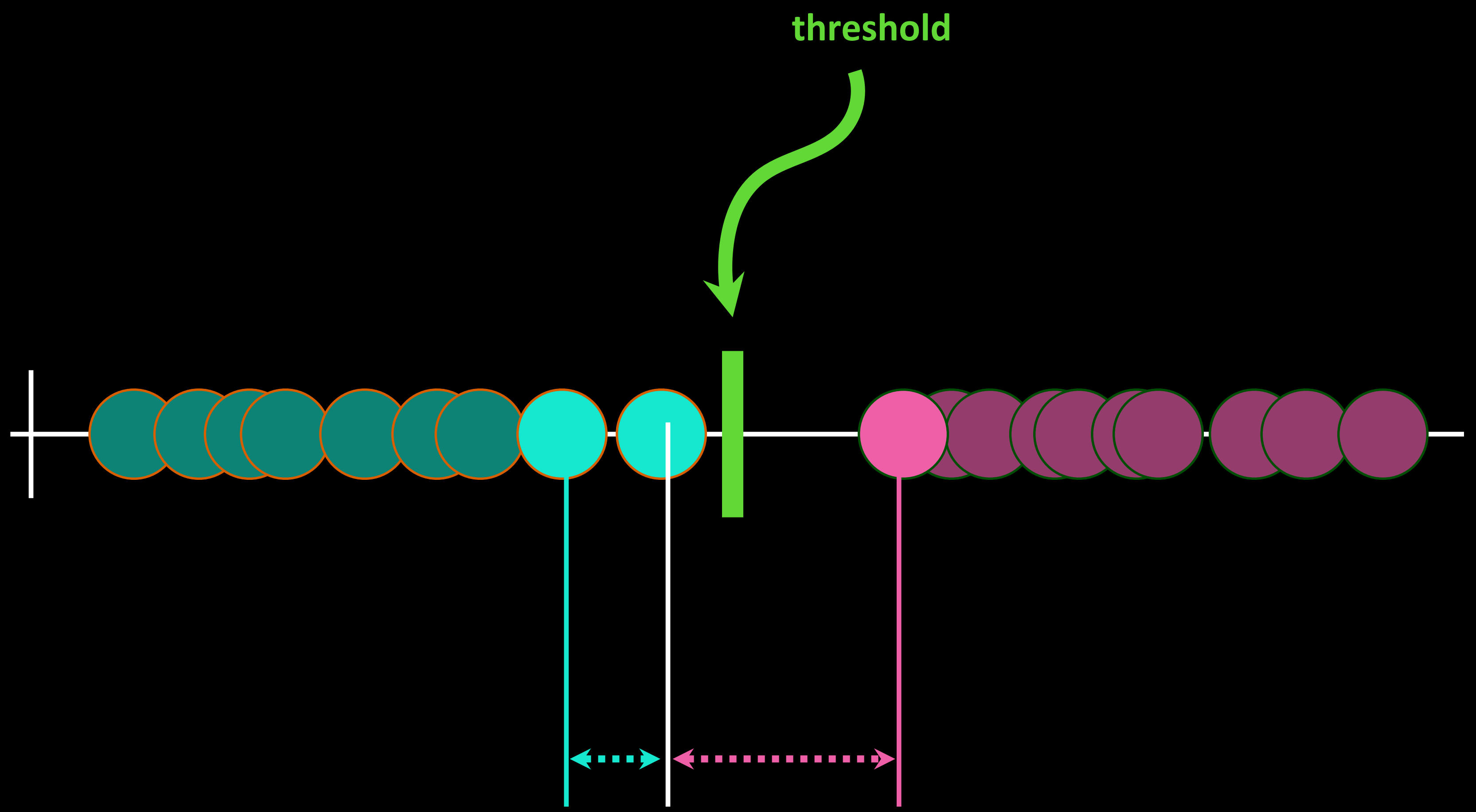

Observations class A
Observations class B
$x_i \rightarrow$


threshold
$x_i \rightarrow$

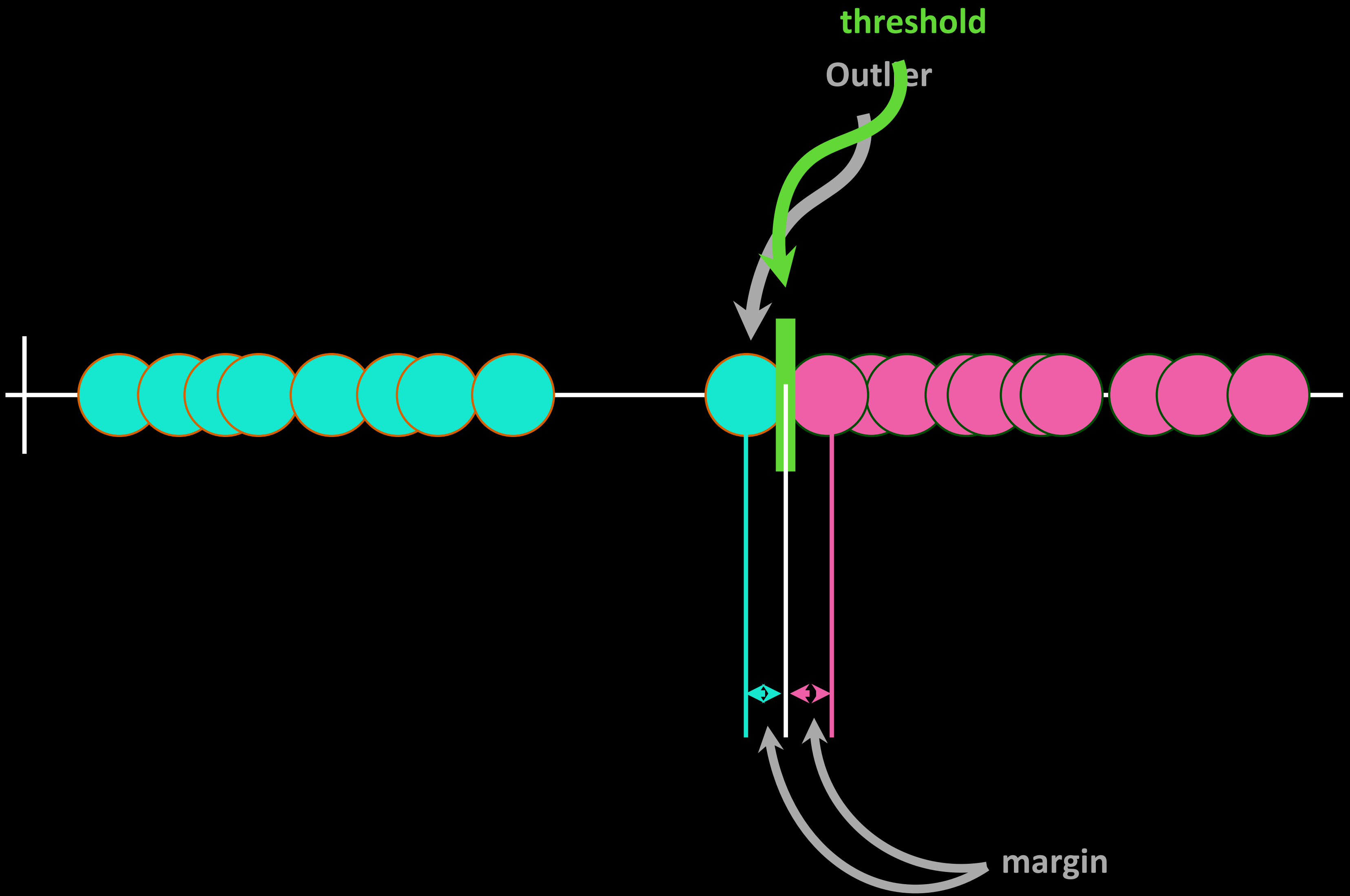

threshold
$x_i \rightarrow$

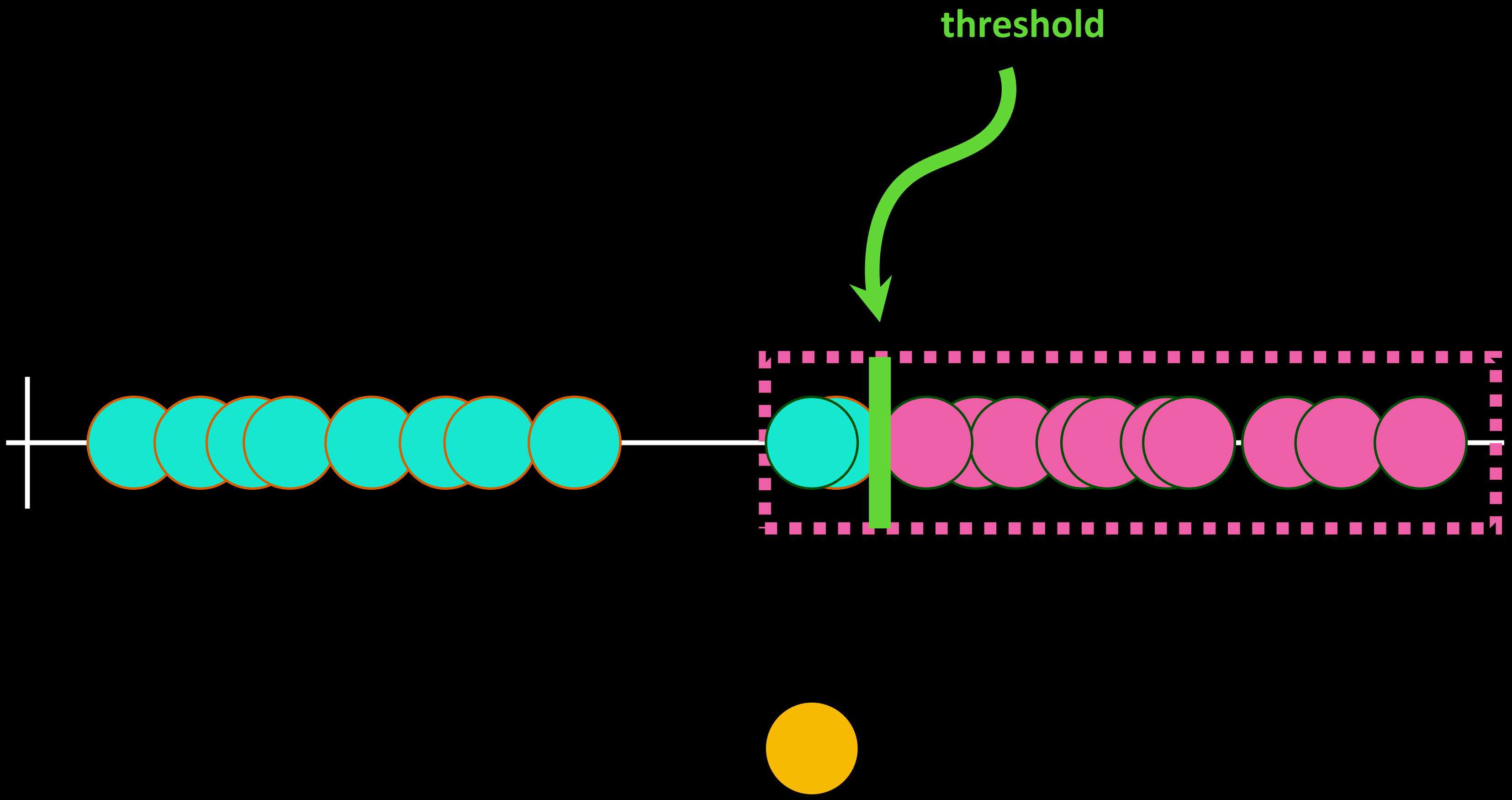

threshold
Max Margin Classifier
margin

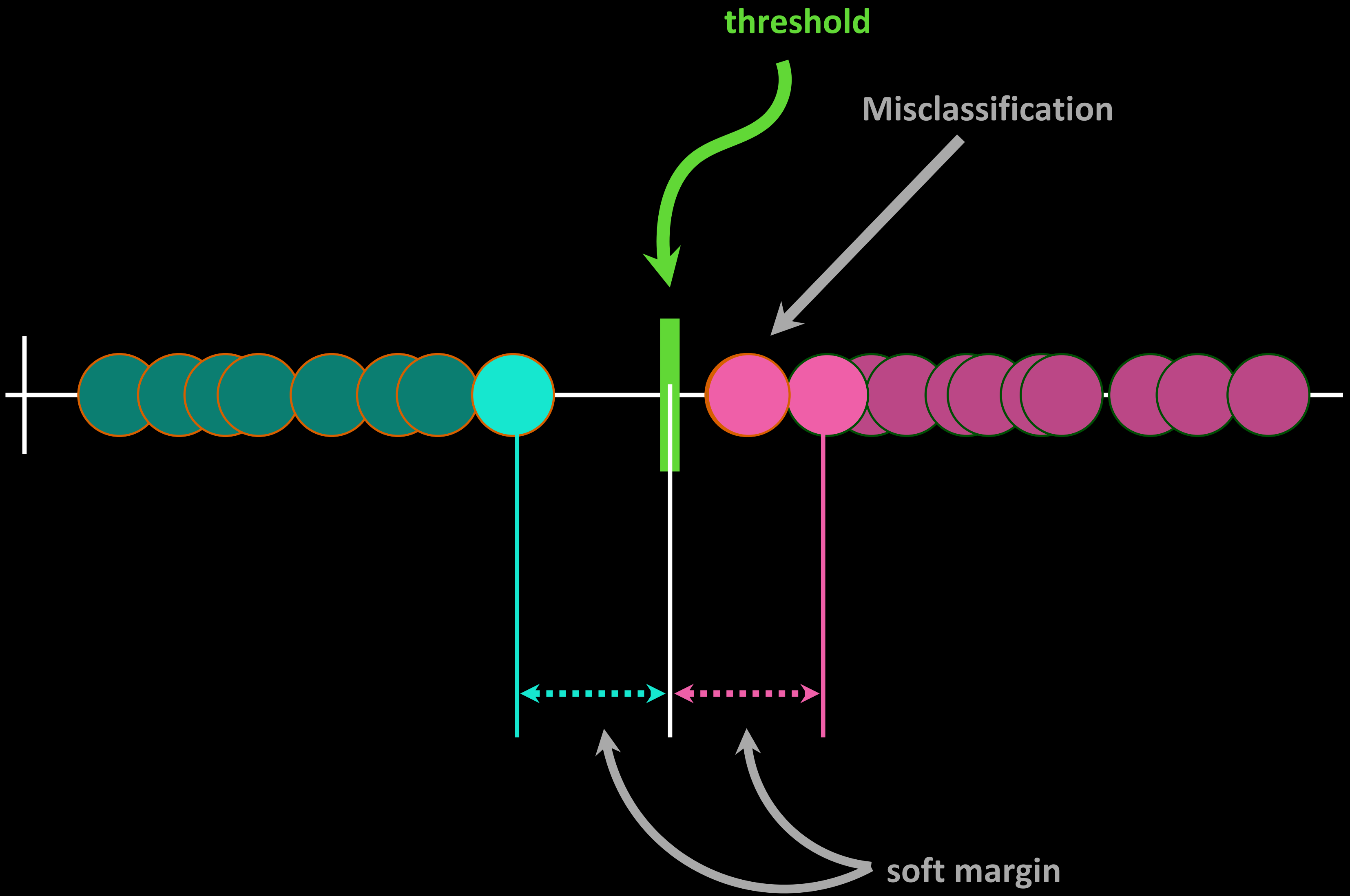

threshold


threshold

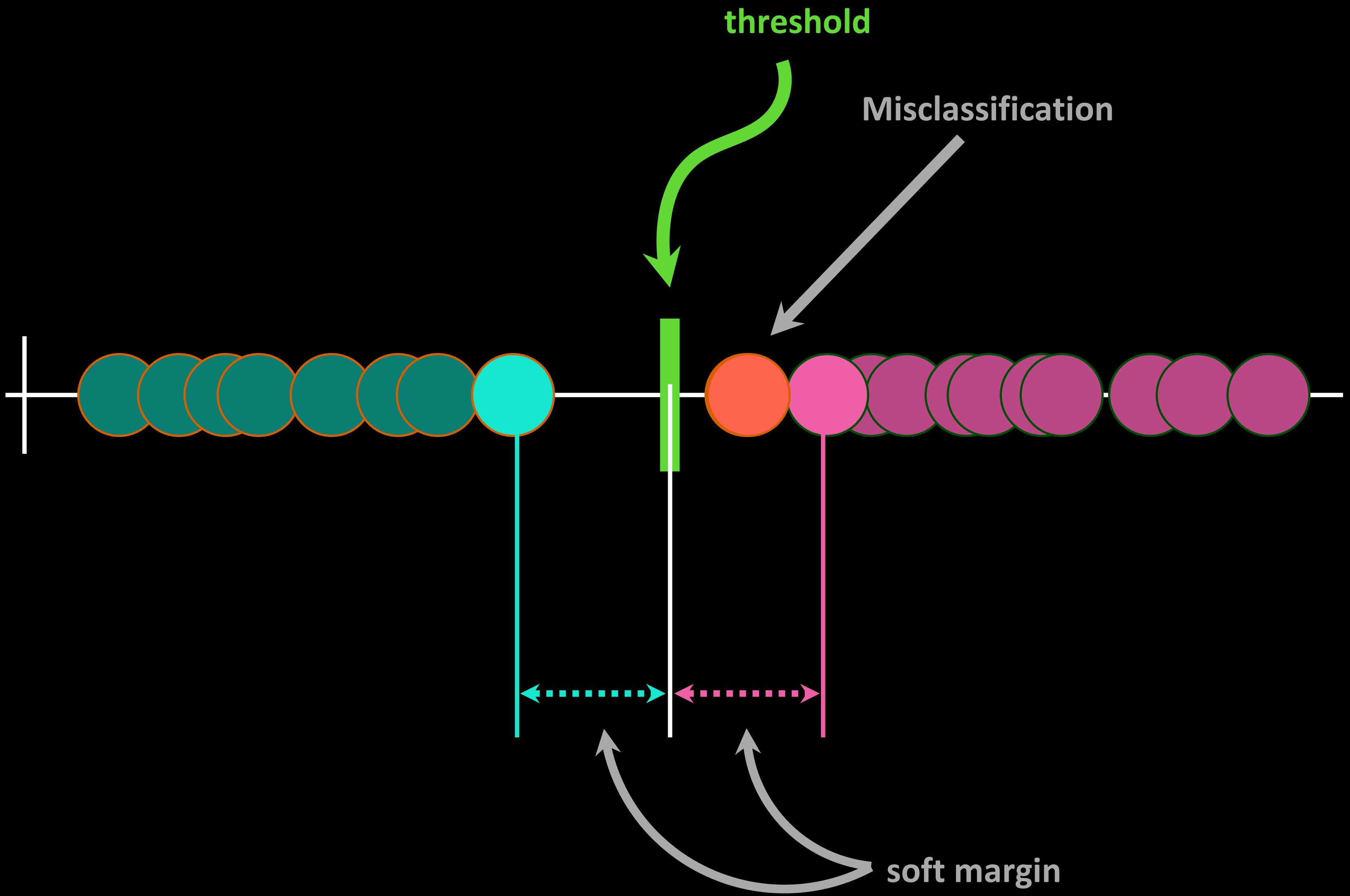
svm:Soft Margin Classifier


threshold
Outlier
margin

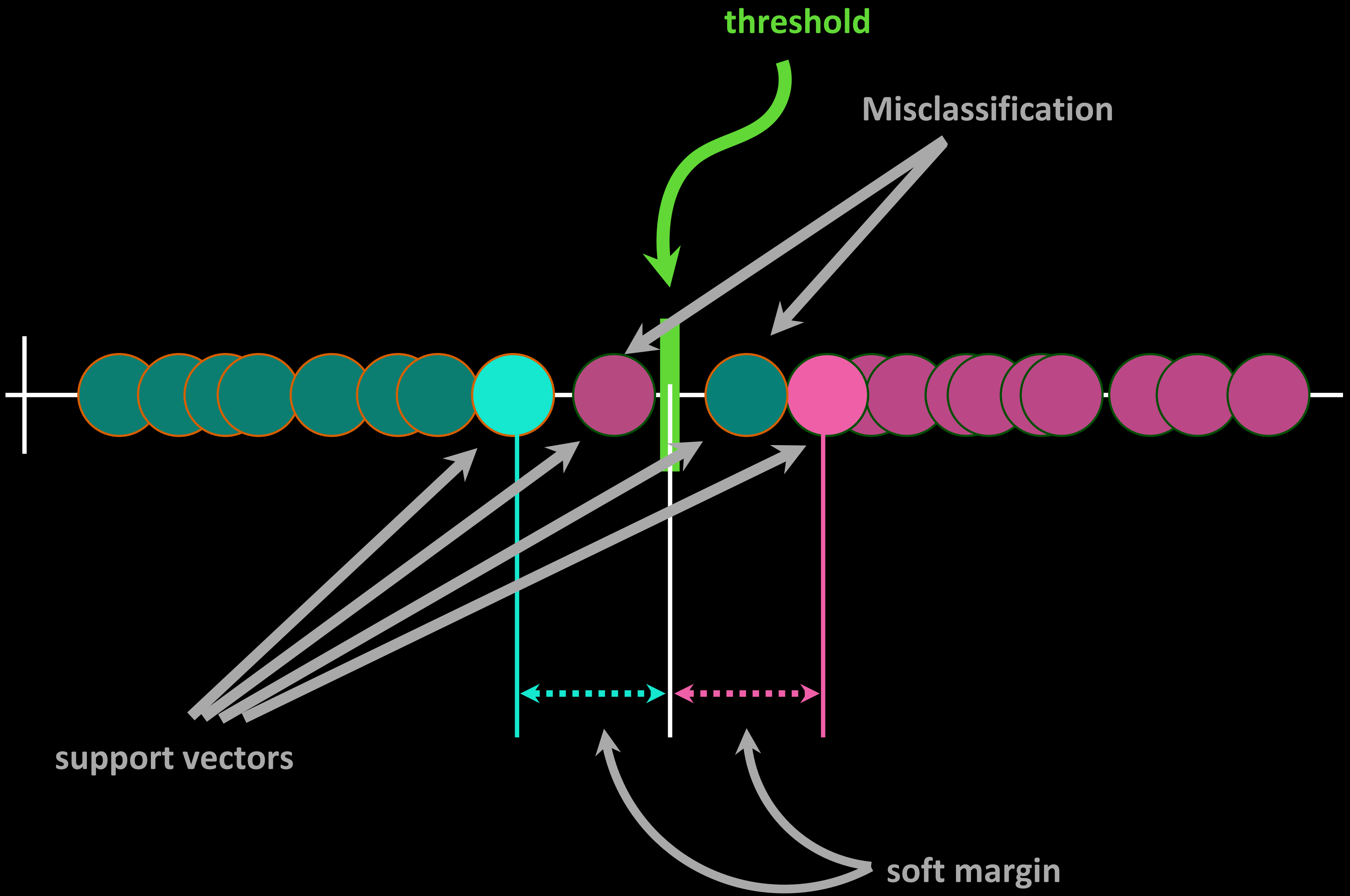

threshold

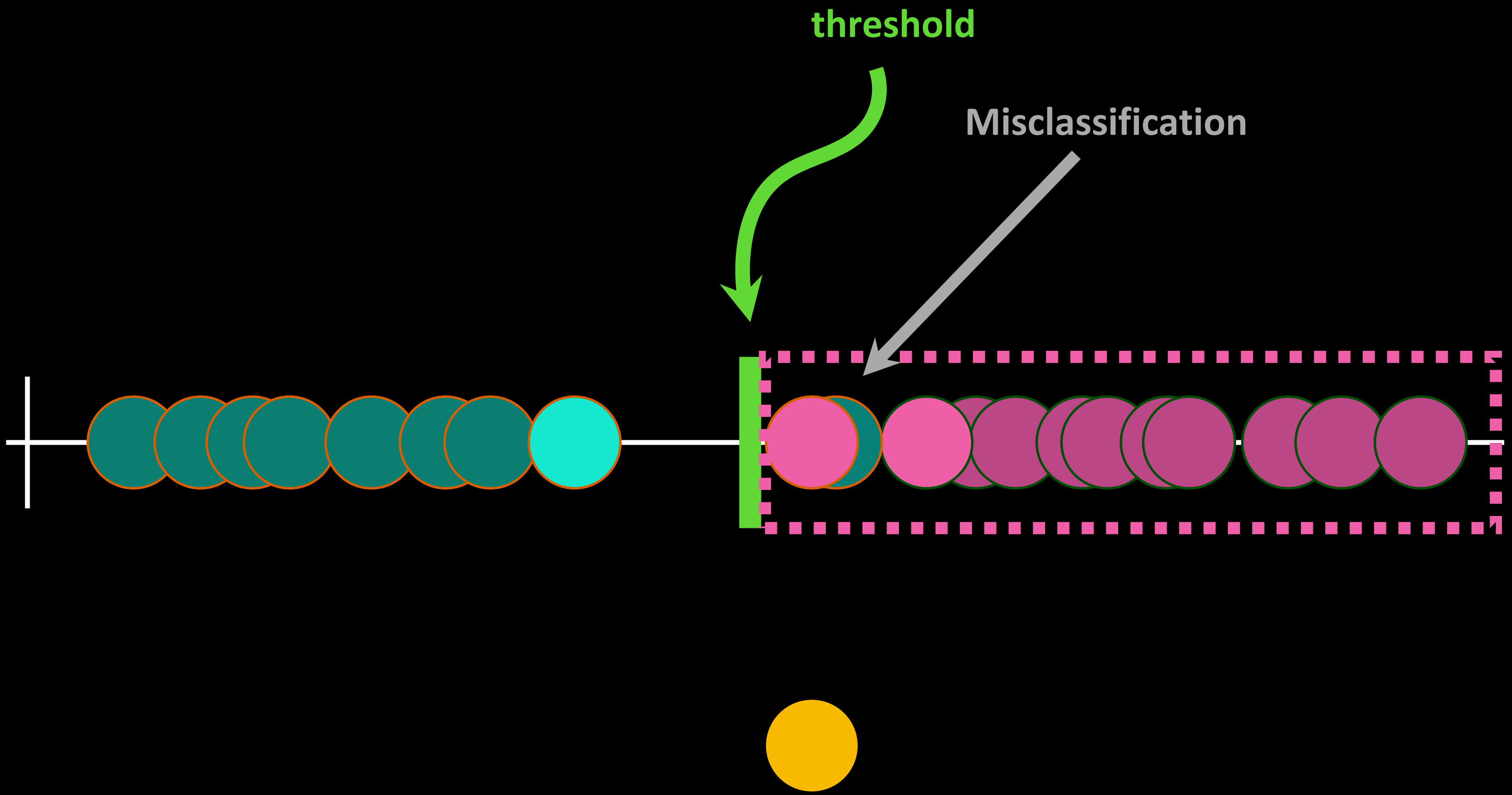

threshold
Misclassification
soft margin

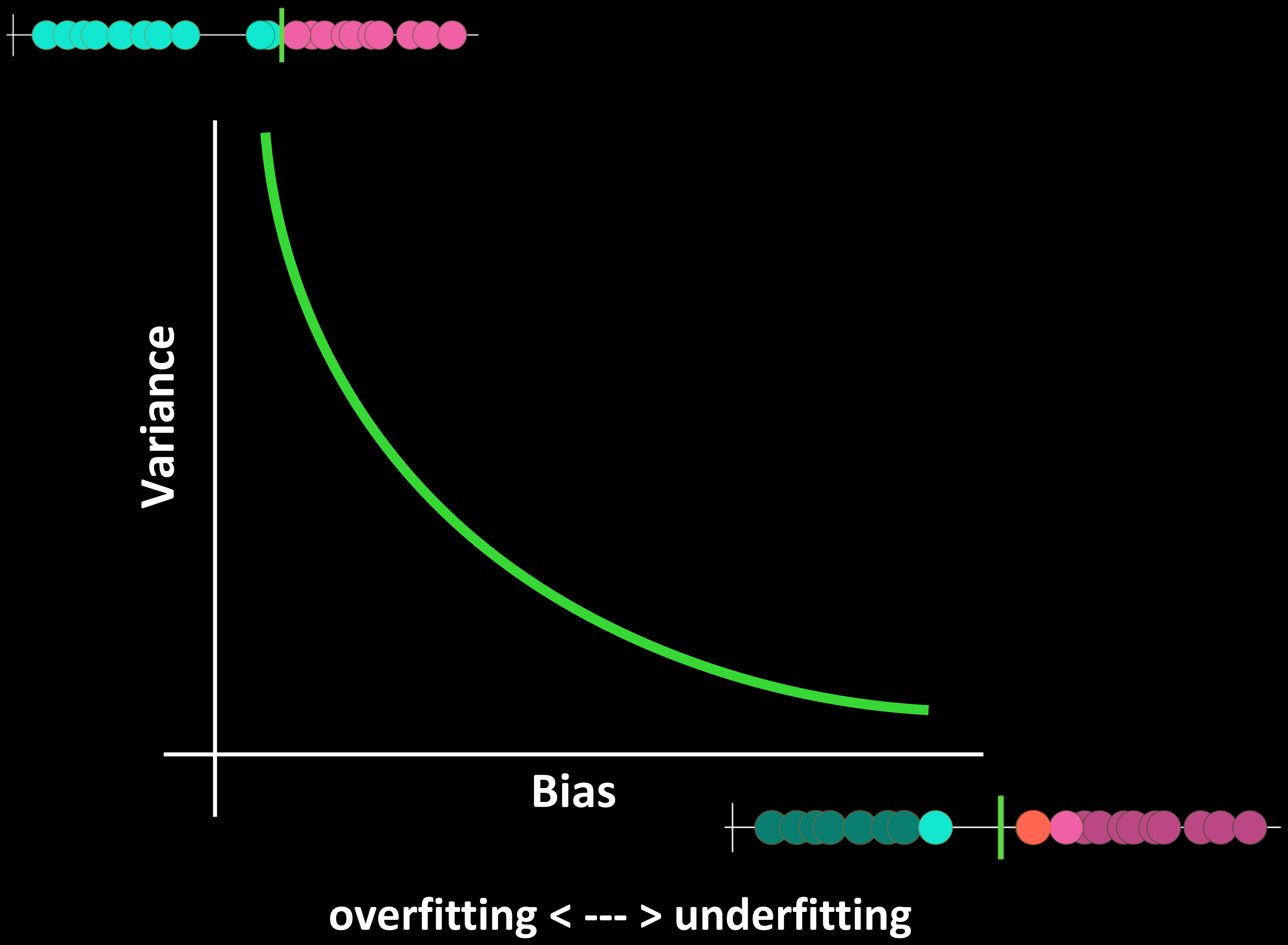
Soft Margin Classifier =
Support Vector Classifier
(SVC)



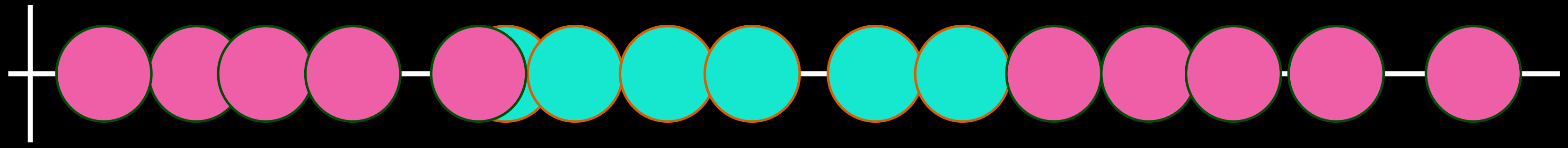
Soft Margin Classifier =
Support Vector Classifier
(SVC)



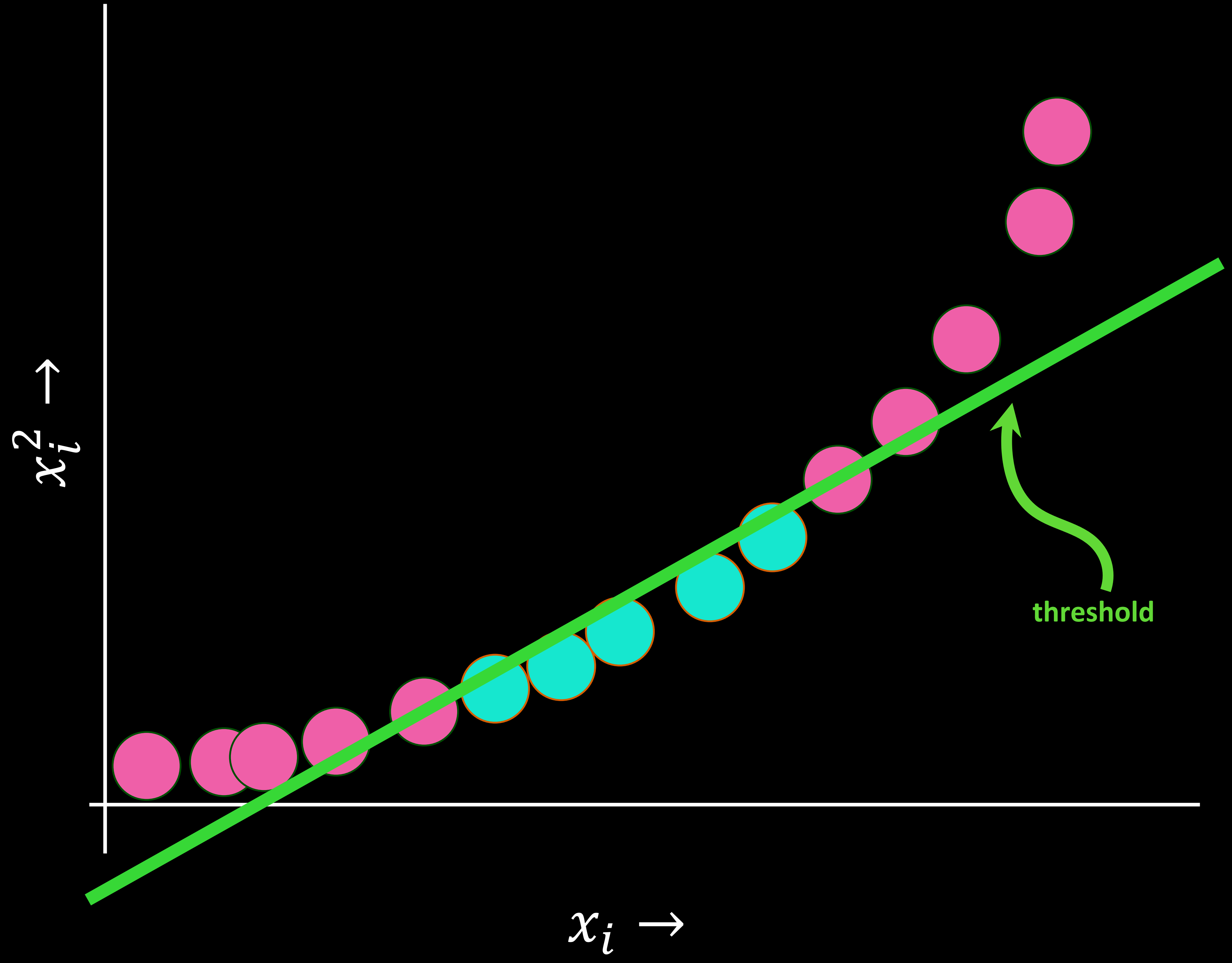

threshold
Misclassification


Variance
Bias
overfitting < --- > underfitting

svm:Kernels



$x_i \rightarrow$


$x_i^2 \rightarrow$
threshold
$x_i \rightarrow$

<u>Plan</u>

1. **Start with low dimensionality**
2. **Introduce higher dimensionality for same data**
3. **Train SVC to differentiate**

How do we know to introduce higher dimensionality?

$$x_2 = x_1^2$$

$$x_2 = \log x_1$$

$$x_2 = x_1^3$$

$$x_2 = \sqrt{1 + x_1}$$

$$x_2 = \frac{x^5}{2x_1 + 4}$$

sklearn.svm.SVC()

kernel: string, optional (default='rbf')

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (n_samples, n_samples).

degree: int, optional (default=3)

Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma{'scale', 'auto'} or float, default='scale'

Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

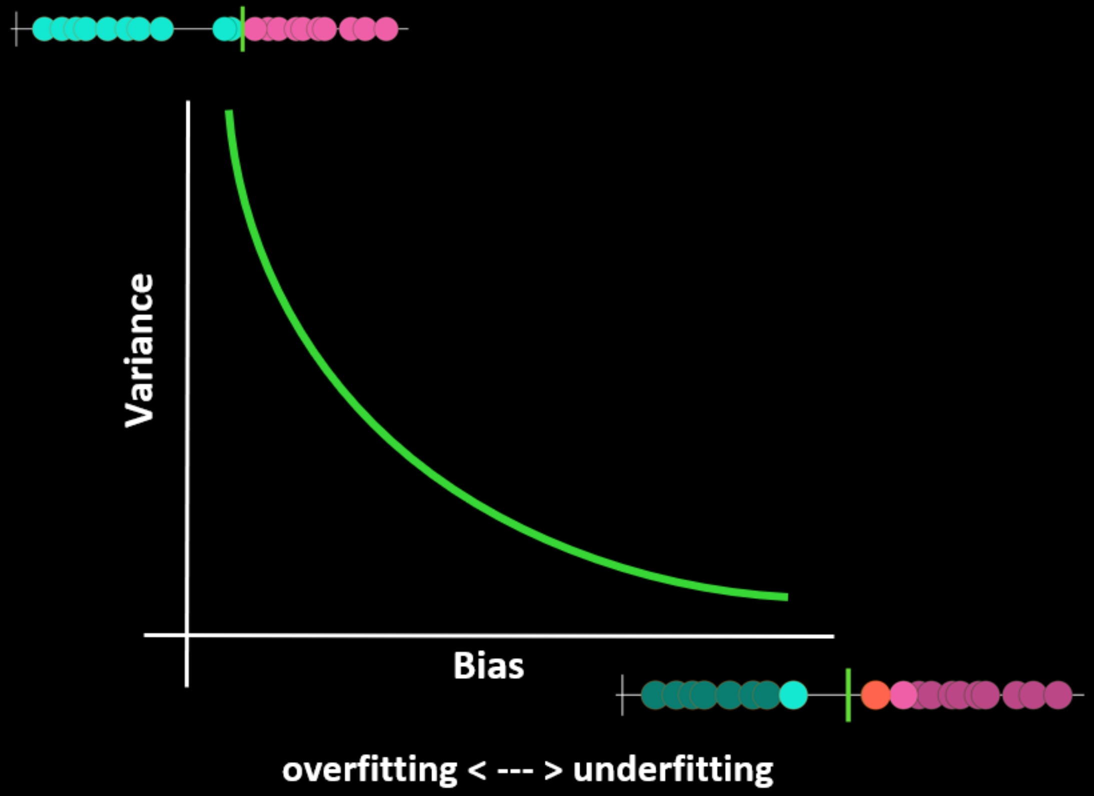
- if gamma='scale' (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features
- if float, must be non-negative.

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

sklearn.svm.SVC()

C: float, optional (default=1.0)

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

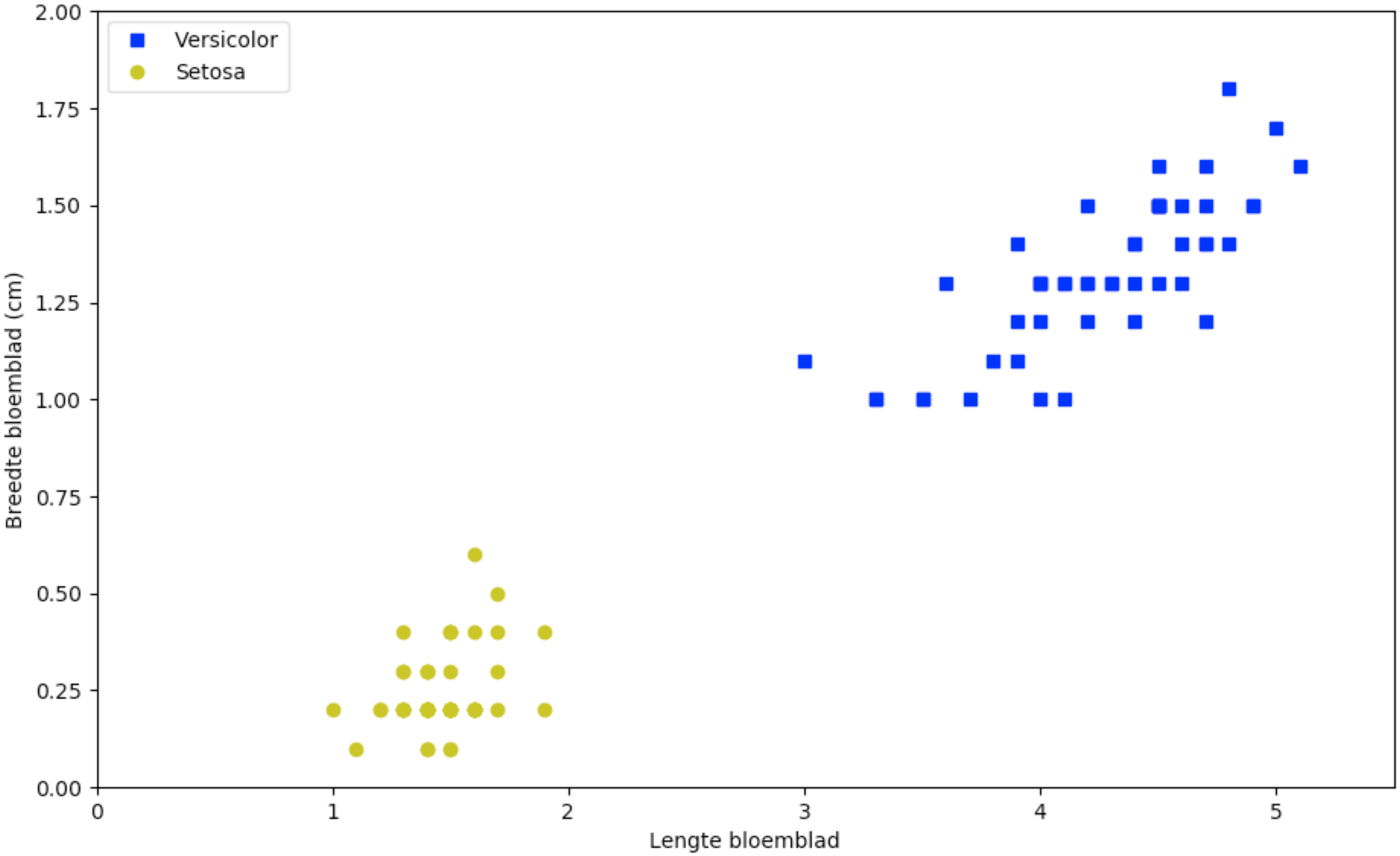


Hoe **lager C**, hoe zachter de marges => **meer bias**, richting **underfitting**

Hoe **hoger C**, hoe harder de marges => **minder bias**, richting **overfitting**

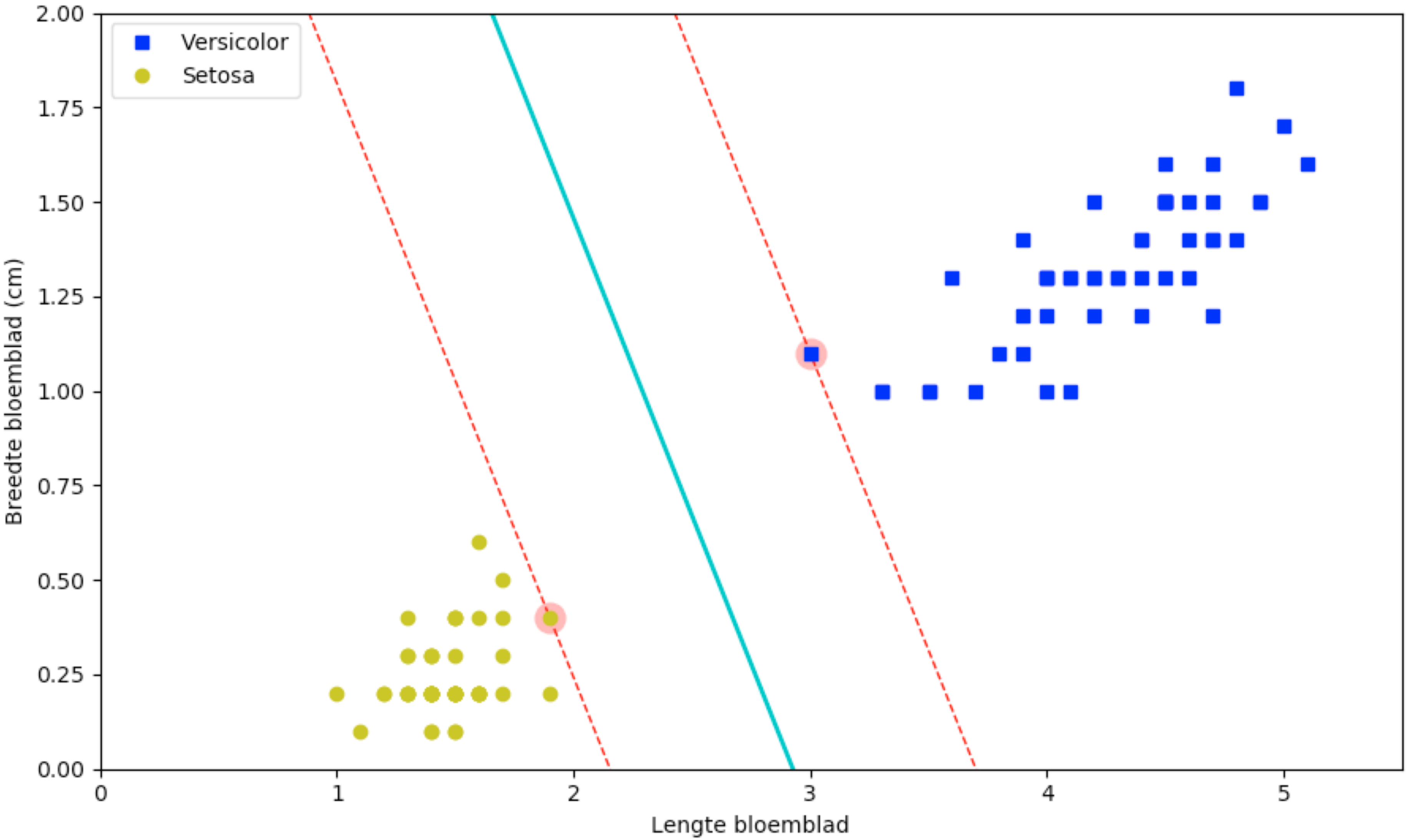
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

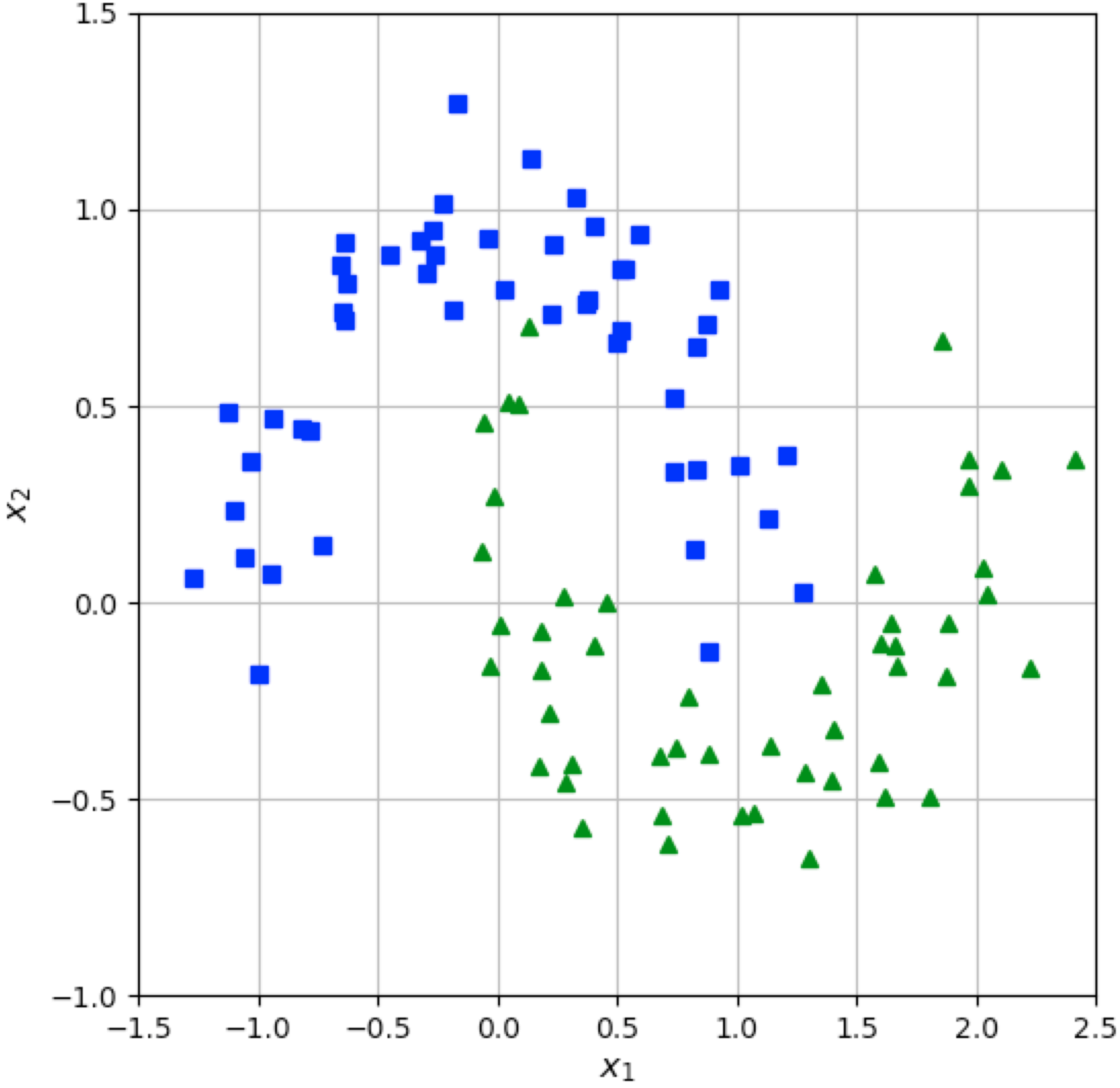
Voorbeelden

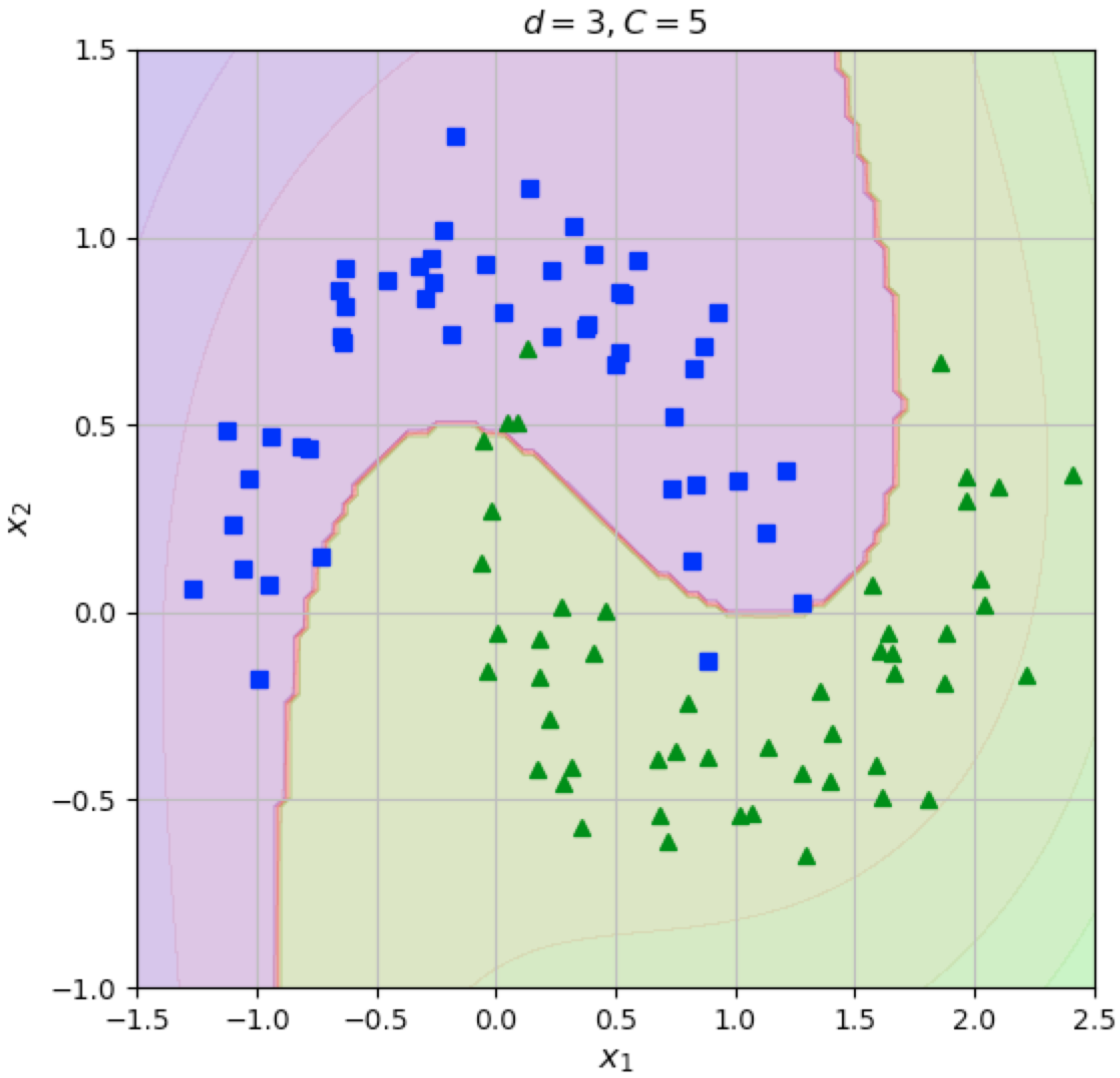


Versicolor
Setosa
Breedte bloemblad (cm)
Lengte bloemblad
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
1
2
3
4
5

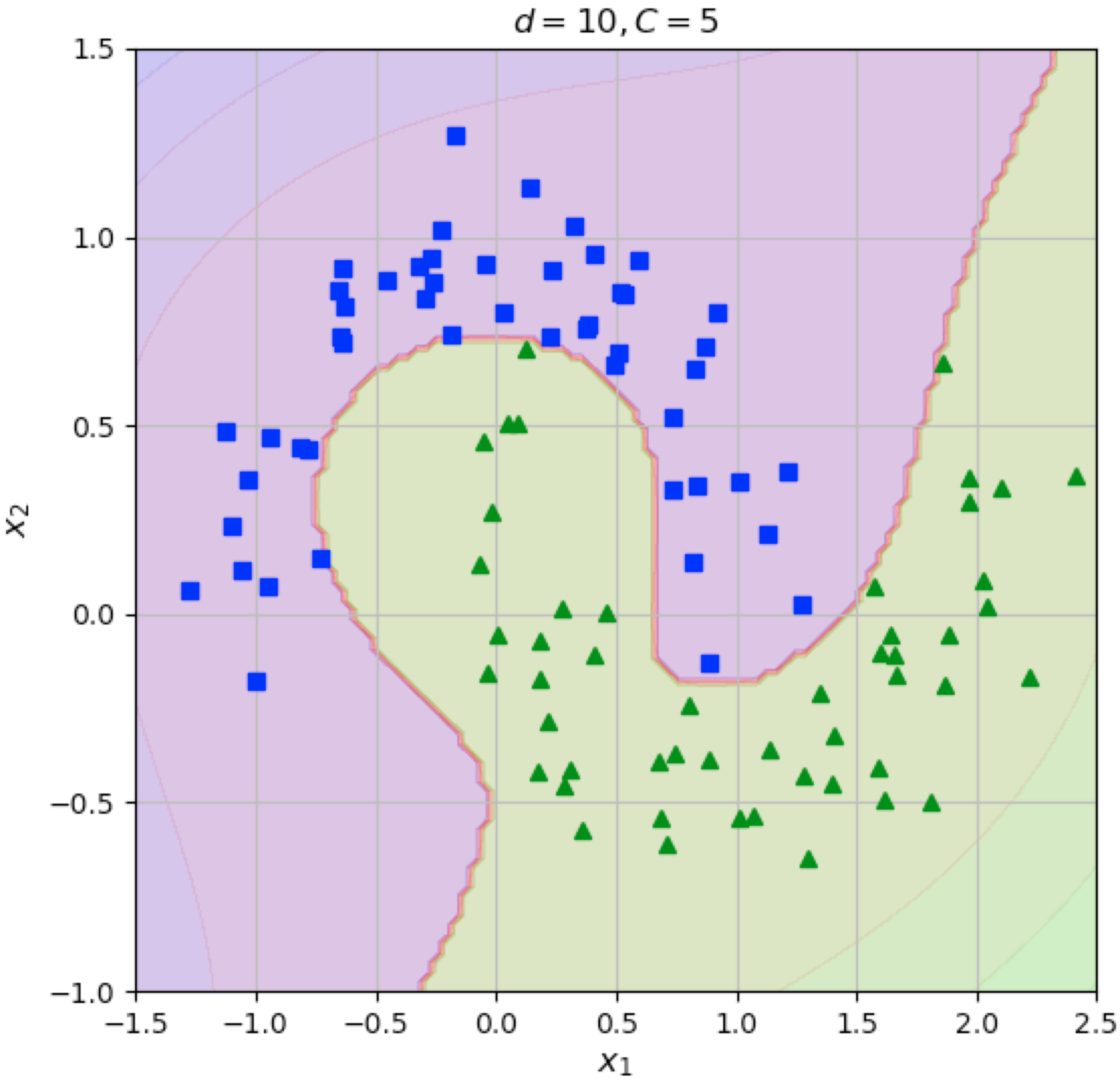
```
svm_clf = SVC(kernel="linear", C=float("inf"))
svm_clf.fit(X, y)
```




1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
x_1
x_2



$d = 3, C = 5$
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

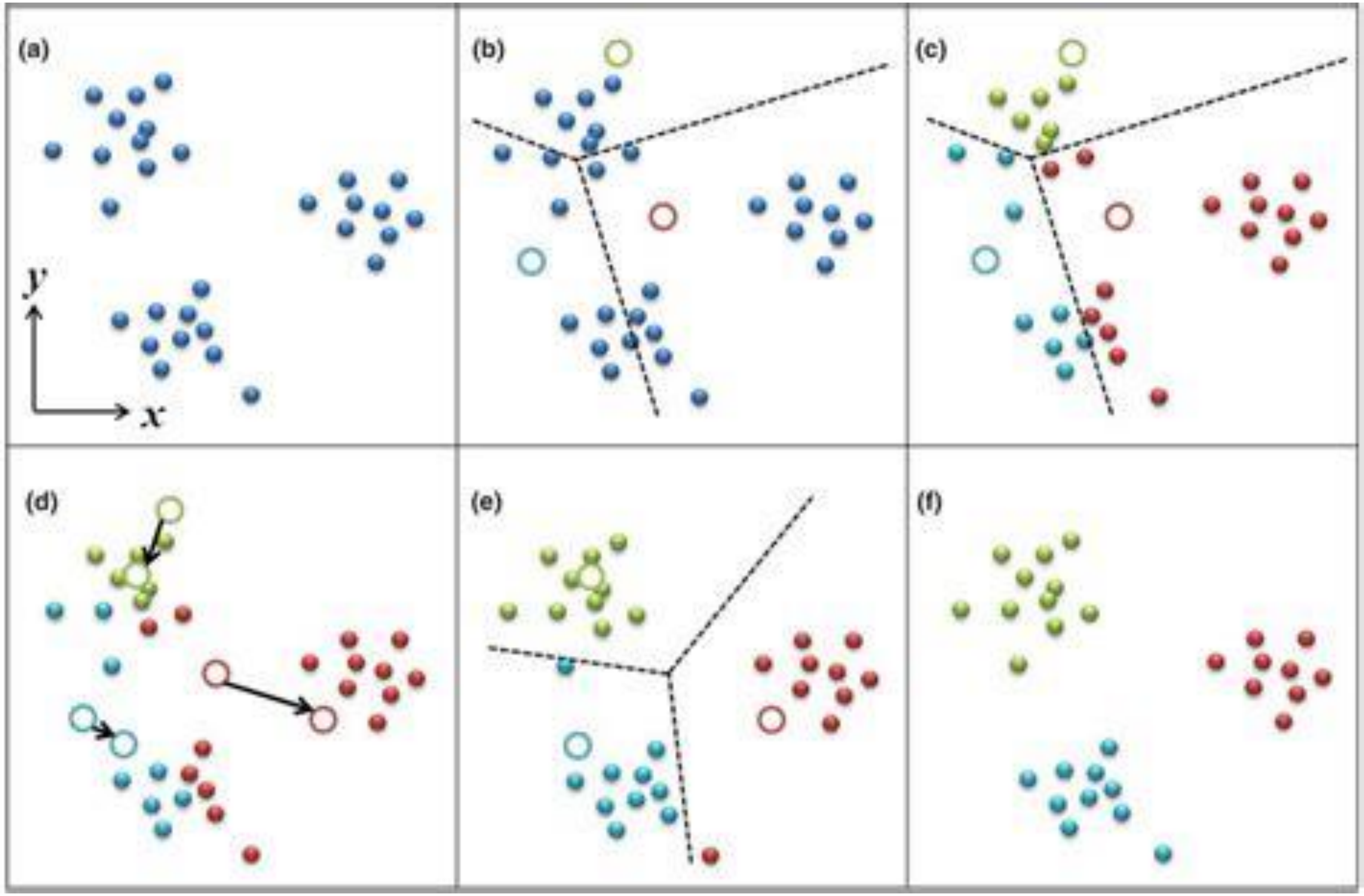
Clustering: algemeen

- Gelijksoortige observaties samen groeperen
  - Zonder te classificeren
- Voorbeeld van *unsupervised learning*

Clustering: k-means

- Kies het gewenste aantal clusters
- Geef elk cluster een *random* centroid (middelpunt)
- Daarna afwisseling van twee stappen:
  - **Assignment**stap: wijs elke observatie toe aan het cluster met de dichtstbijzijnde centroid
  - **Update**stap: herbereken de centroids op basis van de aan elk cluster toegewezen observaties

Clustering: k-means



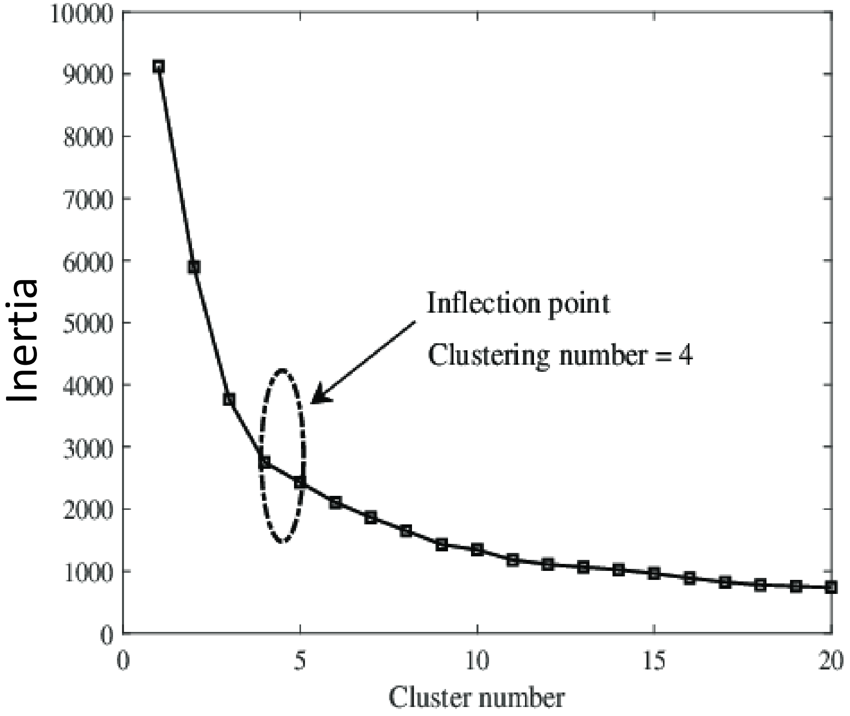
Bron: researchgate.net

Clustering: k-means

- Initialisatie van de **centroids**:
  - Random (niet zo handig)
  - Op basis van voorkennis
  - Kiezen uit meerdere *random* initialisaties
    - Metric: inertia (sum of squared distances to centroids)
  - k-means++
    - Bevorder dat de initiële centroids ver uit elkaar liggen

Clustering: k-means

- Keuze van het **aantal clusters**:
  - “Elbow method”



Bron: researchgate.net

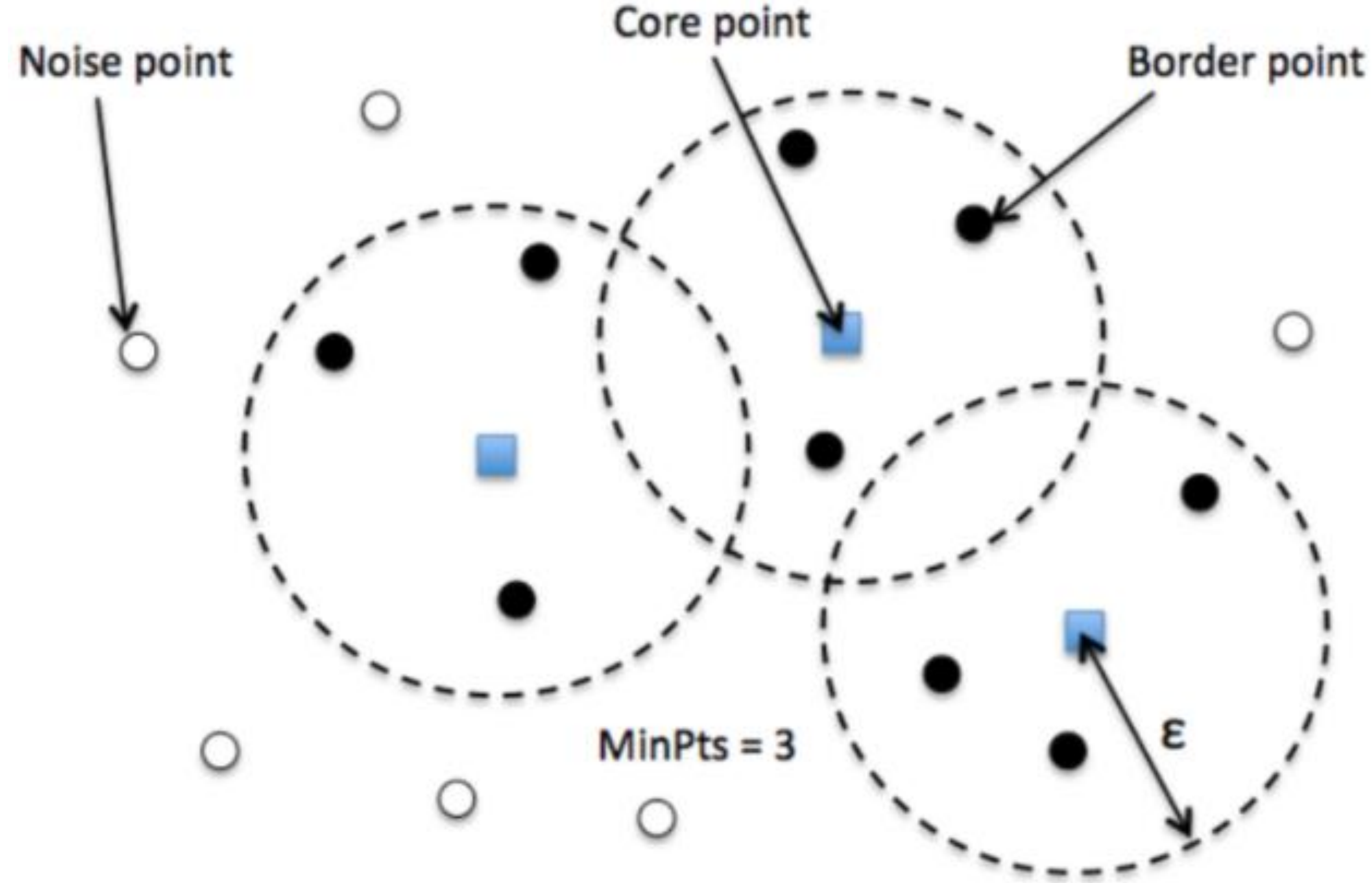
Clustering: DBSCAN

- Density-Based Spatial Clustering of Applications with Noise
- Twee parameters:
  - ε (eps)
  - minPts

Clustering: DBSCAN

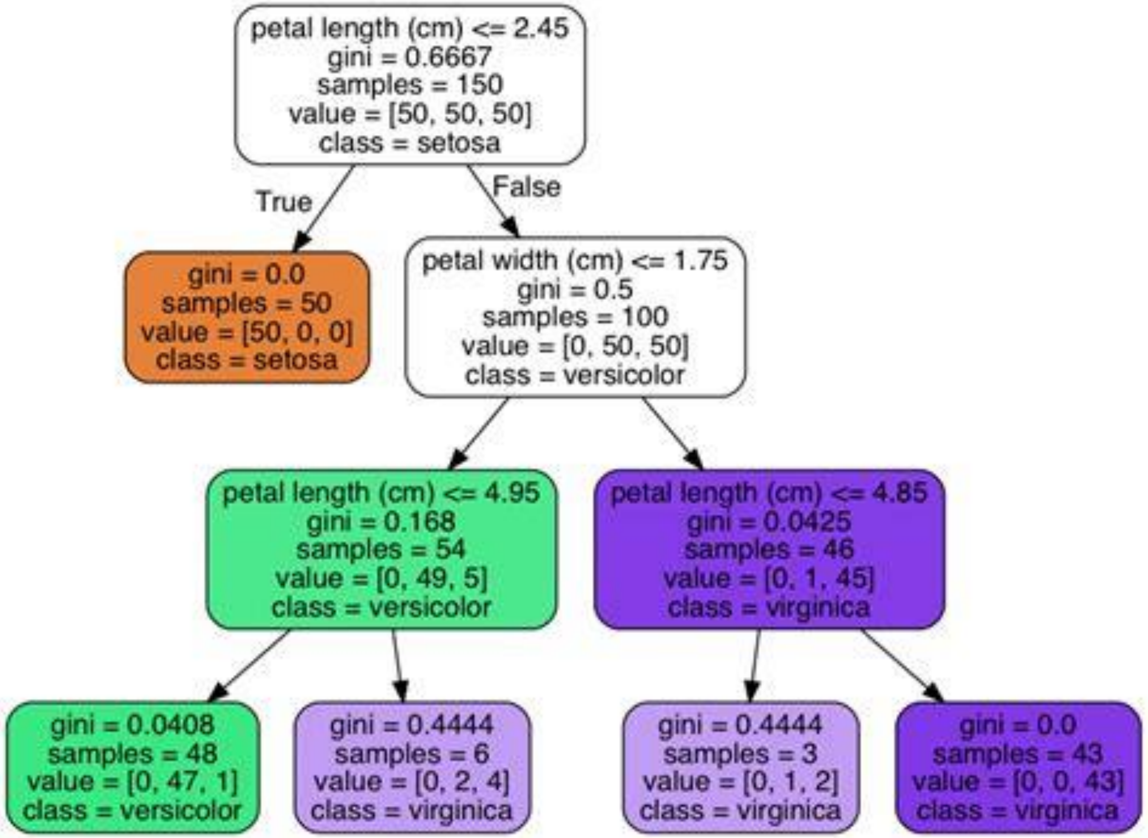
- Het algoritme werkt als volgt:
  - Zoek de observaties in de straal ε van elk punt. Identificeer als **core points** de observaties met meer dan minPts buren.
  - Zoek de **connected components van de core points**, oftewel de core points die binnen straal ε van elkaar liggen. Deze vormen samen een cluster.
  - Wijs elk niet-core point toe aan een naburig cluster als dat cluster binnen straal ε ligt, zo niet dan is het **ruis / een uitbijter (outlier)**.

Clustering: DBSCAN

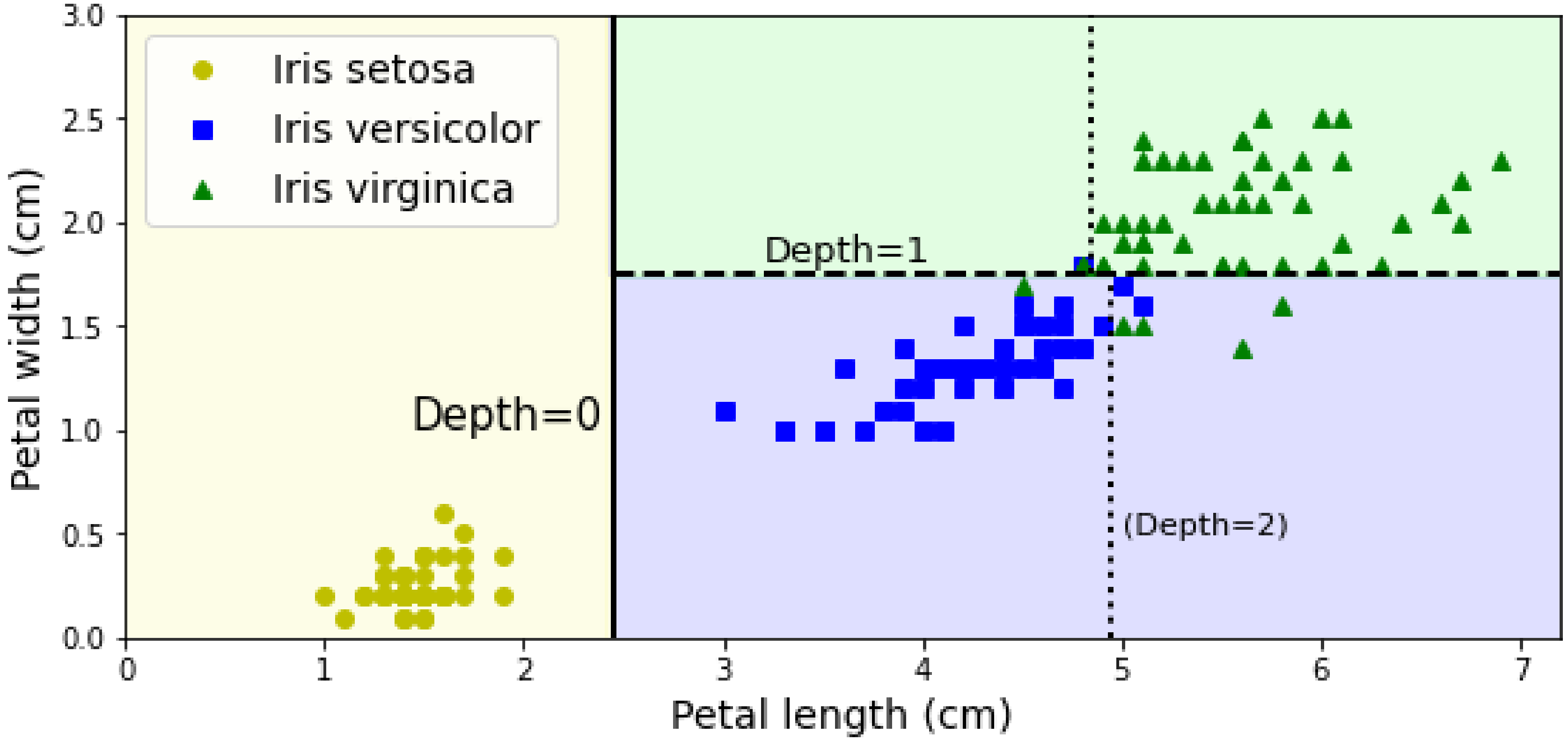


Bron: miro.medium.com

Decision trees

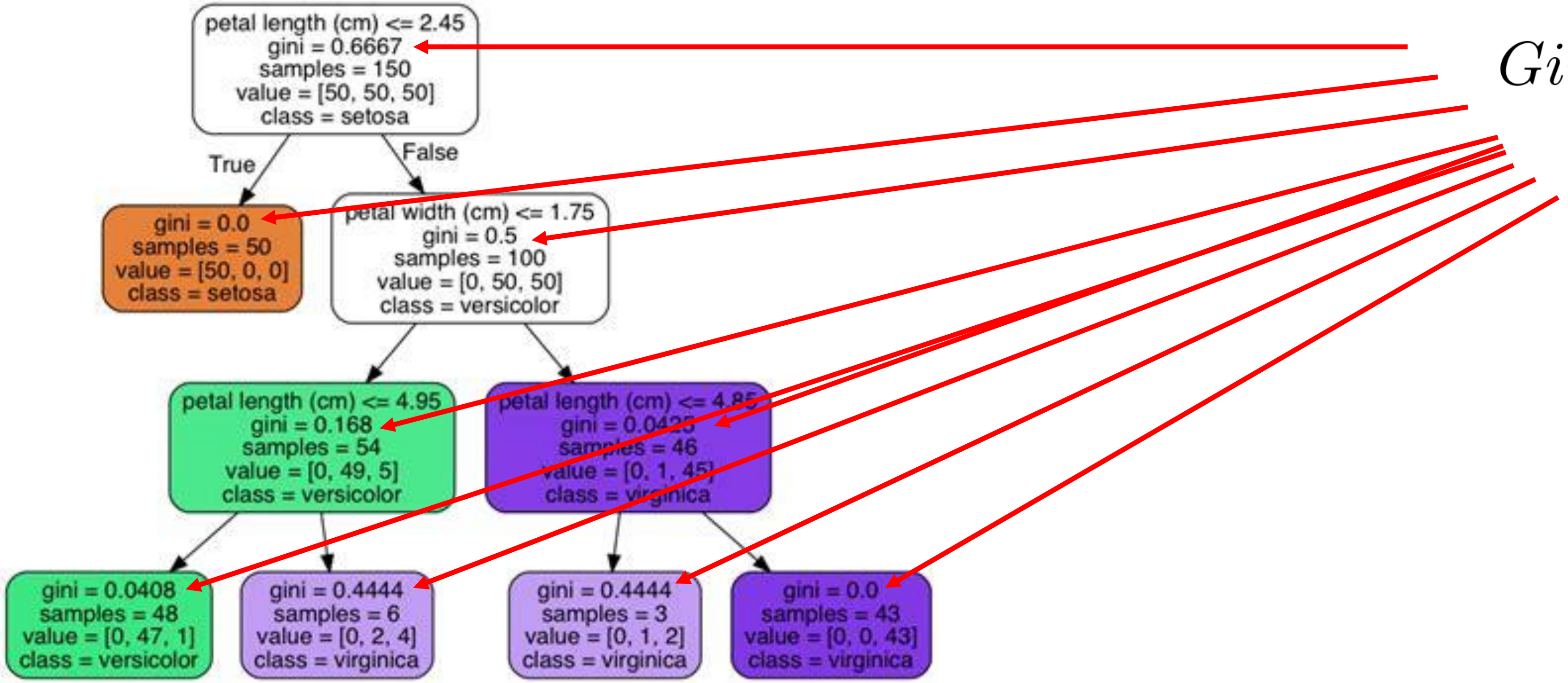


Decision trees: decision boundaries

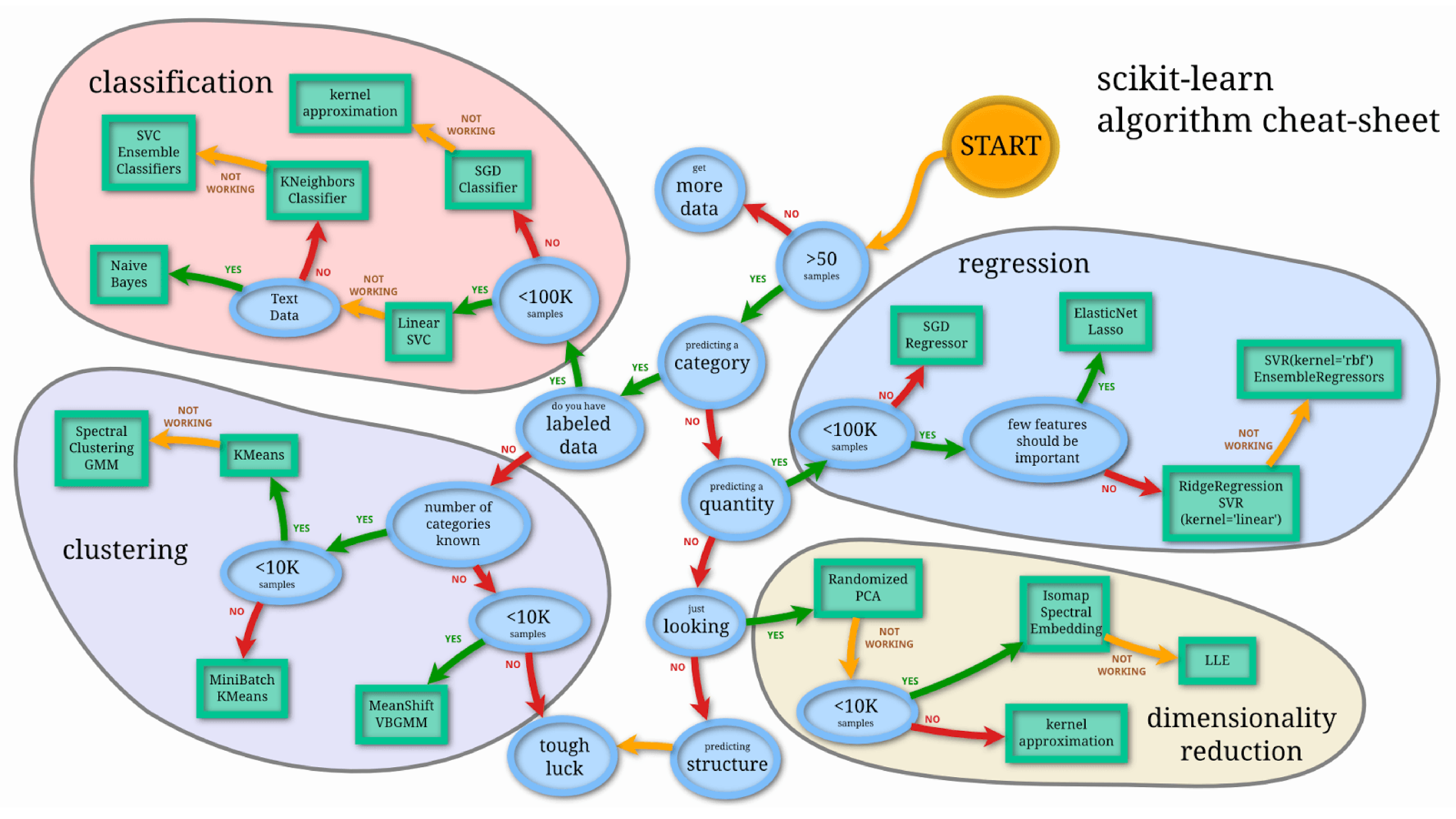


Bron: https://github.com/ageron/handson-ml3/blob/main/06_decision_trees.ipynb

Decision trees: gini-impurity



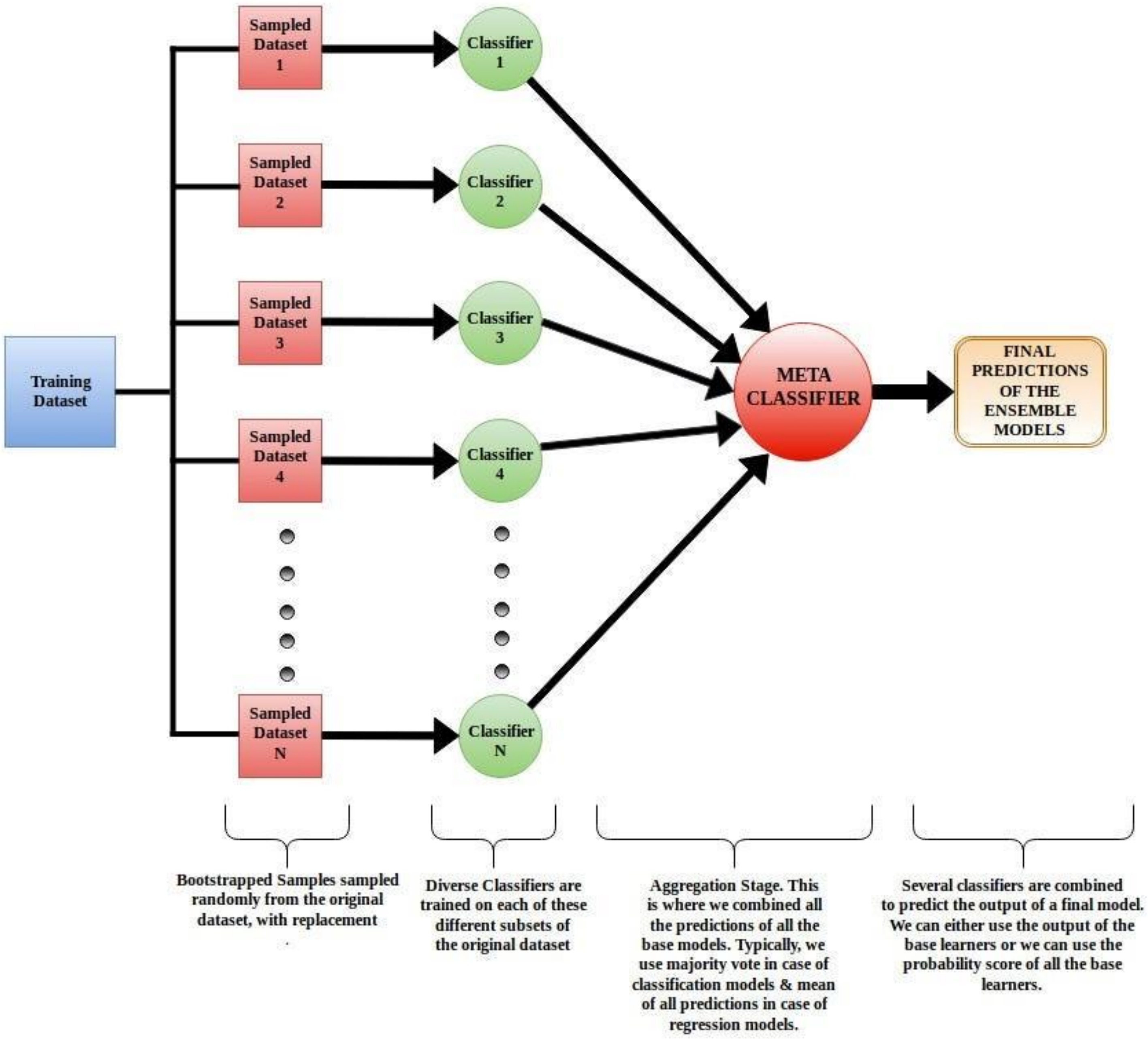
$$Gini(t) = 1 - \sum_{i=1}^{j} P(i|t)^2$$


scikit-learn
algorithm cheat-sheet
START
>50 samples
get more data
predicting a category
do you have labeled data
predicting a quantity
just looking
predicting structure
tough luck
classification
<100K samples
SGD Classifier
kernel approximation
Linear SVC
Text Data
Naive Bayes
KNeighbors Classifier
SVC Ensemble Classifiers
clustering
number of categories known
<10K samples
KMeans
Spectral Clustering GMM
MiniBatch KMeans
MeanShift VBGMM
regression
few features should be important
SGD Regressor
ElasticNet Lasso
RidgeRegression SVR (kernel='linear')
SVR(kernel='rbf') EnsembleRegressors
dimensionality reduction
Randomized PCA
Isomap Spectral Embedding
LLE
YES
NO
NOT WORKING

Live Notebooks

- Decision trees
- k-means
- DBSCAN

Ensemble Learning



THE DIFFERENT STAGES OF A BAGGING ALGORITHM

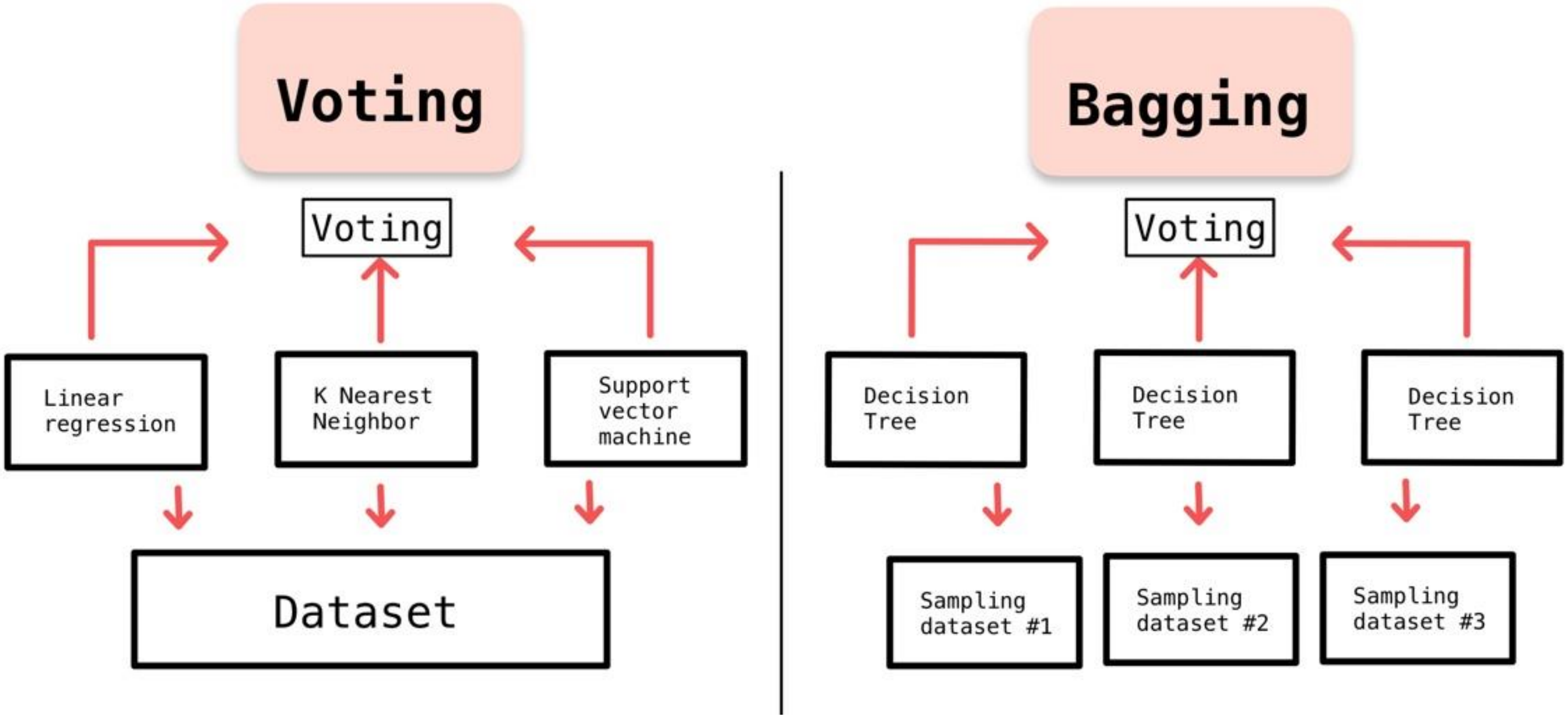
Voting, bagging en pasting (1)

- Voting: verschillende soorten classifiers
  - Onafhankelijk en ongecorreleerd (voorzover mogelijk)
  - Elk met een eigen uitkomst
- Hard voting: de *vaakst* voorspelde uitkomst wint
- Soft voting: de uitkomst met de *gemiddeld* hoogste waarschijnlijkheid wint

Voting, bagging en pasting (2)

- Bagging = **B**ootstrap **Agg**regat**ing**
- Bagging en Pasting: meerdere keren dezelfde classifier
  - Elk getraind op een andere subset van de data
  - Met teruglegging: Bagging
  - Zonder teruglegging: Pasting
- Output: meest voorkomende classificatie (hard voting)

Voting, bagging en pasting (3)



Bron: medium.com

Voting in sklearn

sklearn.ensemble.VotingClassifier

class sklearn.ensemble.**VotingClassifier**(*estimators*, *, *voting='hard'*, *weights=None*, *n_jobs=None*, *flatten_transform=True*, *verbose=False*) ¶ [source]

Lijst van tuples: (naam, classifier)
Bijvoorbeeld:
`('lr', LogisticRegression(random_state=42))`

hard of soft

Bagging in sklearn

sklearn.ensemble.BaggingClassifier

class sklearn.ensemble.**BaggingClassifier**(*estimator=None, n_estimators=10, *, max_samples=1.0, max_features=1.0, bootstrap=True, bootstrap_features=False, oob_score=False, warm_start=False, n_jobs=None, random_state=None, verbose=0, base_estimator='deprecated'*) ¶ [source]

False => pasting

Bijvoorbeeld:

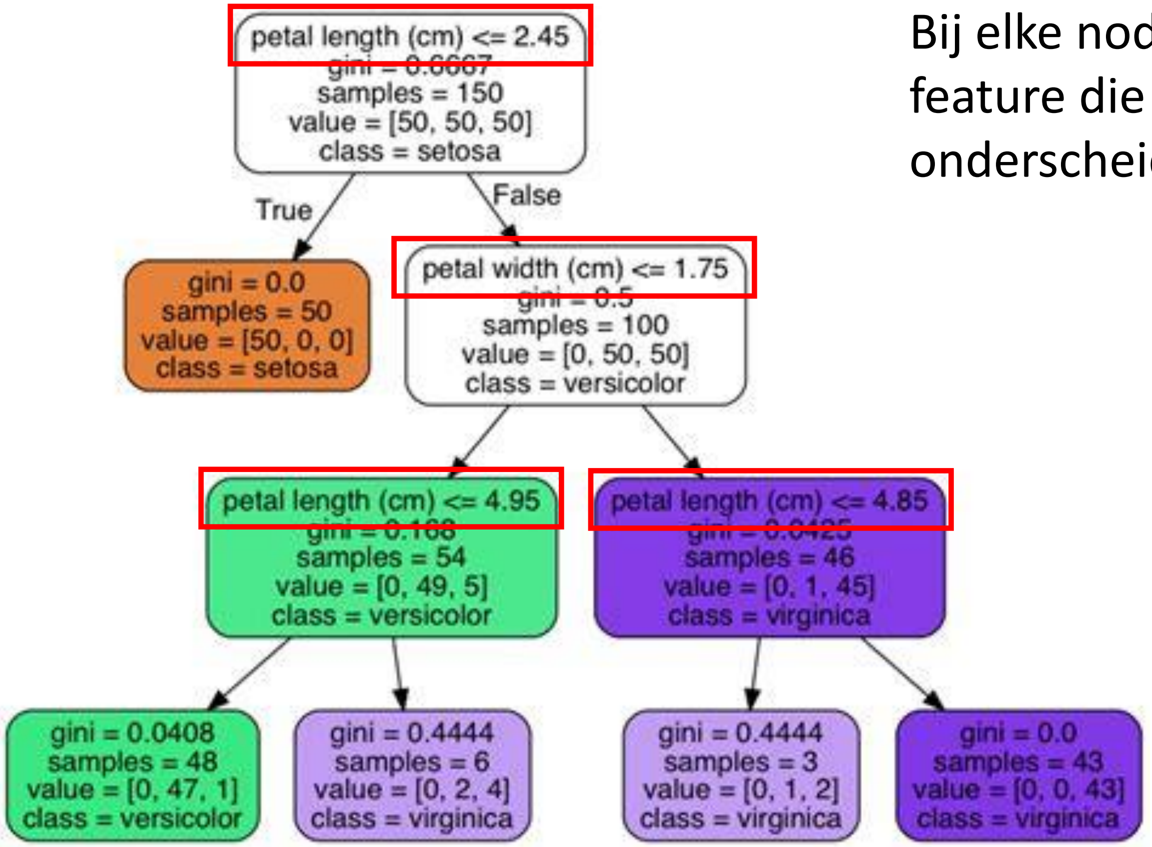
```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

bag_clf = BaggingClassifier(DecisionTreeClassifier(),
        n_estimators=500, max_samples=100, n_jobs=-1, random_state=42)
bag_clf.fit(X_train, y_train)
```

Intermezzo

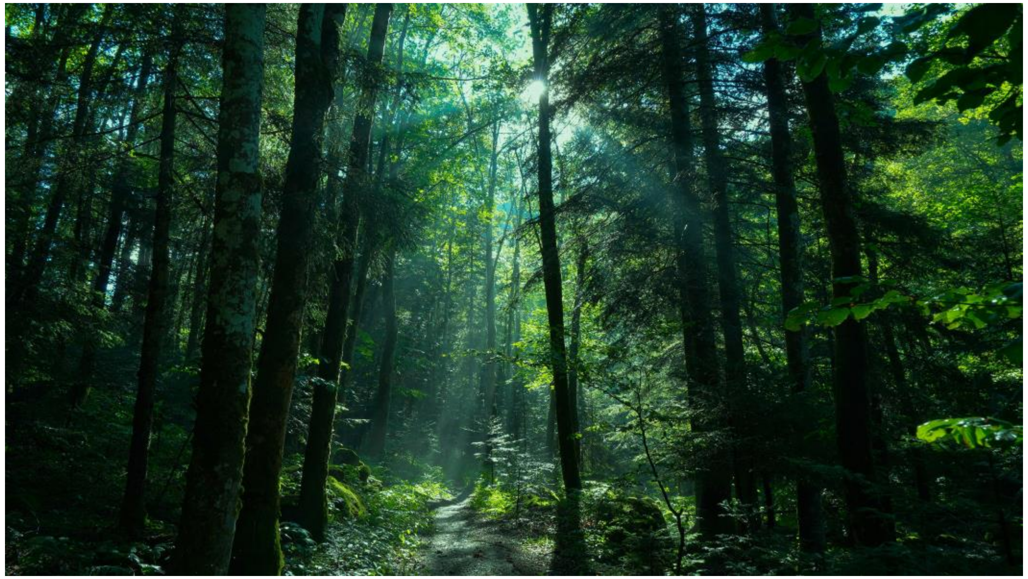
- Live Notebook over Voting classifiers

Reminder: Decision Trees



Bij elke node kiezen we de feature die het meeste onderscheid kan maken

Random Forest



- Ensemble van Decision Trees
- Gebruikt bagging als aggregator
- Wat is er random aan?
  - Bij het splitsen op een node wordt er niet uit *alle* features gekozen…
  - …maar uit een random subset
    - By default worden er $\sqrt{n}$ features van de n overwogen
  - Gevolg: meer diversiteit in de bomen
  - Minder variantie, meer bias
  - Voorkomt overfitting van het model als geheel

RandomForest in sklearn

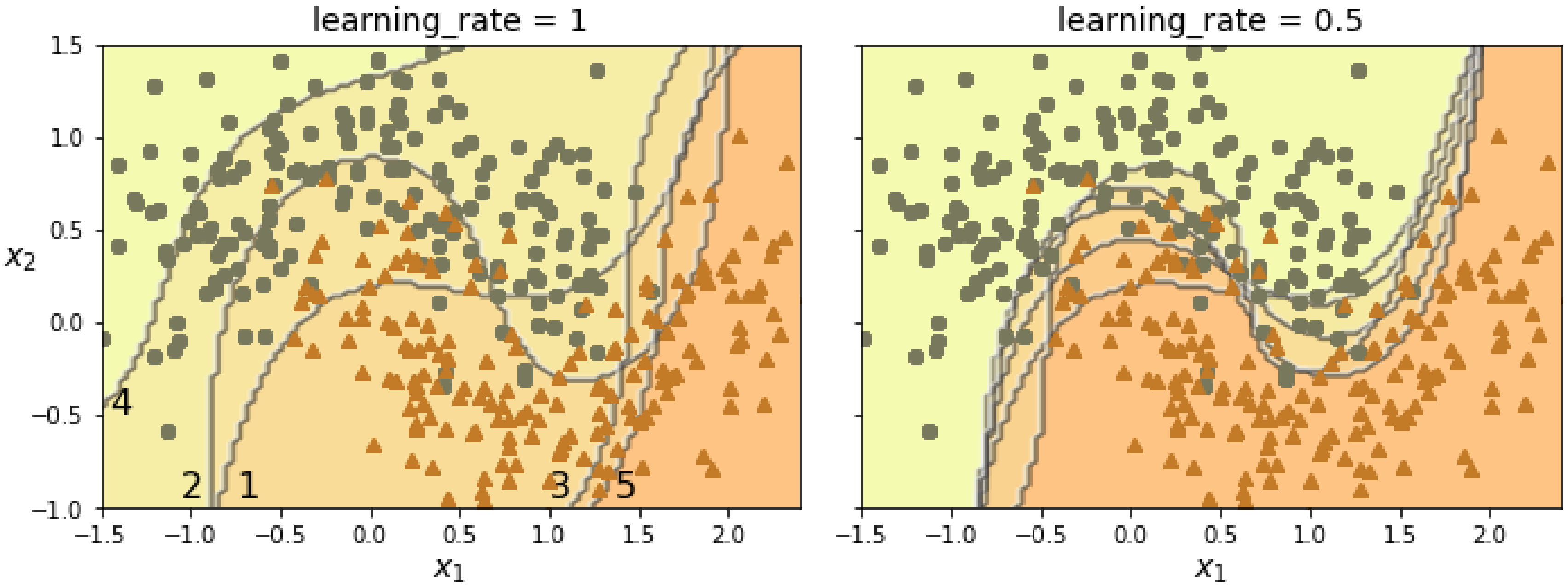
sklearn.ensemble.RandomForestClassifier

class `sklearn.ensemble.`**`RandomForestClassifier`**(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

Boosting (1)

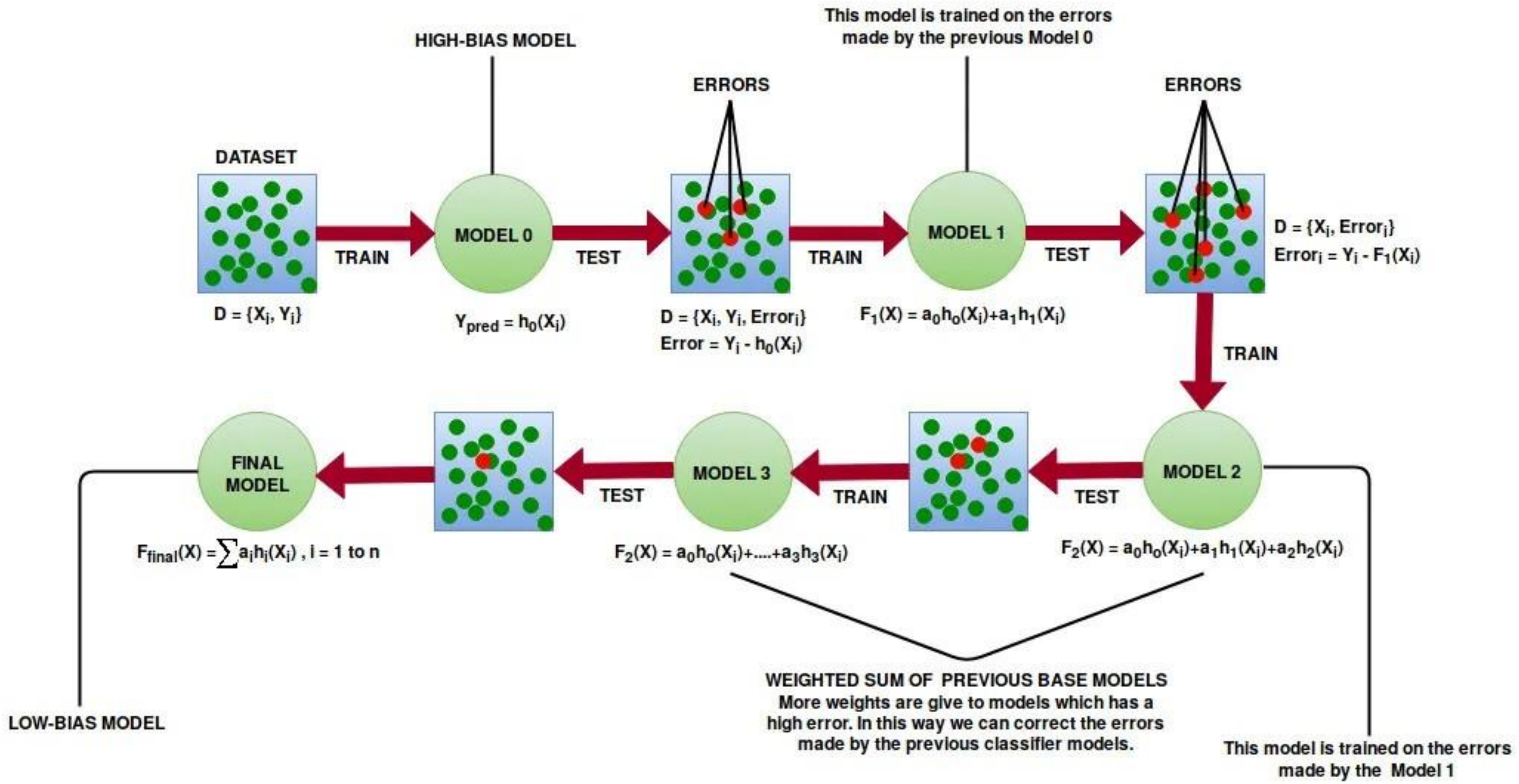
- Sequentieel trainen van classifiers
  - Nadeel: niet parallelliseerbaar
- Algoritmes:
  - AdaBoost
    - Fout geclassificeerde instances door het vorige model krijgen hoger gewicht in (de beoordeling van) het volgende model
  - Gradient Boosting
    - Volgende model wordt getraind op de error (y-y_pred) van het vorige
- Learning rate η

Boosting (2) - AdaBoost



AdaBoost met verschillende Learning Rates

Boosting (3) – gradient boosting

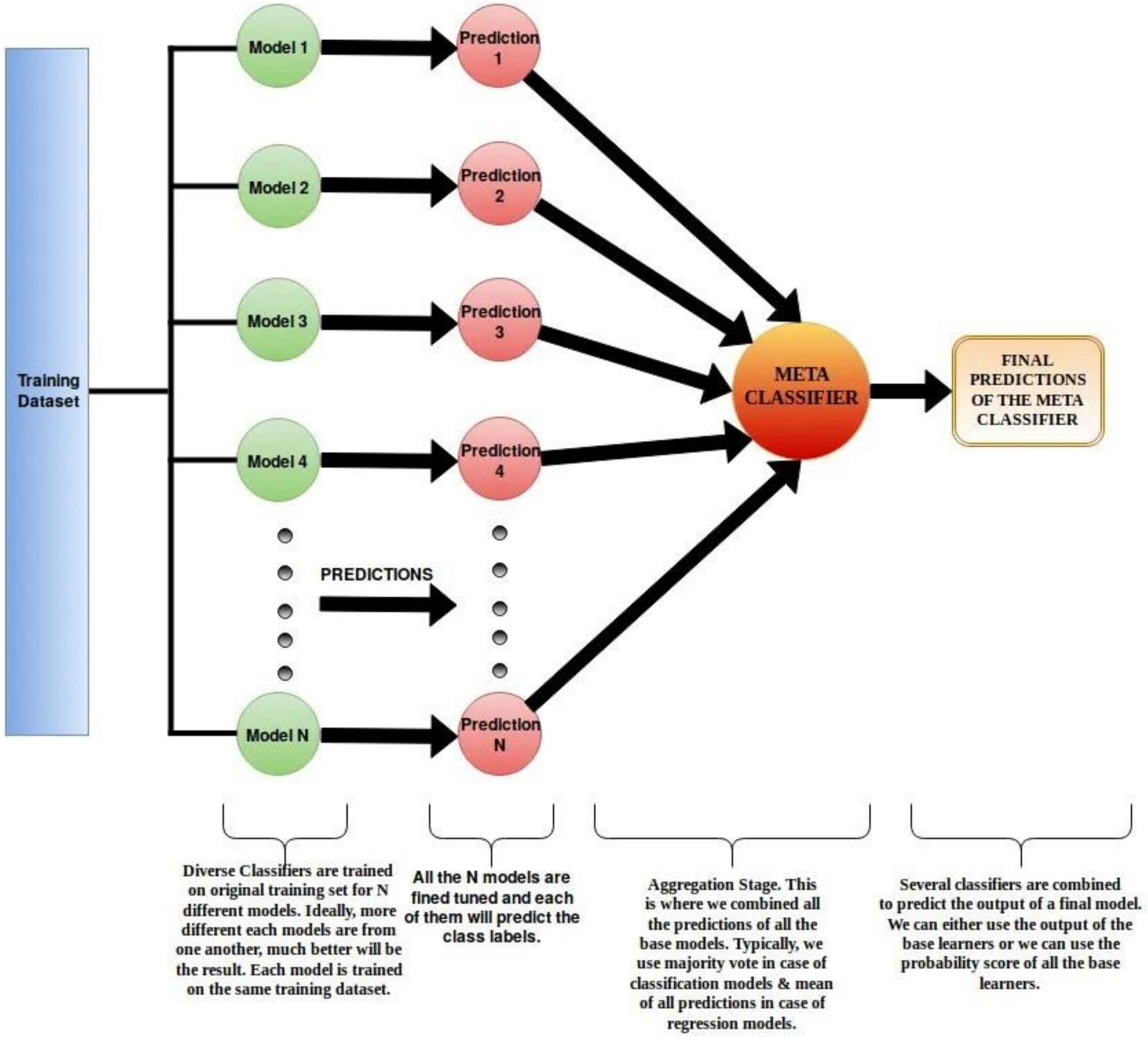


THE DIFFERENT STAGES OF A BOOSTING ALGORITHM

Stacking (1)

- Alternatief voor Voting en Bagging/Pasting
- Gebruik geen simpel algoritme zoals Voting…
- …maar gebruik weer een Machine Learning-model!
- Dit model wordt gevoed door de onderliggende classifier-modellen
- *Blender* of *Meta learner*

Stacking (2)



THE DIFFERENT STAGES OF A STACKING ALGORITHM

Afsluiting

- Live Notebook over Random Forests