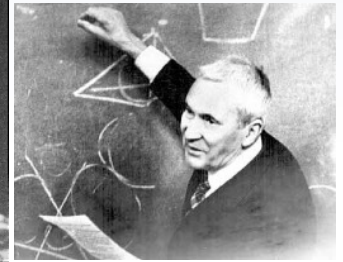
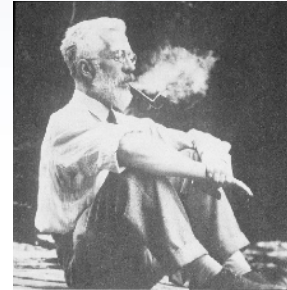
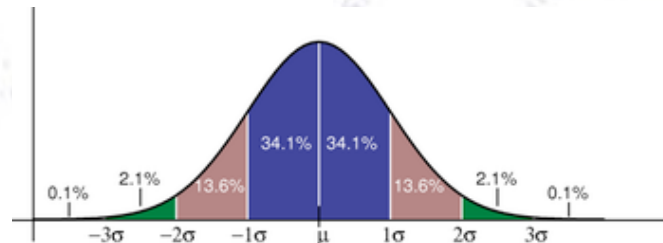


# Machine Learning

## An introduction



Troels C. Petersen (NBI)

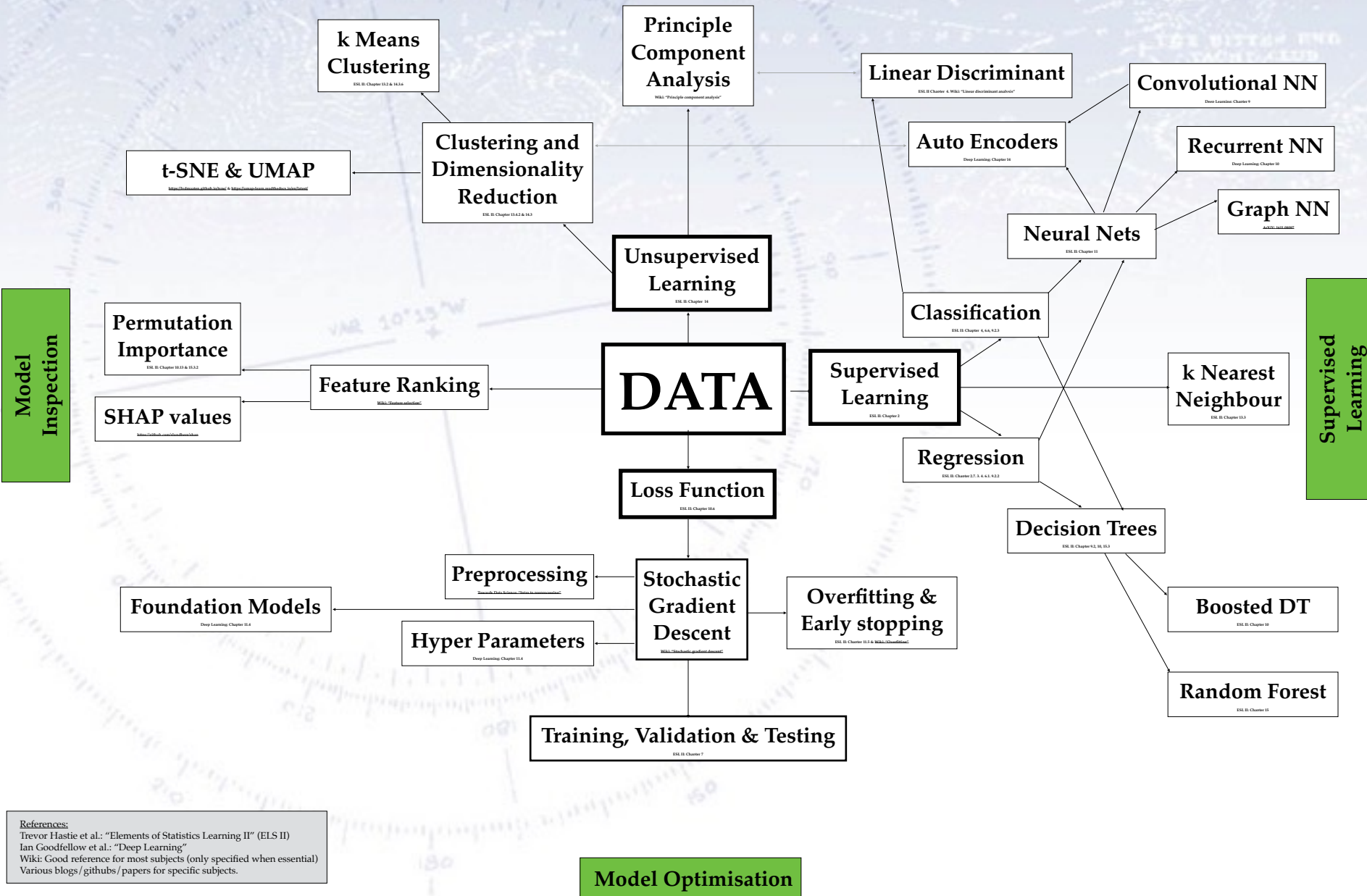


*"Statistics is merely a quantisation of common sense - Machine Learning is a sharpening of it!"*

# Applied Machine Learning

# Overview of subjects

Version 1.4, 29. March 2025



The background is a faded nautical chart. It features a large compass rose with degree markings from 0 to 360. Concentric circles around the center represent depth contours, with labels like 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330, 340, 350, 360, 370, 380, 390, 400, 410, 420, 430, 440, 450, 460, 470, 480, 490, 500, 510, 520, 530, 540, 550, 560, 570, 580, 590, 600, 610, 620, 630, 640, 650, 660, 670, 680, 690, 700, 710, 720, 730, 740, 750, 760, 770, 780, 790, 800, 810, 820, 830, 840, 850, 860, 870, 880, 890, 900, 910, 920, 930, 940, 950, 960, 970, 980, 990, 1000. The word 'MAGNETIC' is visible in the upper left quadrant. In the upper right quadrant, the text '1ST BITTER END YACHT CLUB' is visible. The overall image is a light blue and white color scheme.

# What is ML?

# What is Machine Learning?

While there is no formal definition, an early attempt is the following intuition:

“Machine learning programs can perform tasks without being explicitly programmed to do so.”

[Arthur Samuel, US computer pioneer 1901-1990]



# What is Machine Learning?

While there is no formal definition, an early attempt is the following intuition:

“Machine learning programs can perform tasks without being explicitly programmed to do so.”

[Arthur Samuel, US computer pioneer 1901-1990]

“Little Peter is capable of finding his way home without being explicitly taught to do so.”

# What is Machine Learning?

While there is no formal definition, an early attempt is the following intuition:

"A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$  if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ ."

[T. Mitchell, "Machine Learning" 1997]

# What is Machine Learning?

While there is no formal definition, an early attempt is the following intuition:

"A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$  if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ ."

[T. Mitchell, "Machine Learning" 1997]

"Little Peter is said to learn from traveling around with respect to finding his way home and the time it takes, if his ability to find his way home, as measured by the time it takes, improves as he travels around."

# What is Machine Learning?

While there is no formal definition, an early attempt is the following intuition:

"A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$  if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ ."

[T. Mitchell, "Machine Learning" 1997]

"Little Peter is said to learn from traveling around with respect to finding his way home and the time it takes, if his ability to find his way home, as measured by the time it takes, improves as he travels around."

Under all circumstances, ML allows the analysis and understanding of data, that is complex in terms of both size, dimensionality, quality, and relations [TP].

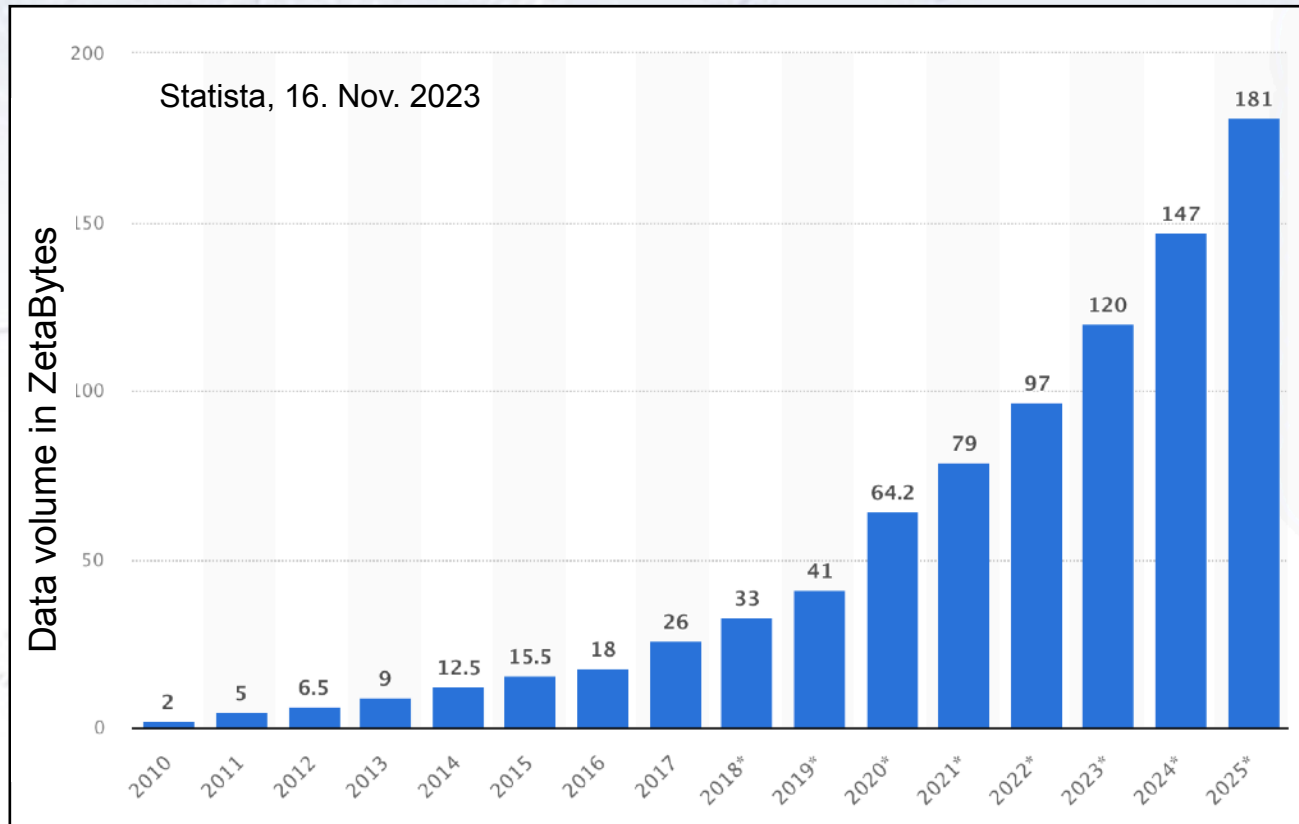




# Why ML?

# The Data “Deluge”

Shown is the total amount of data created, captured, and copied.



However, only a small portion (~2%) of this data is stored for a longer period of time.

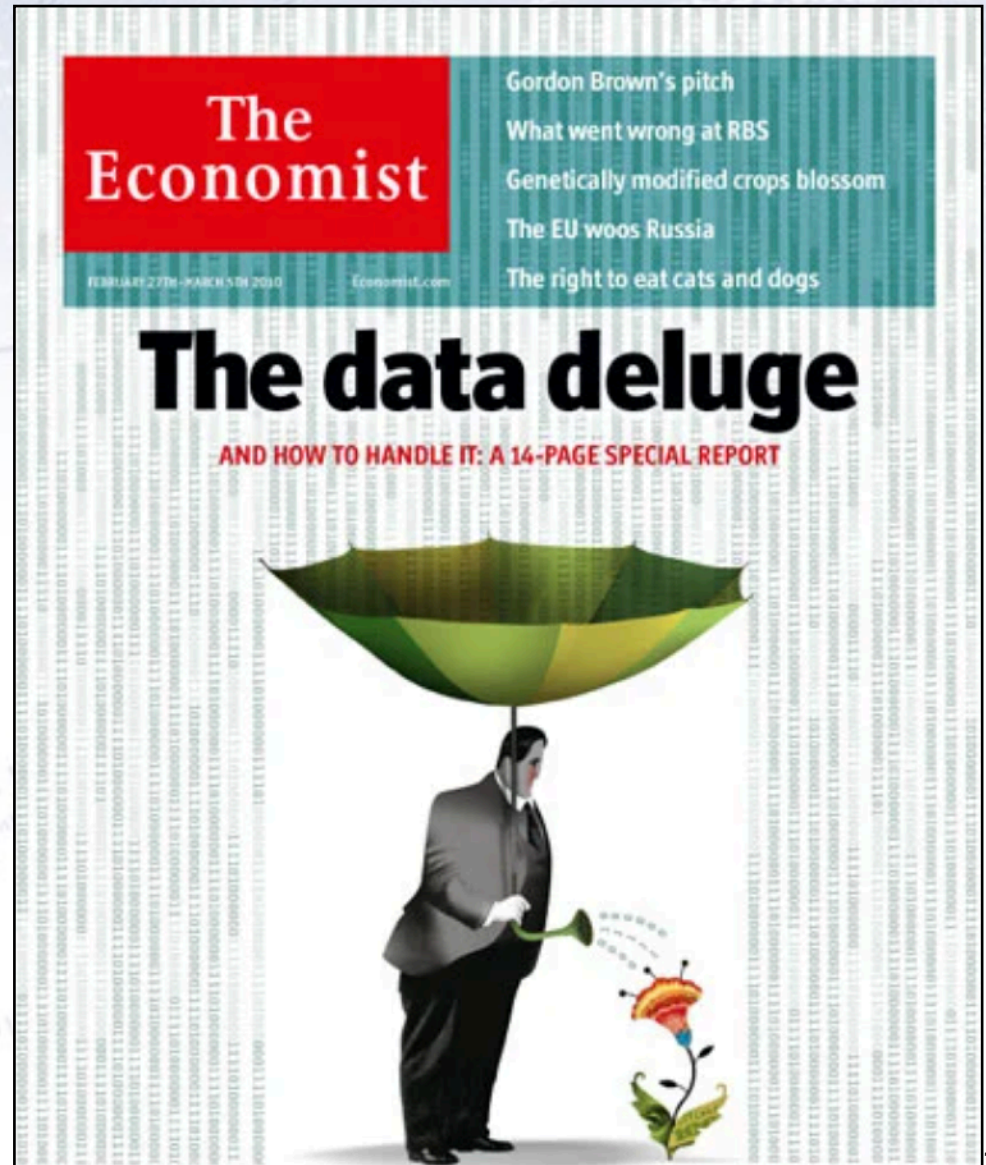
# ML and the Data “Deluge”

The amount of data in the world is growing very fast.

In order to consider this data, Machine Learning (ML) has become a standard tool.

This is due to the easy access to large data sets, but also the growth in data storage and processing capabilities.

By now ML can also consider text, images, sound, etc.





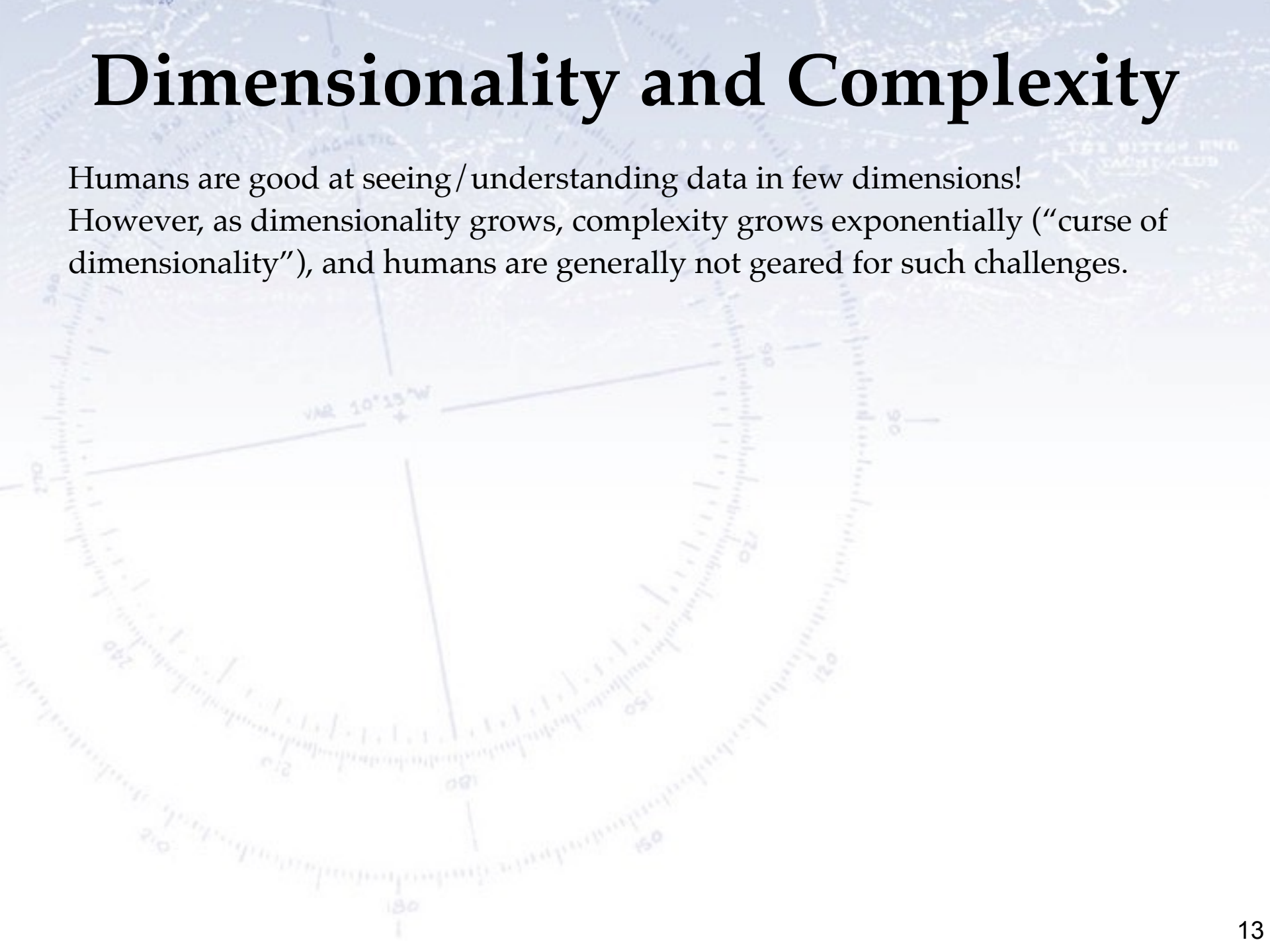
# Humans vs. ML



# Dimensionality and Complexity

Humans are good at seeing/understanding data in few dimensions!

However, as dimensionality grows, complexity grows exponentially (“curse of dimensionality”), and humans are generally not geared for such challenges.



# Dimensionality and Complexity

Humans are good at seeing/understanding data in few dimensions!

However, as dimensionality grows, complexity grows exponentially (“curse of dimensionality”), and humans are generally not geared for such challenges.

	Low dim.	High dim.
Linear	Humans: Computers:	Humans: Computers:
Non-linear	Humans: Computers:	Humans: Computers:

Computers, on the other hand, are OK with high dimensionality, albeit the growth of the challenge, but have a harder time facing non-linear issues.

However, through smart algorithms, computers have learned to deal with it all!

**That is essentially what Machine Learning has enabled!**

# Dimensionality and Complexity

Humans are good at seeing/understanding data in few dimensions!

However, as dimensionality grows, complexity grows exponentially (“curse of dimensionality”), and humans are generally not geared for such challenges.

	Low dim.	High dim.
Linear	Humans: Computers:	Humans: Computers:
Non-linear	Humans: Computers:	Humans: Computers:

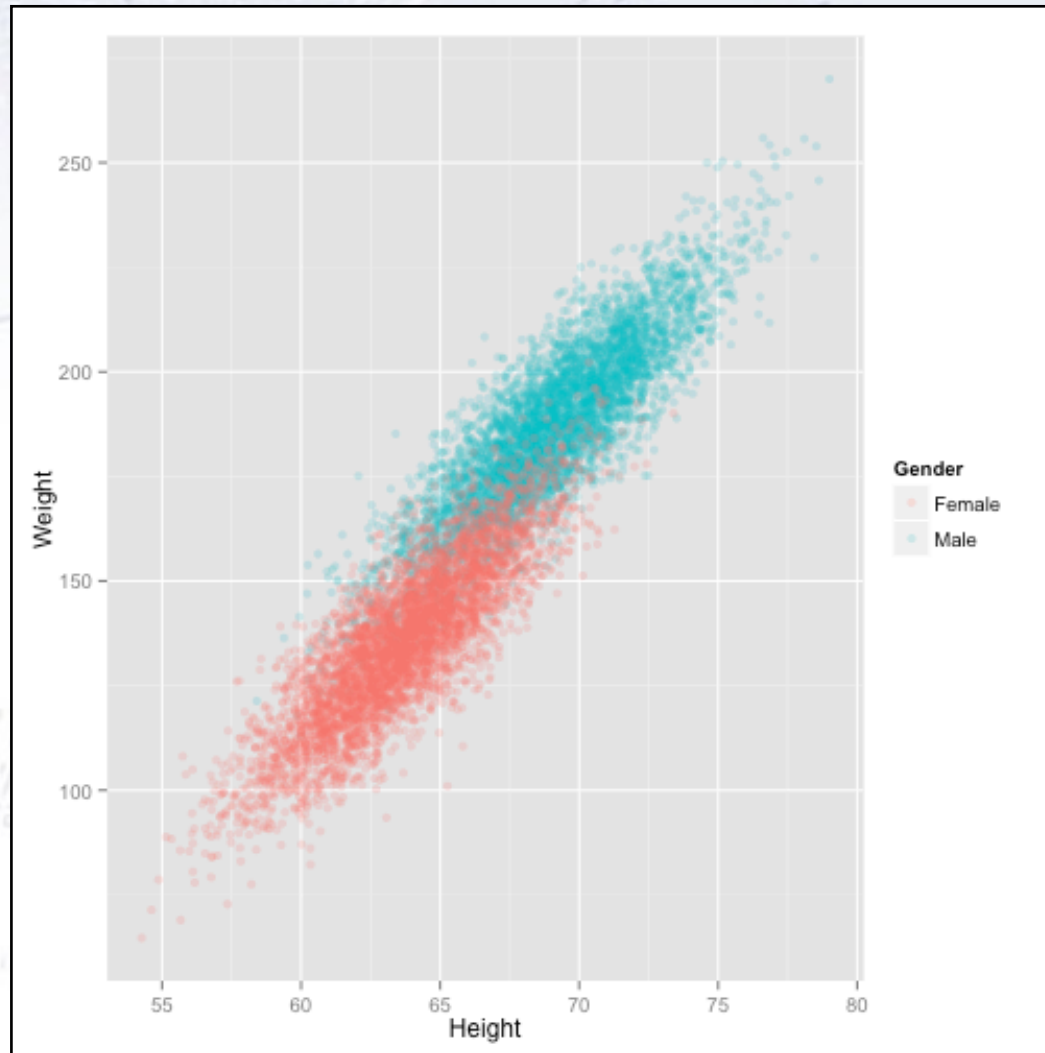
Computers, on the other hand, are OK with high dimensionality, albeit the growth of the challenge, but have a harder time facing non-linear issues.

However, through smart algorithms, computers have learned to deal with it all!

**That is essentially what Machine Learning has enabled!**

# Dimensionality and Complexity

Humans & Computers are good at seeing/understanding linear data in few dimensions:





# Dimensionality and Complexity

Humans are good at seeing/understanding data in few dimensions!

However, as dimensionality grows, complexity grows exponentially (“curse of dimensionality”), and humans are generally not geared for such challenges.

	Low dim.	High dim.
Linear	Humans: ✓ Computers: ✓	Humans: Computers:
Non-linear	Humans: Computers:	Humans: Computers:

Computers, on the other hand, are OK with high dimensionality, albeit the growth of the challenge, but have a harder time facing non-linear issues.

However, through smart algorithms, computers have learned to deal with it all!

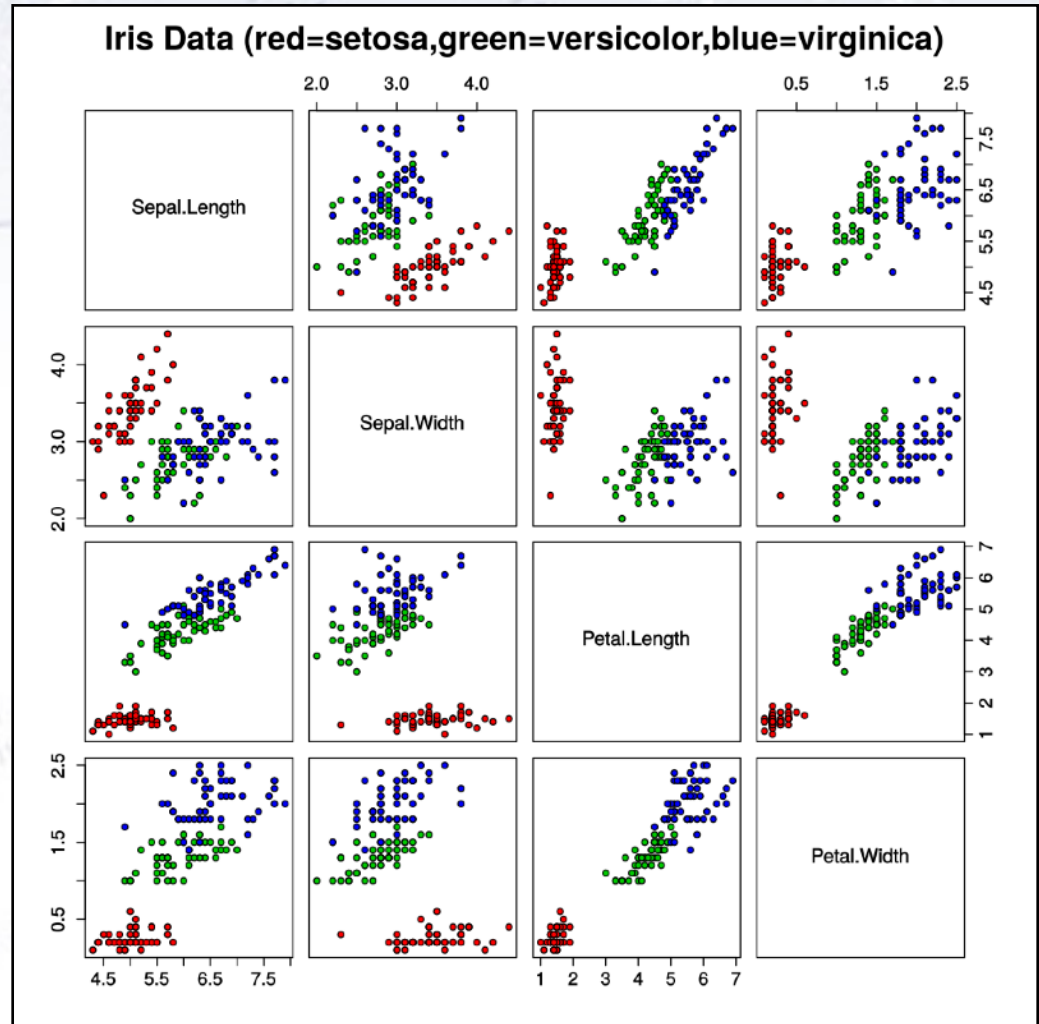
**That is essentially what Machine Learning has enabled!**

# Dimensionality and Complexity

However, when the dimensionality goes beyond 3D, we are lost, even for simple linear data. Computers are not...

Shown is the famous  
Fisher Iris dataset:  
150 irises (3 kinds) with  
4 measurements for each.

**4 dimensional data!**



# Dimensionality and Complexity

Humans are good at seeing/understanding data in few dimensions!

However, as dimensionality grows, complexity grows exponentially (“curse of dimensionality”), and humans are generally not geared for such challenges.

	Low dim.	High dim.
Linear	Humans: ✓ Computers: ✓	Humans: ÷ Computers: ✓
Non-linear	Humans: Computers:	Humans: Computers:

Computers, on the other hand, are OK with high dimensionality, albeit the growth of the challenge, but have a harder time facing non-linear issues.

However, through smart algorithms, computers have learned to deal with it all!

**That is essentially what Machine Learning has enabled!**





Jackson Pollock



# Dimensionality and Complexity

Humans are good at seeing/understanding data in few dimensions!

However, as dimensionality grows, complexity grows exponentially (“curse of dimensionality”), and humans are generally not geared for such challenges.

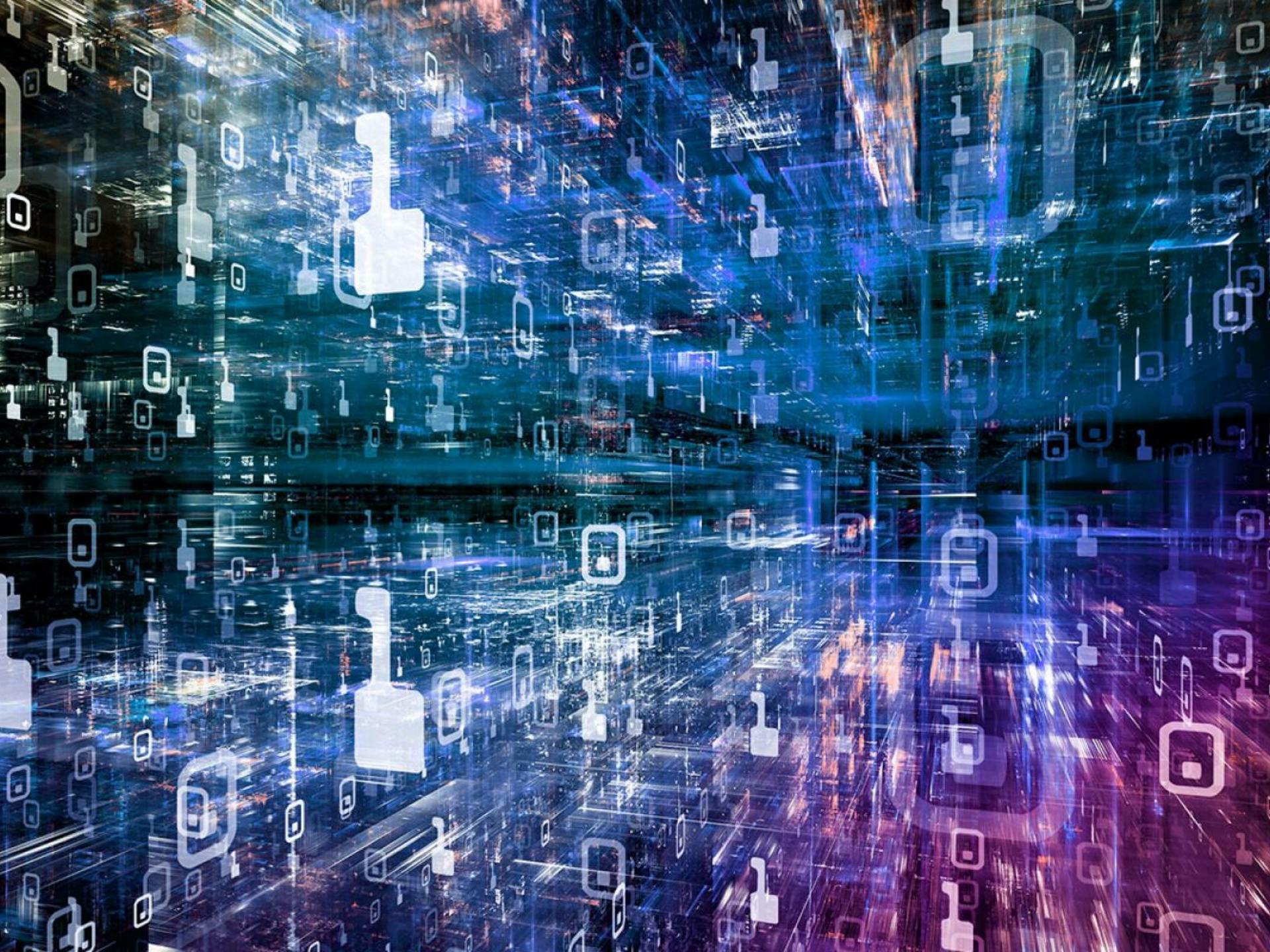
	Low dim.	High dim.
Linear	Humans: ✓ Computers: ✓	Humans: ÷ Computers: ✓
Non-linear	Humans: ✓ Computers: (✓)	Humans: Computers:

Computers, on the other hand, are OK with high dimensionality, albeit the growth of the challenge, but have a harder time facing non-linear issues.

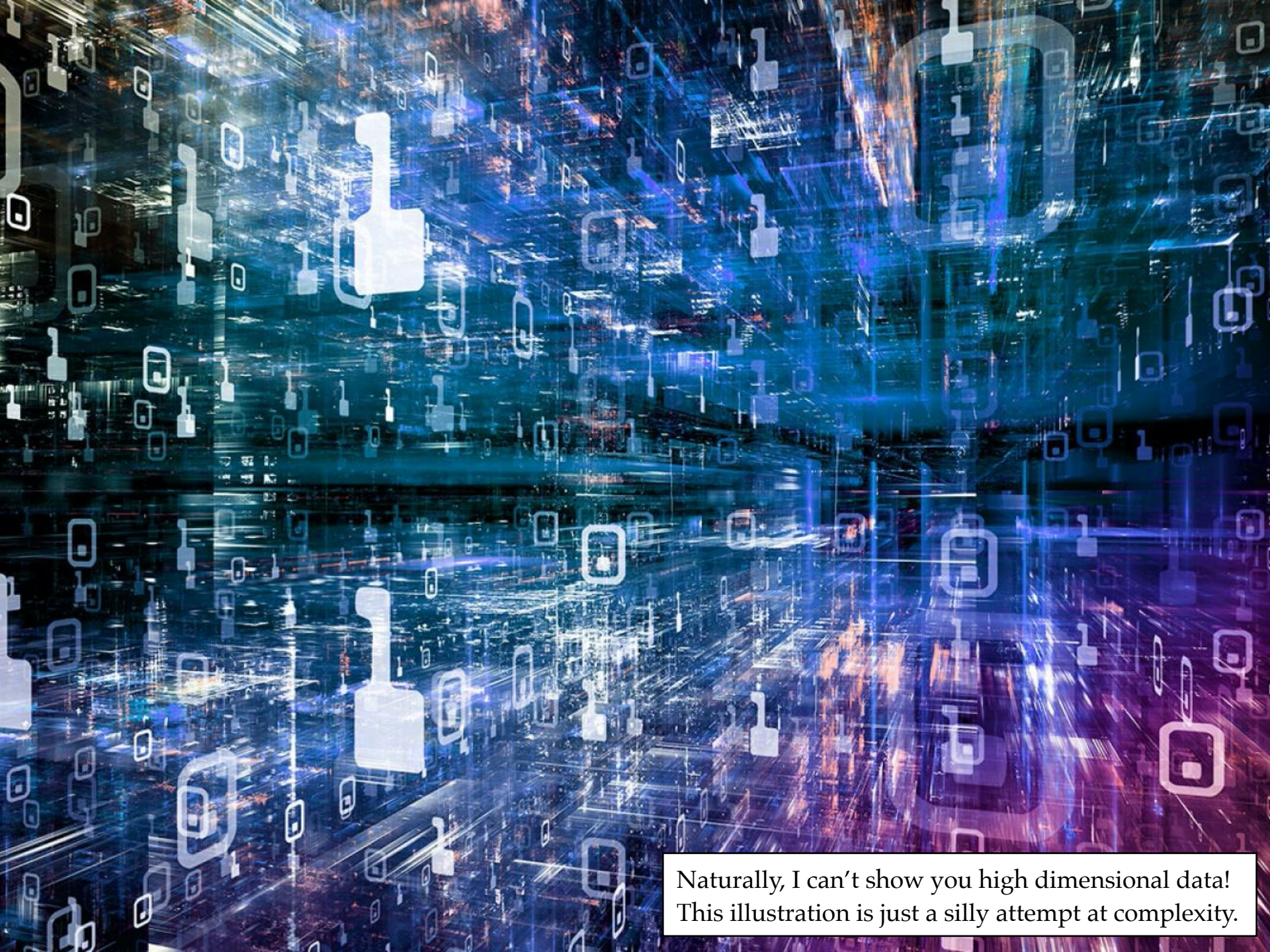
However, through smart algorithms, computers have learned to deal with it all!

**That is essentially what Machine Learning has enabled!**









Naturally, I can't show you high dimensional data!  
This illustration is just a silly attempt at complexity.



# Dimensionality and Complexity

Humans are good at seeing/understanding data in few dimensions!

However, as dimensionality grows, complexity grows exponentially (“curse of dimensionality”), and humans are generally not geared for such challenges.

	Low dim.	High dim.
Linear	Humans: ✓ Computers: ✓	Humans: ÷ Computers: ✓
Non-linear	Humans: ✓ Computers: (✓)	Humans: ÷ Computers: (✓)

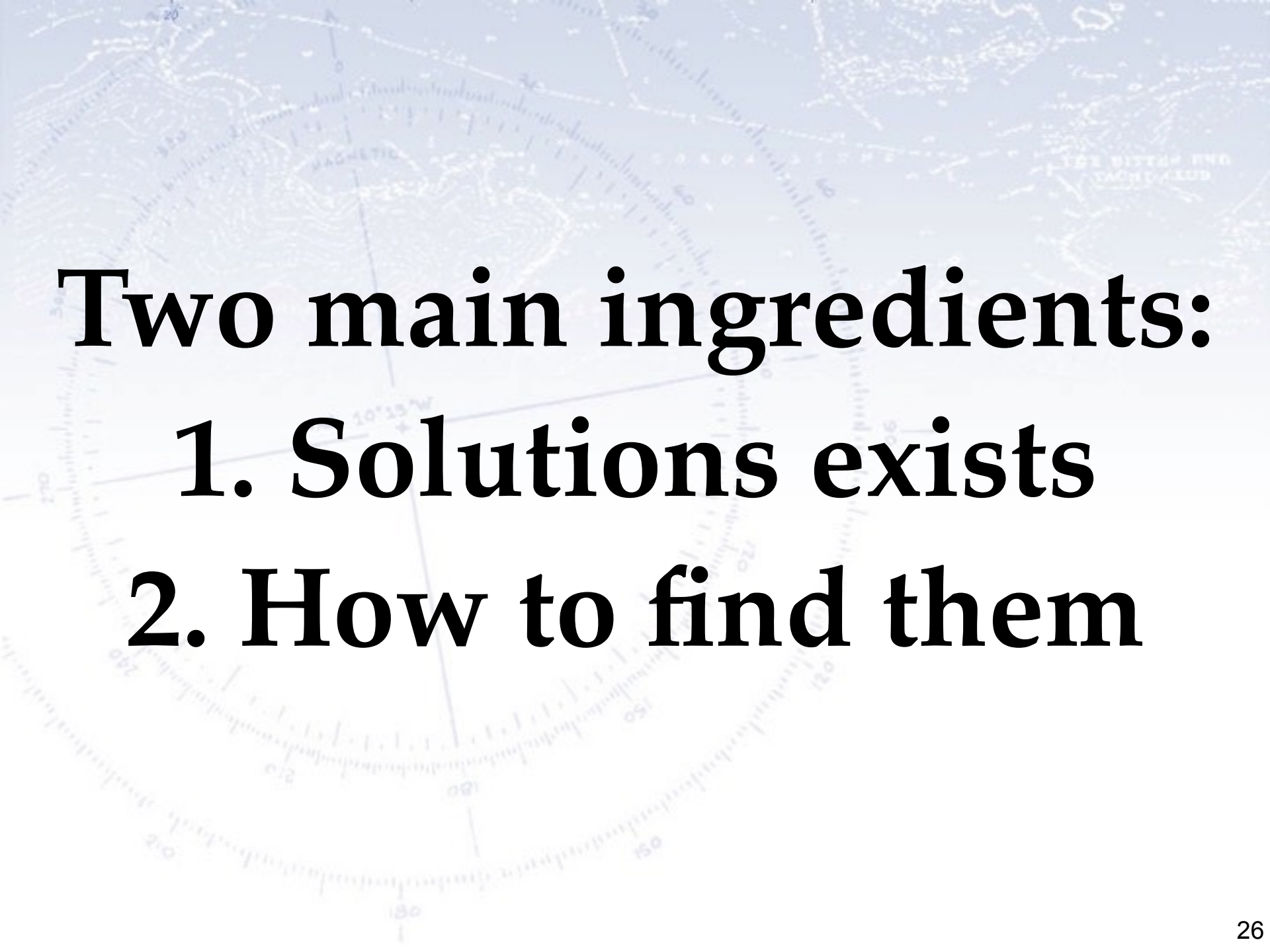
Computers, on the other hand, are OK with high dimensionality, albeit the growth of the challenge, but have a harder time facing non-linear issues.

However, through smart algorithms, computers have learned to deal with it all!  
**That is essentially what Machine Learning has enabled!**





# Two main ingredients



**Two main ingredients:**

- 1. Solutions exists**
- 2. How to find them**

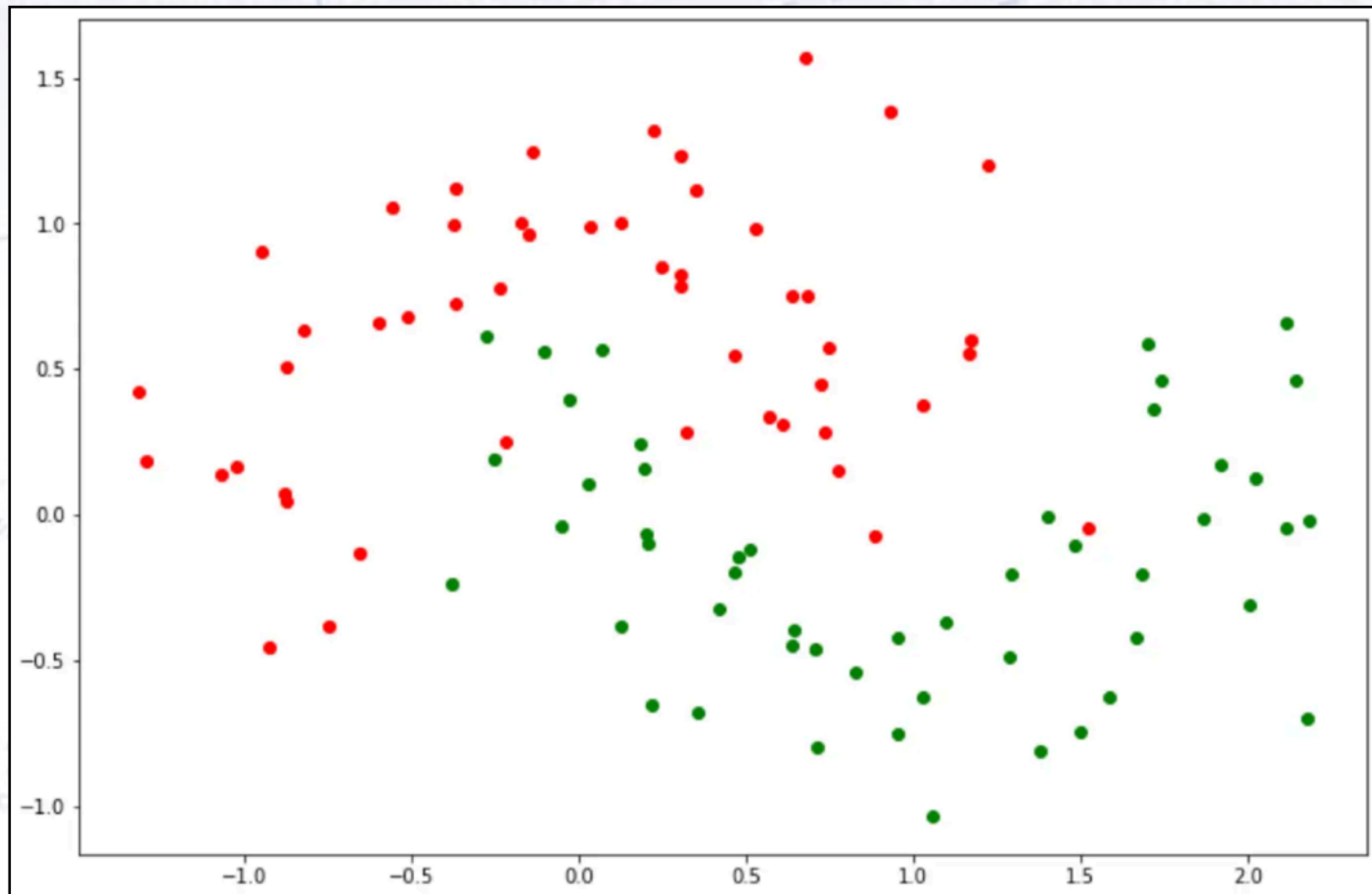


# Solutions exists

(Technically called Universal Approximation Theorems)

# Where to separate?

Look at the red and green points, and imagine that you wanted to draw a curve that separates these.

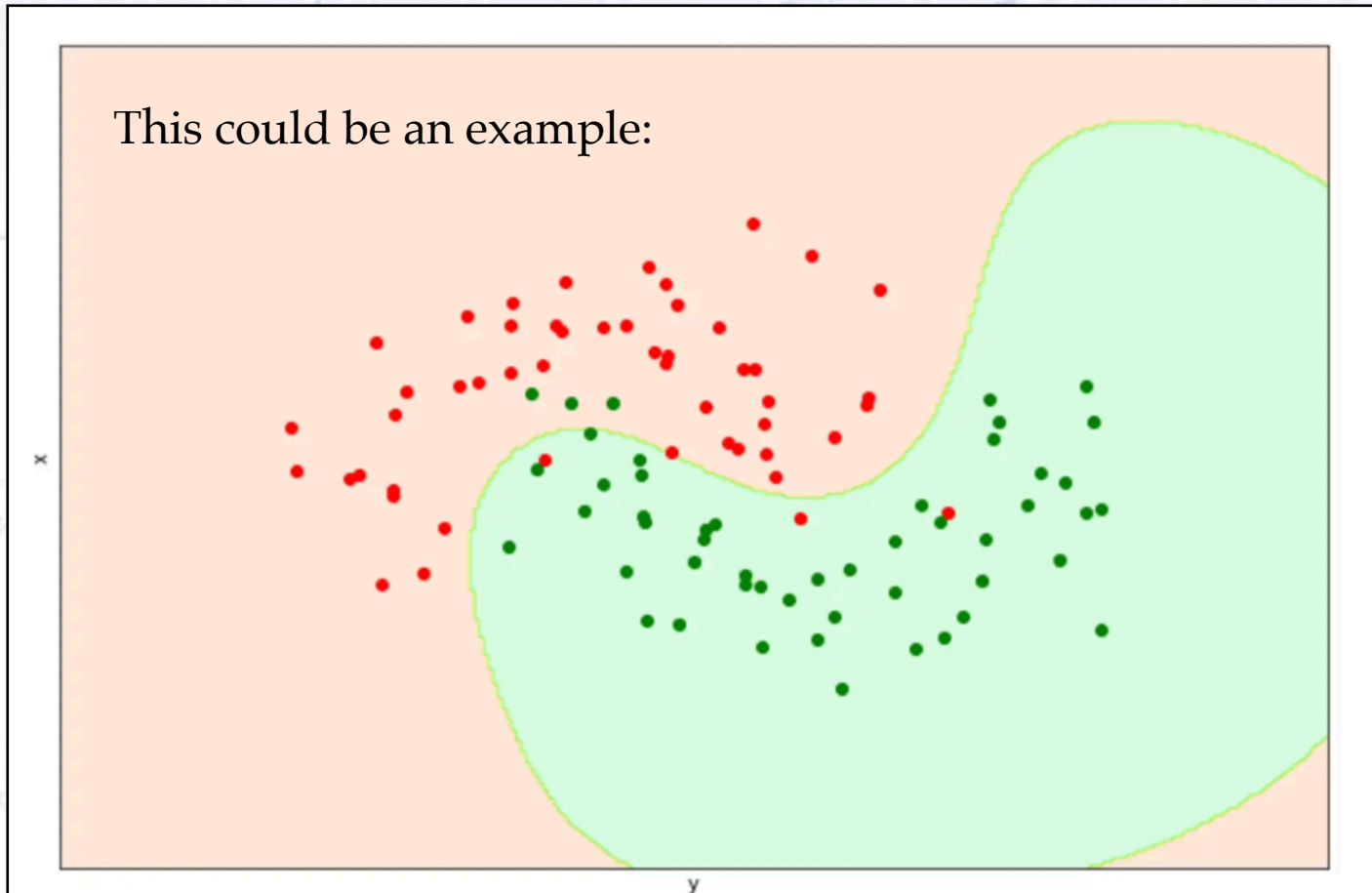




# Where to separate?

Look at the red and green points, and imagine that you wanted to draw a curve that separates these.

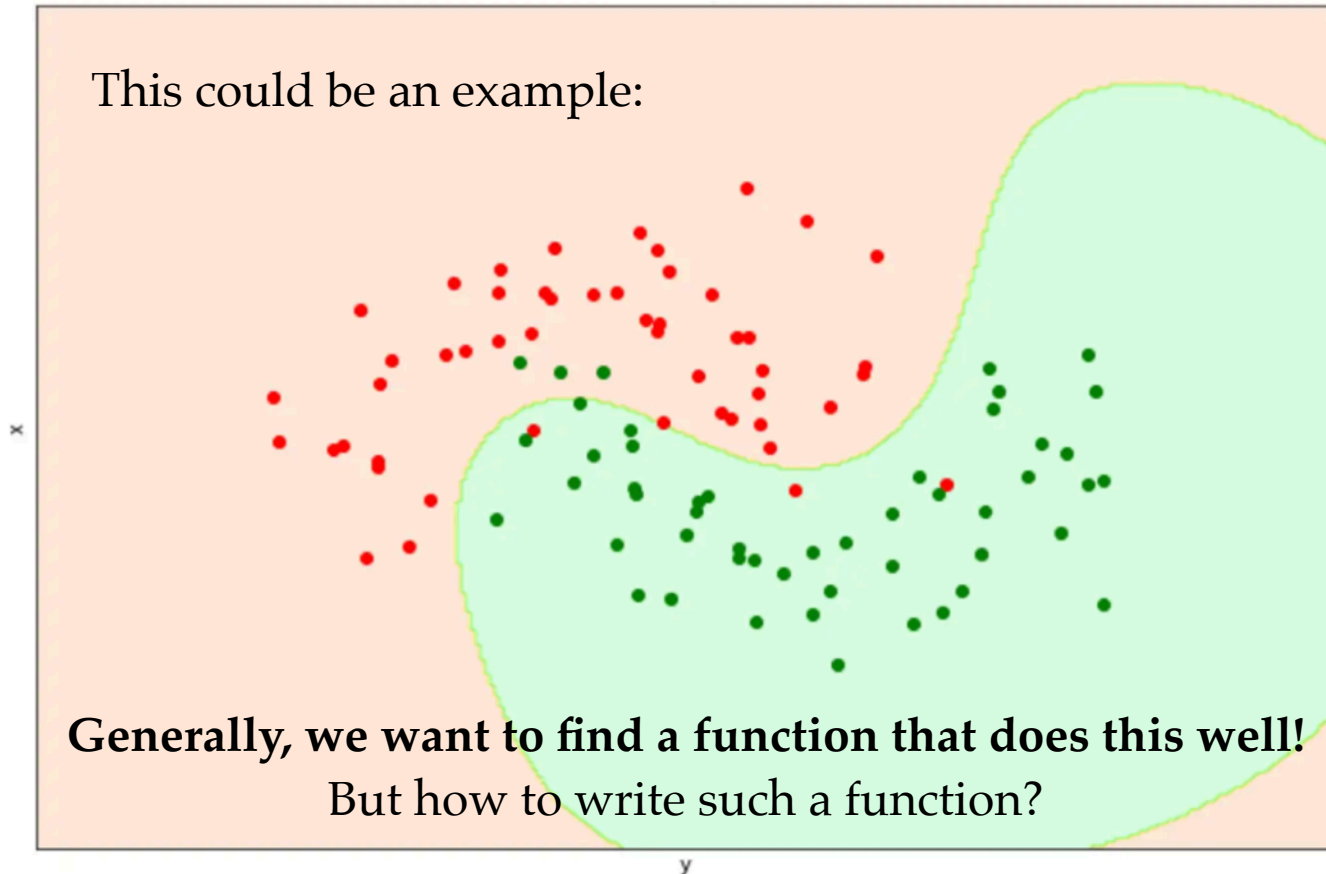
This could be an example:



# Where to separate?

Look at the red and green points, and imagine that you wanted to draw a curve that separates these.

This could be an example:



**Generally, we want to find a function that does this well!**  
But how to write such a function?

# Universal Approx. Theorems

A simple function can be obtained simply by asking a lot of questions:

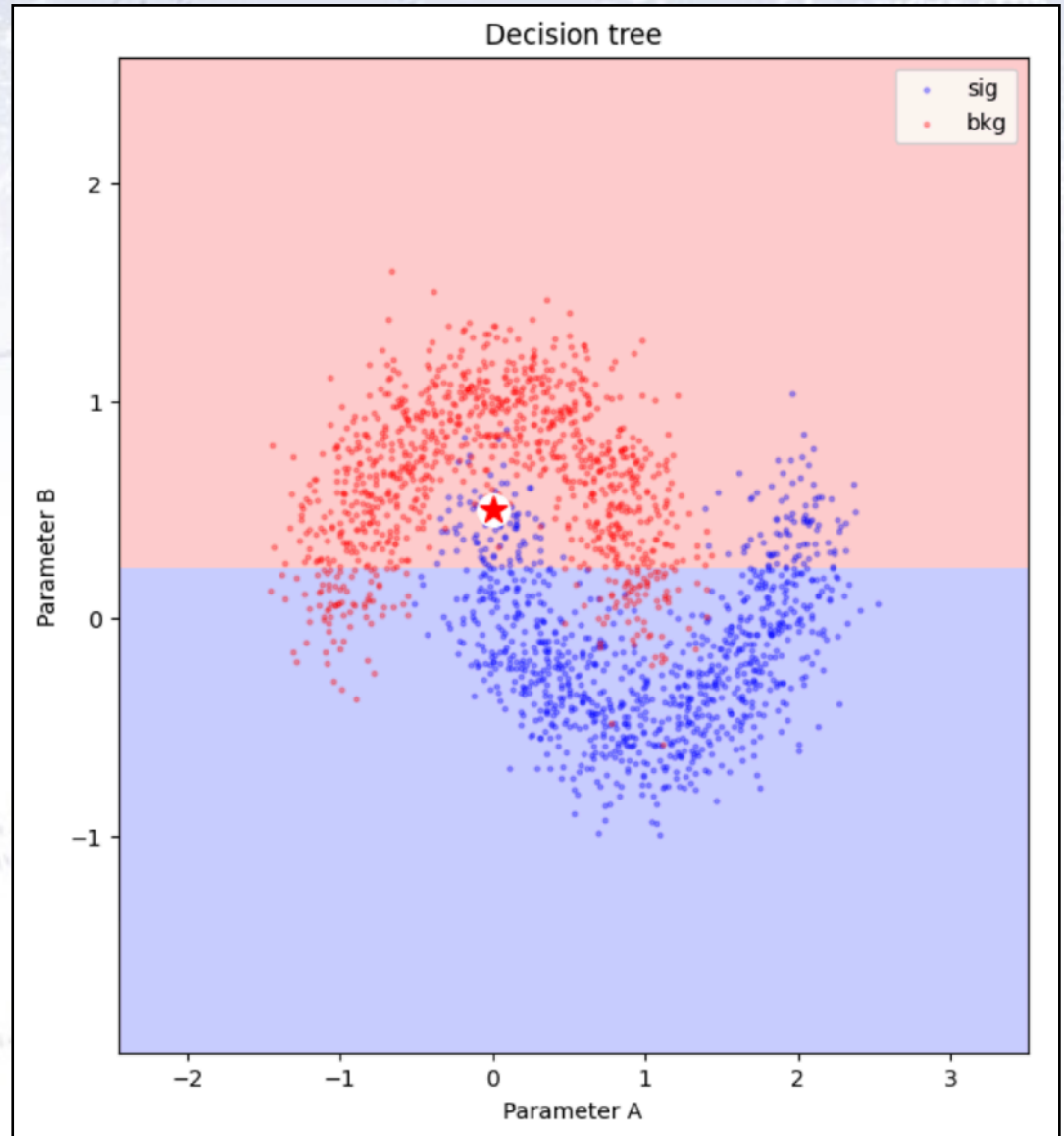
Question: Is  $B > 0.23$ ?

Answer: Yes  $\rightarrow$  Red

Answer: No  $\rightarrow$  Blue

This question is illustrated in the drawing by the horizontal line with red and blue on the sides.

A Decision Tree consists of asking many such questions, corresponding to setting a lot of lines.



# Universal Approx. Theorems

A simple function can be obtained simply by asking a lot of questions:

Question: Is  $B > 0.23$ ?

Answer: Yes  $\rightarrow$  Red

Answer: No  $\rightarrow$  Blue

This question is illustrated in the drawing by the horizontal line with red and blue on the sides.

A Decision Tree consists of asking many such questions, corresponding to setting a lot of lines.





# Universal Approx. Theorems

A simple function can be obtained simply by asking a lot of questions:

Question: Is  $B > 0.23$ ?

Answer: Yes  $\rightarrow$  Red

Answer: No  $\rightarrow$  Blue

This question is illustrated in the drawing by the horizontal line with red and blue on the sides.

A Decision Tree consists of asking many such questions, corresponding to setting a lot of lines.



# Universal Approx. Theorems

A simple function can be obtained simply by asking a lot of questions:

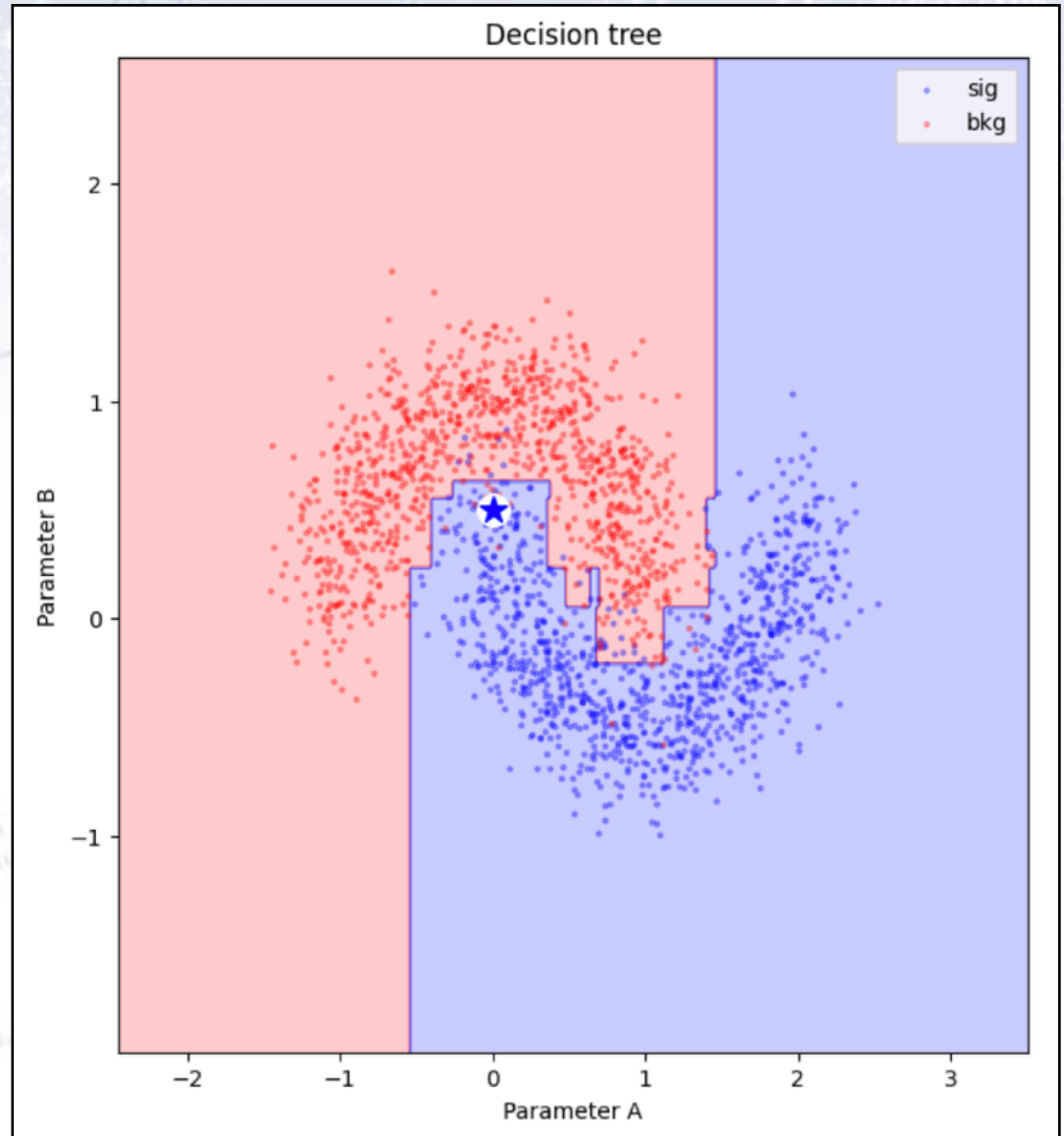
Question: Is  $B > 0.23$ ?

Answer: Yes  $\rightarrow$  Red

Answer: No  $\rightarrow$  Blue

This question is illustrated in the drawing by the horizontal line with red and blue on the sides.

A Decision Tree consists of asking many such questions, corresponding to setting a lot of lines.



# Universal Approx. Theorems

A simple function can be obtained simply by asking a lot of questions:

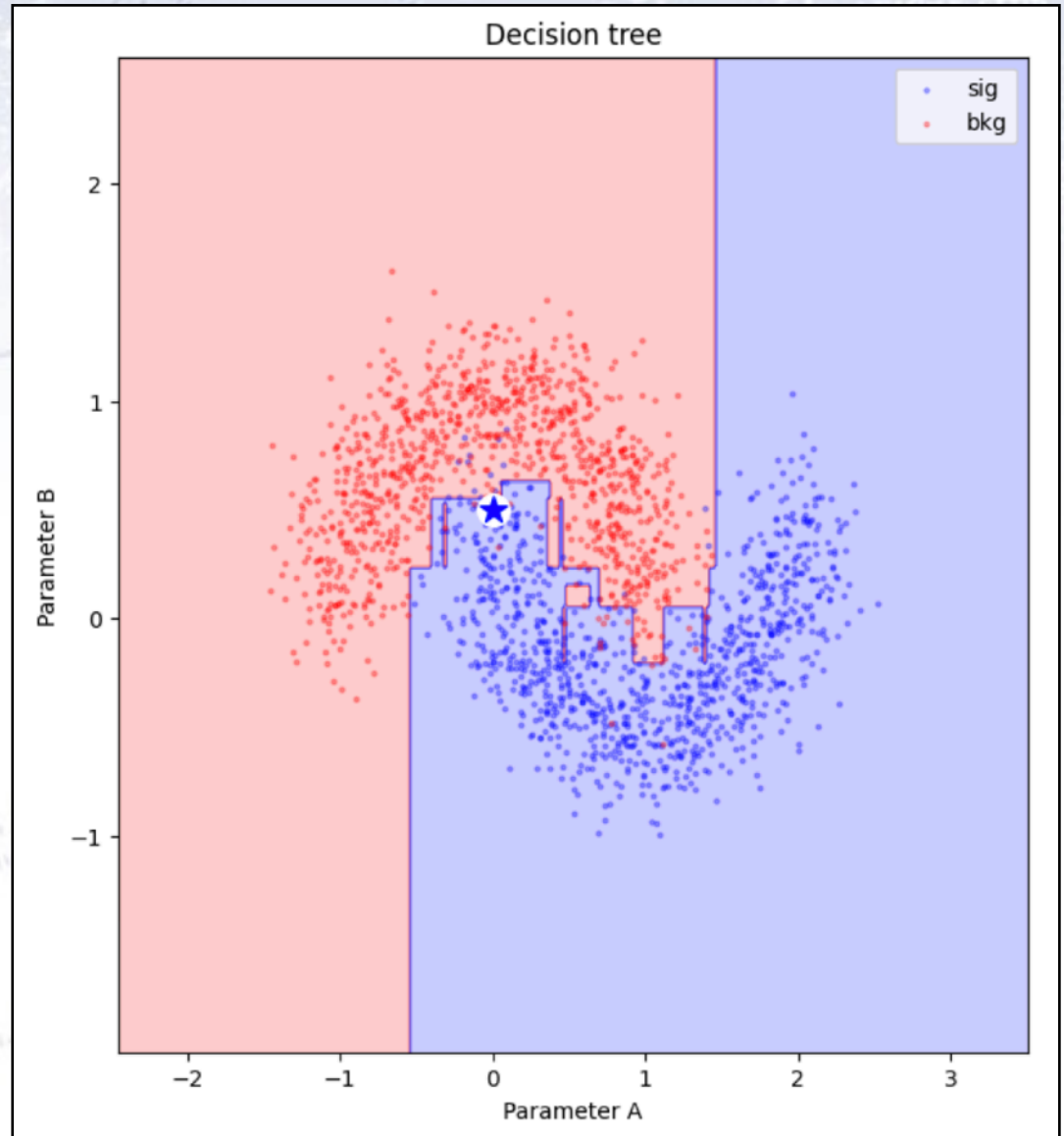
Question: Is  $B > 0.23$ ?

Answer: Yes  $\rightarrow$  Red

Answer: No  $\rightarrow$  Blue

This question is illustrated in the drawing by the horizontal line with red and blue on the sides.

A Decision Tree consists of asking many such questions, corresponding to setting a lot of lines.





# Universal Approx. Theorems

A simple function can be obtained simply by asking a lot of questions:

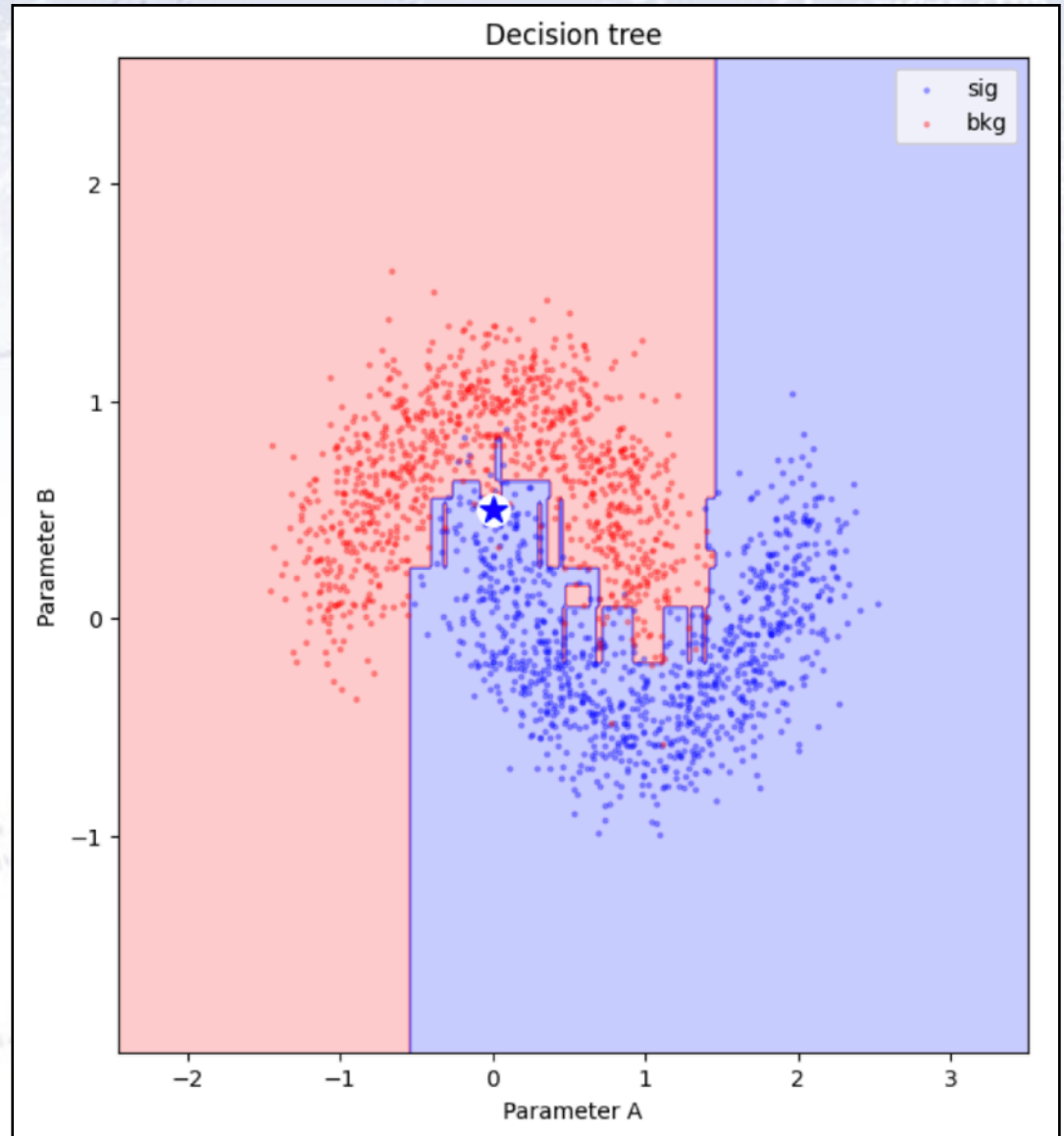
Question: Is  $B > 0.23$ ?

Answer: Yes  $\rightarrow$  Red

Answer: No  $\rightarrow$  Blue

This question is illustrated in the drawing by the horizontal line with red and blue on the sides.

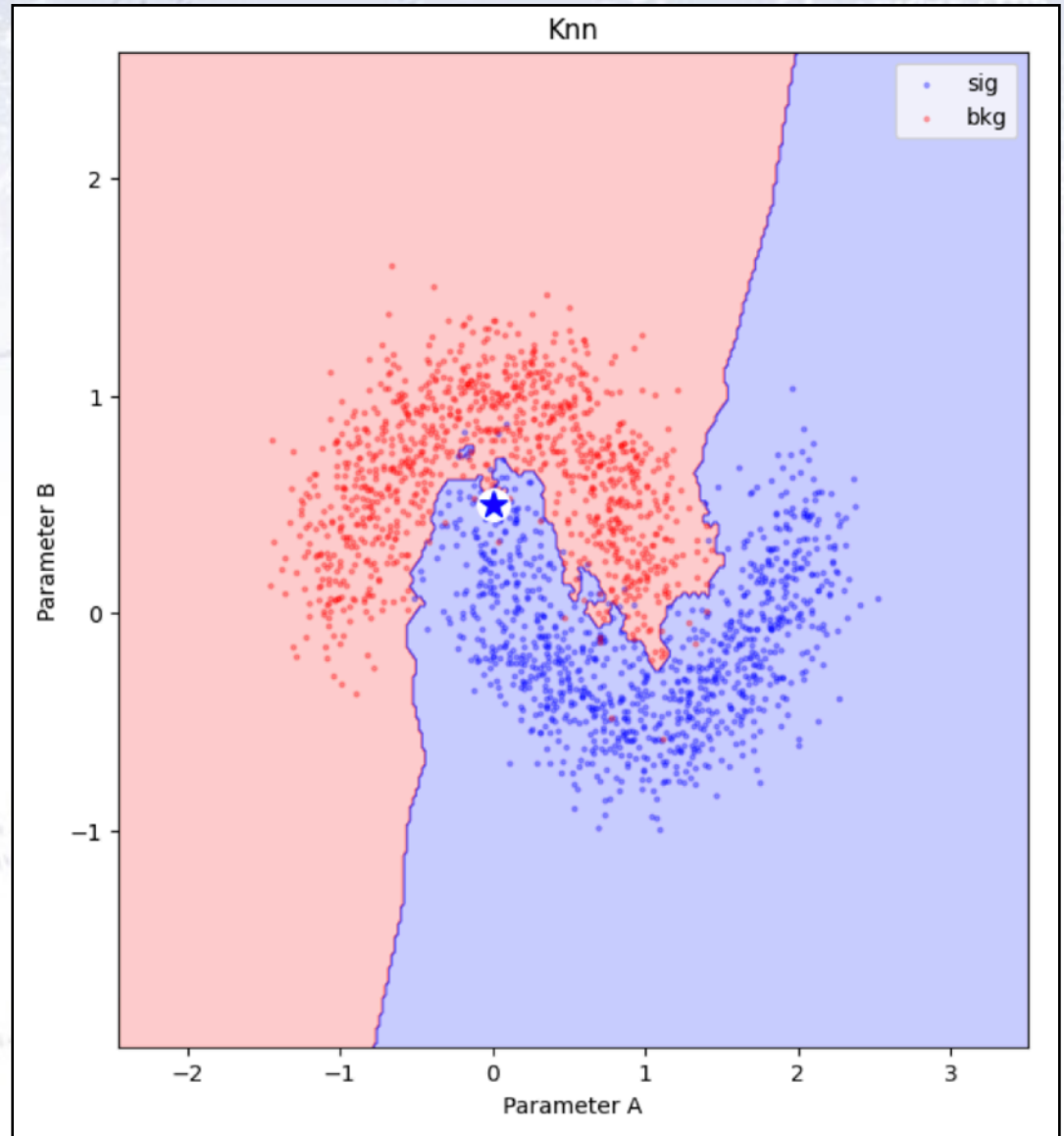
A Decision Tree consists of asking many such questions, corresponding to setting a lot of lines.



# Universal Approx. Theorems

An alternative method would be to simply ask what type (i.e. color) the majority of the  $k$  nearest neighbours have.

In the drawing  $k=3$ , leading to a rather “ragged” border.



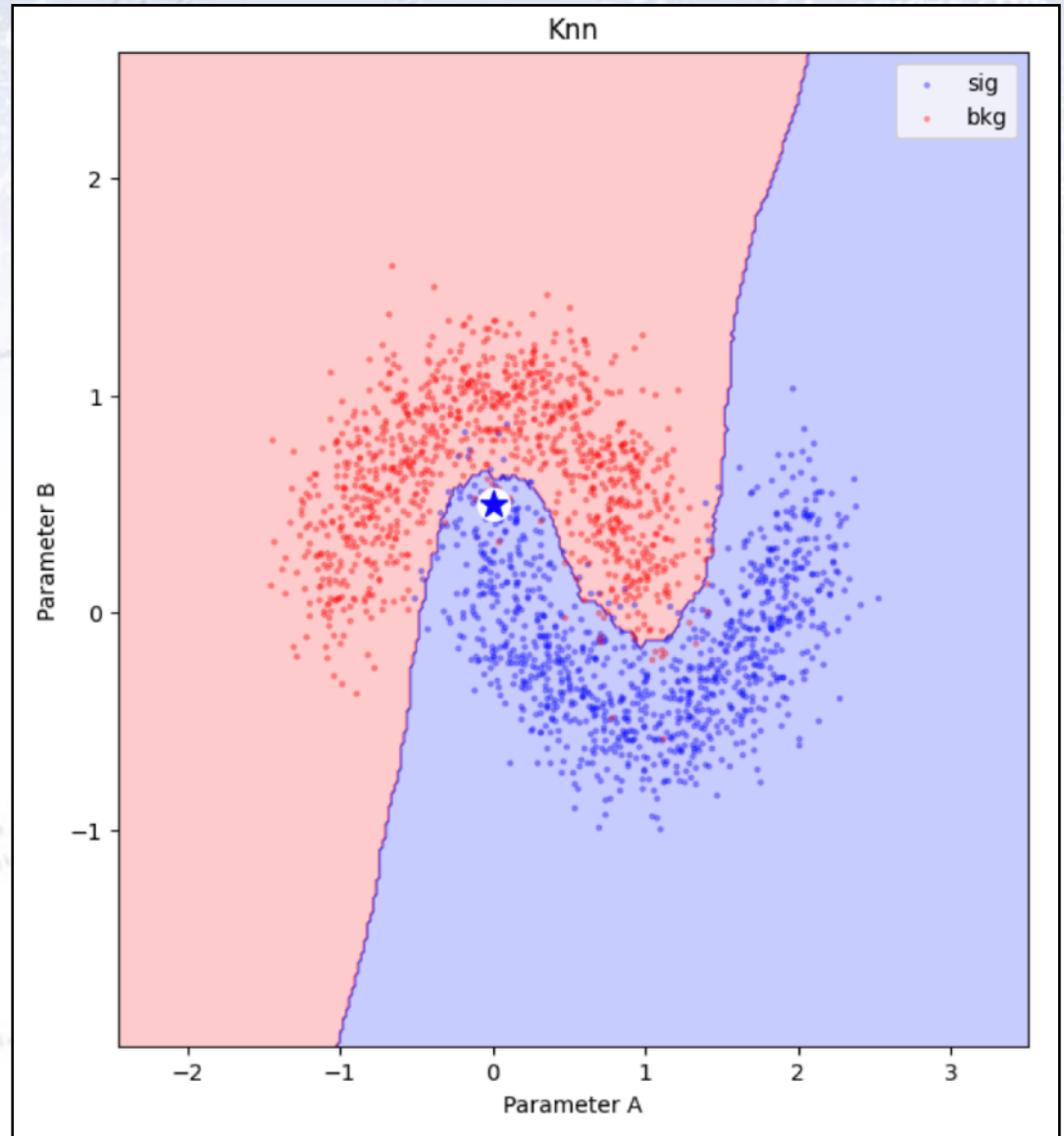
# Universal Approx. Theorems

An alternative method would be to simply ask what type (i.e. color) the majority of the  $k$  nearest neighbours have.

In the drawing  $k=3$ , leading to a rather “ragged” border.

However, increasing the number of neighbours considered to  $k=30$  gives a more smooth border.

This method is called “**k-nearest neighbours**”.

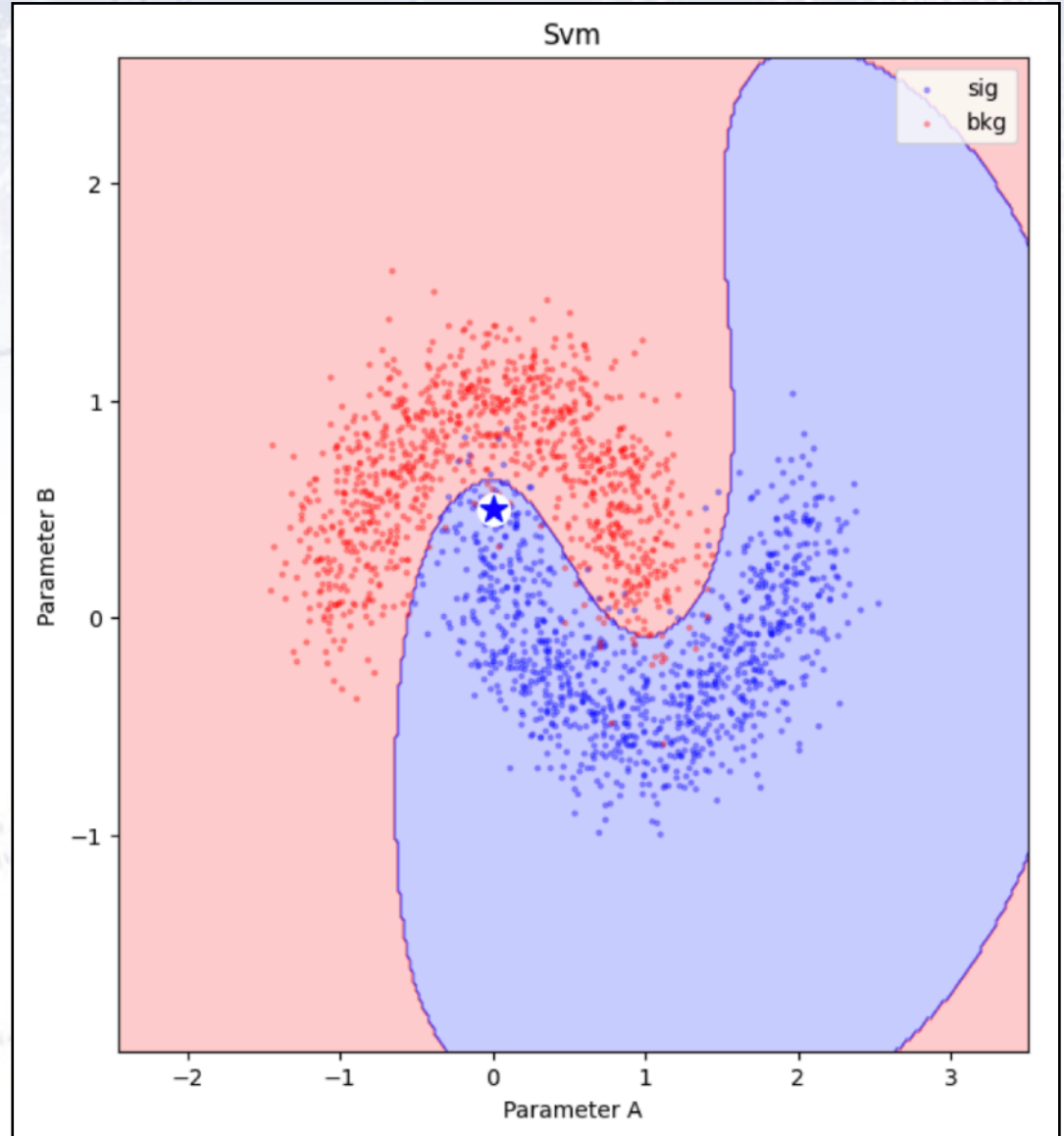




# Universal Approx. Theorems

Other methods are based on entirely different principles.

There is a wealth of different methods.

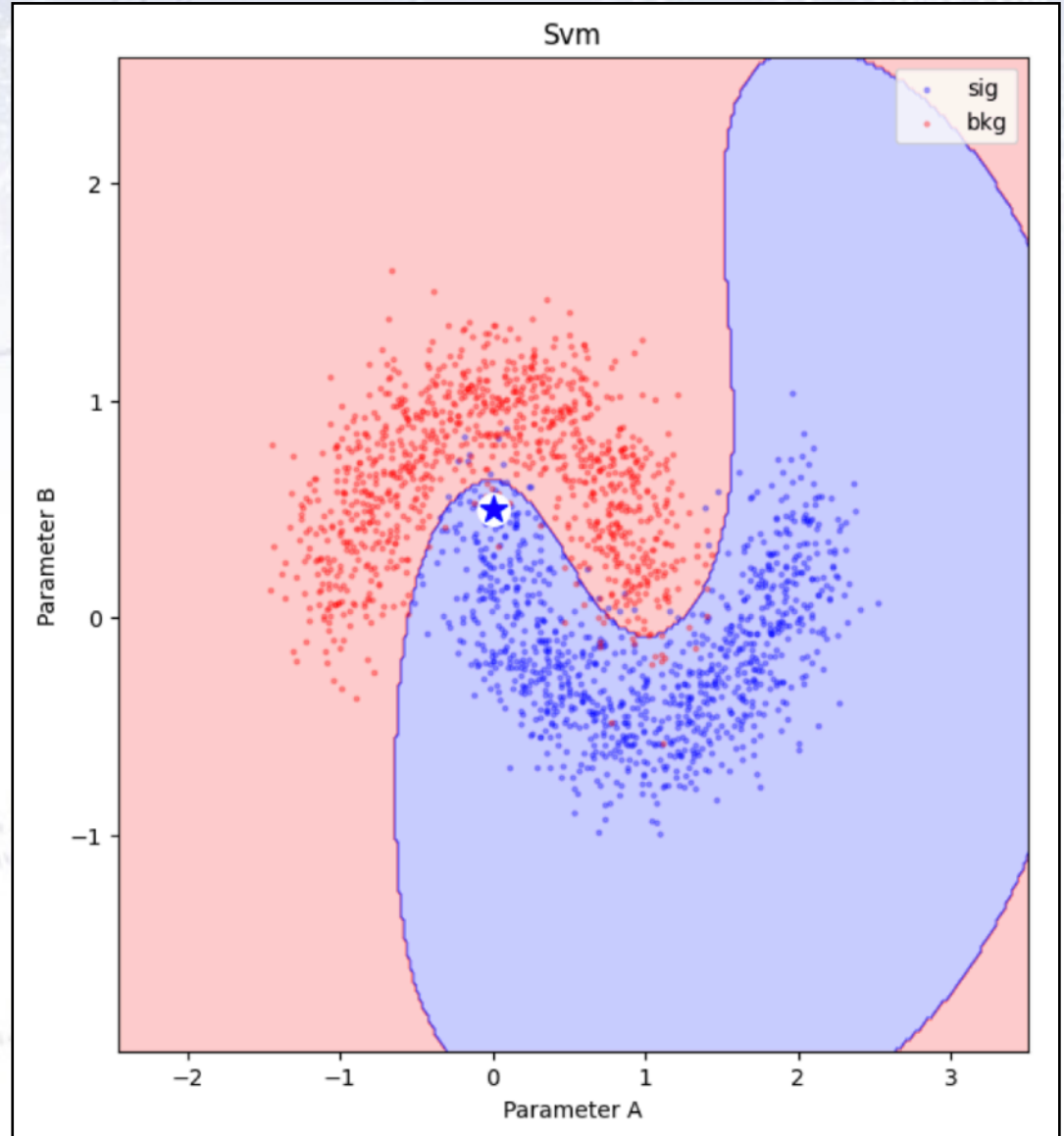


# Universal Approx. Theorems

Other methods are based on entirely different principles.

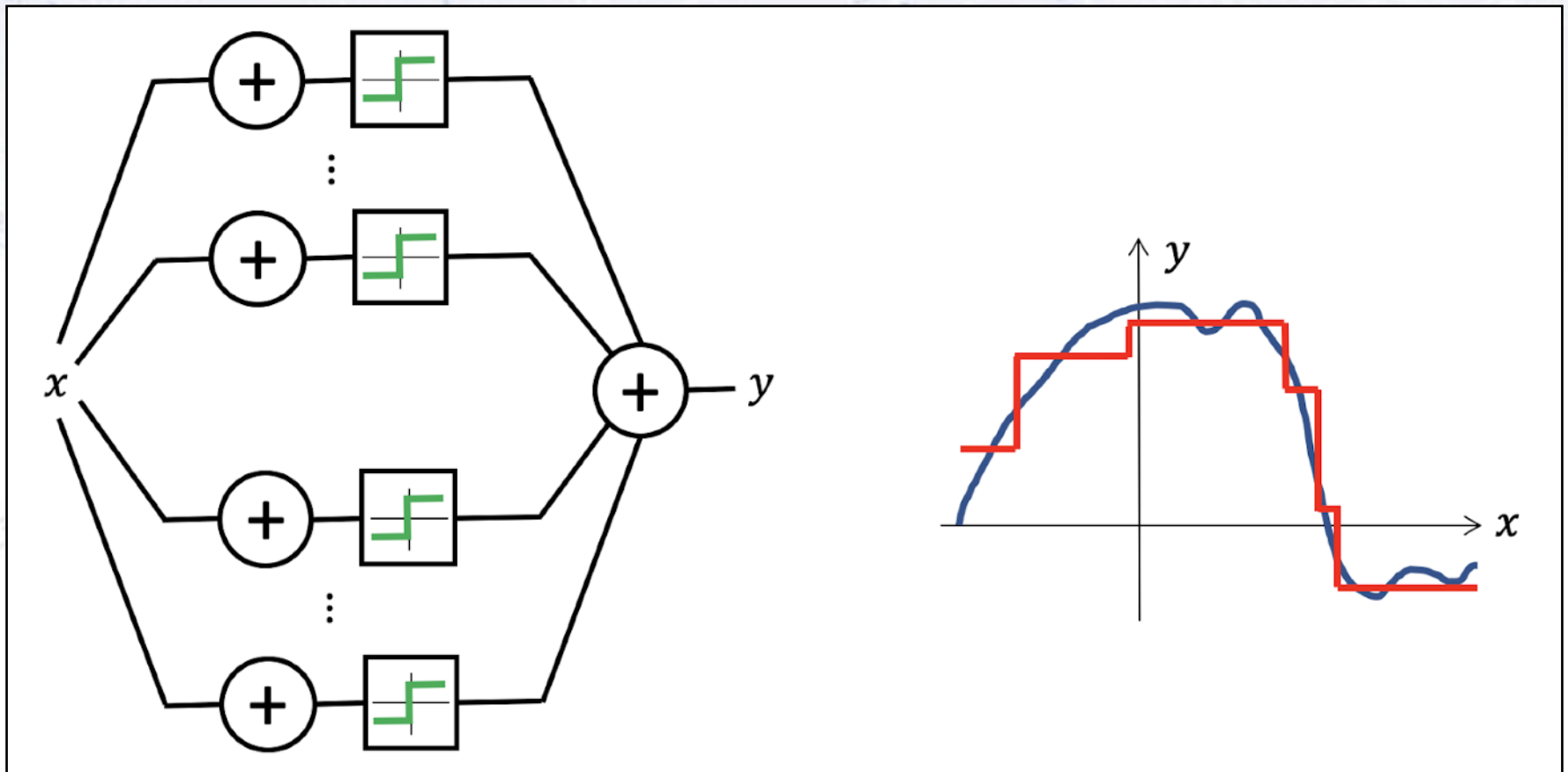
There is a wealth of different methods.

**The main idea is, that they are all capable of getting close to the optimal solution.**



# Universal Approx. Theorems

Such approximations typically entails a **large amount of parameters**, for which the UATs give no recipe on how to find - only that such a construction is possible.





# Universal Approximation Theorems

**Theorem 5.1.1 (Universal Approximation Theorem)** <sup>10</sup> Let  $\sigma$  be a non-constant, bounded, and monotone-increasing continuous function. Let  $I_{m_0}$  denote the  $m_0$ -dimensional unit hypercube  $[0, 1]^{m_0}$ . The space of continuous functions on  $I_{m_0}$  is denoted as  $C(I_{m_0})$ . Then given any function  $f \in C(I_{m_0})$  and  $\epsilon > 0$  there exists a set of real constants  $a_i, b_i$  and  $w_{ij}$ , where  $i = 1, \dots, m_1$  and  $j = 1, \dots, m_0$  such that we may define

$$F(x_1, \dots, x_{m_0}) = \sum_{i=1}^{m_1} a_i \sigma \left( \sum_{j=1}^{m_0} w_{ij} x_j + b_i \right) \quad (5.6)$$

as an approximate realization of the function  $f$ ; that is,

$$|F(x_1, \dots, x_{m_0}) - f(x_1, \dots, x_{m_0})| < \epsilon \quad (5.7)$$

for all  $x_1, x_2, \dots, x_{m_0}$  that lie in the input space.

# Universal Approximation Theorems

**Theorem 5.1.1 (Universal Approximation Theorem)** <sup>10</sup> Let  $\sigma$  be a non-

## Summary:

Neural Networks etc. can approximate functions in any dimension very well!

$$F(x_1, \dots, x_{m_0}) = \sum_{i=1} a_i \sigma \left( \sum_{j=1} w_{ij} x_j + b_i \right) \quad (5.6)$$

*as an approximate realization of the function  $f$ ; that is,*

$$|F(x_1, \dots, x_{m_0}) - f(x_1, \dots, x_{m_0})| < \epsilon \quad (5.7)$$

*for all  $x_1, x_2, \dots, x_{m_0}$  that lie in the input space.*

# Universal Approximation Theorems

**Theorem 5.1.1 (Universal Approximation Theorem)** <sup>10</sup> *Let  $\sigma$  be a non-*

## Summary:

Neural Networks etc. can approximate functions in any dimension very well!

$$F(x_1, \dots, x_{m_0}) = \sum a_i \sigma \left( \sum w_{ij} x_j + b_i \right) \quad (5.6)$$

“Deep Learning is not as impressive as you think...  
it’s mere interpolation resulting from glorified curve fitting”  
[Yann Lecun, 2021]

*for all  $x_1, x_2, \dots, x_{m_0}$  that lie in the input space.*



# Universal Approx. Theorems

One main ingredient behind ML are **Universal Approximation Theorems (UAT)**.

These imply that Neural Networks can approximate a very wide variety of functions given simple function constraints and enough degrees of freedom.

This typically entails a large amount of weights, for which the UATs give no recipe on how to find - only that such a construction is possible.

# Universal Approx. Theorems

One main ingredient behind ML are **Universal Approximation Theorems (UAT)**.

These imply that Neural Networks can approximate a very wide variety of functions given simple function constraints and enough degrees of freedom.

This typically entails a large amount of weights, for which the UATs give no recipe on how to find - only that such a construction is possible.

**Part of this course is learning how to find these!**

Decision Trees and K-Nearest Neighbour algorithms are also capable of “universal approximation” (i.e. have forms of UATs).

A UAT has also been worked out for Graph Neural Networks... in 2020!

# Universal Approx. Theorems

Regarding UATs, as far as learning is concerned, whether the class is really universal or not is not overly important:

If one assumes that there is no noise in the training set, then there will still be infinitely many functions that pass through all training points and not all of them will have the same error on an unseen point (i.e. the test set).



# Universal Approx. Theorems

Regarding UATs, as far as learning is concerned, whether the class is really universal or not is not overly important:

If one assumes that there is no noise in the training set, then there will still be infinitely many functions that pass through all training points and not all of them will have the same error on an unseen point (i.e. the test set).

Thus, one can ask for what sort of functions the approximation applies.

All differentiable functions? Typically, NNs are restricted to this class.

All continuous functions ? All measurable functions? All computable functions?

As it turns out, the real deal is characterising that class of functions that can be approximated.

# Universal Approx. Theorems

Regarding UATs, as far as learning is concerned, whether the class is really universal or not is not overly important:

If one assumes that there is no noise in the training set, then there will still be infinitely many functions that passes through all training points and not all of them will have the same error on an unseen point (i.e. the test set).

Thus, one can ask for what sort of functions the approximation applies.

All differentiable functions? Typically, NNs are restricted to this class.

All continuous functions ? All measurable functions? All computable functions?

As it turns out, the real deal is characterising that class of functions that can be approximated.

**However, we don't really care about that - we simply assume, that with enough liberty/complexity, the functions can approximate what we want.**



# How to find these

(Technically called Stochastic Gradient Descent)

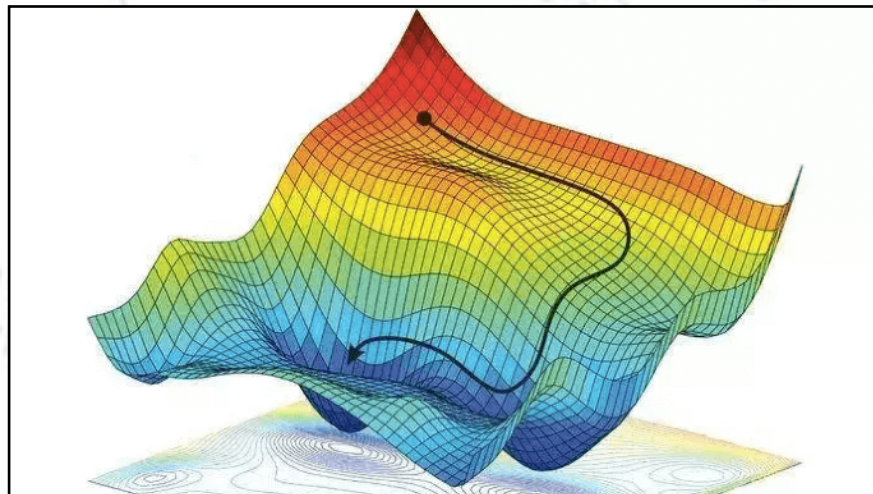


# Stochastic Gradient Descent

The way to obtain the parameters / weights of ML algorithms, is generally by **Stochastic Gradient Descent**.

This “back propagation” algorithm works by computing the gradient of the loss function (to be optimised) with respect to each weight using the chain rule.

One thus computes the gradient one layer at a time, iterating backwards from the last layer (avoiding redundancies). See Goodfellow et al. for details.



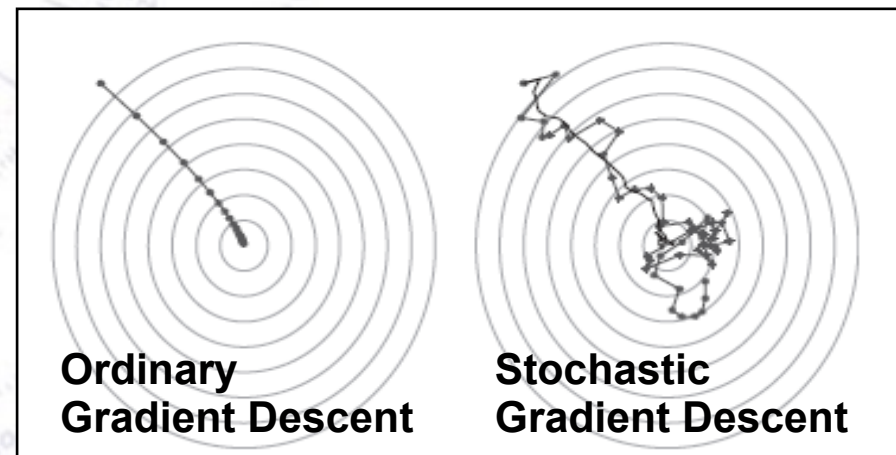
# Stochastic Gradient Descent

The way to obtain the parameters / weights of ML algorithms, is generally by **Stochastic Gradient Descent**.

This “back propagation” algorithm works by computing the gradient of the loss function (to be optimised) with respect to each weight using the chain rule.

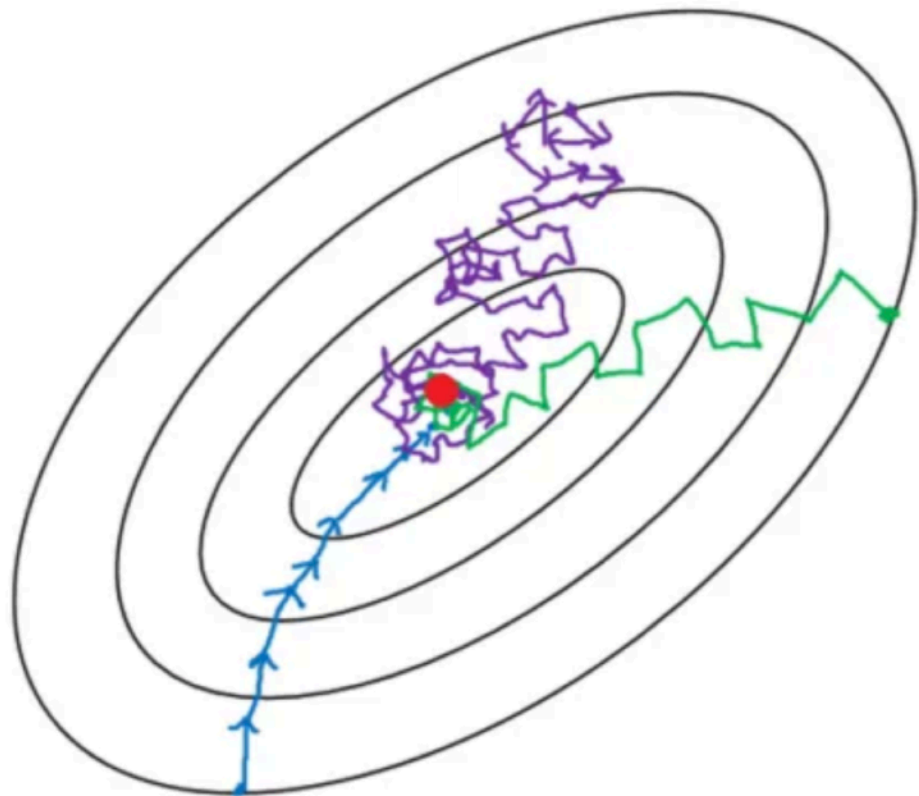
One thus computes the gradient one layer at a time, iterating backwards from the last layer (avoiding redundancies). See Goodfellow et al. for details.

The gradient descent is made stochastic (and fast) by only considering a fraction (called a “batch”) of the data, when calculating the step in the search for optimal parameters for the algorithm. This allow for stochastic jumping, that avoids local (false) minima.

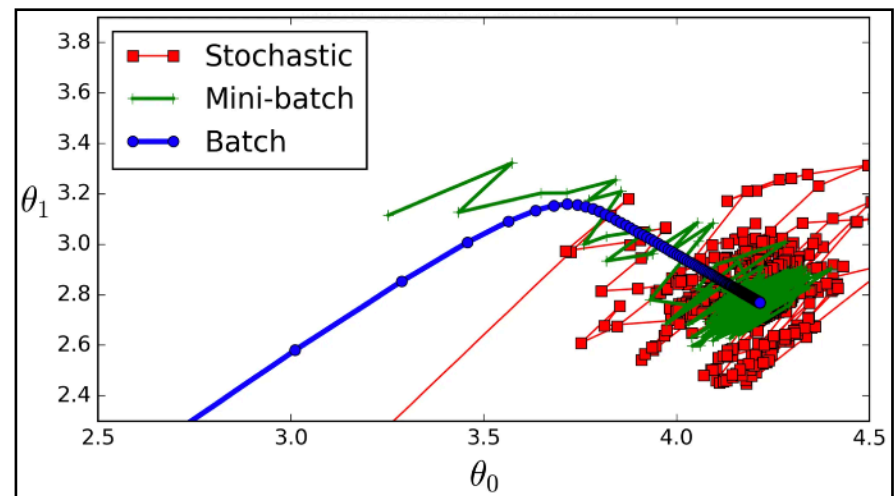


# Stochastic Gradient Descent

The way to obtain the parameters/weights of ML algorithms, is generally by **Stochastic Gradient Descent**.



- Batch gradient descent
- Mini-batch gradient Descent
- Stochastic gradient descent





# Ingredients for ML

So now we know that at least in principle:

- a solution exists (Universal Approximation Theorem) and
- that it can be found (Stochastic Gradient Descent).

But this does not in reality make us capable of getting ML results.

We (at least) also need:

- actual functions/ algorithms for making approximations
- knowledge about how to tell them what to learn
- a scheme for how to use the data we have available

# Ingredients for ML

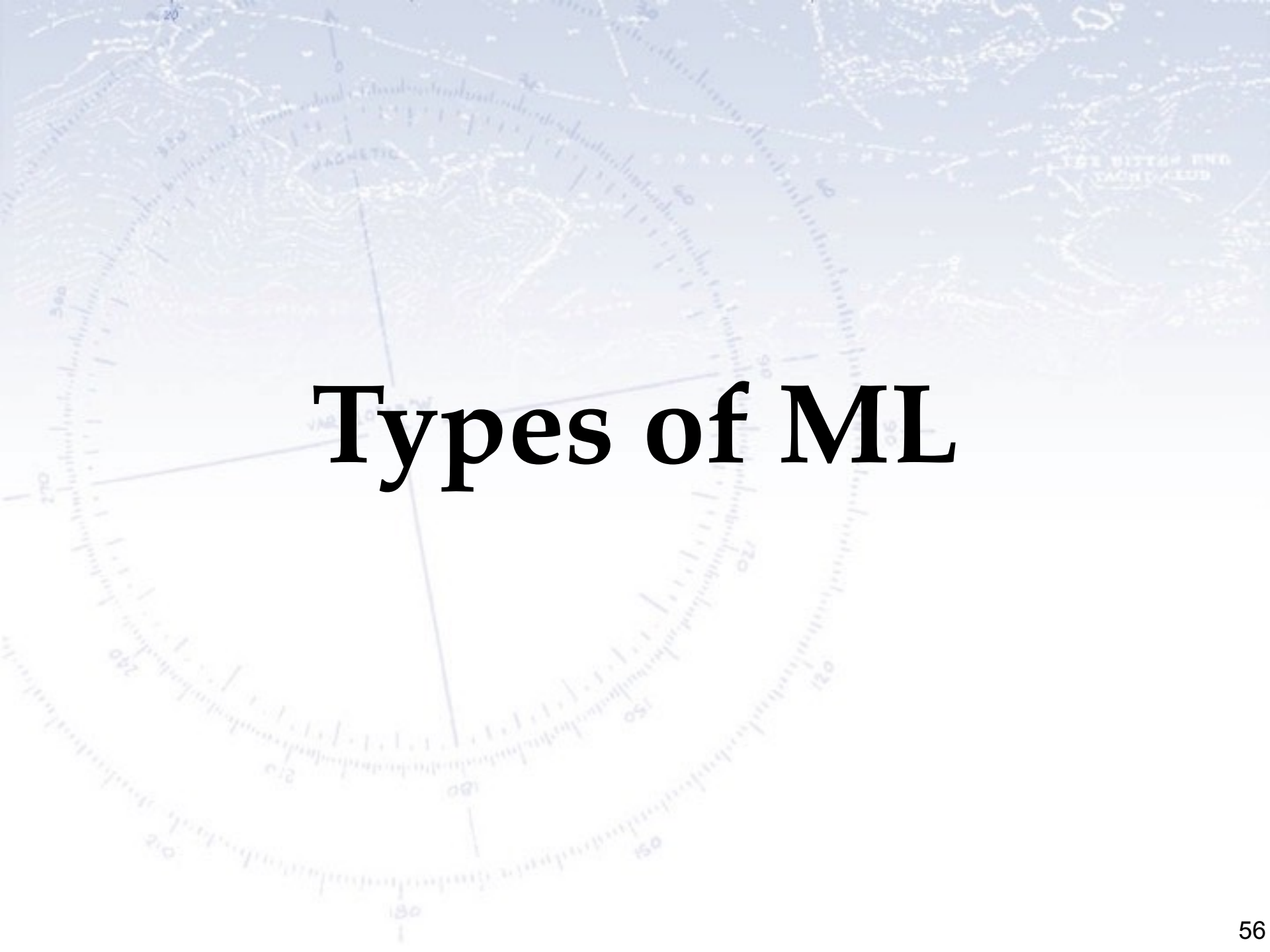
So now we know that at least in principle:

- a solution exists (Universal Approximation Theorem) and
- that it can be found (Stochastic Gradient Descent).

But this does not in reality make us capable of getting ML results.

We (at least) also need:

- actual functions/ algorithms for making approximations  
**Boosted Decision Trees (BDTs) & Neural Networks (NNs)**
- knowledge about how to tell them what to learn  
**Loss functions (and now to minimise these)**
- a scheme for how to use the data we have available  
**Training, validation, and testing samples & Cross Validation**

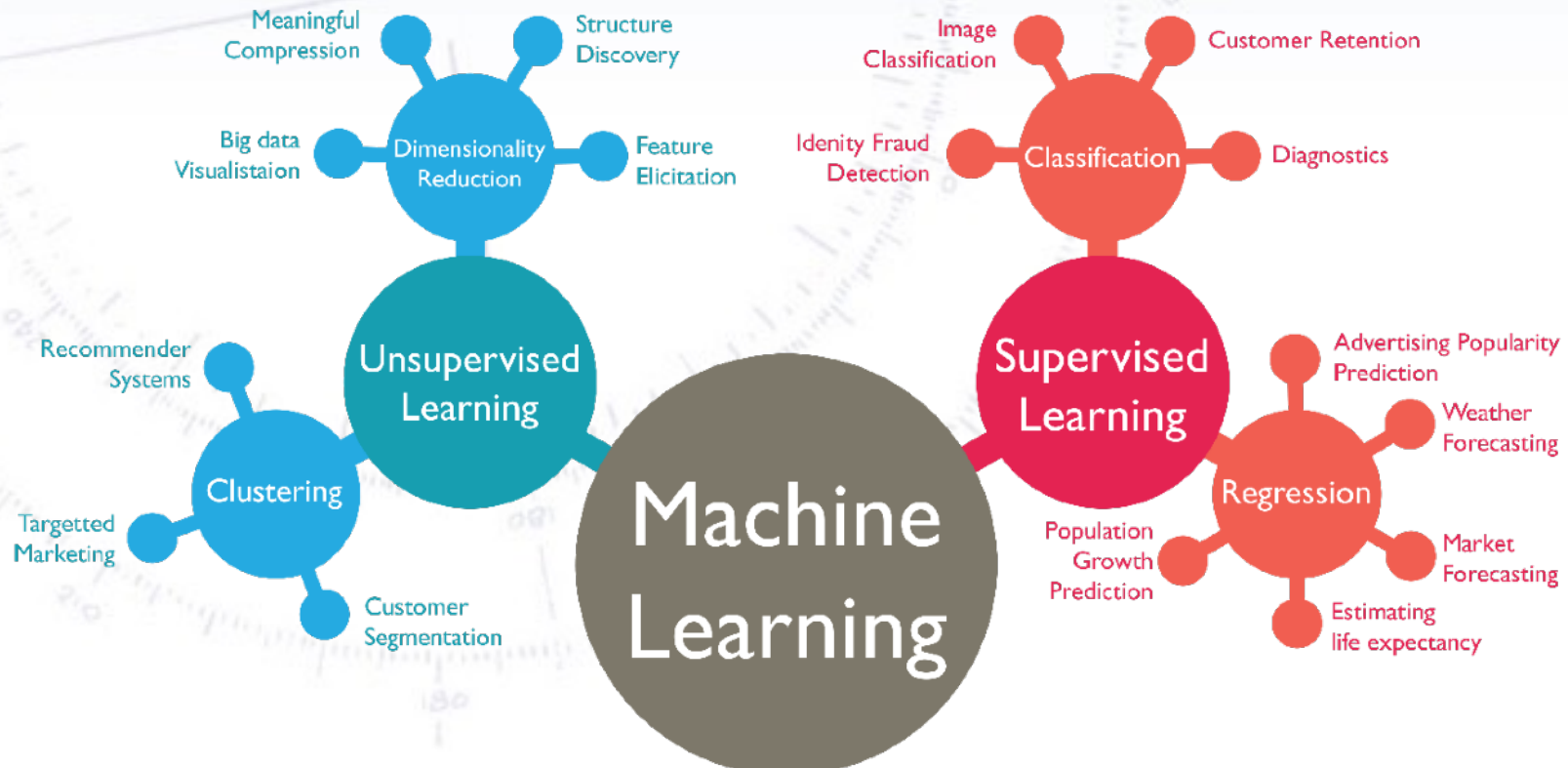


# Types of ML



# Unsupervised vs. Supervised Classification vs. Regression

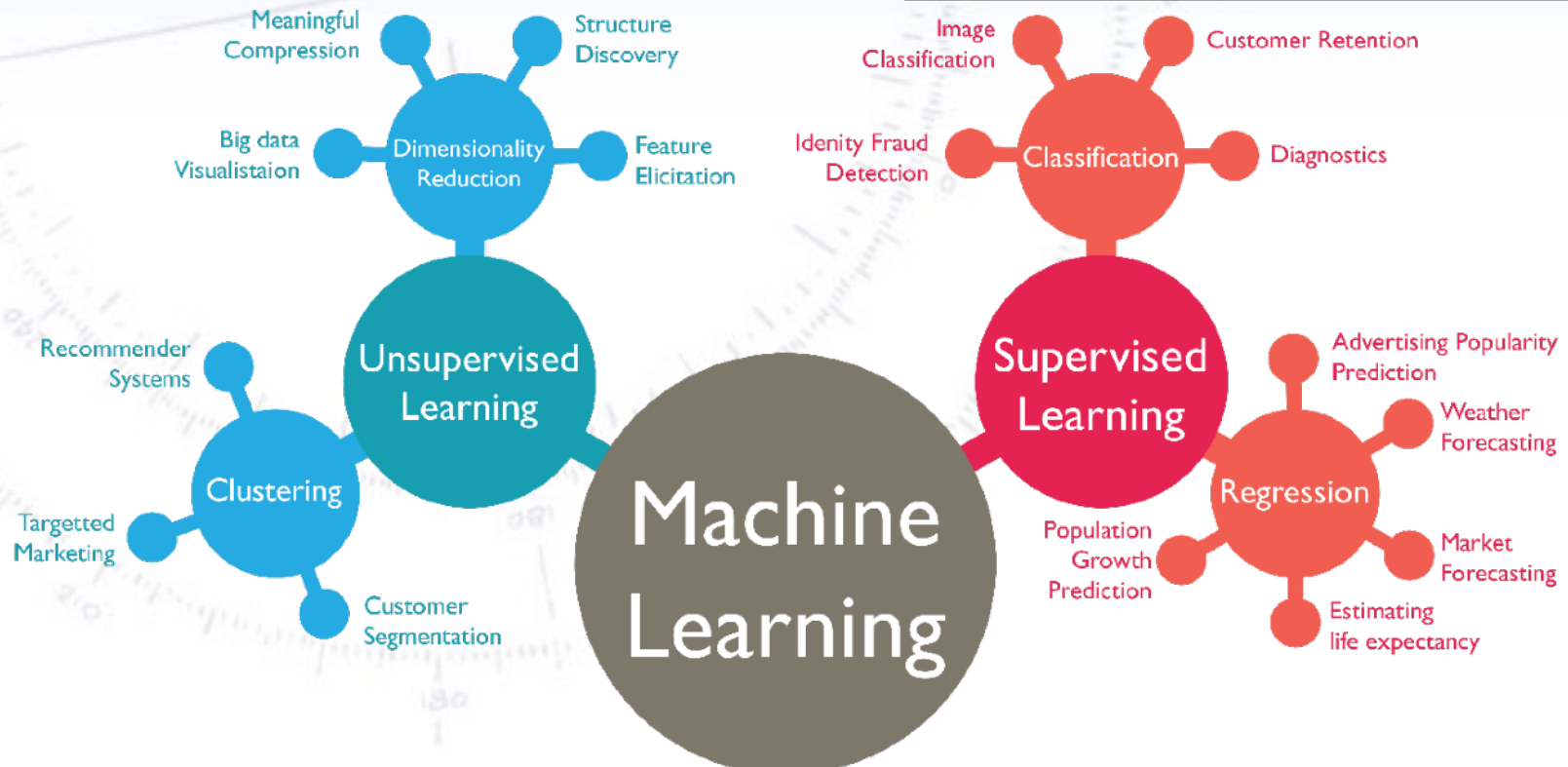
Machine Learning can be supervised (you have correctly labelled examples) or unsupervised (you don't)... [or reinforced]. Following this, one can be using ML to either classify (is it A or B?) or for regression (estimate of X).



# Unsupervised vs. Supervised Classification vs. Regression

Machine Learning can be supervised (you have correctly labelled examples) or unsupervised (you don't)... [or reinforced]. Following this, one can be using ML to either classify (is it A or B?) or for regression (estimate of X).

**We will be mostly on this side!**

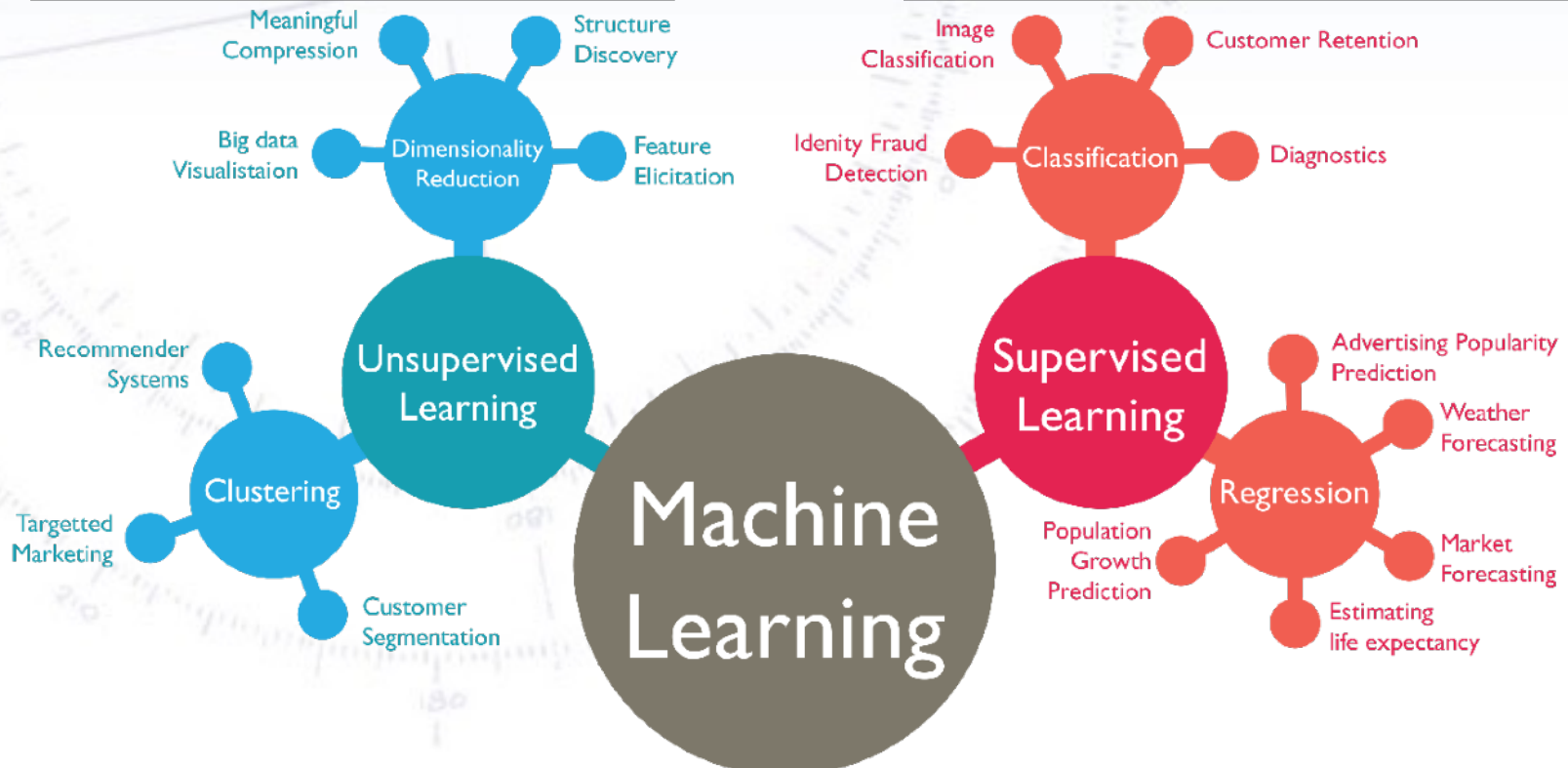


# Unsupervised vs. Supervised Classification vs. Regression

Machine Learning can be supervised (you have correctly labelled examples) or unsupervised (you don't)... [or reinforced]. Following this, one can be using ML to either classify (is it A or B?) or for regression (estimate of X).

**But of course also over here!**

**We will be mostly on this side!**

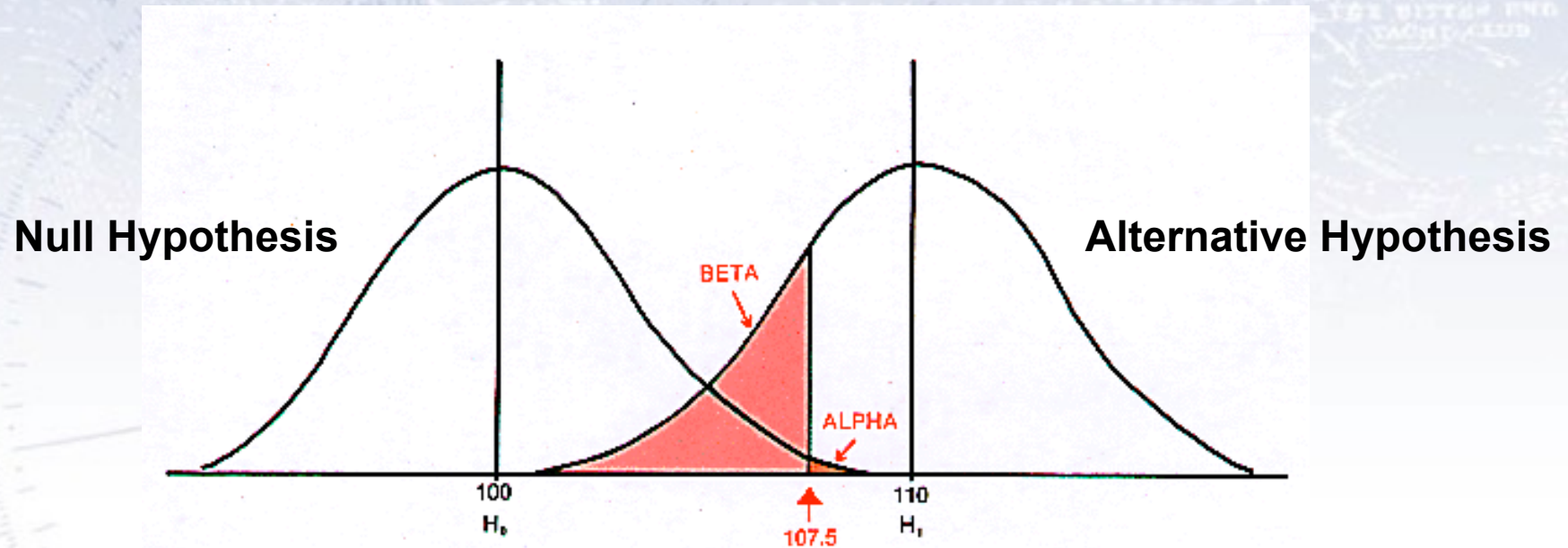






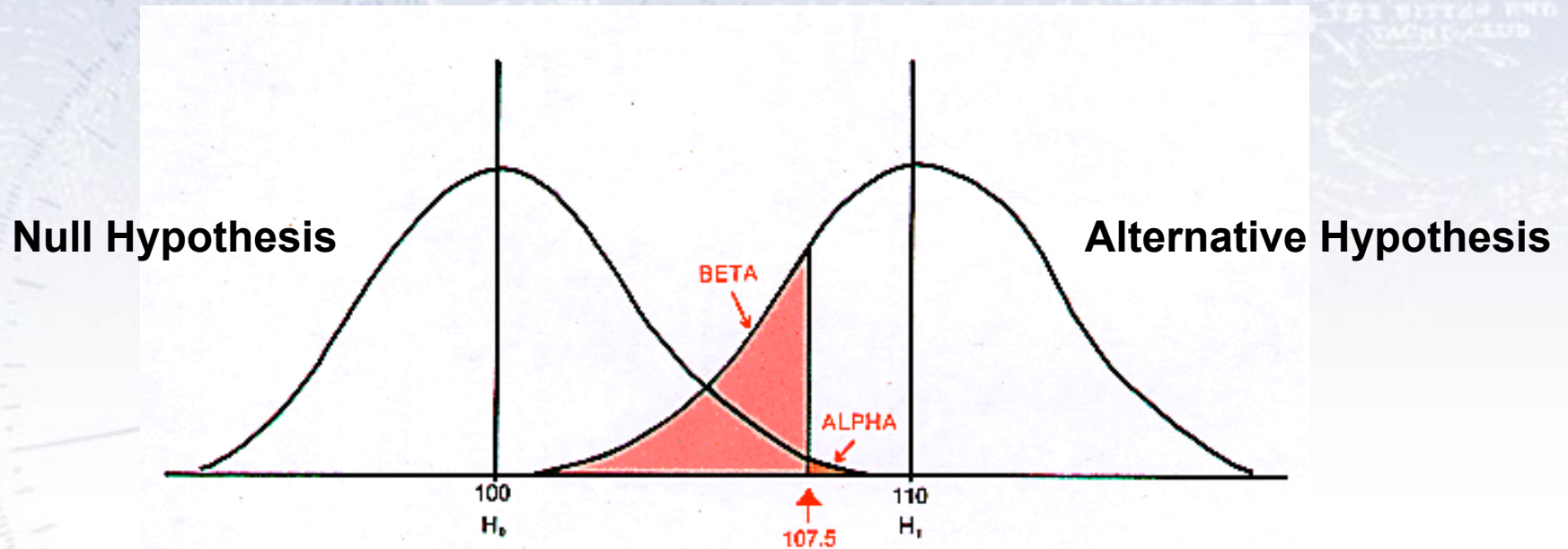
# Target of ML

# Classification



		REALITY	
		Null is True	Null is False
STATISTICAL DECISION:	Do Not Reject Null	$1 - \alpha$ Correct	$\beta$ Type II error
	Reject Null	$\alpha$ Type I error	$1 - \beta$ Correct

# Classification



Machine Learning typically enables a better separation between hypothesis

**DECISION:**

Reject Null

$\alpha$ Type I error	$1 - \beta$ Correct
--------------------------	------------------------



# Hypothesis testing

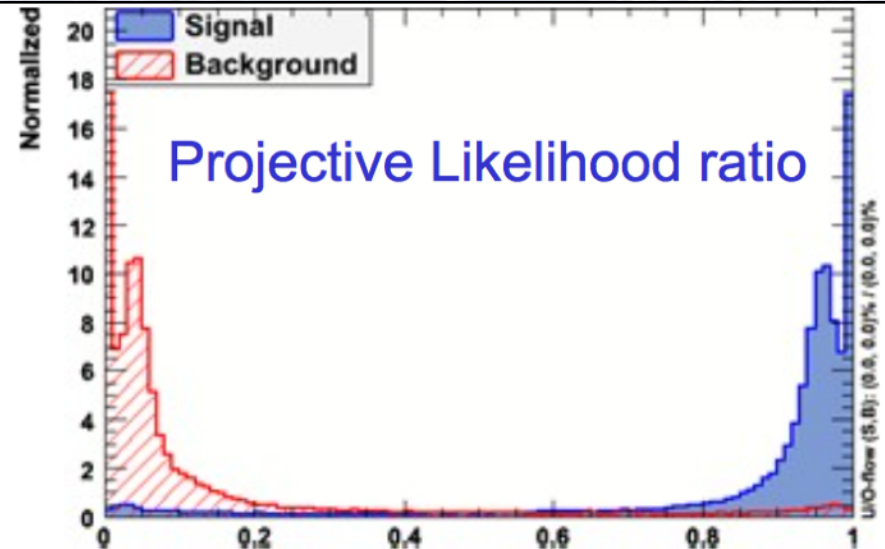
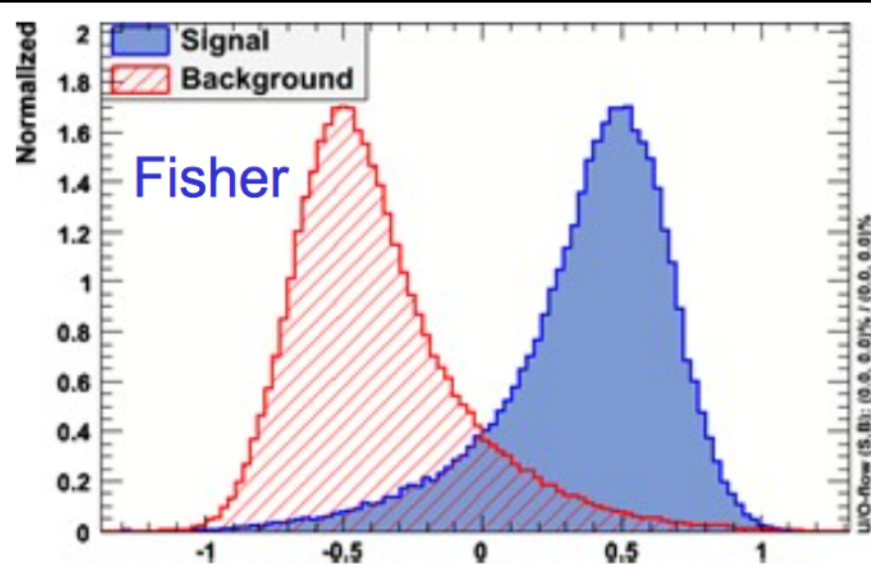
Hypothesis testing is like a criminal trial. The basic “null” hypothesis is **Innocent** (called  $H_0$ ) and this is the hypothesis we want to test, compared to an “alternative” hypothesis, **Guilty** (called  $H_1$ ).

Innocence is initially assumed, and this hypothesis is only rejected, if enough evidence proves otherwise, i.e. that the probability of innocence is very small (“beyond reasonable doubt”). This is summarised in a **Contingency Table**:

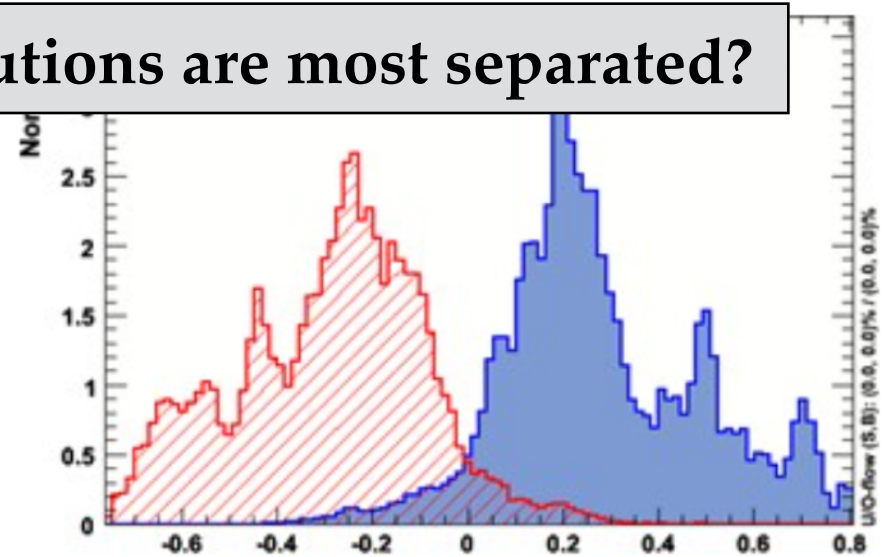
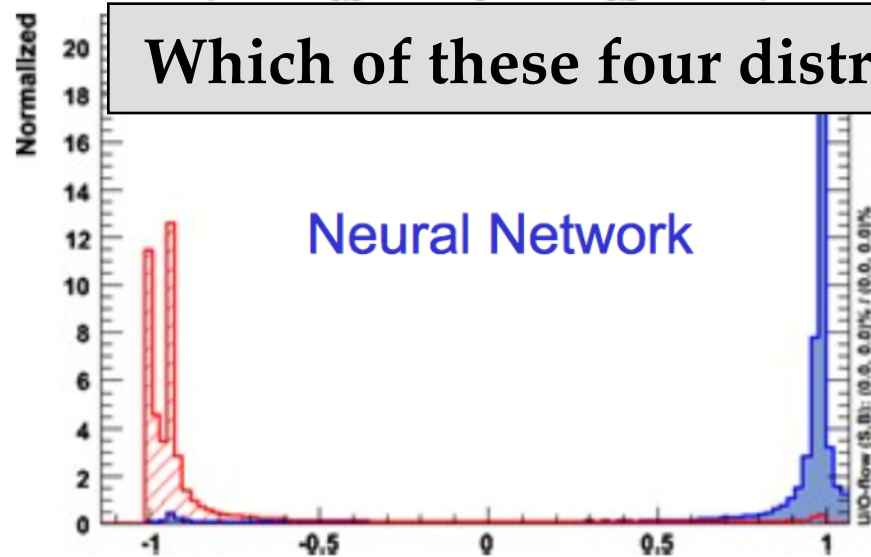
	Truly innocent ( $H_0$ is true)	Truly guilty ( $H_1$ is true)
Acquittal (Accept $H_0$ )	Right decision True Positive (TP)	<b>Wrong decision</b> False Negative (FN)
Conviction (Reject $H_0$ )	<b>Wrong decision</b> False Positive (FP)	Right decision True Negative (TN)

The rate of FP and FN are correlated, and one can only choose one of these!

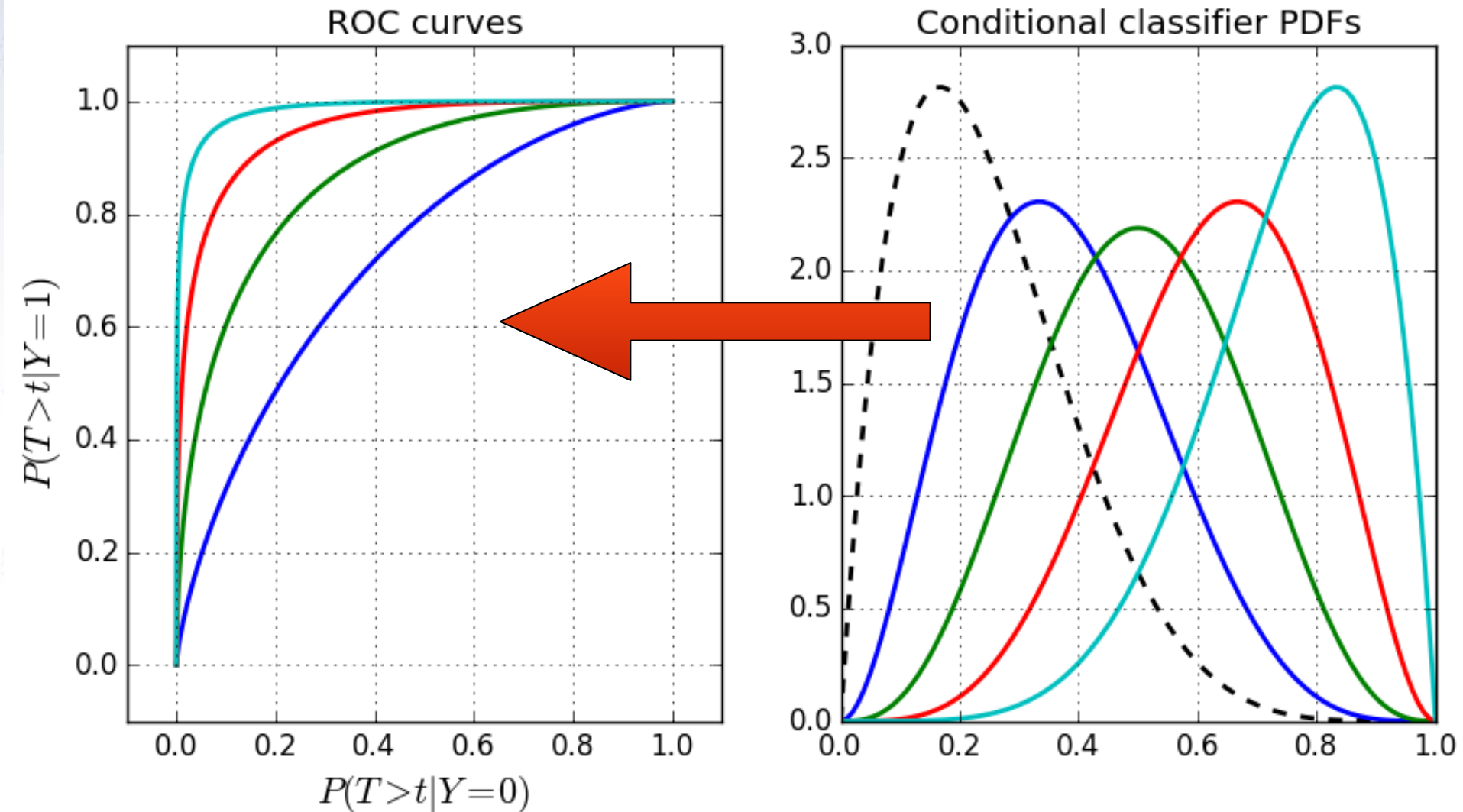
# Measuring separation



Which of these four distributions are most separated?

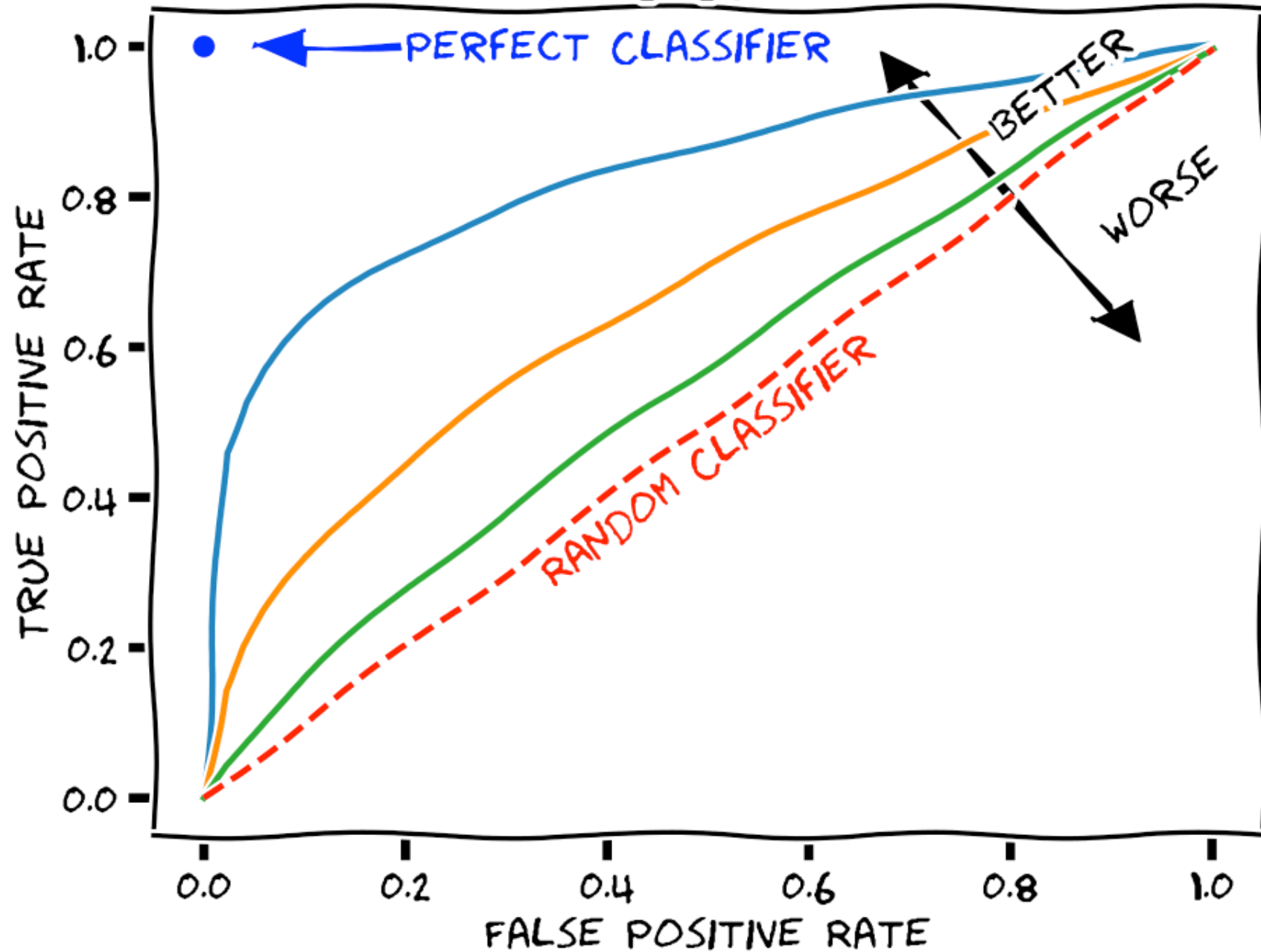


# Simple case

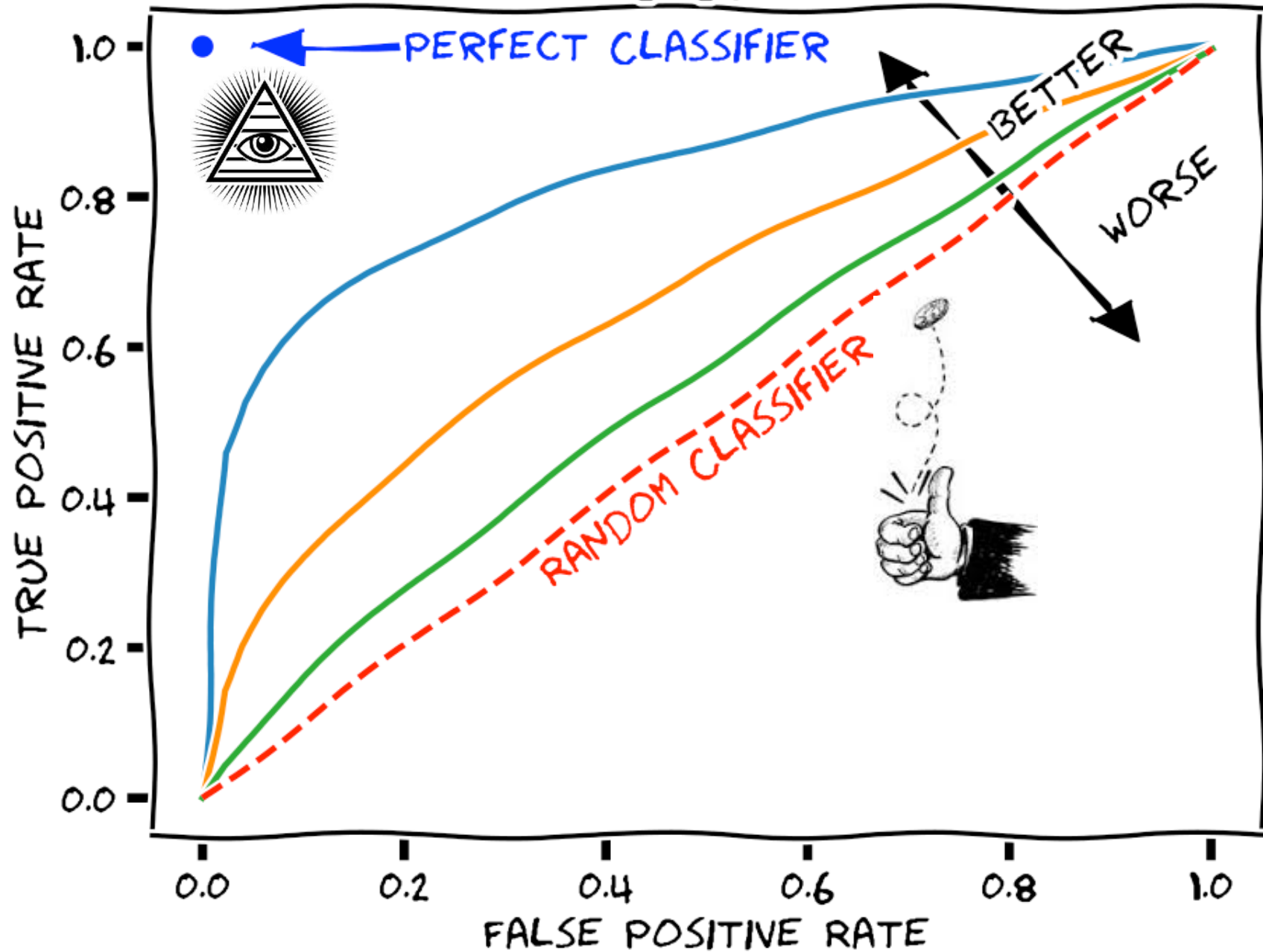




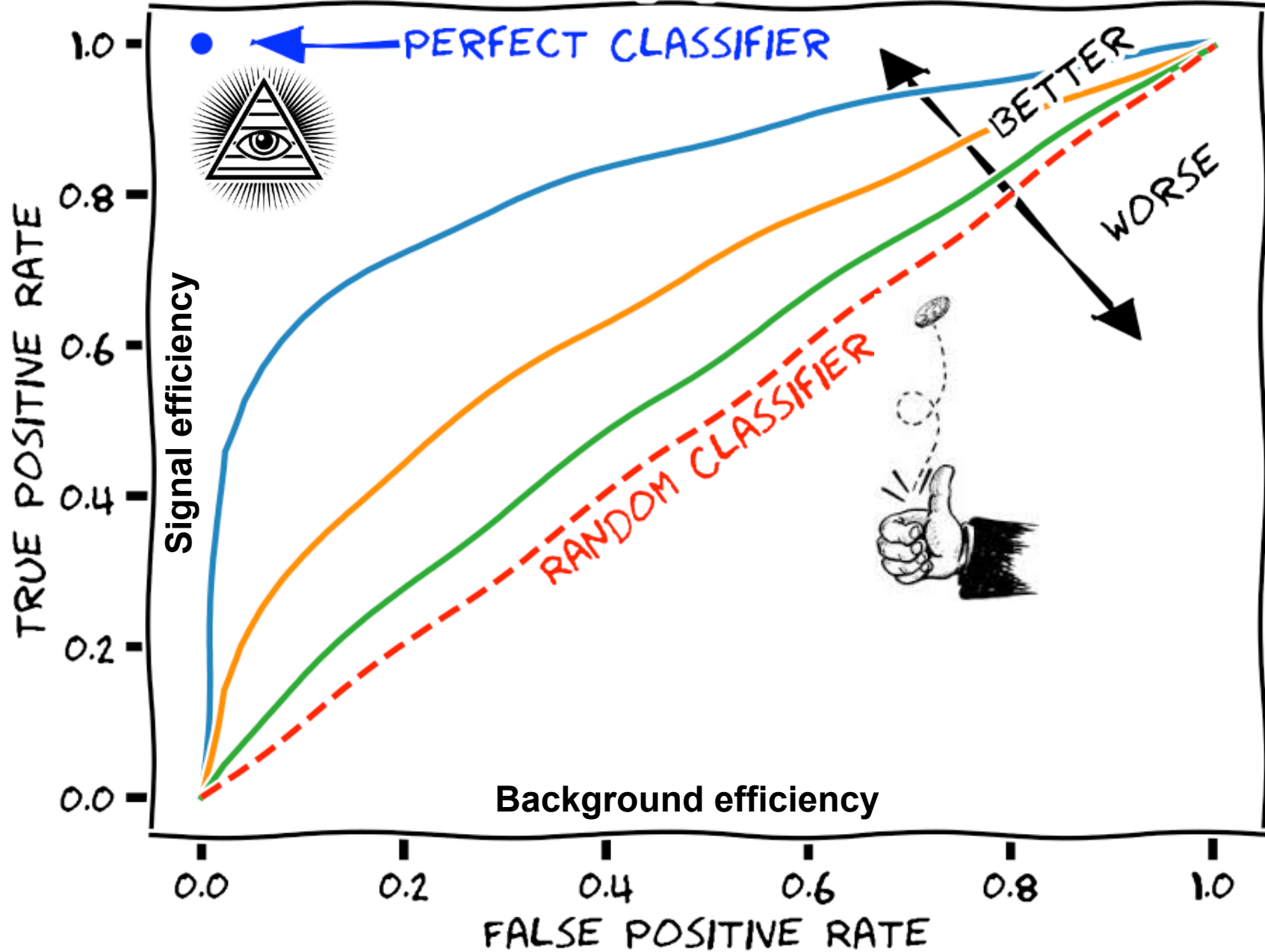
# ROC CURVE



# ROC CURVE

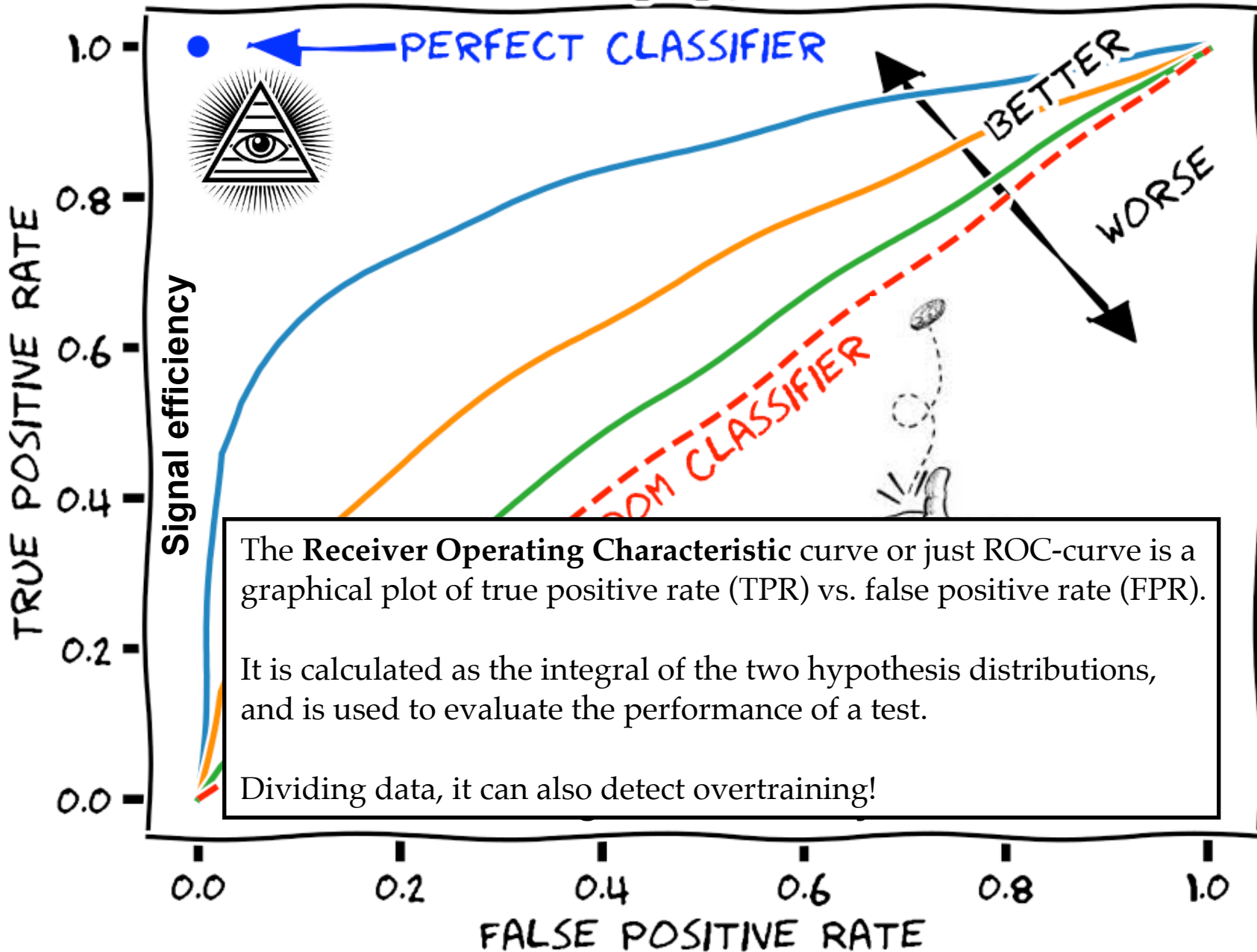


# ROC CURVE





# ROC CURVE



# Which metric to use?

There are a ton of metrics in hypothesis testing, see below. However, those in the boxes below are the most central ones.

One metric - not mentioned here - is the Area Under the Curve (AUC), which is simply an integral of the ROC curve (thus 1 is perfect score). This is sometimes used to optimise performance (loss), but not great!

		True condition			
Total population		Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$
Predicted condition	Predicted condition positive	True positive	False positive, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$
	Predicted condition negative	False negative, Type II error	True negative	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR-}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR-) = $\frac{\text{FNR}}{\text{TNR}}$	
				F <sub>1</sub> score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$	

# Matthew's Correlation Coefficient

Given a Contingency Table:

	Got well	Remained ill
Medicin	28	5
No Medicin	19	9

One of the commonly used measures of separation the MCC, which (in this case) is the Pearson  $\phi$ , and related to the ChiSquare:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Read more at:

[https://en.wikipedia.org/wiki/Phi\\_coefficient](https://en.wikipedia.org/wiki/Phi_coefficient)

However, when optimising an algorithm and giving continuous scores in the range  $]0,1[$ , there are other things to consider (see talk on Loss Functions).





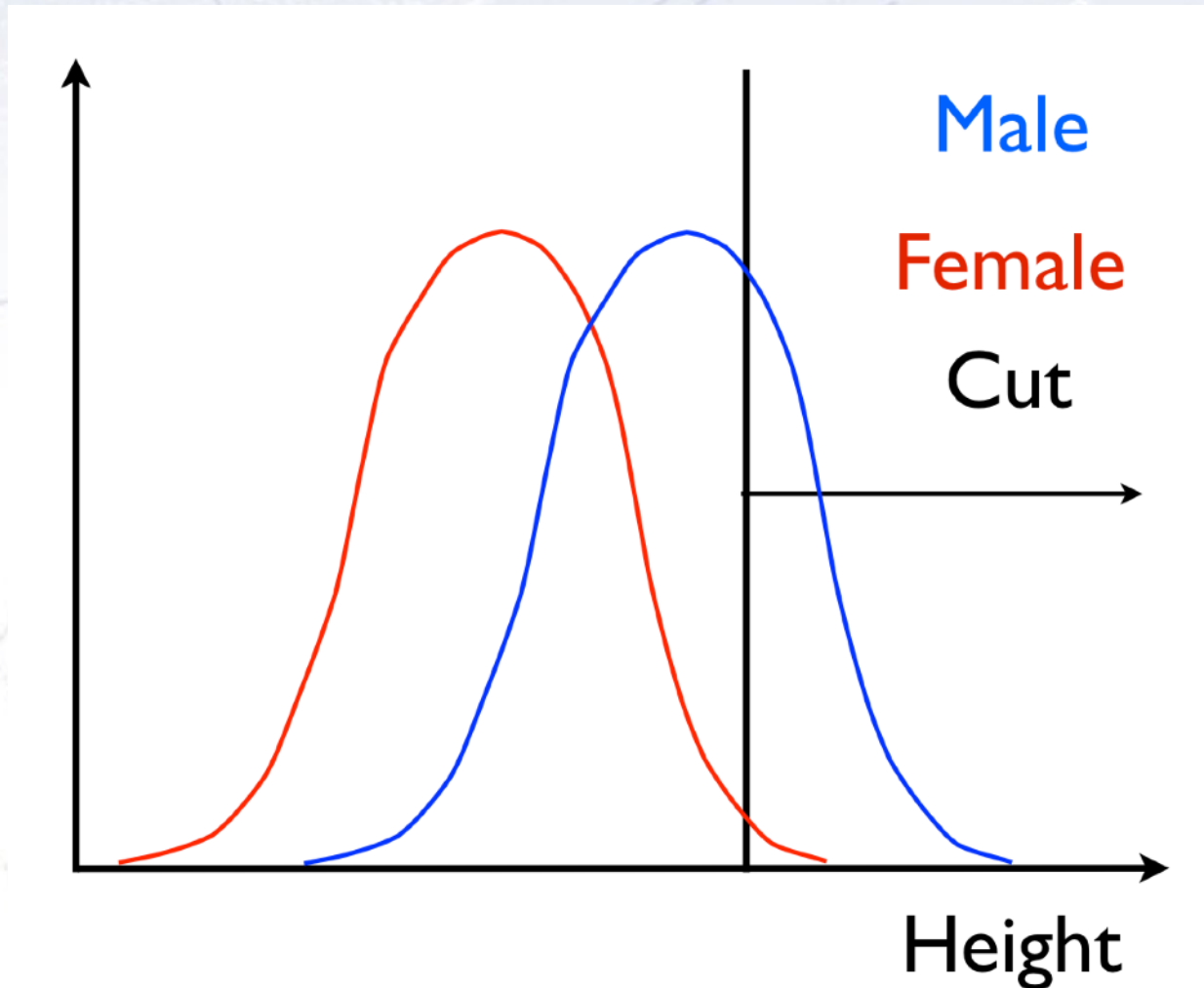
The background is a faded nautical chart. It features concentric circles representing magnetic isotherms, with labels such as 0, 30, 60, 90, 120, 150, 180, 210, 240, 270, and 300. A compass rose is visible, with a label 'VAR 10° 15' W'. In the upper right, there is a label 'THE BITTER END YACHT CLUB'. The text 'MAGNETIC' is also visible near the top center.

# The linear analysis case

# Simple Example

**Problem:** You want to figure out a method for getting sample that is mostly male!

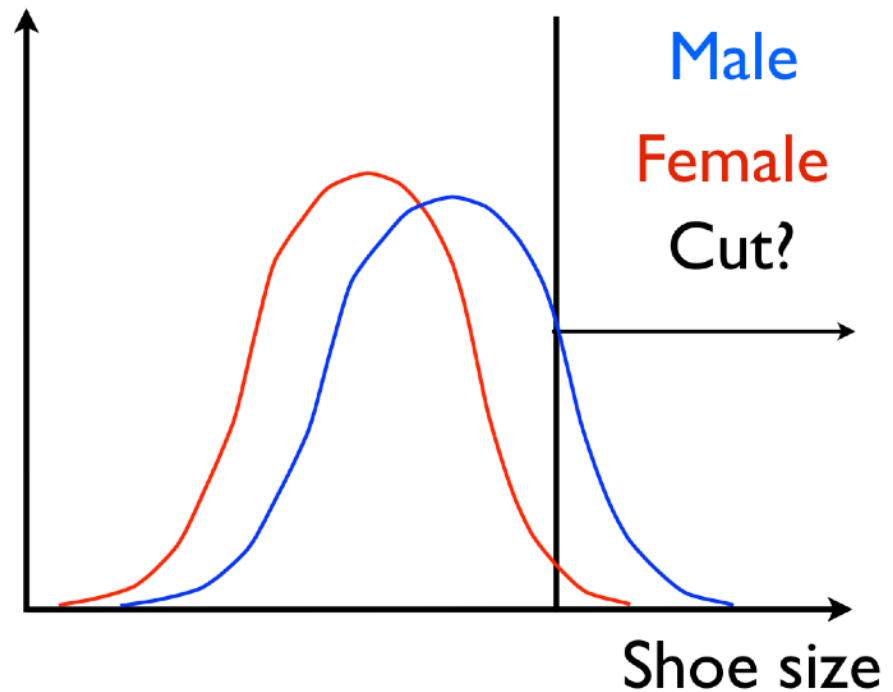
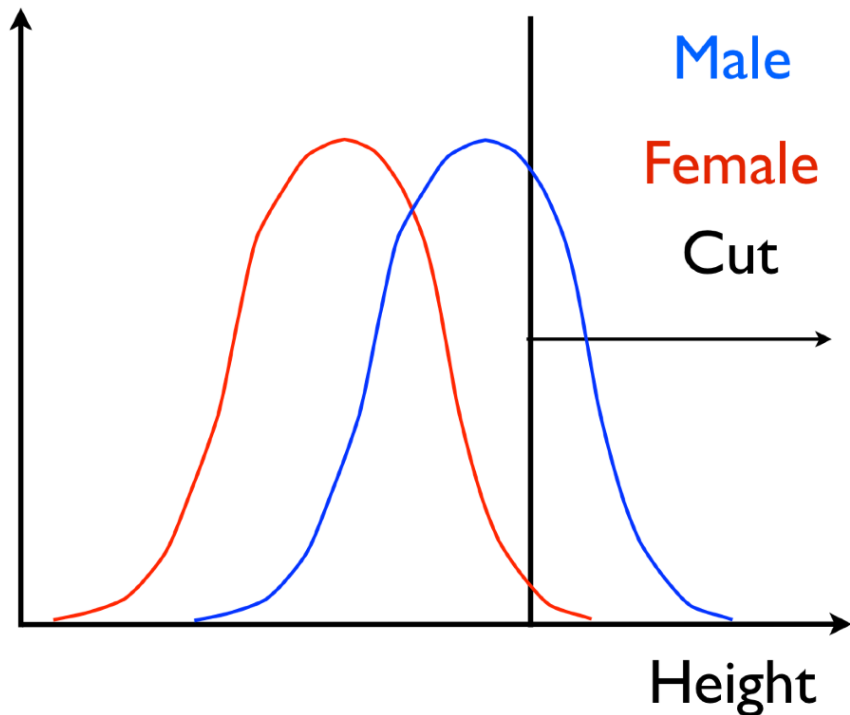
**Solution:** Gather height data from 10000 people, Estimate cut with 95% purity!



# Simple Example

**Additional data:** The data you find also contains shoe size!

**How to use this?** Well, it is more information, but should you cut on it?



The question is, what is the best way to use this (possibly correlated) information!



# Simple Example

So we look if the data is correlated, and consider the options:

Cut on each var?

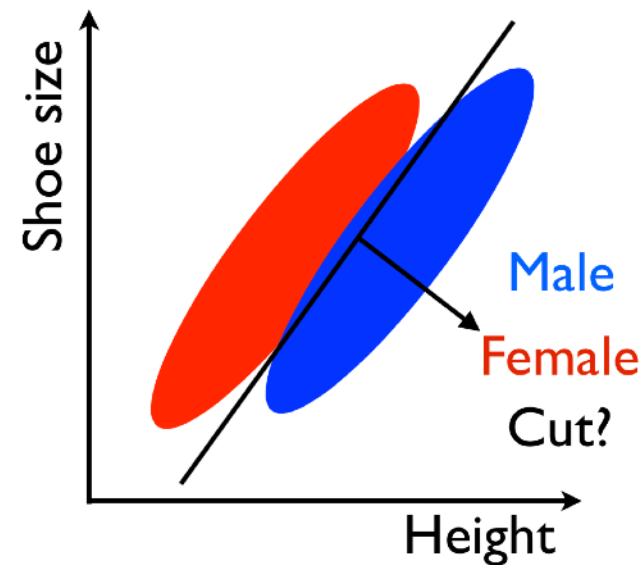
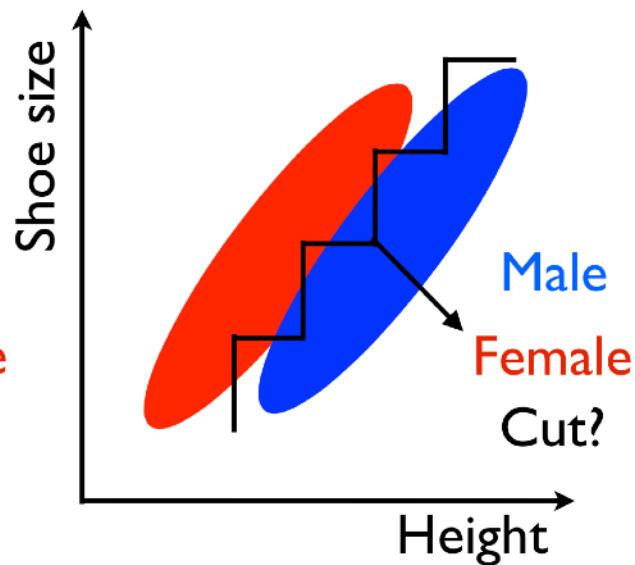
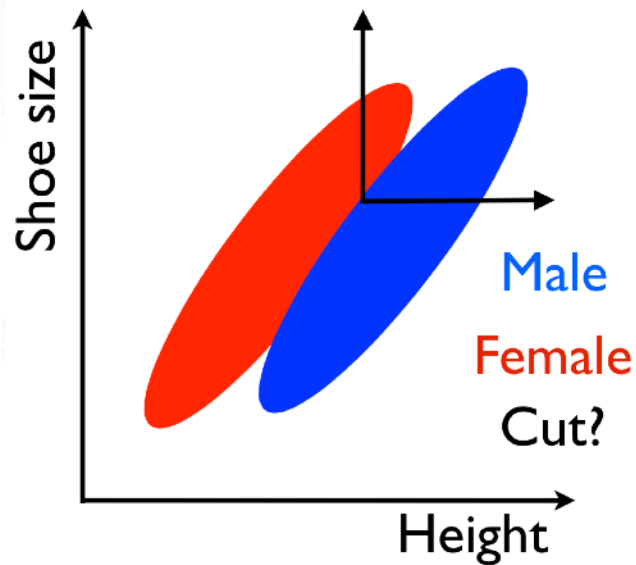
**Poor efficiency!**

Advanced cut?

**Clumsy and  
hard to implement**

Combine var?

**Smart and  
promising**



The latter approach is the Fisher discriminant!

It has the advantage of being simple and applicable in many dimensions easily!

# Simple Example

So we look if the data is correlated, and consider the options:

**Cut on each var?**

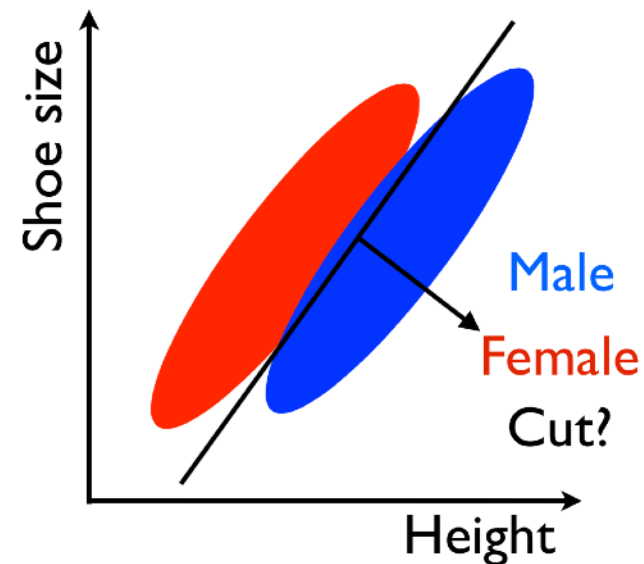
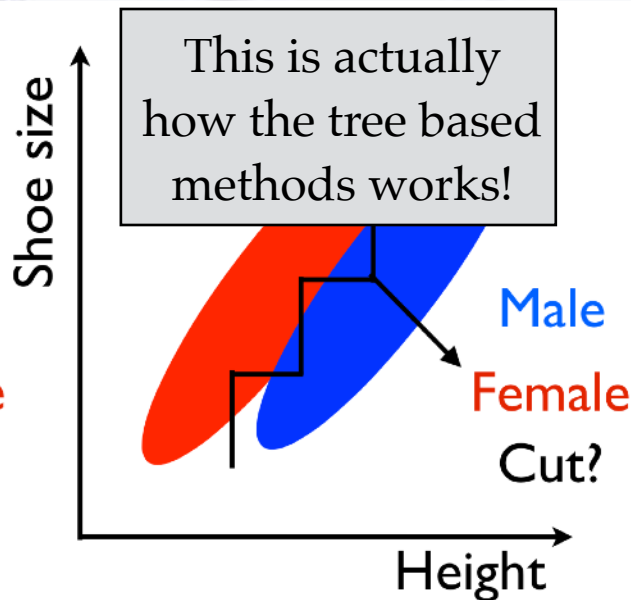
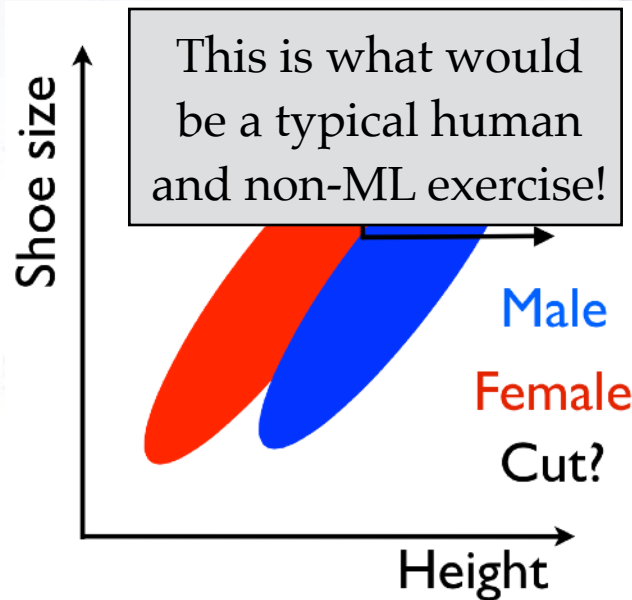
**Poor efficiency!**

**Advanced cut?**

**Clumsy and  
hard to implement**

**Combine var?**

**Smart and  
promising**



The latter approach is the Fisher discriminant!




It has the advantage of being simple and applicable in many dimensions easily!

# Separating data

Fisher's friend, Anderson, came home from picking Irises in the Gaspé peninsula...

## 180 MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS

Table I

<i>Iris setosa</i>				<i>Iris versicolor</i>				<i>Iris virginica</i>			
Sepal length	Sepal width	Petal length	Petal width	Sepal length	Sepal width	Petal length	Petal width	Sepal length	Sepal width	Petal length	Petal width
5.1	3.5	1.4	0.2	7.0	3.2	4.7	1.4	6.3	3.3	6.0	2.5
4.9	3.0	1.4	0.2	6.4	3.2	4.5	1.5	5.8	2.7	5.1	1.9
4.7	3.2	1.3	0.2	6.9	3.1	4.9	1.5	7.1	3.0	5.9	2.1
4.6	3.1	1.5	0.2	5.5	2.3	4.0	1.3	6.3	2.9	5.6	1.8
											
5.8	4.0	1.2	0.2	5.6	2.9	3.6	1.3	5.8	2.8	5.1	2.4
5.7	4.4	1.5	0.4	6.7	3.1	4.4	1.4	6.4	3.2	5.3	2.3
5.4	3.9	1.3	0.4	5.6	3.0	4.5	1.5	6.5	3.0	5.5	1.8
5.1	3.5	1.4	0.3	5.8	2.7	4.1	1.0	7.7	3.8	6.7	2.2
5.7	3.8	1.7	0.3	6.2	2.2	4.5	1.5	7.7	2.6	6.9	2.3



# Fisher's Linear Discriminant

You want to separate two types/classes (A and B) of events using several measurements.

Q: How to combine the variables?

A: Use the Fisher Discriminant:

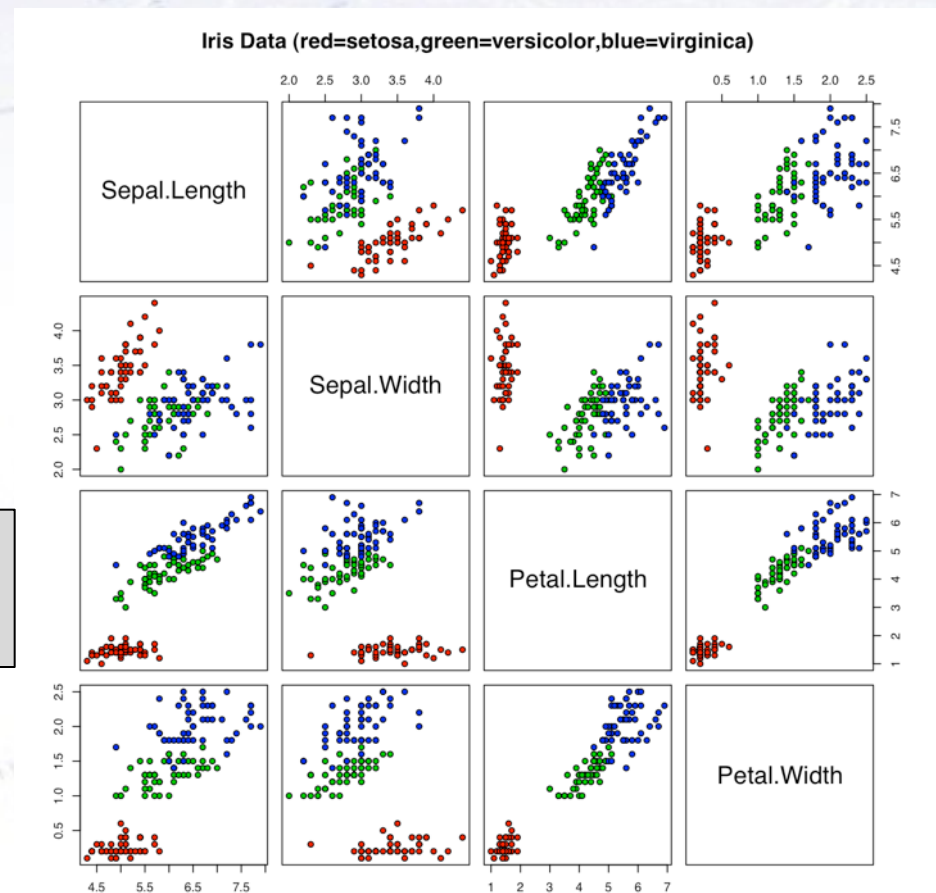
$$\mathcal{F} = w_0 + \vec{w} \cdot \vec{x}$$

Q: How to choose the values of  $w$ ?

A: Inverting the covariance matrices:

$$\vec{w} = (\Sigma_A + \Sigma_B)^{-1} (\vec{\mu}_A - \vec{\mu}_B)$$

This can be calculated analytically, and incorporates the linear correlations into the separation capability.





# Fisher's Linear Discriminant

You want to separate two types/classes (A and B) of events using several measurements.

**Q:** How to combine the variables?

**A:** Use the Fisher Discriminant:

ments are given. We shall first consider the question: What linear function of the four measurements

$$X = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 + \lambda_4 x_4$$

will maximize the ratio of the difference between the specific means to the standard deviations within species? The observed means and their differences are shown in Table II.

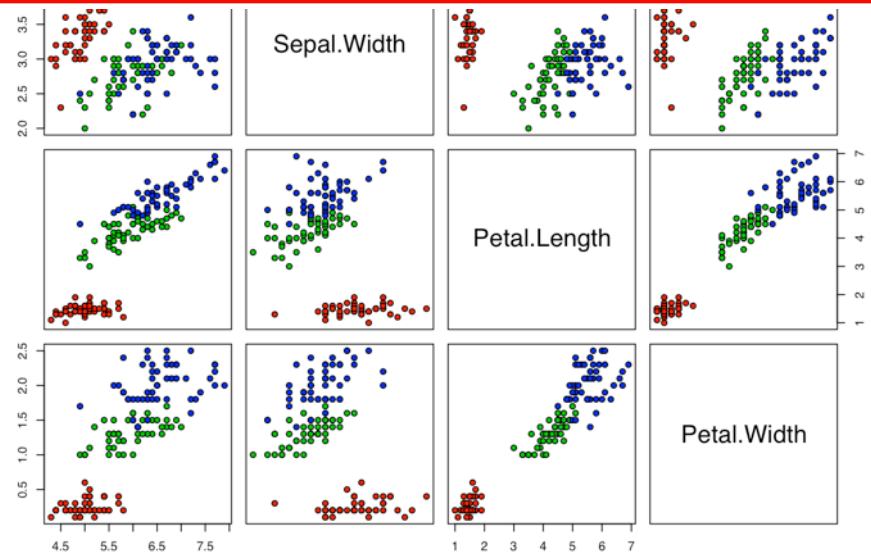
**Q:** How to choose the values of  $w$ ?

**A:** Inverting the covariance matrices:

$$\vec{w} = (\Sigma_A + \Sigma_B)^{-1} (\vec{\mu}_A - \vec{\mu}_B)$$

This can be calculated analytically, and incorporates the linear correlations into the separation capability.

Iris Data (red=setosa,green=versicolor,blue=virginica)



# Fisher's Linear Discriminant

The details of the formula are outlined below:

You have two samples, A and B, that you want to separate.

For each input variable (x), you calculate the mean ( $\mu$ ), and form a vector of these.

$$\vec{w} = (\Sigma_A + \Sigma_B)^{-1} (\vec{\mu}_A - \vec{\mu}_B)$$

Using the input variables (x), you calculate the covariance matrix ( $\Sigma$ ) for each species (A/B), add these and invert.

Given weights (w), you take your input variables (x) and combine them linearly as follows:

$$\mathcal{F} = w_0 + \vec{w} \cdot \vec{x}$$

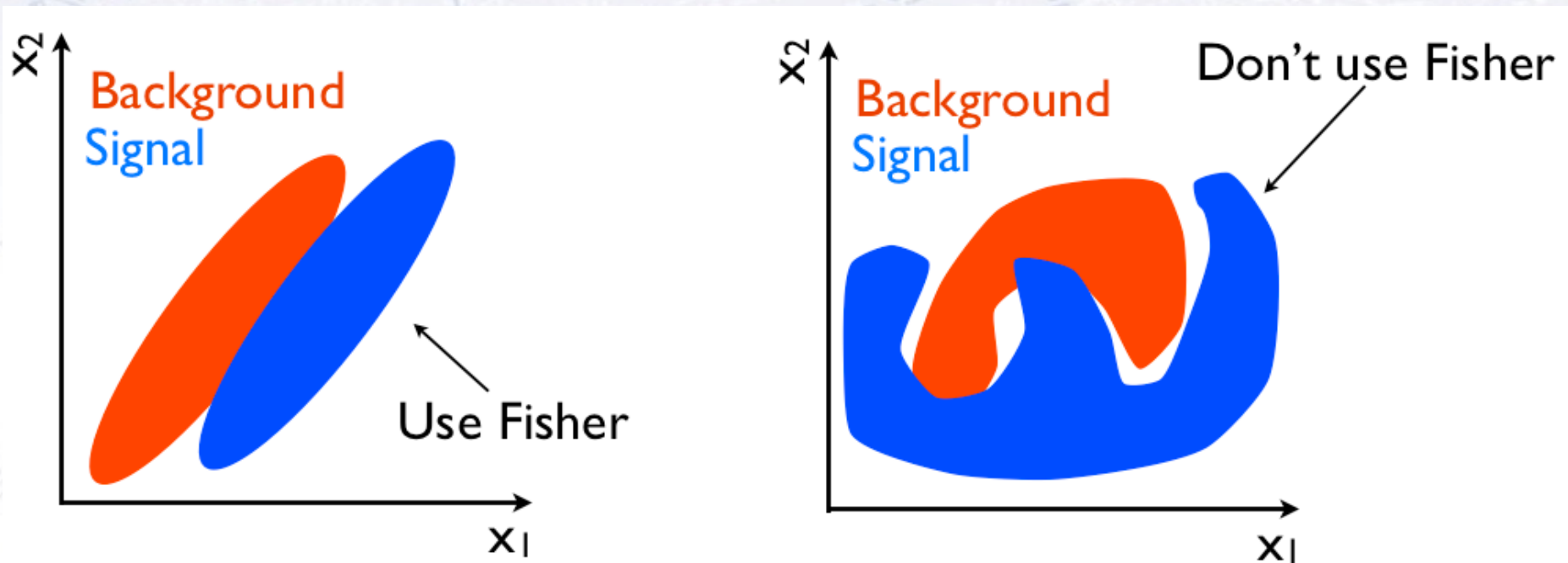
F is what you base your decision on.



# The non-linear case

# Non-linear cases

While the Fisher Discriminant uses all separations and **linear correlations**, it does not perform optimally, when there are **non-linear correlations** present:



If the PDFs of signal and background are known, then one can use a likelihood. But this is **very rarely** the case, and hence one should move on to the Fisher. However, if correlations are non-linear, more “tough” methods are needed...



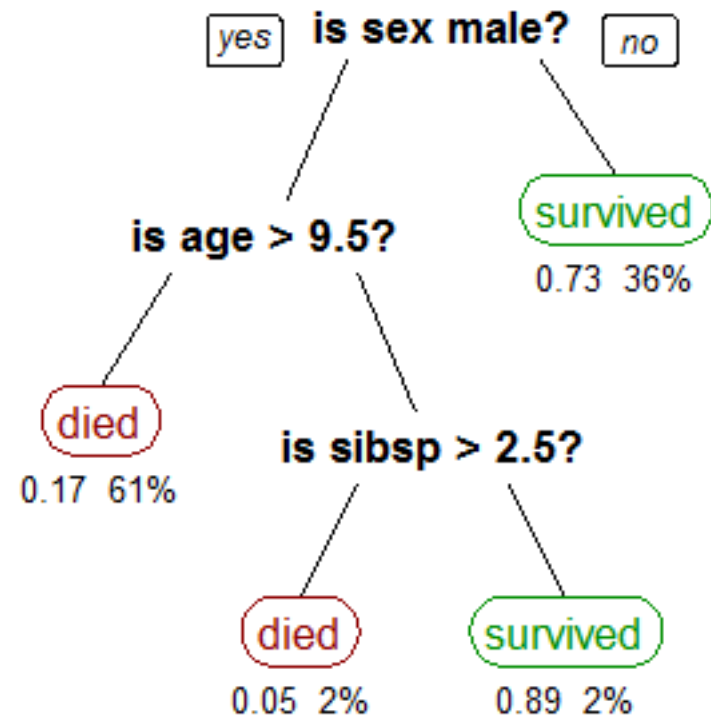
# (Boosted) Decision Trees

Can become very complex.

Good for discrete problems.  
“Good for all problems!!!”

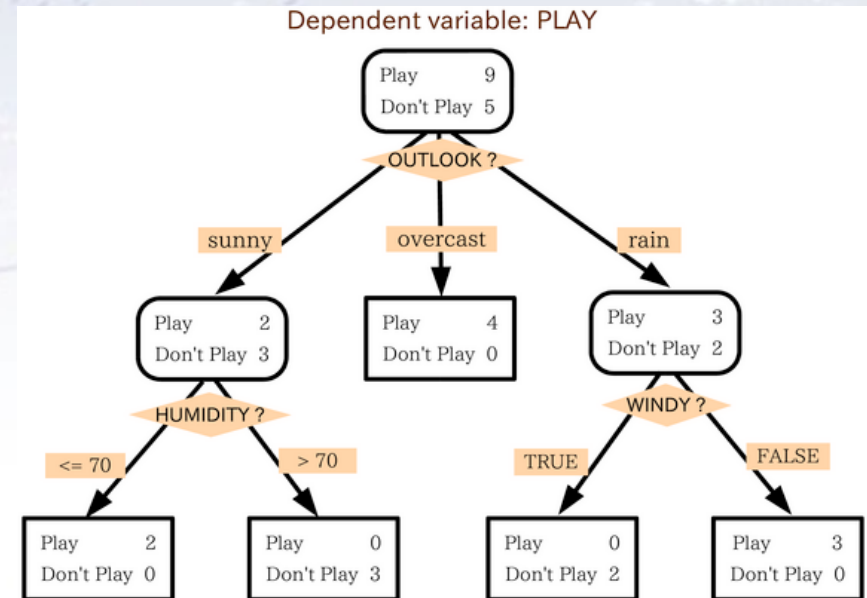
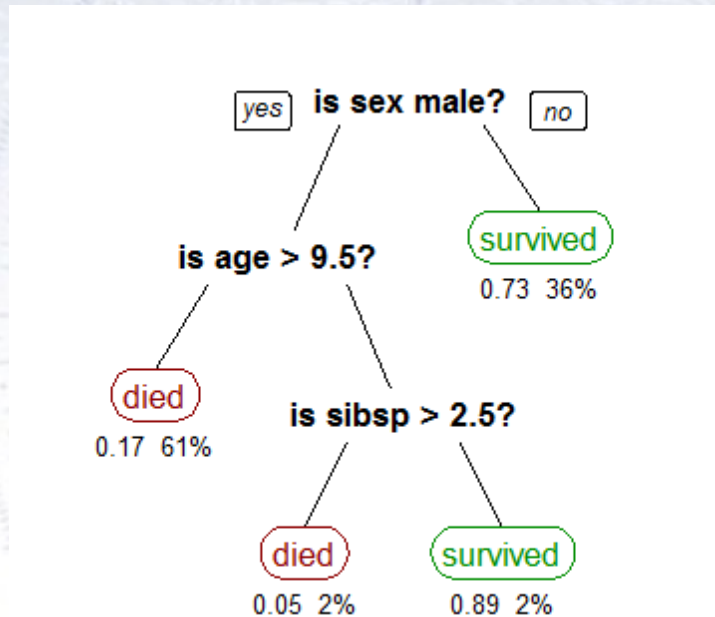
Not always highest efficiency,  
though...

Boosting adds to separation.



\* Example decision tree on a simple algorithm for predicting survival of Titanic!

# Boosted Decision Trees (BDT)



*Decision tree learning uses a **decision tree** as a **predictive model** which maps observations about an item to conclusions about the item's target value. It is one of the predictive modelling approaches used in **statistics**, **data mining** and **machine learning**.*

[Wikipedia, Introduction to Decision Tree Learning]

# Neural Networks

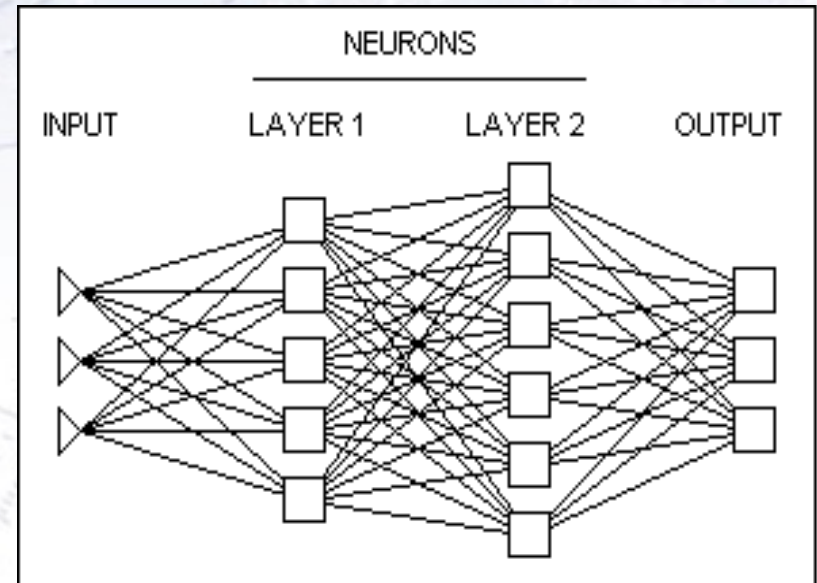
Can become very complex.

Good for continuous problems.

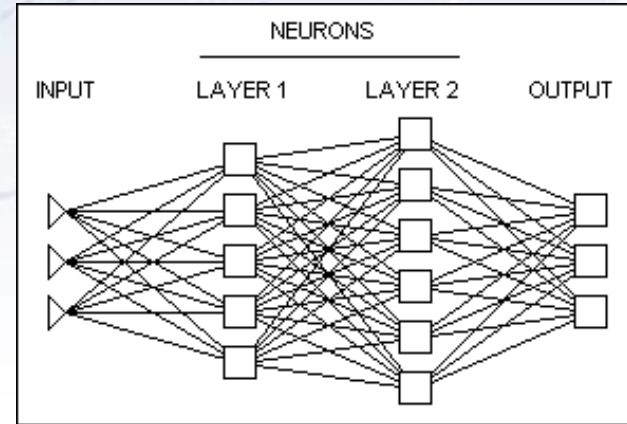
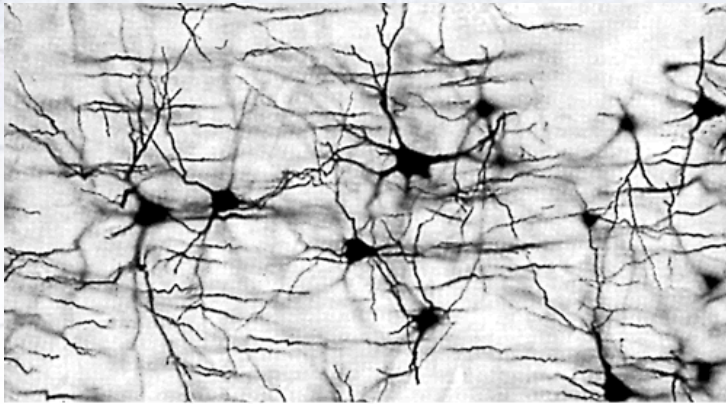
Sometimes hard to train!

Very versatile approach that can also be applied to images, text, etc.

Easily produces multiple outputs.



# Neural Networks (NN)

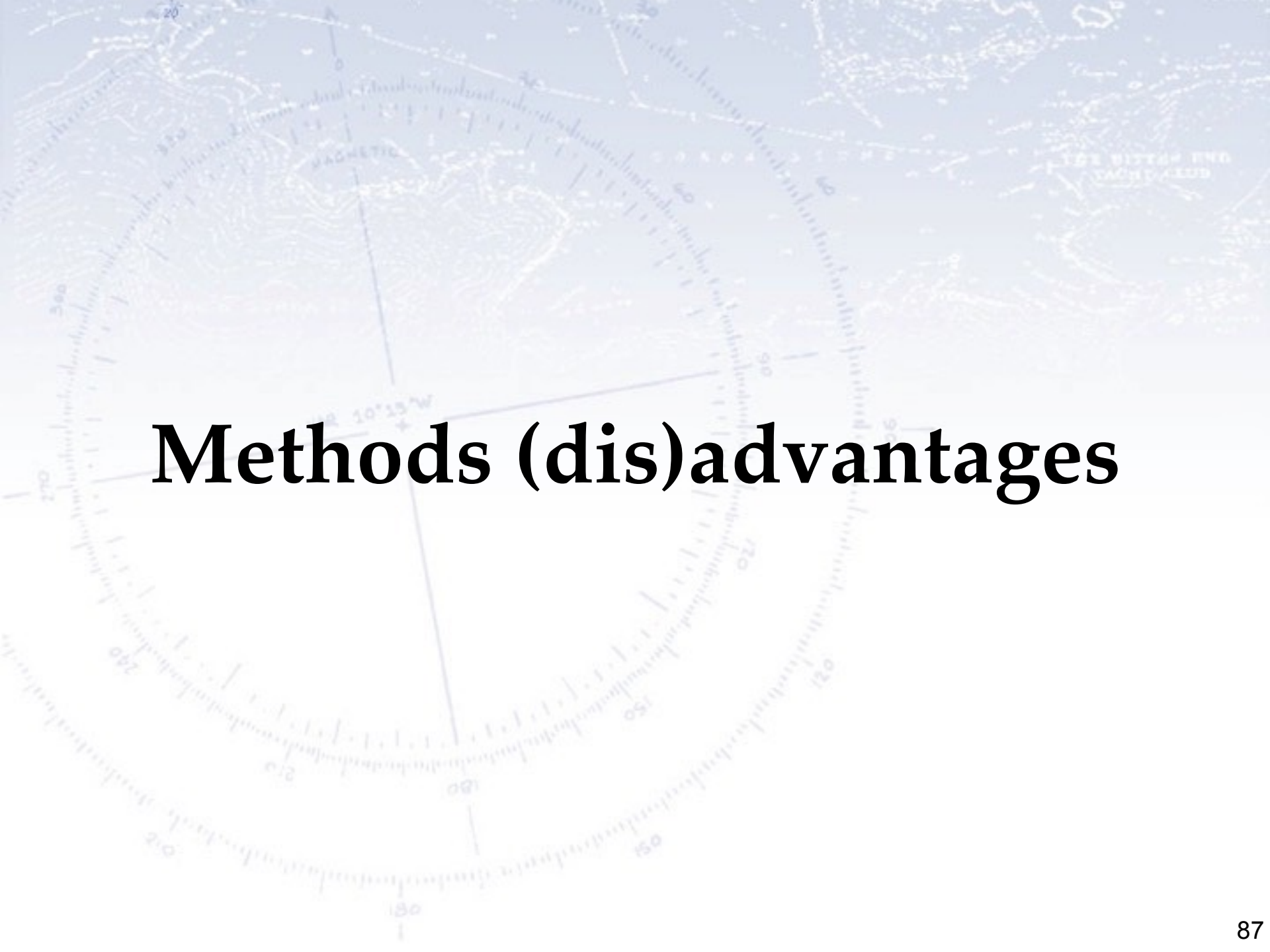


*In machine learning and related fields, artificial neural networks (ANNs) are computational models inspired by an animal's central nervous systems (in particular the brain) which is capable of **machine learning** as well as **pattern recognition**.*

***Neural networks** have been used to solve a wide variety of tasks that are hard to solve using ordinary rule-based programming, including **computer vision** and **speech recognition**.*

[Wikipedia, Introduction to Artificial Neural Network]





The background is a nautical chart of the North Atlantic Ocean. It features concentric magnetic isotherms (lines of equal magnetic intensity) labeled with values such as 300, 250, 200, 150, 100, 50, and 0. A prominent line is labeled 'MAGNETIC'. Another line is labeled '10° 15' W'. In the upper right corner, the text 'BITTER END YACHT CLUB' is visible. The chart also shows various navigational lines and coordinates.

# Methods (dis)advantages

# Method's (dis-)advantages

Another comparison is done in Elements of Statistical Learning II (ESL II), where linear methods are not included.

As can be seen, Neural Networks are “difficult” in almost all respects, but performant.

For trees, the case is almost the opposite.

However, I don't agree with the evaluation of the predictive power of trees.

At least not for normal structured data.

Characteristic	Neural Nets	SVM	Trees	MARS	k-NN, Kernels
Natural handling of data of “mixed” type	▼	▼	▲	▲	▼
Handling of missing values	▼	▼	▲	▲	▲
Robustness to outliers in input space	▼	▼	▲	▼	▲
Insensitive to monotone transformations of inputs	▼	▼	▲	▼	▼
Computational scalability (large $N$ )	▼	▼	▲	▲	▼
Ability to deal with irrelevant inputs	▼	▼	▲	▲	▼
Ability to extract linear combinations of features	▲	▲	▼	▼	◆
Interpretability	▼	▼	◆	▲	▼
Predictive power	▲	▲	▼	◆	▲

# Method's (dis-)advantages

Another comparison is done in Elements of Statistical Learning II (ESL II), where linear methods are not included.

As can be seen, Neural Networks are “difficult” in almost all respects, but performant.

For trees, the case is almost the opposite.

However, I don't agree with the evaluation of the predictive power of trees.

At least not for normal structured data.

**For tabular data, I disagree!**

Characteristic	Neural Nets	SVM	Trees	MARS	k-NN, Kernels
Natural handling of data of “mixed” type	▼	▼	▲	▲	▼
Handling of missing values	▼	▼	▲	▲	▲
Robustness to outliers in input space	▼	▼	▲	▼	▲
Insensitive to monotone transformations of inputs	▼	▼	▲	▼	▼
Computational scalability (large $N$ )	▼	▼	▲	▲	▼
Ability to deal with irrelevant inputs	▼	▼	▲	▲	▼
Ability to extract linear combinations of features	▲	▲	▼	▼	◆
Interpretability	▼	▼	◆	▲	▼
Predictive power	▲	▲	▼	◆	▲

# Method's (dis-)advantages

Another comparison is done in Elements of Statistical Learning II (ESL II), where linear methods are not included.

As can be seen, Neural Networks are “difficult” in almost all respects, but performant.

For trees, the case is almost the opposite.

However, I don't agree with the evaluation of the predictive power of trees.

At least not for normal structured data.

**For tabular data, I disagree!**

Characteristic	Neural Nets	SVM	Trees	MARS	k-NN, Kernels
Natural handling of data of “mixed” type	▼	▼	▲	▲	▼
Handling of missing values	▼	▼	▲	▲	▲
Robustness to outliers in input space	▼	▼	▲	▼	▲
Insensitive to monotone transformations of inputs	▼	▼	▲	▼	▼
Computational scalability (large $N$ )	▼	▼	▲	▲	▼
Ability to deal with irrelevant inputs	▼	▼	▲	▲	▼
Ability to extract linear combinations of features	▲	▲	▼	▼	◆
Interpretability	▼	▼	◆	▲	▼
Predictive power	▲	▲	▼	◆	▲

...and others do too [<https://arxiv.org/abs/2110.01889>]

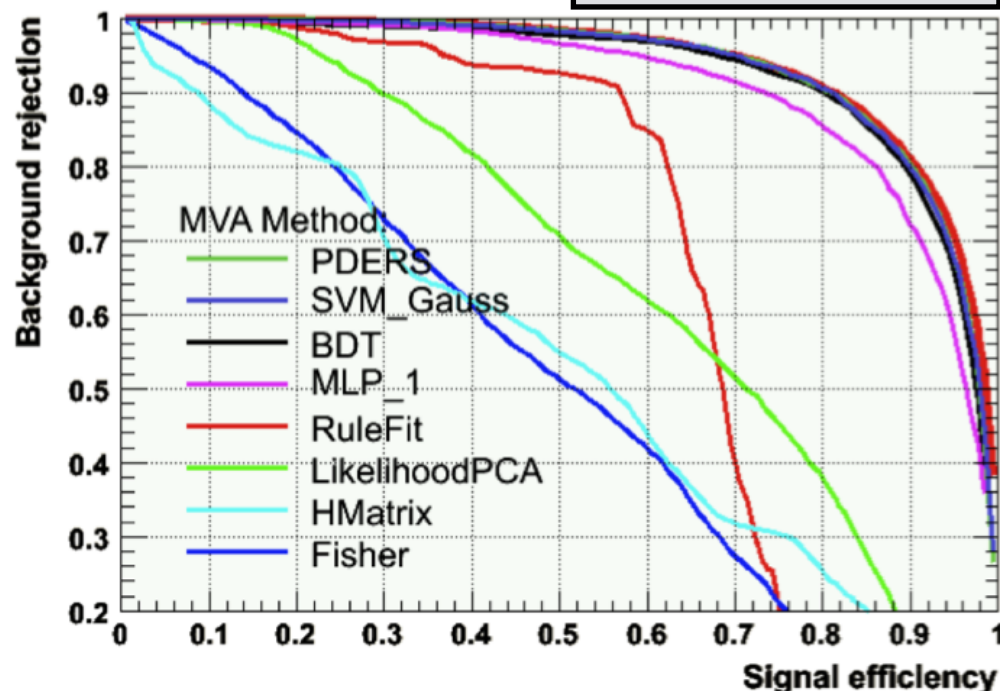
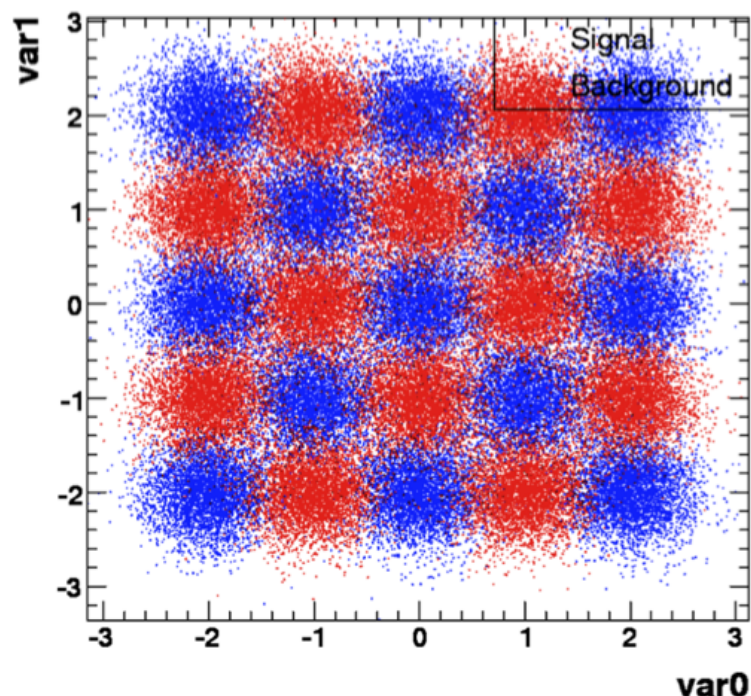
From ESL II, Chapter 10.7



# Performance comparison

Left figure shows the distribution of signal and background used for test.  
Right figure shows the resulting separation using various MVA methods.

**ROC curves:**



The theoretical limit is known from the Neyman-Pearson lemma using the (known/correct) PDFs in a likelihood.

In all fairness, this is a case that is great for the BDT...

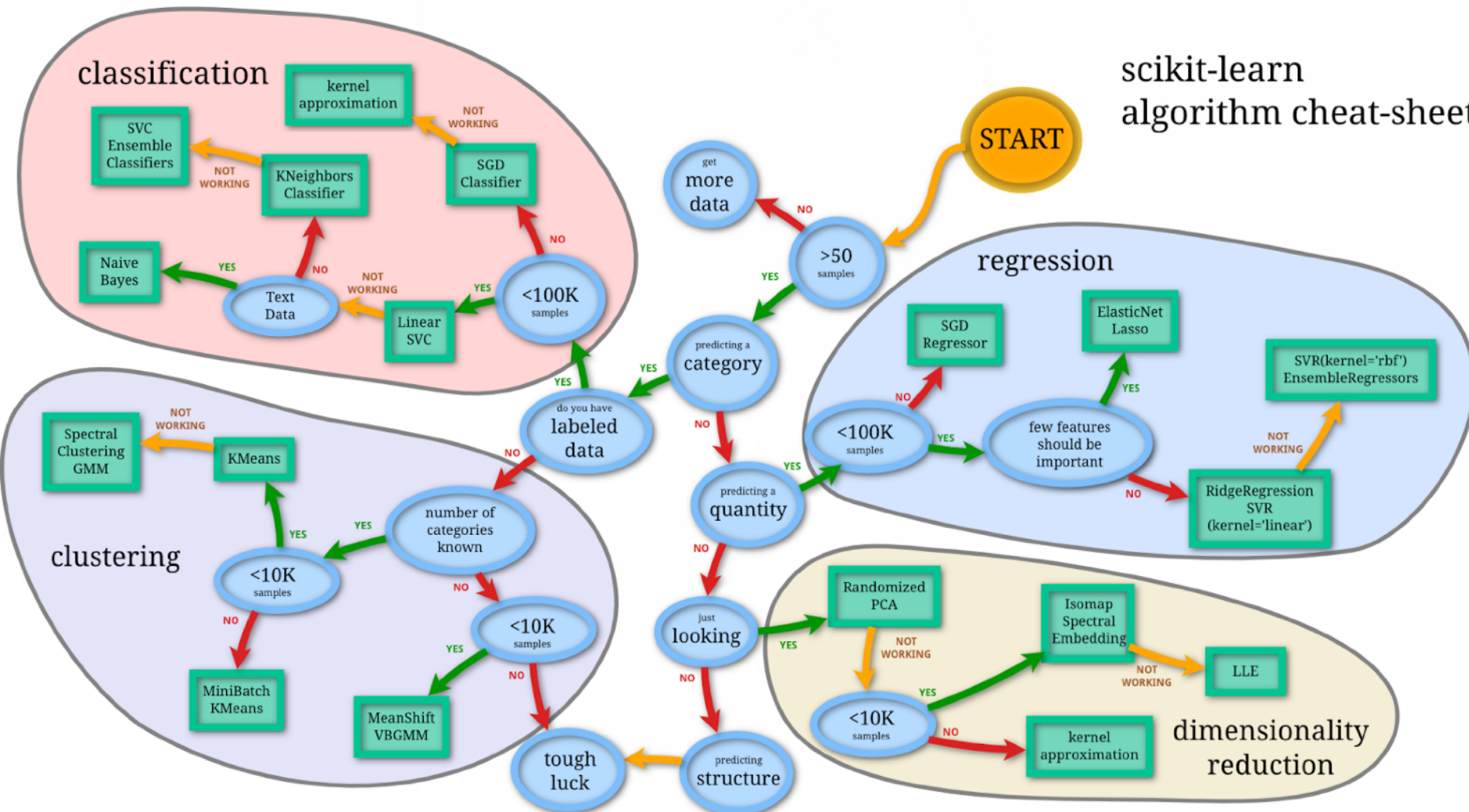


# How to choose method?

# Which method to use?

There is no good / simple answer to this, though people have tried, e.g.:

scikit-learn  
algorithm cheat-sheet

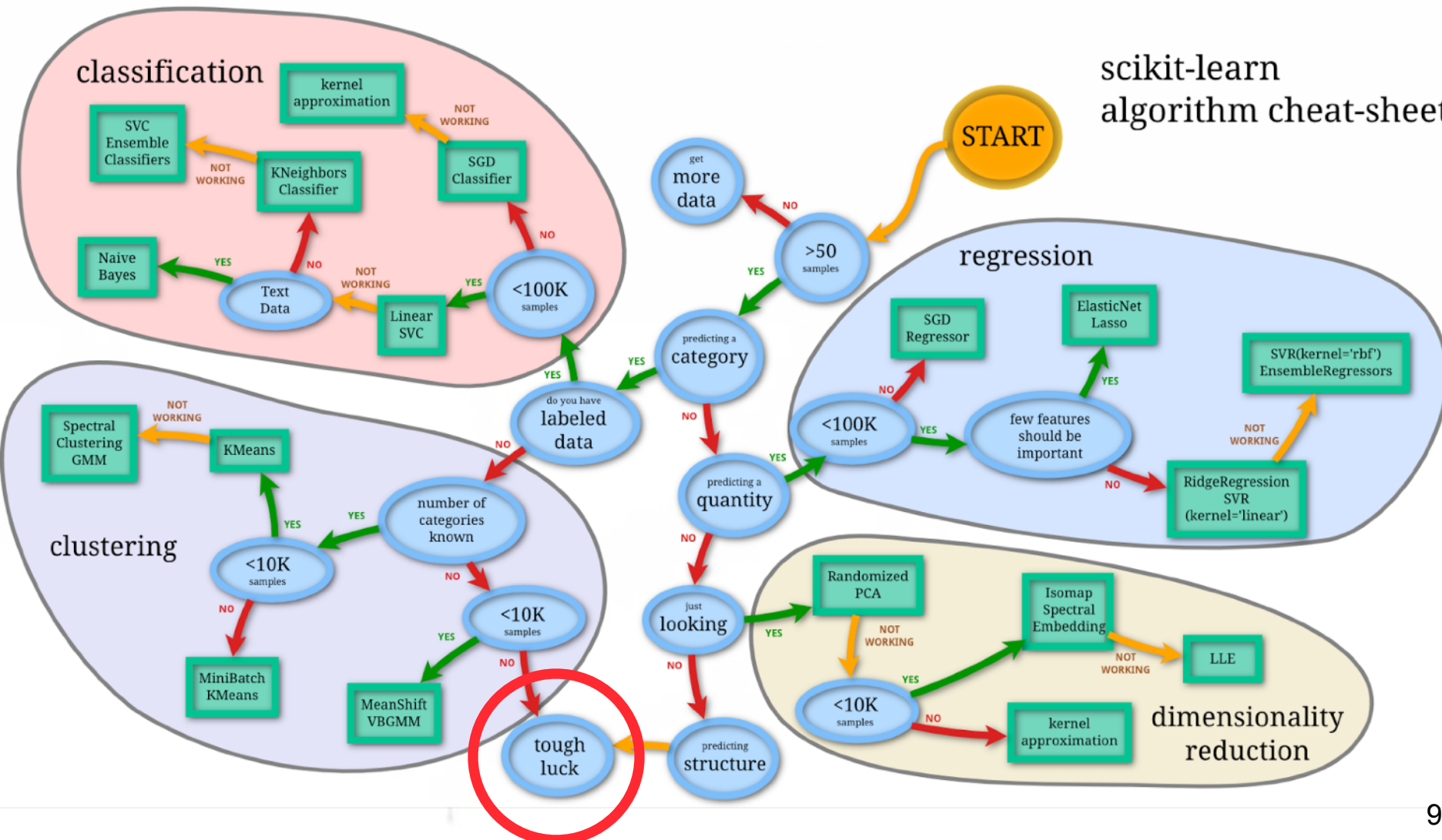




# Which method to use?

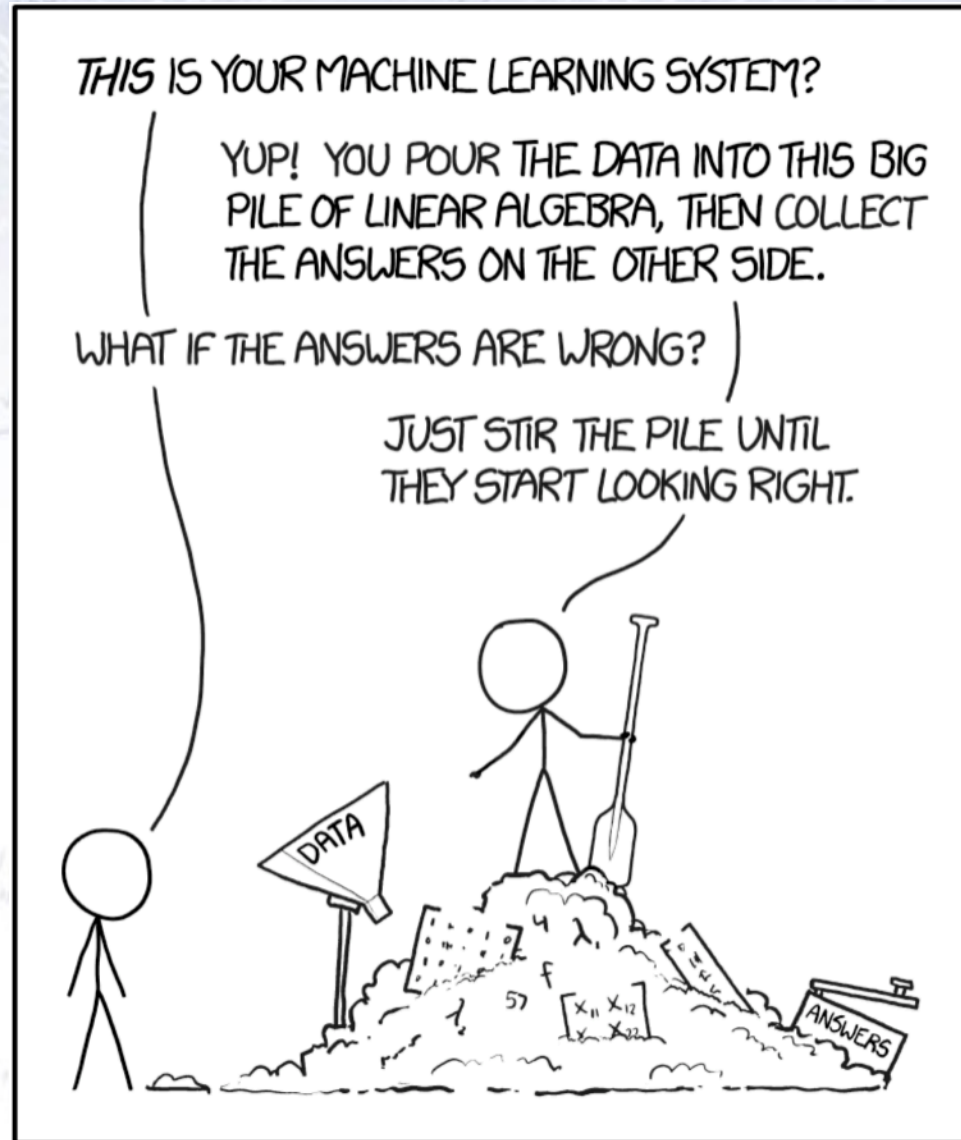
There is no good / simple answer to this, though people have tried, e.g.:

scikit-learn  
algorithm cheat-sheet





# What is Machine Learning?



“I keep saying the sexy job in the next ten years will be statisticians.”

[Hal Varian in 2009, Chief economist of Google, Berkeley professor]



“I keep saying the sexy job in the next ten years will be statisticians.”

[Hal Varian in 2009, Chief economist of Google, Berkeley professor]



Well, Hal, what is the sexy job in this decade?  
Machine Learning expert/data scientist?!?