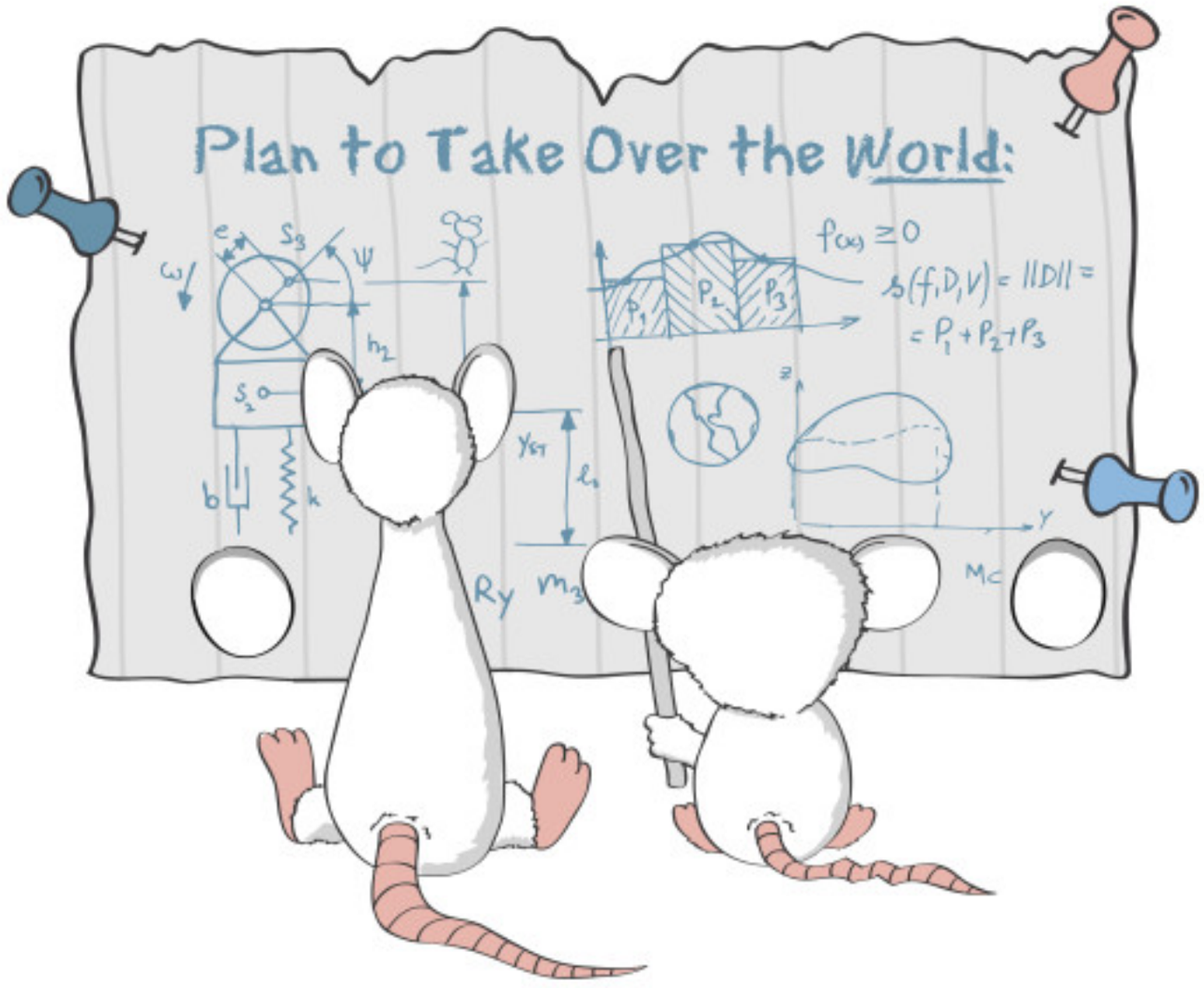


Lecture #3

F5611 Machine Learning for Astronomers
by Martin Topinka

https://github.com/toastmaker/f5611-ML4A

- 24 Nov 2020 Antonio D'Isanto (MPI, Heidelberg) on redshift estimation
- 15 Dec 2020 Ashish Mahabal (Caltech) on ZTF transient classification

Lecture by ***Antonio D'Isanto*** (MPI, Heidelberg)

24 Nov 2020

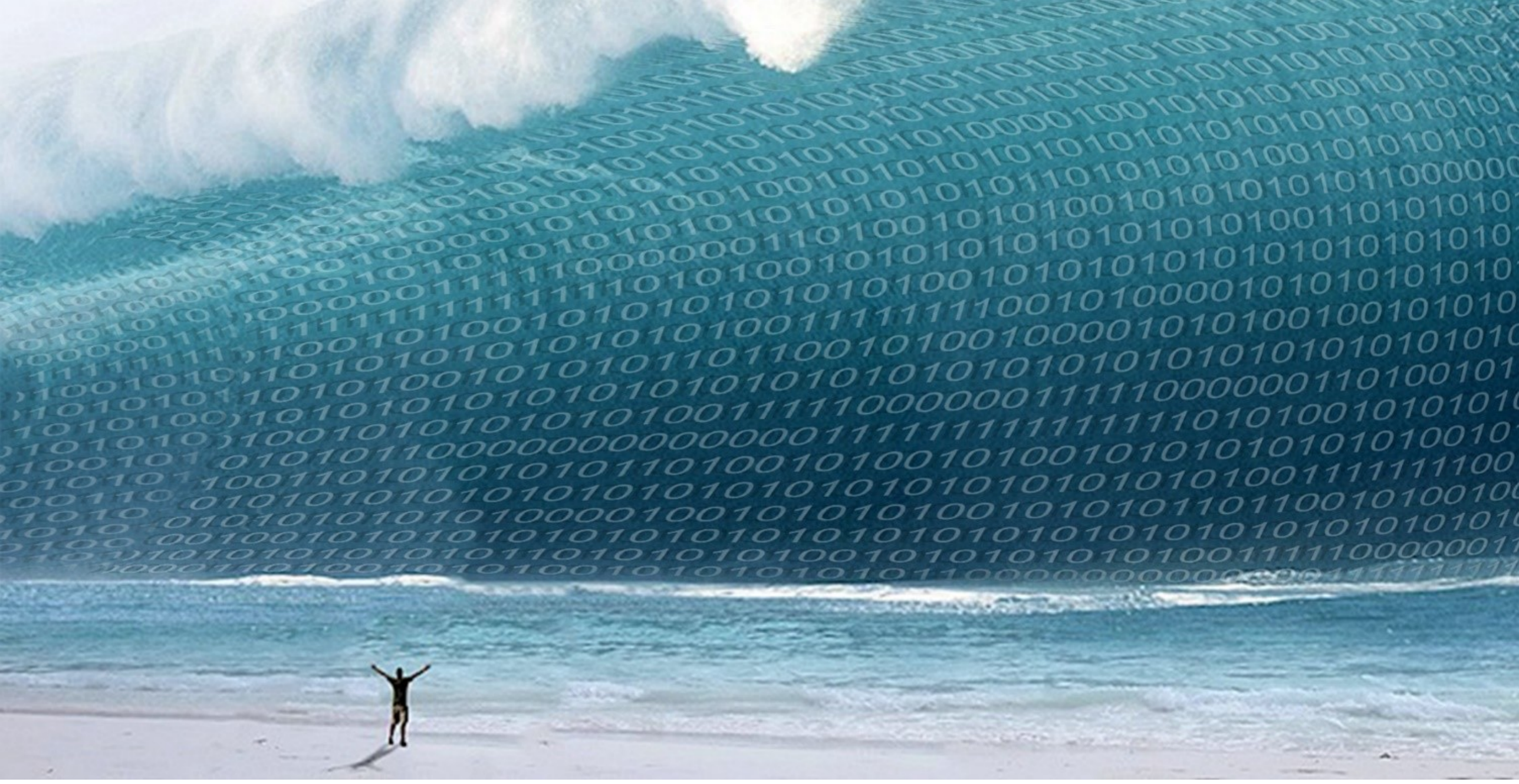
The two worlds of photometric redshift estimation via machine learning: fully automatic vs feature based

The problem of photometric redshift estimation is a major subject in astronomy, since the need of estimating distances for a huge number of sources, as required by the data deluge of the recent years. The ability to estimate redshifts through spectroscopy does not scale with this avalanche of data and photometric redshifts provide the required redshift estimates at the cost of some precision. The success of several forthcoming missions is highly dependent on the availability of photometric redshifts.

Machine learning provides a powerful and efficient solution to the problem of photometric redshift estimation.

In this lecture, I will present two models. The first is fully-automatised, based on the combination of a convolutional neural network with a mixture density network, to predict probabilistic multimodal redshifts directly from images. The second model is features-based, performing a massive combination of photometric parameters to apply a forward selection in a huge feature space. The proposed models perform very efficiently compared to some of the most common models used in the literature. Particular focus is dedicated to the correct estimation of the errors and prediction quality.

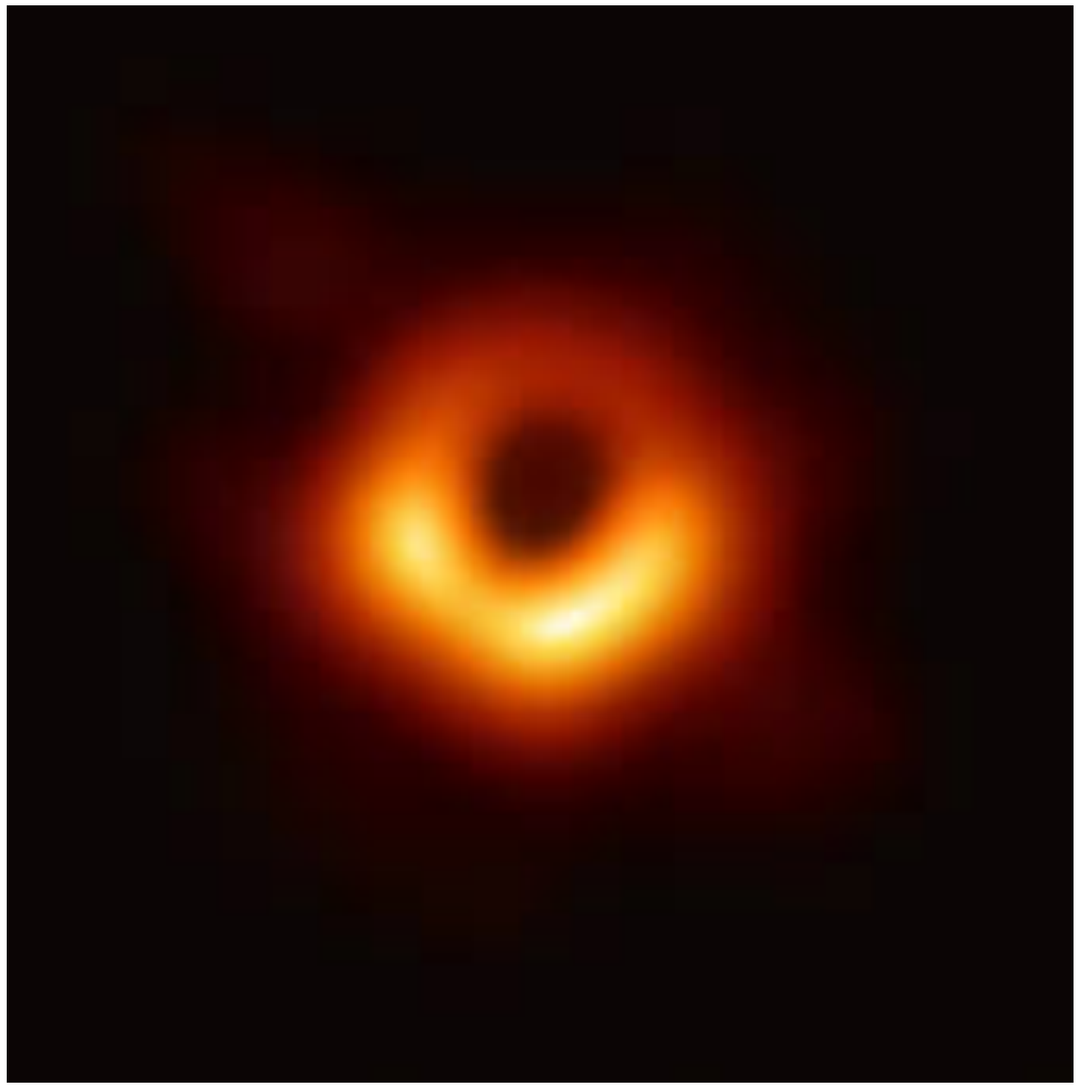
The proposed models are very general and can be applied to different topics in astronomy and beyond.



The information volumes and rates grow exponentially ⇒ Most data will never be seen by a human

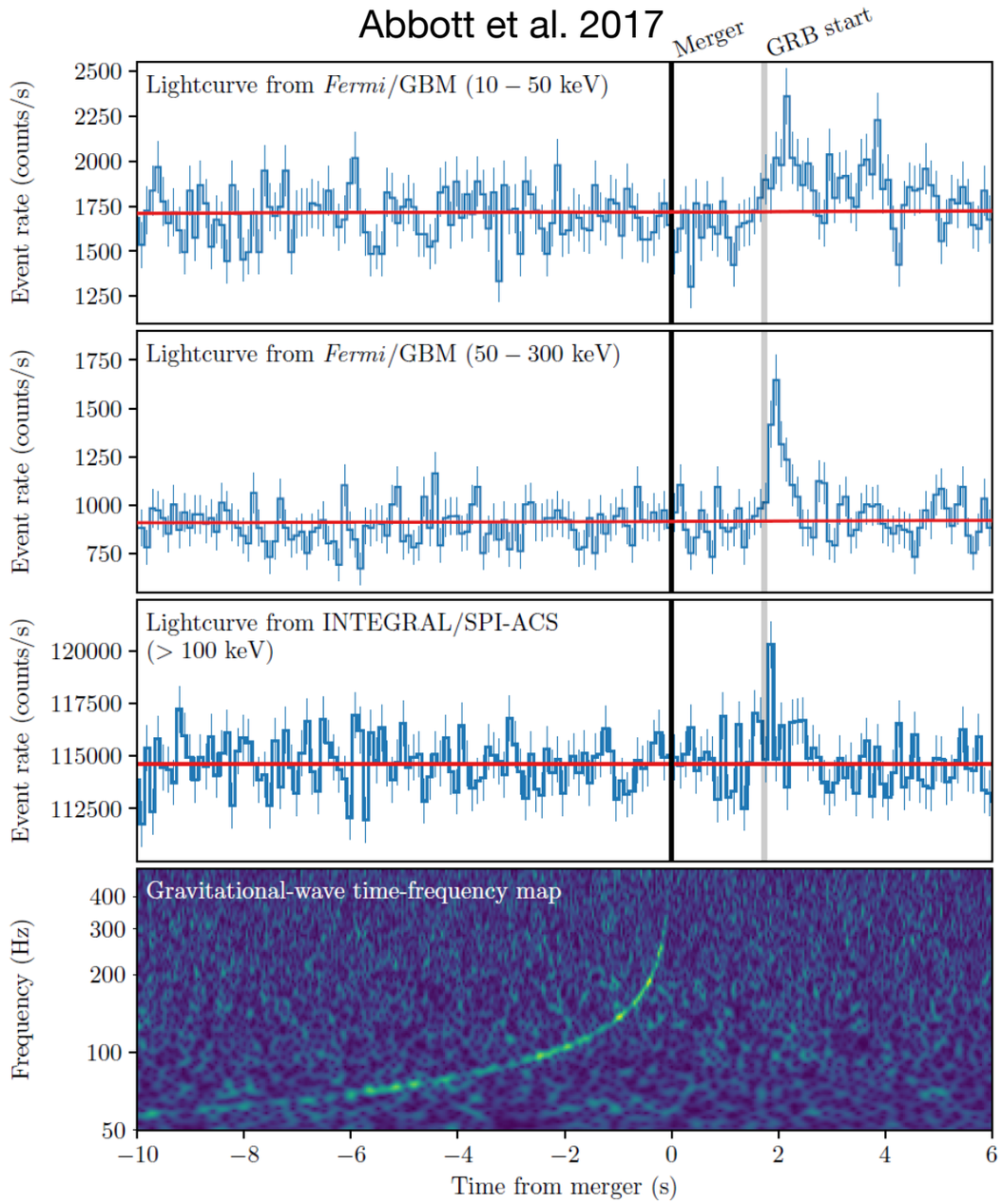
Increase in the data information content ⇒ Data-driven vs hypothesis driven science

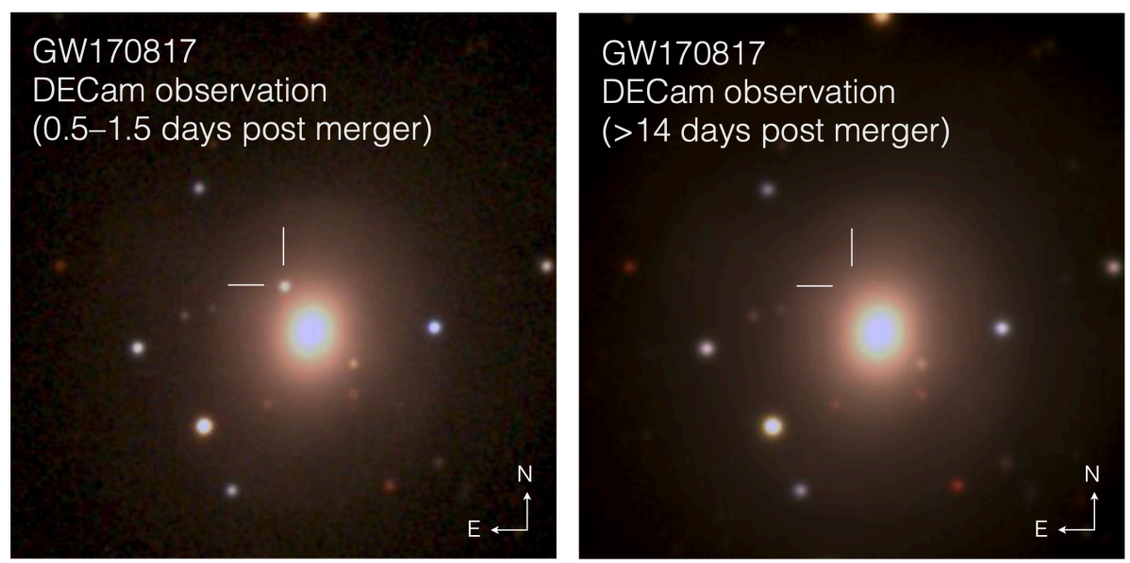
Increase in the information complexity ⇒ There are patterns in the data that cannot be comprehended by human directly



Credit: EHT Collaboration

Abbott et al. 2017



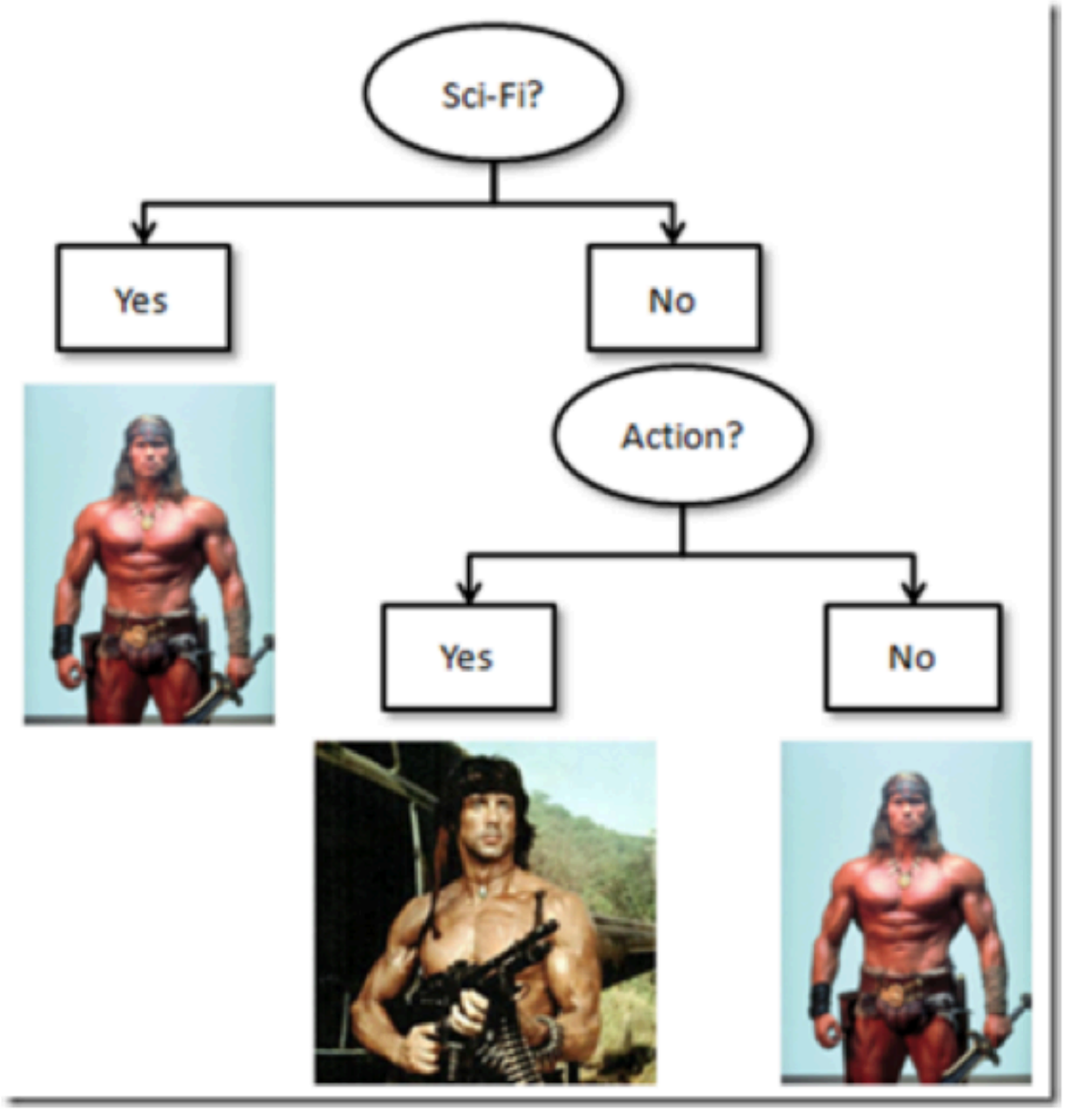


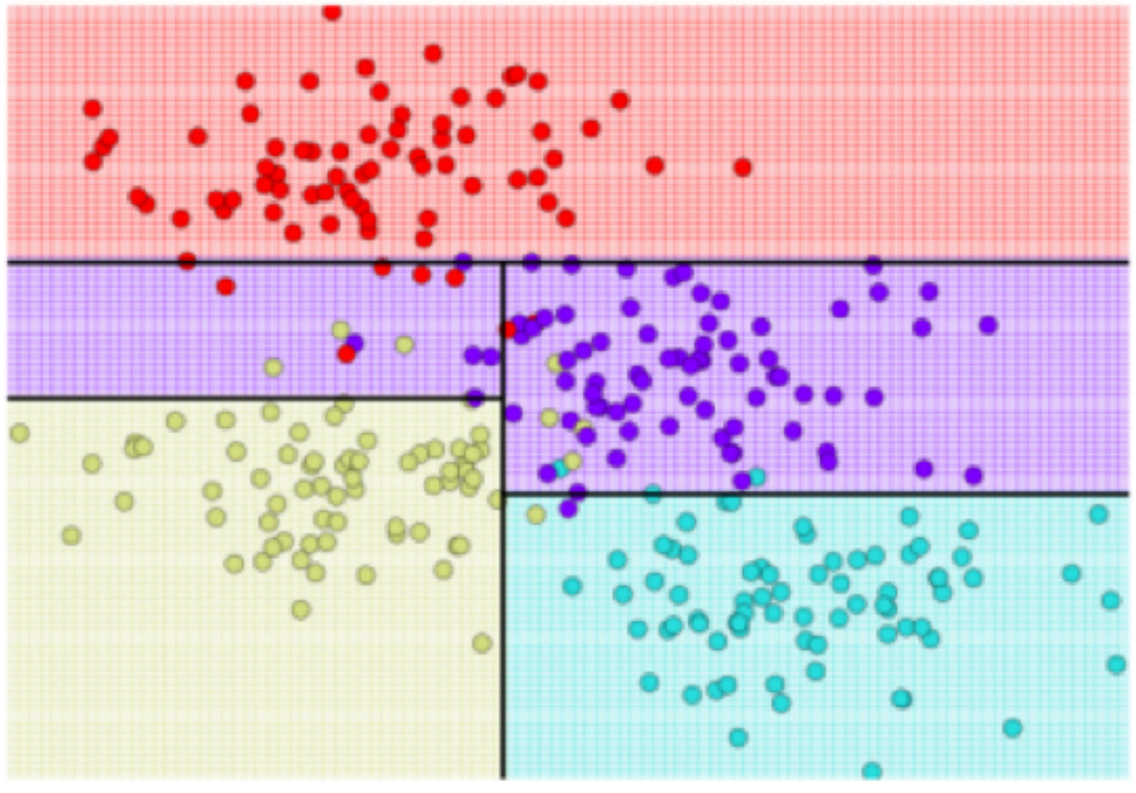
Soares-Santos et al. 2017

More advanced algorithms

- **Ensemble methods:**
    - Bagging (averaging lowers variance) "bootstrapping"
    - Voting (majority wins)
      (or train a simple linear regressor of the classifiers)
    - Boosting
- **Neural networks**

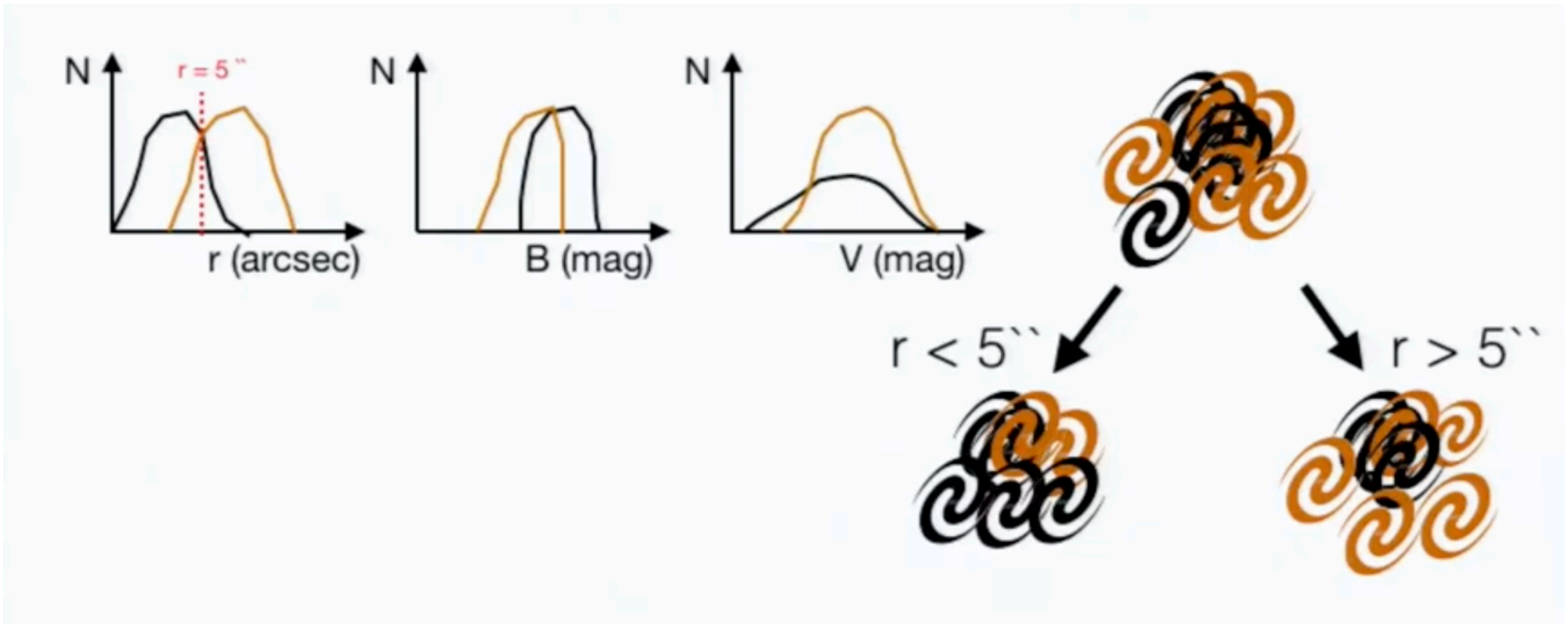
Random Forest (Ensemble of Decision Trees)





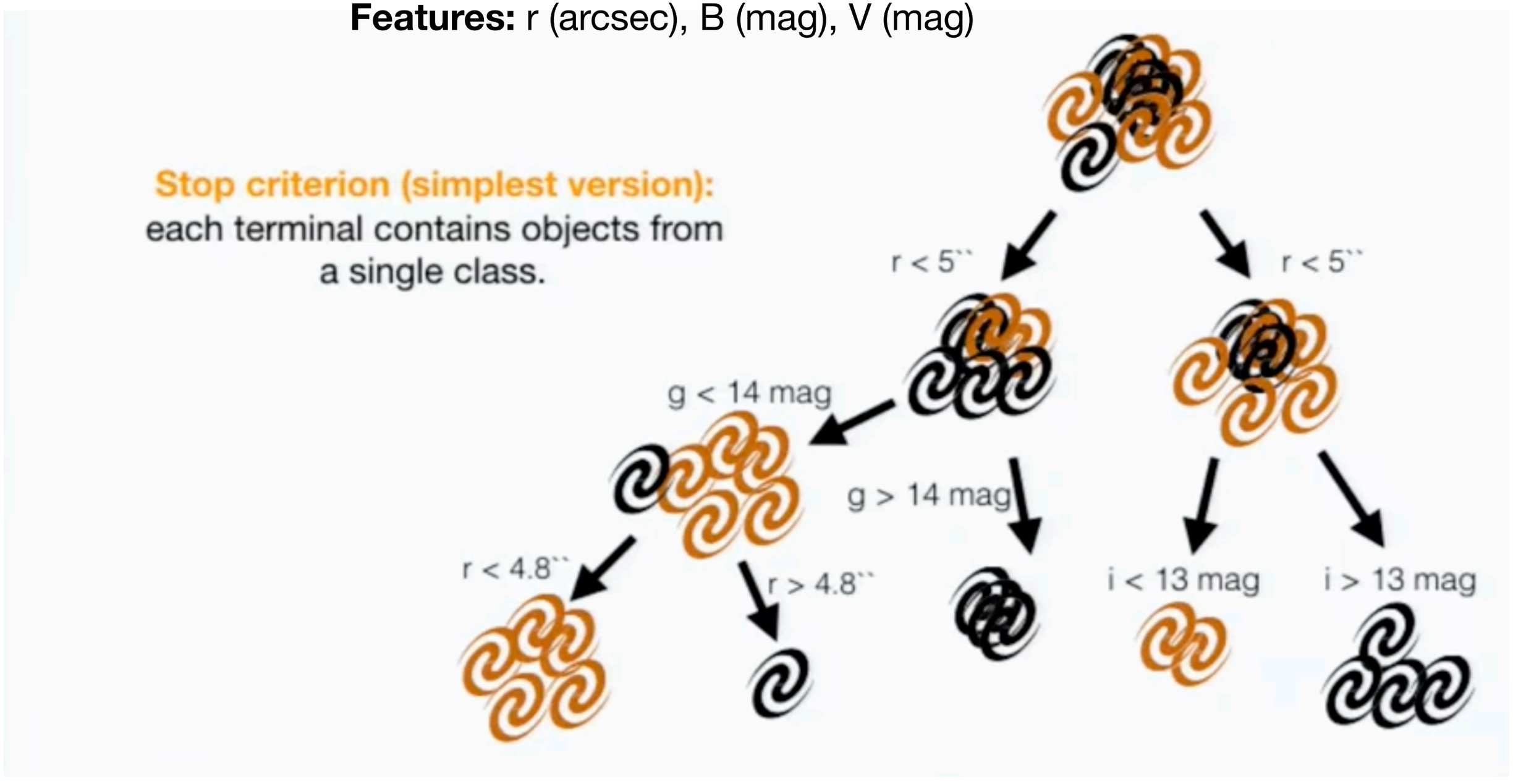
Construction of Decision Tree

Input training set: a list of galaxies
Classes: "black" galaxies and "brown" galaxies
Features: r (arcsec), B (mag), V (mag)



Construction of Decision Tree

Input training set: a list of galaxies
Classes: "black" galaxies and "brown" galaxies
Features: r (arcsec), B (mag), V (mag)

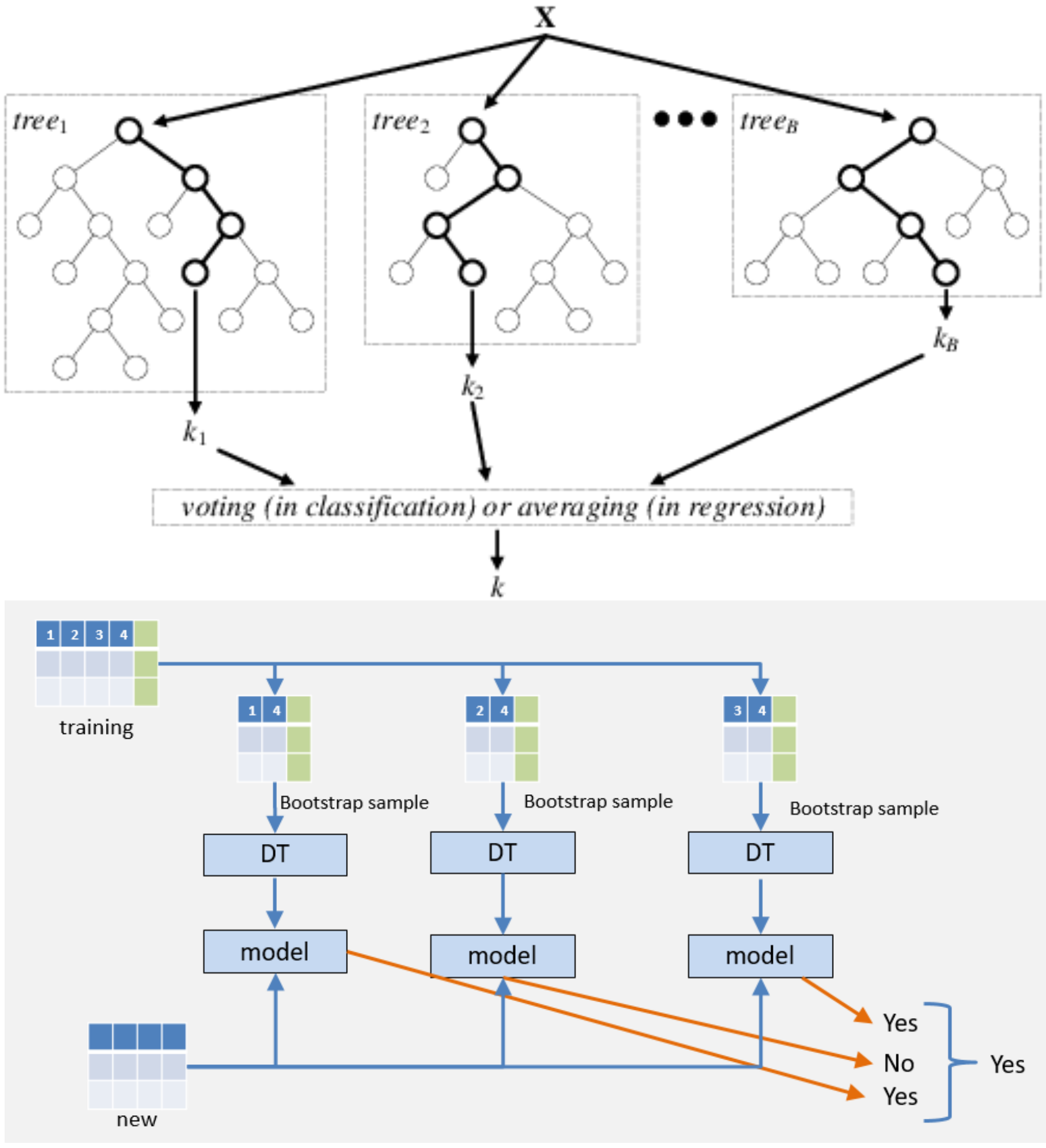


Random Forest

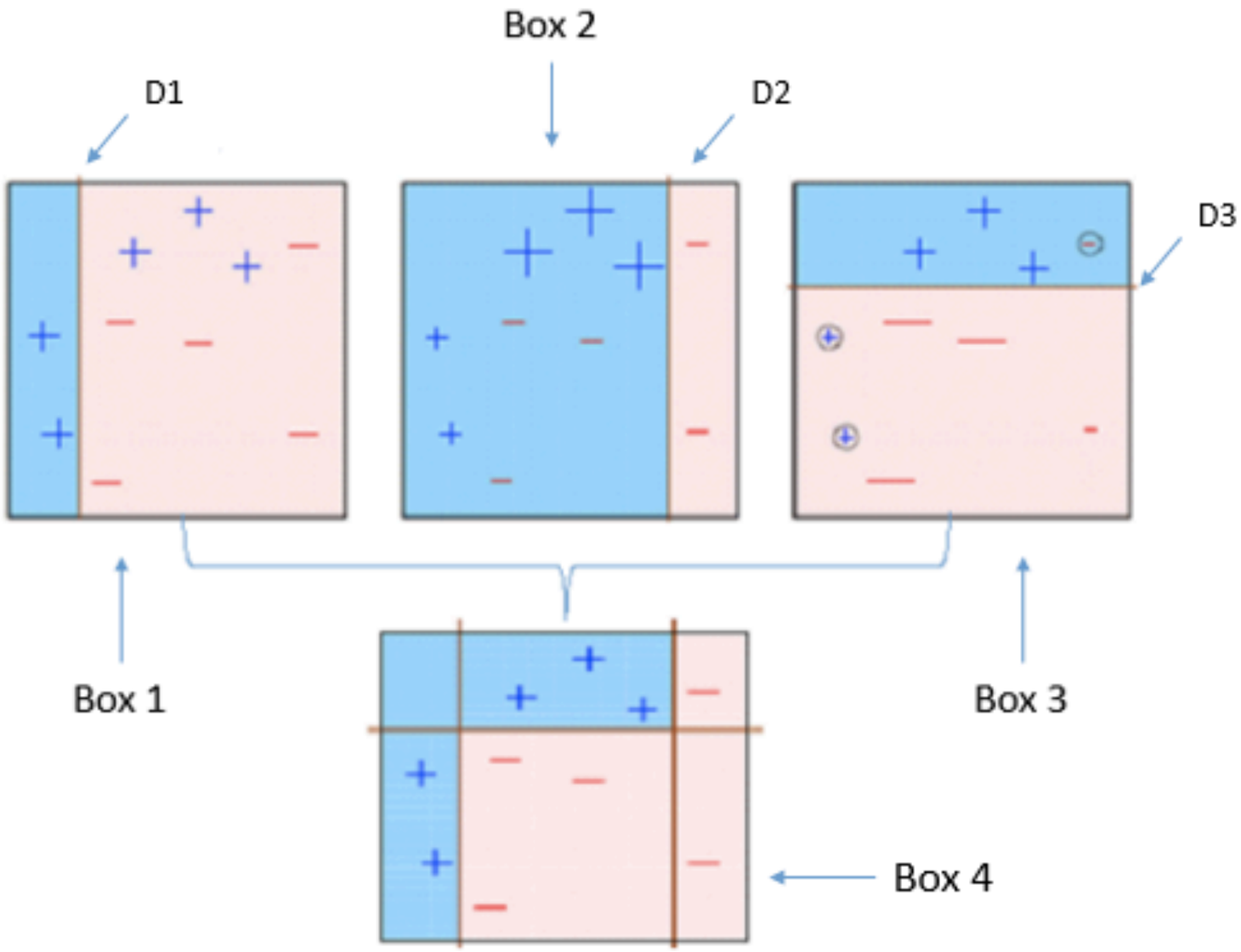
=

collection of randomised trees

(random data subsample, random feature)



Voting (max/average) with weights (gradient descent)

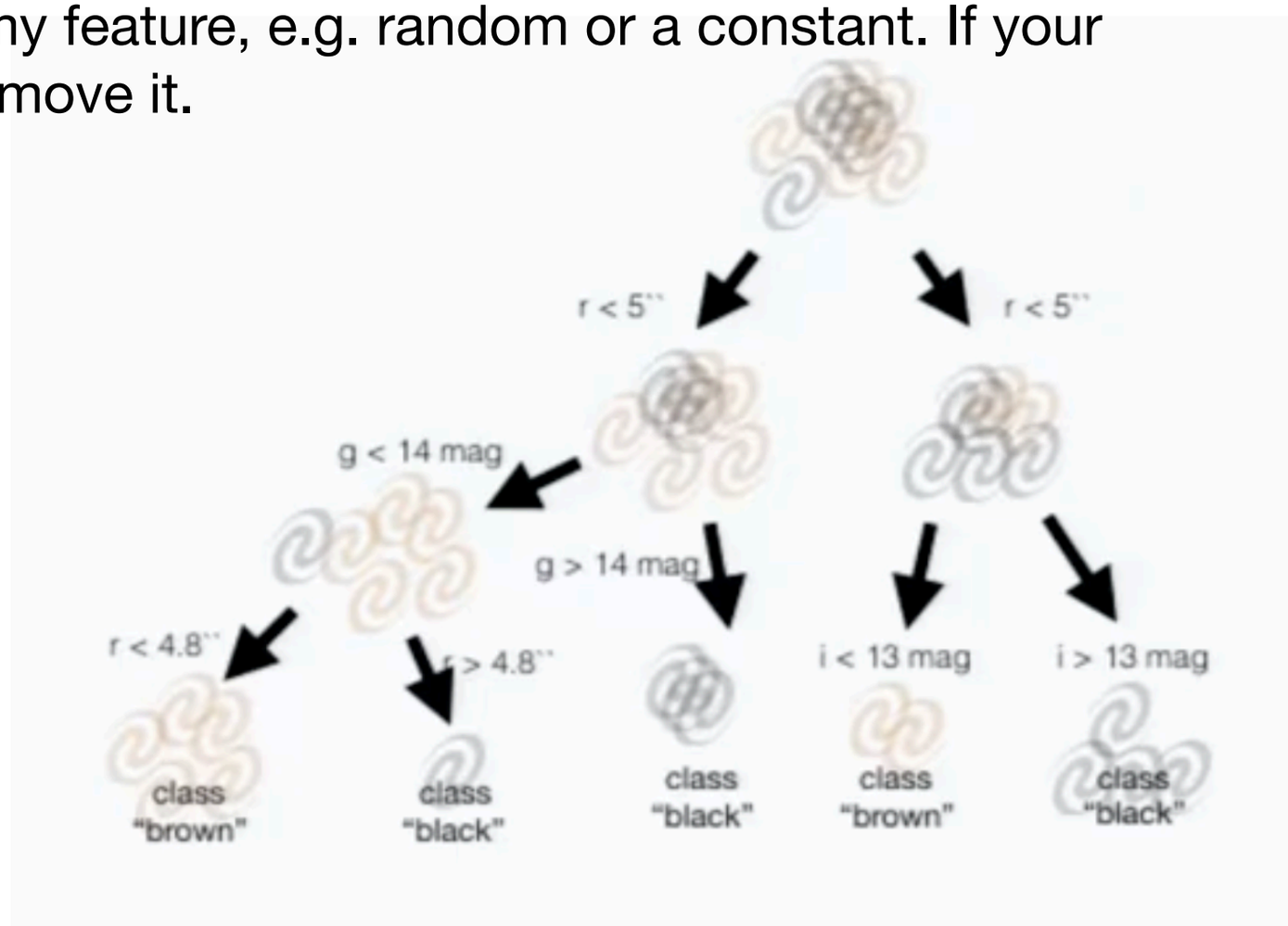


Feature importance & feature selection

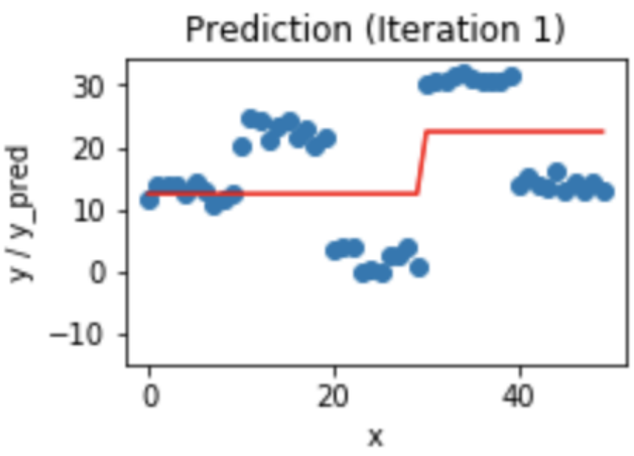
DT: The higher a feature is in the tree, the more important it is for classification.

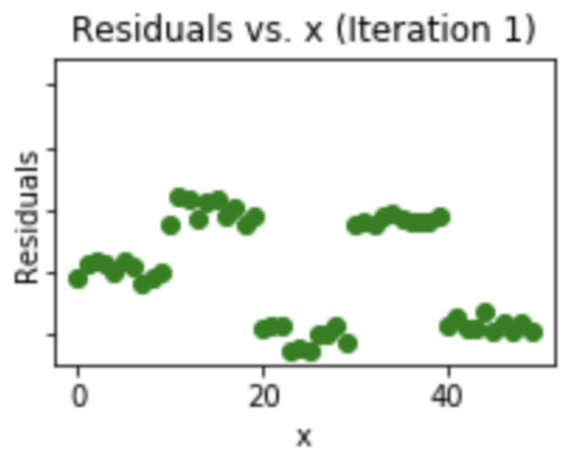
RF: Omitting a feature makes separates data faster, feature has low importance and vice versa.

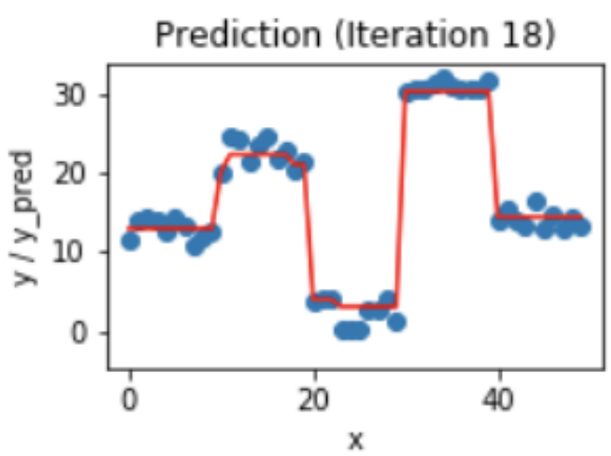
Trick: add a non-informative dummy feature, e.g. random or a constant. If your physical feature is ranked lower, remove it.

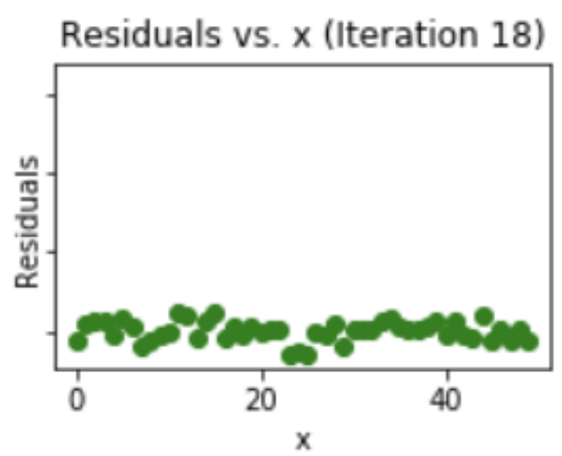


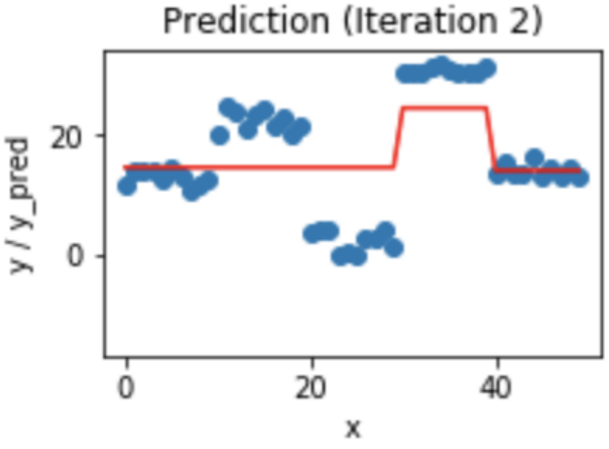
Gradient Boosted Trees

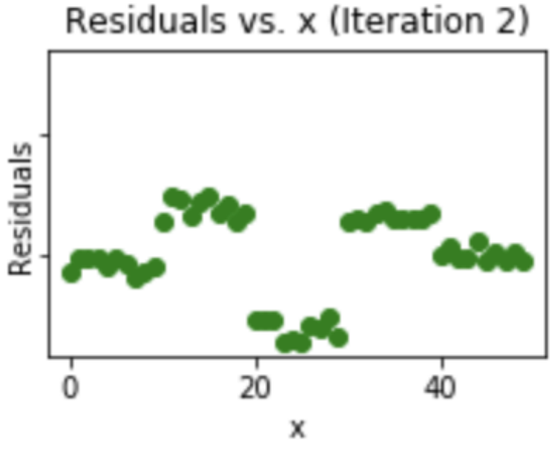


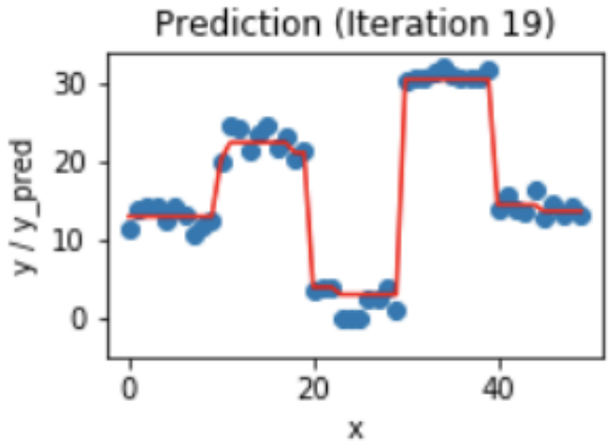


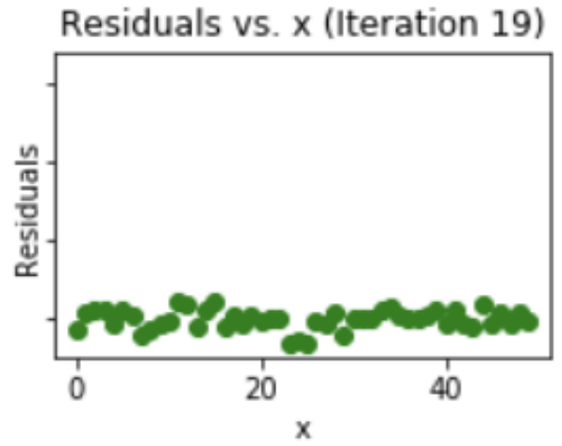


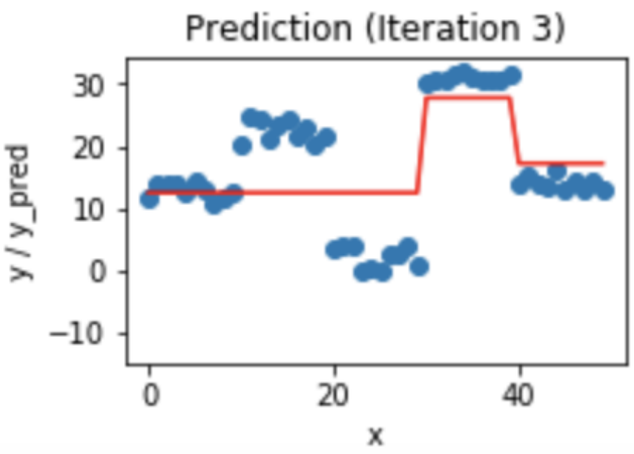


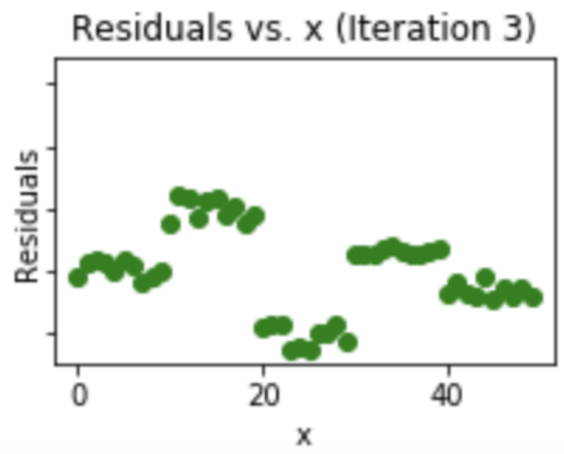


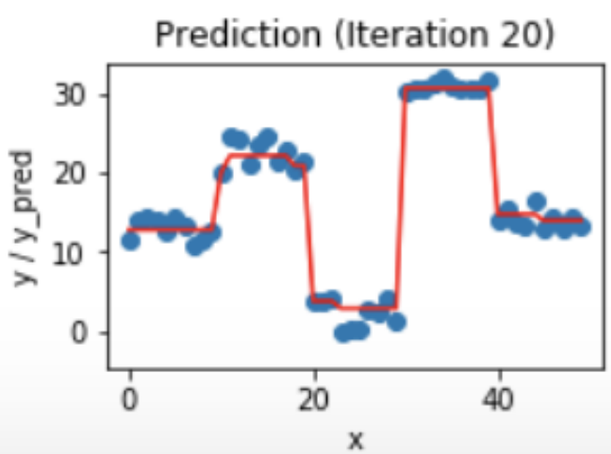


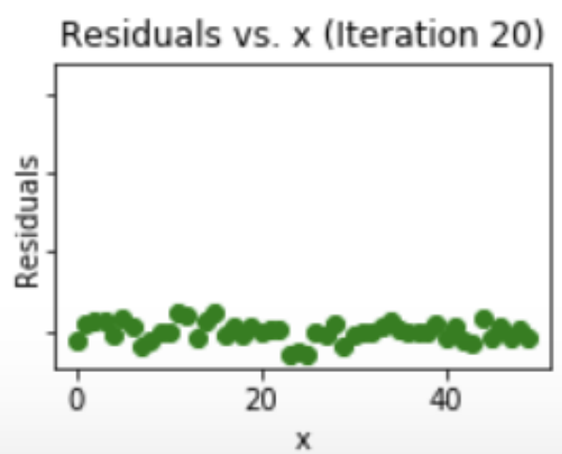












Decision Tree

Random Forest

AdaBoost

.75

.82

.85

Decision Tree

Random Forest

AdaBoost

.82

.85

.85

Decision Tree

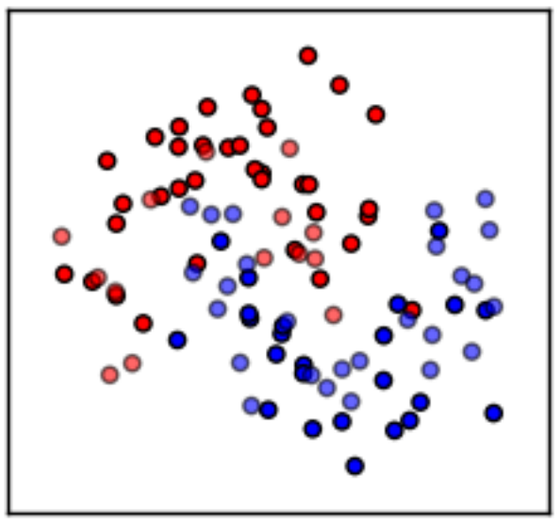
Random Forest

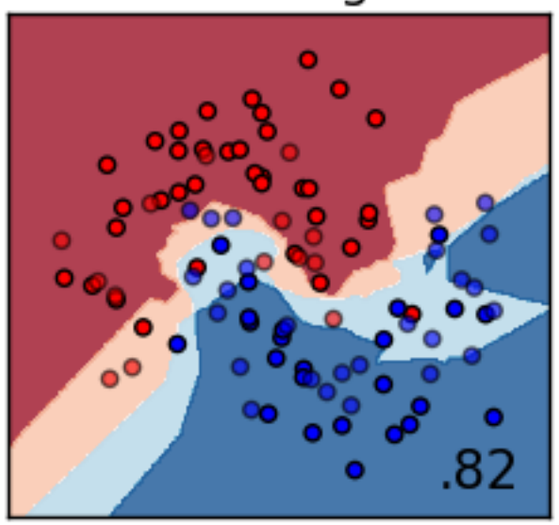
AdaBoost

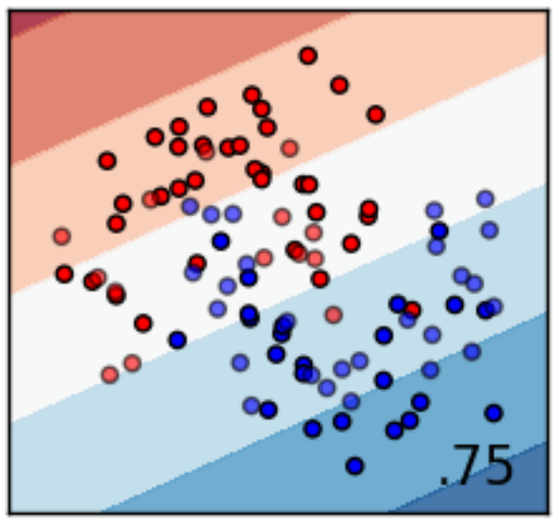
.88

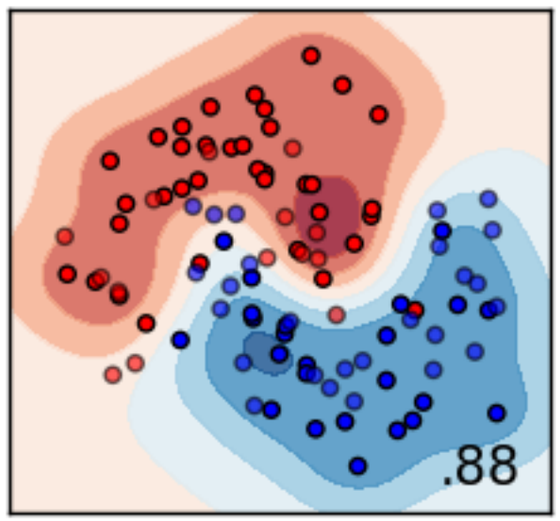
.88

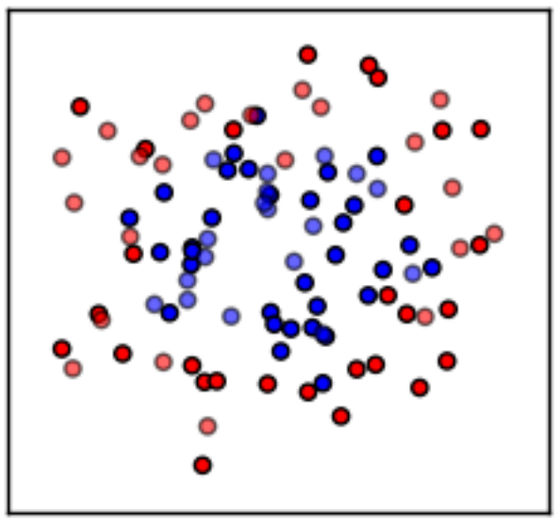
.90

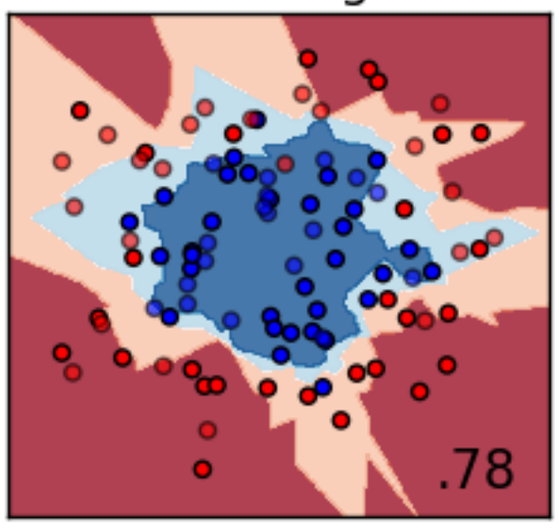


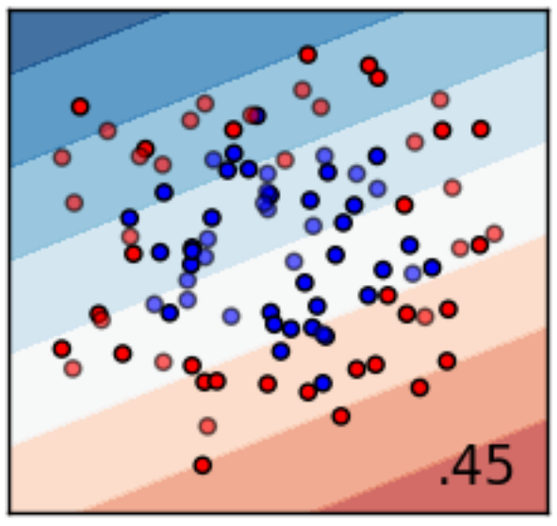

Nearest Neighbors
.82

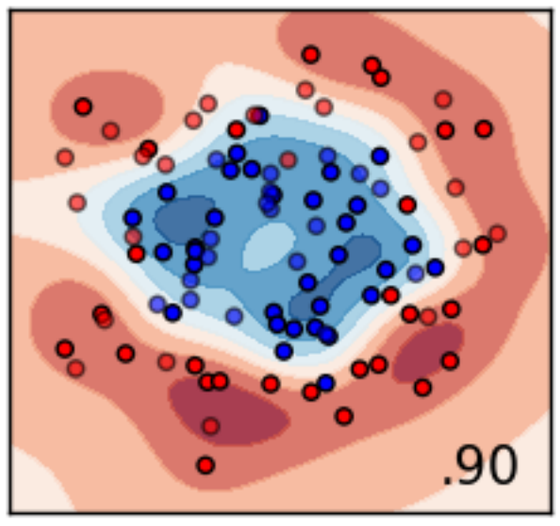

Linear SVM
.75

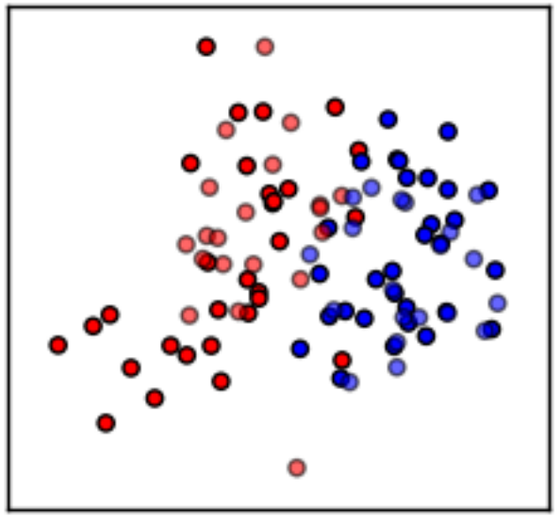

RBF SVM
.88

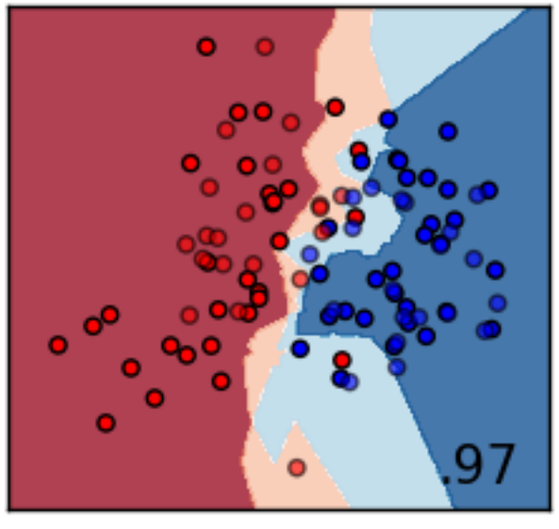


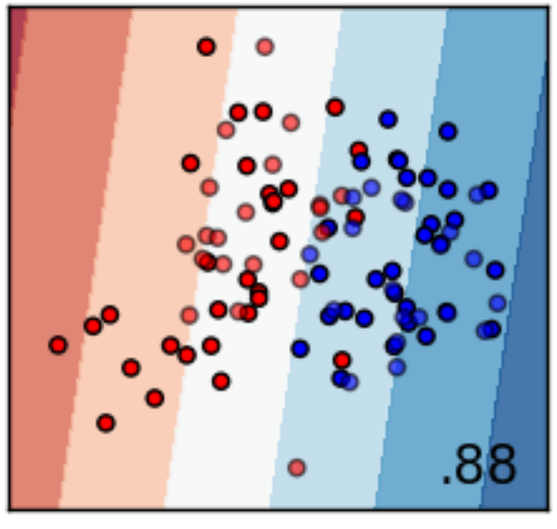

Nearest Neighbors
.78

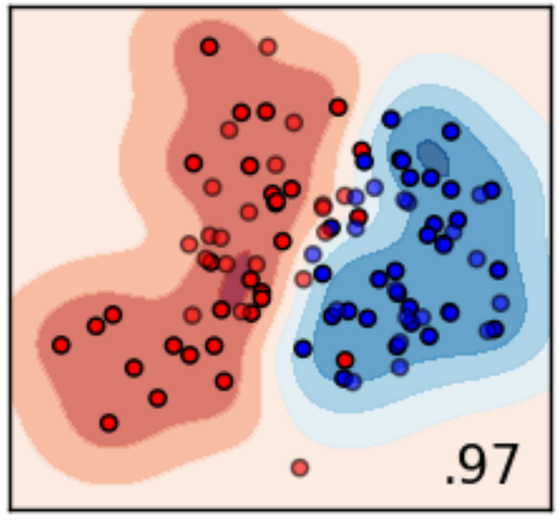

Linear SVM
.45


RBF SVM
.90




Nearest Neighbors
.97

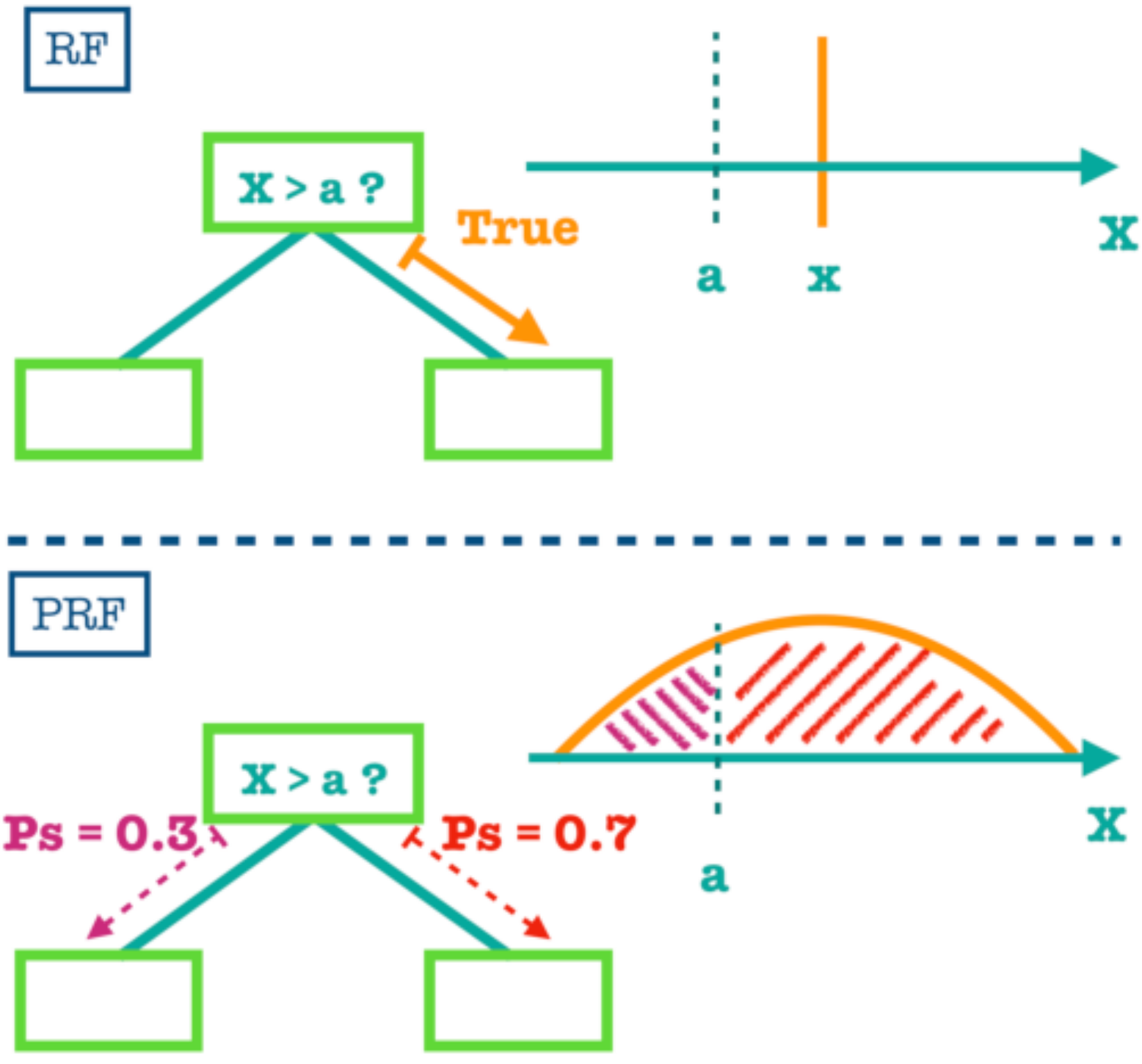

Linear SVM
.88


RBF SVM
.97

Top 5 (supervised)

Algorithm	Comments
Neural Networks	• Take long to train - lot of CPU • Overfits • Requires lot of data
Gradient Boosted Trees	• Fast • Overfit danger
Random Forest	• Robust to overfitting
SVM w/non-linear kernel	• Pretty good
Gaussian Processes	• non-parametric fitting

Probabilistic Random Forest

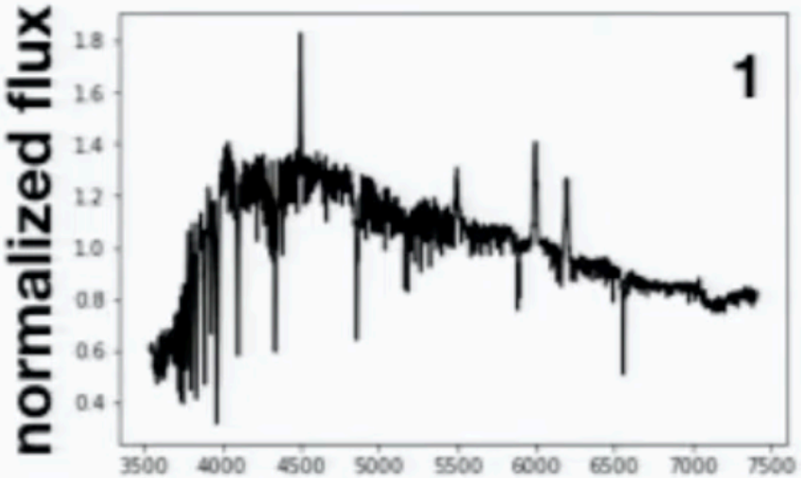


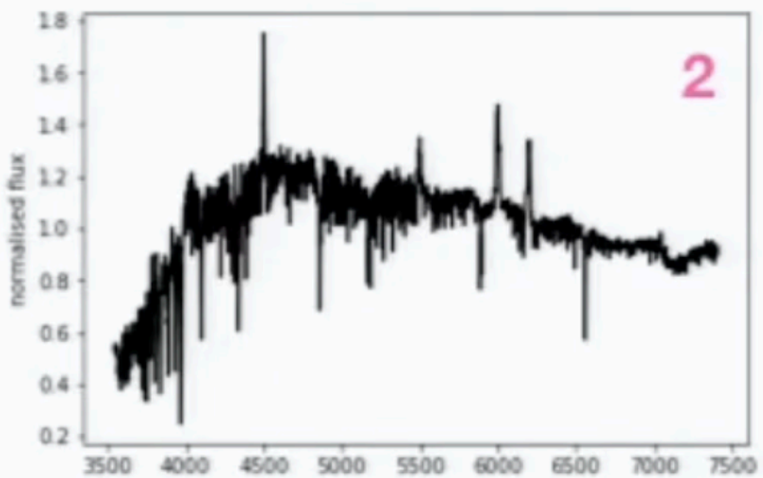
Reis et al. 2018

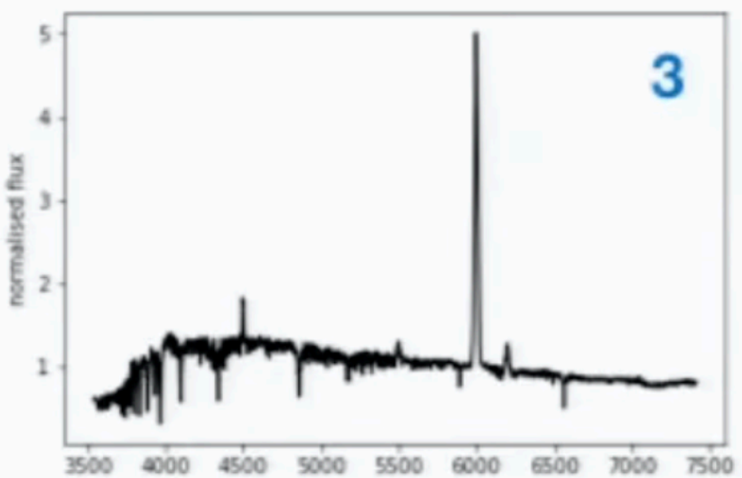
Unsupervised Random Forest

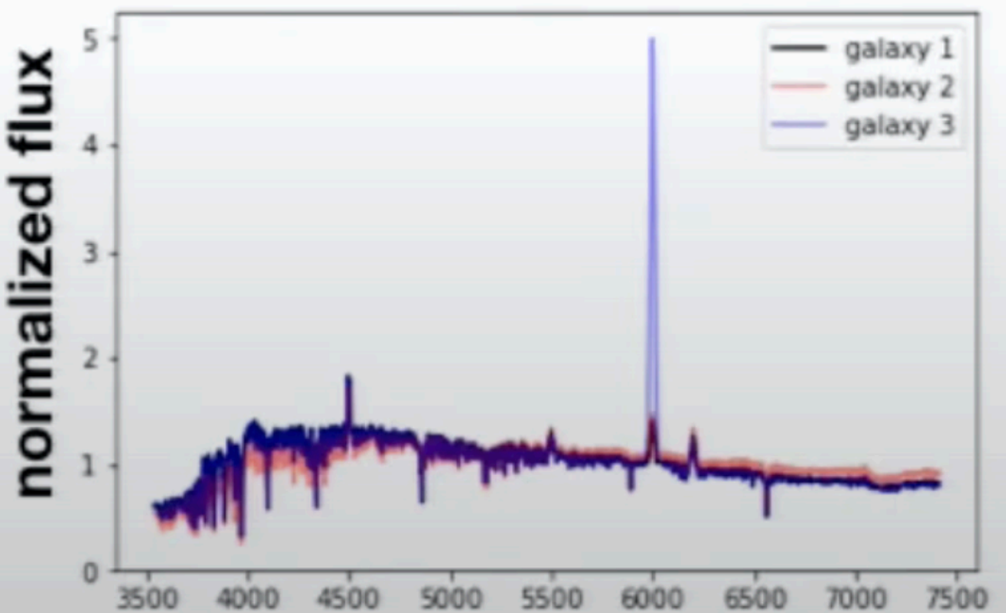
Random Forest can be used as an unsupervised algorithm, to produce pair-wise similarity for the objects in our sample.

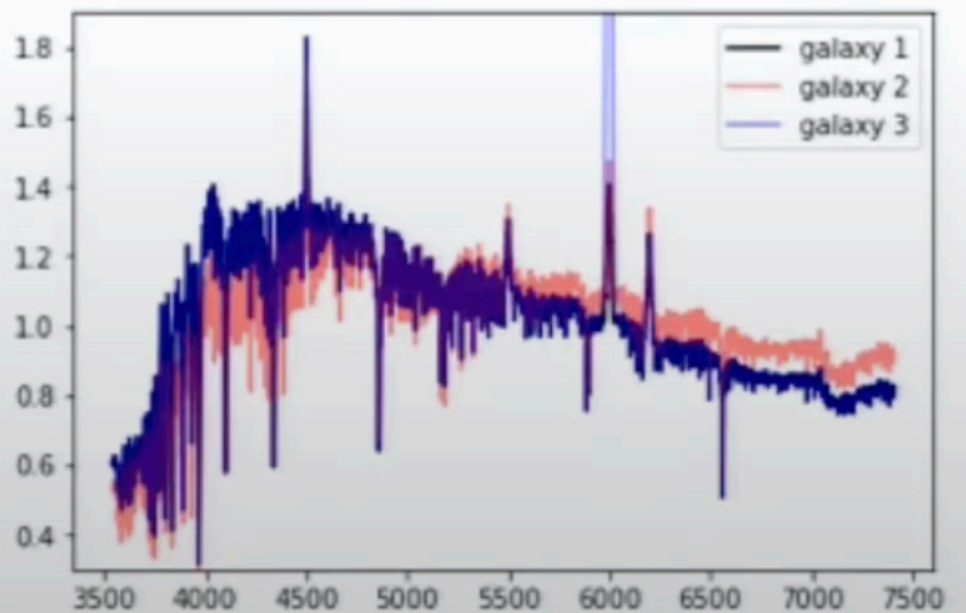
Why do we need to measure distances between objects?









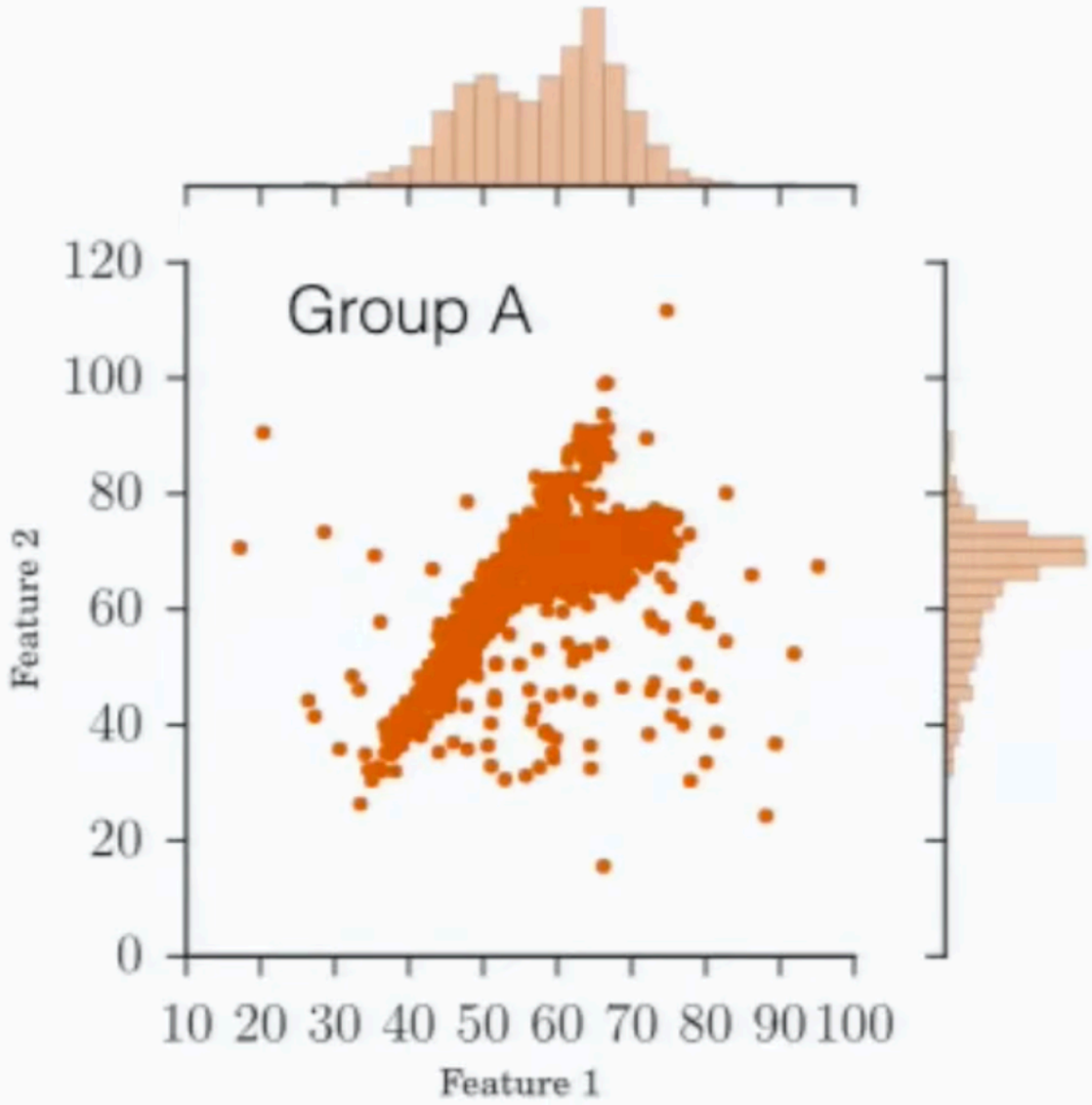


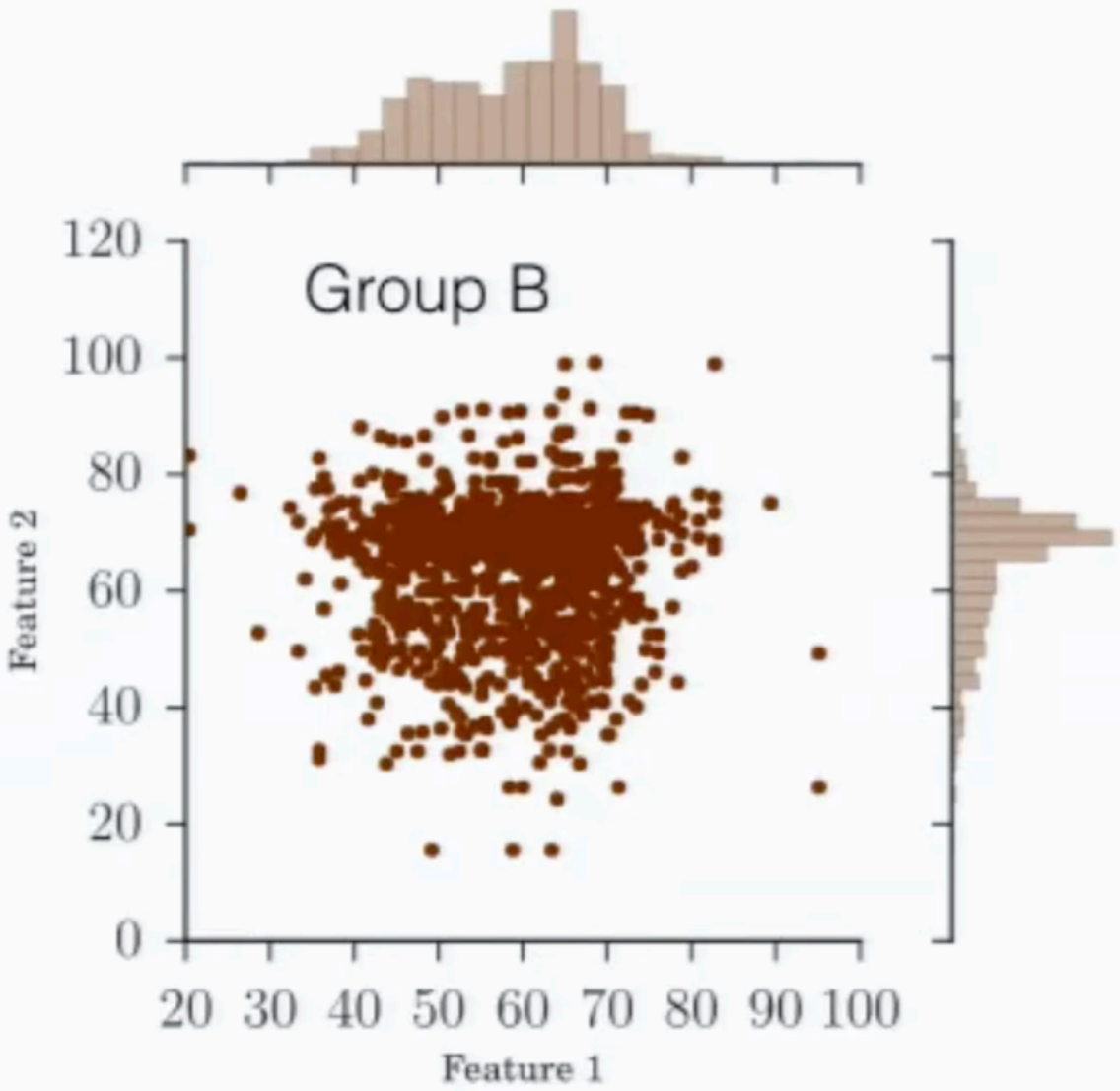
Unsupervised Random Forest

Random Forest can be used as an unsupervised algorithm, to produce pair-wise similarity for the objects in our sample.

Input dataset: a list of objects with measured features, but no labels!

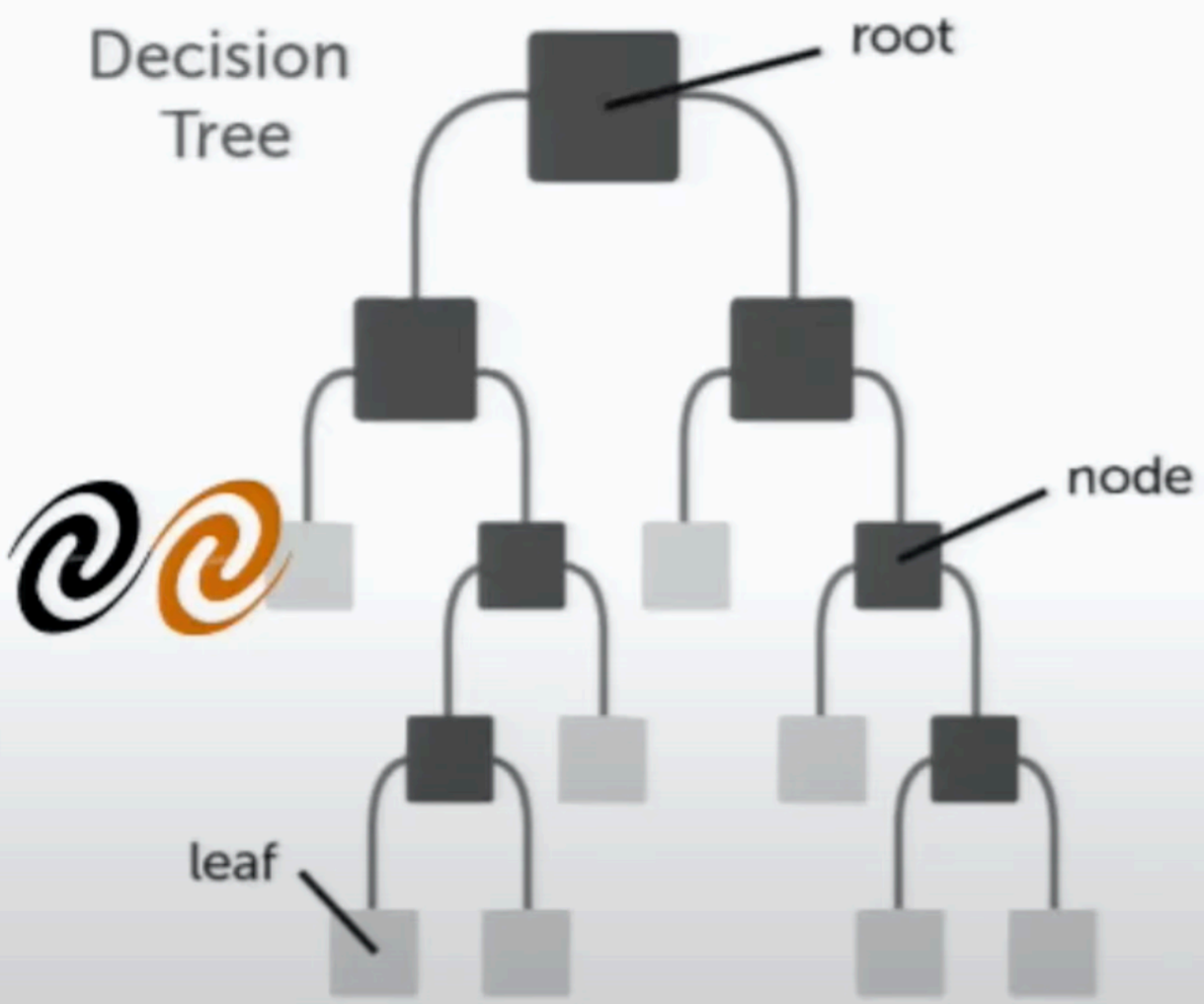
Random Forest is trained to distinguish between real and synthetic datasets.

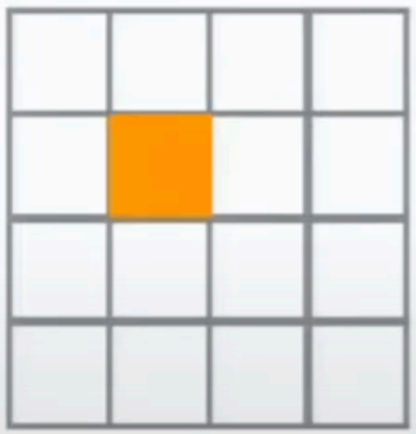




Unsupervised Random Forest

We train the Random Forest to distinguish between groups A and B.
For group A (real data), we propagate the objects and obtain a similarity matrix.



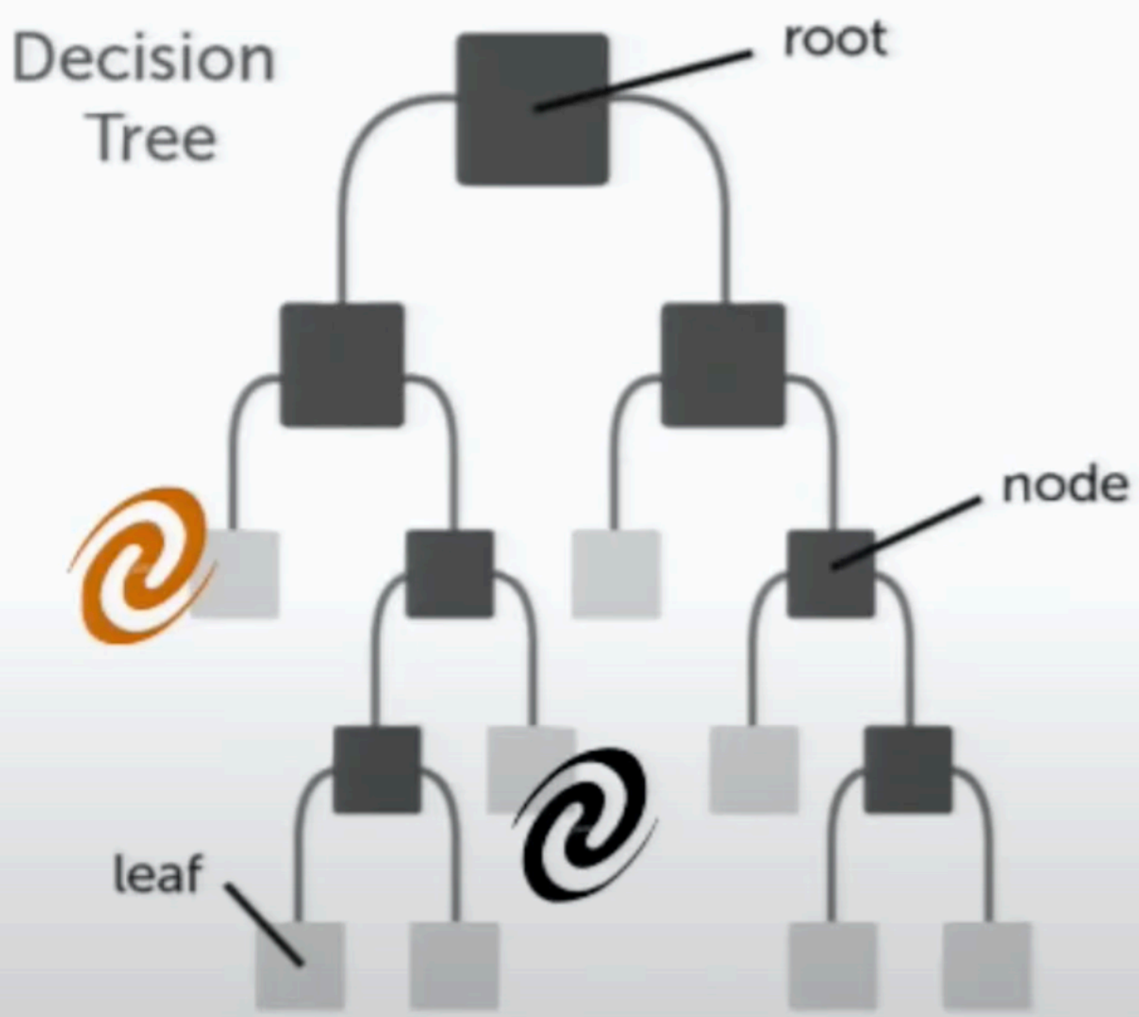


similarity matrix

similarity += 1

Unsupervised Random Forest

We train the Random Forest to distinguish between groups A and B.
For group A (real data), we propagate the objects and obtain a similarity matrix.



similarity matrix

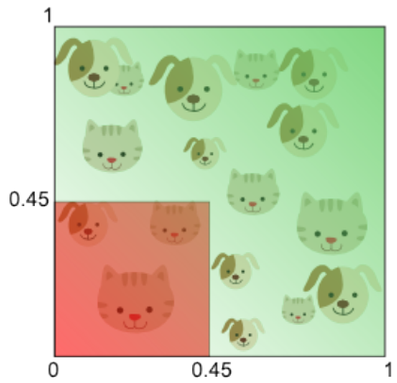
similarity += 0

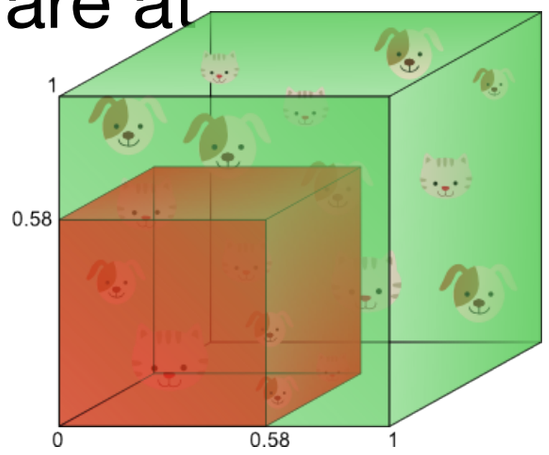
Curse of Dimensionality

- Too many features
  - Expensive to store
  - Slowing down computation
  - Subject to *Dimensionality curse*
- Sample space gets harder and harder to fill as dimensions grow
- A reason why too many features lead to overfitting as data become **sparse**

- *“If people can see in multi-dimensions we would not need machine learning”*
- More and more data needed to fill the same % of space
- Distance measure degenerates, most of the points are at the surface of a sphere (distance r)

$$\frac{V_{\text{hypersphere}}}{V_{\text{hypercube}}} = \frac{\pi^{d/2}}{d2^{d-1}\Gamma(d/2)} \rightarrow 0 \text{ as } d \rightarrow \infty$$





How Many Shades of Gray Can you Distinguish?

Answer…

Value encodes continuous variables (less well)

How Many Colors?

Hue encodes nominal variables

Answer…

How Many Shades of Gray Can you Distinguish?

Value easily encodes ordinal variables

Value encodes continuous variables (less well)

How Many Colors?

Hue encodes nominal variables

Answer…

How Many Shades of Gray Can you Distinguish?

Value easily encodes ordinal variables

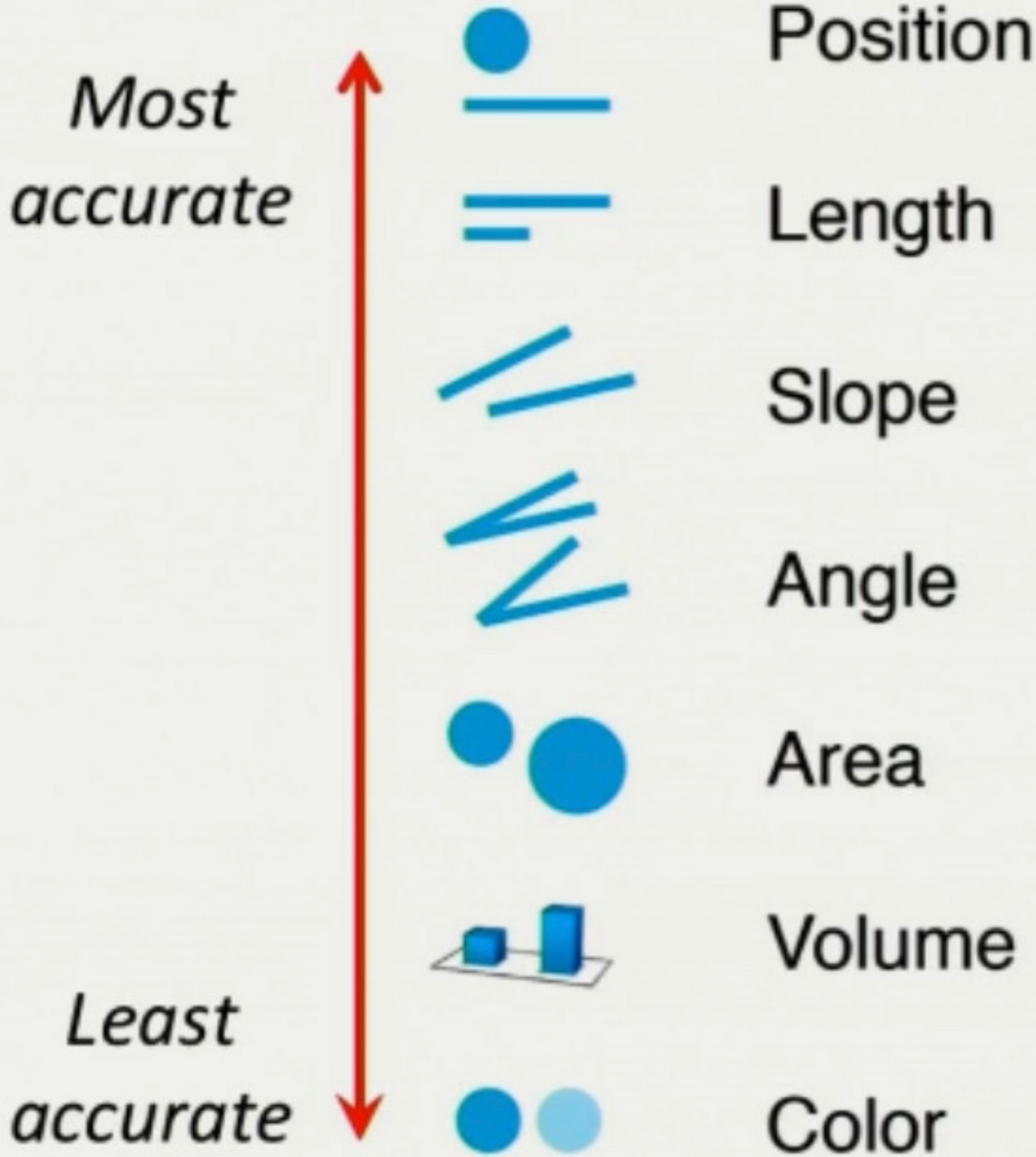
Value encodes continuous variables (less well)

How Many Colors?

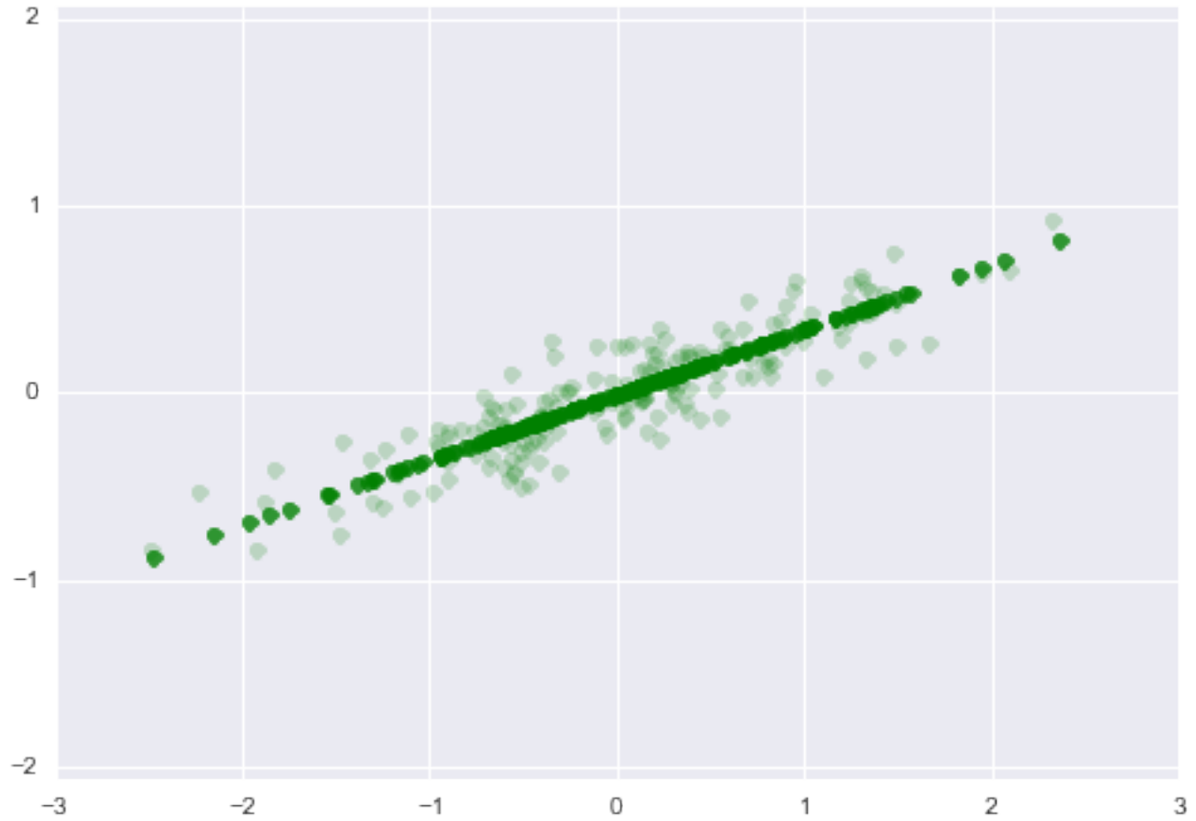
Hue encodes nominal variables

(from S. Davidoff)

Relative accuracy of the visualisation space axes:

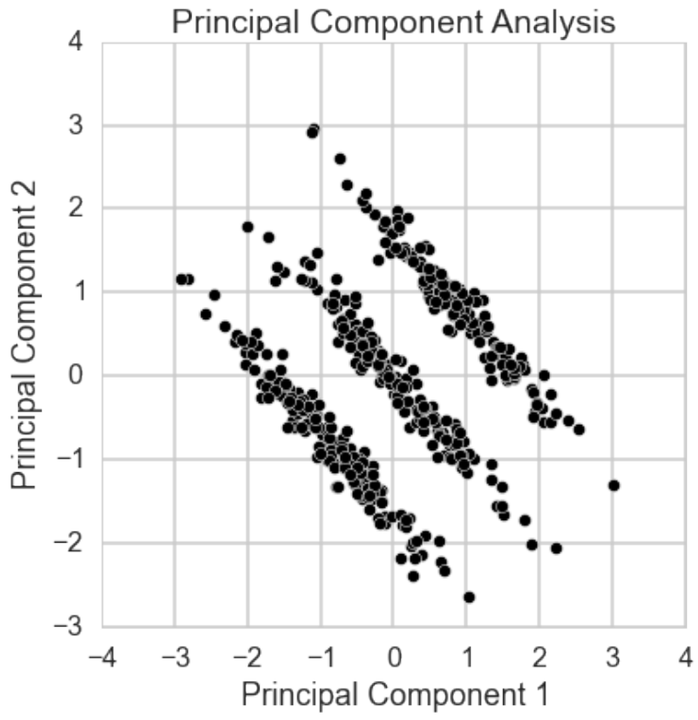


PCA (unsupervised)

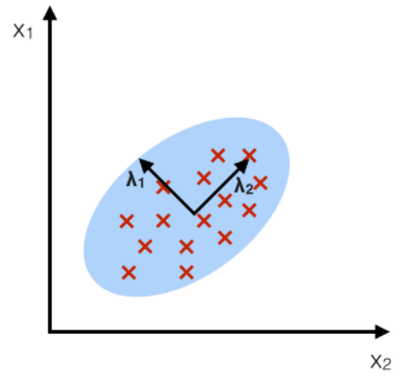


$$var(x) = \frac{\sum (x_i - \bar{x})^2}{N}$$

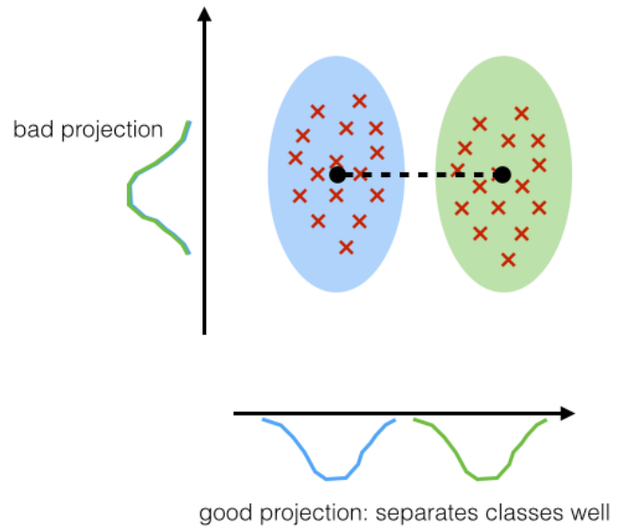
$$cov(x, y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{N}$$



PCA:
component axes that maximize the variance

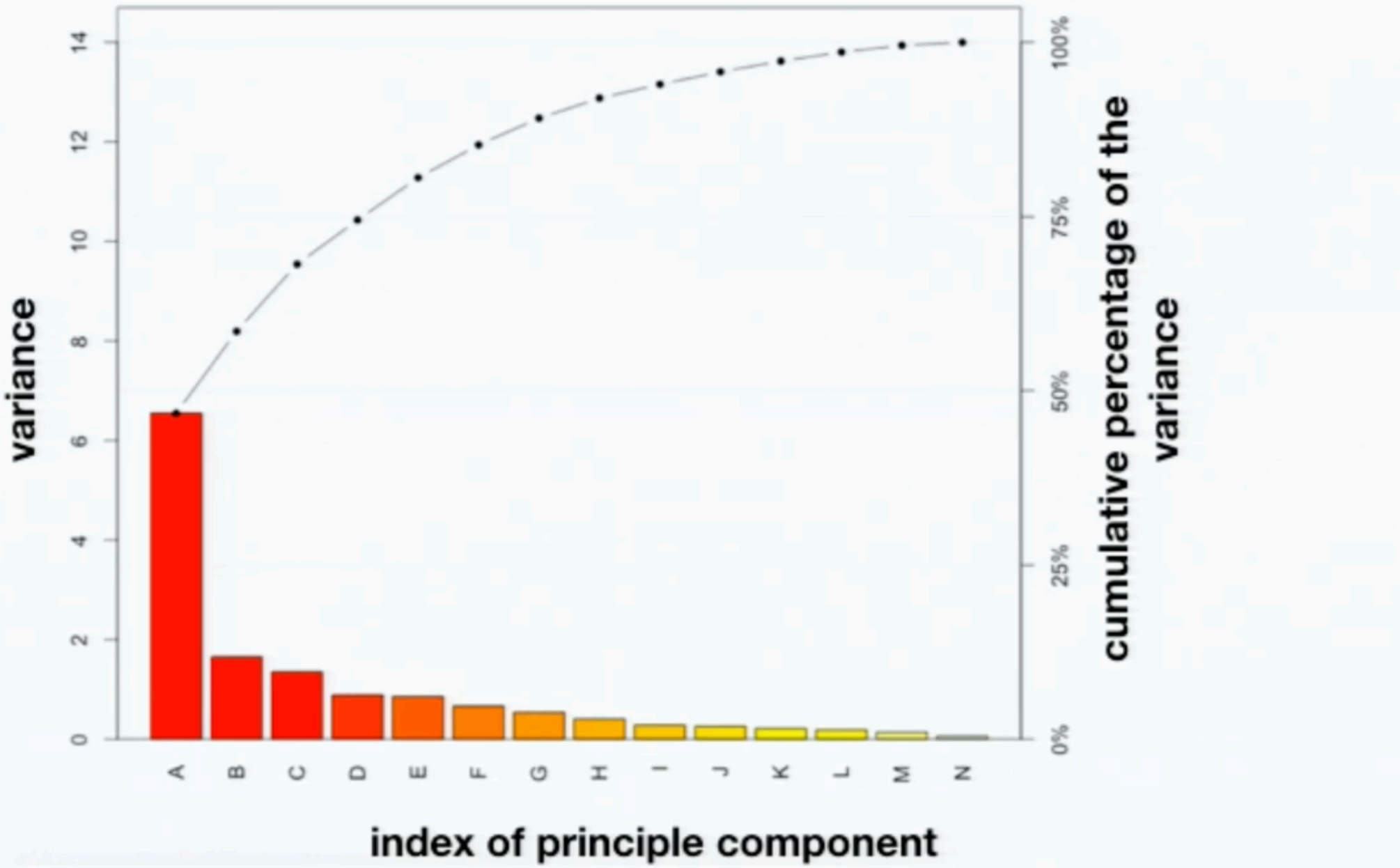


LDA:
maximizing the component axes for class-separation



Principle Component Analysis (PCA)

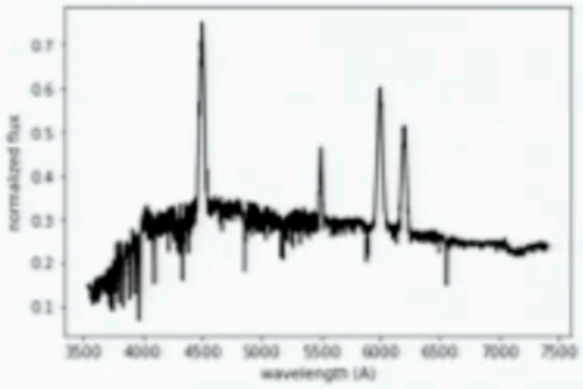
PCA allows us to compress the data, by representing each object as a projection on the first principle components.



Principle Component Analysis (PCA)

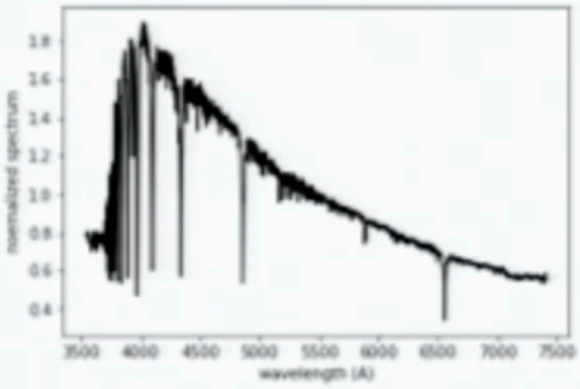
The principle components **may** represent the true **building blocks** of the objects in our dataset.

observed object



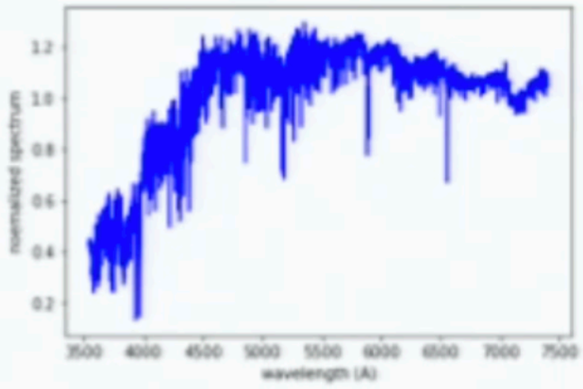
= A *

principle comp. 1



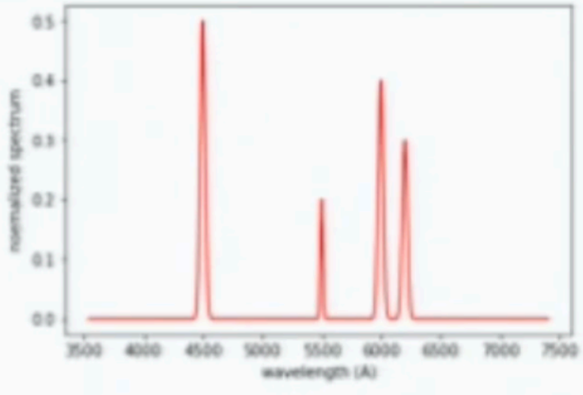
+ B *

principle comp. 2



+ C *

principle comp. 3



Cons:

PCA is a **linear** model, cannot be sensitive to non-linear attributes of the data, e.g. dust extinction, redshift etc.

Why do we need dimensionality reduction?

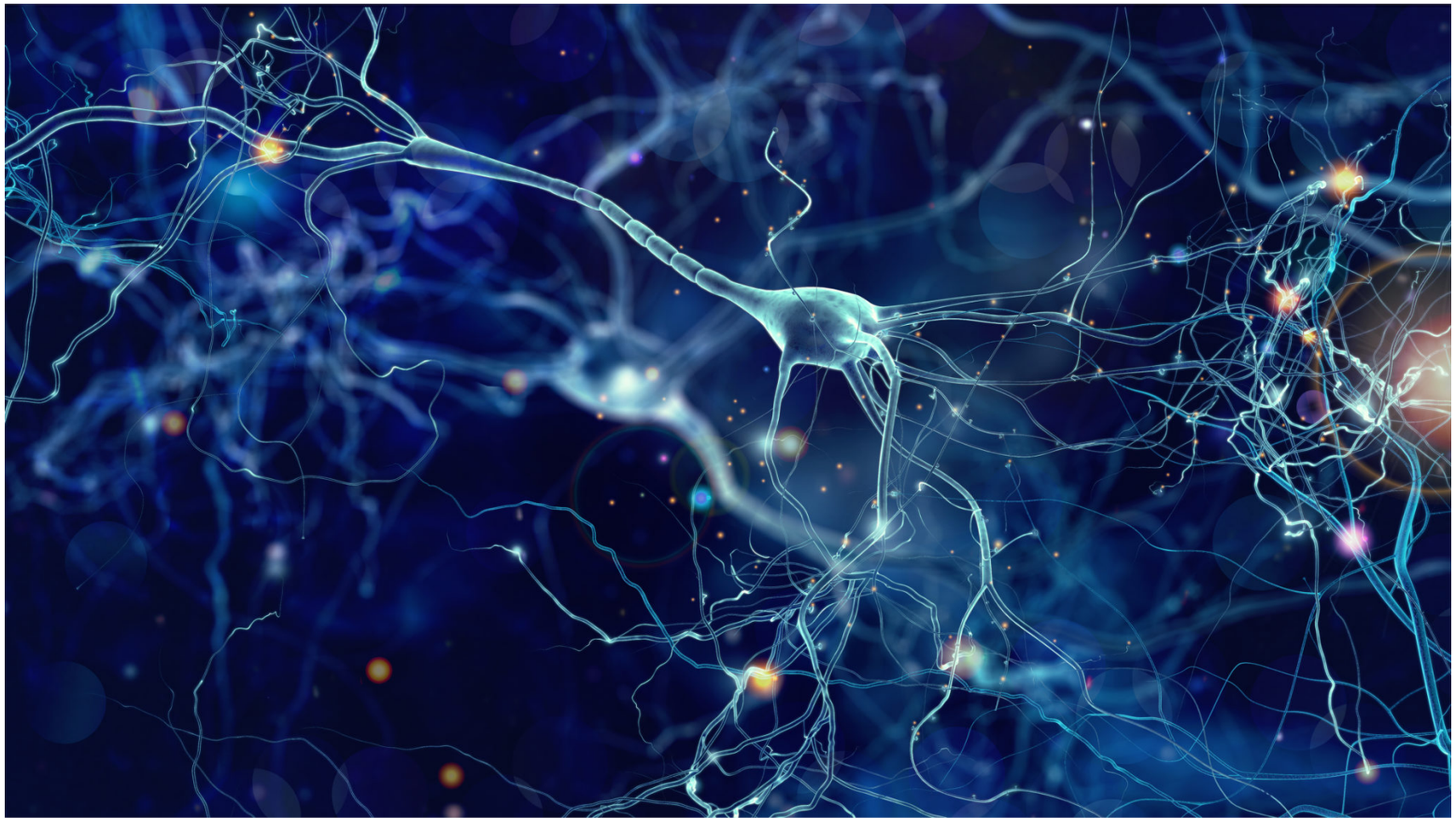
- Improve performance of supervised learning algorithms:
  - features can be correlated and redundant
  - most algorithms cannot handle 1000's of features
- Compressing data (SKA, LSST…)
- Data visualisation and interpretation
- Uncover complex trends
- Look for outliers and "unknown unknowns"

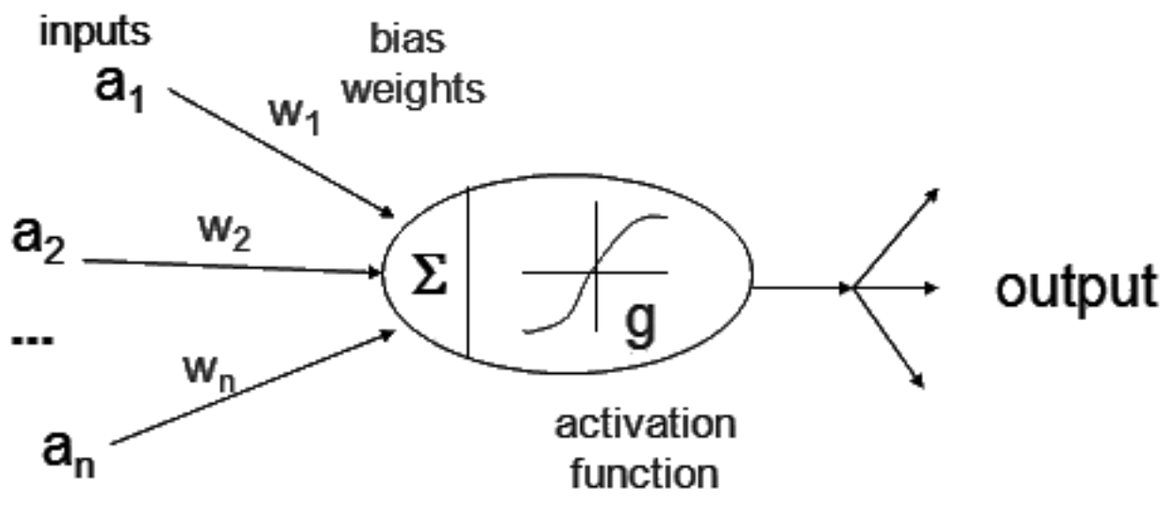
“Deep learning (neural networks) is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it….”

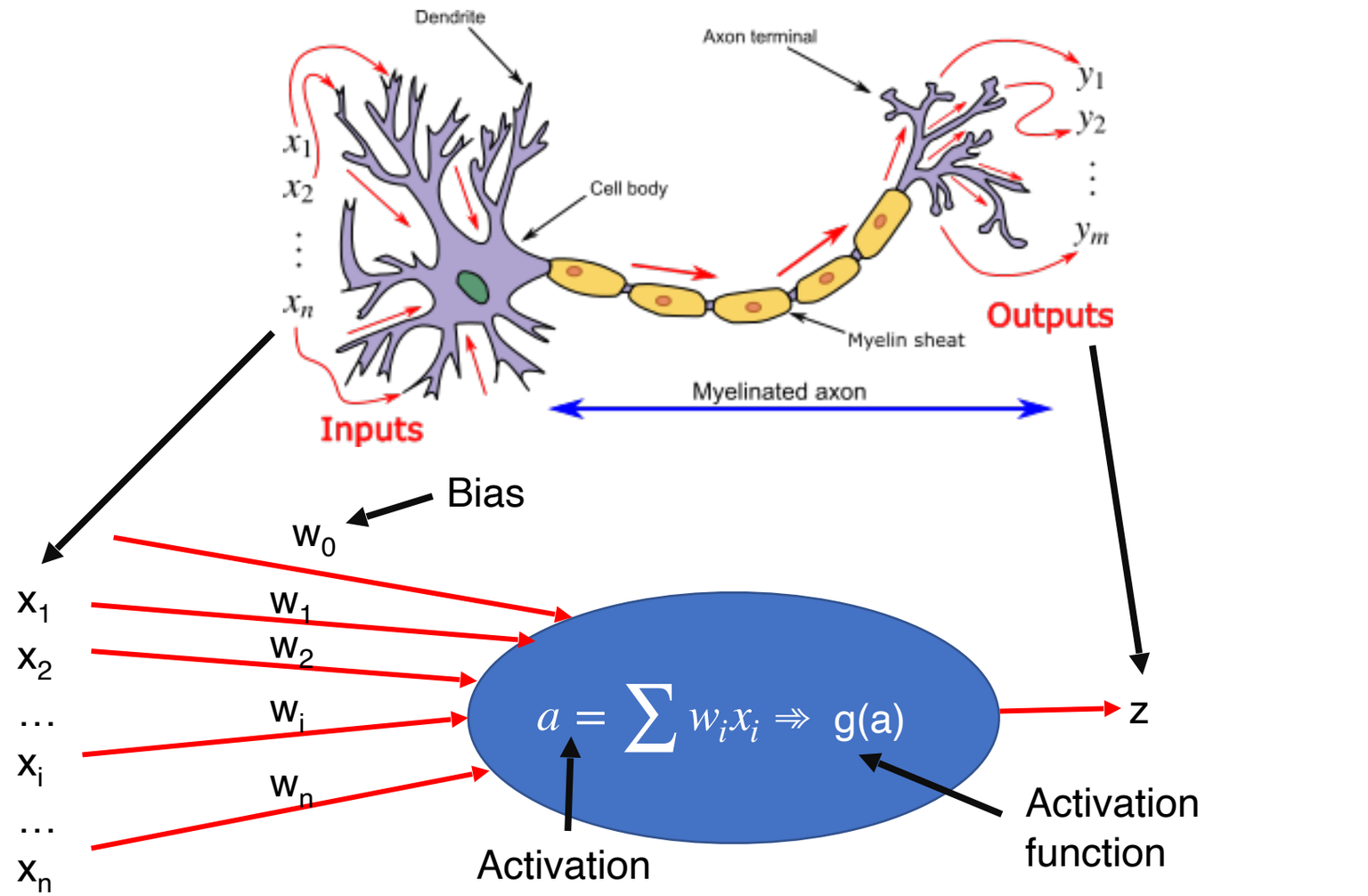
– a ML expert

Buzz words: machine learning, deep learning, AI, neural network, Bayesian analysis, gravitational waves, exoplanets, dark matter…

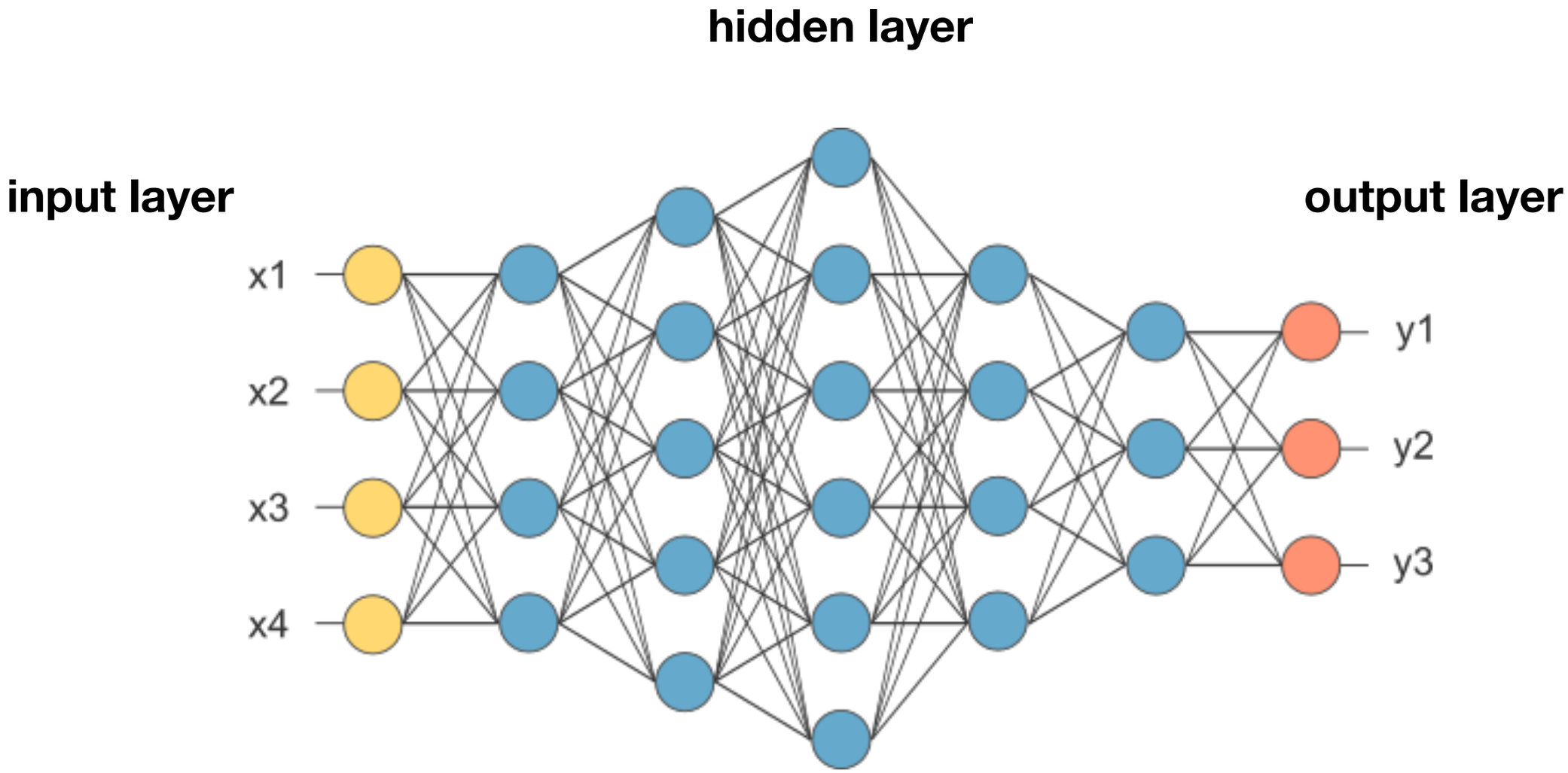
According to a report by MMC Ventures, close to 40% of European companies claiming to use AI don’t actually use it

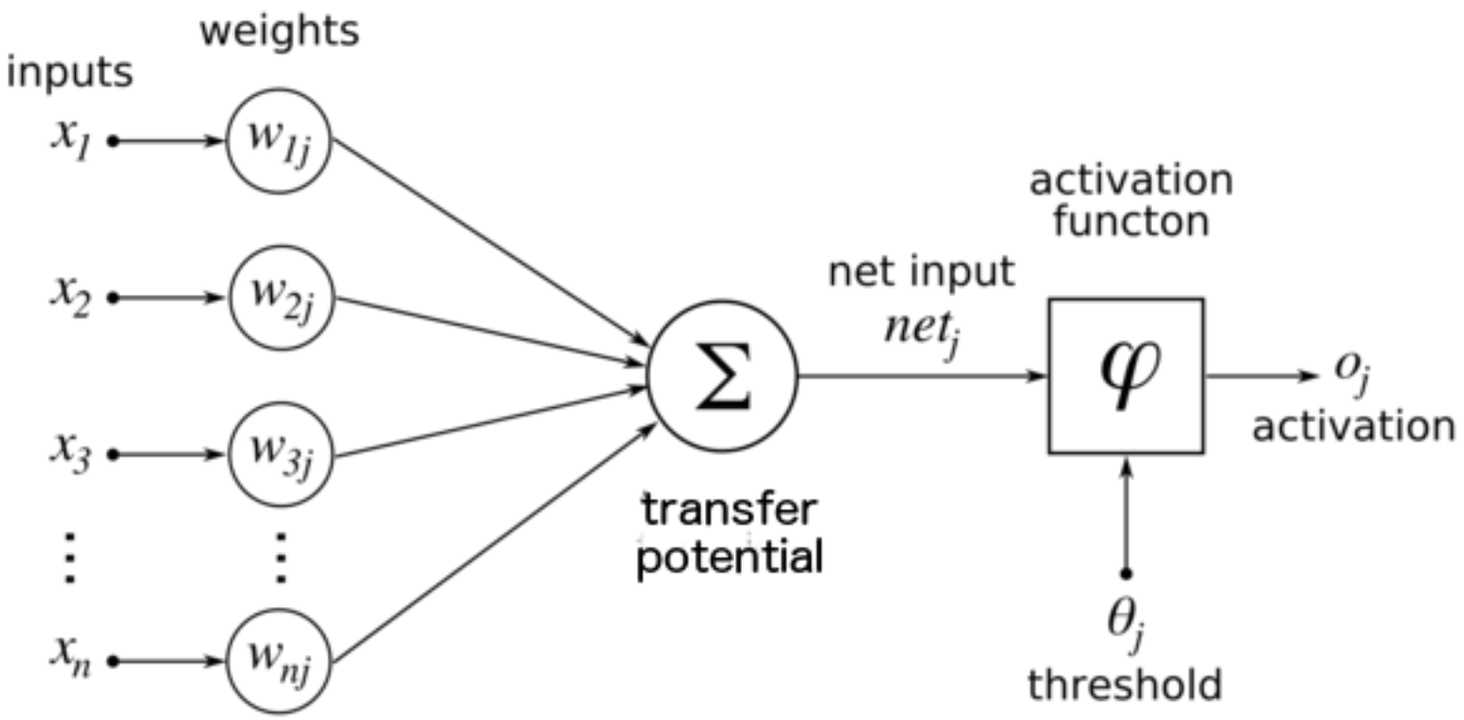



inputs
a_1
a_2
...
a_n
w_1
w_2
w_n
bias
weights
Σ
g
activation
function
output



g = identity or sigmoid
Linear/logistic regression

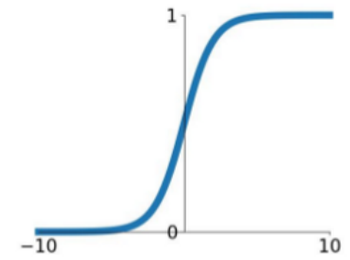

hidden layer
input layer
output layer
x1
x2
x3
x4
y1
y2
y3



Activation Functions

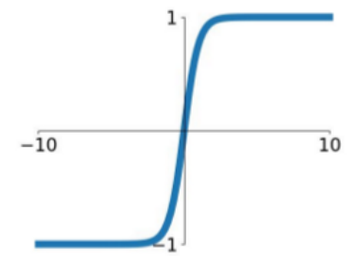
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



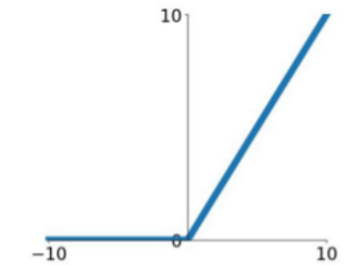
tanh

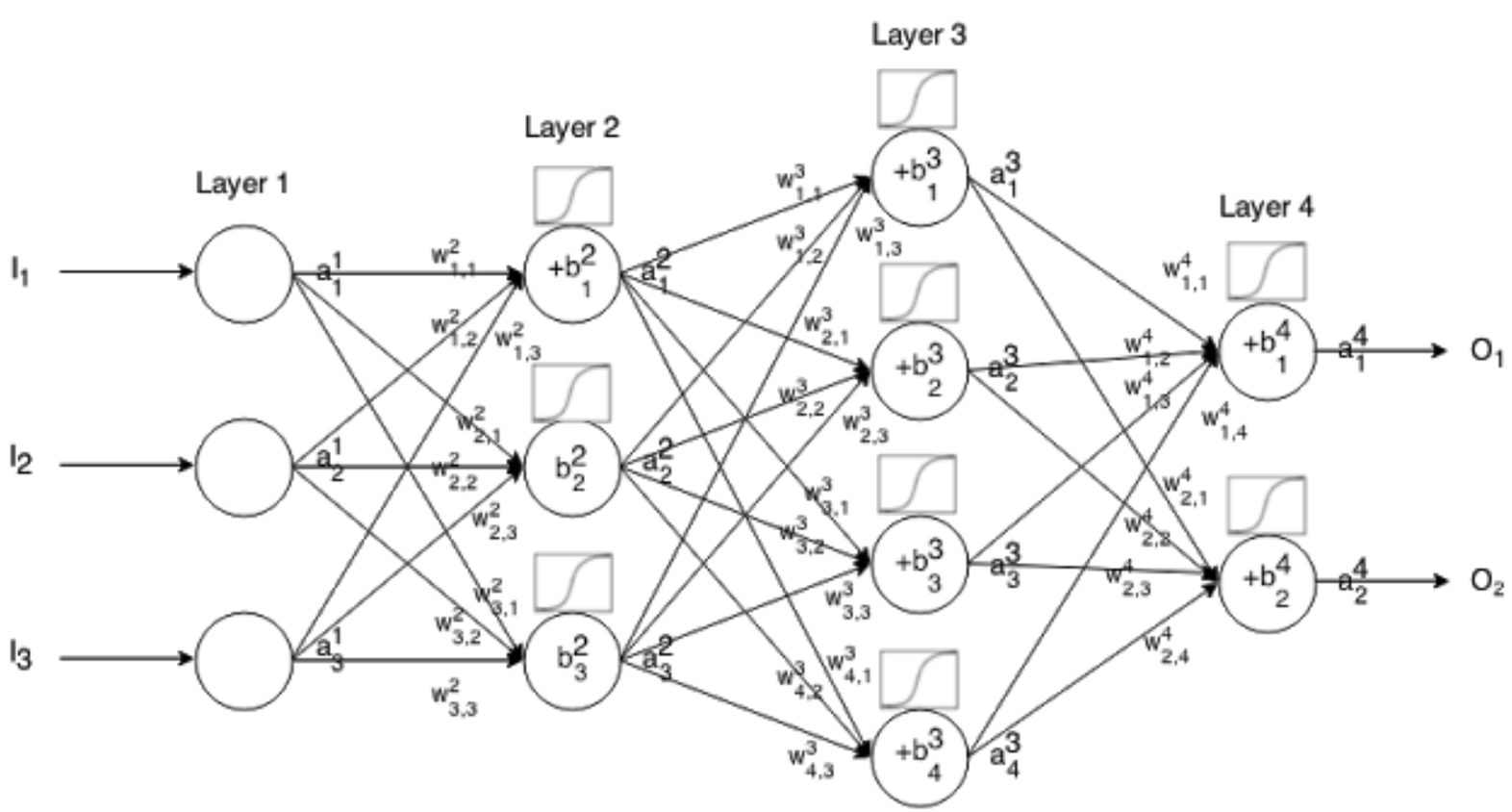
$$\tanh(x)$$

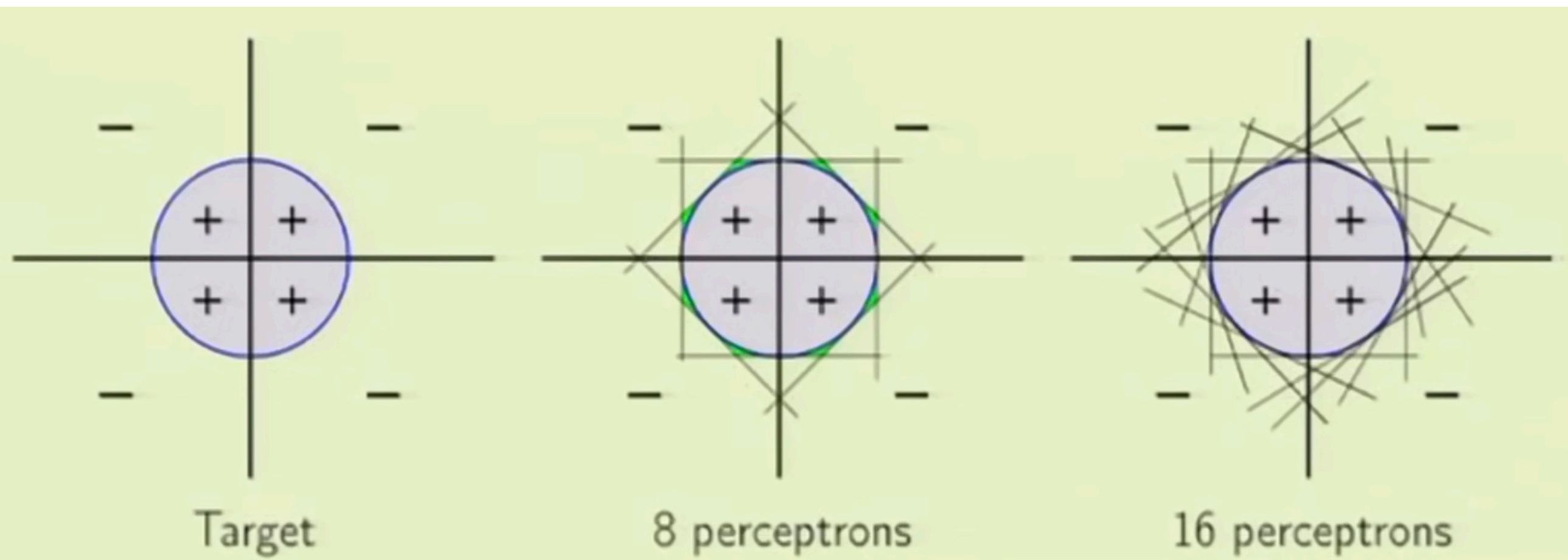


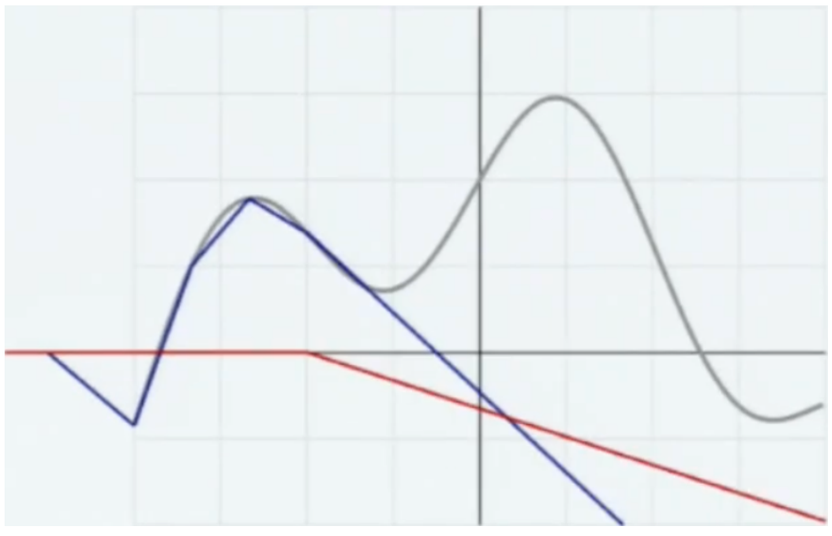
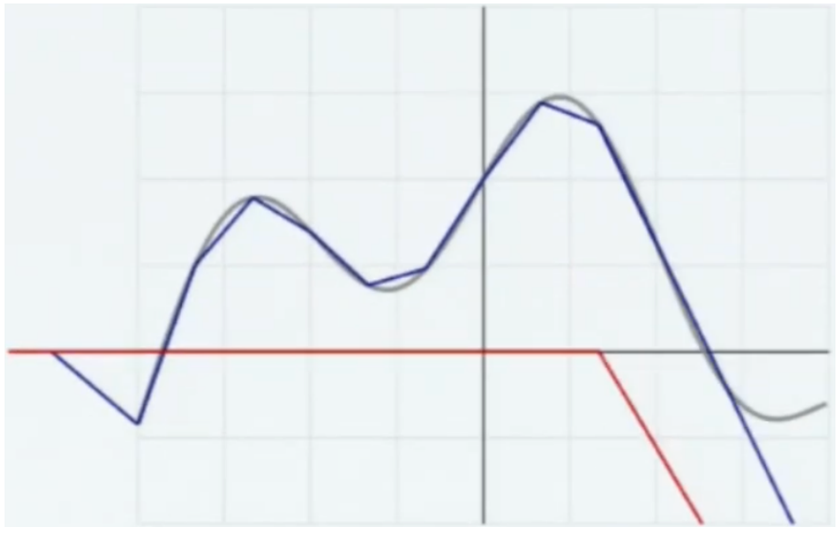
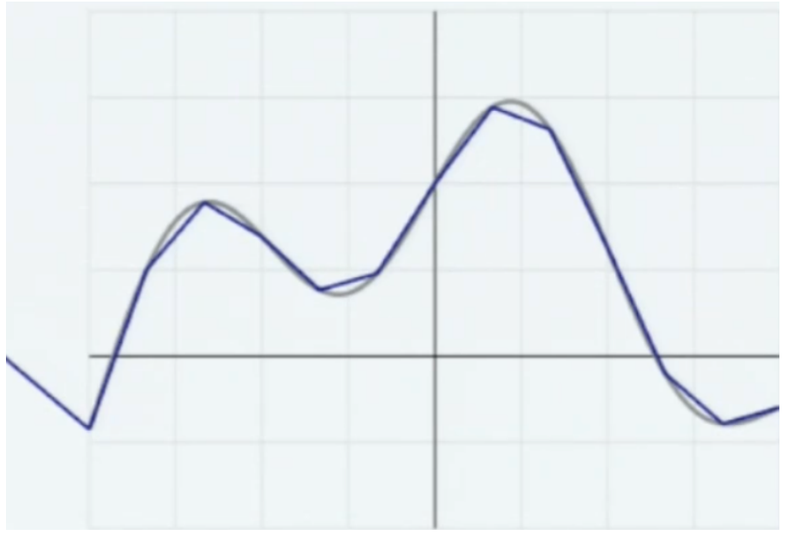
ReLU

$$\max(0, x)$$






Target
8 perceptrons
16 perceptrons

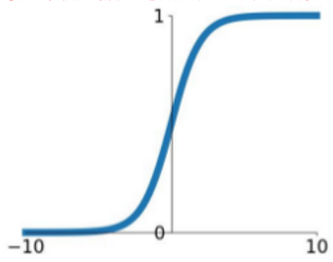




Activation Functions

[0,1] "probability" final layer for classification

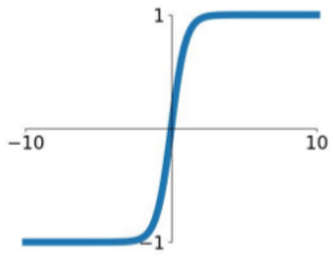
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



tanh

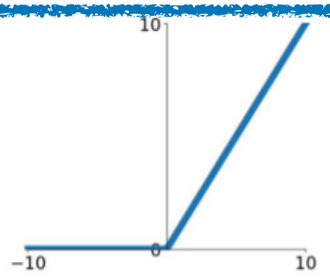
$$\tanh(x)$$



robust against "vanishing gradient"

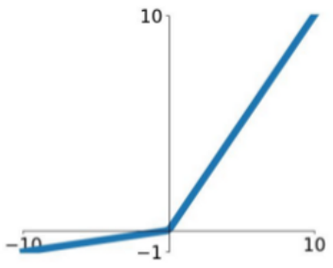
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

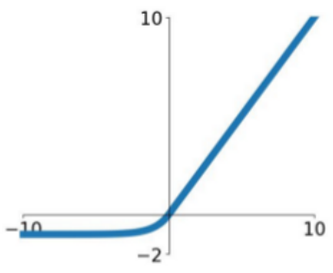


Maxout

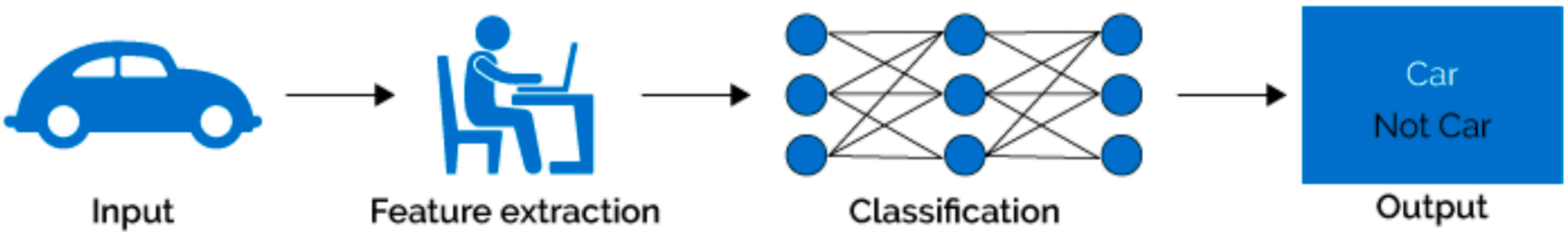
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

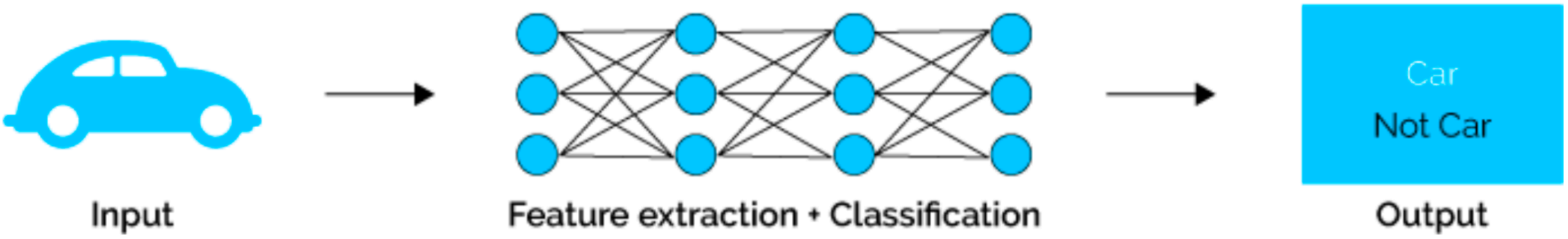
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



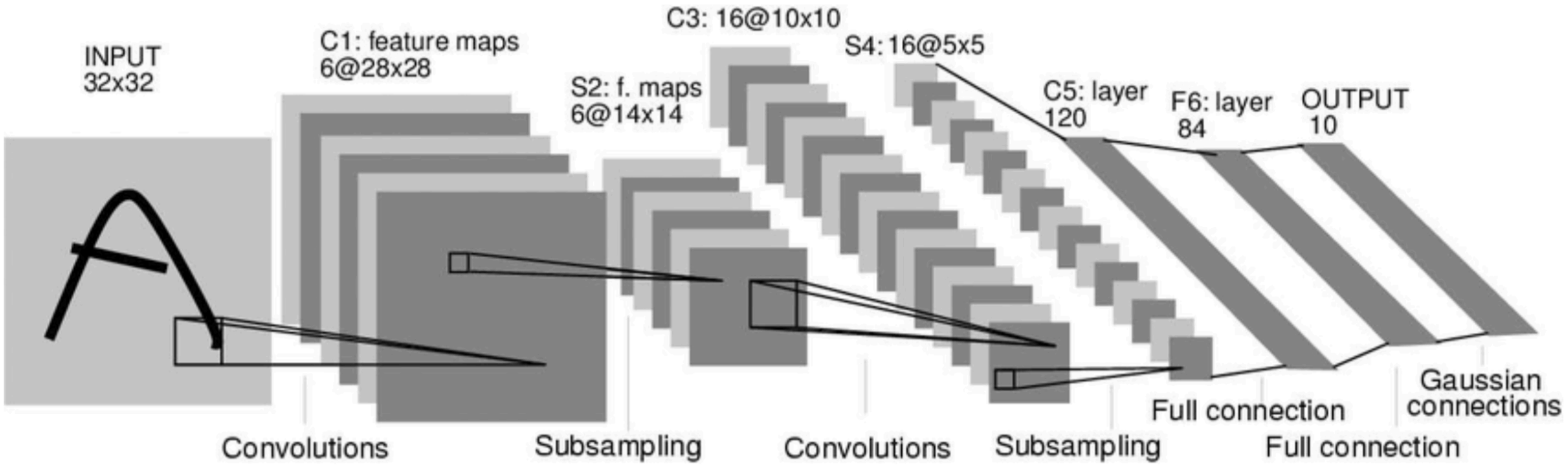
Machine Learning

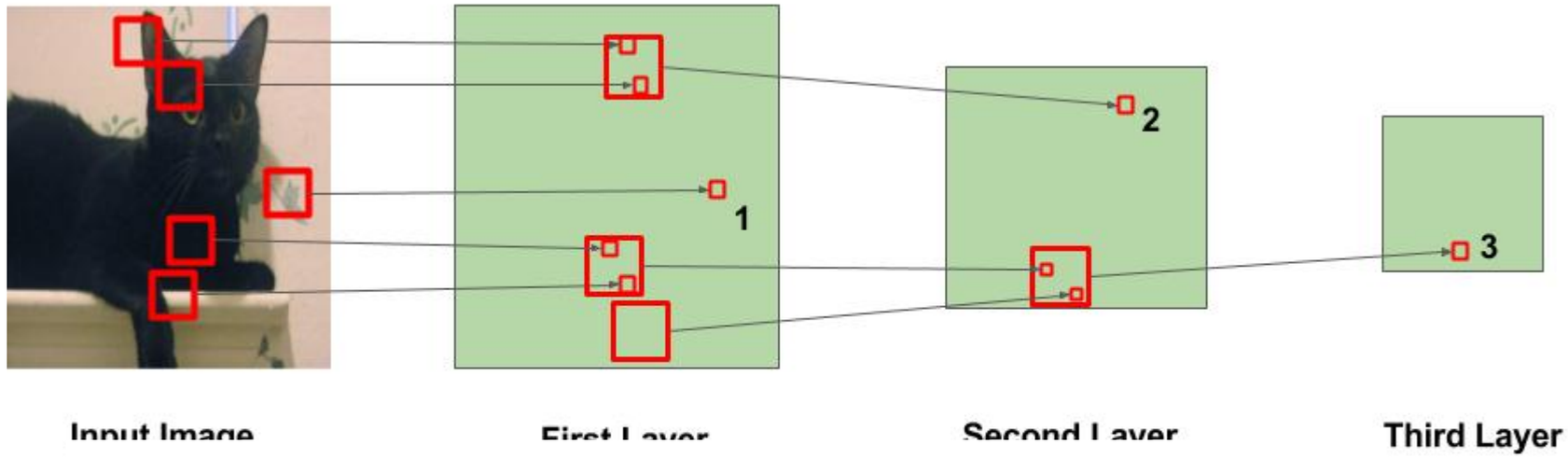


Deep Learning

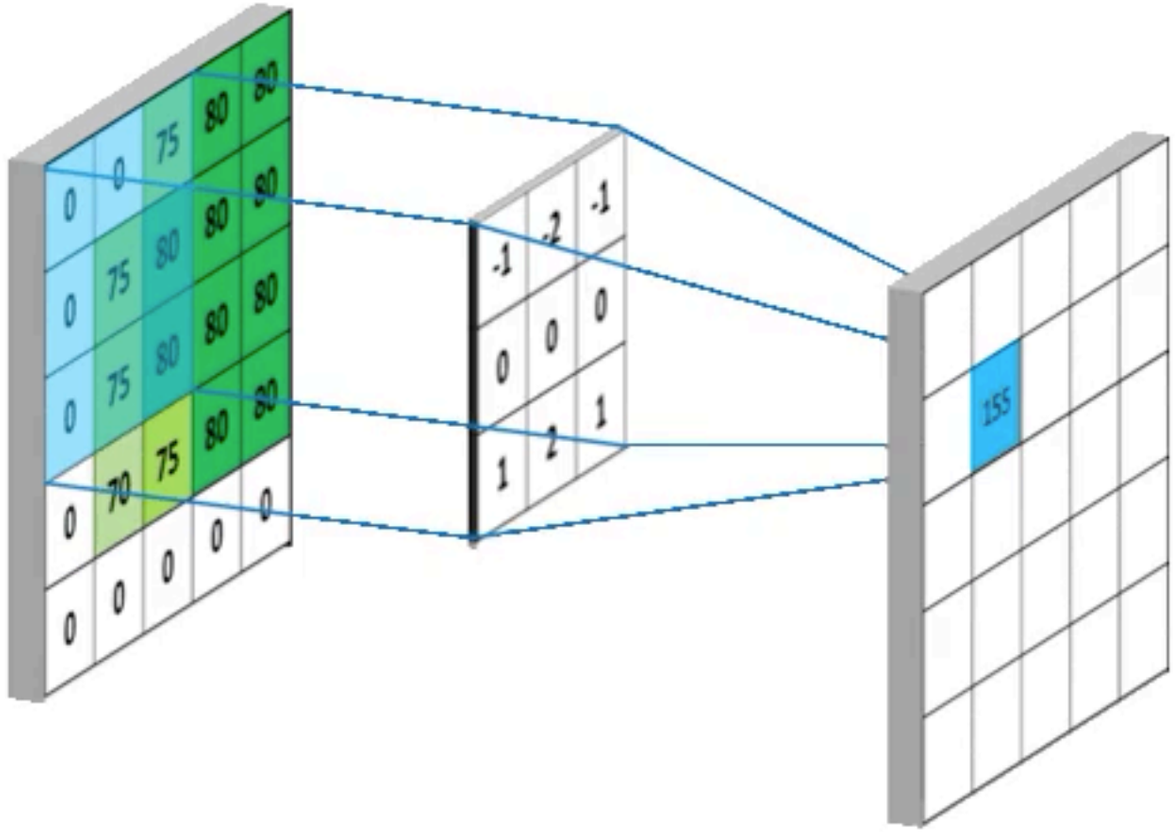


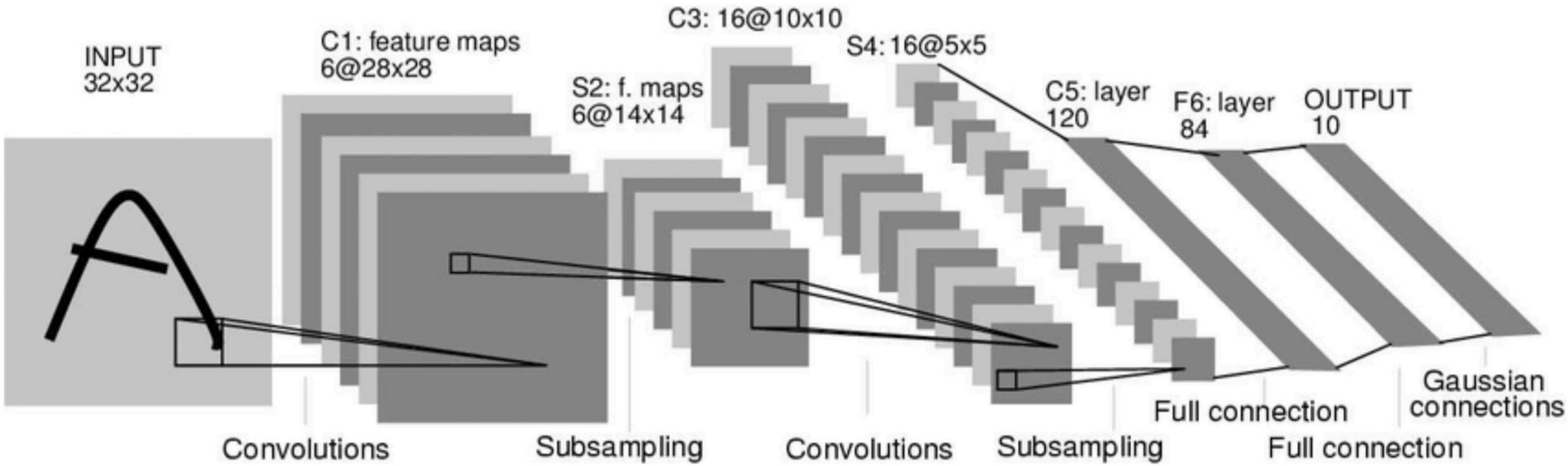
Convolutional Neural Networks





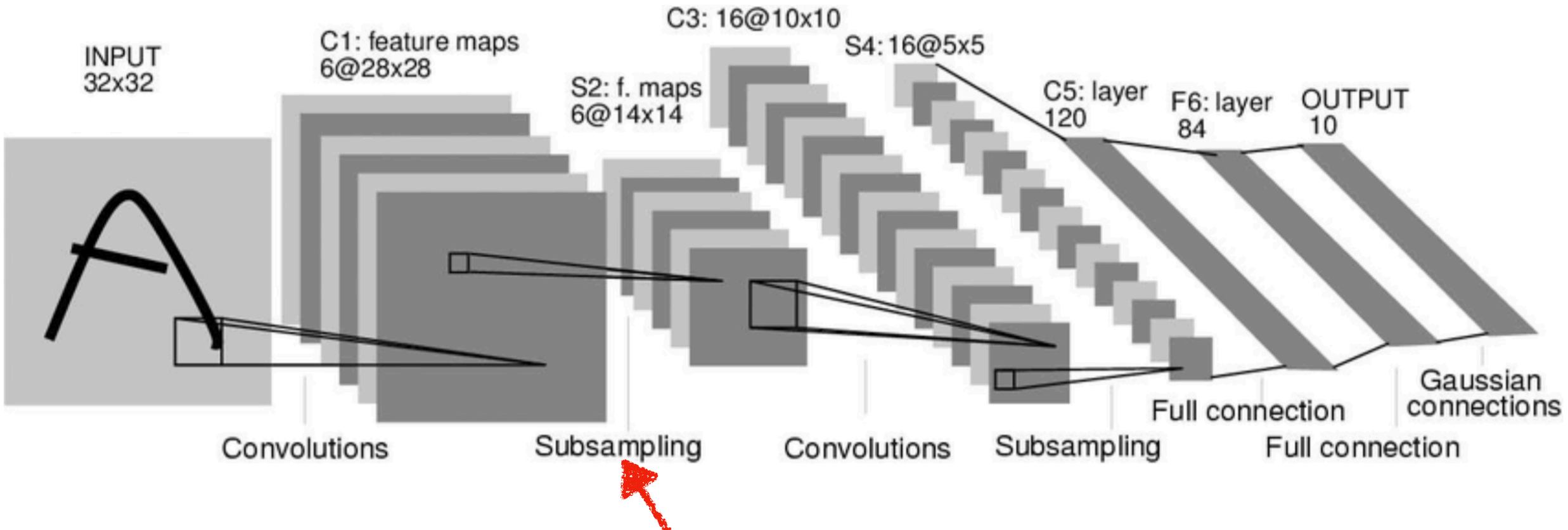
Convolution (with a kernel that is a parameter to be fit)





feature kernels

flattening
and
fully connected
neural network



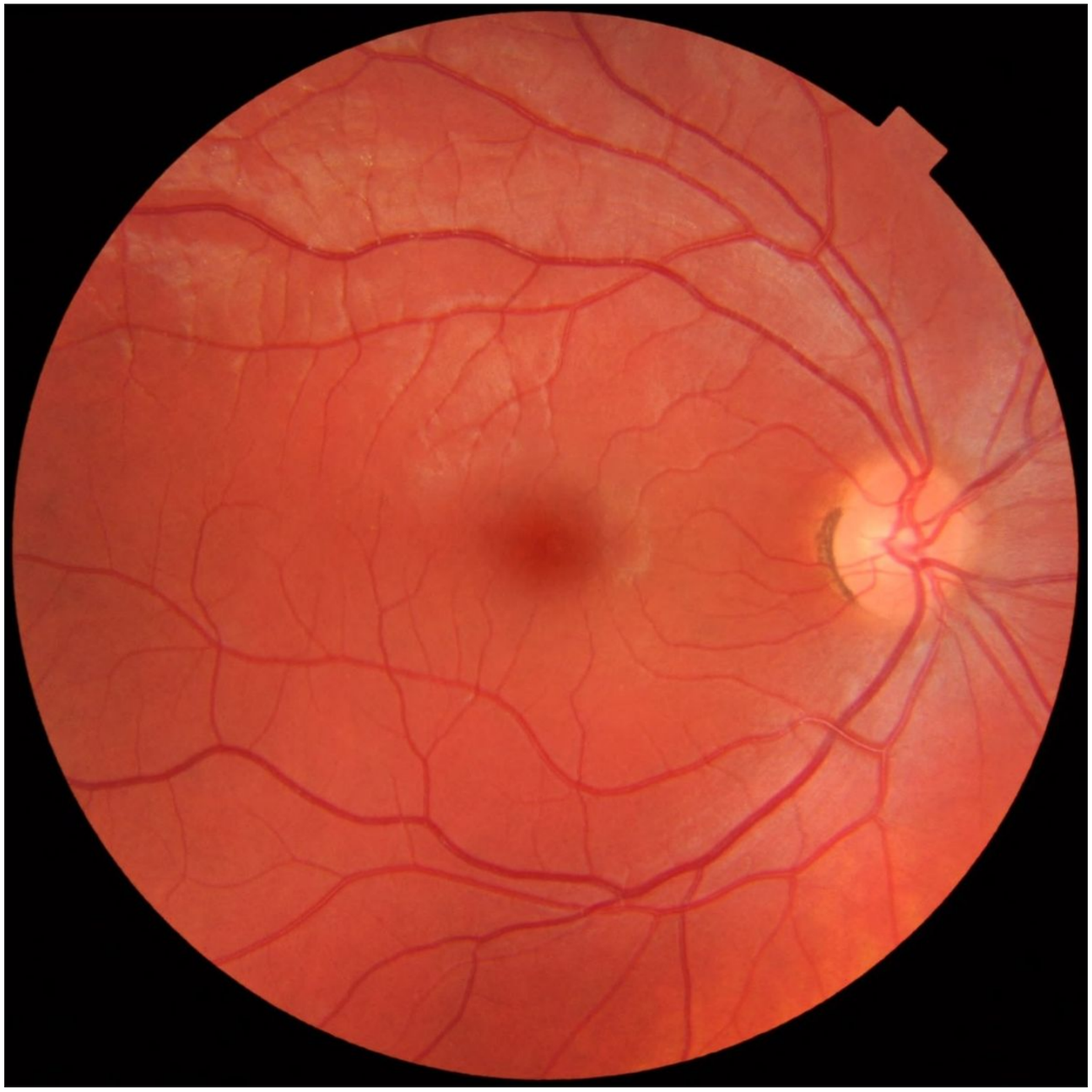
pooling (max, min, mean…)

steps in convolution can be also used

12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

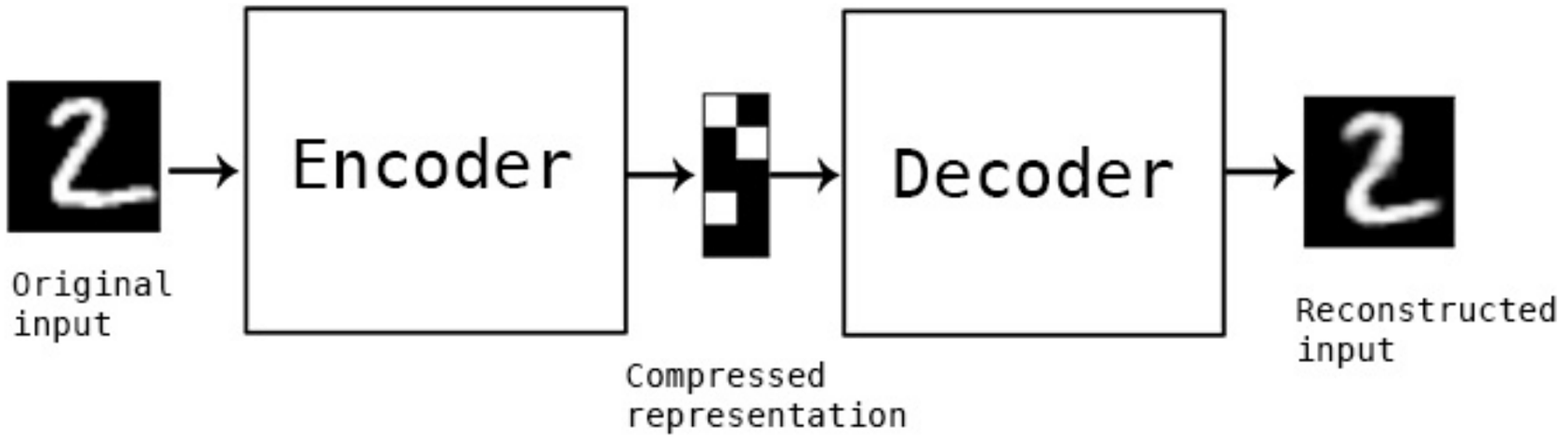
2×2 Max-Pool

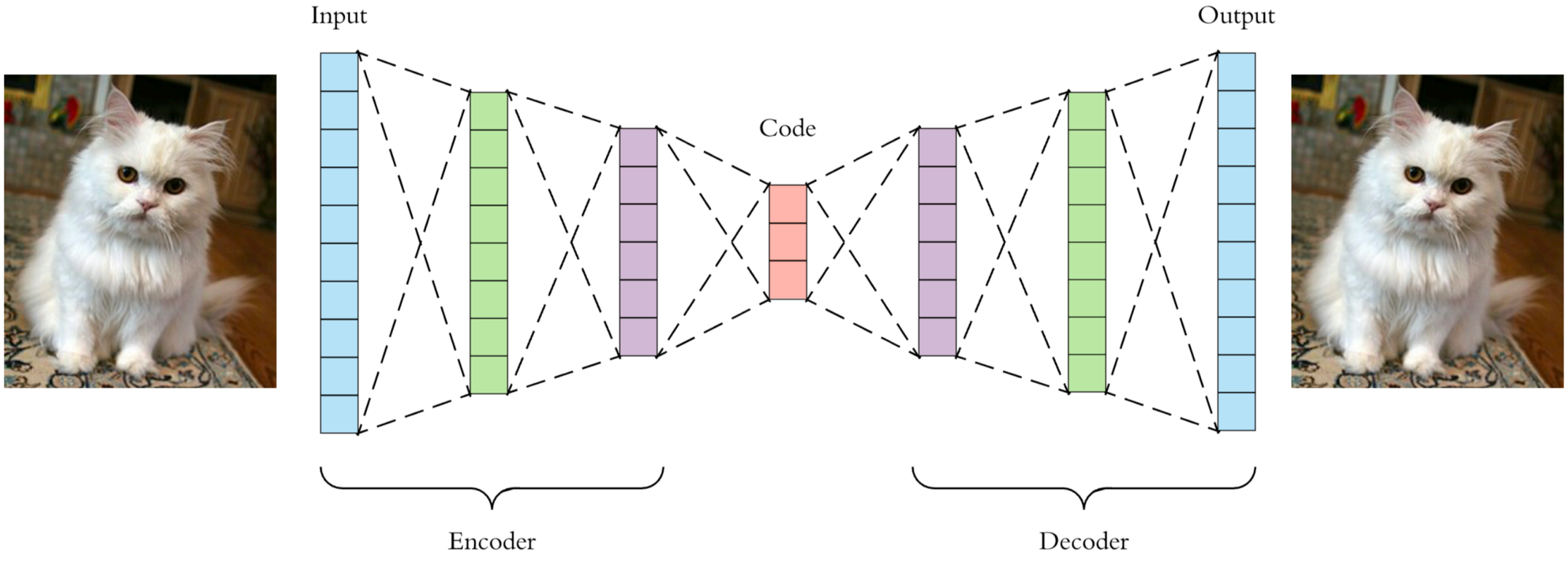
20	30
112	37



Deep learning can identify the patients gender with 95% accuracy!

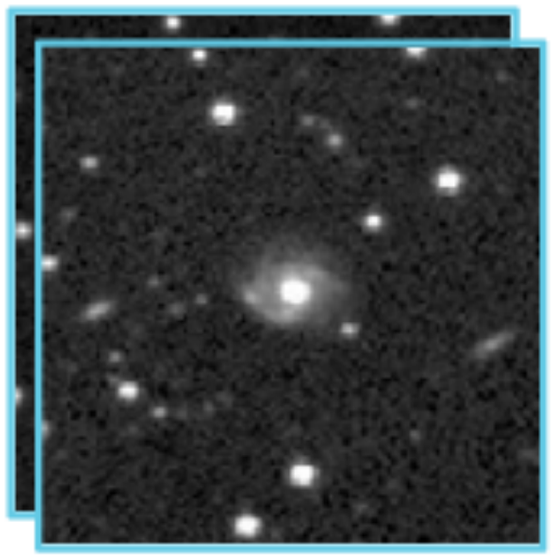
Encoder-decoders (Autoencoders)

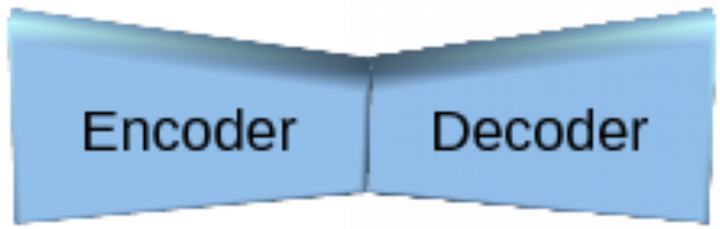




Loss = ∑(original-product)²

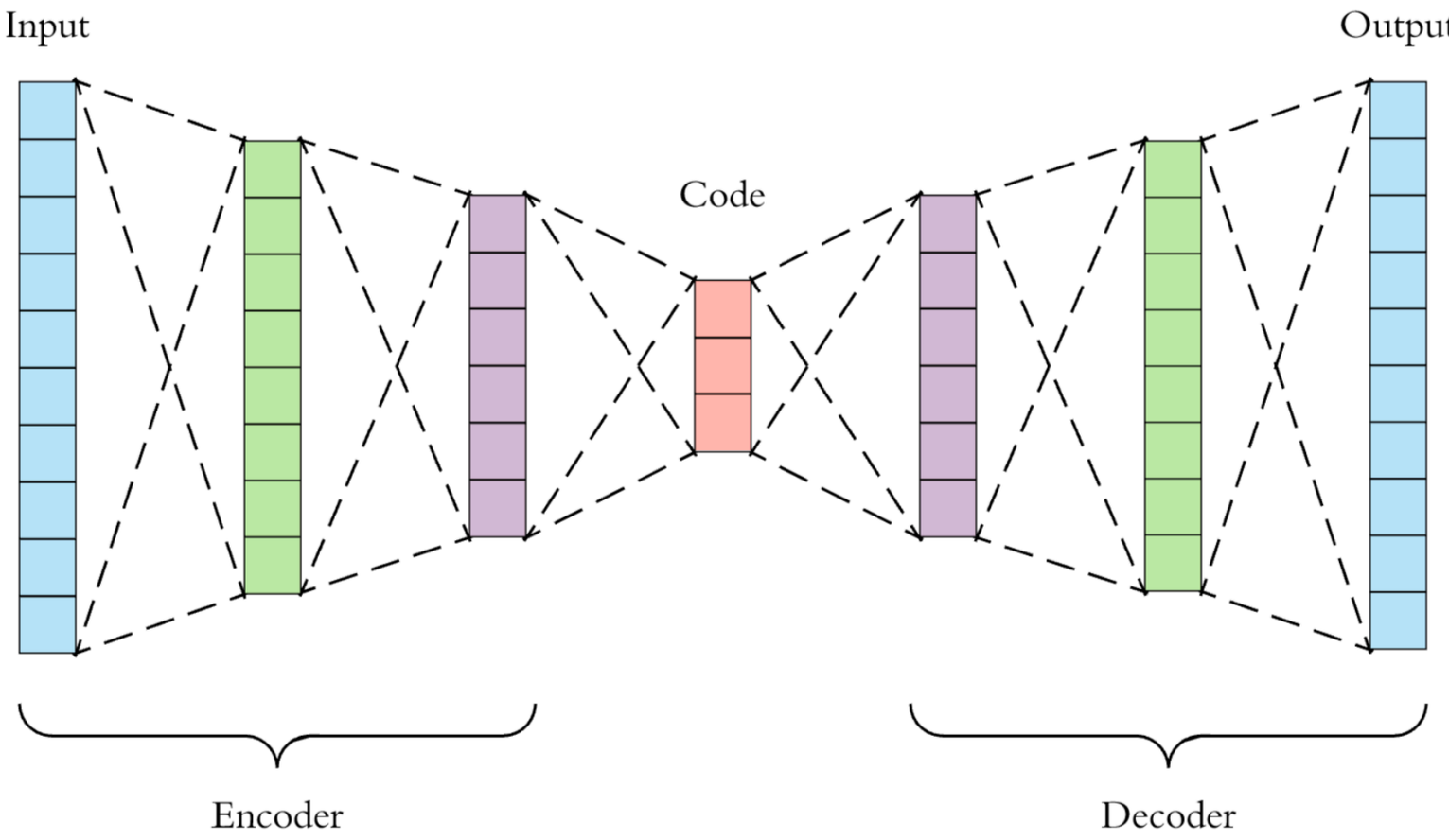
TransiNet



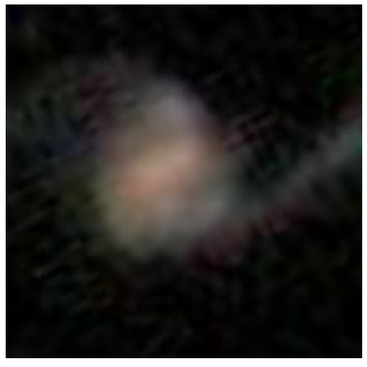

Encoder
Decoder

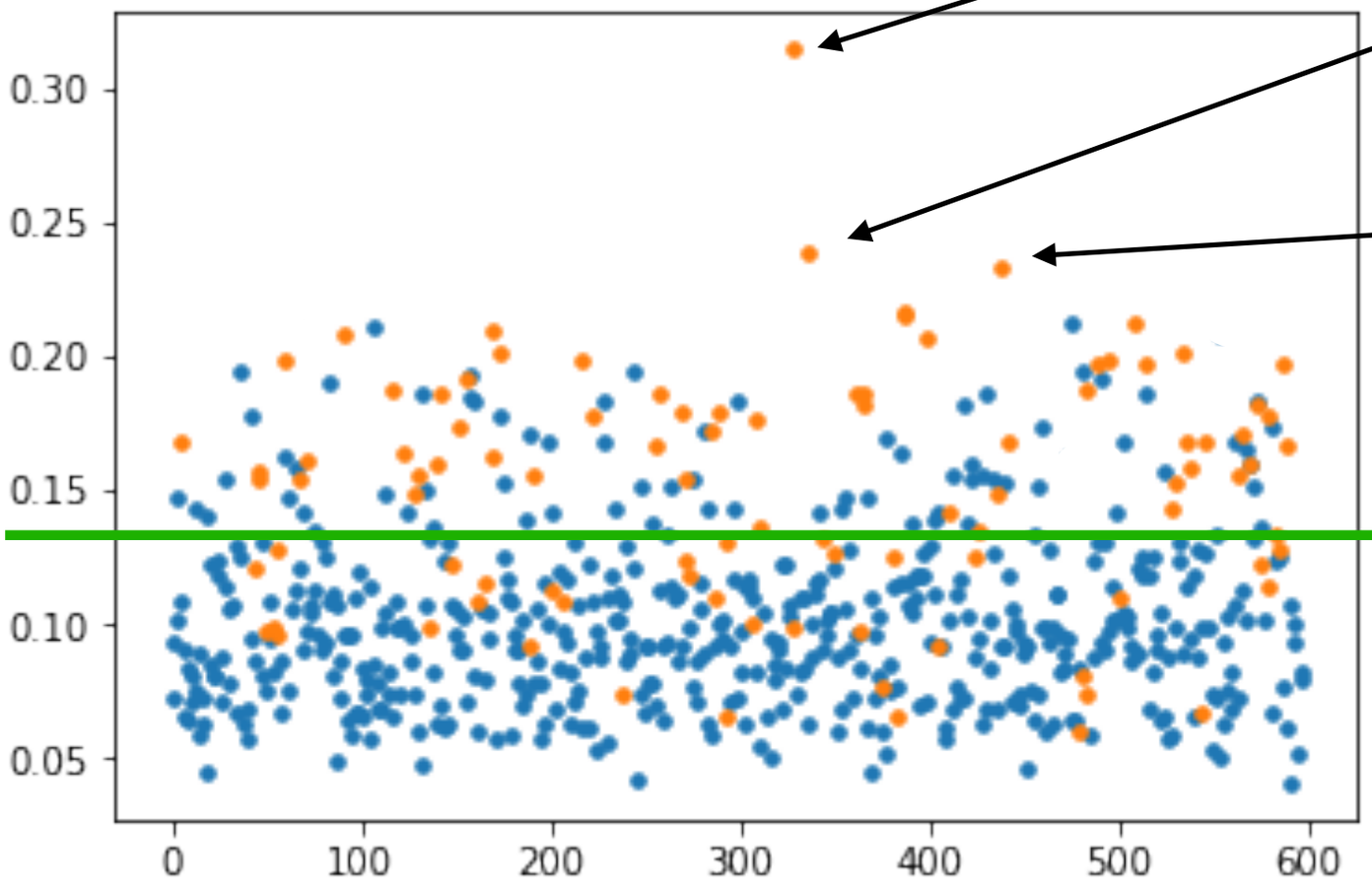
Input
Conv1
7x7 : 128
Conv2
5x5 : 256
Conv3
5x5 : 512
Conv3_1
3x3 : 512
Conv4
3x3 : 1024
Conv4_1
3x3 : 1024
Conv5
3x3 : 1024
Conv5_1
3x3 : 1024
Conv6
3x3 : 2048
Conv6_1
3x3 : 2048
L1 Loss
Output
Upconv0
4x4 : 32
Upconv1
4x4 : 64
Upconv2
4x4 : 128
Upconv3
4x4 : 256
Upconv4
4x4 : 512
Upconv5
4x4 : 1024

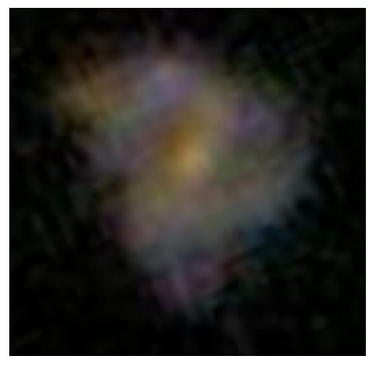
E.g. De-noising



Anomaly detection

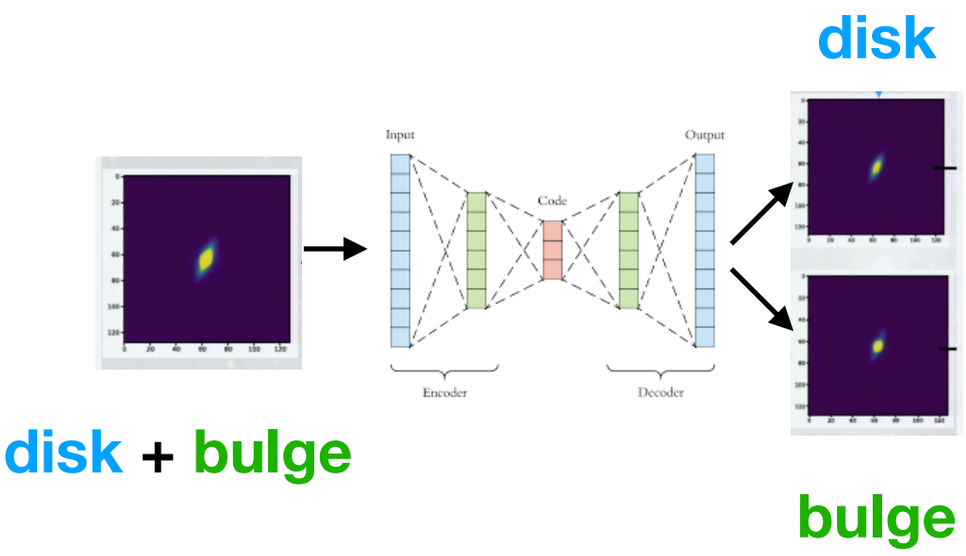






Encoder/decoder usage

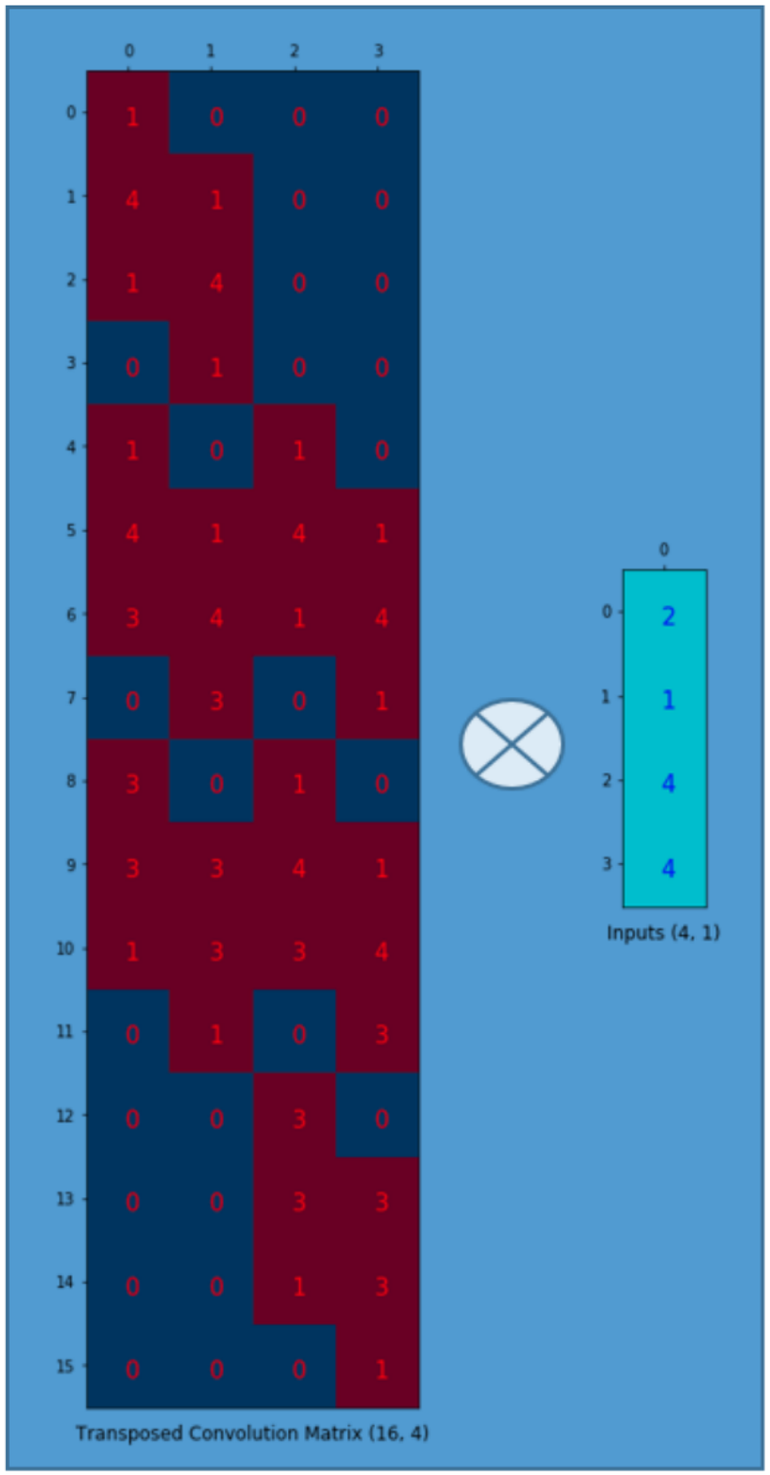
- Dimensionality reduction
- Denoising
- Outlier detection (measure of reconstruction error)
- Image segmentation
- Deblending
- Predicting next move
- Many others…
- (Can be considered as an unsupervised learning)

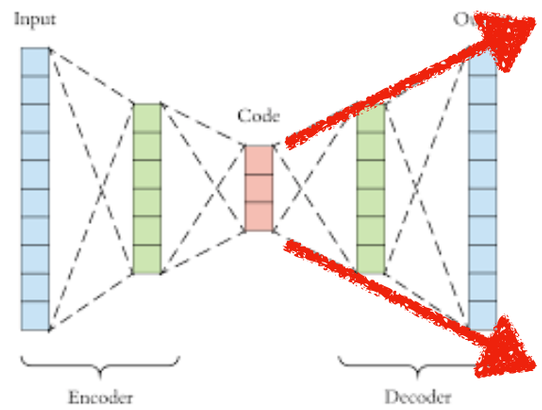


Transposed convolution (up or inverse convolution)

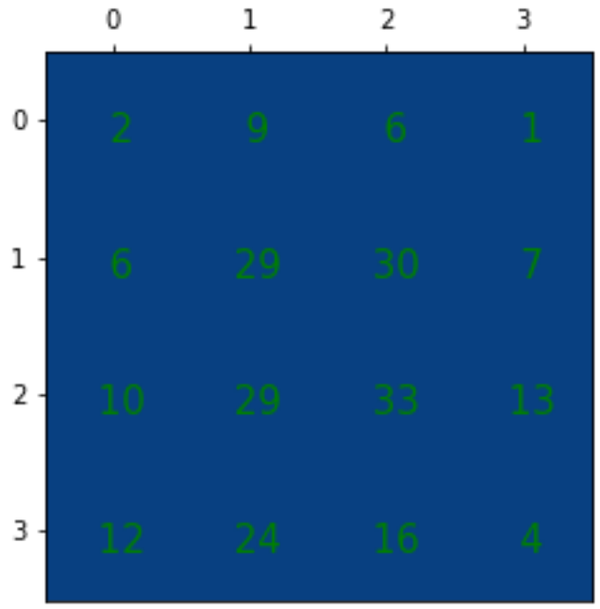
4x4 kernel (filter)

↓ **restructure**





reshape



find inverse matrix

THIS IS YOUR MACHINE LEARNING SYSTEM?

YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.

WHAT IF THE ANSWERS ARE WRONG?

JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.