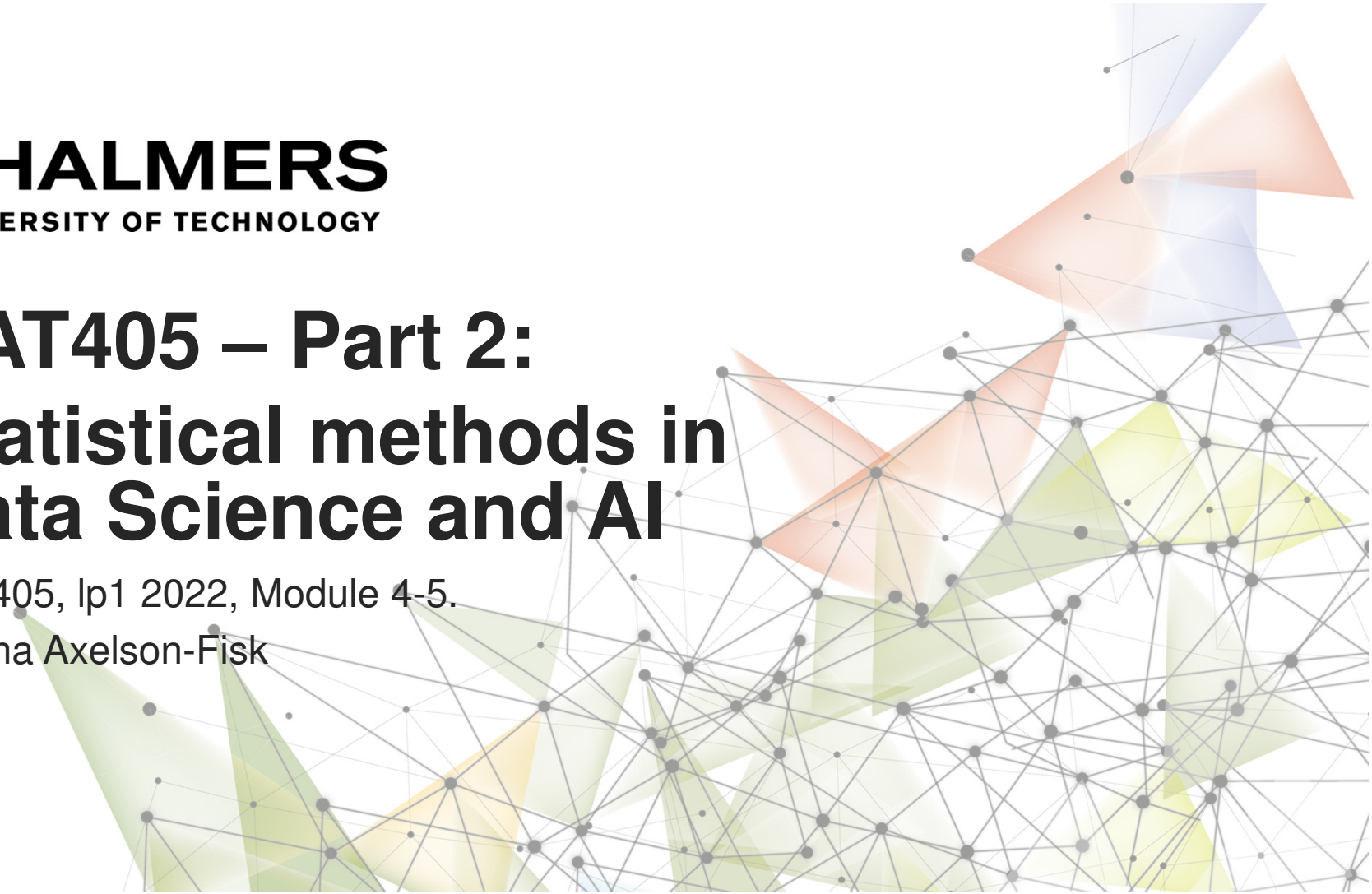
CHALMERS
UNIVERSITY OF TECHNOLOGY

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

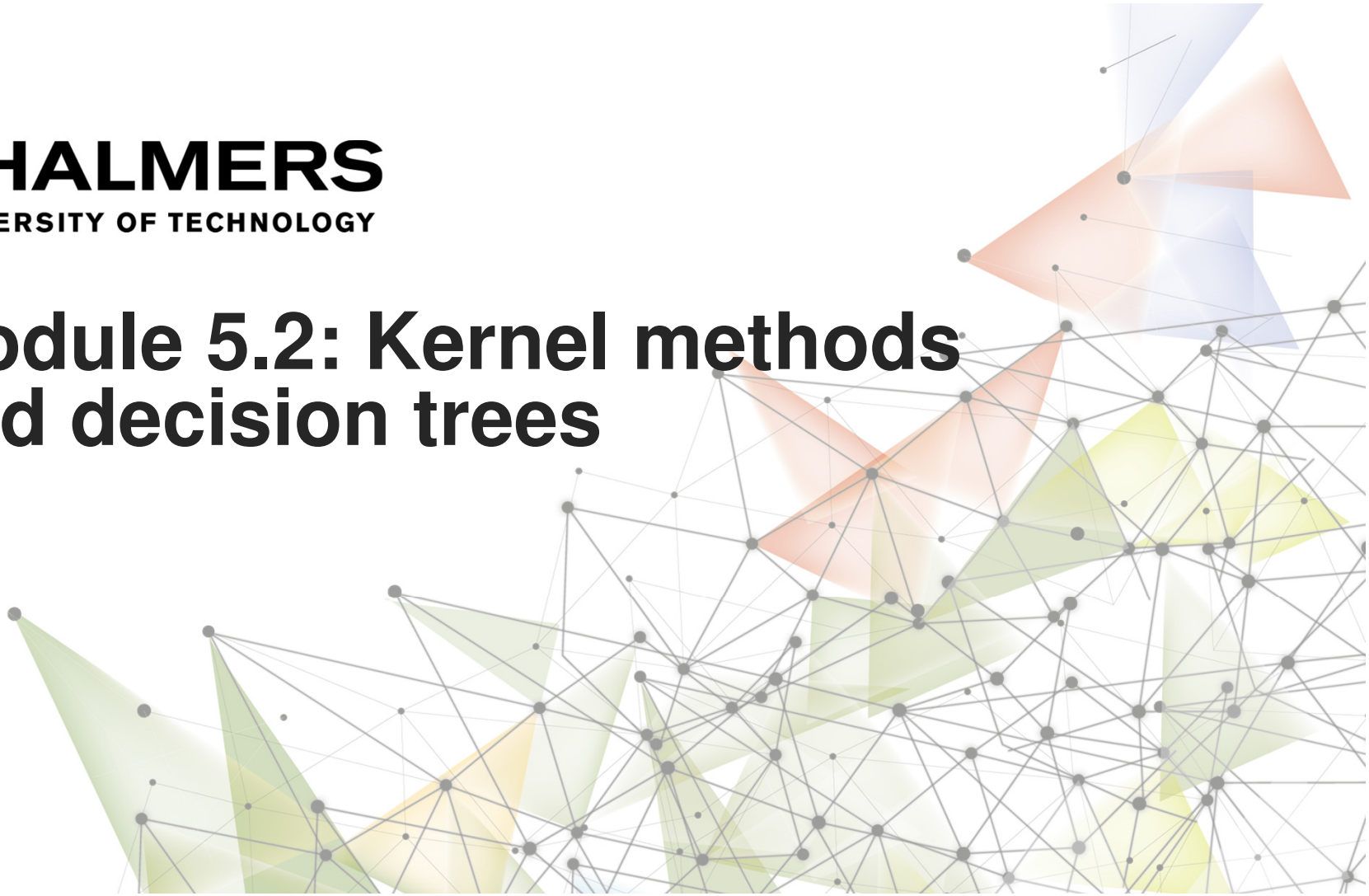
DAT405, lp1 2022, Module 4-5.

Marina Axelson-Fisk



CHALMERS
UNIVERSITY OF TECHNOLOGY

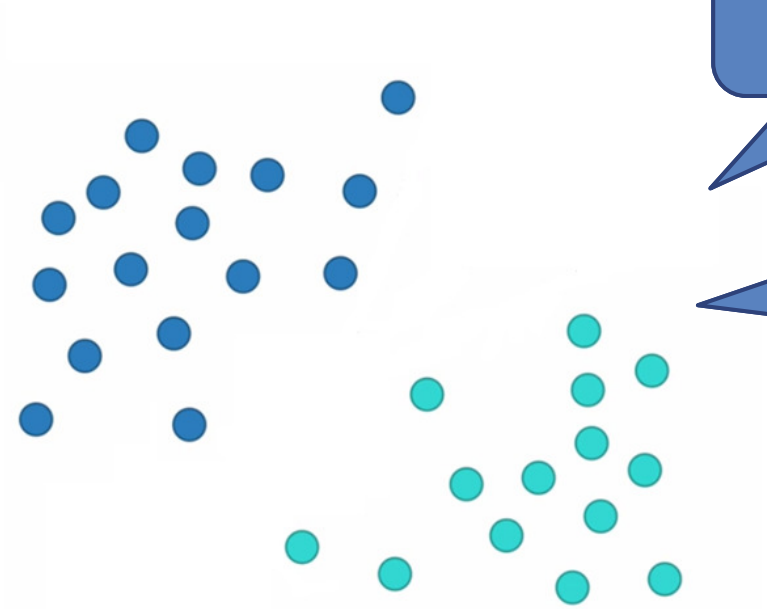
# Module 5.2: Kernel methods and decision trees



# Today's topics:

- Support Vector Machines (SVMs)
- Overlapping data
- Kernels
- Logistic Regression
- Gaussian Processes
- Decision Trees
- Random Forests

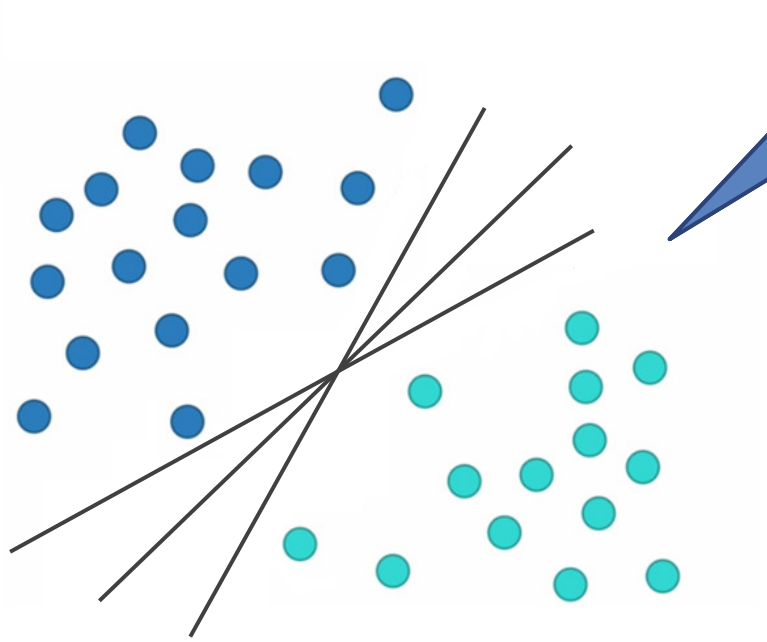
# Support vector machines



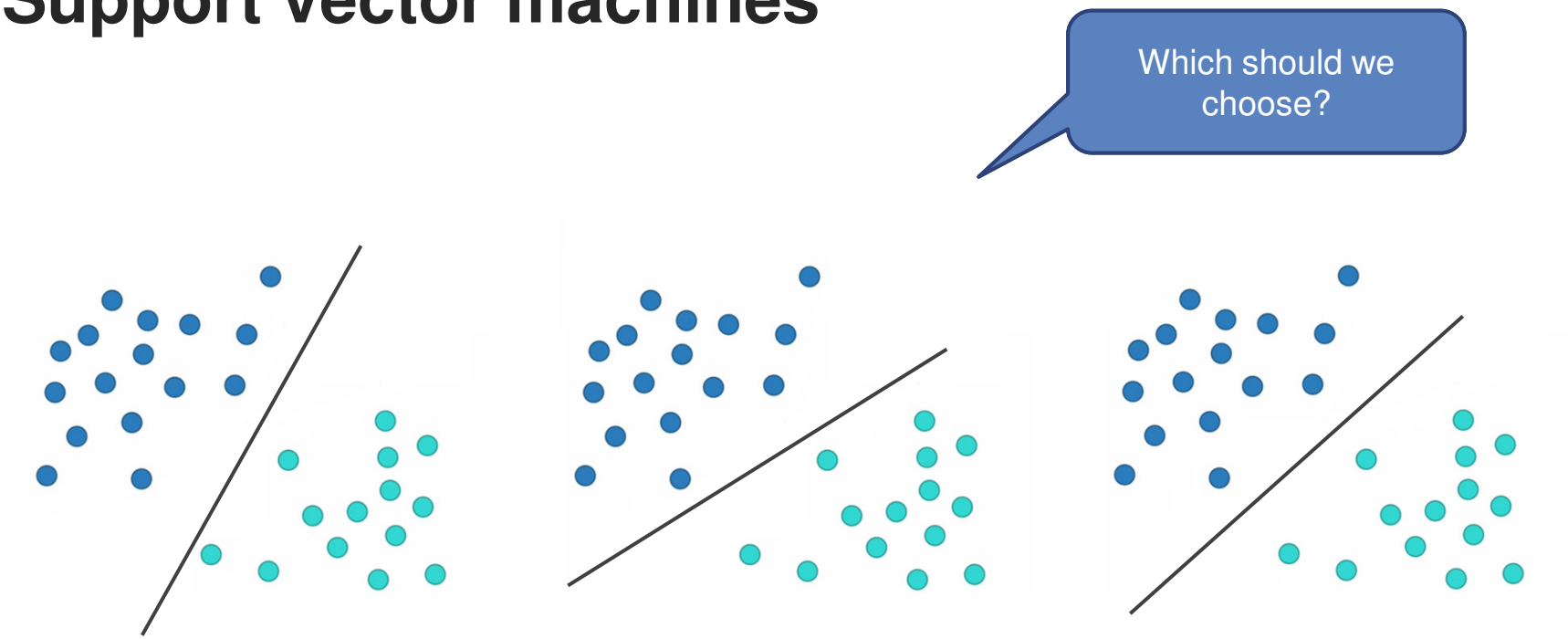
In classification we are often faced with this problem:

Find the optimal separation of the two classes in the data.

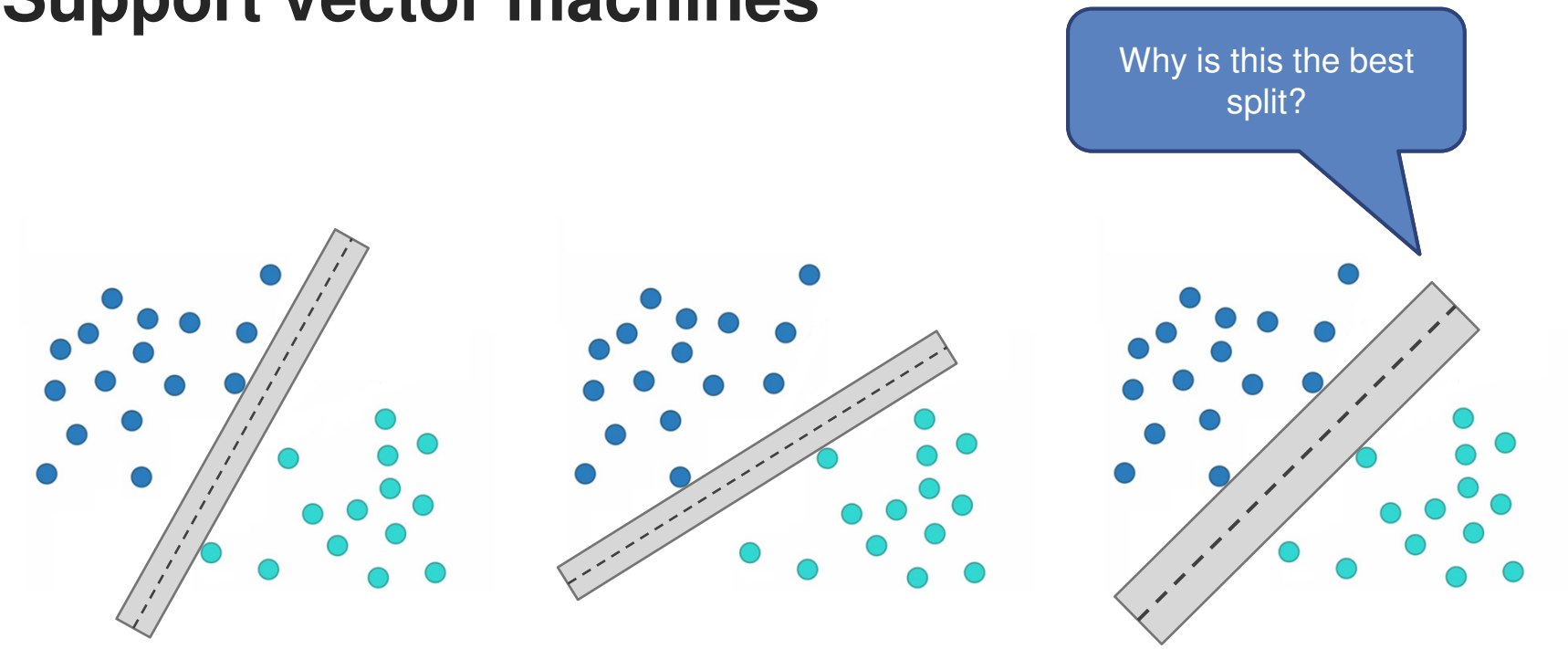
# Support vector machines



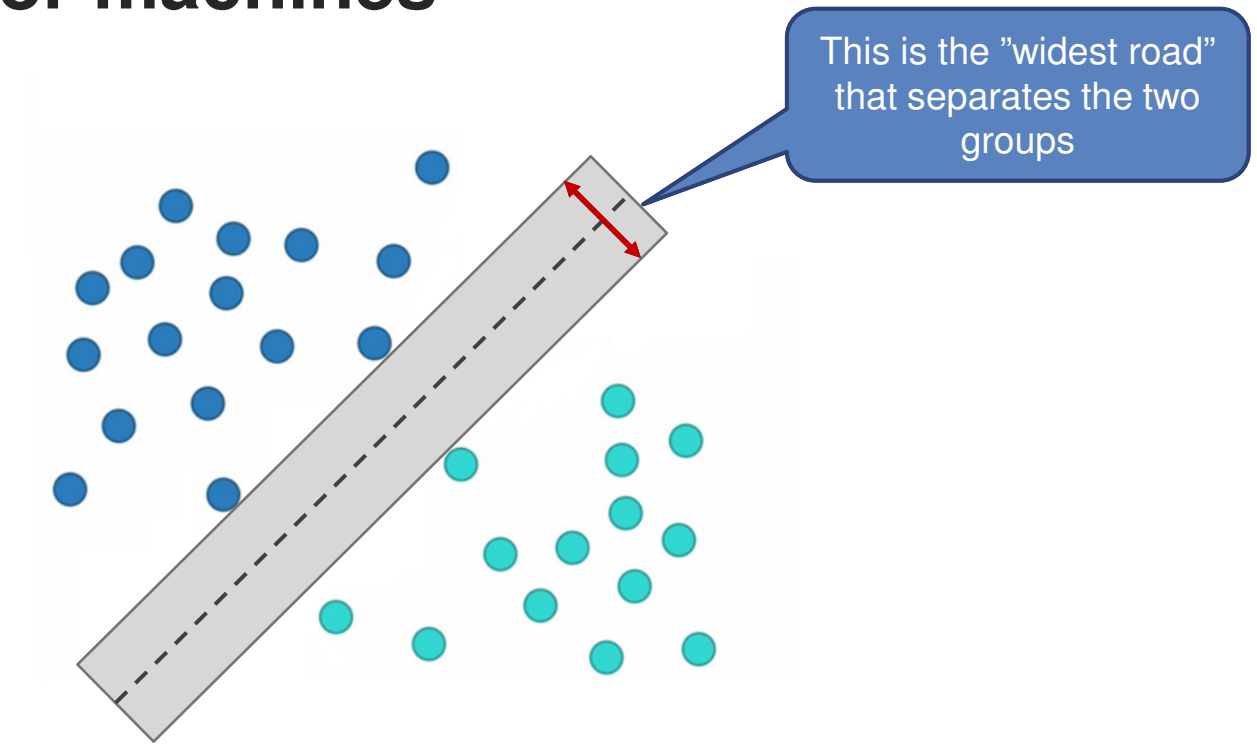
# Support vector machines



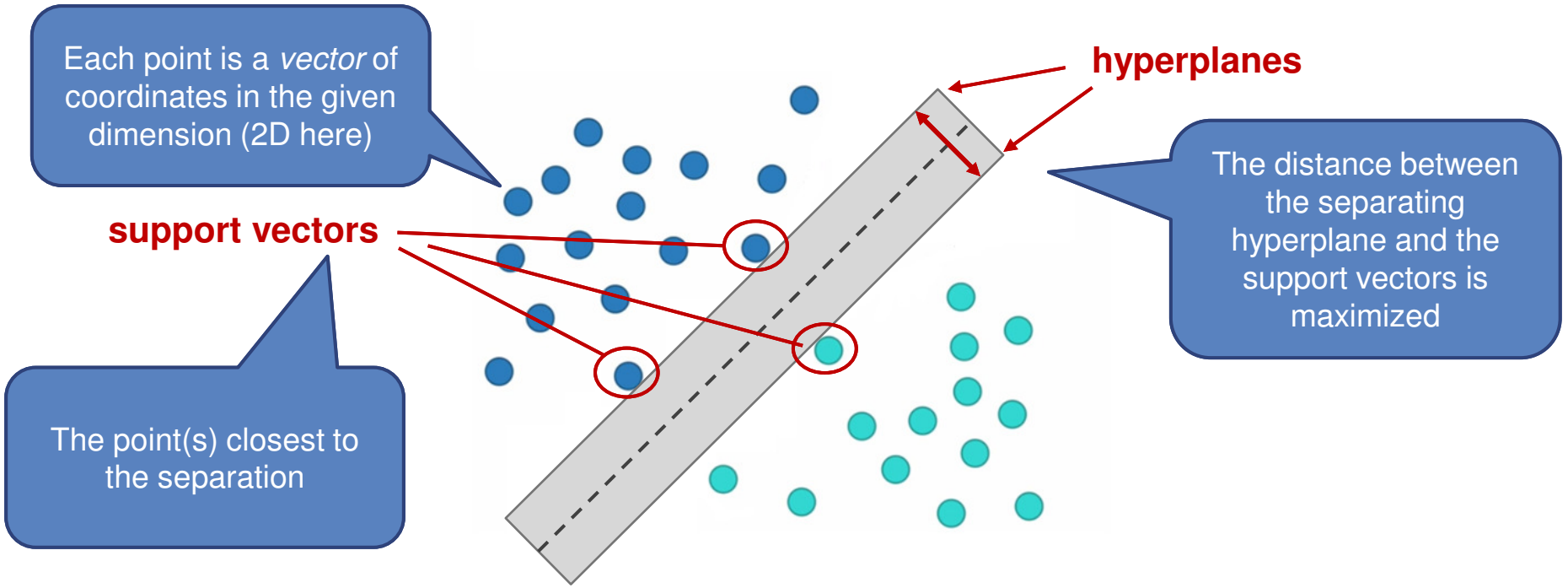
# Support vector machines



# Support vector machines



# Support vector machines



# Cupcakes and muffins: what's the difference?

**Cupcakes**

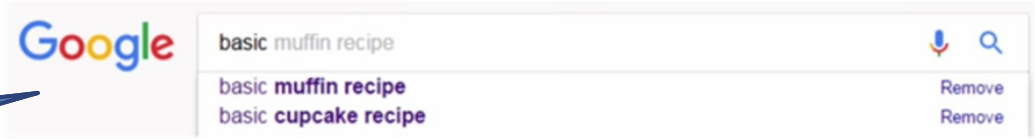
versus

**Muffins**

Let's proceed scientifically!

# Collect data

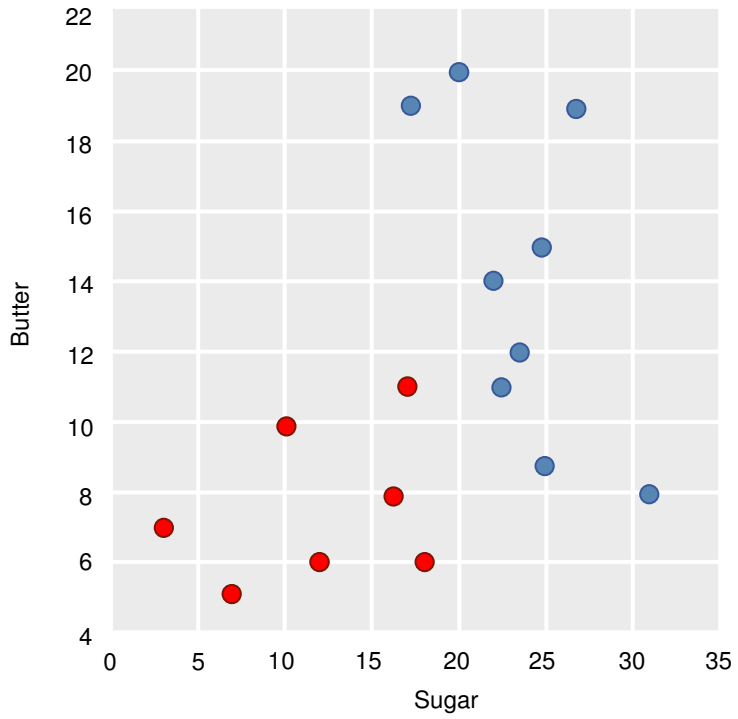
Find 9 recipes of each kind.




Express the ingredients as weight percentages.

| Type | Flour | Milk | Sugar | Butter | Egg | Baking Powder | Vanilla | Salt |
|---|---|---|---|---|---|---|---|---|
| Muffin | 55 | 28 | 3 | 7 | 5 | 2 | 0 | 0 |
| Muffin | 47 | 24 | 12 | 6 | 9 | 1 | 0 | 0 |
| Muffin | 47 | 23 | 18 | 6 | 4 | 1 | 0 | 0 |
| Muffin | 50 | 25 | 12 | 6 | 5 | 2 | 1 | 0 |
| Muffin | 55 | 27 | 3 | 7 | 5 | 2 | 1 | 0 |
| Muffin | 54 | 27 | 7 | 5 | 5 | 2 | 0 | 0 |
| Muffin | 47 | 26 | 10 | 10 | 4 | 1 | 0 | 0 |
| Muffin | 50 | 17 | 17 | 8 | 6 | 1 | 0 | 0 |
| Muffin | 50 | 17 | 17 | 11 | 4 | 1 | 0 | 0 |
| Cupcake | 39 | 0 | 26 | 19 | 14 | 1 | 1 | 0 |
| Cupcake | 34 | 17 | 20 | 20 | 5 | 2 | 1 | 0 |
| Cupcake | 39 | 13 | 17 | 19 | 10 | 1 | 1 | 0 |
| Cupcake | 38 | 15 | 23 | 15 | 8 | 0 | 1 | 0 |
| Cupcake | 42 | 18 | 25 | 9 | 5 | 1 | 0 | 0 |
| Cupcake | 36 | 14 | 21 | 14 | 11 | 2 | 1 | 0 |
| Cupcake | 38 | 15 | 31 | 8 | 6 | 1 | 1 | 0 |
| Cupcake | 36 | 16 | 24 | 12 | 9 | 1 | 1 | 0 |
| Cupcake | 34 | 17 | 23 | 11 | 13 | 0 | 1 | 0 |

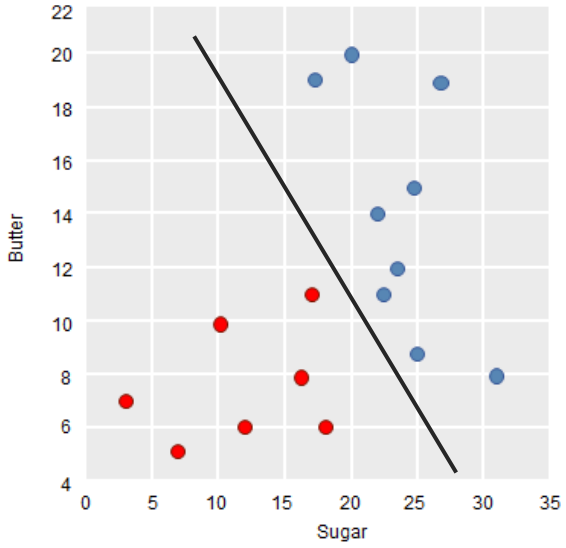
# Plotting the data

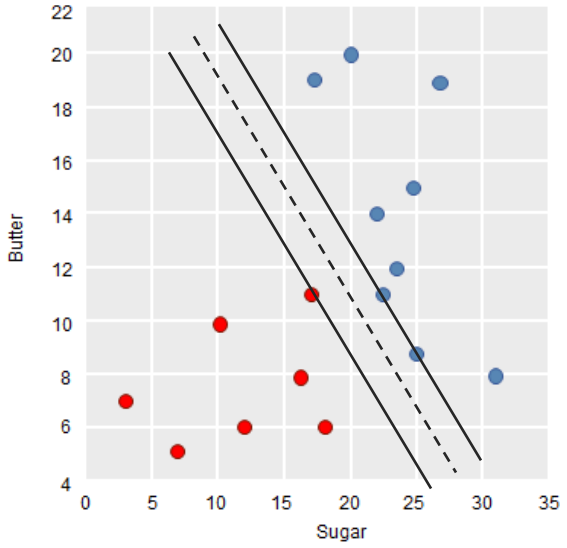



Focus on features ”sugar” and ”butter” only

# Applying an SVM

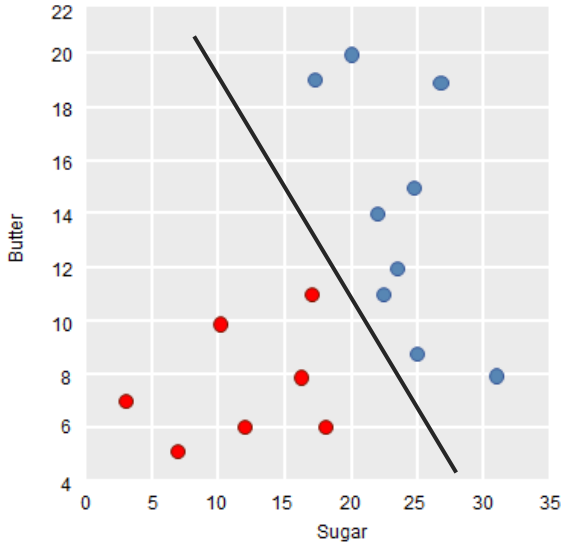
Finding the SVM model



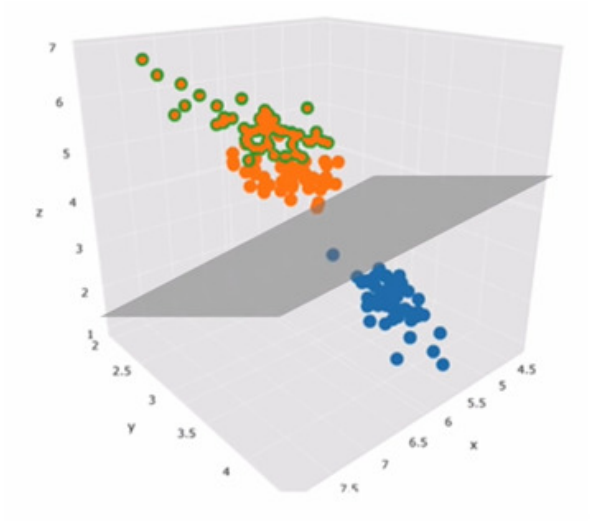



# SVMs separating hyperplanes

## 2D: Separate with line



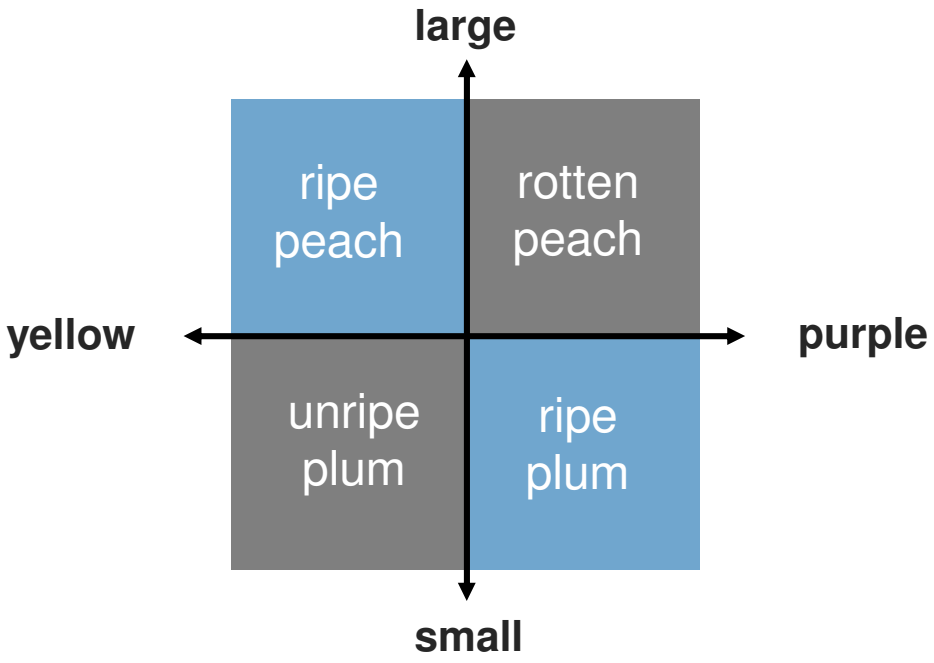

## 3D: Separate with plane



# Example: peaches and plums

- A certain batch of fruits are either
  - small or large
  - yellow or purple
- A small yellow fruit is an unripe plum. Not good to eat.
- A small purple fruit is a ripe plum. Good to eat.
- A large yellow fruit is a ripe peach. Good to eat.
- A large purple fruit is a rotten peach. Not good to eat.

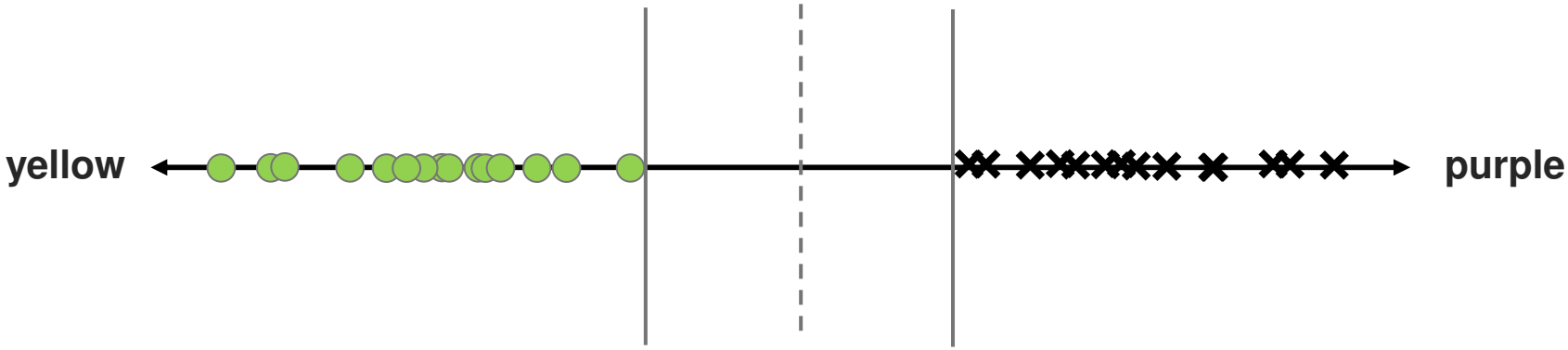
# Example: peaches and plums



# Example: peaches and plums

# Overlapping data

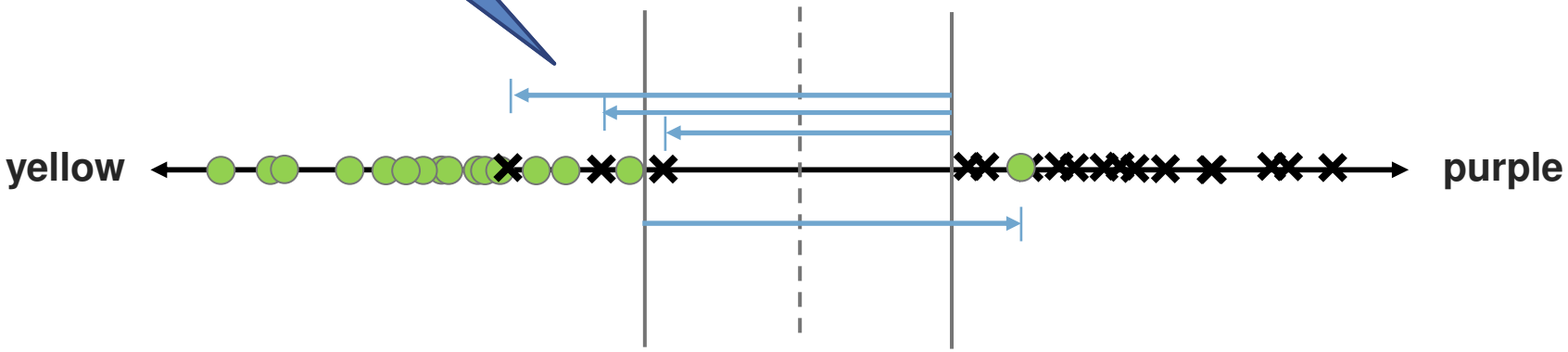
peaches



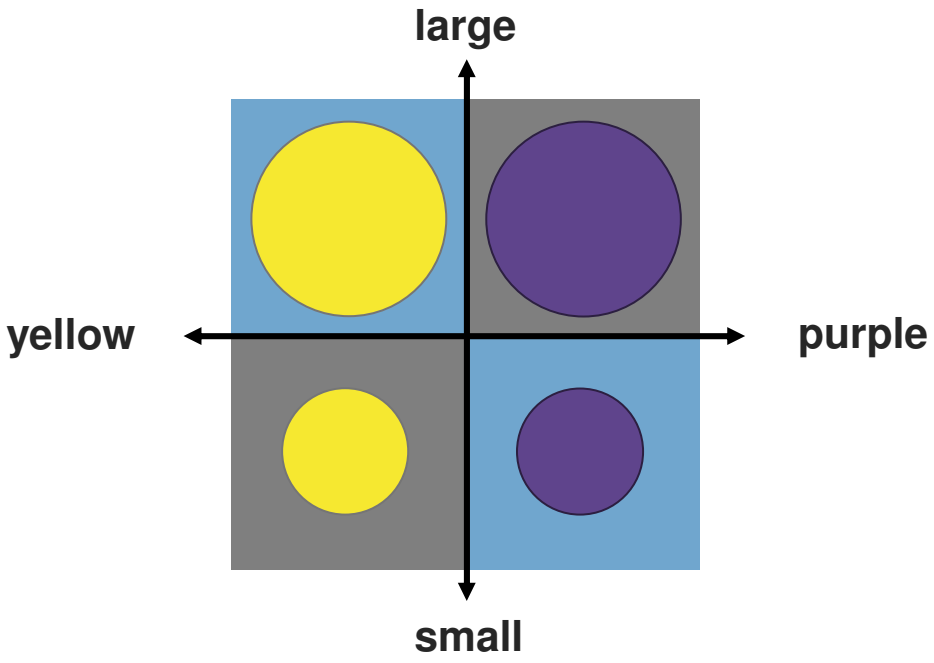
# Overlapping data

You can still use SVMs when the data is not linearly separable.

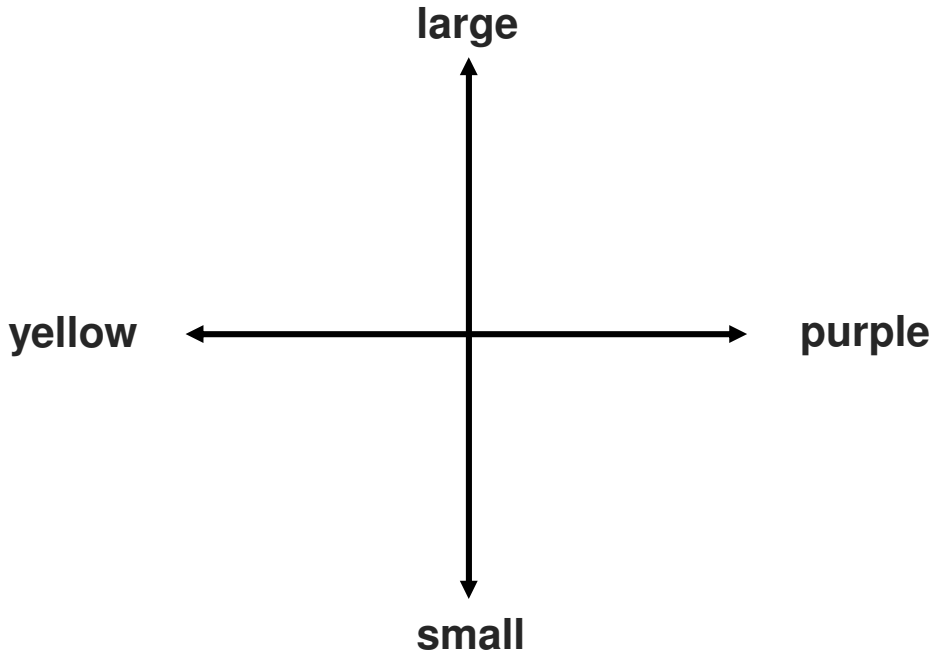
**peaches**



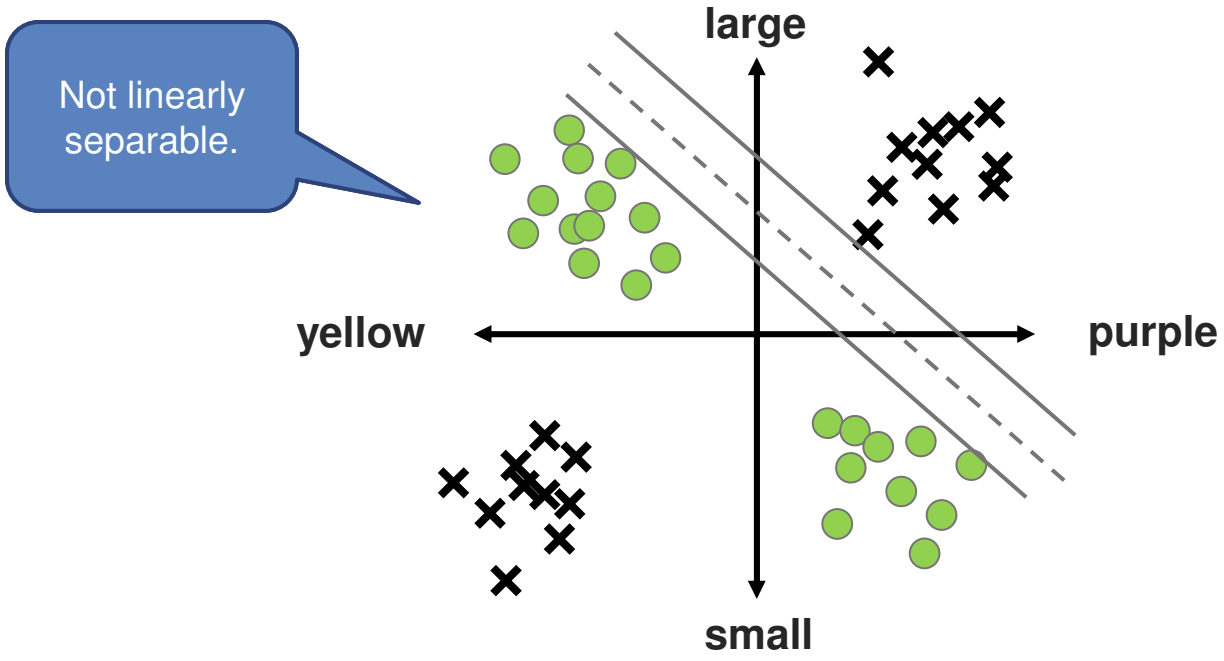
# Example: peaches and plums



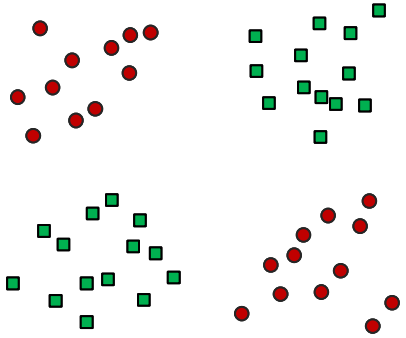
# Example: peaches and plums



# Example: peaches and plums

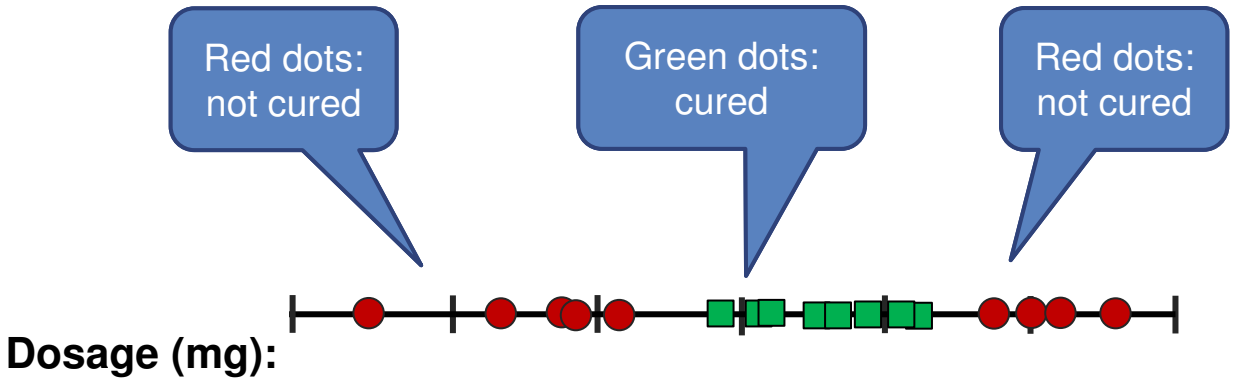


# Nonlinear classification

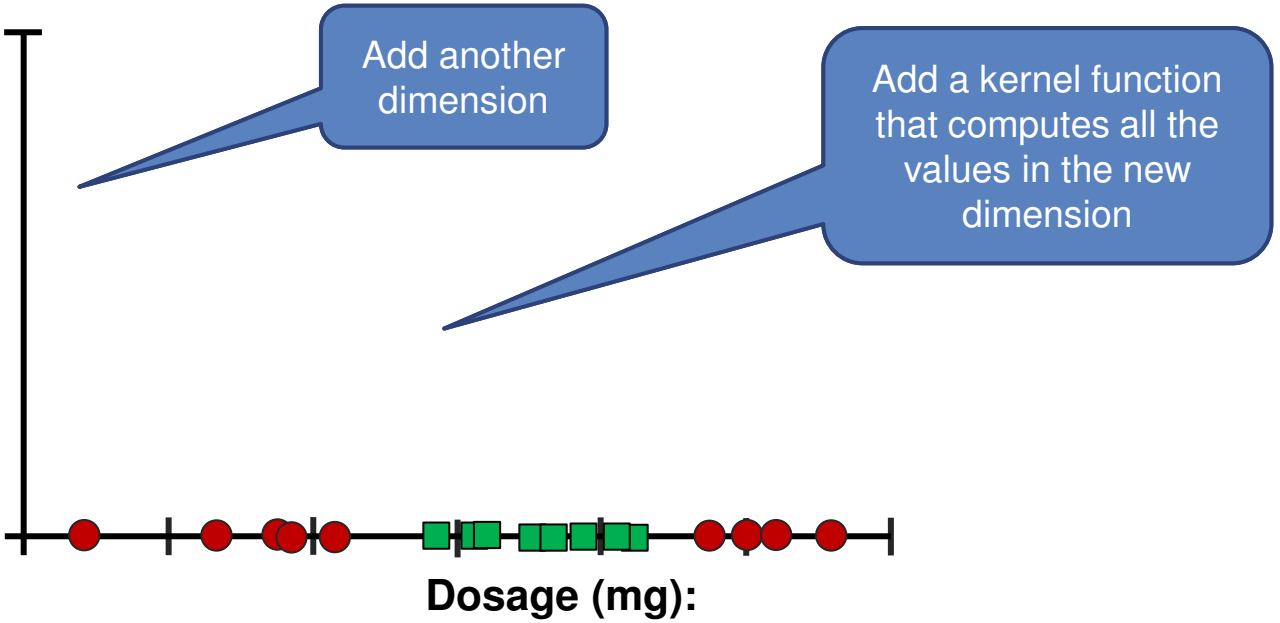


- There is no linear classifier that can separate red from green.

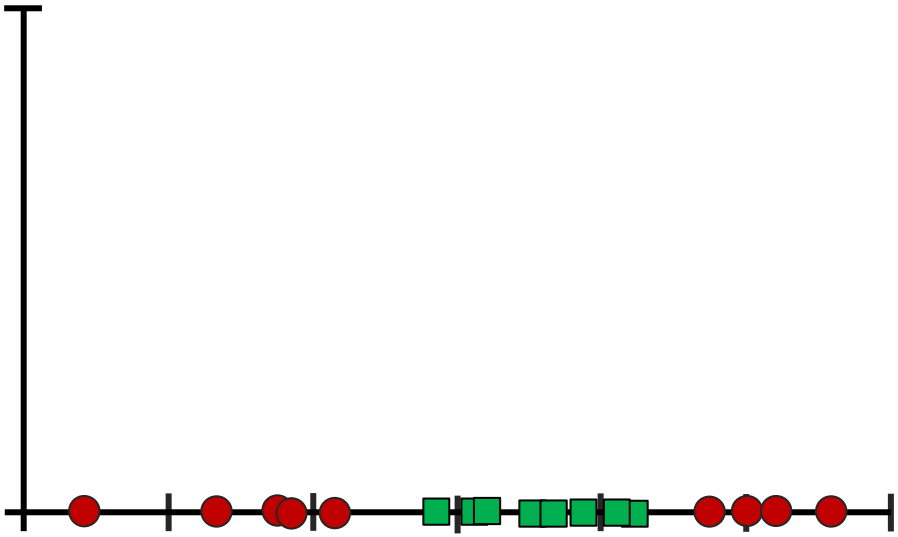
# Example: medicine dosage



# Example: medicine dosage

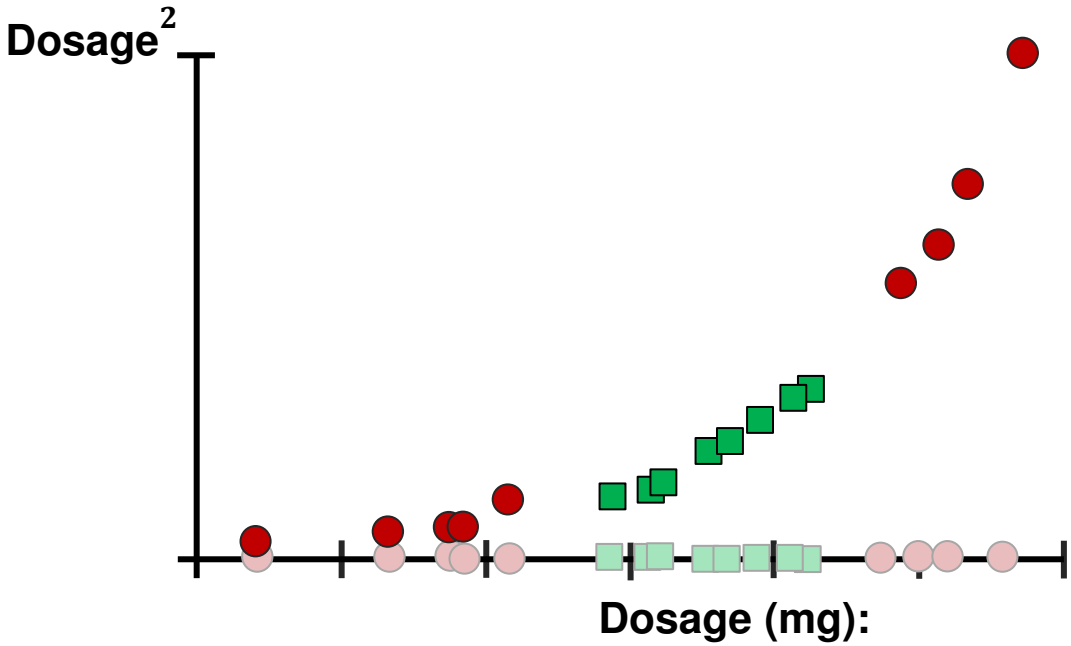


# Example: medicine dosage




Let's try kernel function $y = x^2$. If that doesn't work we can try something else.

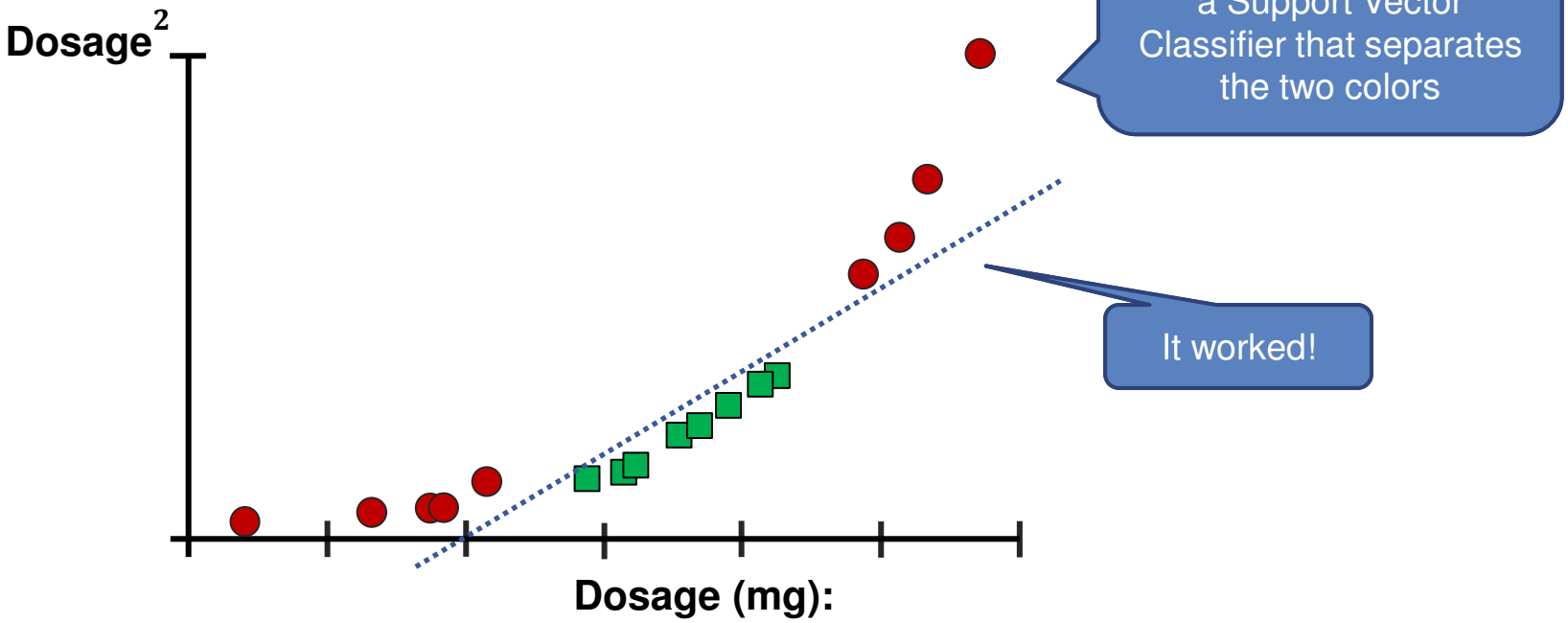
# Example: medicine dosage



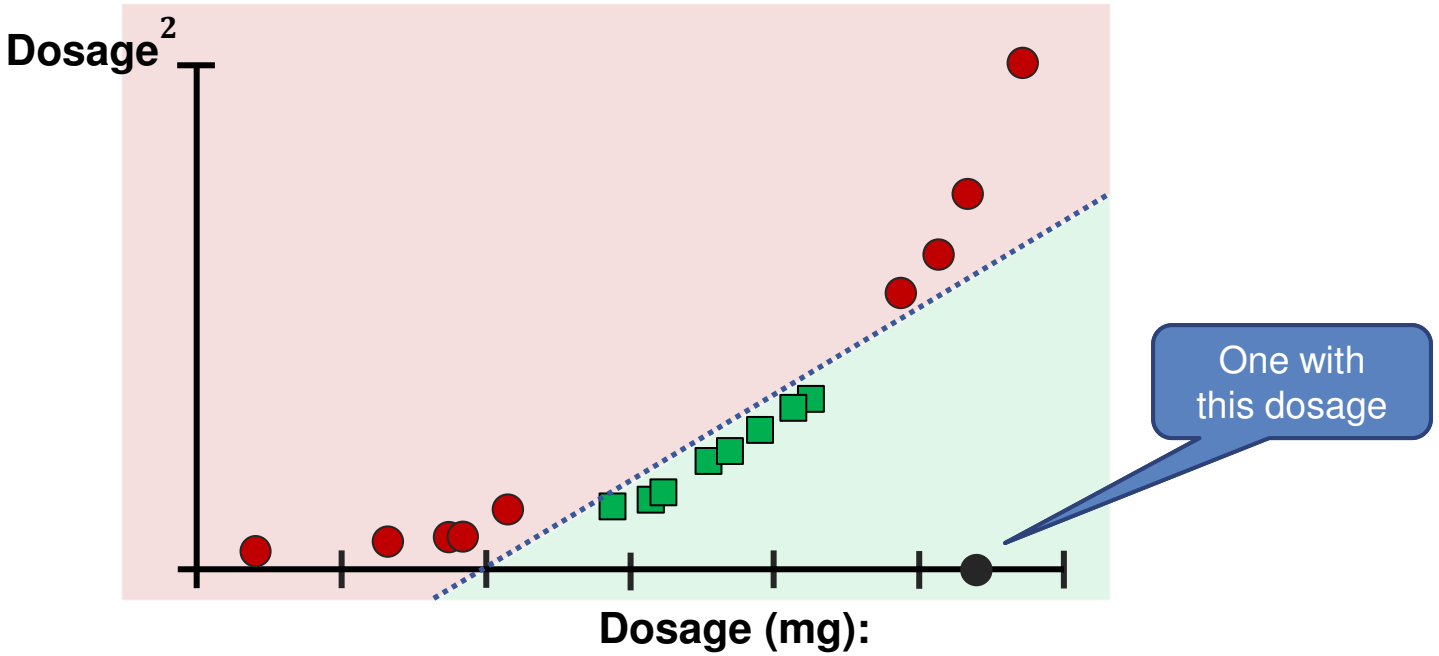

Let's try kernel function $y = x^2$. If that doesn't work we can try something else.

And then use $\text{Dosage}^2$ on the $y$-axis for the remaining observations

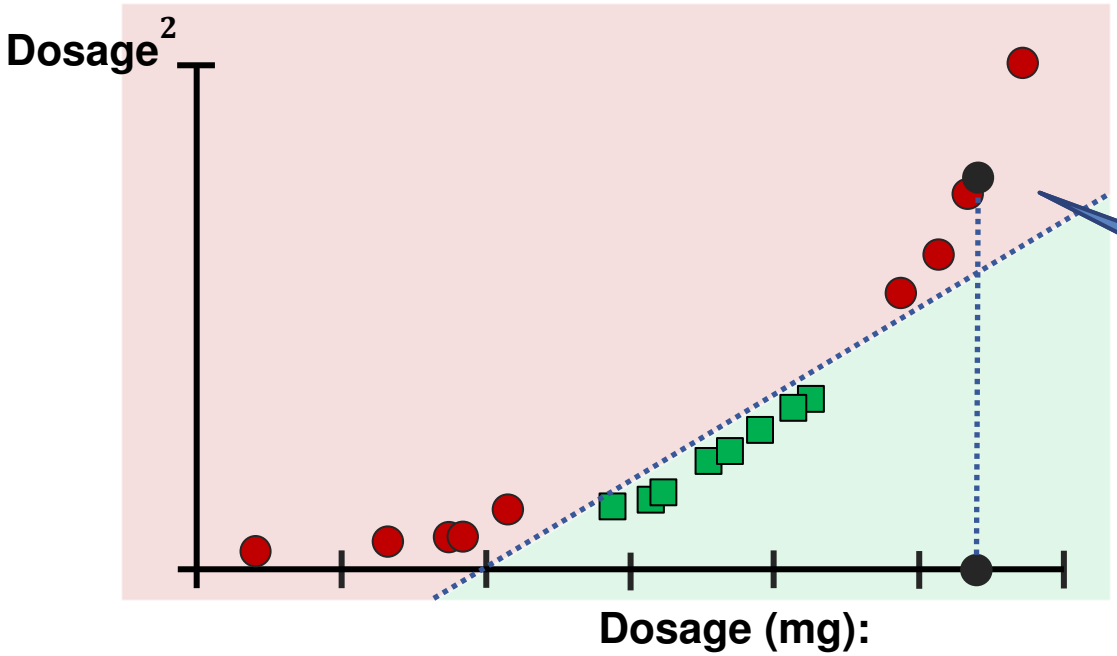
# Example: medicine dosage



# To classify a new data point …



# To classify a new data point …

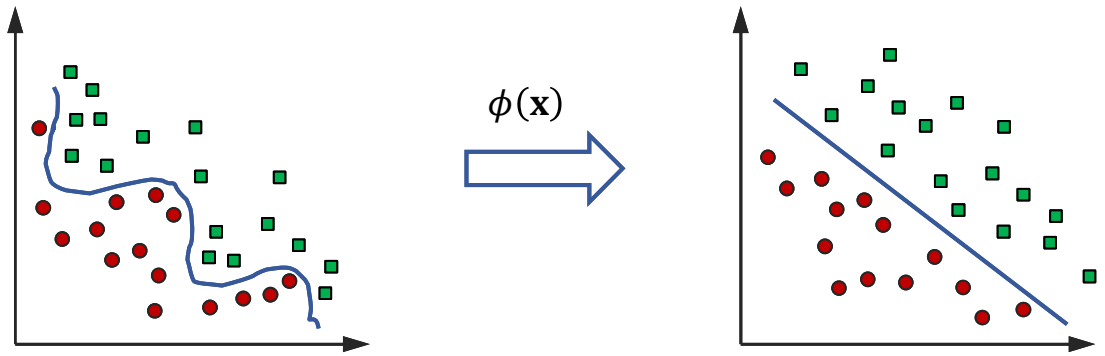



Then we calculate the $y$-axis value by squaring the dosage …

And check on which side of the line it falls

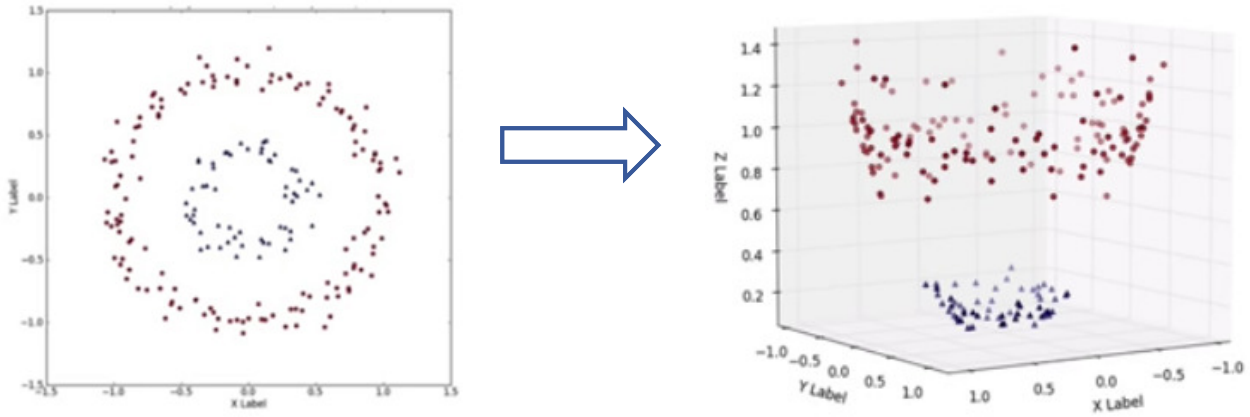
# Kernel methods: motivation

- **Solution:**
  - **map the data into a (possibly high-dimensional) vector space where the relation becomes linear**
  - **apply the linear algorithm in this space**



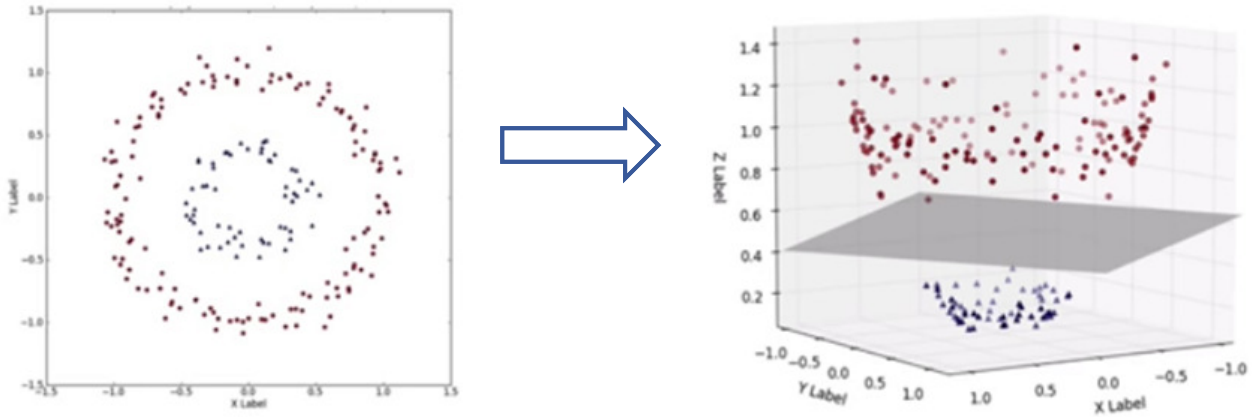
# Kernel methods: motivation

- **Solution:**
  - **map the data into a (possibly high-dimensional) vector space where the relation becomes linear**
  - **apply the linear algorithm in this space**

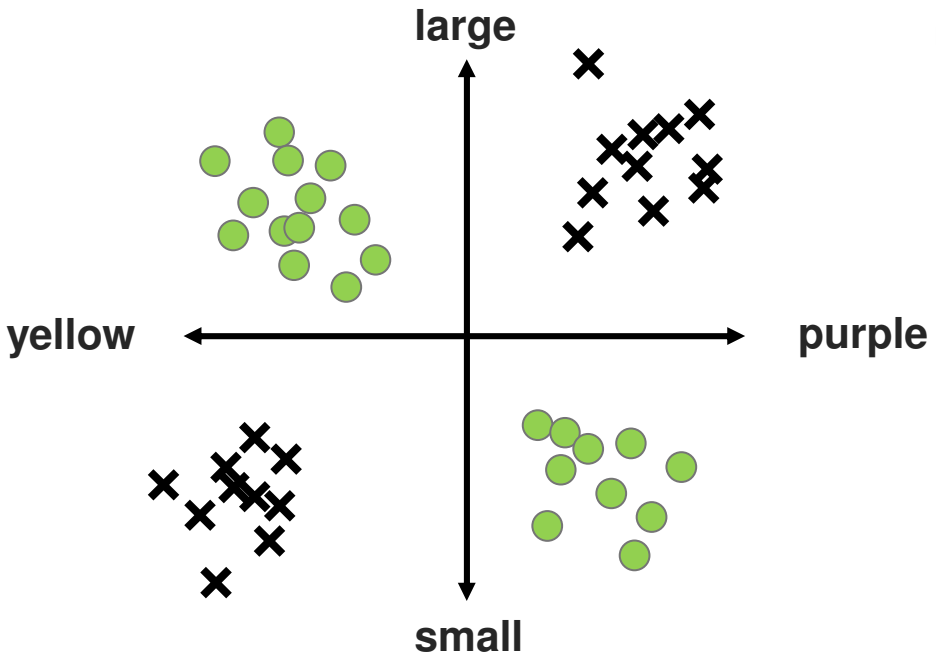


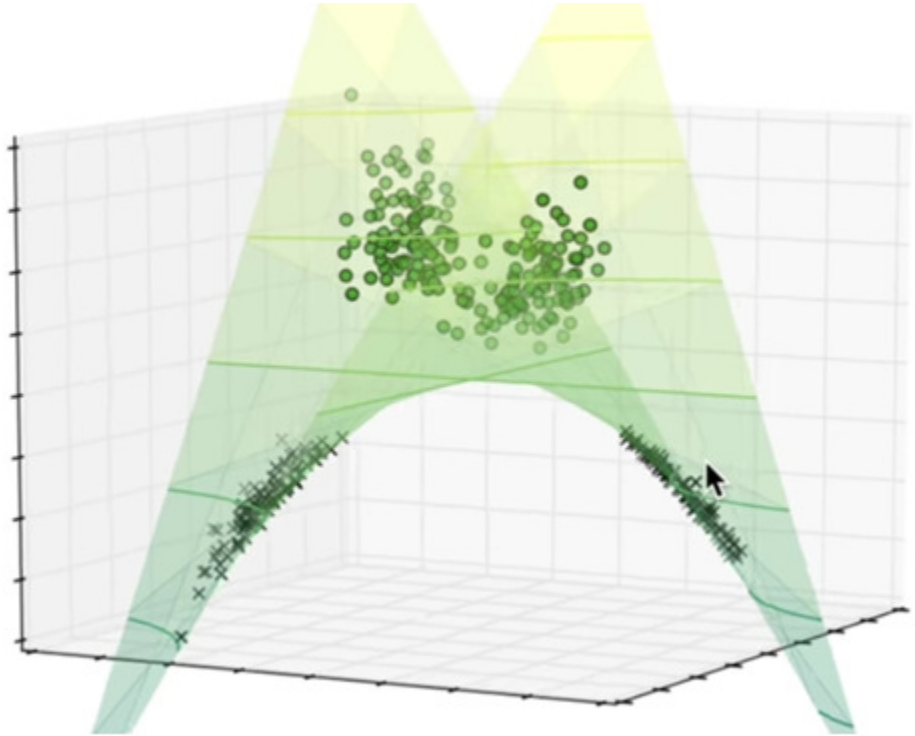
# Kernel methods: motivation

- **Solution:**
  - **map the data into a (possibly high-dimensional) vector space where the relation becomes linear**
  - **apply the linear algorithm in this space**



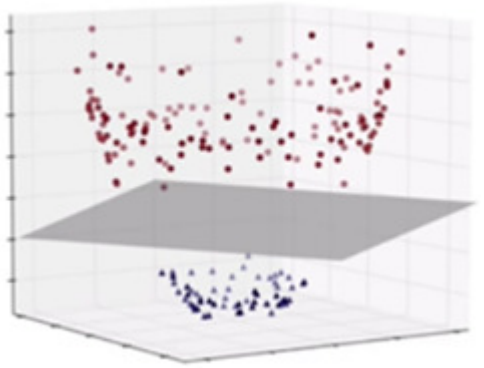
# Example: peaches and plums





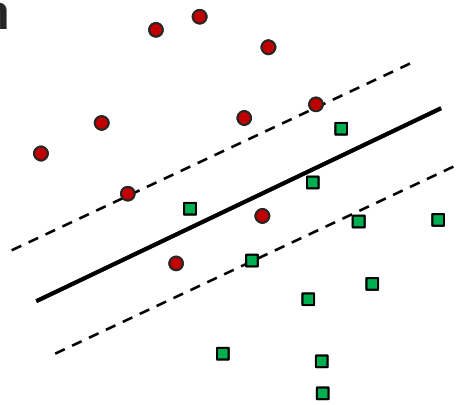
# Kernels

- How can we find a good kernel function – one that enables a decent separation between classes?
- Any function that does the job will do.
- Use a machine to search inside a class of functions (e.g. polynomials of degree 2)



# How to choose the mapping $\phi(\mathbf{x})$?

- Choosing an optimal feature space is non-trivial
- The kernel trick reduces this to choosing the best kernel, and determine the corresponding implicit mapping $\phi(\mathbf{x})$.
- Performance of the algorithm highly depends on the kernel
- The best kernel depends on the specific problem
- Kernels can be applied to
  - Numeric vectors
  - Strings
  - Trees
  - Graphs



# Radial basis function (RBF) kernels

- The ***squared exponential*** (SE) or ***Gaussian*** kernel

$$k(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{z})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mathbf{z})\right)$$

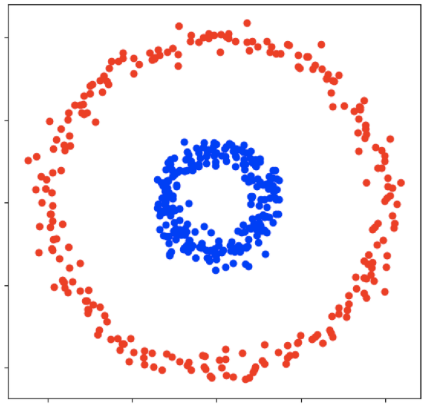
- If the covariance matrix $\boldsymbol{\Sigma}$ is diagonal, we get

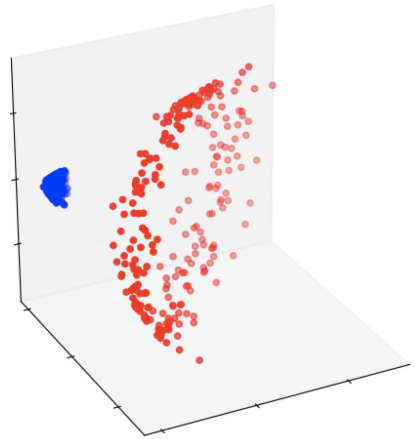
$$k(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{1}{2}\sum_j \frac{1}{\sigma^2}(x_j - z_j)^2\right)$$

- If $\boldsymbol{\Sigma}$ is spherical

$$k(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{z}\|^2\right)$$

RBF kernel



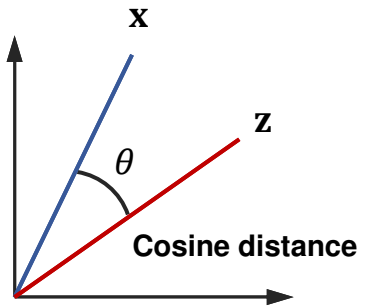


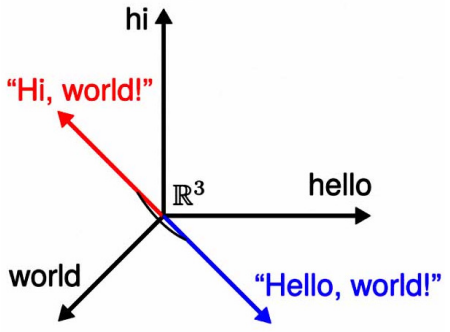
# Kernels for comparing text documents

- **Two text documents represented by vectors $\mathbf{x}$ and $\mathbf{z}$**
- **If $x_{ij}$ = the number of times word $j$ occurs in document $i$**

$$k(\mathbf{x}, \mathbf{z}) = \frac{\mathbf{x}^{\mathrm{T}}\mathbf{z}}{\|\mathbf{x}\|_2 \|\mathbf{z}\|_2} = \cos(\theta)$$

**is called the *cosine similarity* of the documents.**



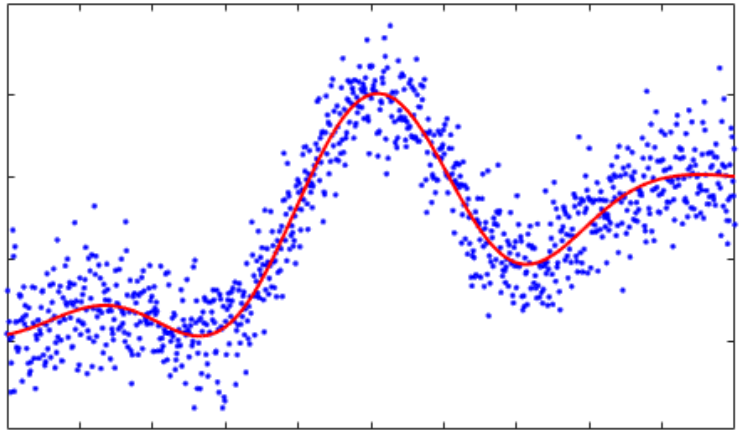
# Matern kernel

- **Commonly used in Gaussian processes**

$$k(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}r}{l}\right)^{\nu} B_\nu\left(\frac{\sqrt{2\nu}r}{l}\right) \rightarrow \textbf{SE kernel as } \nu \rightarrow \infty$$

**where** $r = \|\mathbf{x} - \mathbf{z}\|$, $\nu \geq 0$, $l > 0$ **and** $B_\nu$ **a modified Bessel function.**

$$k(r) = \exp(-r/l) \textbf{ when } \nu = 1/2$$



# String kernels

- **Consider two strings x and z of lengths $D_{\mathbf{x}}$ and $D_{\mathbf{z}}$, defined on a protein alphabet**
  - $\mathcal{A} = \{\text{A,R,N,D,C,E, Q, G,H, I, L,K,M, F, P, S, T,W, Y, V}\}$

  **where**

  **x ($D_{\mathbf{x}} = 110$):**

  IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTEEIFQGIGTL
  ESQTVQGGTVERLFKNLSLIKKYIDGQKKKCGEERRRVNQFLDYLQEFLGV
  MNTEWI

  **z ($D_{\mathbf{z}} = 153$):**

  PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAERLQENLQAY
  RTFHVLLARLLEDQQVHFTPTEGDFHQAIHTLLLQVAAFAYQIEELMILLE
  YKIPRNEADGMLFEKKLWGLKVLQELSQWTVRSIHDLRFISSHQTGIP

**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 $\mathcal{A}$.**

# String kernels

- If $w_s = 1$ for a nonempty substring $|s| > 0$:

  $\phi(\mathbf{x}) =$ number of times each char in $\mathcal{A}$ occurs in $\mathbf{x}$

  - bag-of-characters model
- If we require each substring $s$ to be surrounded by white space

  $\phi(\mathbf{x}) =$ number of times each word $s$ occurs in $\mathbf{x}$

  - bag-of-words model
- If we only consider strings with $|s| = k$ we get the $k$-spectrum kernel

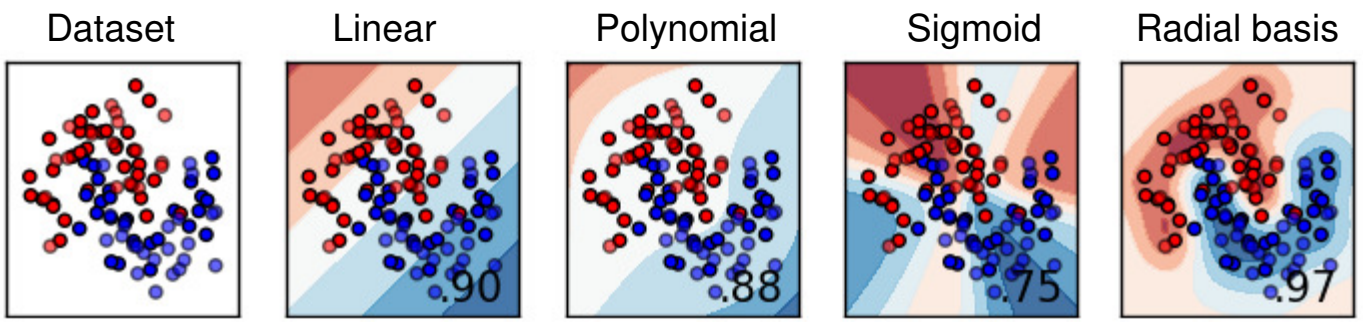
**Similarity measure:**
**number of common substrings**

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

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

# Which kernel to use?

- **Start with the simplest one and work your way up**



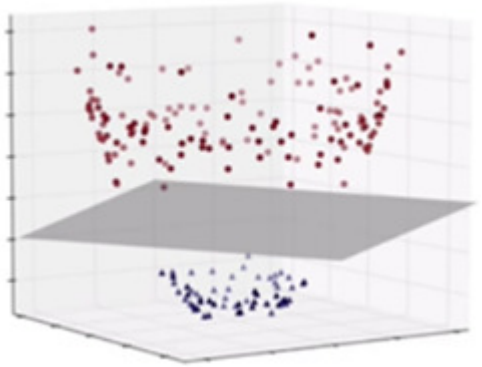
# One-vs-rest trick

- SVMs use hyperplanes to split spaces into two halves. They are binary classifiers.
- To classify non-binary datasets, we can use a simple trick:
  - Example: labels of apples, oranges and bananas
  - Make three binary classifiers:
    - Apples vs non-apples
    - Oranges vs non-oranges
    - Bananas vs non-bananas
  - If labelled as ’apple’ in the first, return ’apple’, otherwise put through next classifier, etc. Label ’other’ is also possible.

# Quick evaluation of SVMs

**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**

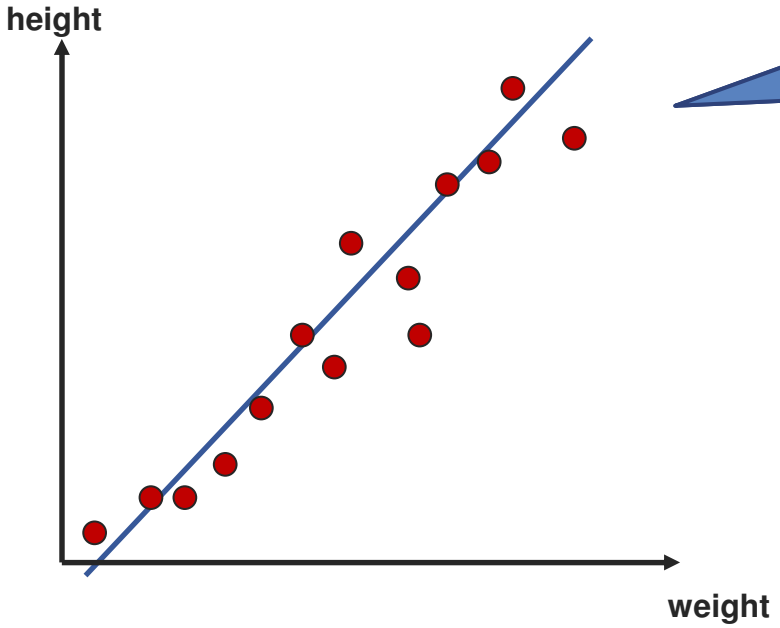


**Disadvantages**

- **Slow on large datasets (compared to Naïve Bayes)**
- **Works poorly with overlapping classes**
- **Kernel type must be selected manually.**

# Logistic regression

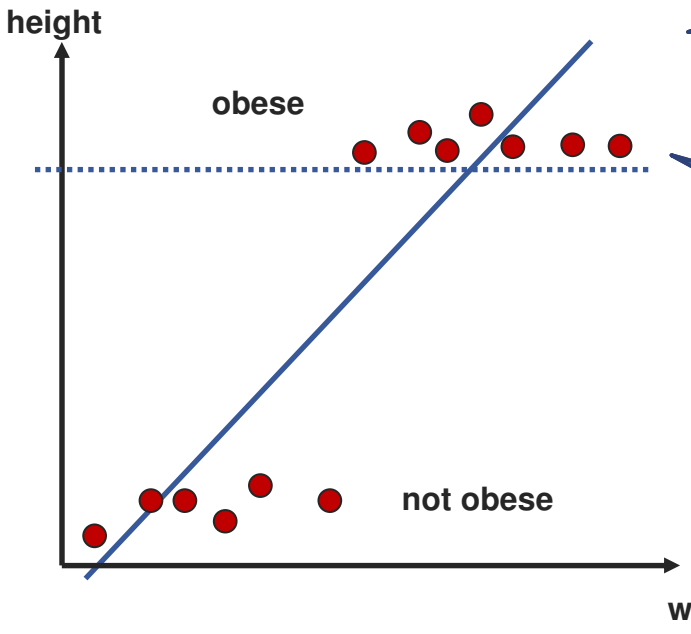
## Example: obese mice




In linear regression we attempt to fit a linear model to observed data:
$y = ax + b$

# Logistic regression

## Example: obese mice



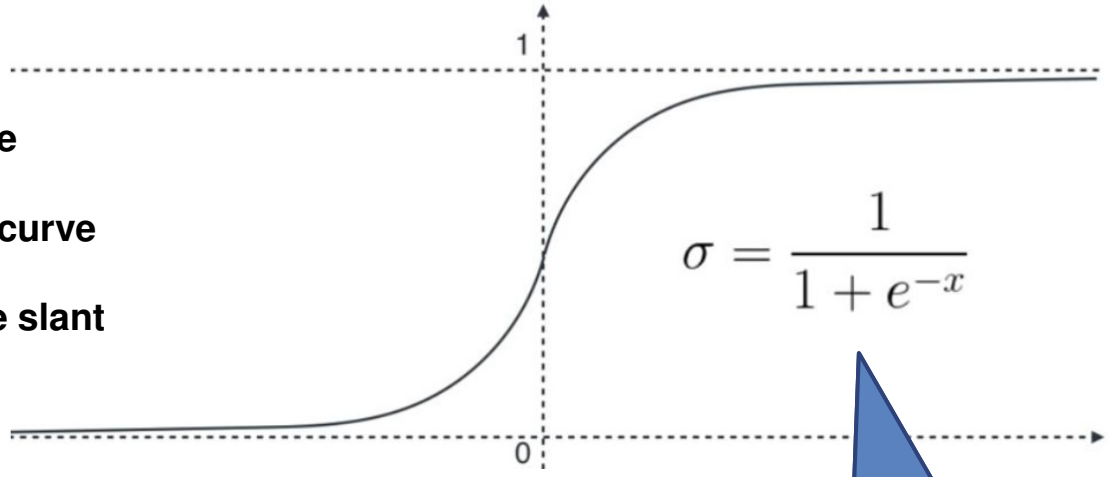

Assume now that we add another attribute: obesity

The regression line does not manage to separate this two discrete sets of points

We see that the probability of obesity increases with weight

# The logistic function

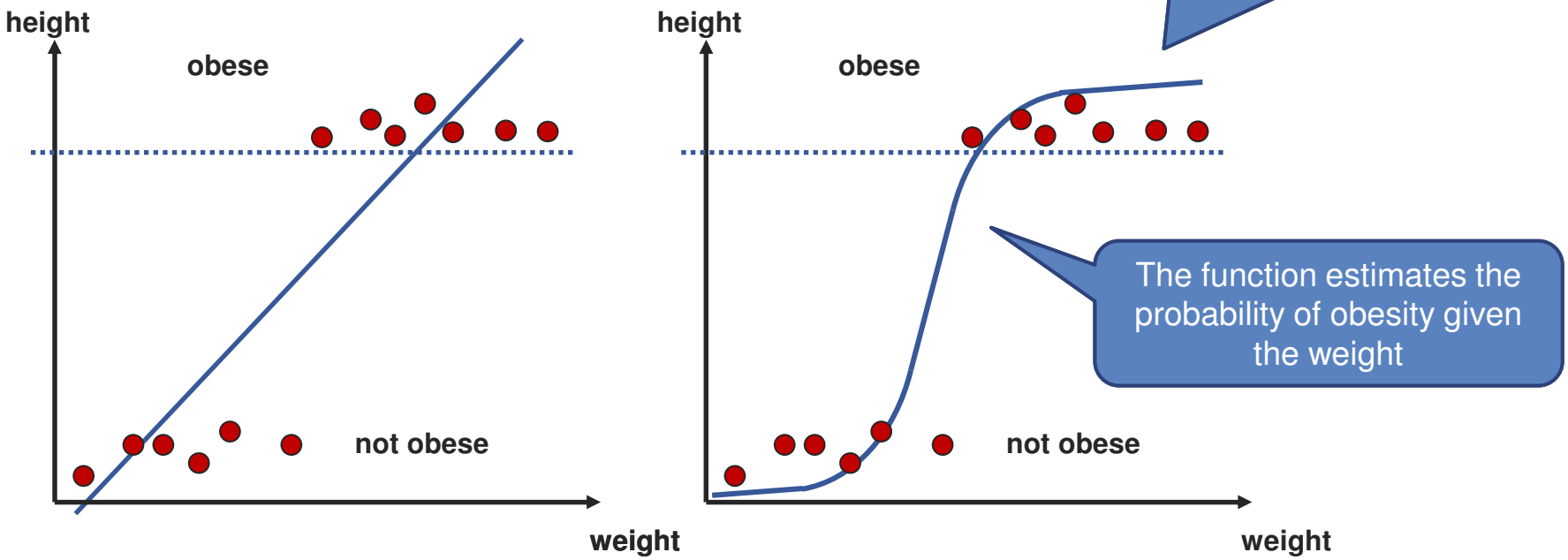
- **Squishes all numbers into the ”probability range” (0,1)**
- **Using $x + w$ we can shift the curve left or right.**
- **Using $kx + w$ we can vary the slant as well**




Also called the sigmoid function

# Logistic regression

## Example: obese mice

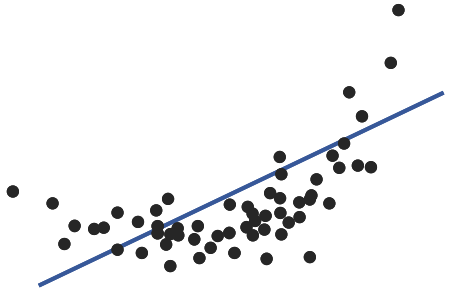


# Gaussian processes

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

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

  **where $f$ is assumed to be linear: $f(x) = \beta_0 + \beta_1 x$**
- **Bayesian linear regression: determine a posterior distribution over the parameters $\beta_0$ and $\beta_1$ that gets updated whenever new data is available.**
- **Gaussian processes: finds a posterior distribution over the possible *functions* $f(x)$ consistent with the observed data and a suitable prior.**

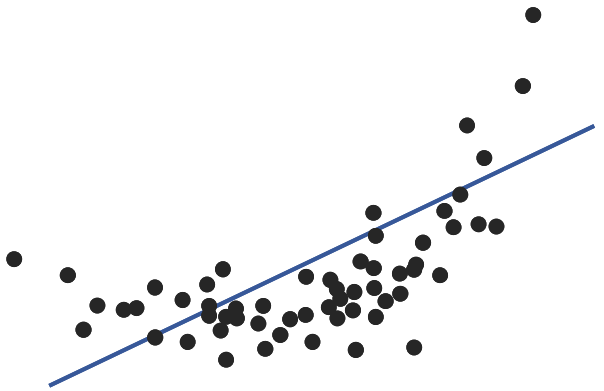


# Gaussian processes

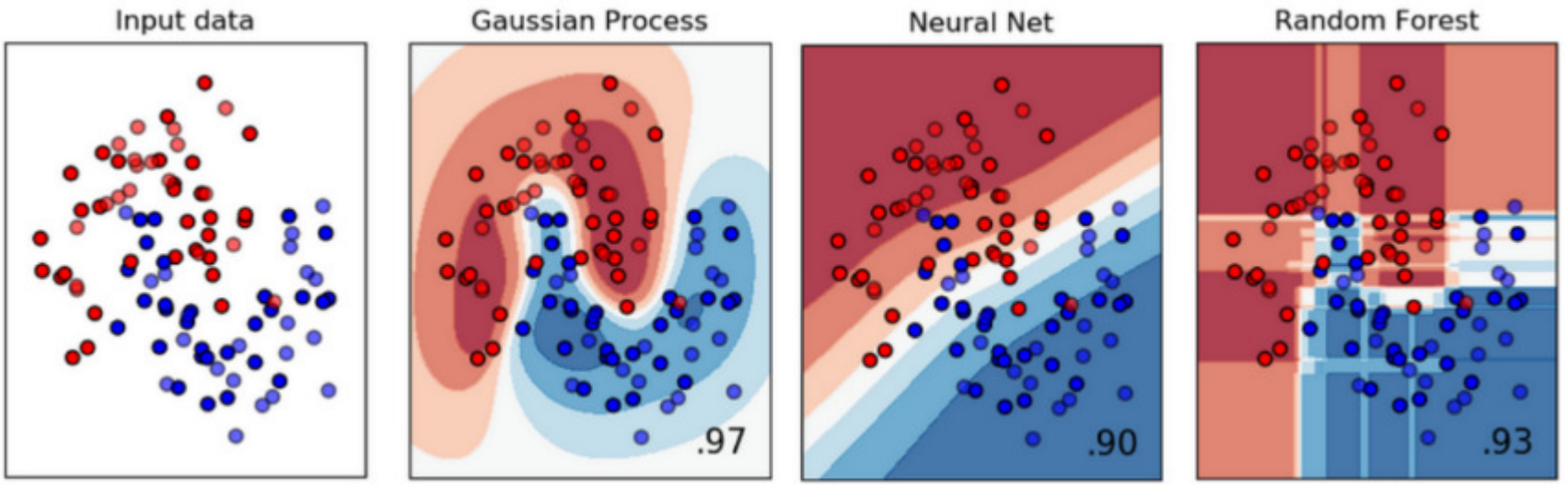
- **The current example isn't really linear.**
- **Quadratic function**

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$$

- **Three parameters to estimate:** $\beta_0, \beta_1, \beta_2$
- **But what if we don't know how many parameters we should use?**
- **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

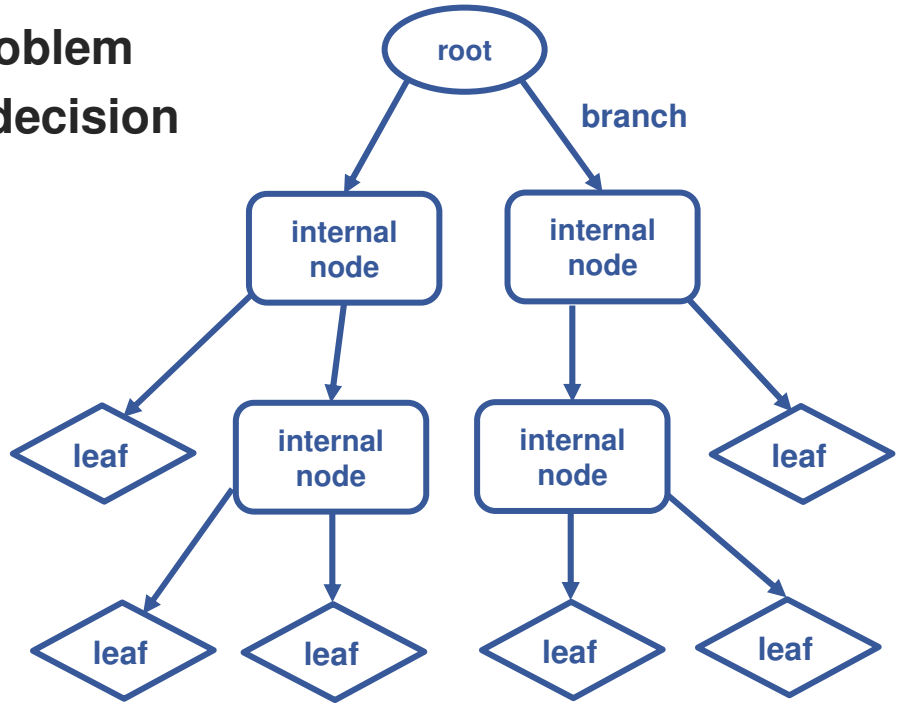


# Decision trees

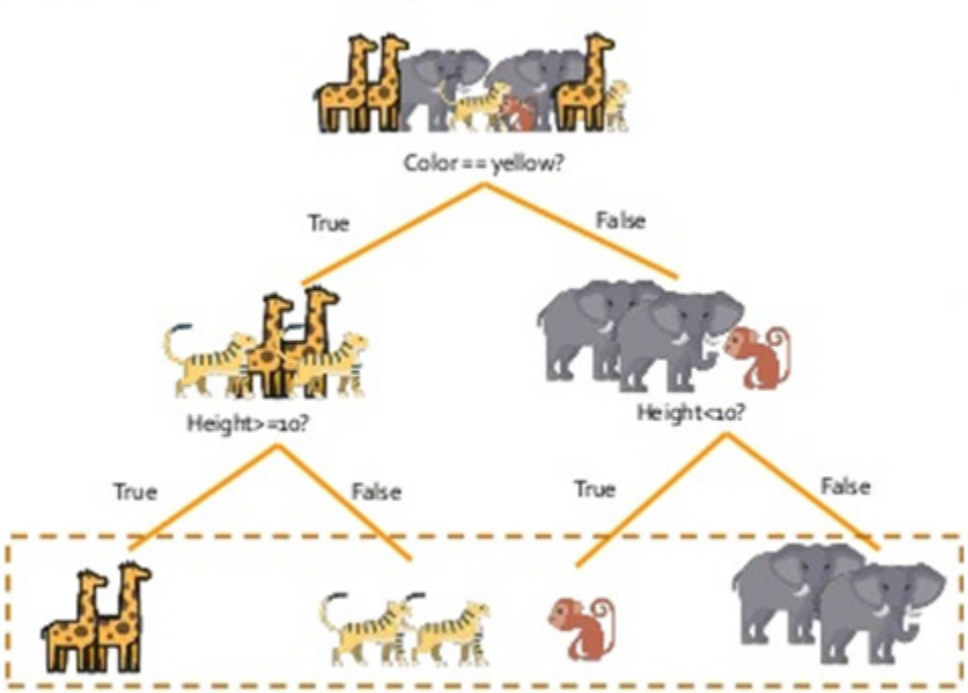
- A **decision tree** is a tree shaped diagram used to determine a course of action.
- It can be used for decision making and classification

# Decision trees

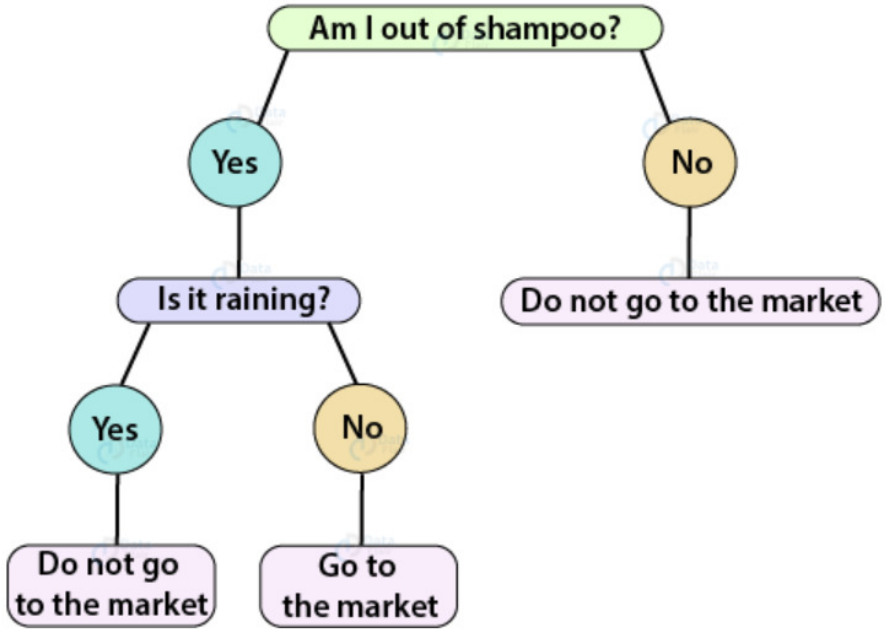
- Each **node** represents a question/problem
- Each **branch** represents an answer/decision
- Each **leaf** represents a class label



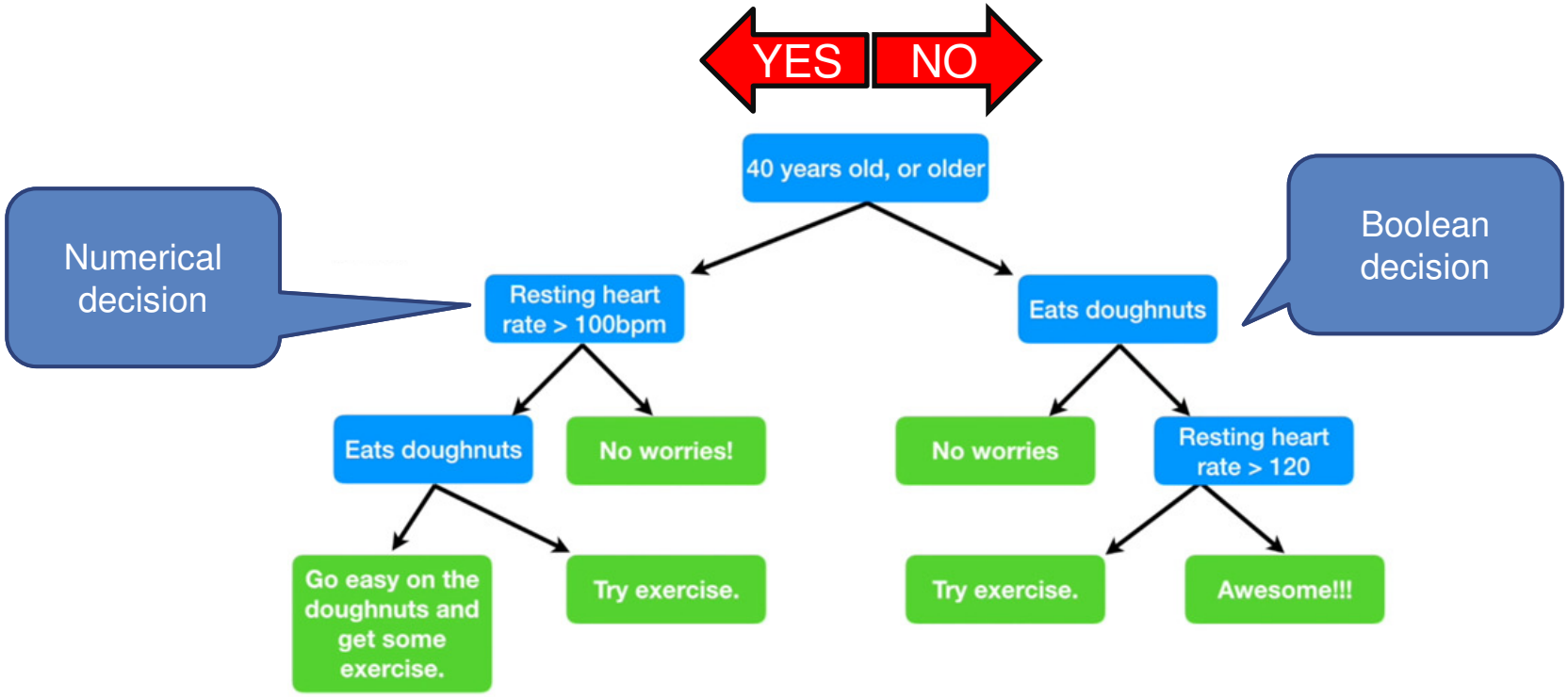
# Example: classification of animals



# Example: classification of activities

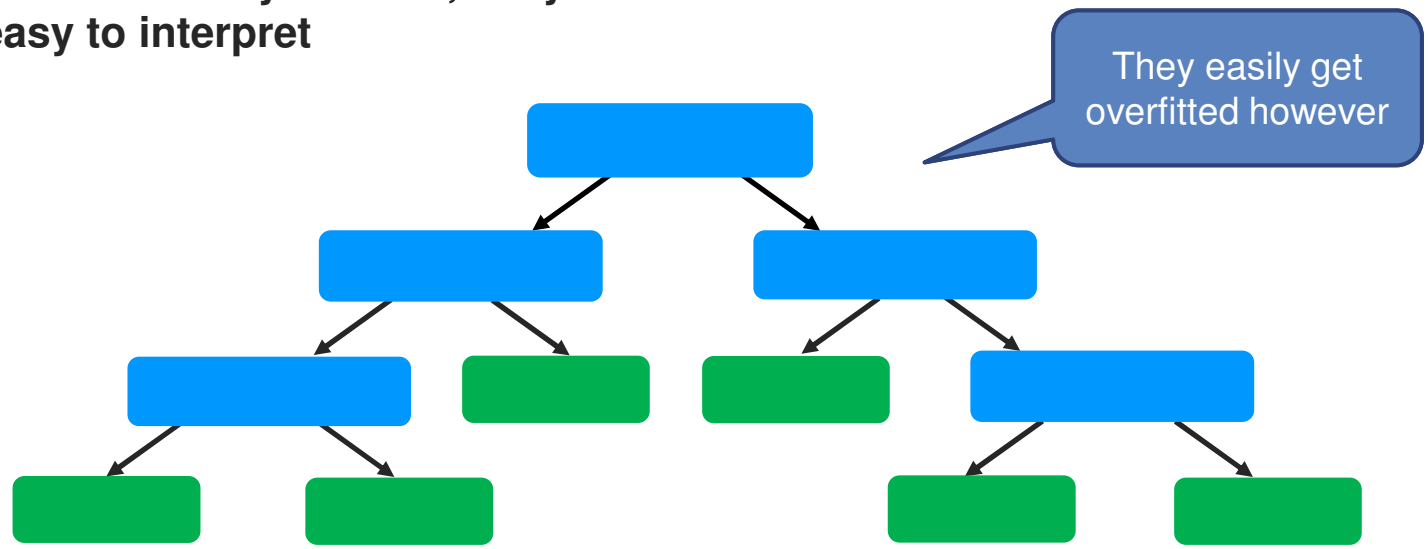


# Example: (questionable) health advice



# Decision trees – advantages

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



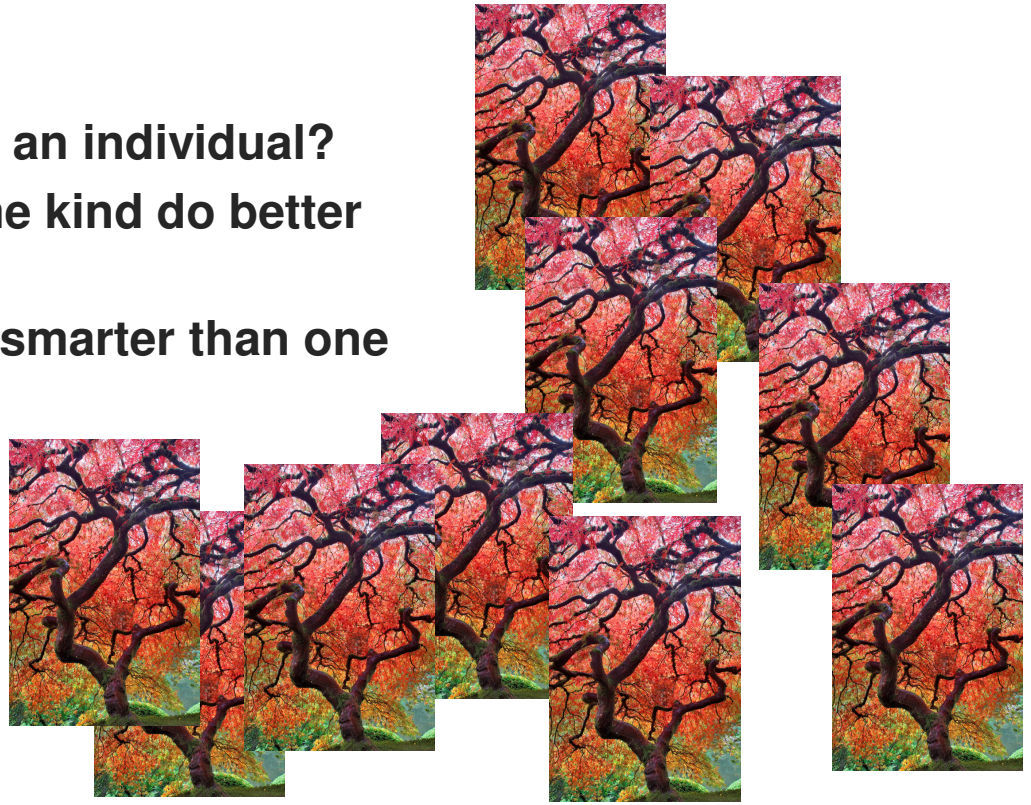
# Decision trees – drawbacks

To quote from ***The Elements of Statistical Learning*** (aka The Bible of Machine Learning), "Trees have one aspect that prevents them from being the ideal tool for predictive learning, namely **inaccuracy**."

In other words, they work great with the data used to create them, but **they are not flexible when it comes to classifying new samples**.

# Ensemble methods

- Can a crowd be smarter than an individual?
- Can many models of the same kind do better than one?
- Can many trees (a forest) be smarter than one tree?

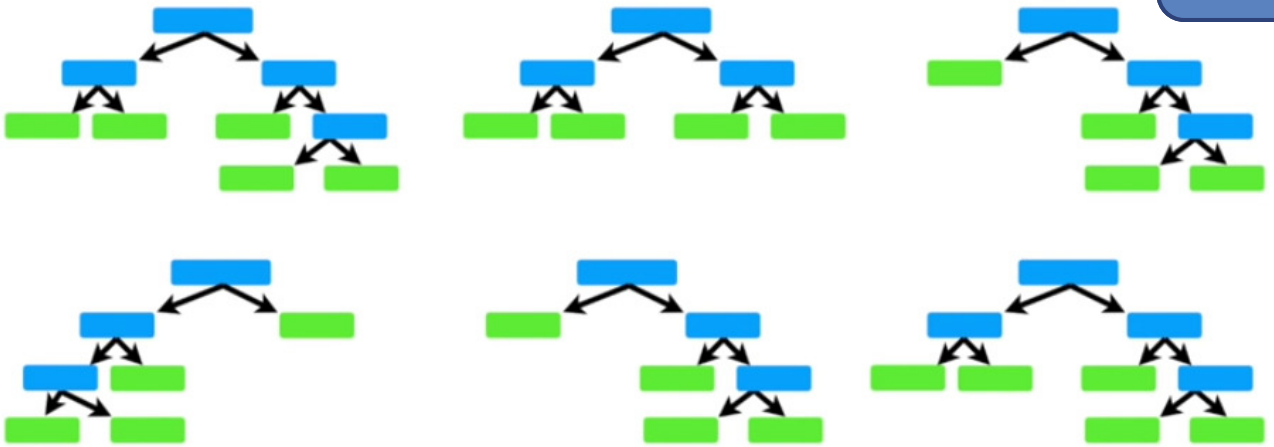


# Random forests

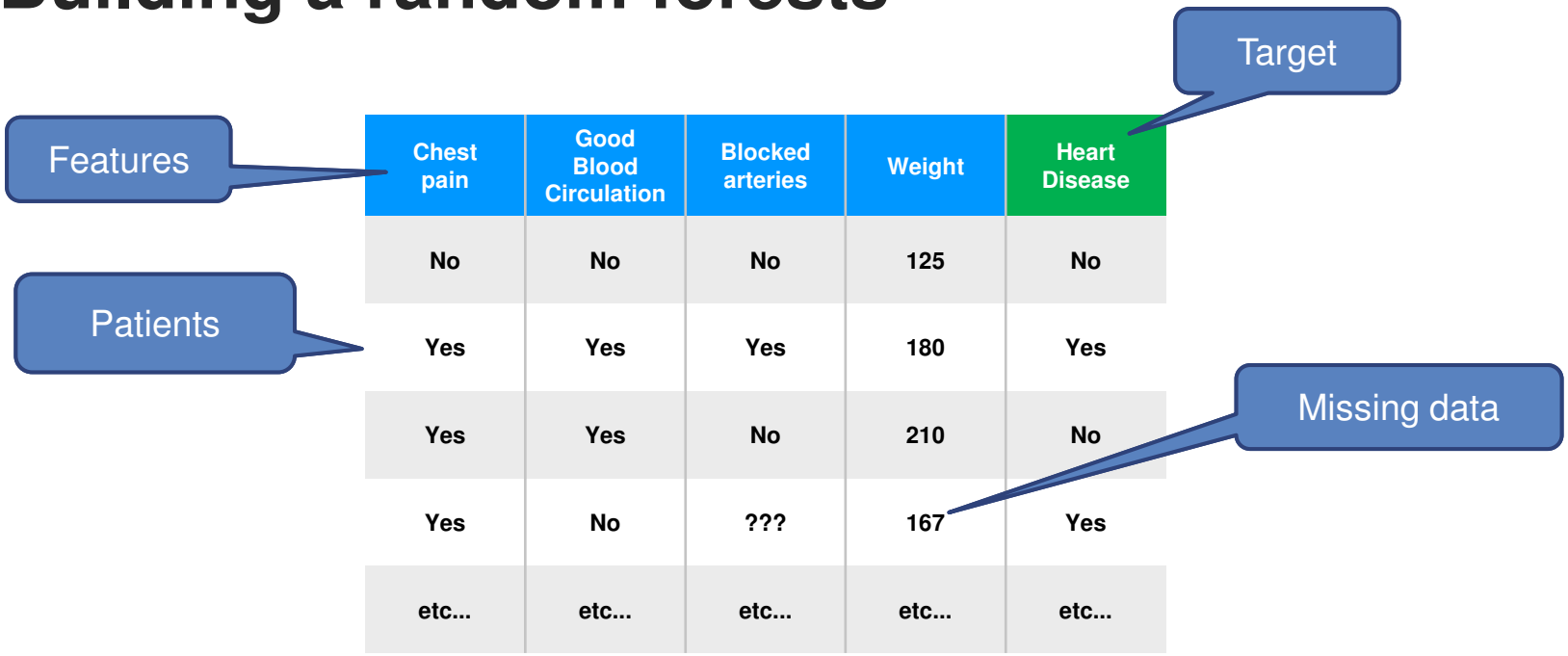
The good news is that **Random Forests** combine the simplicity of decision trees with flexibility resulting in a vast improvement in accuracy.

A random forest is a set of trees that ”vote” to produce a decision

Random forests is an ensemble method



# Building a random forests




| Chest pain | Good Blood Circulation | Blocked arteries | Weight | Heart Disease |
|---|---|---|---|---|
| No | No | No | 125 | No |
| Yes | Yes | Yes | 180 | Yes |
| Yes | Yes | No | 210 | No |
| Yes | No | ??? | 167 | Yes |
| etc... | etc... | etc... | etc... | etc... |

# Building random forests

Step 1: create a bootstrapped dataset by randomly picking lines from the dataset

## Original dataset

| Chest pain | Good Blood Circulation | Blocked arteries | Weight | Heart Disease |
|---|---|---|---|---|
| No | No | No | 125 | No |
| Yes | Yes | Yes | 180 | Yes |
| Yes | Yes | No | 210 | No |
| Yes | No | Yes | 167 | Yes |
| etc... | etc... | etc... | etc... | etc... |

## Bootstrapped dataset

| Chest pain | Good Blood Circulation | Blocked arteries | Weight | Heart Disease |
|---|---|---|---|---|
| Yes | Yes | Yes | 180 | Yes |
| No | No | No | 125 | No |
| Yes | No | Yes | 167 | Yes |
| Yes | No | Yes | 167 | Yes |

Same line twice

# Building random forests

## Bootstrapped dataset

| Chest pain | Good Blood Circulation | Blocked arteries | Weight | Heart Disease |
|---|---|---|---|---|
| Yes | Yes | Yes | 180 | Yes |
| No | No | No | 125 | No |
| Yes | No | Yes | 167 | Yes |
| Yes | No | Yes | 167 | Yes |

Step 2: create a decision tree of the bootstrapped dataset, but only use a random subset of features (columns) at each step

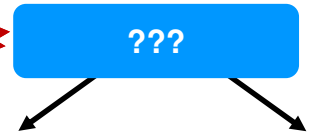
E.g. use only two columns in each step here

# Building random forests

We randomly selected **Good Blood Circulation** and **Blocked Arteries** as root candiddates

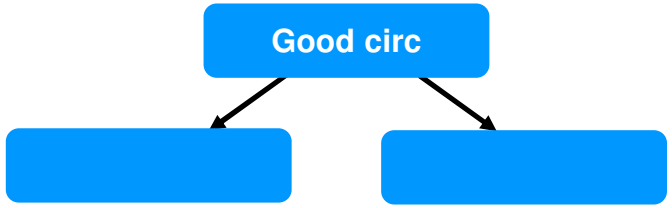
## Bootstrapped dataset

| Chest pain | Good Blood Circulation | Blocked arteries | Weight | Heart Disease |
|---|---|---|---|---|
| Yes | Yes | Yes | 180 | Yes |
| No | No | No | 125 | No |
| Yes | No | Yes | 167 | Yes |
| Yes | No | Yes | 167 | Yes |



# Building random forests

**Assume that Good Blood Circulation did the best job in separating samples**

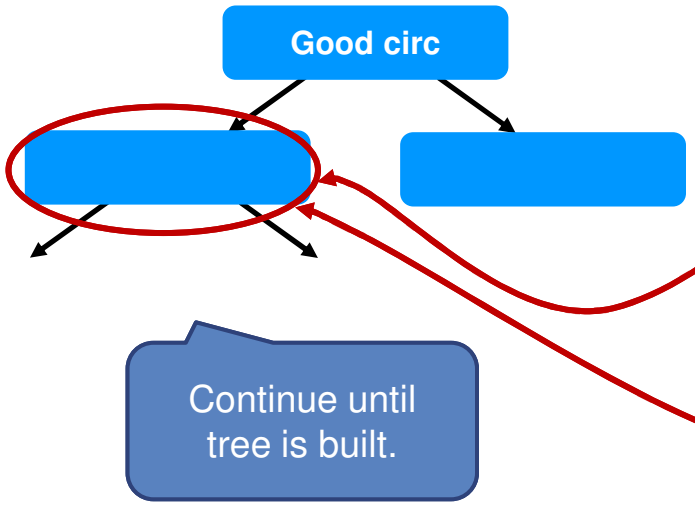



**Bootstrapped dataset**

| Chest pain | Good Blood Circulation | Blocked arteries | Weight | Heart Disease |
|---|---|---|---|---|
| Yes | Yes | Yes | 180 | Yes |
| No | No | No | 125 | No |
| Yes | No | Yes | 167 | Yes |
| Yes | No | Yes | 167 | Yes |

# Building random forests

**For the next level we randomly select two features again**




**Bootstrapped dataset**

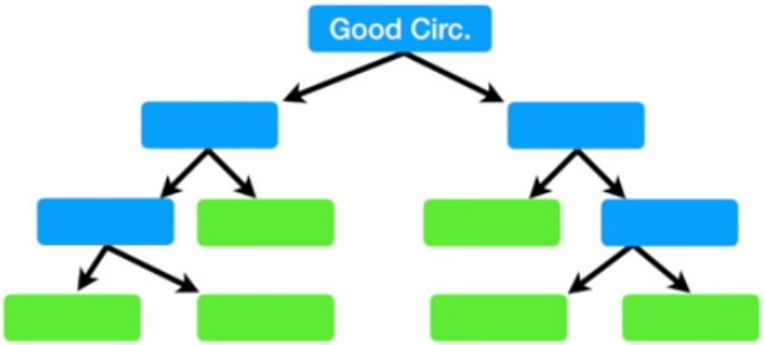
| Chest pain | Good Blood Circulation | Blocked arteries | Weight | Heart Disease |
|---|---|---|---|---|
| Yes | Yes | Yes | 180 | Yes |
| No | No | No | 125 | No |
| Yes | No | Yes | 167 | Yes |
| Yes | No | Yes | 167 | Yes |

# Building random forests

We built a tree…

1) Using a bootstrapped dataset

2) Only considering a random a subset of variables at each step.

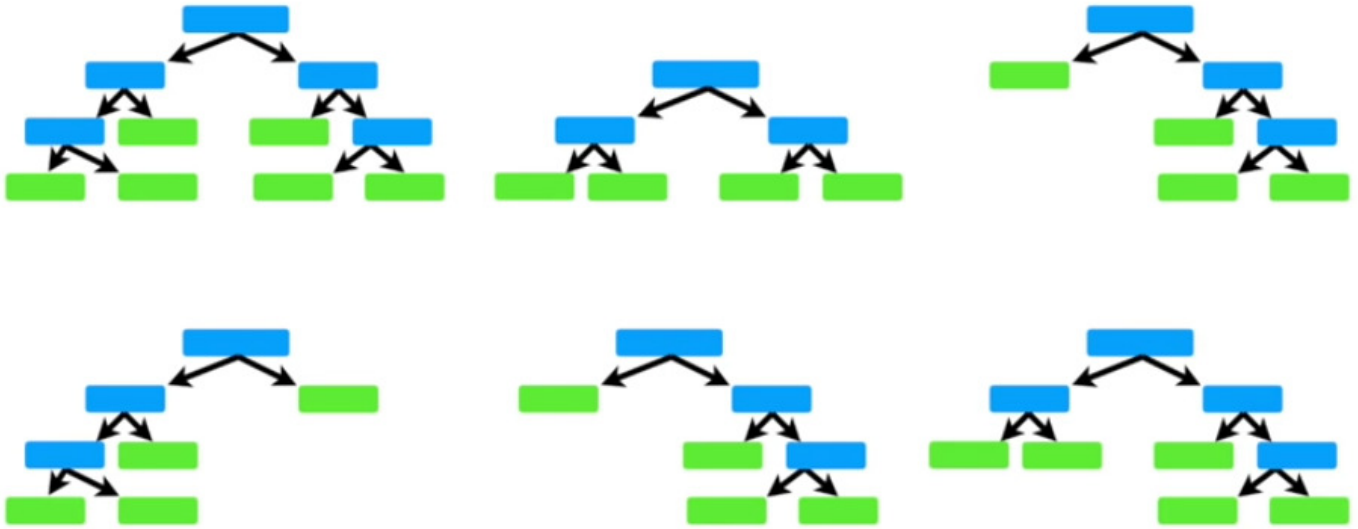



Bootstrapped Dataset

| Chest Pain | Good Blood Circ. | Blocked Arteries | Weight | Heart Disease |
|---|---|---|---|---|
| Yes | Yes | Yes | 180 | Yes |
| No | No | No | 125 | No |
| Yes | No | Yes | 167 | Yes |
| Yes | No | Yes | 167 | Yes |

# Building random forests

**Now go back to Step 1 and repeat:** Make a new bootstrapped dataset and build a tree considering a subset of variables at each step.



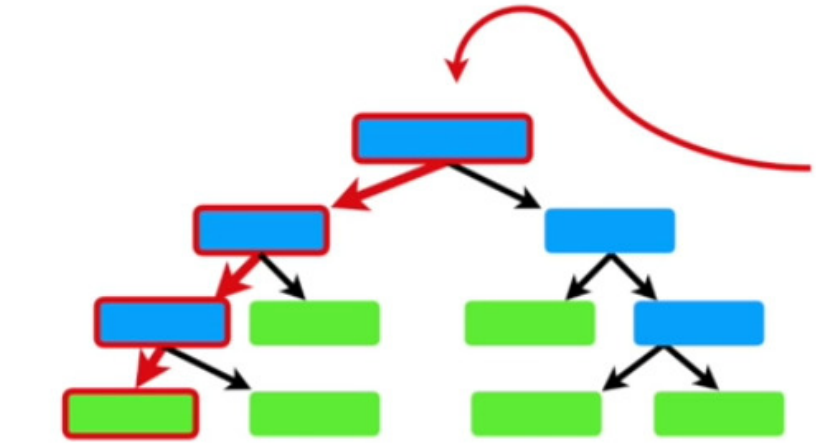
# Using random forests

**Suppose we have this new patient…**

| Chest Pain | Good Blood Circ. | Blocked Arteries | Weight | Heart Disease |
|---|---|---|---|---|
| Yes | No | No | 168 | |

…and now we want to know if they have heart disease or not.

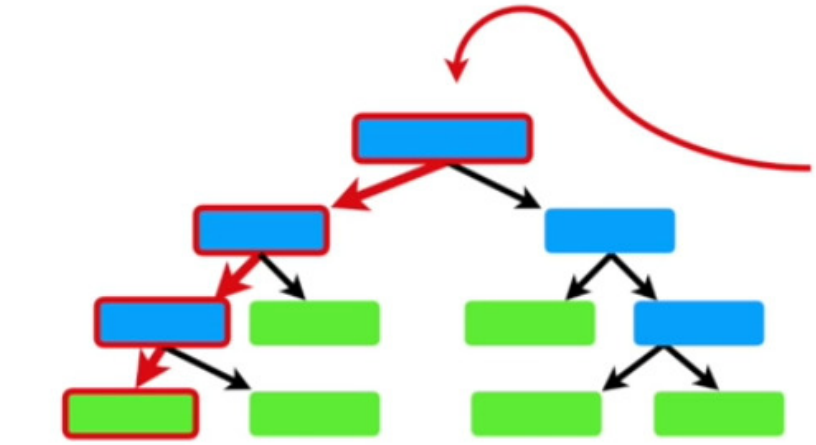
# Using random forests



| Chest Pain | Good Blood Circ. | Blocked Arteries | Weight | Heart Disease |
|---|---|---|---|---|
| Yes | No | No | 168 | |

The first tree says "Yes"…

# Using random forests



The first tree says "Yes"…

| Chest Pain | Good Blood Circ. | Blocked Arteries | Weight | Heart Disease |
|---|---|---|---|---|
| Yes | No | No | 168 | |

| Heart disease | |
|---|---|
| Yes | No |
| 1 | 0 |

Keeping track of the votes

# Using random forests

Check the decisions of the other trees as well

| Chest Pain | Good Blood Circ. | Blocked Arteries | Weight | Heart Disease |
|---|---|---|---|---|
| Yes | No | No | 168 | YES |

| Heart disease | |
|---|---|
| Yes | No |
| 5 | 1 |

All 6 trees have voted! Then we make an aggregated decision.

# Using random forests

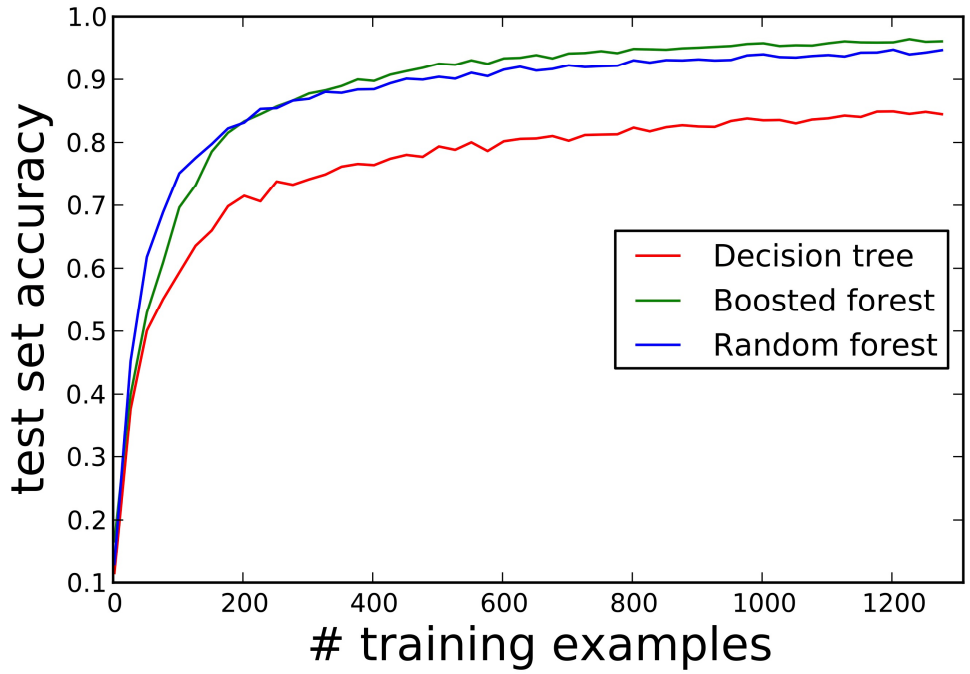
**In other words…**

**… change the number of variables used per step…**

1) **Build a Random Forest**
2) **Estimate the accuracy of a Random Forest**

**Typically, we start by using the square root of the number of variables and then try a few settings above and below that value**

# Forests vs trees



# Random forests

## Advantages

- Random forests are considered a highly accurate and robust method because of the number of decision trees participating in the process.
- It 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.

# Random forests

## Disadvantages

- Random forests is **slow** in generating predictions because it has multiple decision trees. Whenever it makes a prediction, all the trees in the forest have to make a prediction for the same given input and then perform voting on it. This whole process is time-consuming.
- 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.

# Comparing classification methods

- **Accuracy**
- **Scalability**
- **Manual steps**
- **Multiclass or binary class**
- **Interpretability/explainability**

Comparison of 14 different families of classification algorithms on 115 binary datasets

# Comparing classification methods

Search this file...

| | Gender | Age | Salary | Purchased Iphone |
|---|---|---|---|---|
| 2 | Male | 19 | 19000 | 0 |
| 3 | Male | 35 | 20000 | 0 |
| 4 | Female | 26 | 43000 | 0 |
| 5 | Female | 27 | 57000 | 0 |
| 6 | Male | 19 | 76000 | 0 |
| 7 | Male | 27 | 58000 | 0 |
| 8 | Female | 27 | 84000 | 0 |
| 9 | Female | 32 | 150000 | 1 |

Neural networks were not included

```
Logistic Regression: Mean Accuracy = 82.75% — SD Accuracy = 11.37%
K Nearest Neighbor: Mean Accuracy = 90.50% — SD Accuracy = 7.73%
Kernel SVM: Mean Accuracy = 90.75% — SD Accuracy = 9.15%
Naive Bayes: Mean Accuracy = 85.25% — SD Accuracy = 10.34%
Decision Tree: Mean Accuracy = 84.50% — SD Accuracy = 8.50%
Random Forest: Mean Accuracy = 88.75% — SD Accuracy = 8.46%
```

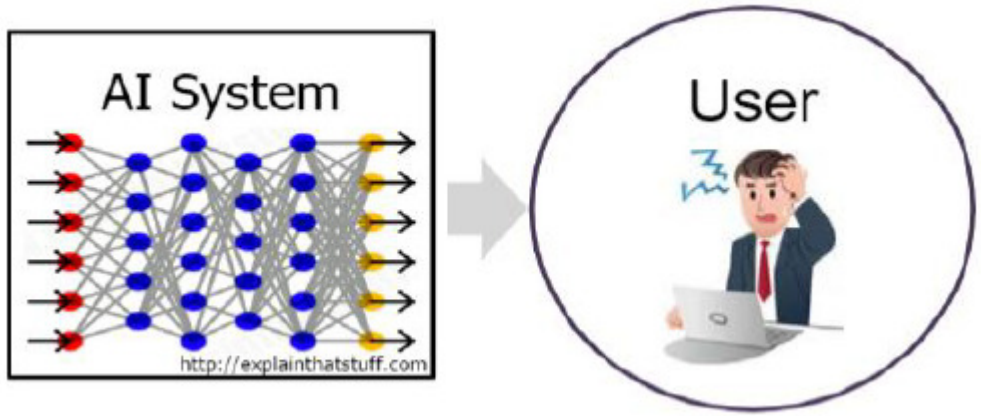
In general no outstanding winner overall. Depends on domain and other.

Best accuracy: SVMs and random forests.

# Explainable decision support

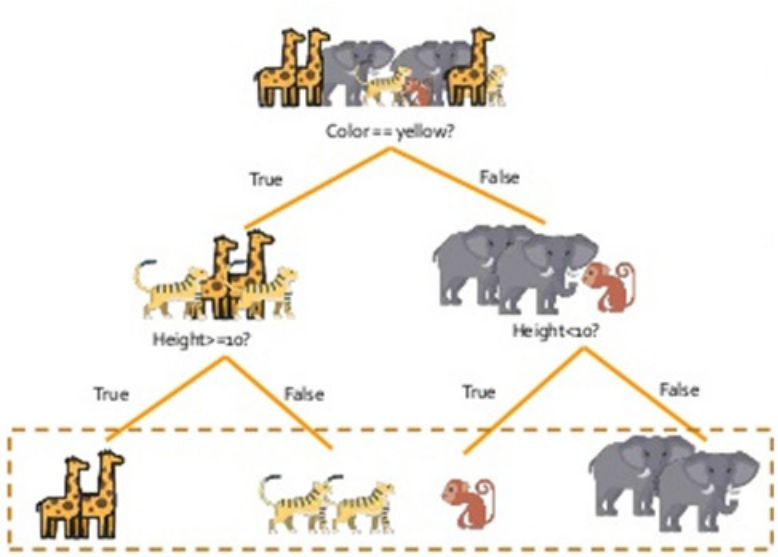
## ”I did it because the machine said so”

- Maybe OK when recommending a song
- But not OK in medical decision making and many other situations



# Explainable decision support

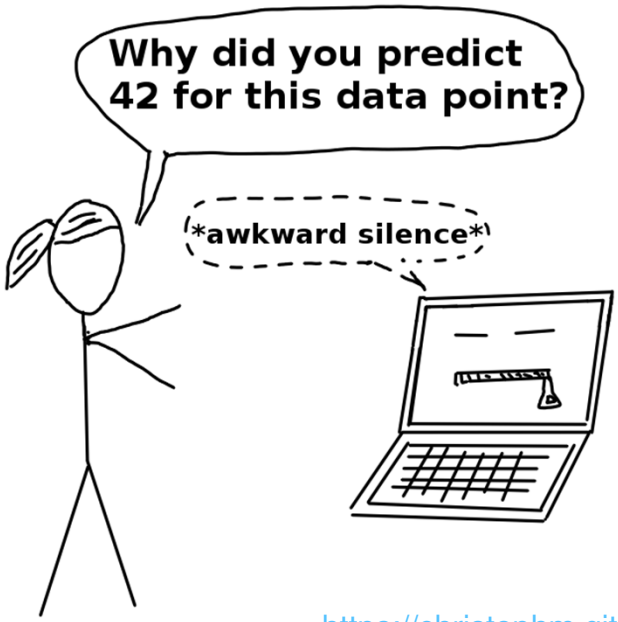
**More explainable**




**Less explainable**

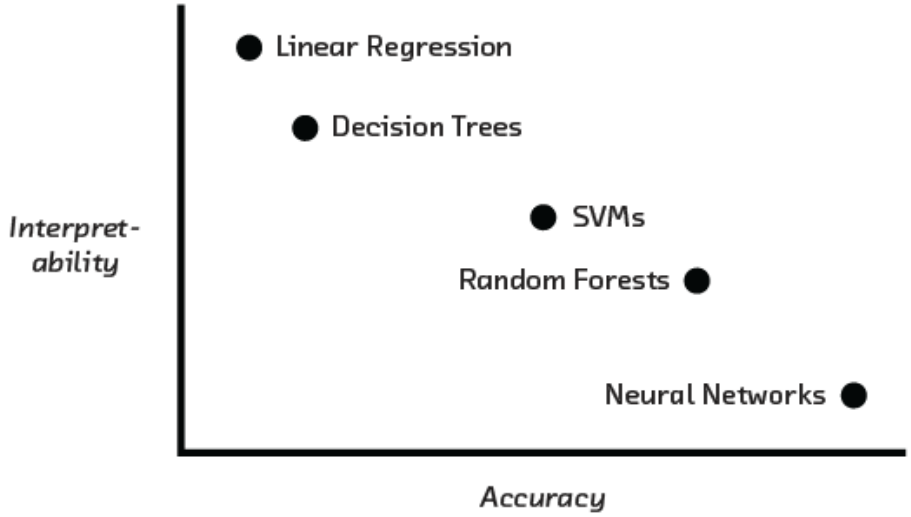
$$\text{species} = \text{round}(3.4 \cdot \text{color}^2 - 8.6 \cdot \text{height})$$

# Explainable decision support



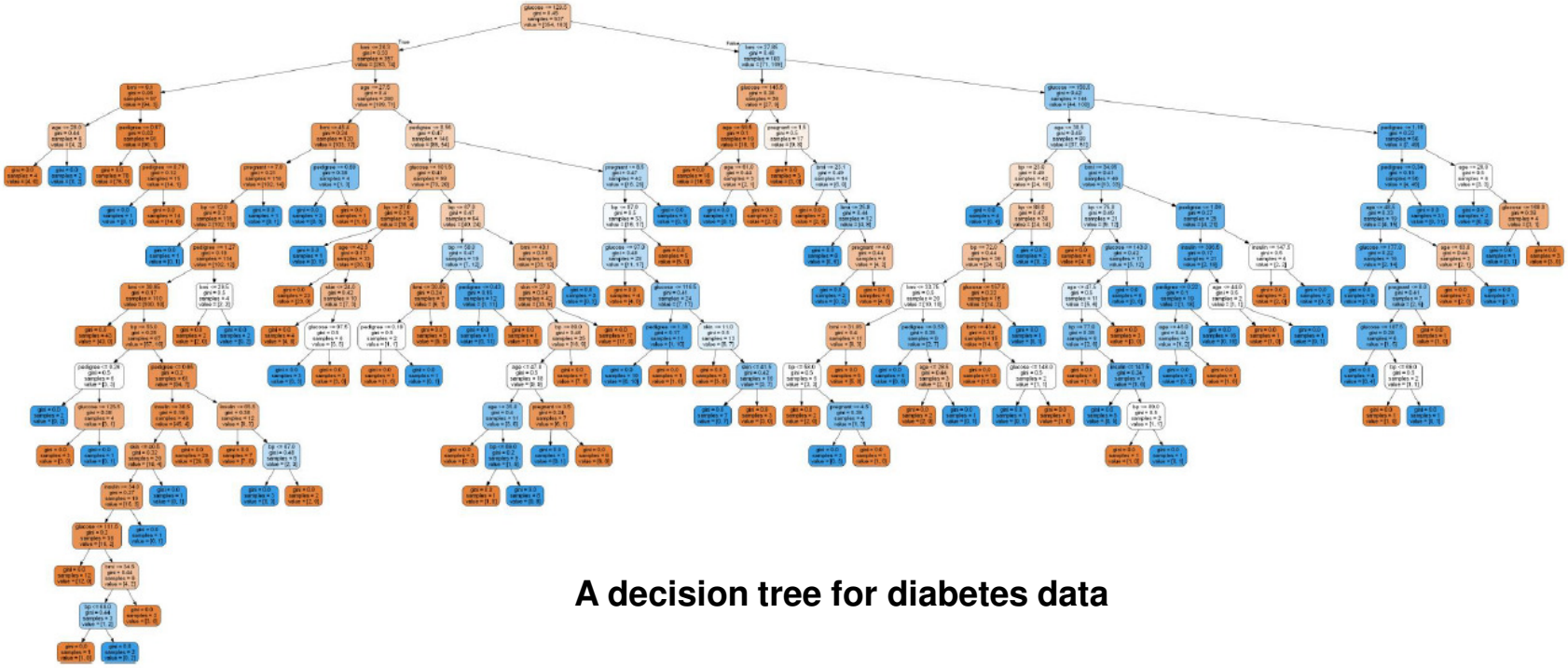

https://christophm.github.io/interpretable-ml-book/terminology.html

# Explainable decision support




https://ff06-2020.fastforwardlabs.com/

# Explainability in practice?



**A decision tree for diabetes data**