CHALMERS
UNIVERSITY OF TECHNOLOGY

DAT405 – Part 2: Statistical methods in Data Science and AI

DAT405, lp2 2020, Module 4-5.

Lecture 10

In Bishop (Pattern Recognition and Machine Learning):

Sections 6 intro, 6.1, 6.2, 7 intro 7.1.1-7.1.4

Section 14 intro and 14.4

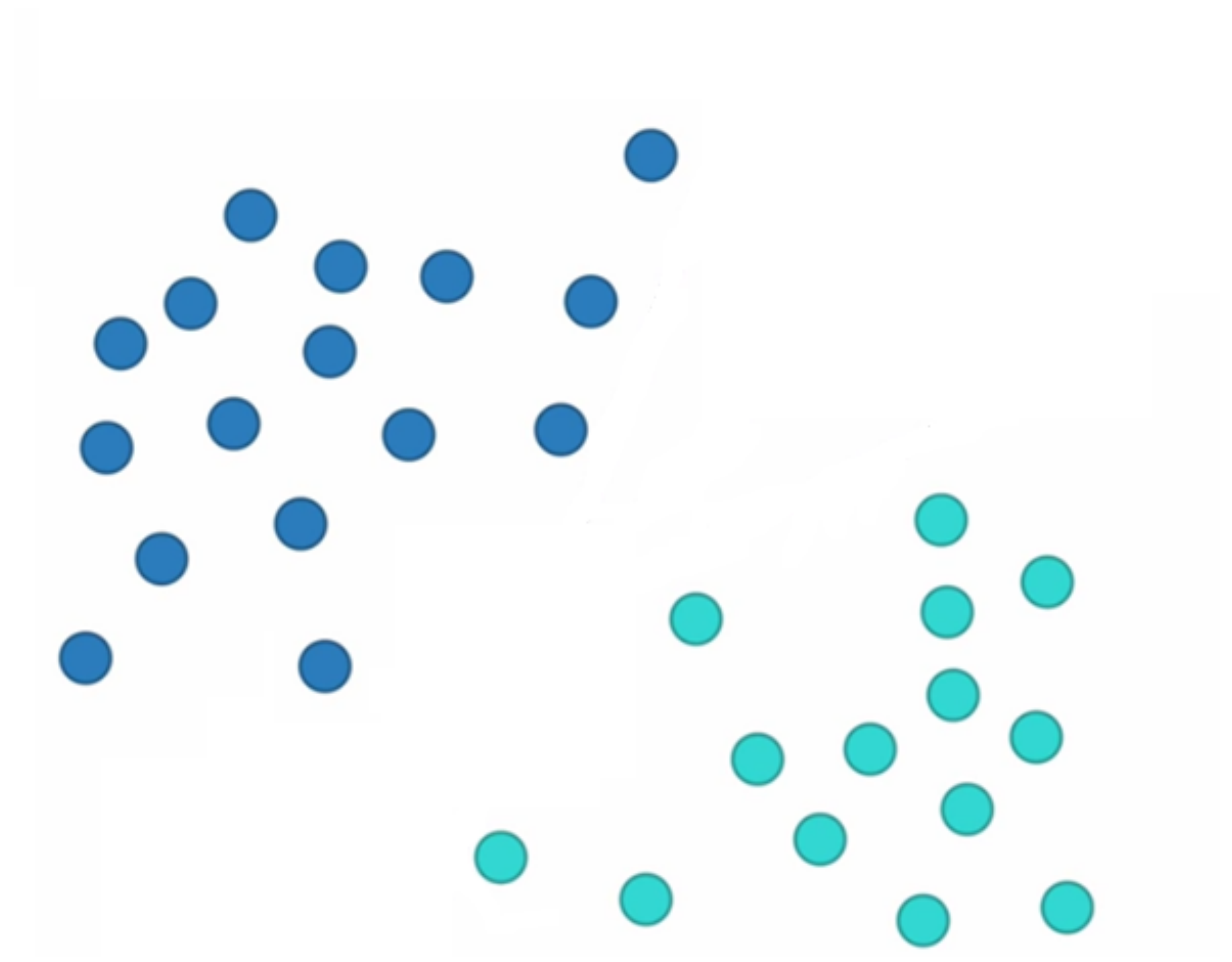
Simon Olsson

Polls

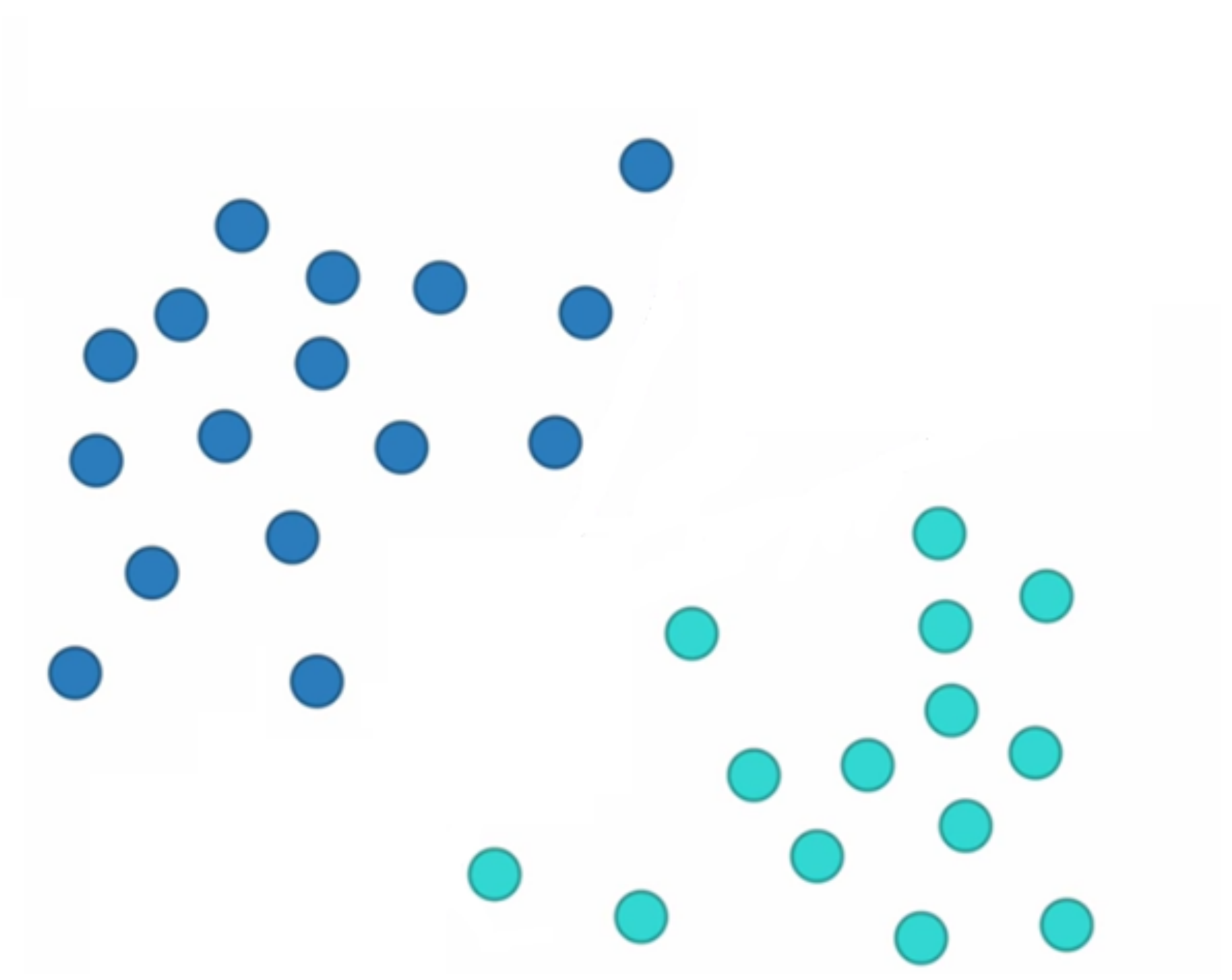
Notebook example

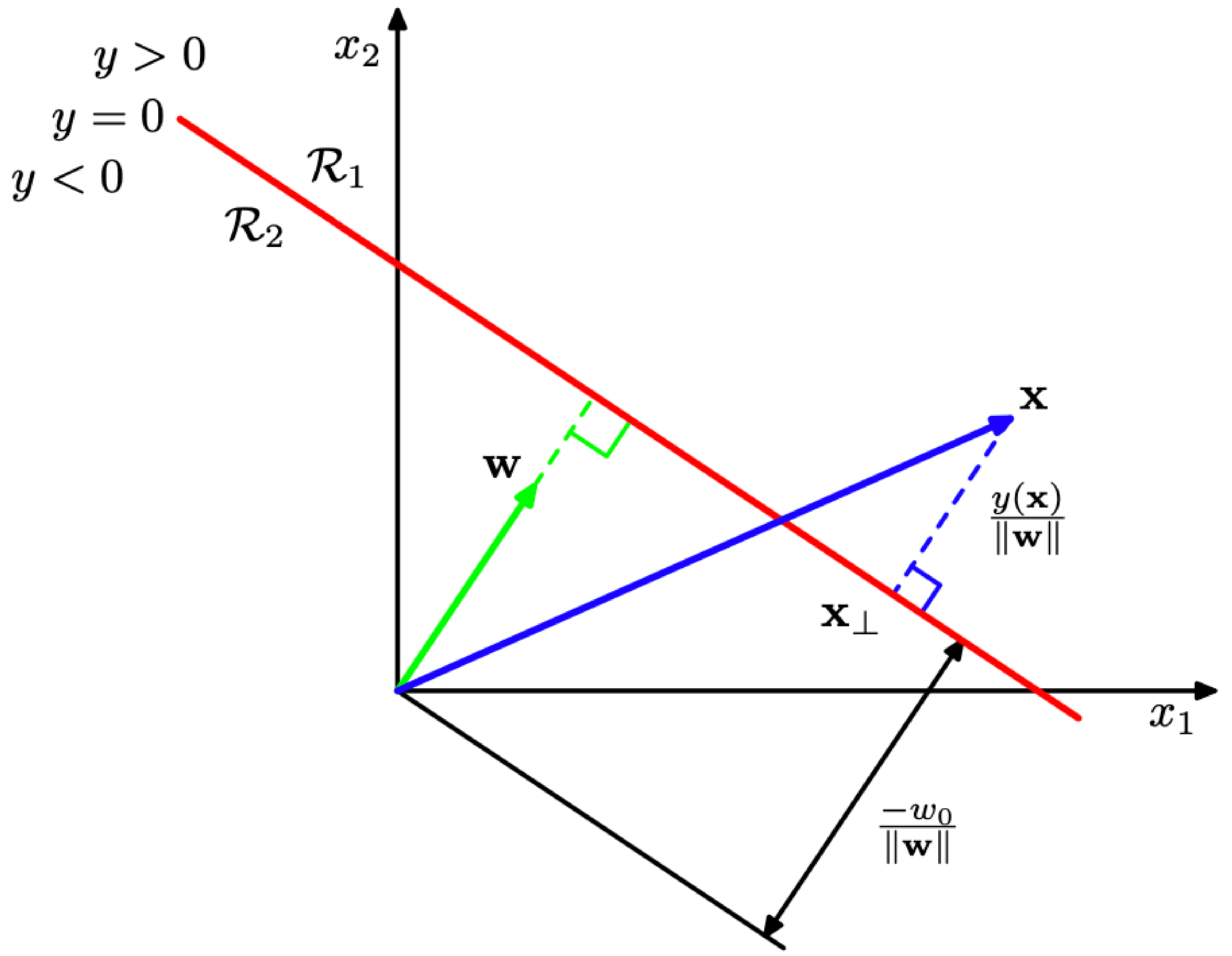
Module 5.2: Kernel methods and Decision trees

Classification using Support Vector Machines

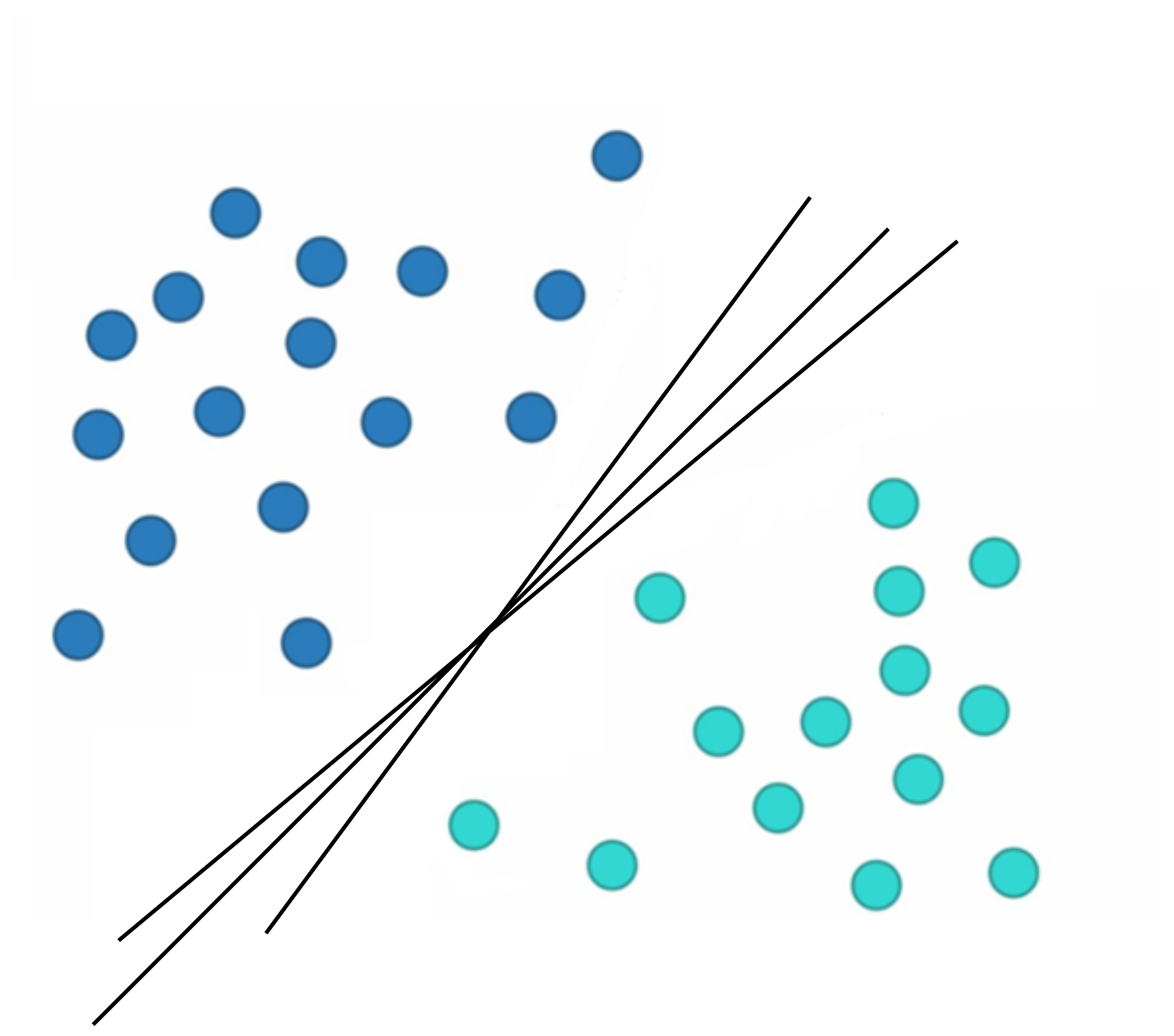


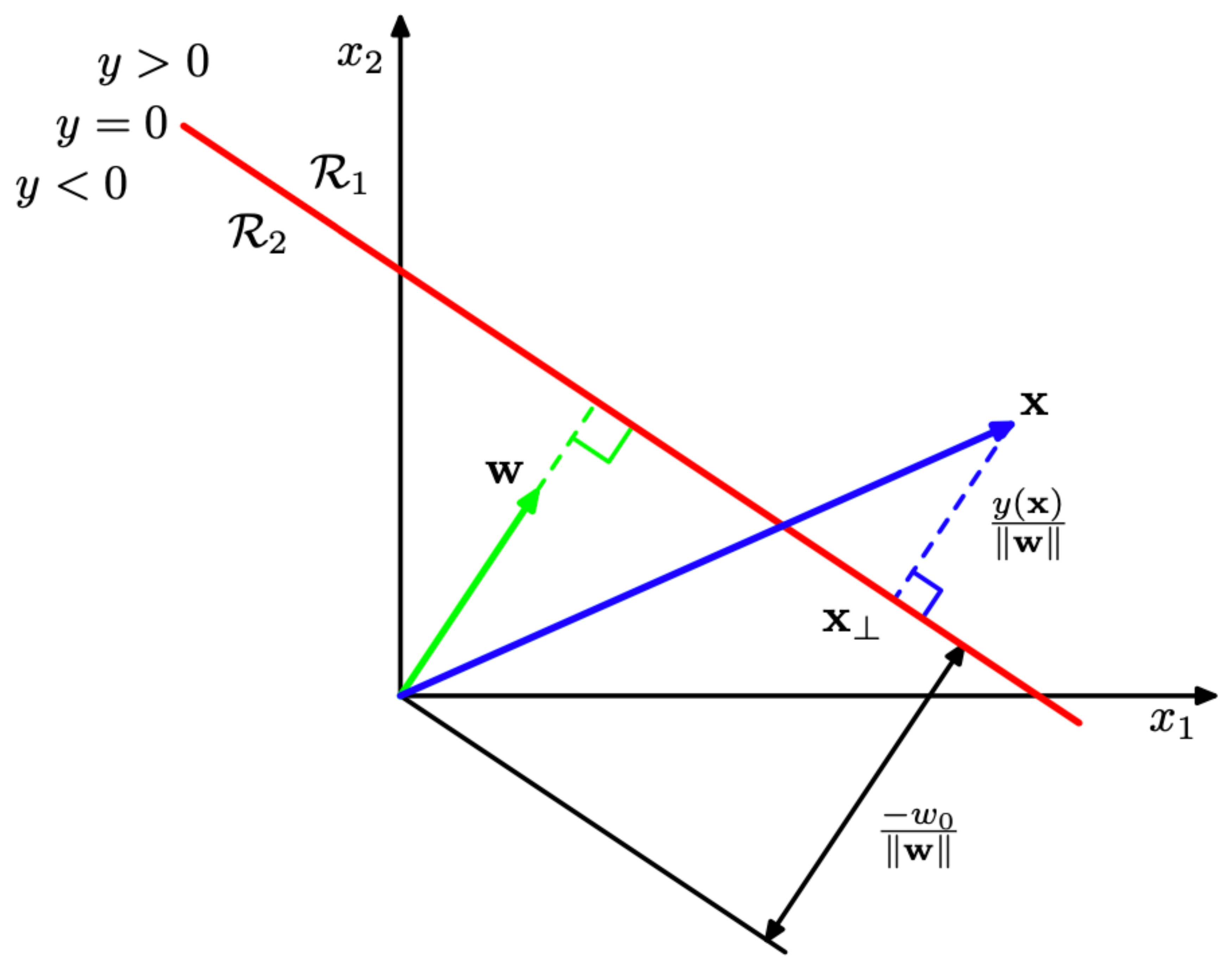
Classification using Support Vector Machines





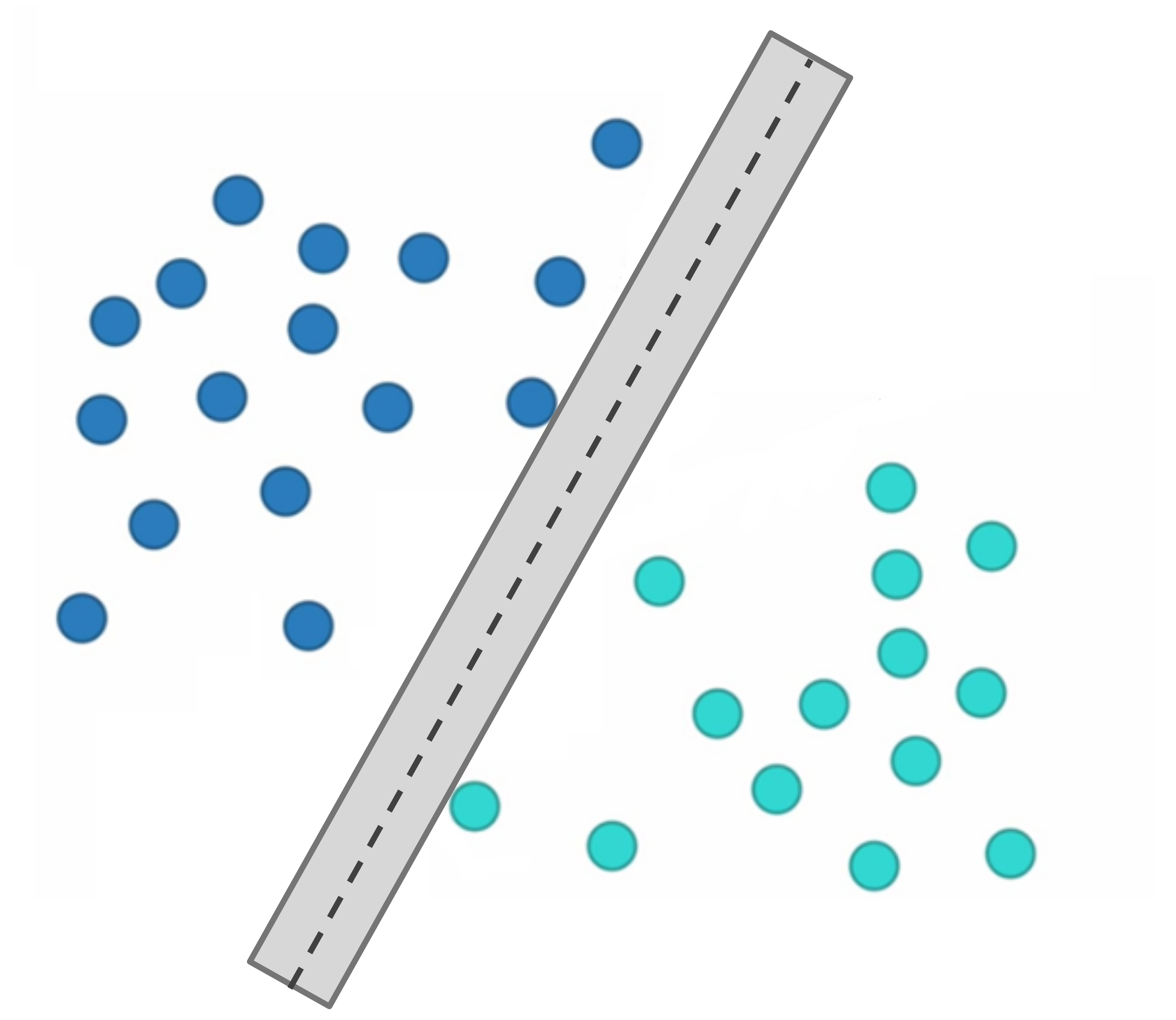
Classification using Support Vector Machines



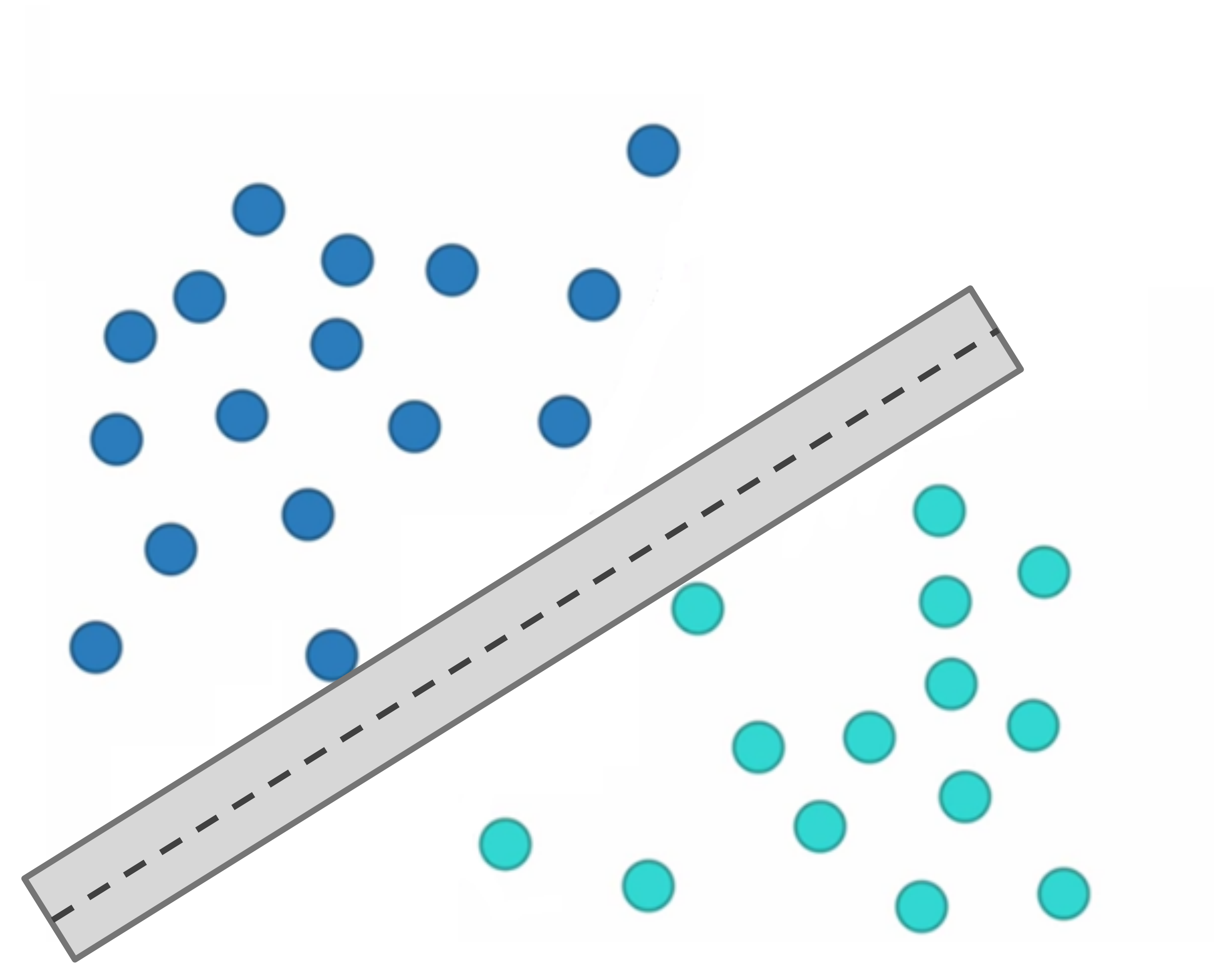


Classification using Support Vector Machines

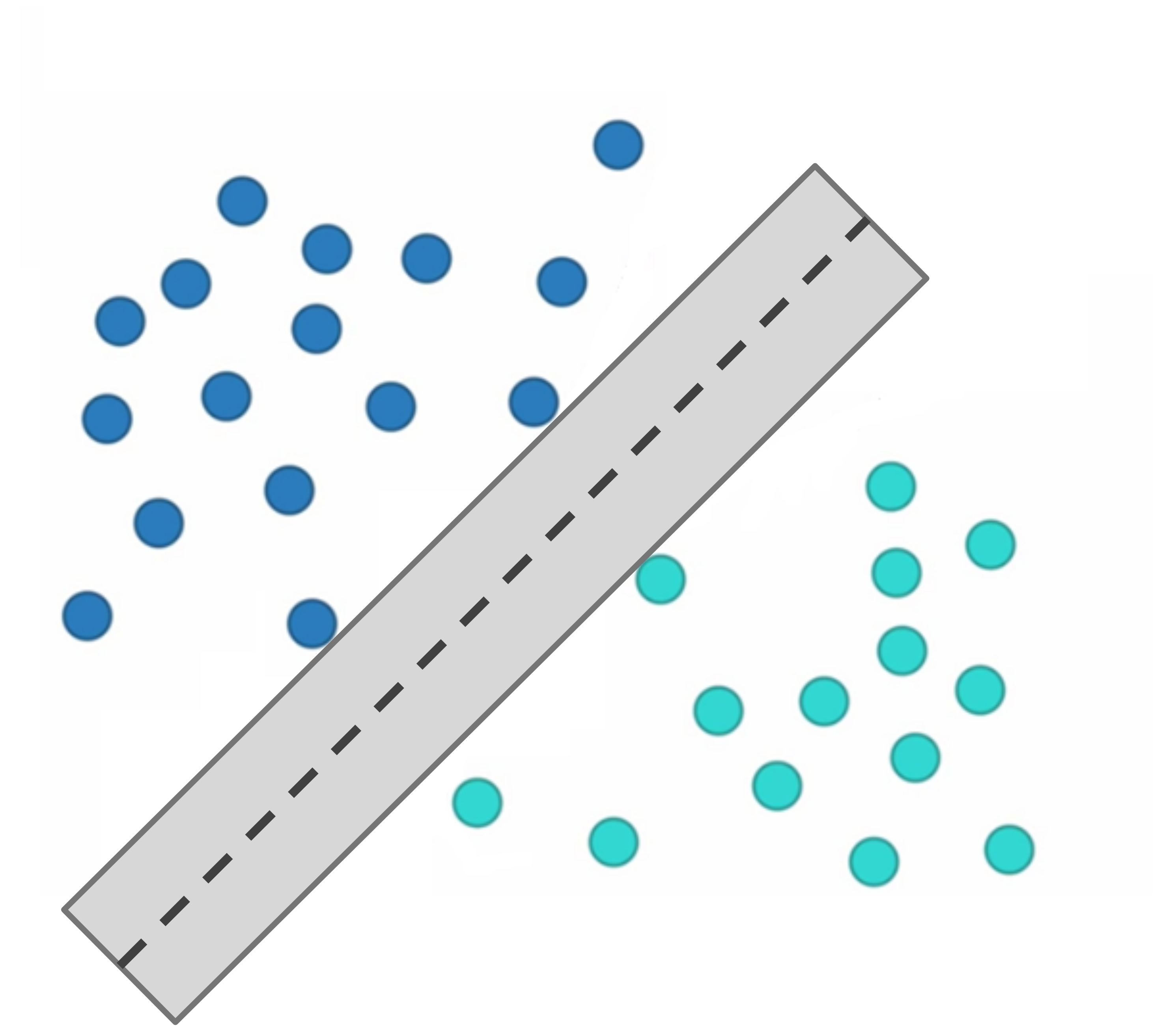
A



B

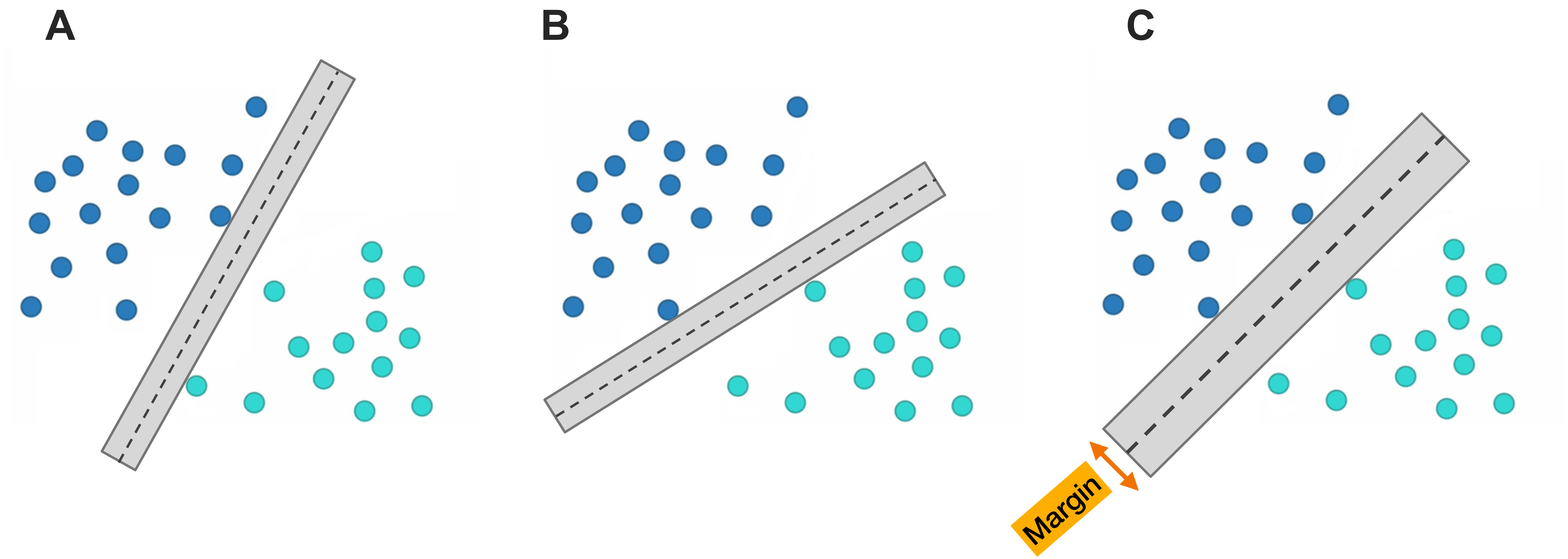


C



What decision plane do you think is the best?

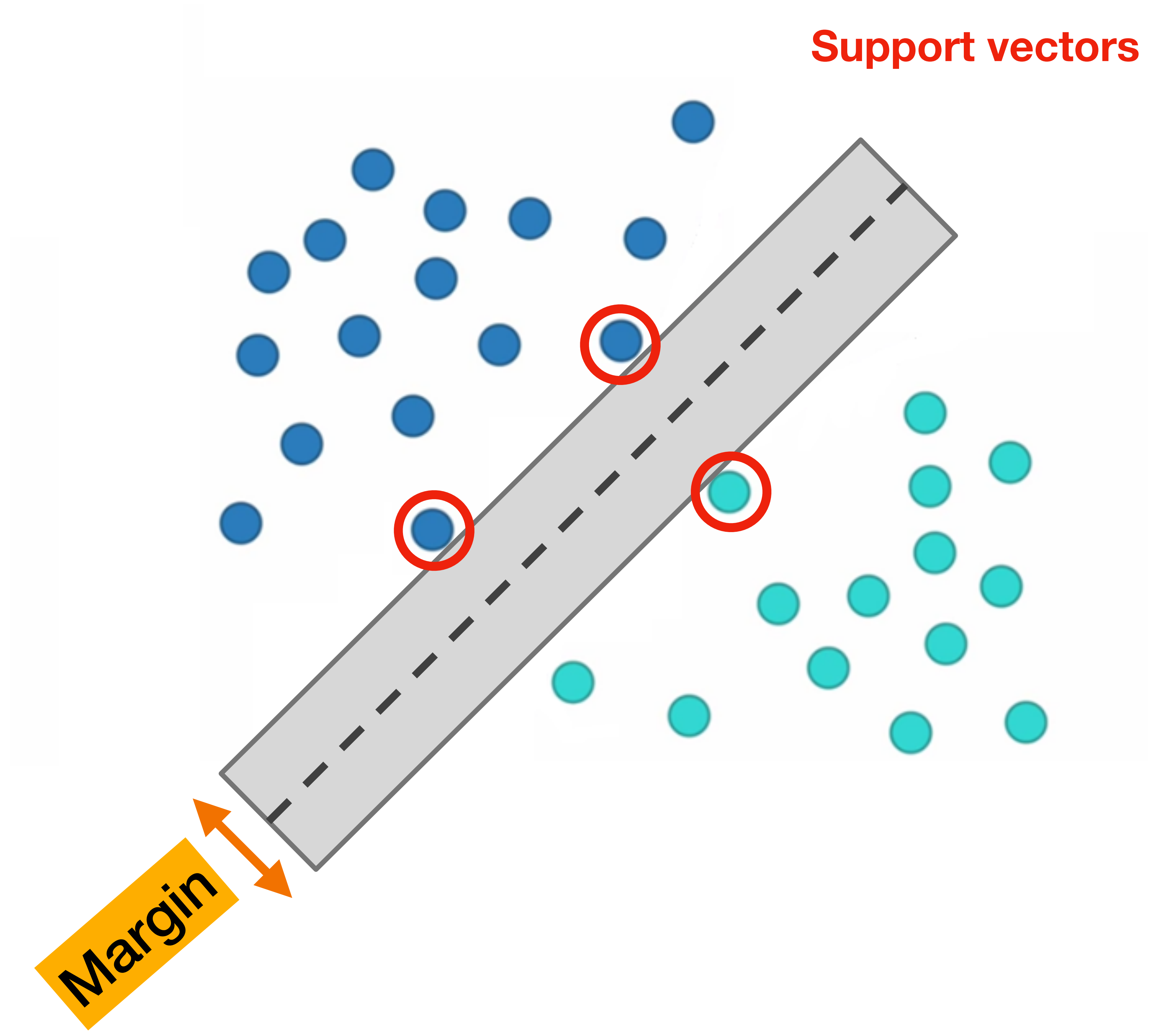
Classification using Support Vector Machines



What decision plane do you think is the best?

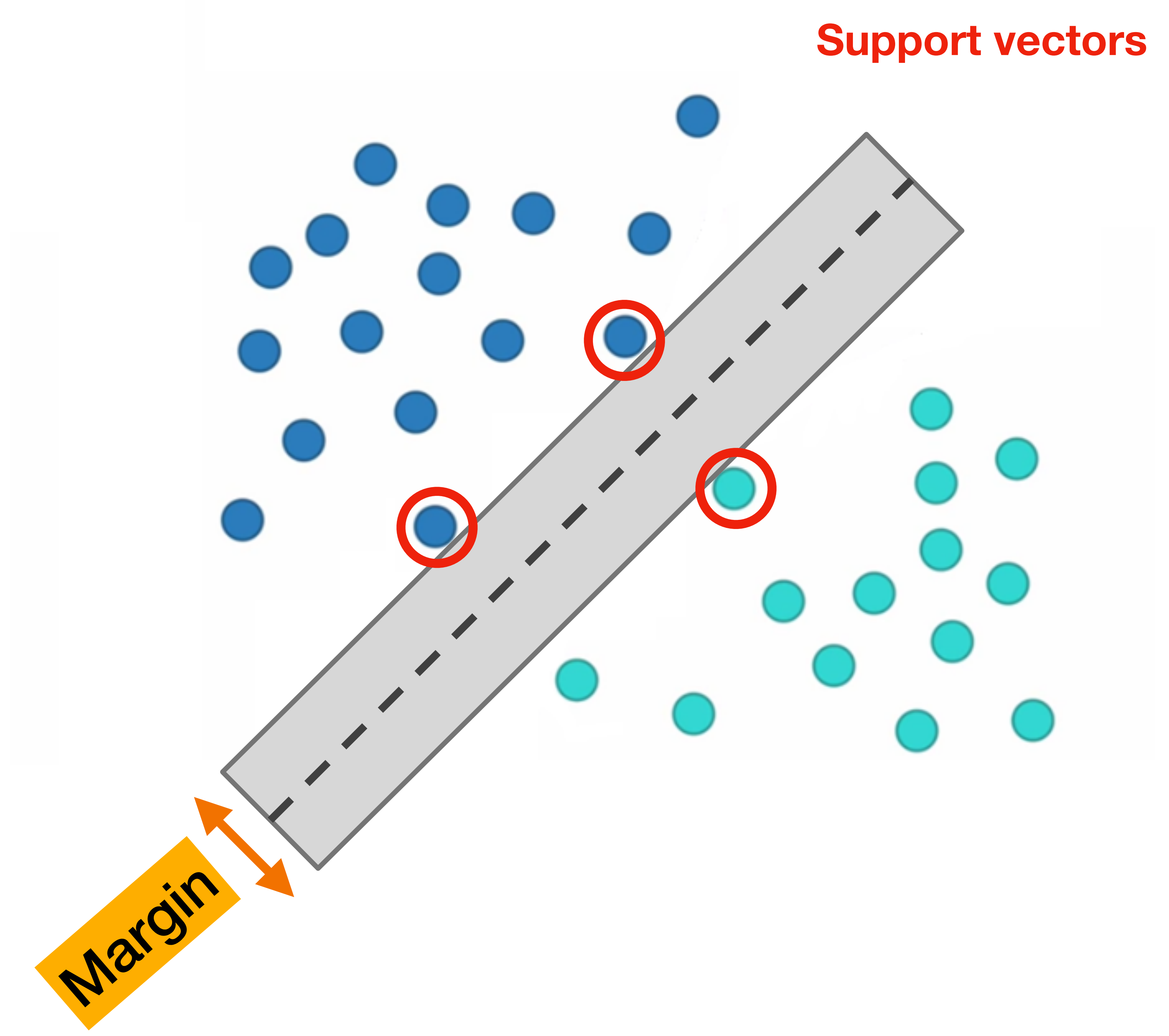
C: It has the largest "margin" — better generalisation!

Classification using Support Vector Machines



The SVM algorithm defines the optimal splitting "hyperplane". The data-points closest to the splitting hyper-plane are called support vectors.

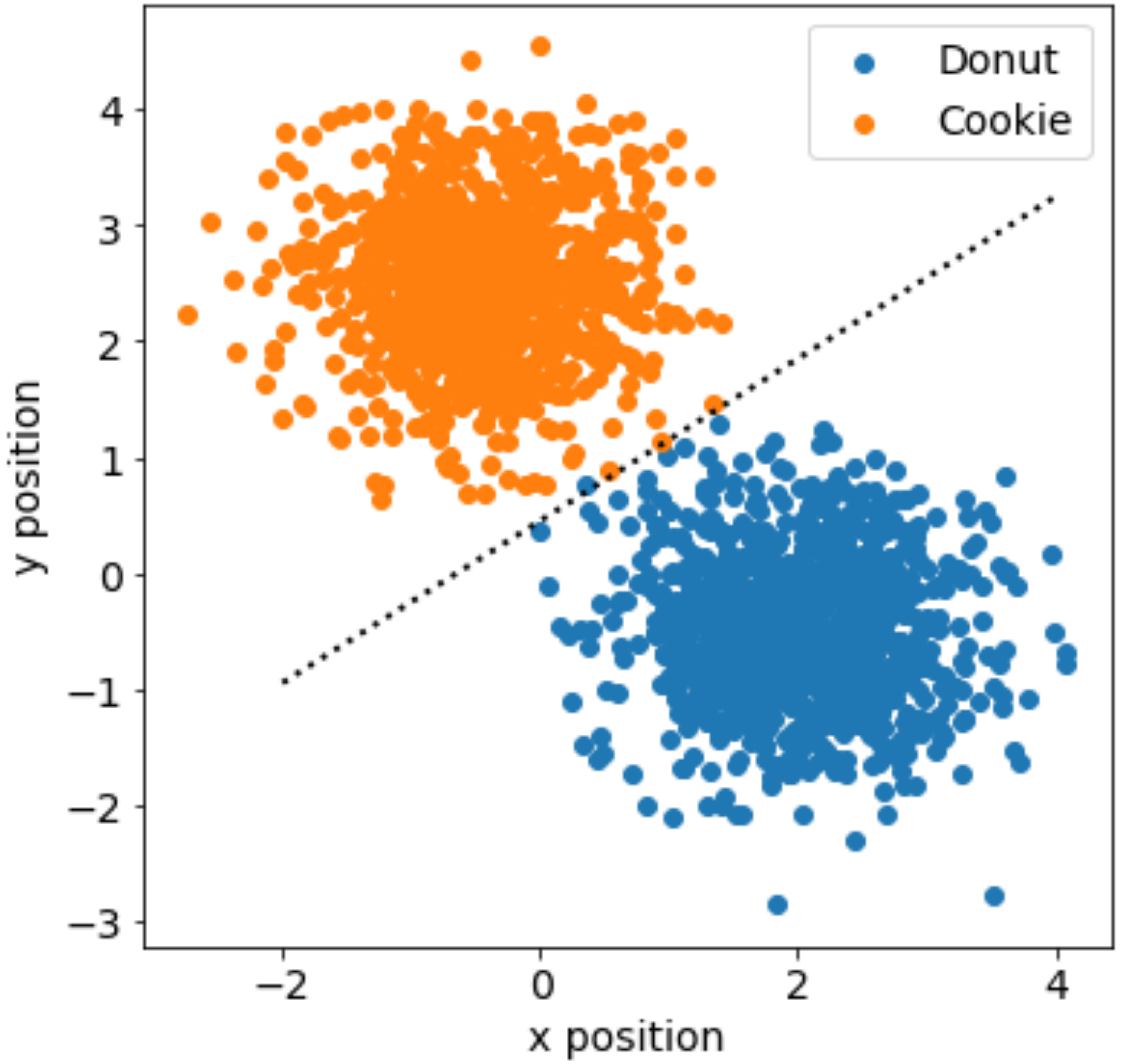
Classification using Support Vector Machines



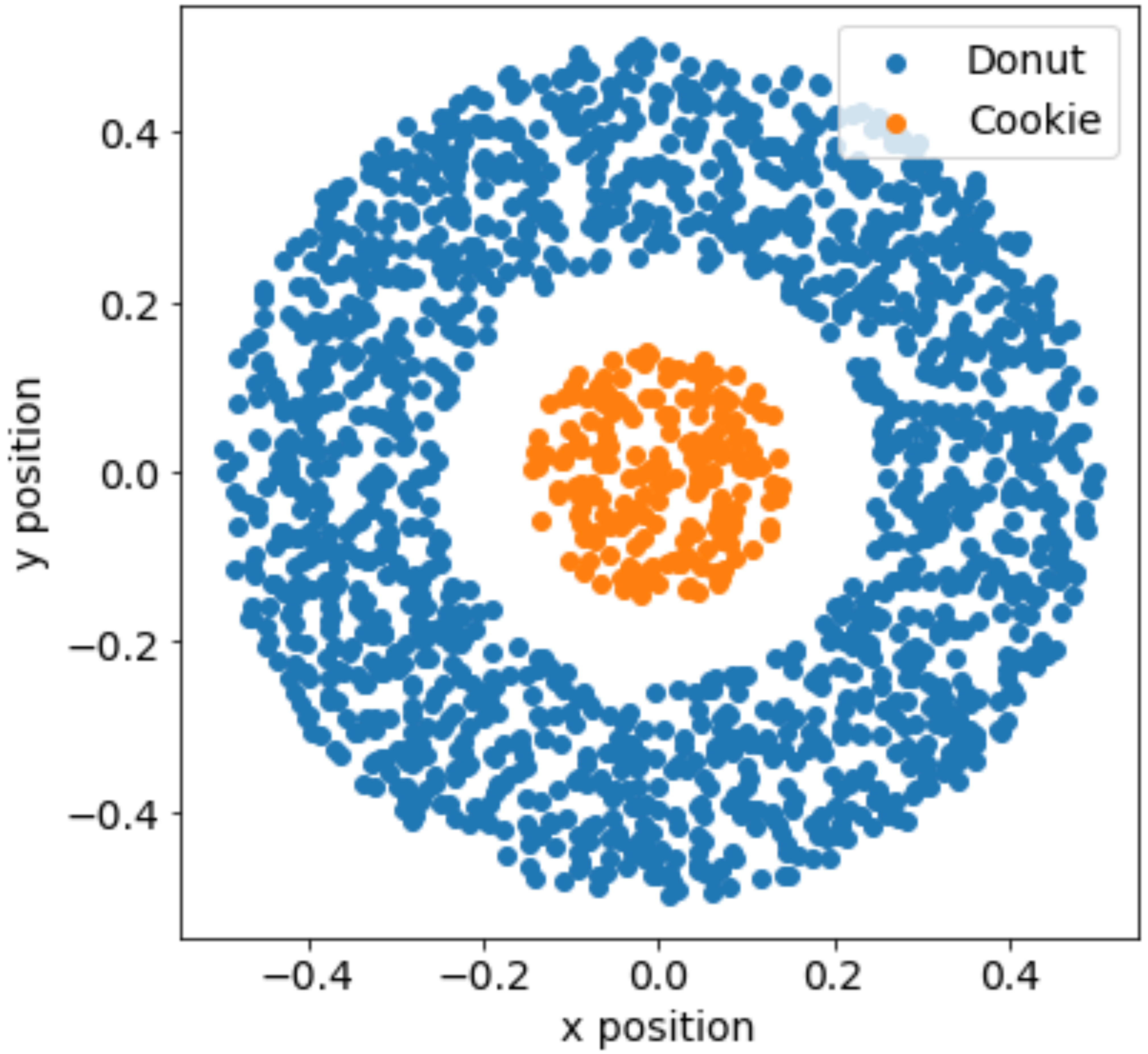
Look at:
Bishop Chapter 7
And
Jupiter Notebook on Canvas
For practical examples

The SVM algorithm defines the optimal splitting "hyperplane". The data-points closest to the splitting hyper-plane are called support vectors.

Linear problems

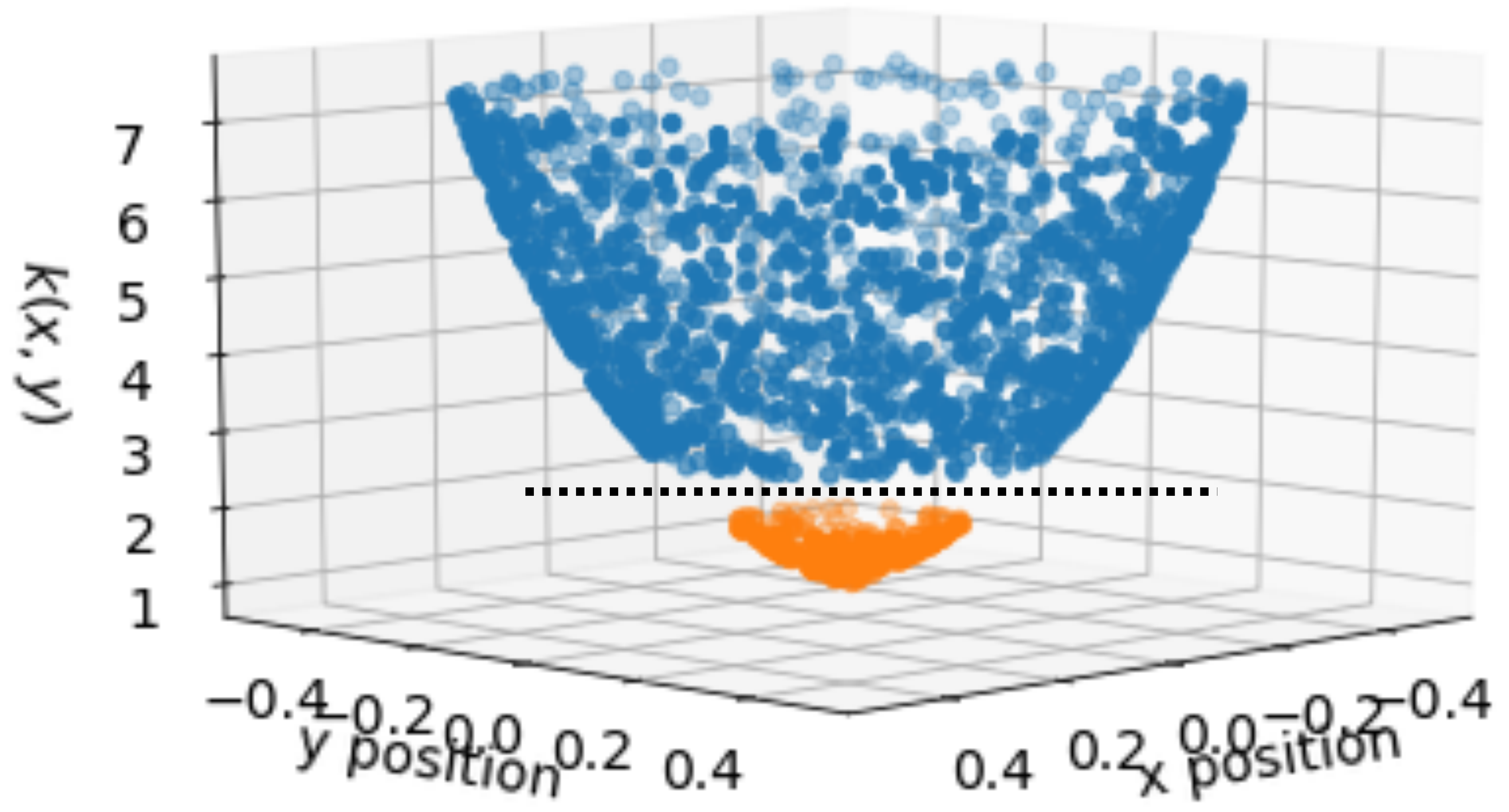


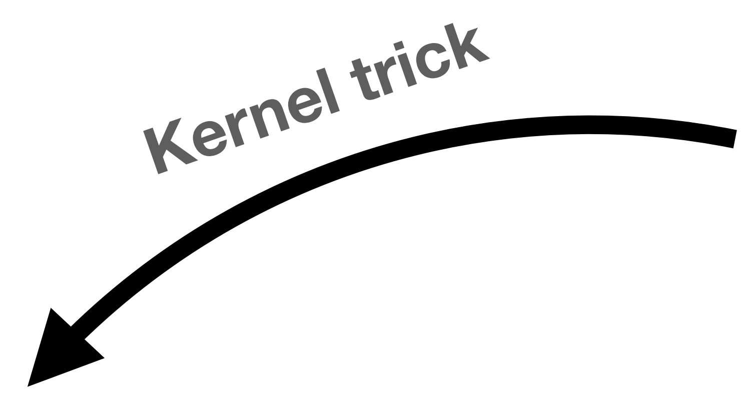
Non-linear problems

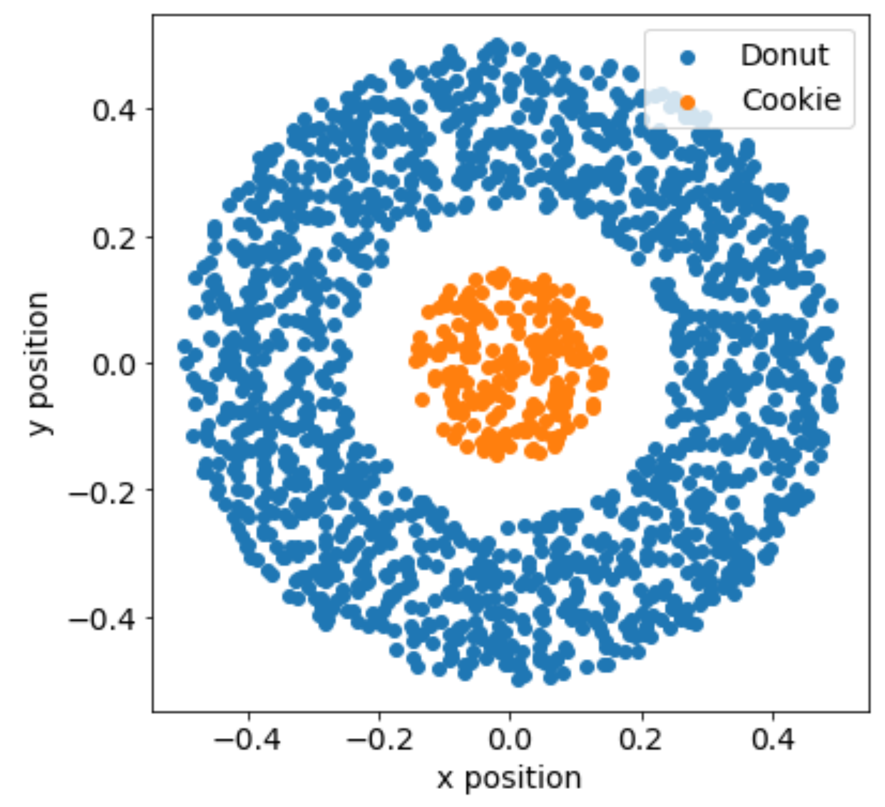


How do we draw a linear decision boundary?

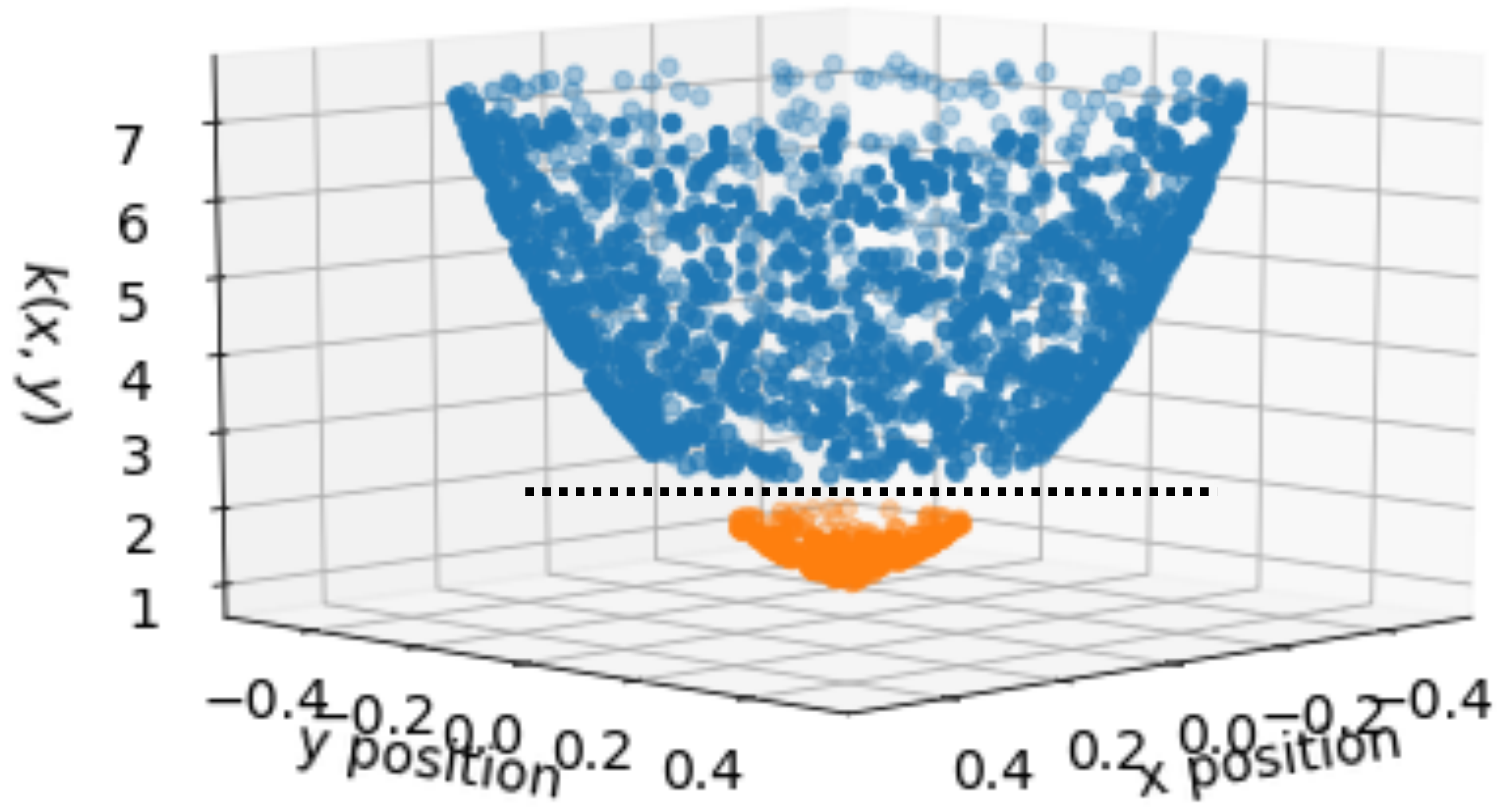
Non-linear problems

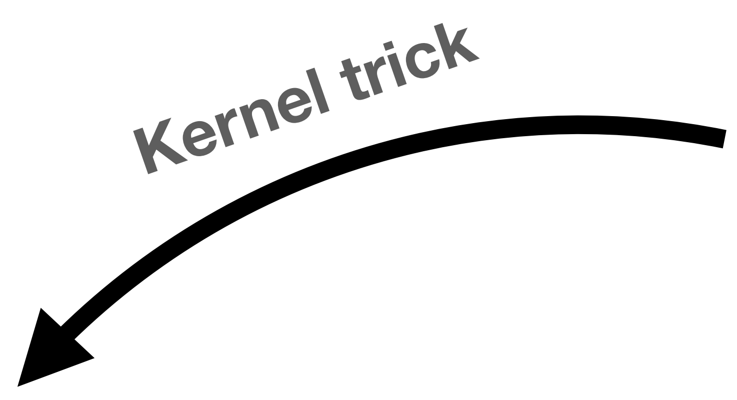


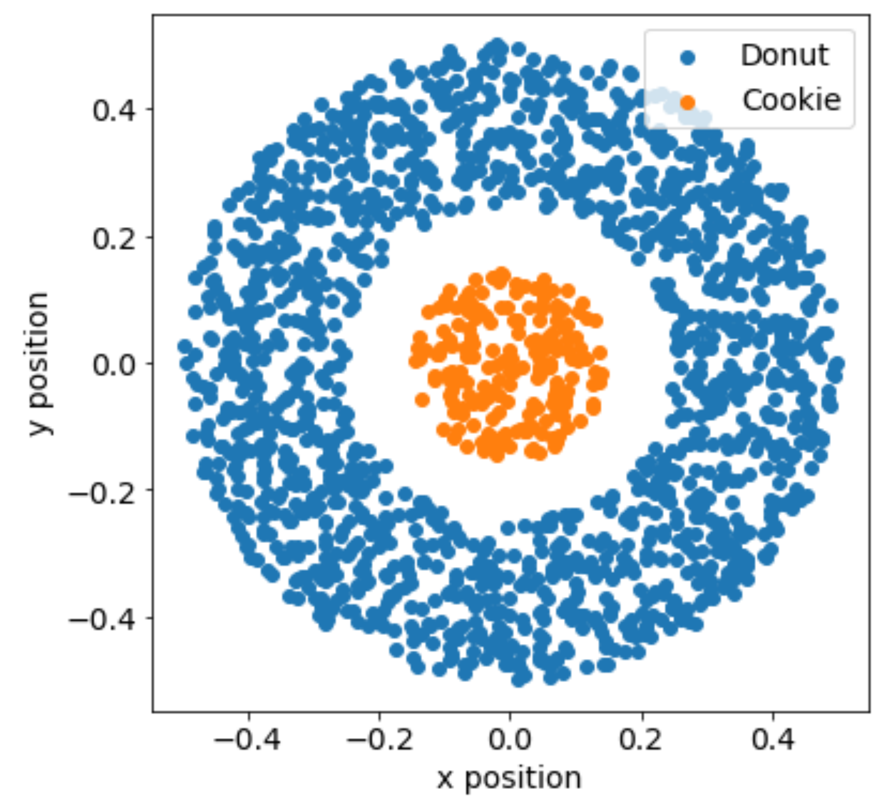




Non-linear problems







Kernels allow us to *implicitly* operate in high dimensional feature spaces to avoid learning non-linear functions.

Parametric models

Fixed basis functions:

$\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\mathbf{x}), \ldots \phi_{M-1}(\mathbf{x}))^\top$

Used in:

- Linear regression: $\mathbf{y} = \mathbf{a}^\top \boldsymbol{\phi}(\boldsymbol{x}) + b$
- Linear classification: $\mathbf{y} = f(\mathbf{a}^\top \boldsymbol{\phi}(\boldsymbol{x}) + b)$

Where we 'train' models using

- Maximum Likelihood Estimation, point estimation ($\mathbf{a}, b$)
- Specification of a posterior distribution $p(\mathbf{a}, b \mid \mathbf{X}, \mathbf{y})$

Parametric models

Fixed basis functions:

$\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\mathbf{x}), \ldots \phi_{M-1}(\mathbf{x}))^\top$

Used in:

- Linear regression: $\mathbf{y} = \mathbf{a}^\top \boldsymbol{\phi}(\boldsymbol{x}) + b$
- Linear classification: $\mathbf{y} = f(\mathbf{a}^\top \boldsymbol{\phi}(\boldsymbol{x}) + b)$

Where we 'train' models using

- Maximum Likelihood Estimation, point estimation ($\mathbf{a}, b$)
- Specification of a posterior distribution $p(\mathbf{a}, b \mid \mathbf{X}, \mathbf{y})$

Once our models are trained — our data is discarded.

Data is represented only by parameters $\mathbf{a}$ and b

Non-parametric models

No explicit parameters to be estimated.
We can work in high-dimensional (even infinite dimensional) function spaces.

Our data is not discarded after specifying the model, it is part of the model.

Kernel methods

Kernel methods use a (subset) of our data during predictions.

Our data enters via a *kernel function*

$$k(\mathbf{x}', \mathbf{x}) = k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$$

The kernel measures the similarity between two points in our feature space defined by the function $\boldsymbol{\phi}(\mathbf{x})$.

Kernel methods

Kernel methods use a (subset) of our data during predictions.

Our data enters via a *kernel function*

$$k(\mathbf{x}', \mathbf{x}) = k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$$

The kernel measur
two points in our fe
the function $\boldsymbol{\phi}(\mathbf{x})$.

What does the first equal sign imply in the definition of the kernel function?

Revisiting linear regression — with regularisation

In regularised linear regression we want to?

Revisiting linear regression — with regularisation

In regularised linear regression we want to

Minimize the squared error of prediction:

$$\ell(\mathbf{a}) = \frac{1}{2}\sum_{i=1}^{N}(\mathbf{a}^\top \boldsymbol{\phi}(\mathbf{x}_i) - y_i)^2 + \frac{\lambda}{2}\mathbf{a}^\top \mathbf{a}$$

Revisiting linear regression — with regularisation

In regularised linear regression we want to

Minimize the squared error of prediction:

$$\ell(\mathbf{a}) = \frac{1}{2}\sum_{i=1}^{N}(\mathbf{a}^\top\boldsymbol{\phi}(\mathbf{x}_i) - y_i)^2 + \frac{\lambda}{2}\mathbf{a}^\top\mathbf{a}$$

Tikhonov regularization (L2)
Gaussian Prior

Revisiting linear regression — with regularisation

In regularised linear regression we want to

Minimize the squared error of prediction:

$$\ell(\mathbf{a}) = \frac{1}{2}\sum_{i=1}^{N}(\mathbf{a}^\top \boldsymbol{\phi}(\mathbf{x}_i) - y_i)^2 + \frac{\lambda}{2}\mathbf{a}^\top\mathbf{a}$$

The regularizer changes our Maximum Likelihood estimator from last week from $\hat{\mathbf{a}} = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{y}$ to:

$\hat{\mathbf{a}} = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\mathbb{I}_M)^{-1}\boldsymbol{\Phi}^\top\mathbf{y}$ with $\boldsymbol{\Phi} = \{\boldsymbol{\phi}(\mathbf{x}_1)^\top, \ldots, \boldsymbol{\phi}(\mathbf{x}_M)^\top\} \in \mathbb{R}^{N\times M}$

Revisiting linear regression

Our estimator for regularised linear regression is

$\hat{\mathbf{a}} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \mathbb{I}_M)^{-1} \boldsymbol{\Phi}^\top \mathbf{y}$ with $\boldsymbol{\Phi} = \{\boldsymbol{\phi}(\mathbf{x}_1)^\top, \ldots, \boldsymbol{\phi}(\mathbf{x}_M)^\top\} \in \mathbb{R}^{N \times M}$

We can use the matrix inversion lemma from Bishop (C.5)
$(P^{-1} + B^\top R^{-1} B)^{-1} B^\top R^{-1} = P B^\top (B P B^\top + R)^{-1}$

Revisiting linear regression

Our estimator for regularised linear regression is

$\hat{\mathbf{a}} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \mathbb{I}_M)^{-1} \boldsymbol{\Phi}^\top \mathbf{y}$ with $\boldsymbol{\Phi} = \{\boldsymbol{\phi}(\mathbf{x}_1)^\top, \ldots, \boldsymbol{\phi}(\mathbf{x}_M)^\top\} \in \mathbb{R}^{N \times M}$

We can use the matrix inversion lemma from Bishop (C.5)

$(P^{-1} + B^\top R^{-1} B)^{-1} B^\top R^{-1} = PB^\top (BPB^\top + R)^{-1}$

$\mathbb{I}_N$

To rewrite $\hat{\mathbf{a}}$ in the following form

$\hat{\mathbf{a}} = \boldsymbol{\Phi}^\top (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \lambda \mathbb{I}_N)^{-1} \mathbf{y} = \boldsymbol{\Phi}^\top (\mathbf{K} + \lambda \mathbb{I}_N)^{-1} \mathbf{y}$

Revisiting linear regression

Our estimator for regularised linear regression is

$\hat{\mathbf{a}} = (\mathbf{\Phi}^\top \mathbf{\Phi} + \lambda \mathbb{I}_M)^{-1} \mathbf{\Phi}^\top \mathbf{y}$ with $\mathbf{\Phi} = \{\boldsymbol{\phi}(\mathbf{x}_1)^\top, \ldots, \boldsymbol{\phi}(\mathbf{x}_M)^\top\} \in \mathbb{R}^{N \times M}$

We can use the matrix inversion lemma from Bishop (C.5)
$(P^{-1} + B^\top R^{-1} B)^{-1} B^\top R^{-1} = PB^\top (BPB^\top + R)^{-1}$

$\mathbb{I}_N$

To rewrite $\hat{\mathbf{a}}$ in the following form
$\hat{\mathbf{a}} = \mathbf{\Phi}^\top (\mathbf{\Phi}\mathbf{\Phi}^\top + \lambda \mathbb{I}_N)^{-1} \mathbf{y} = \mathbf{\Phi}^\top (\mathbf{K} + \lambda \mathbb{I}_N)^{-1} \mathbf{y}$

With the Gramian matrix $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^\top$ with $K_{ij} = \boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}_j)$

Revisiting linear regression

Our estimator for regularised linear regression is

$\hat{\mathbf{a}} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \mathbb{I}_M)^{-1} \boldsymbol{\Phi}^\top \mathbf{y}$ with $\boldsymbol{\Phi} = \{\boldsymbol{\phi}(\mathbf{x}_1)^\top, \ldots, \boldsymbol{\phi}(\mathbf{x}_M)^\top\} \in \mathbb{R}^{N \times M}$

We can use the matrix inversion lemma from Bishop (C.5)

$(P^{-1} + B^\top R^{-1} B)^{-1} B^\top R^{-1} = P B^\top (B P B^\top + R)^{-1}$

$\mathbb{I}_N$

To rewrite $\hat{\mathbf{a}}$ in the following form

$\hat{\mathbf{a}} = \boldsymbol{\Phi}^\top (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \lambda \mathbb{I}_N)^{-1} \mathbf{y} = \boldsymbol{\Phi}^\top (\mathbf{K} + \lambda \mathbb{I}_N)^{-1} \mathbf{y}$

With the Gramian matrix $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ with $K_{ij} = \boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}_j)$

A Kernel

Revisiting linear regression — Primal/Dual

We can now write a dual form of the regularised regression problem

$$\hat{\mathbf{a}} = \mathbf{\Phi}^\top \underbrace{(\mathbf{K} + \lambda \mathbb{I}_N)^{-1}\mathbf{y}}_{\boldsymbol{\alpha}}$$

- **Primal variable** perspective: $\mathbf{a} = \mathbf{\Phi}^\top \boldsymbol{\alpha}$
  - Prediction: $y(x') = \mathbf{a}^\top \boldsymbol{\phi}(x')$
- **Dual variable** perspective: $\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbb{I}_N)^{-1}\mathbf{y}$
  - Prediction: $y(x') = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}')$

Revisiting linear regression — Primal/Dual

We can now write a dual form of the regularised regression problem

$$\hat{\mathbf{a}} = \boldsymbol{\Phi}^\top \underbrace{(\mathbf{K} + \lambda \mathbb{I}_N)^{-1}\mathbf{y}}_{\boldsymbol{\alpha}}$$

- **Primal variable** perspective: $\mathbf{a} = \boldsymbol{\Phi}^\top \boldsymbol{\alpha}$
  - Prediction: $y(x') = \mathbf{a}^\top \boldsymbol{\phi}(x')$

Explicit featurization!

- **Dual variable** perspective: $\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbb{I}_N)^{-1}\mathbf{y}$
  - Prediction: $y(x') = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}')$

Linear combination of Kernels (inner products)

Revisiting linear regression — Primal/Dual

Why bother with the dual representation?

$N >> M$

- **Primal variable** perspective: $\mathbf{a} = \boldsymbol{\Phi}^\top \boldsymbol{\alpha}$

  Inversion of an $M \times M$ matrix ($\mathcal{O}(M^3)$)
  - Prediction: $y(x') = \mathbf{a}^\top \boldsymbol{\phi}(x')$
- **Dual variable** perspective: $\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbb{I}_N)^{-1}\mathbf{y}$

  Inversion of an $N \times N$ matrix ($\mathcal{O}(N^3)$)
  - Prediction: $y(x') = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}')$

Revisiting linear regression — Primal/Dual

Why bother with the dual representation?

$N >> M$

- **Primal variable** perspective: $\mathbf{a} = \mathbf{\Phi}^\top \boldsymbol{\alpha}$
  - Prediction: $y(x') = \mathbf{a}^\top \boldsymbol{\phi}(x')$

Inversion of an $M \times M$ matrix ($\mathcal{O}(M^3)$)

- **Dual variable** perspective: $\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbb{I}_N)^{-1}\mathbf{y}$
  - Prediction: $y(x') = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}')$

Inversion of an $N \times N$ matrix ($\mathcal{O}(N^3)$)

We do not need to explicitly featurize our data, but can instead work with the **inner products** of data points in potentially infinite dimensional spaces!

Revisiting linear regression — Primal/Dual

Why bother with the dual representation?

$N >> M$

- **Primal variable** perspective: $\mathbf{a} = \boldsymbol{\Phi}^\top \boldsymbol{\alpha}$
  - Prediction: $y(x') = \mathbf{a}^\top \boldsymbol{\phi}(x')$

Inversion of an $M \times M$ matrix ($\mathcal{O}(M^3)$)

- **Dual variable** perspective: $\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbb{I}_N)^{-1}\mathbf{y}$
  - Prediction: $y(x') = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}')$

Inversion of an $N \times N$ matrix ($\mathcal{O}(N^3)$)

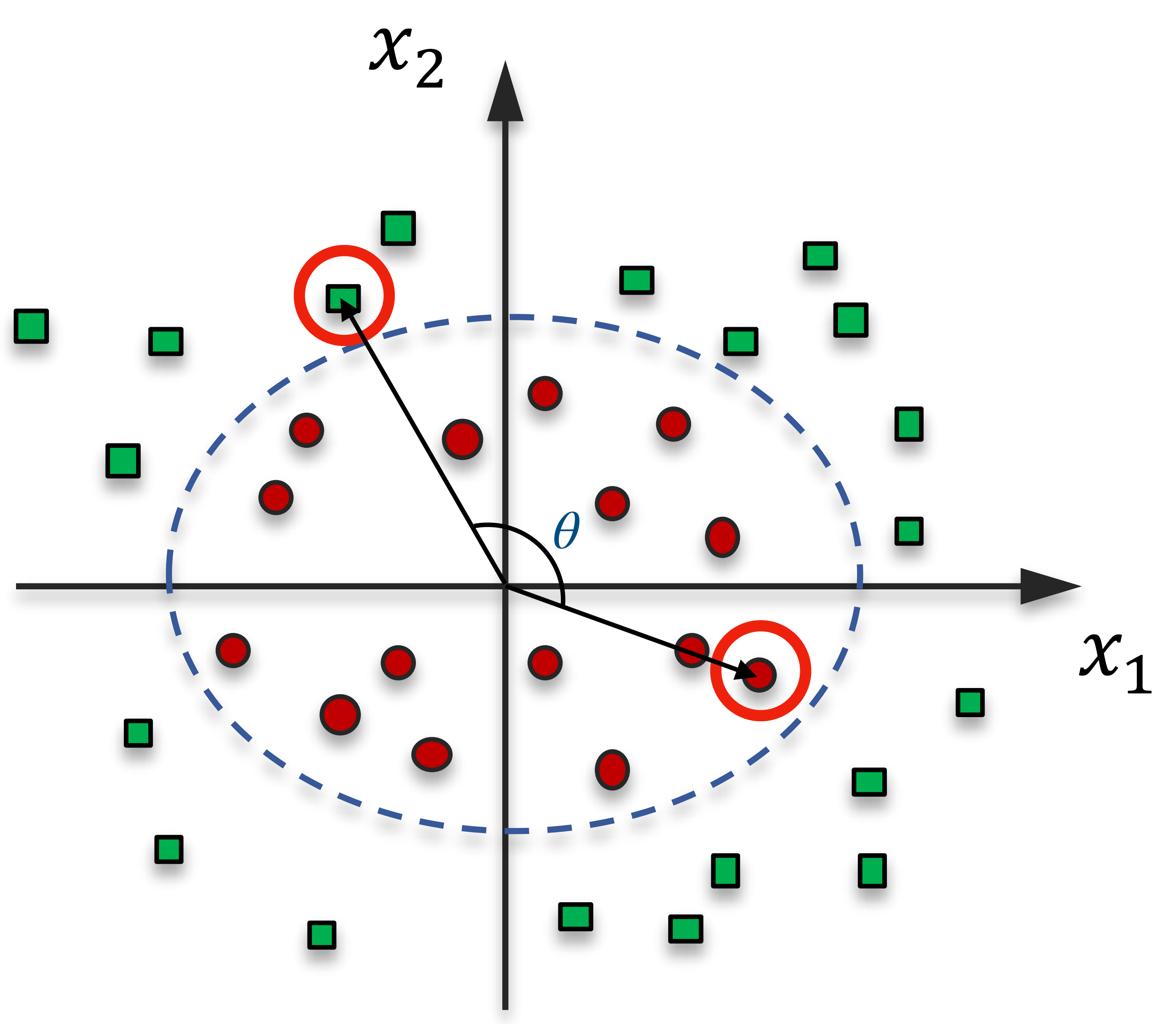
We do not need to explicitly featurize our data, but can instead work with the **inner products** of data points in potentially infinite dimensional spaces!

Kernel trick!

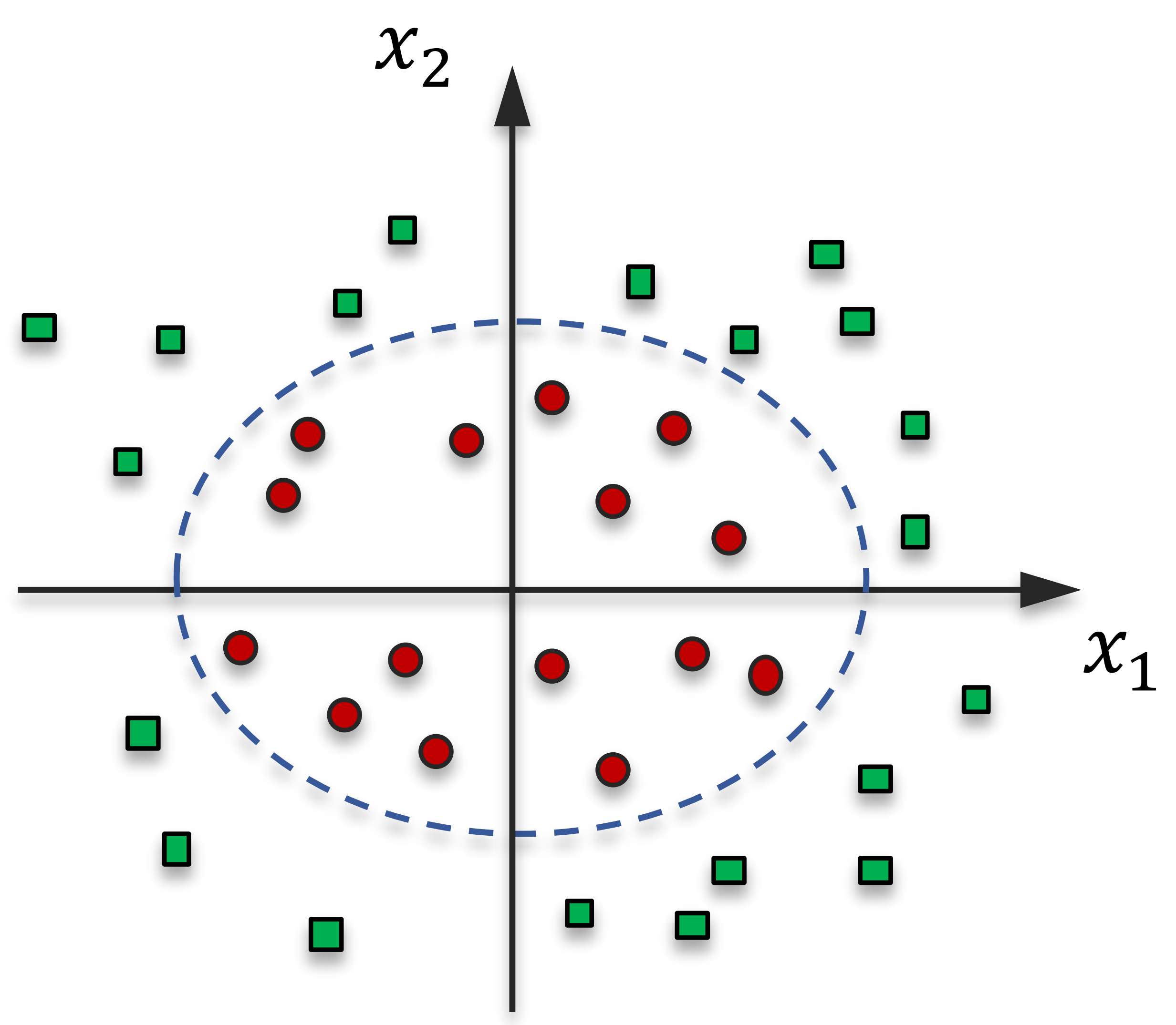
Kernels — different interpretations

- a **kernel** $k(x, z)$ is a *similarity measure* between vectors $x, z \in X$ where X is some abstract space.
- or simply a '*distance measure*' between points in feature space
- Recall the geometric interpretation of the inner-product:

$$\mathbf{a}^\top \mathbf{b} = \|\mathbf{a}\| \; \|\mathbf{b}\| \cos\theta$$

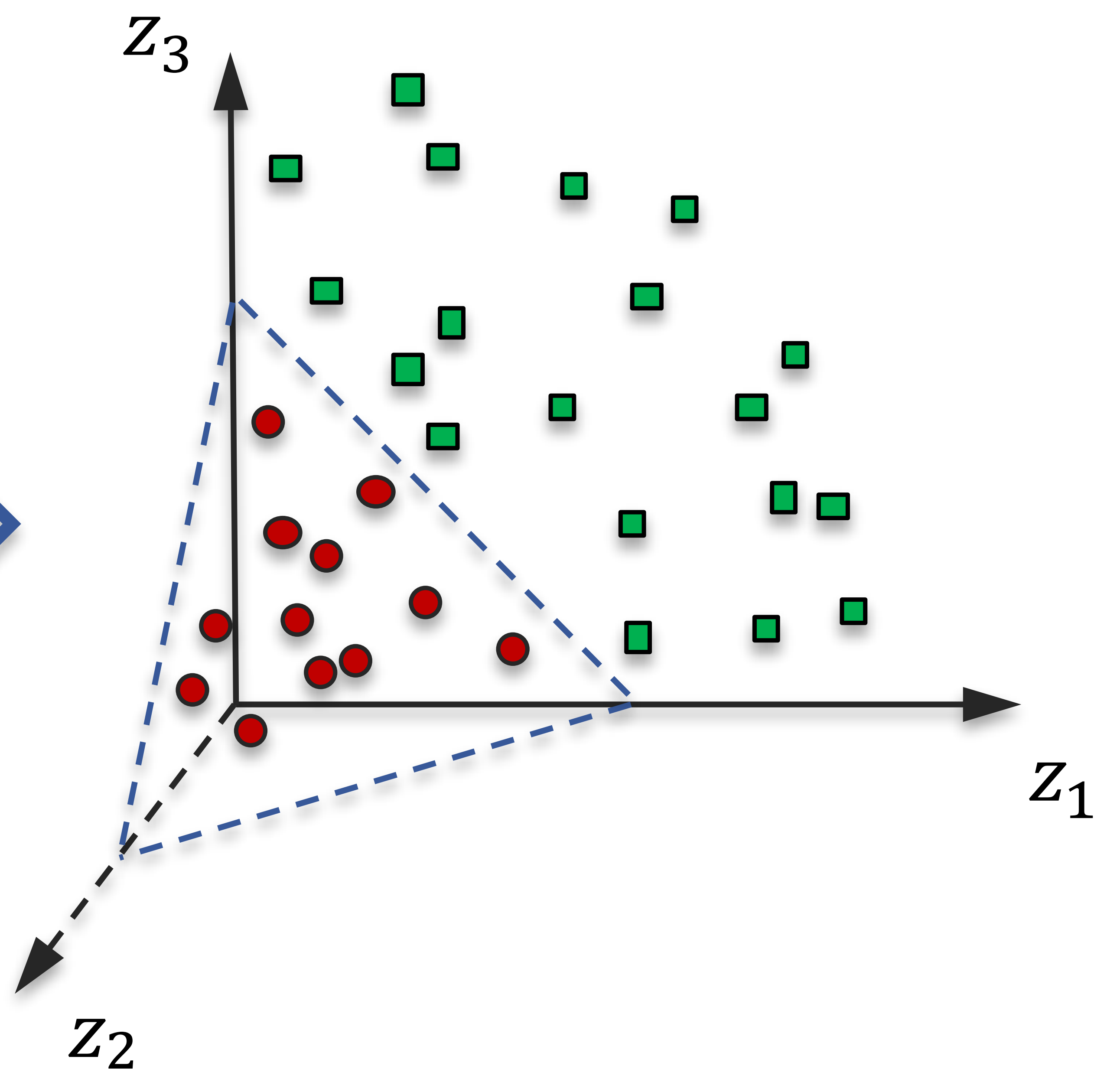


How do we know whether we have a valid Kernel?



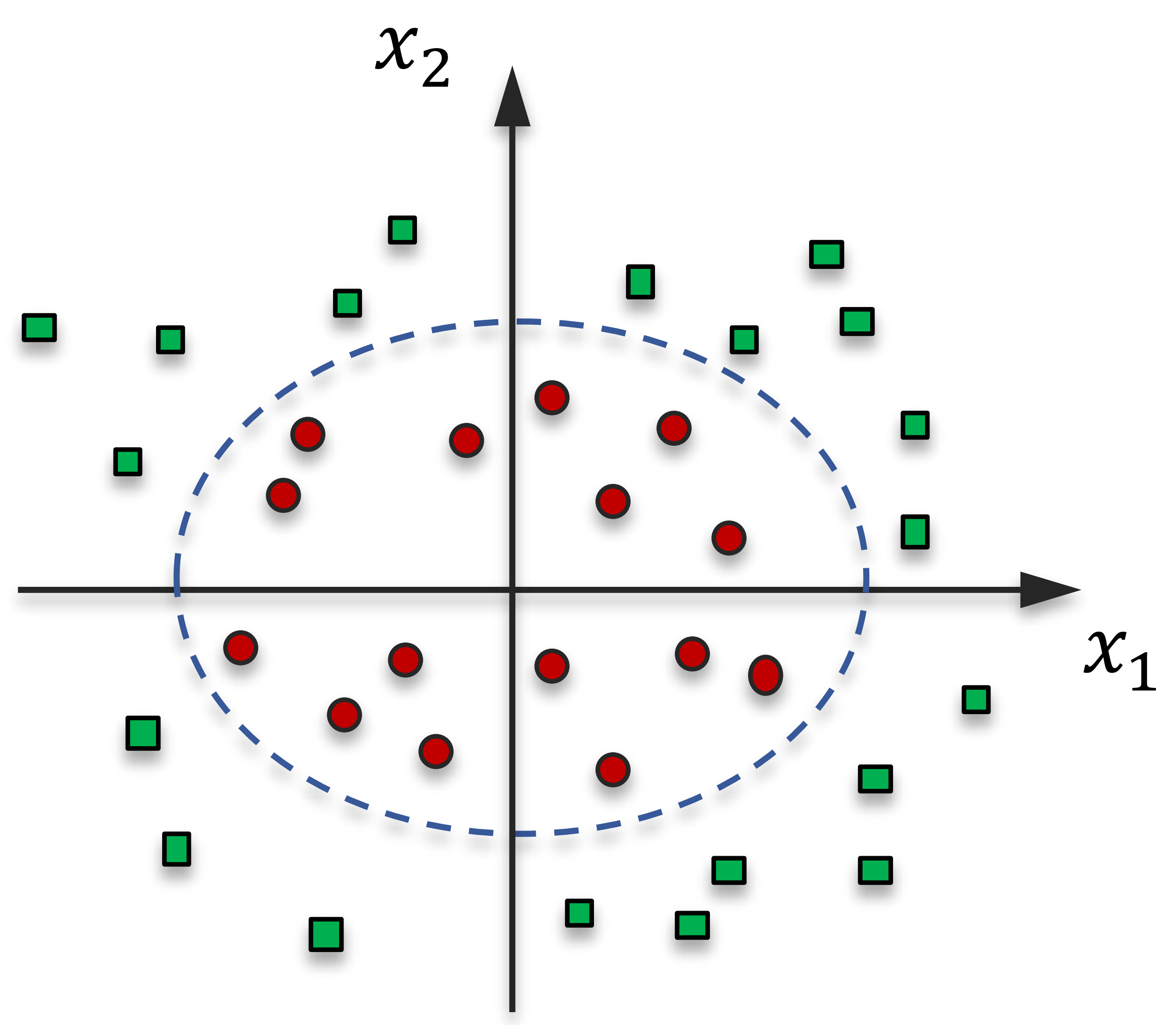
Feature map: $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$

$$\phi(x_1, x_2) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$$



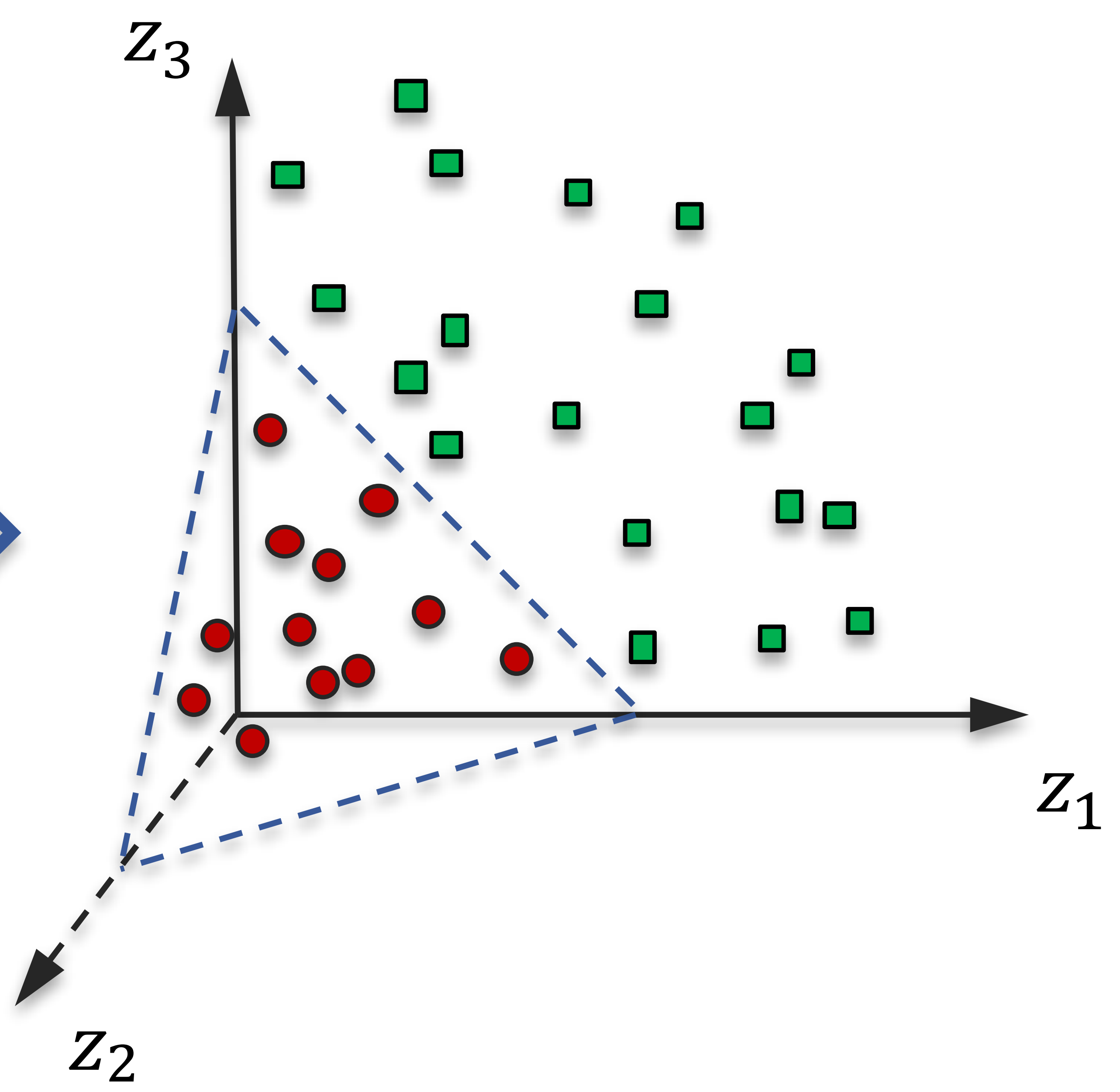
- **Is** $\phi(x_1, x_2)$ **a valid kernel?**

How do we know whether we have a valid Kernel?



Feature map: $\phi: \mathbb{R}^2 \rightarrow \mathbb{R}^3$

$$\phi(x_1, x_2) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$$

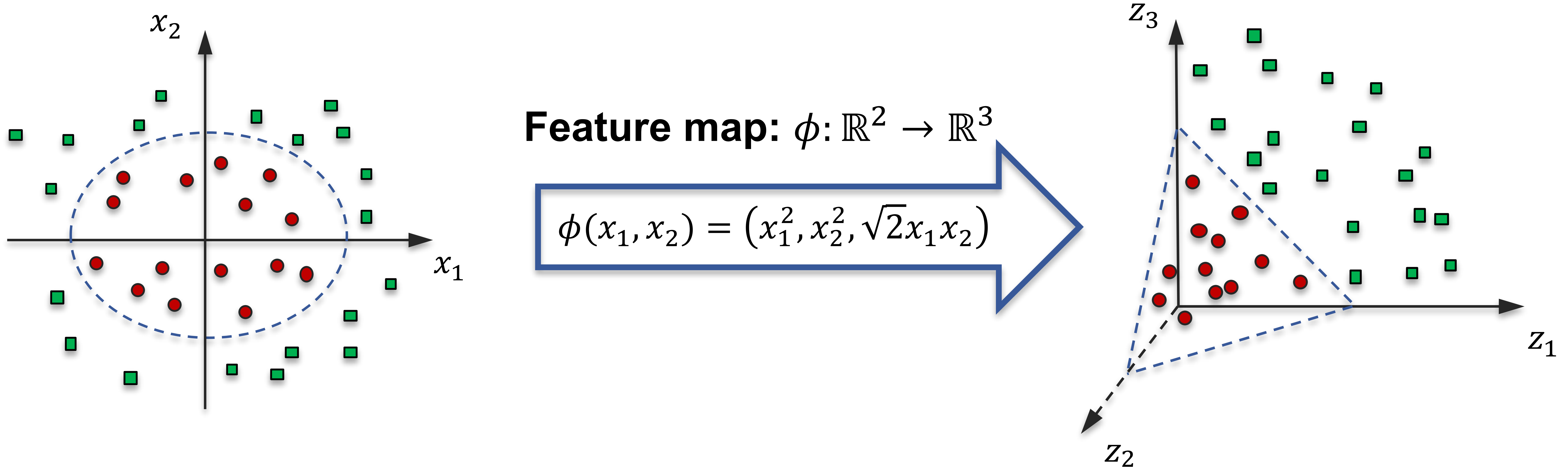


- **Is $\phi(x_1, x_2)$ a valid kernel?**

$$k(x,y) = x^\top y = (x_1y_1 + x_2y_2)^2 = x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2 = (x_1^2, \sqrt{2}x_1x_2, x_2^2)(y_1^2, \sqrt{2}y_1y_2, y_2^2)^\top$$

Poll — Is it?

How do we know whether we have a valid Kernel?



x_2
x_1
Feature map: $\phi: \mathbb{R}^2 \to \mathbb{R}^3$
$\phi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$
z_3
z_1
z_2

- **Is $\phi(x_1, x_2)$ a valid kernel?**

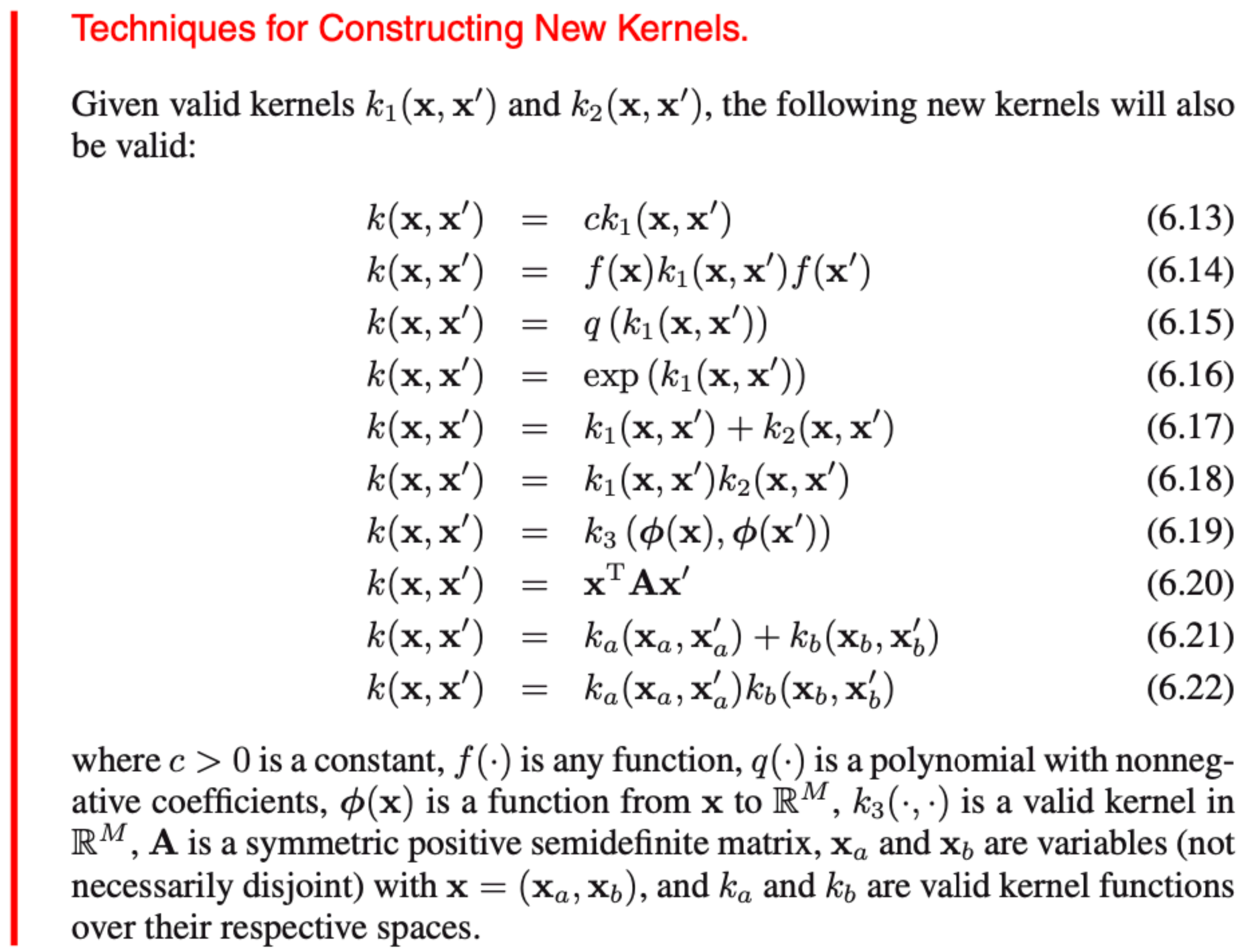
$$k(x,y) = x^\top y = (x_1y_1 + x_2y_2)^2 = x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2 = (x_1^2, \sqrt{2}x_1x_2, x_2^2)(y_1^2, \sqrt{2}y_1y_2, y_2^2)^\top$$

$\downarrow$ **Yes**

$$\phi(\mathbf{x})^\top\phi(\mathbf{y})$$

How do we know whether we have a valid Kernel?

In general …



Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$k(\mathbf{x}, \mathbf{x}') = c k_1(\mathbf{x}, \mathbf{x}') \quad (6.13)$$
$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') \quad (6.14)$$
$$k(\mathbf{x}, \mathbf{x}') = q\left(k_1(\mathbf{x}, \mathbf{x}')\right) \quad (6.15)$$
$$k(\mathbf{x}, \mathbf{x}') = \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) \quad (6.16)$$
$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \quad (6.17)$$
$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}') \quad (6.18)$$
$$k(\mathbf{x}, \mathbf{x}') = k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) \quad (6.19)$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathrm{T}} \mathbf{A} \mathbf{x}' \quad (6.20)$$
$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}_a') + k_b(\mathbf{x}_b, \mathbf{x}_b') \quad (6.21)$$
$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}_a') k_b(\mathbf{x}_b, \mathbf{x}_b') \quad (6.22)$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

From Bishop page 296

How do we choose an appropriate kernel?

- Choosing an optimal feature space is **non-trivial**
- The *kernel trick* reduces this to choosing the best kernel, and determine the corresponding feature ('implicit') mapping $\phi(x)$.
- Kernel choice influence performance of algorithm
- The best kernel depends on the specific problem
- Kernels can be applied to
  - **Numeric vectors**
  - **Strings** — For example DNA sequences or documents
  - **Graphs** — For example molecules

Common kernels — Exhibit 1: Radial basis functions

- The RBF (**radial basis function**) or *Gaussian* kernel takes the form:

$$k(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/2\sigma^2)$$

Common kernels — Exhibit 1: Radial basis functions

- The RBF (**radial basis function**) or *Gaussian* kernel takes the form:

$$k(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/2\sigma^2)$$

- Which we can show is a valid kernel:

$$||\mathbf{x} - \mathbf{y}||^2 = \mathbf{x}^\top\mathbf{x} + \mathbf{y}^\top\mathbf{y} - 2\mathbf{x}^\top\mathbf{y} \Rightarrow$$

$$k(\mathbf{x}, \mathbf{y}) = \exp(-\mathbf{x}^\top\mathbf{x}/2\sigma^2)\exp(-\mathbf{y}^\top\mathbf{y}/2\sigma^2)\exp(-\mathbf{x}^\top\mathbf{y}/2\sigma^2)$$

Common kernels — Exhibit 1: Radial basis functions

- The RBF (**radial basis function**) or *Gaussian* kernel takes the form:

$$k(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/2\sigma^2)$$

- Which we can show is a valid kernel:

$$||\mathbf{x} - \mathbf{y}||^2 = \mathbf{x}^\top\mathbf{x} + \mathbf{y}^\top\mathbf{y} - 2\mathbf{x}^\top\mathbf{y} \Rightarrow$$

$$k(\mathbf{x}, \mathbf{y}) = \exp(-\mathbf{x}^\top\mathbf{x}/2\sigma^2)\exp(-\mathbf{y}^\top\mathbf{y}/2\sigma^2)\exp(-\mathbf{x}^\top\mathbf{y}/2\sigma^2)$$

We have shown that $\mathbf{x}^\top\mathbf{y}$ is a valid kernel

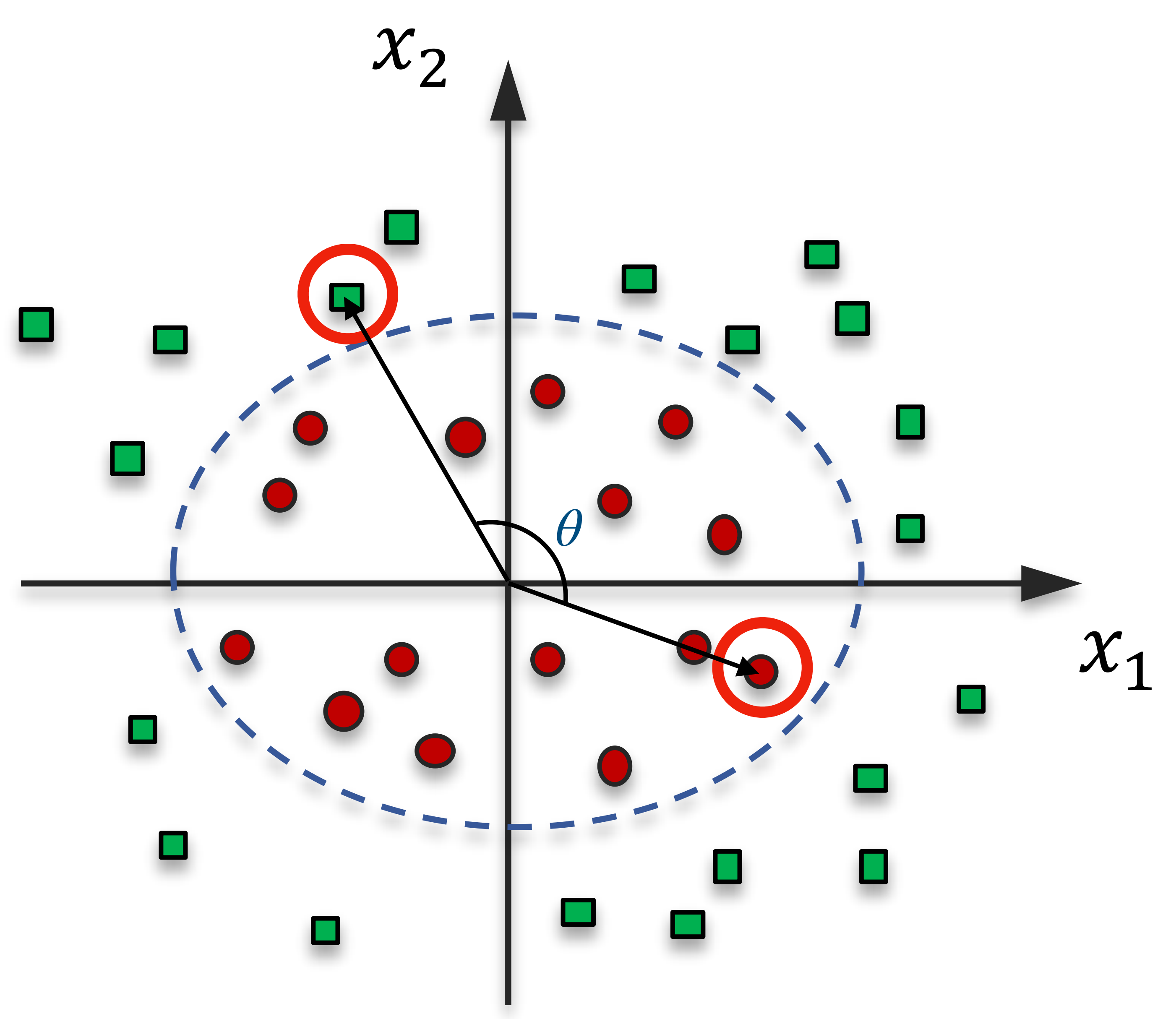
$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}'))$$

Common kernels — Exhibit 2: Cosine similarity

- If x_{ij} is the number of times **word** j occurs in **document** i we can use the *geometric interpretation* of the inner-product to compute the *cosine-similarity* between documents

$$k(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\| \ \|\mathbf{b}\|} = \cos\theta$$



Common kernels — Exhibit 3: String Kernels

- Consider two strings x and z of lengths D_x and D_z, defined on a protein alphabet
- $\mathscr{A}$={A,R,N,D,C,E, Q, G,H, I, L,K,M, F, P, S, T,W, Y, V}
- x (D_x=110):
  - IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTE EIFQGIGTLESQTVQGGTVERLFKNLSLIKKYIDGQKKKC GEERRRVNQFLDYLQEFLGVMNTEWI
- z (D_z=153):
  - PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTE AERLQENLQAYRTFHVLLARLLEDQQVHFTPTEGDFHQAI HTLLLQVAAFAYQIEELMILLEYKIPRNEADGMLFEKKLWG LKVLQELSQWTVRSIHDLRFISSHQTGIP
-

Similarity measure: number of common substrings

$$k(\mathbf{x}, \mathbf{z}) = \sum_{s \in \mathcal{A}^*} w_s \phi_s(\mathbf{x}) \phi_s(\mathbf{z})$$

where s is a substring, $w_s \geq 0$ and $\mathcal{A}^*$ the set of all substrings from $\mathcal{A}$.

Common kernels — Exhibit 4: Matérn Kernel

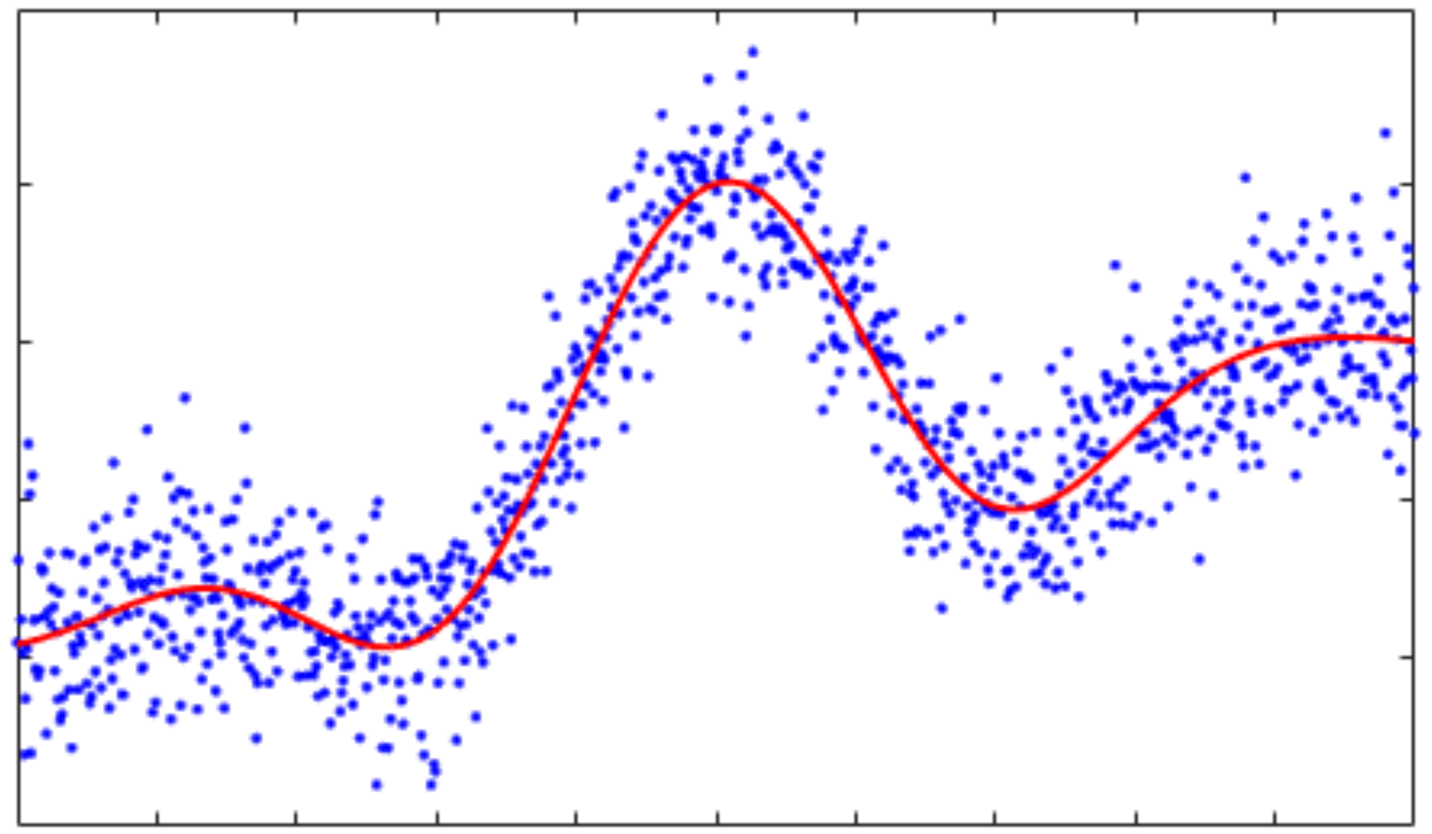
- Matérn Kernel is a popular choice for *Gaussian process* regression

$$k(x,y) = \frac{2^{1-\nu}}{\Gamma(\nu)}\Big(\frac{\sqrt{2\nu}\,|d|}{\ell}\Big)^{\nu} B_\nu\Big(\frac{\sqrt{2\nu}\,|d|}{\ell}\Big)$$

- where $d = ||x-y||$, $\nu \geq 0,\ l > 0$ and B_ν is a modified *Bessel function.*

Special cases:

$$k(x,y) = \exp(-d/\ell) \text{ for } \nu = 1/2$$
$$k(x,y) = \exp(-d^2/\ell) \text{ for } \nu \to \infty$$



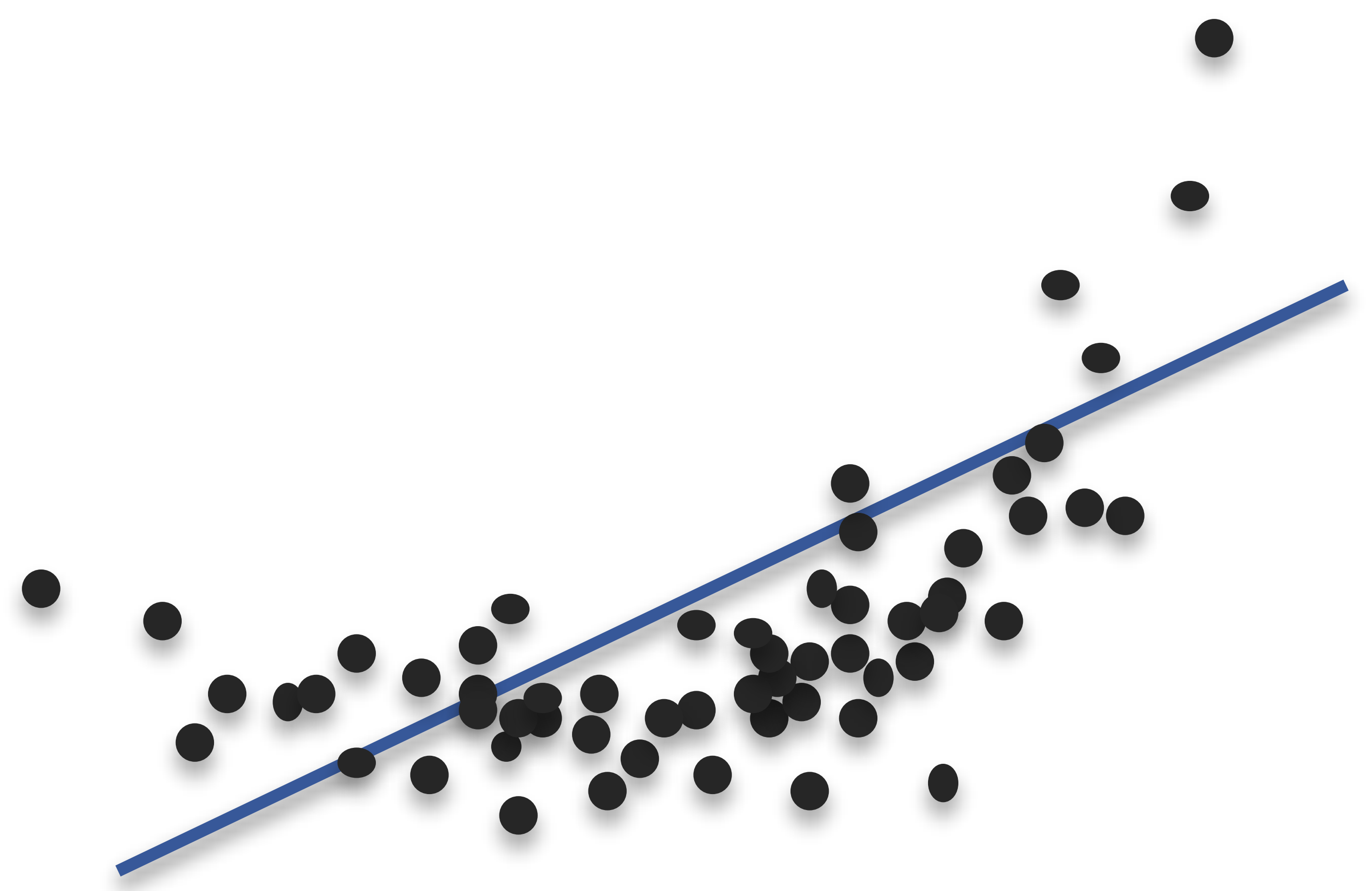
Bertil Matérn
Born in Gothenburg 1917

Gaussian processes

- **Linear regression:** determine a relation f between response y and independent variable x

$$y = f(x) + \epsilon$$
$$f(x) = \beta_0 + \beta_1 x$$

- **Bayesian linear regression:** determine a **posterior distribution** over the unobserved variables β_0, β_1 and update it as new data becomes available.
- **Gaussian processes regression:** finds a **posterior distribution** over the possible functions $f(x)$ consistent with the observed data and a suitable **prior**.

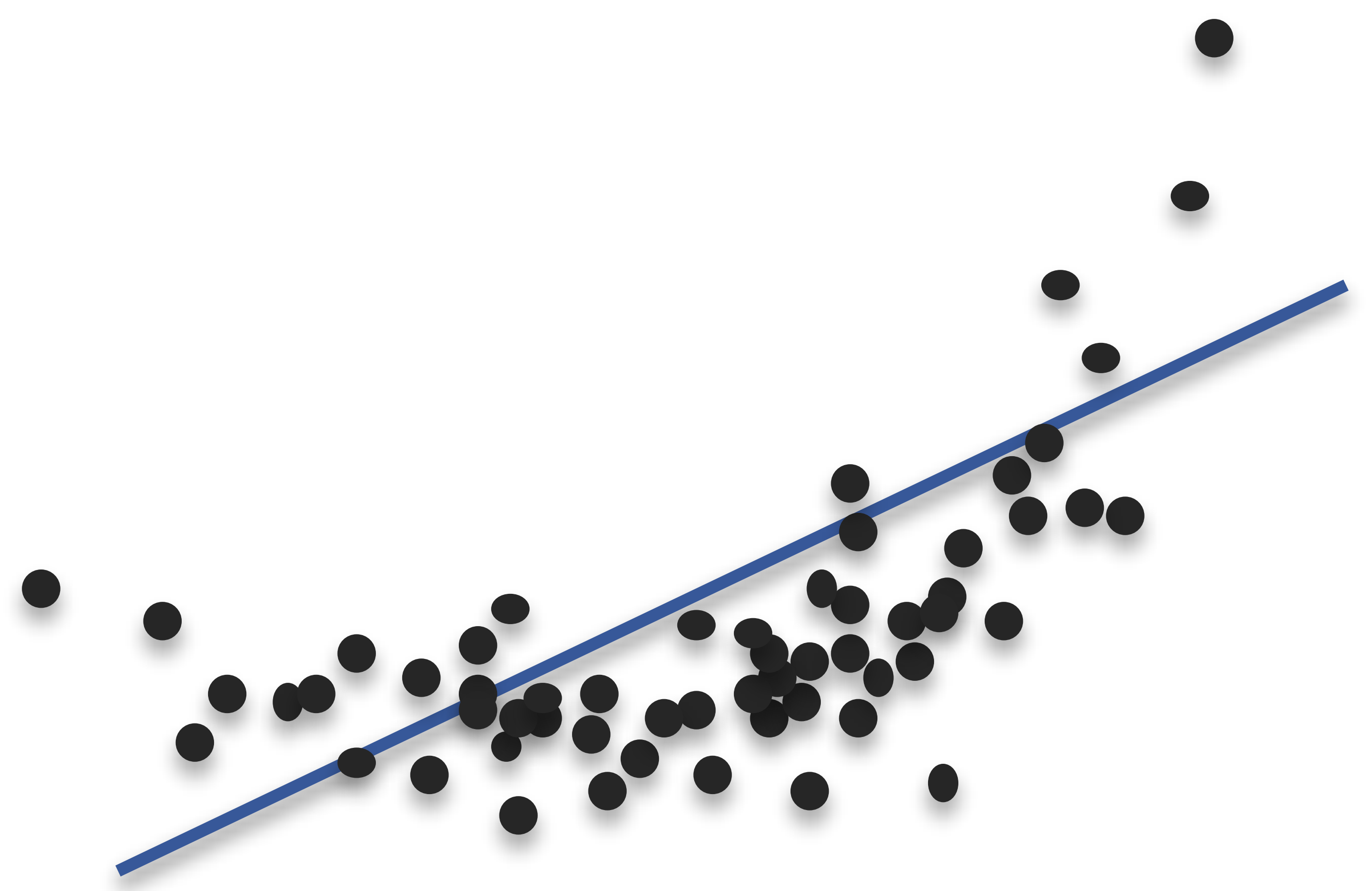


Gaussian processes — A non-parametric approach

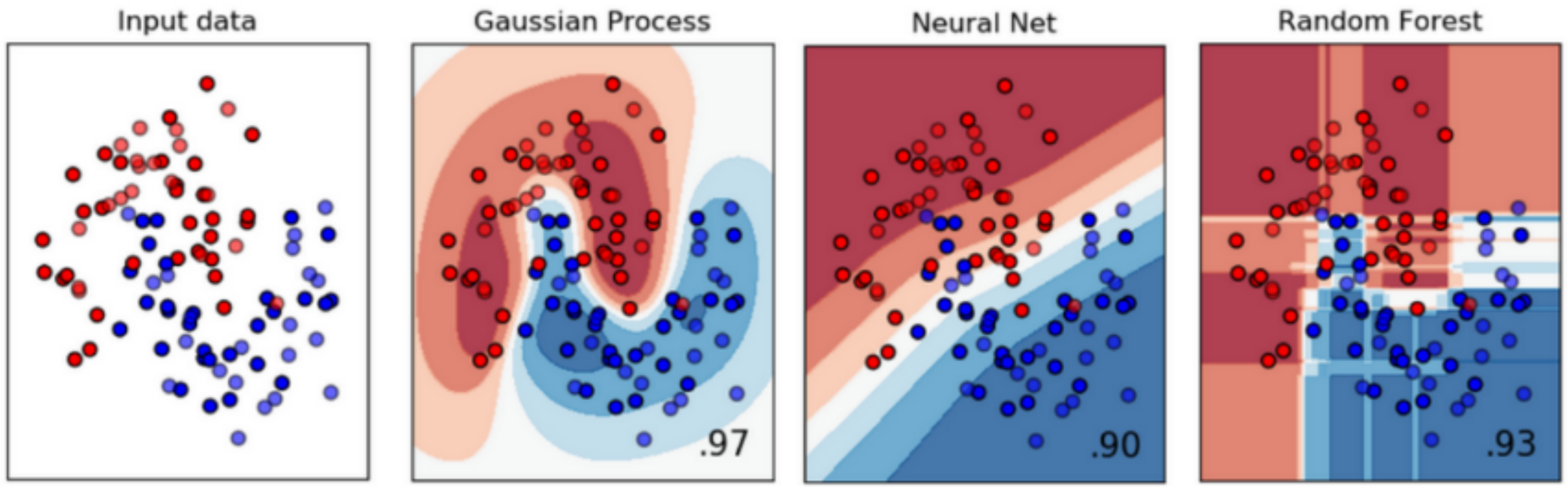
- The current example does **not look linear —** Maybe a **quadratic** function is better?

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2$$

- But then we have **three** <u>unknown parameters</u> $\beta_0, \beta_1, \beta_2$
- But how do we decide *how* many parameters to use? — and *how* do we decide on the <u>parametric form</u> of the function?
- Instead of searching for suitable parameter values for a fixed number of parameters (and a fixed function), we want to search among **all functions** that fit our data.

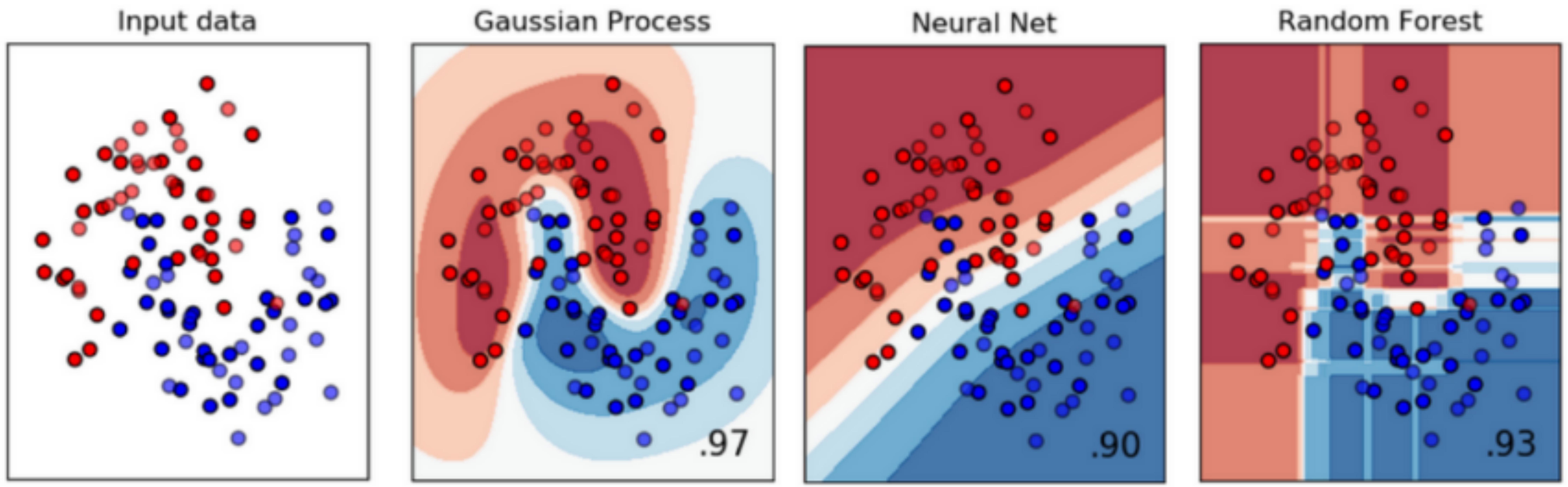


Gaussian processes — classifier comparison



From SK Learn documentation

Gaussian processes — classifier comparison

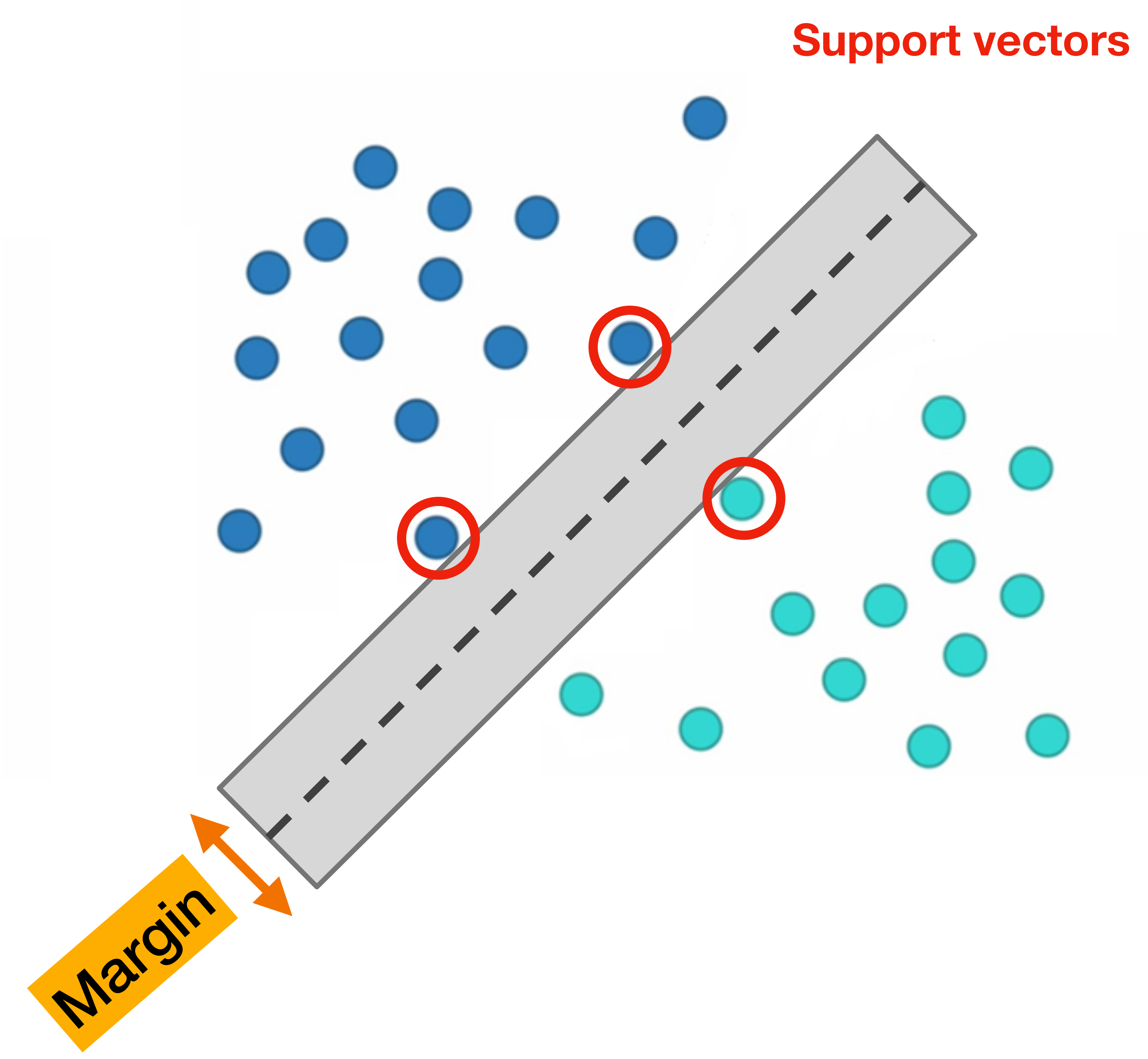


Look at:
Bishop Chapter 6.4
For more details
Dedicated free textbook from CE Rasmussen:
http://www.gaussianprocess.org/gpml/chapters/RW.pdf

From SK Learn documentation

Support vector machines are Kernel methods

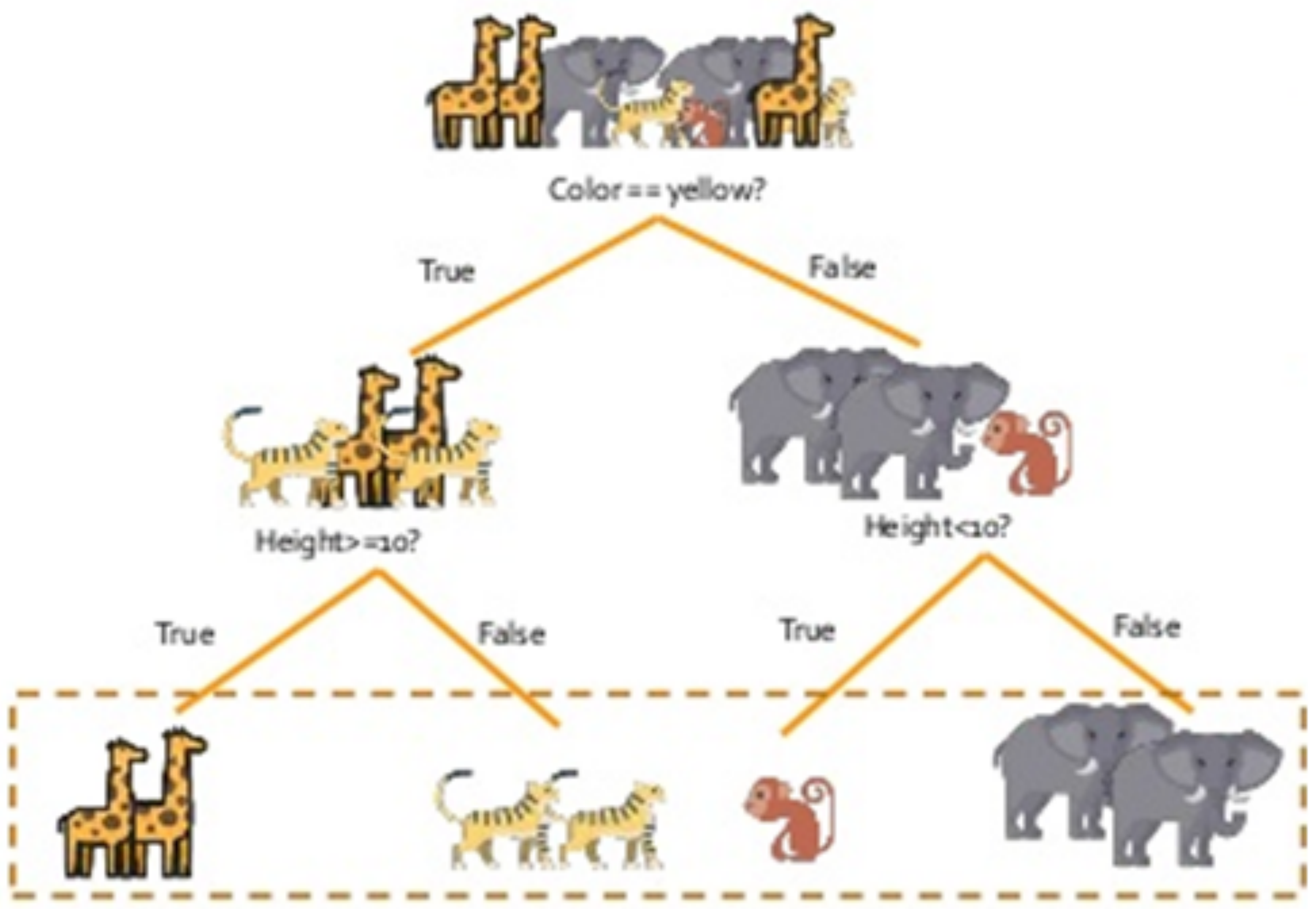
- **Advantages**
  - Works for classification as well as regression
  - Works with high-dimensional space
  - Works for two or more classes (via one-vs-rest strategy)
  - Good accuracy
- **Disadvantages**
  - Slow on large datasets (compared to Naïve Bayes)
  - Works poorly with overlapping classes
  - Kernel type must be selected manually.



Decision trees

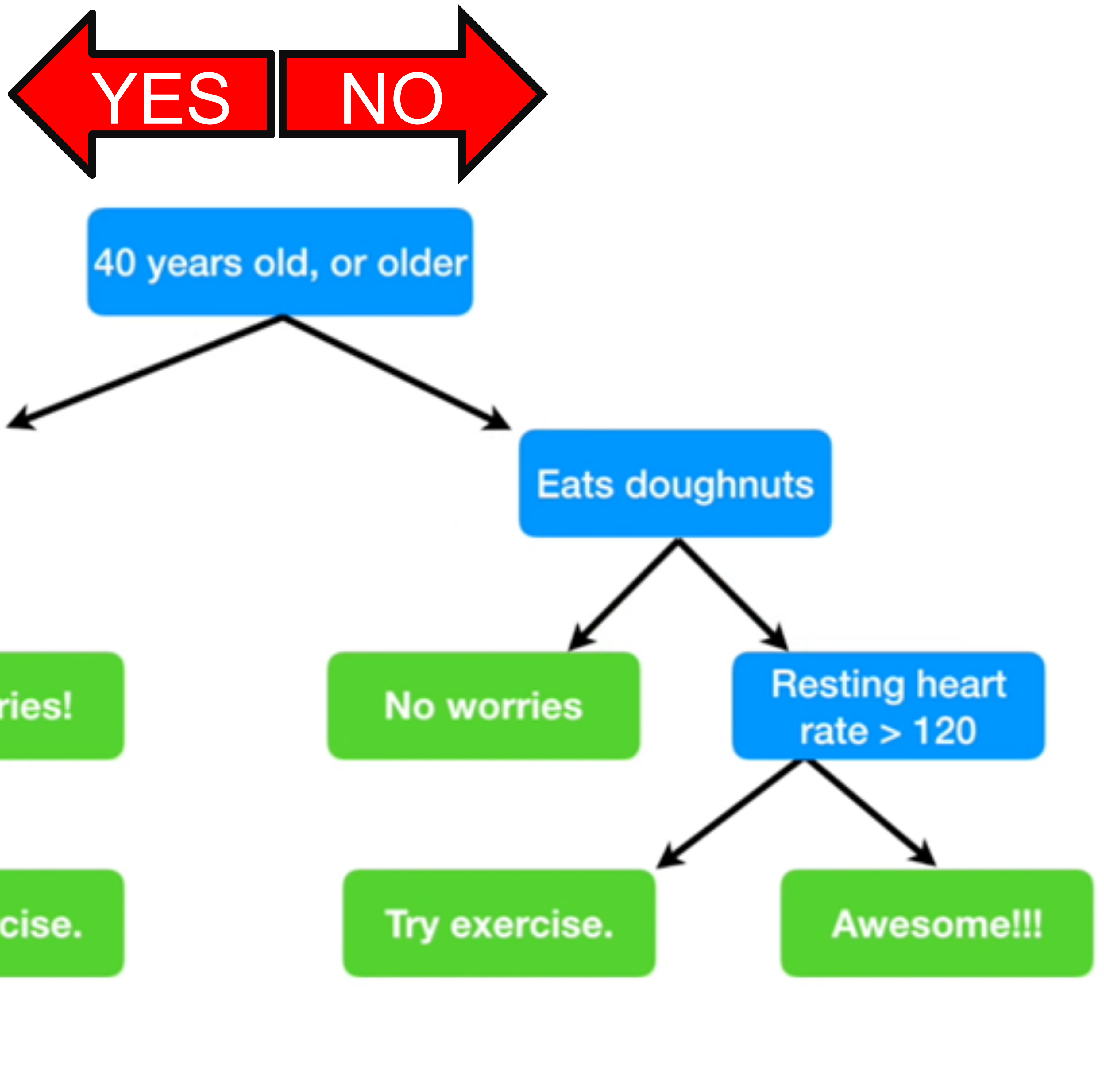
- **Decision trees** is a class of regression and classification methods. They are called **trees** as they can be interpreted and visualised using a tree structured graph,
  - Each **node** represents a question/problem
  - Each **branch** represents an answer/decision
  - Each **leaf** represents a class label/response prediction

Classification of Animals



(Questionable) health advice

Here's a more complicated decision tree…



From data to decision tree

Features Response/Class

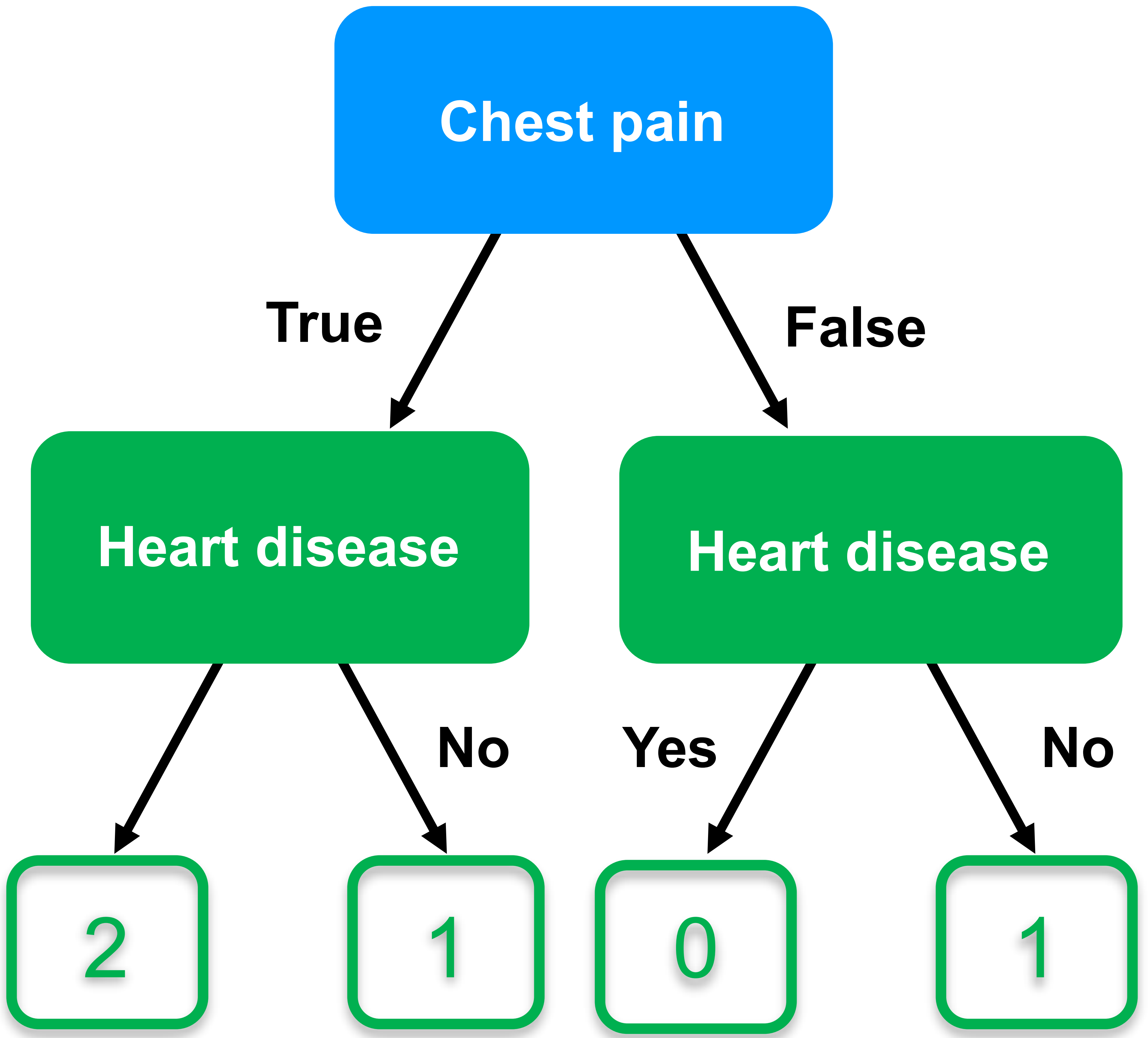
Chest pain	Diastolic blood pressure	Systolic blood pressure	Blocked arteries	Heart Disease
No	90	140	No	No
Yes	150	190	Yes	Yes
Yes	45	120	No	No
Yes	80	130	???	Yes
etc...	etc...	etc...	etc...	etc...

From data to decision tree

- **Decision trees** can be built in many ways
- Ask **Q1** at the root, **Q2** everywhere at the next level and so on.
  - But in this way the tree will be big $\approx 2^n$ for n features/questions.
- **Regularities** in data can allow the trees to be smaller.

Chest pain	Diastolic blood pressur	Systolic blood	Blocked arterie	Heart Disease
No	90	140	No	No
Yes	150	190	Yes	Yes
Yes	45	120	No	No
Yes	80	130	???	Yes
etc...	etc...	etc...	etc...	etc...

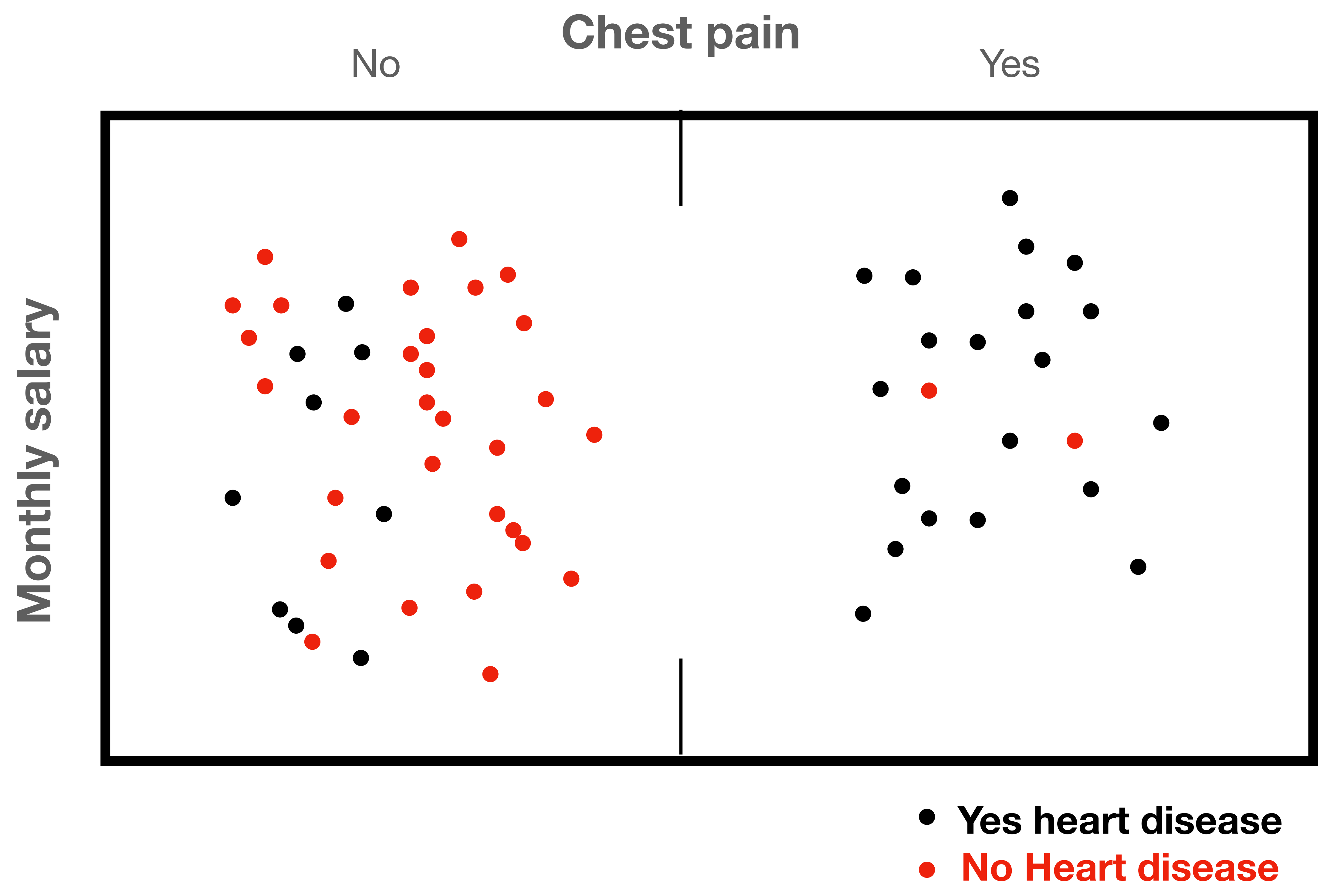
Feature analysis



Chest pain	Diastolic blood pressur	Systolic blood	Blocked arterie	Heart Disease
No	90	140	No	No
Yes	150	190	Yes	Yes
Yes	45	120	No	No
Yes	80	130	???	Yes
etc...	etc...	etc...	etc...	etc...

Area impurity

- Every time we ask a **question** (add a branch to the tree) we split off a new area in our feature space.
- The more homogenous the **response/ class** is in an area the less "impure" it is.
- **Less impurity** should mean less variation within response/class — That means better predictions!



Measures of impurity

- **Gini impurity**

$$I_G(p) = \sum_{i=1}^{J} p_i(1-p_i)$$

- **Information Gain**

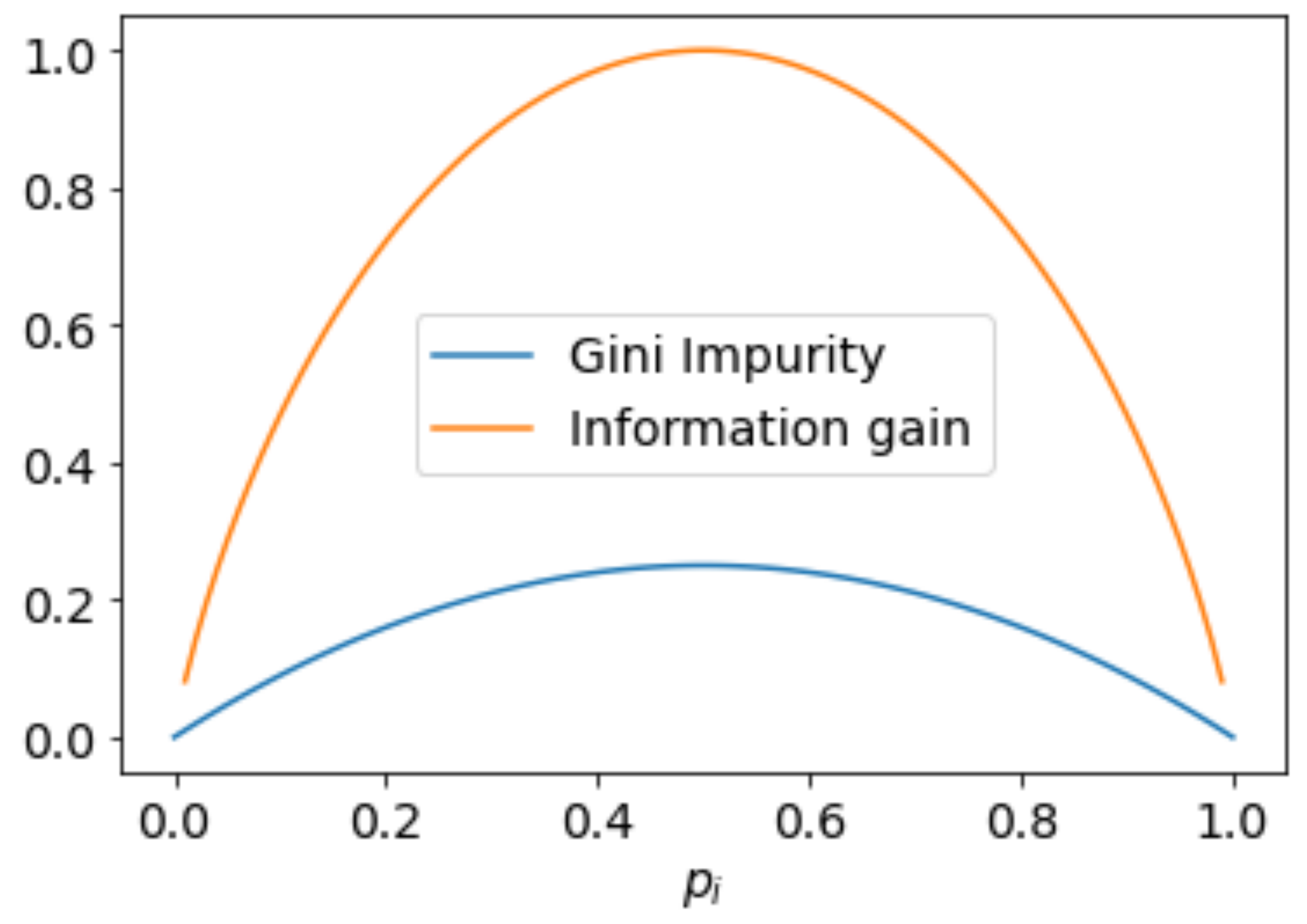
$$H(p) = -\sum_{i=1}^{J} p_i \log_2 p_i$$

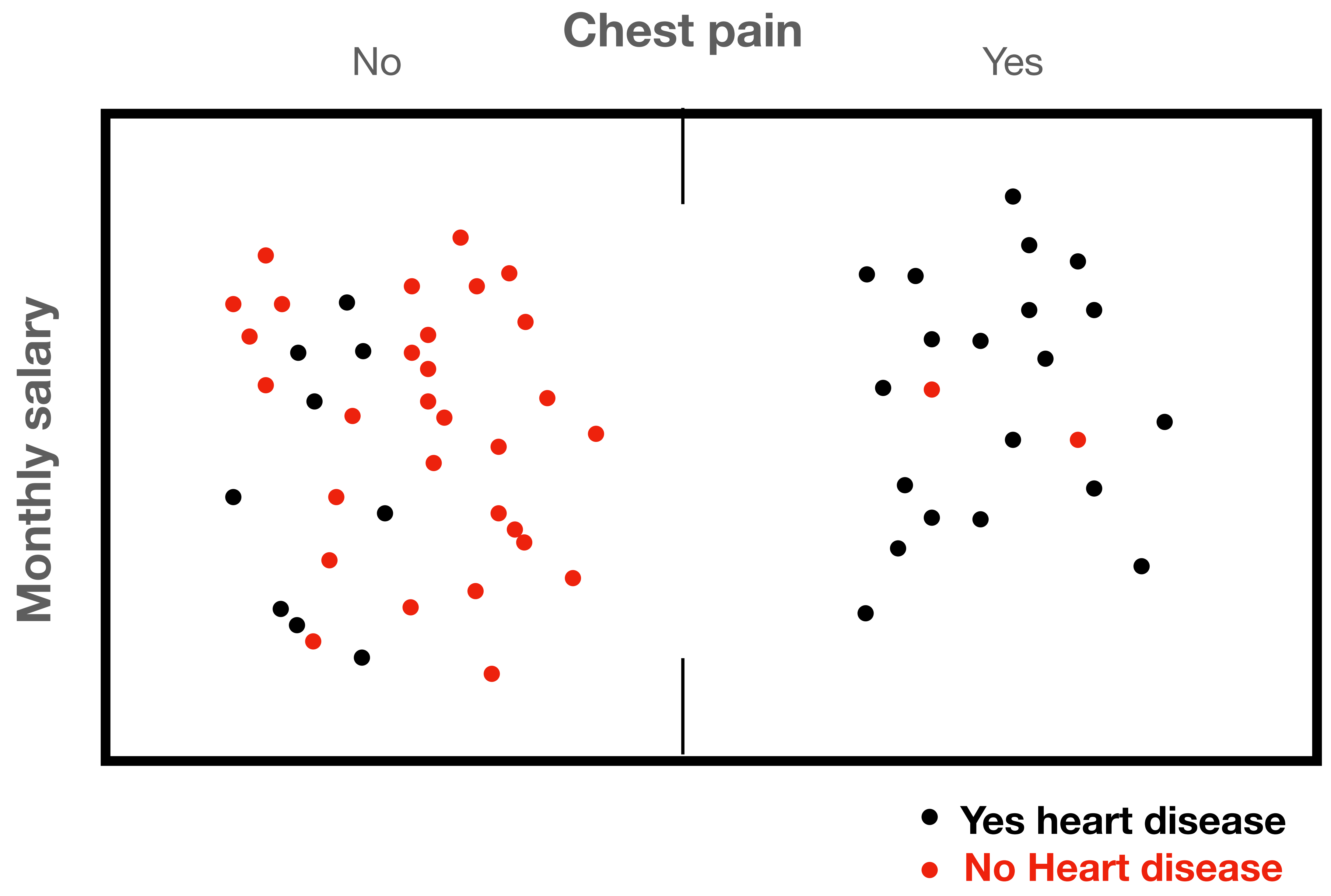
Where we have J classes (fx. Yes/No, Apple/Orange/Banana) p_i is the fraction assigned to class i (fx. Yes) in the area.

- **Variance reduction** (for regression, y is a continuous response)

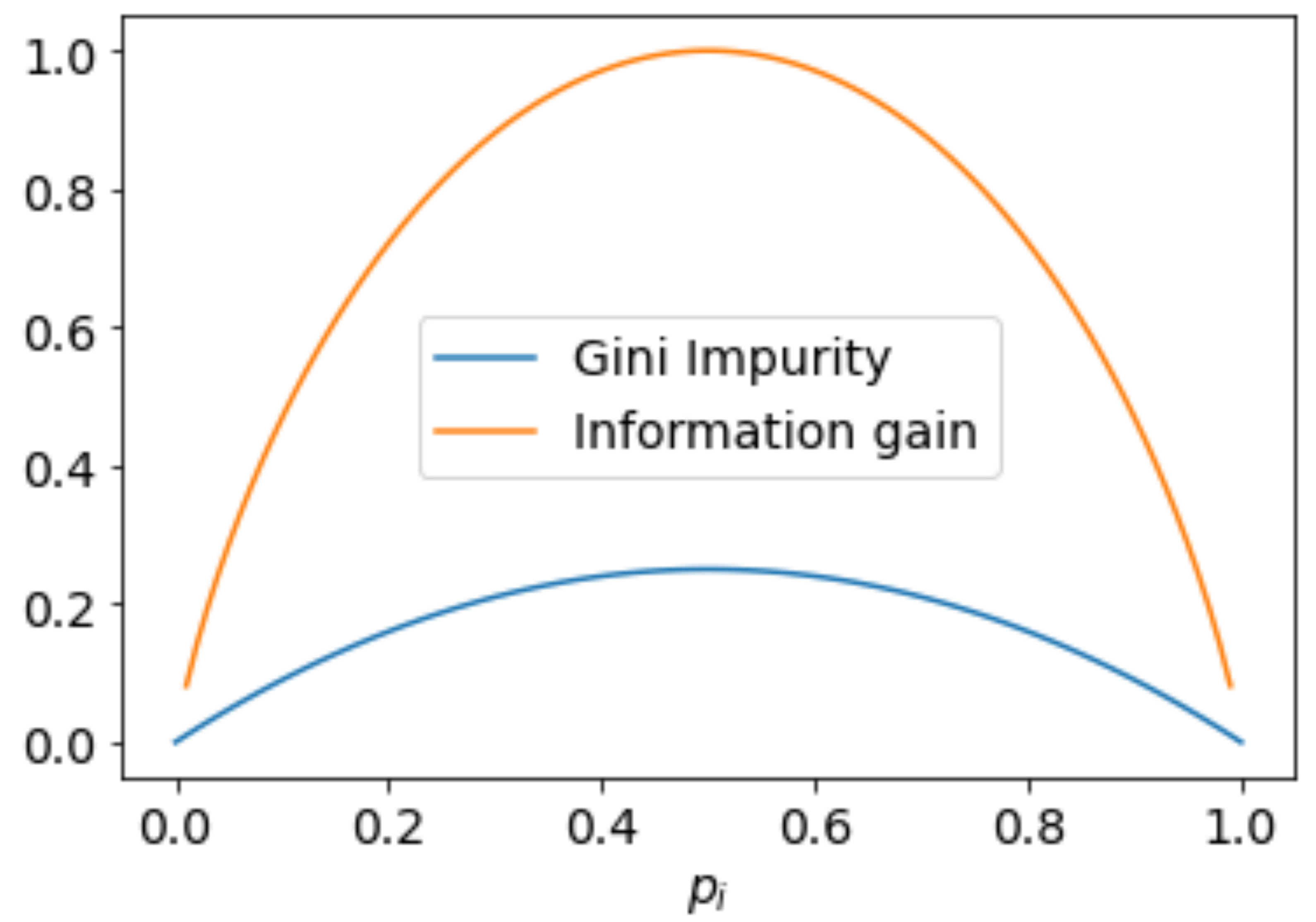
$$I_V(N) = \frac{1}{|S|^2}\sum_{i\in S}\sum_{j\in S}\frac{1}{2}(y_i-y_j)^2 - \left(\frac{1}{|S_+|^2}\sum_{i\in S_+}\sum_{j\in S_+}\frac{1}{2}(y_i-y_j)^2 + \frac{1}{|S_-|^2}\sum_{i\in S_-}\sum_{j\in S_-}\frac{1}{2}(y_i-y_j)^2\right)$$

S, S_+, S_- are the set of pre-split sample indices, set of sample indices for which the split test is **true**, and set of sample indices for which the split test is **false**.





Measures of impurity



- **Gini impurity**

$$I_G(p) = \sum_{i=1}^{J} p_i(1 - p_i)$$

- **Information Gain**

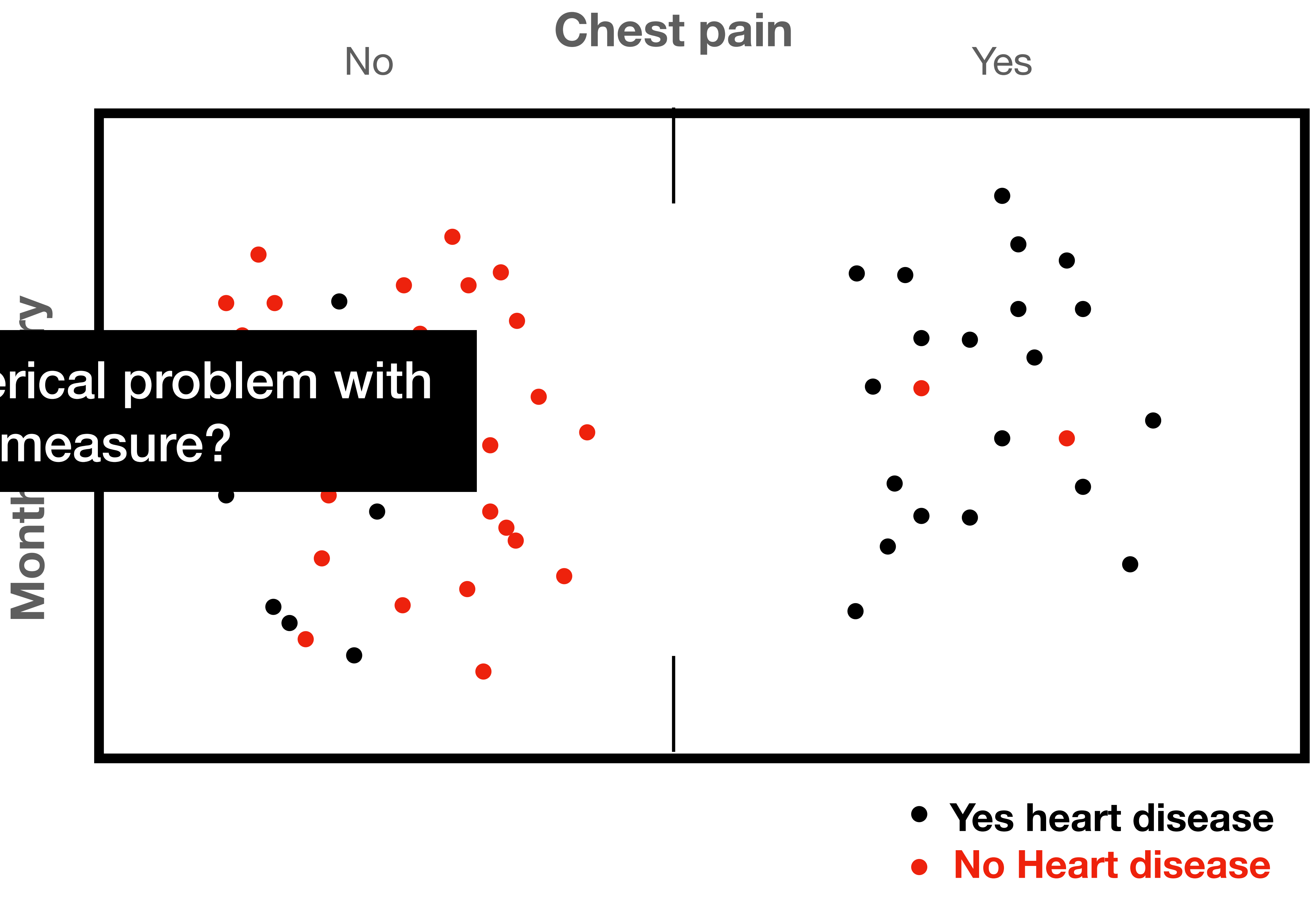
$$H(p) = -\sum_{i=1}^{J} p_i \log_2 p_i$$

Where we have J classes (fx. Yes/No, Apple/Orange/Banana) p_i is the fraction assigned to class i (fx. Yes) in the area.

- **Variance reduction** (for regression, y is

$$I_V(N) = \frac{1}{|S|^2}\sum_{i\in S}\sum_{j\in S}\frac{1}{2}(y_i - y_j)^2 - \left(\frac{1}{|S_+|^2}\sum_{i\in S_+}\sum_{j\in S_+}\frac{1}{2}(y_i - y_j)^2 + \frac{1}{|S_-|^2}\sum_{i\in S_-}\sum_{j\in S_-}\frac{1}{2}(y_i - y_j)^2\right)$$

S, S_+, S_- are the set of pre-split sample indices, set of sample indices for which the split test is **true**, and set of sample indices for which the split test is **false**.



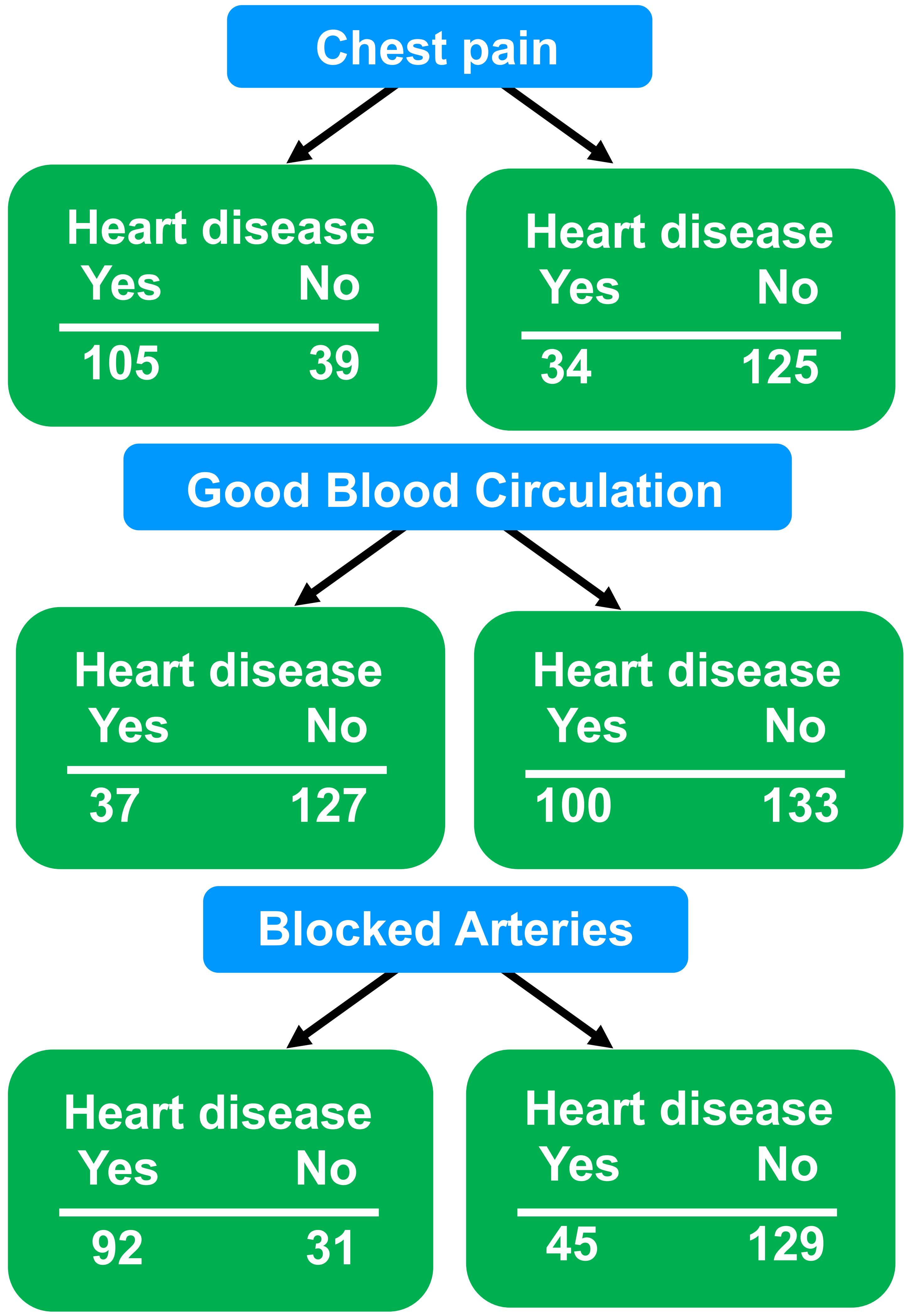
Poll: what is a potential numerical problem with the Information gain measure?

Building trees

Gini impurity for chest pain = 0.364

Gini impurity for Good Blood Circulation = 0.360

Gini impurity for Blocked Arteries = 0.381

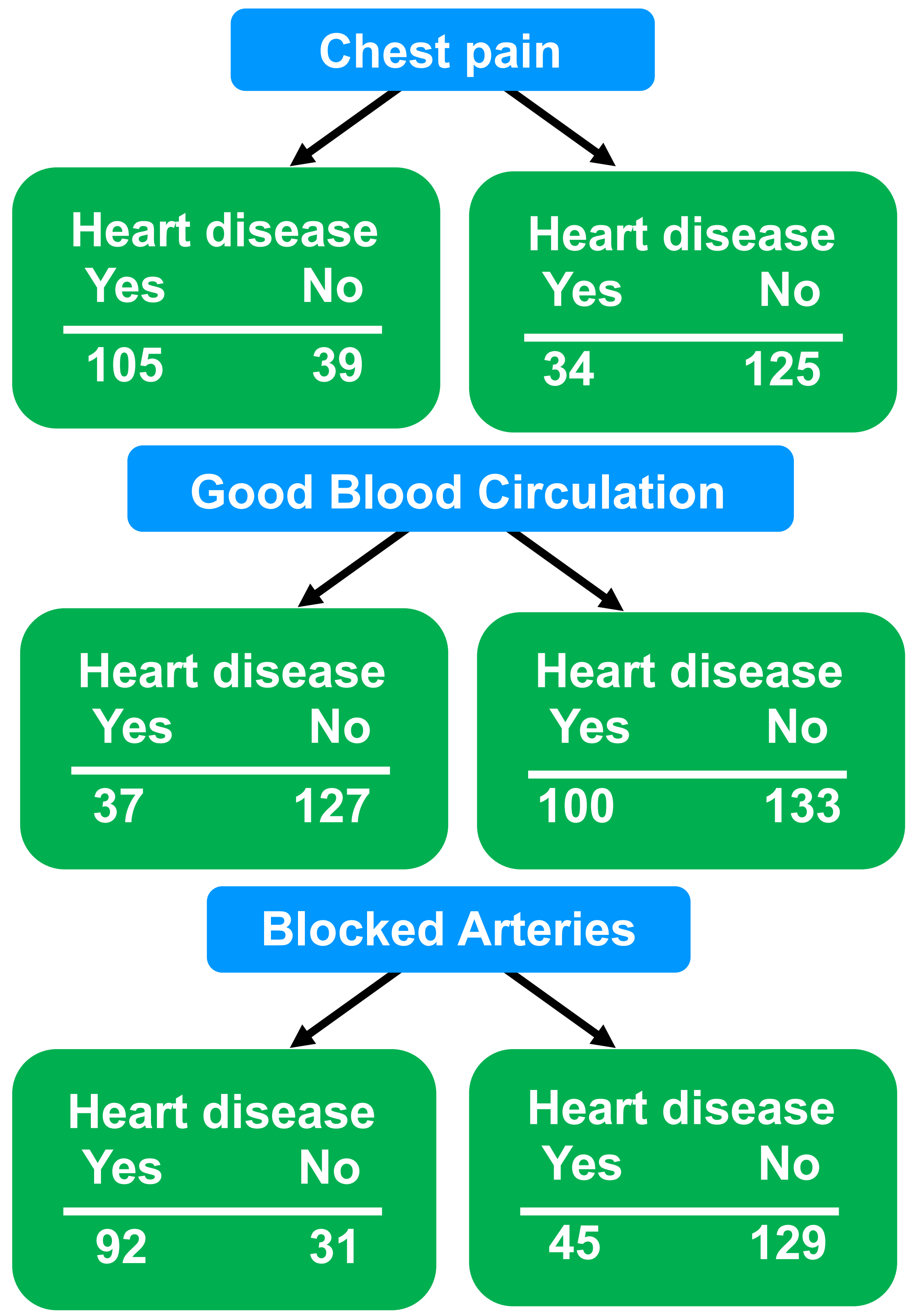


Building trees

Gini impurity for chest pain = 0.364

Gini impurity for Good Blood Circulation = 0.360

Gini impurity for Blocked Arteries = 0.381


Chest pain
Heart disease Yes No 105 39
Heart disease Yes No 34 125
Good Blood Circulation
Heart disease Yes No 37 127
Heart disease Yes No 100 133
Blocked Arteries
Heart disease Yes No 92 31
Heart disease Yes No 45 129

Repeat splitting in descending order until a pre-selected number of splits is reached.
Do not split variables that do not improve purity/variance scores.

Building trees — splitting continuous variables

Chest pain	Diastolic blood pressure	Systolic blood pressure	Blocked arteries	Heart Disease
No	90	140	No	No
Yes	150	190	Yes	Yes
Yes				No
Yes	80	130	???	Yes
etc...	etc...	etc...	etc...	etc...

Poll: How can we split continuous variables?
They do not have a discrete class

Decision trees — pros and cons

Decision trees are
easy to build,
easy to use and
easy to interpret

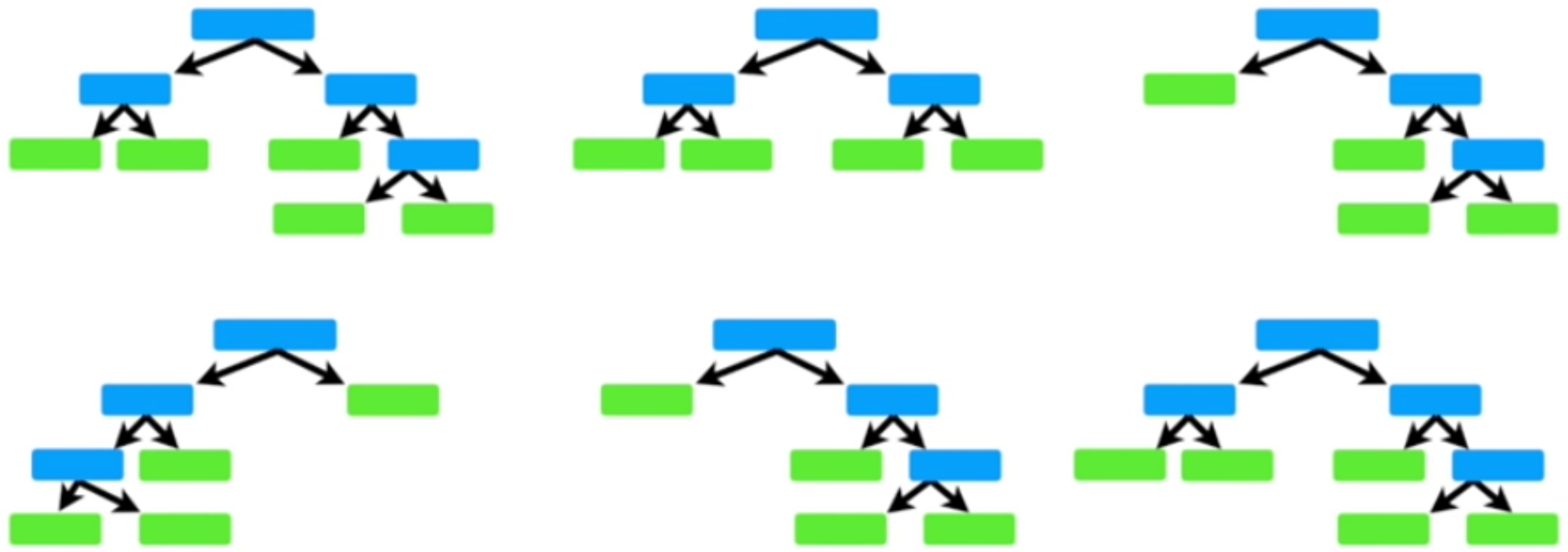
Decision trees — pros and cons

- ▲ Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- ▲ Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.
- ▲ Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- ▲ Trees can easily handle qualitative predictors without the need to create dummy variables.
- ▼ Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.
- ▼ Additionally, trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree.

Taken from **An Introduction to Statistical Learning**
(James, Witten, Hastie, and Tibshirani)

Random Forests — an *ensemble* method

The good news is that **Random Forests** combine the simplicity of decision trees with flexibility resulting in a vast improvement in accuracy.



We build many trees (a forest) and **have a majority vote**

Random Forests — an *ensemble* method

Chest pain	Diastolic blood pressure	Systolic blood pressure	Blocked arteries	Heart Disease
No	90	140	No	No
Yes	150	190	Yes	Yes
Yes	45	120	No	No
Yes	80	130	???	Yes
etc...	etc...	etc...	etc...	etc...

Same dataset

Random Forests — an *ensemble* method

Chest pain	Diastolic blood pressure	Systolic blood pressure	Blocked arteries	Heart Disease
No	90	140	No	No
Yes	150	190	Yes	Yes
Yes	45	120	No	No
Yes	80	130	???	Yes
etc...	etc...	etc...	etc...	etc...

Bootstrap

Chest pain	Diastolic blood pressure	Systolic blood pressure	Blocked arteries	Heart Disease
No	90	140	No	No
Yes	180	220	Yes	Yes
No	89	120	Yes	Yes
Yes	80	130	???	Yes

Sub-sample some random patients

Random Forests — an *ensemble* method

Bootstrapped data set

Chest pain	Diastolic blood pressure	Systolic blood pressure	Blocked arteries	Heart Disease
No	90	140	No	No
Yes	180	220	Yes	Yes
No	89	120	Yes	Yes
Yes	80	130	???	Yes

Sub-sample some random patients

Select Random Feature Subset →

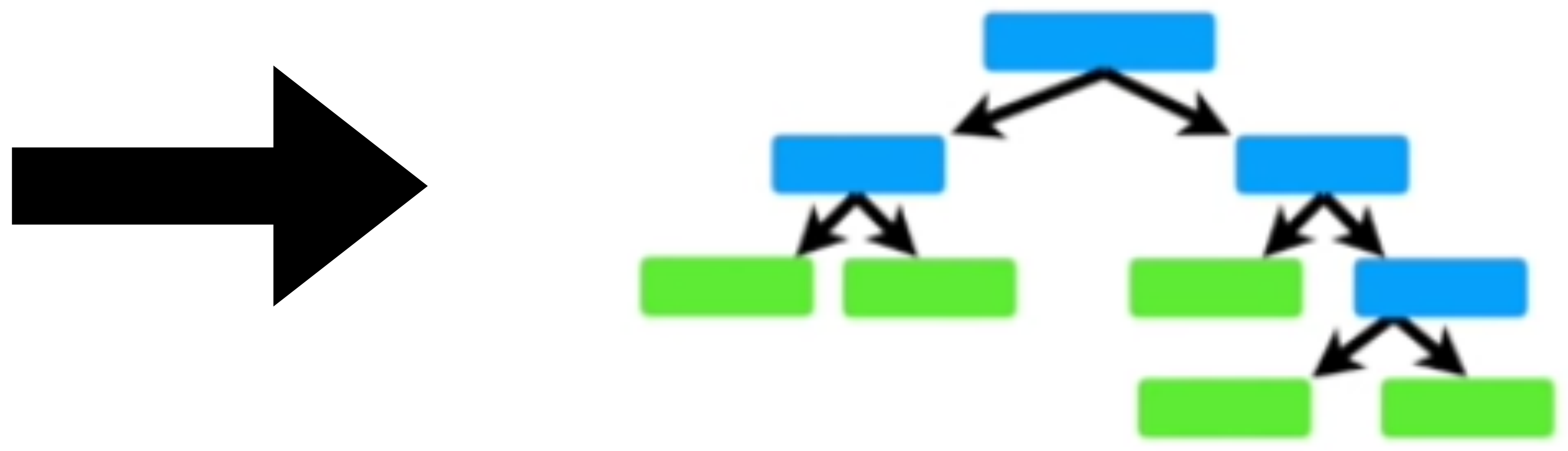
Chest pain	Diastolic blood pressure	Systolic blood pressure	Blocked arteries	Heart Disease
No	90	140	No	No
Yes	180	220	Yes	Yes
No	89	120	Yes	Yes
Yes	80	130	???	Yes

Random Forests — an *ensemble* method

Bootstrapped data set on selected features

Chest pain	Diastolic blood pressure	Systolic blood pressure	Blocked arteries	Heart Disease
No	90	140	No	No
Yes	180	220	Yes	Yes
No	89	120	Yes	Yes
Yes	80	130	???	Yes

Build a Tree

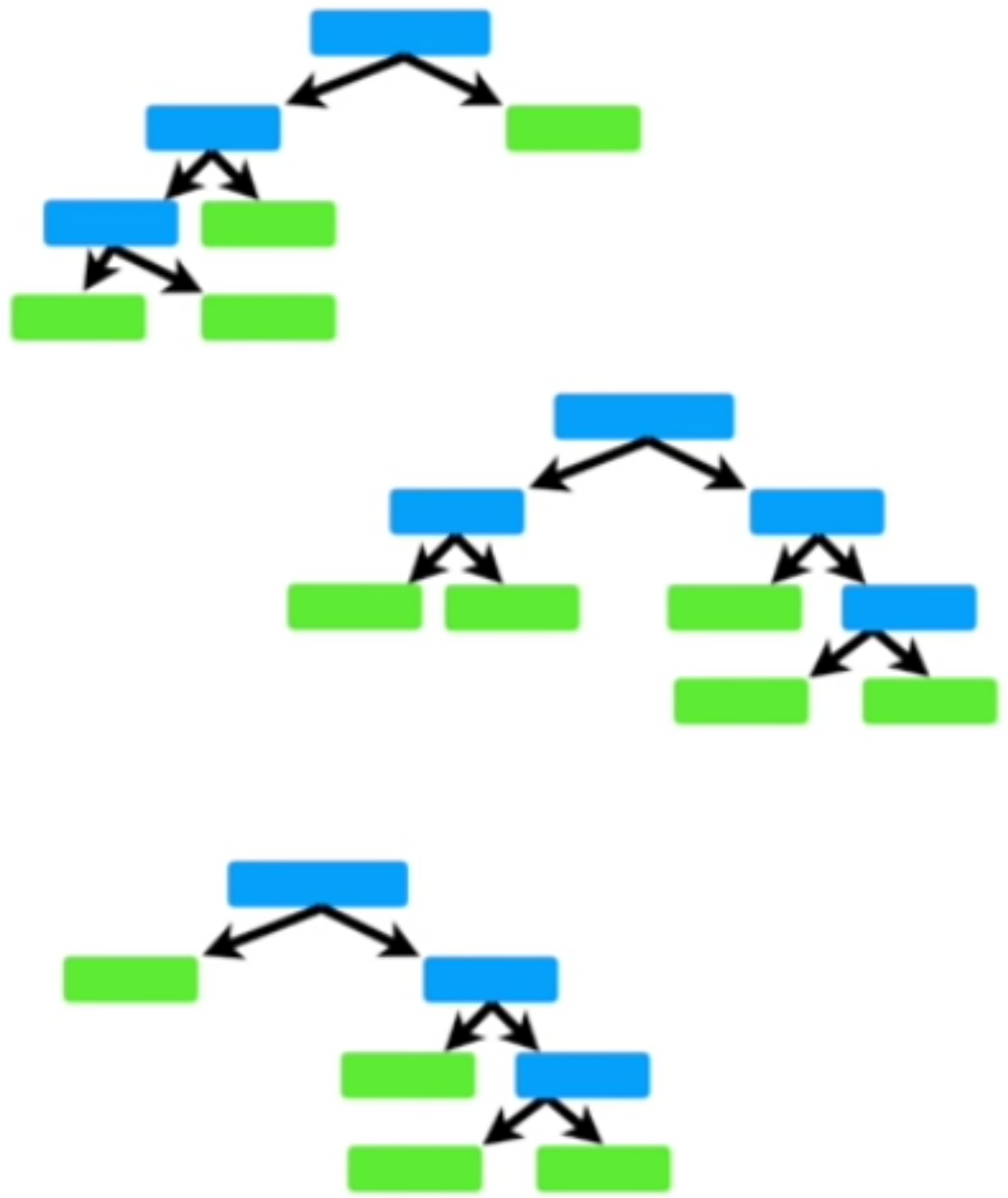


Random Forests — an *ensemble* method

- Repeat process until N trees are built (a **random forest**)
  1. Bootstrap random dataset
  2. Randomly select features
  3. Build new tree

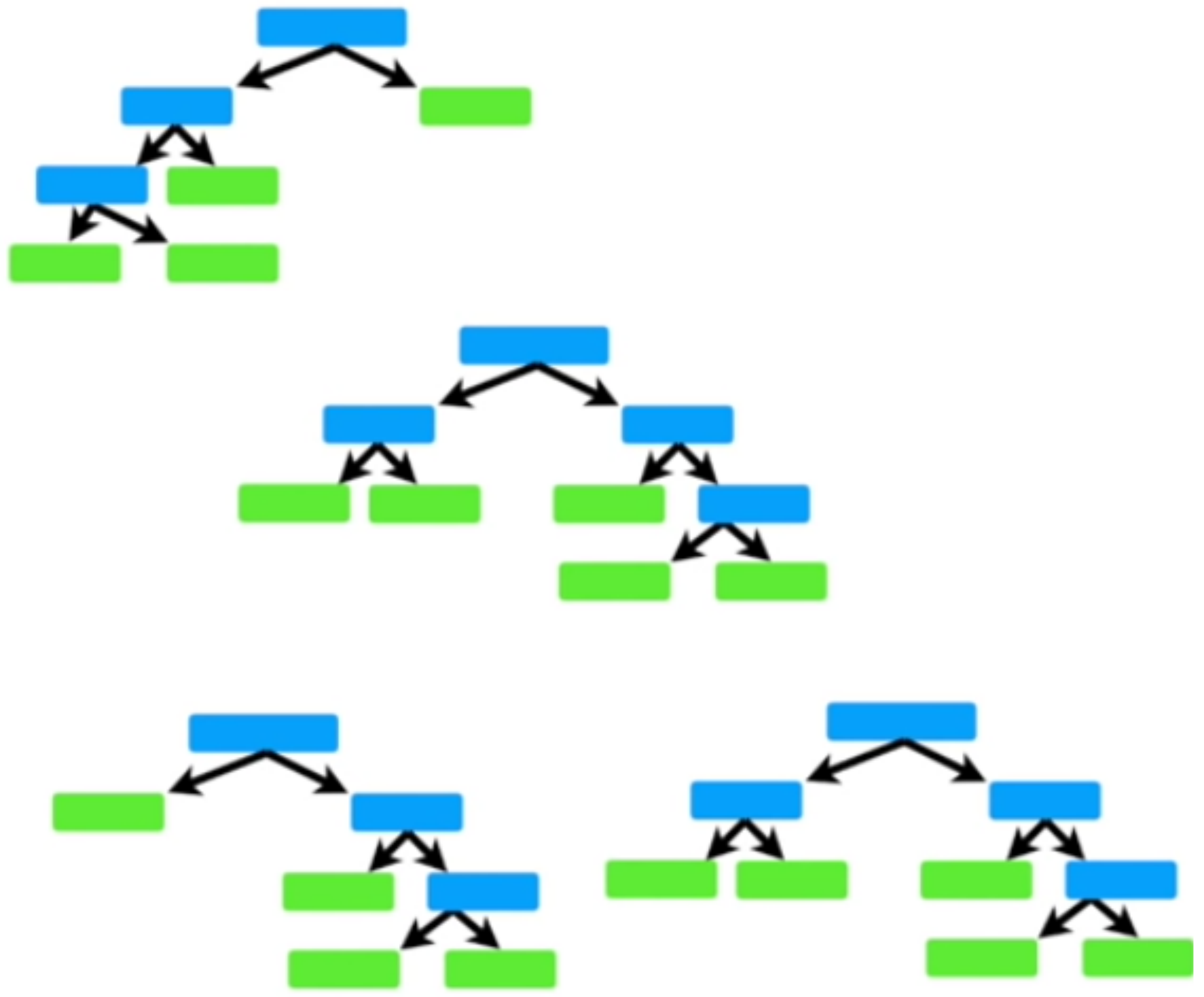
Random Forests — an *ensemble* method

- Repeat process until N trees are built (a **random forest**)
  1. Bootstrap random dataset
  2. Randomly select features
  3. Build new tree



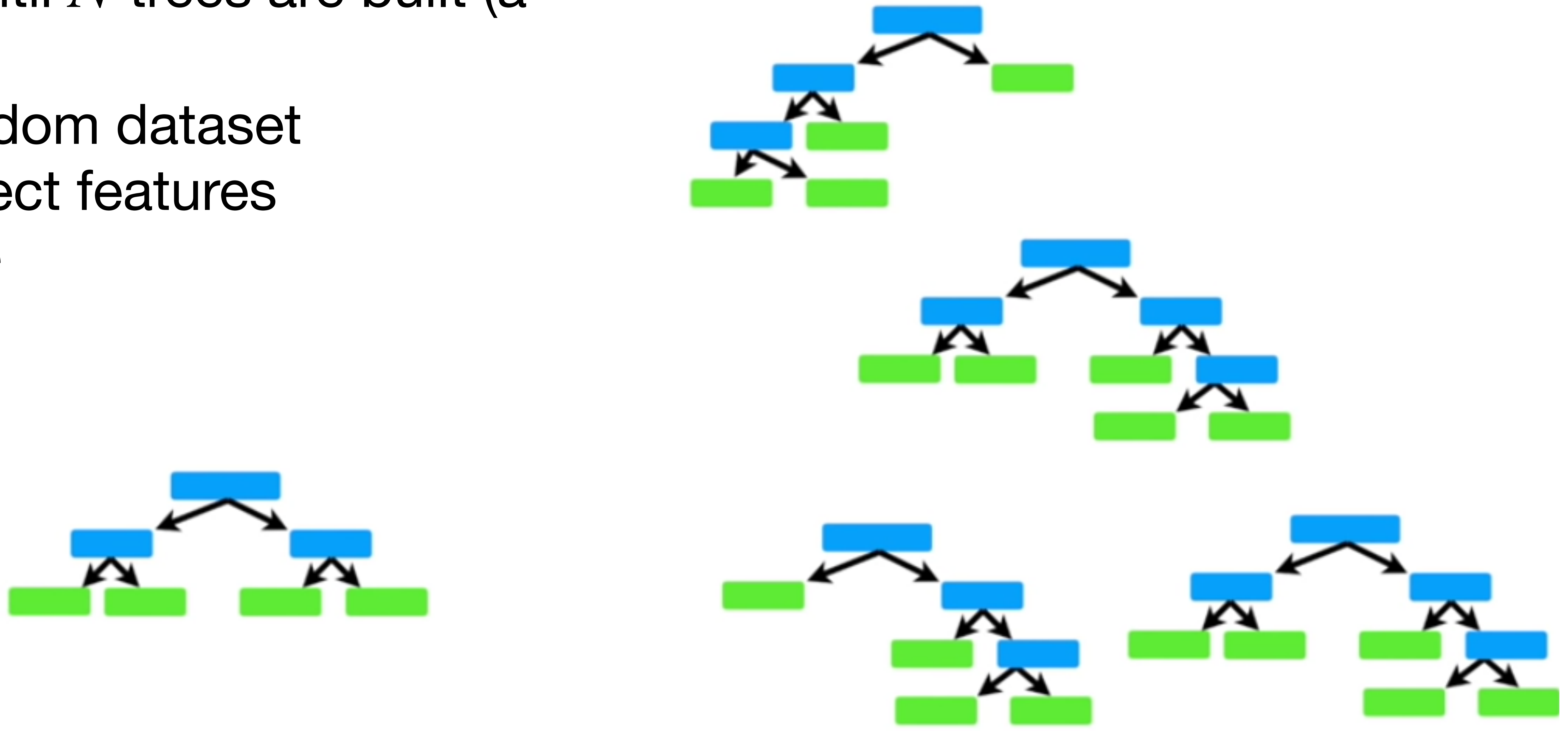
Random Forests — an *ensemble* method

- Repeat process until N trees are built (a **random forest**)
  1. Bootstrap random dataset
  2. Randomly select features
  3. Build new tree



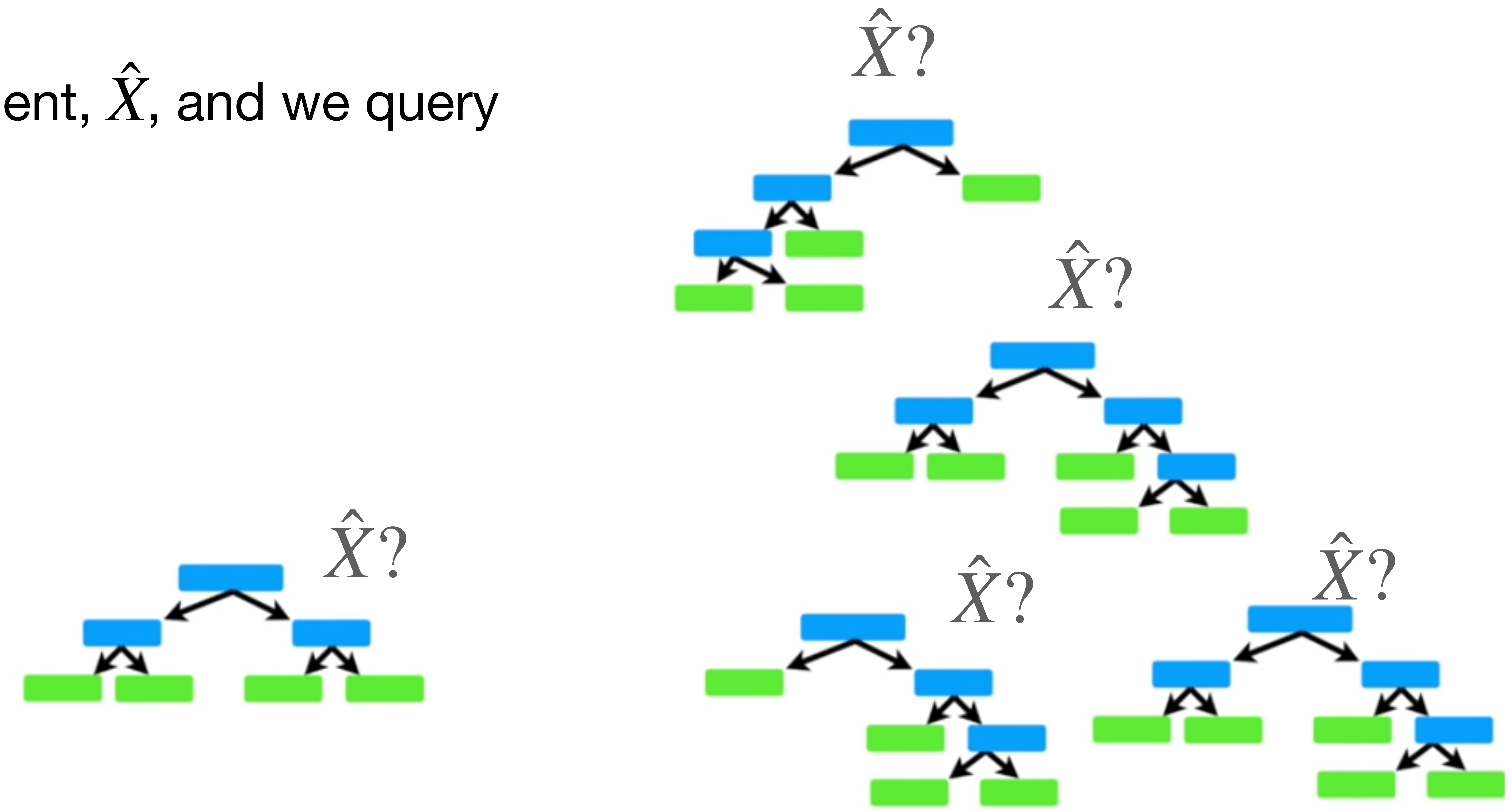
Random Forests — an *ensemble* method

- Repeat process until N trees are built (a **random forest**)
  1. Bootstrap random dataset
  2. Randomly select features
  3. Build new tree



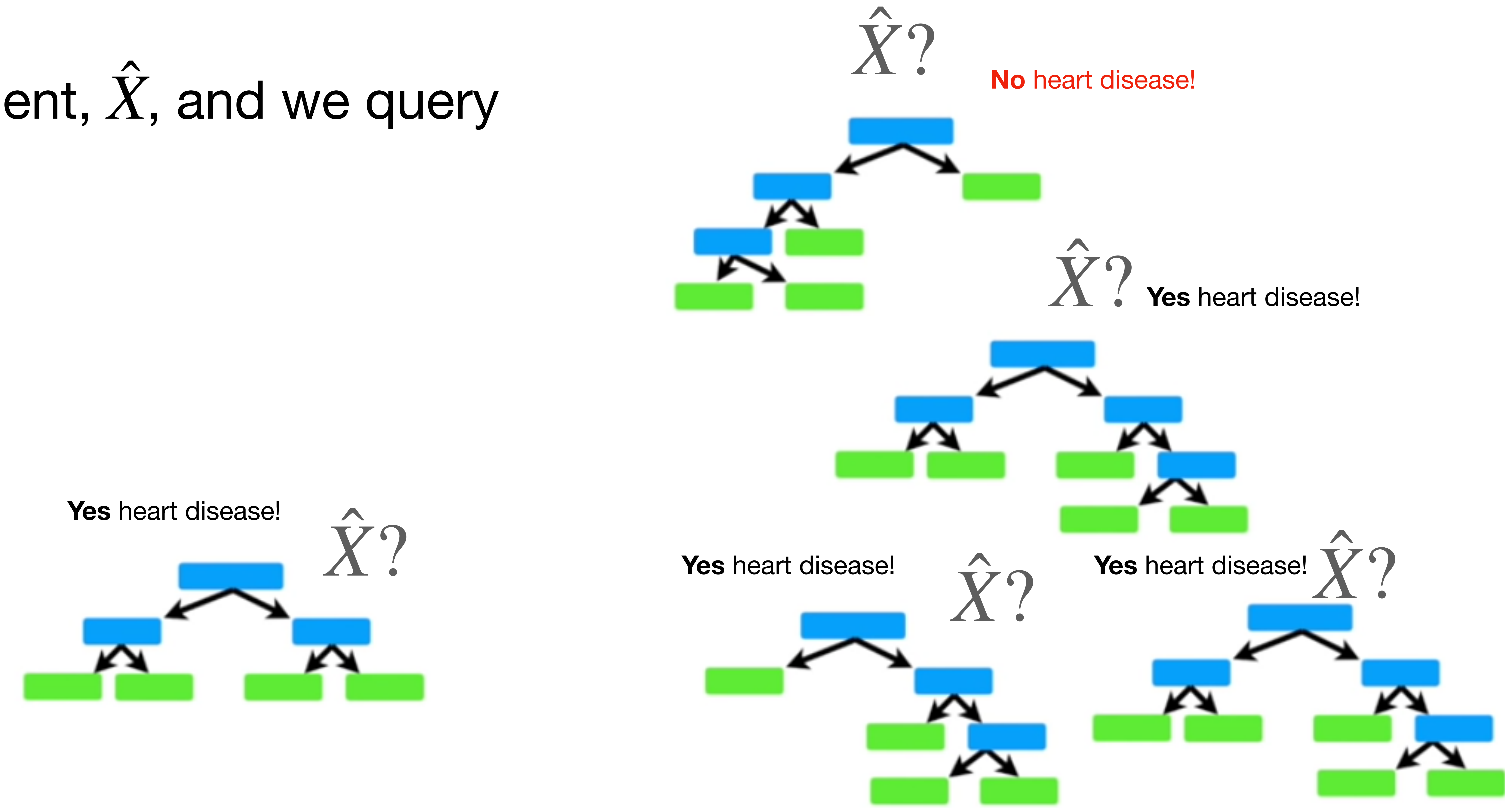
Random Forests — prediction by majority vote

- We get a new patient, $\hat{X}$, and we query all our trees



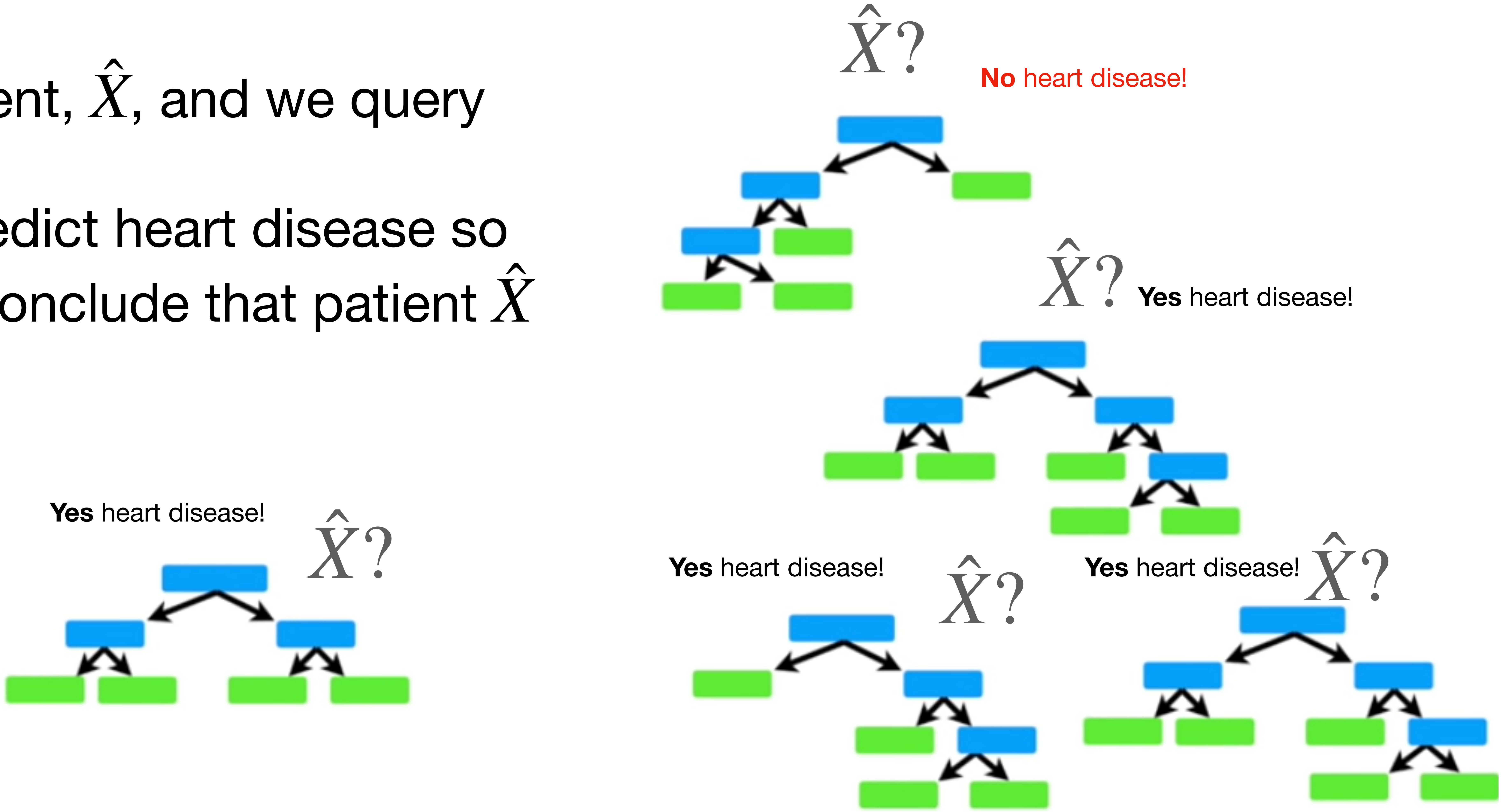
Random Forests — prediction by majority vote

- We get a new patient, $\hat{X}$, and we query all our trees

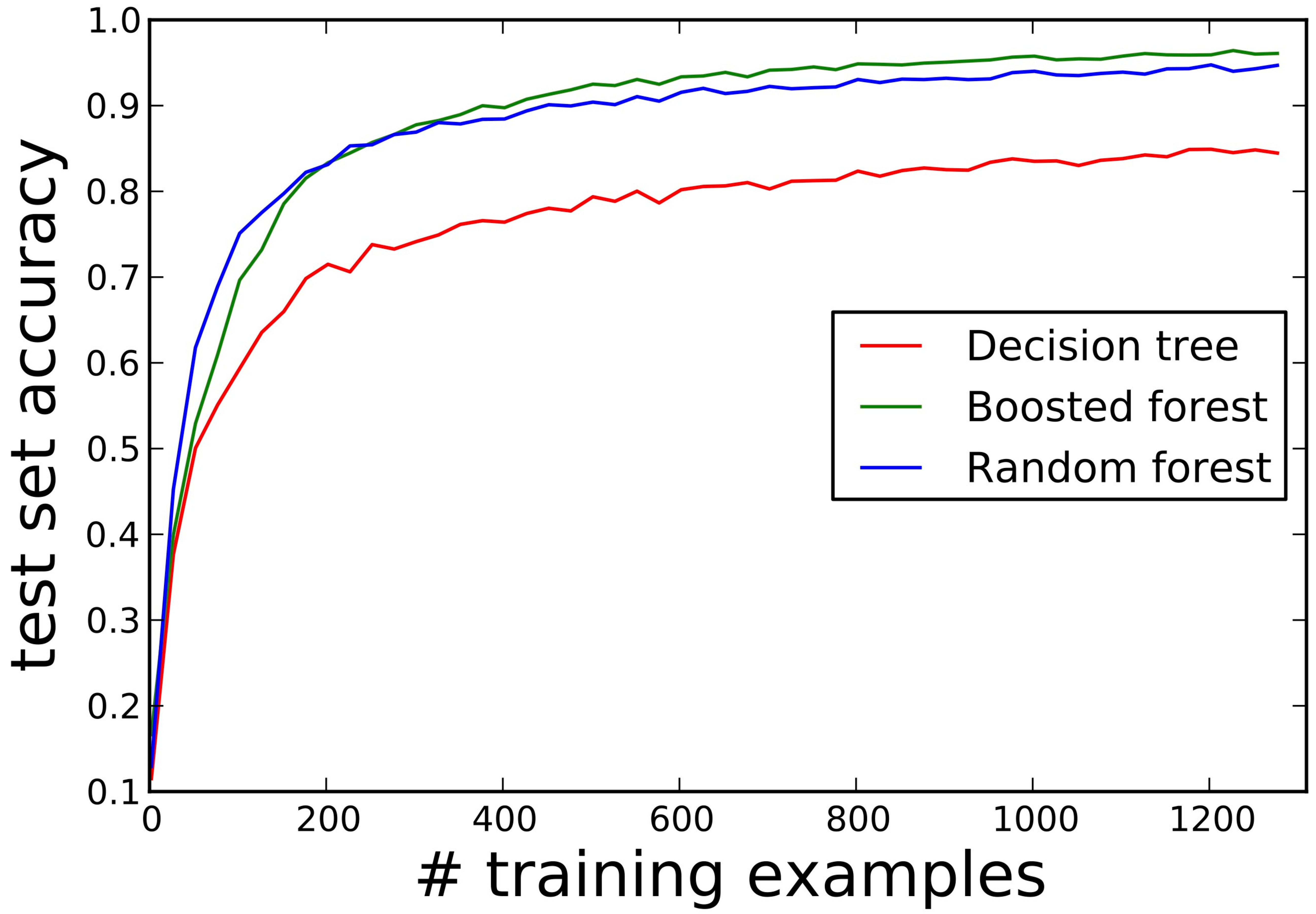


Random Forests — prediction by majority vote

- We get a new patient, $\hat{X}$, and we query all our trees
- 4 out of 5 trees predict heart disease so we unfortunately conclude that patient $\hat{X}$ must be sick.



Random Forests — accuracy



Random Forests — pros and cons

- **Pros**
  - **Random forests** is considered as a **highly accurate** and robust method because of the number of decision trees participating in the process.
  - The method does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.
  - **Random forests** can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.

- **Cons**
  - Random forests can be **slow** in generating predictions because it has multiple decision trees.
  - The model is **difficult to interpret** compared to a **decision tree**, where you can easily make a decision by following the path in the tree.