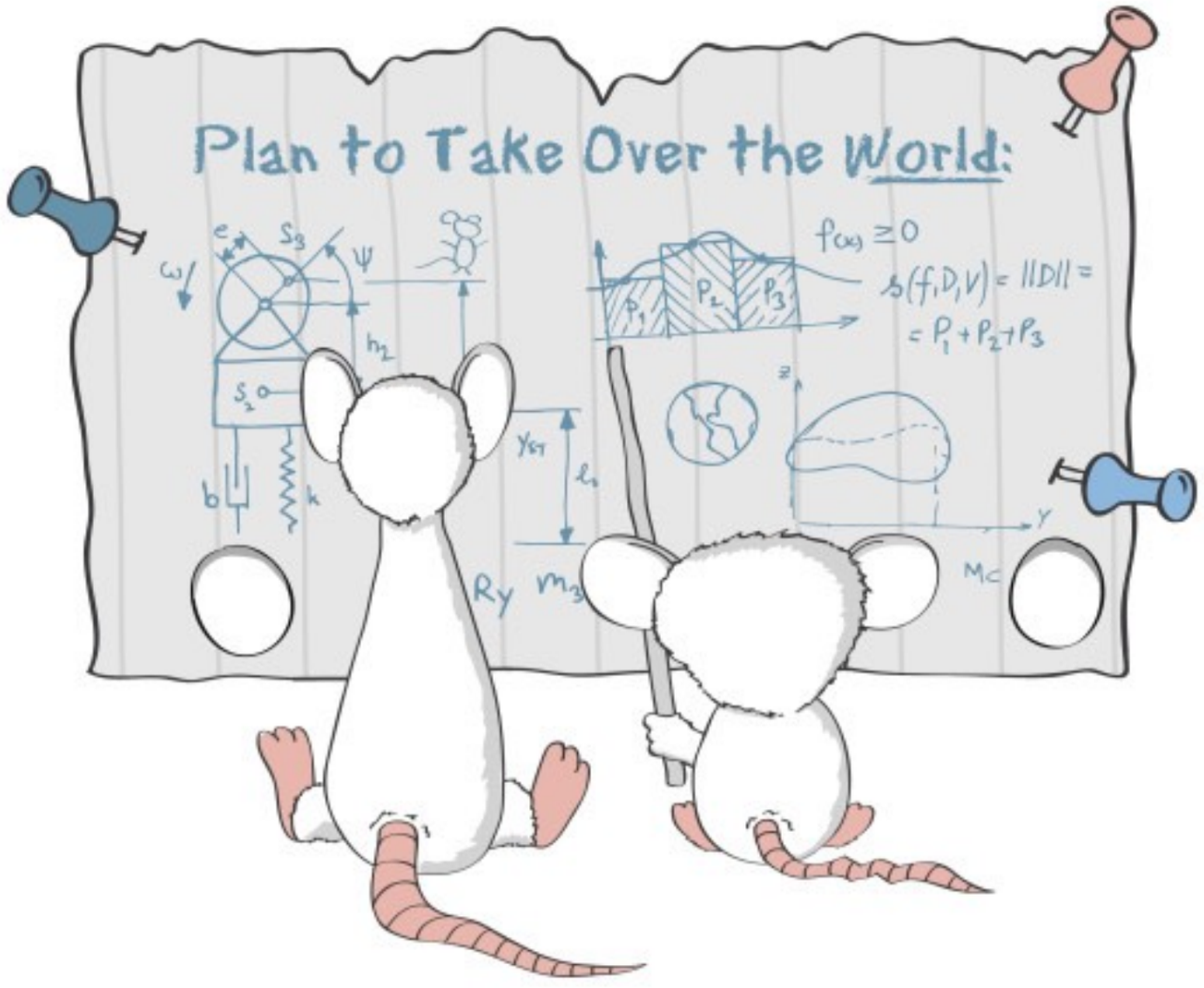


Lecture #3

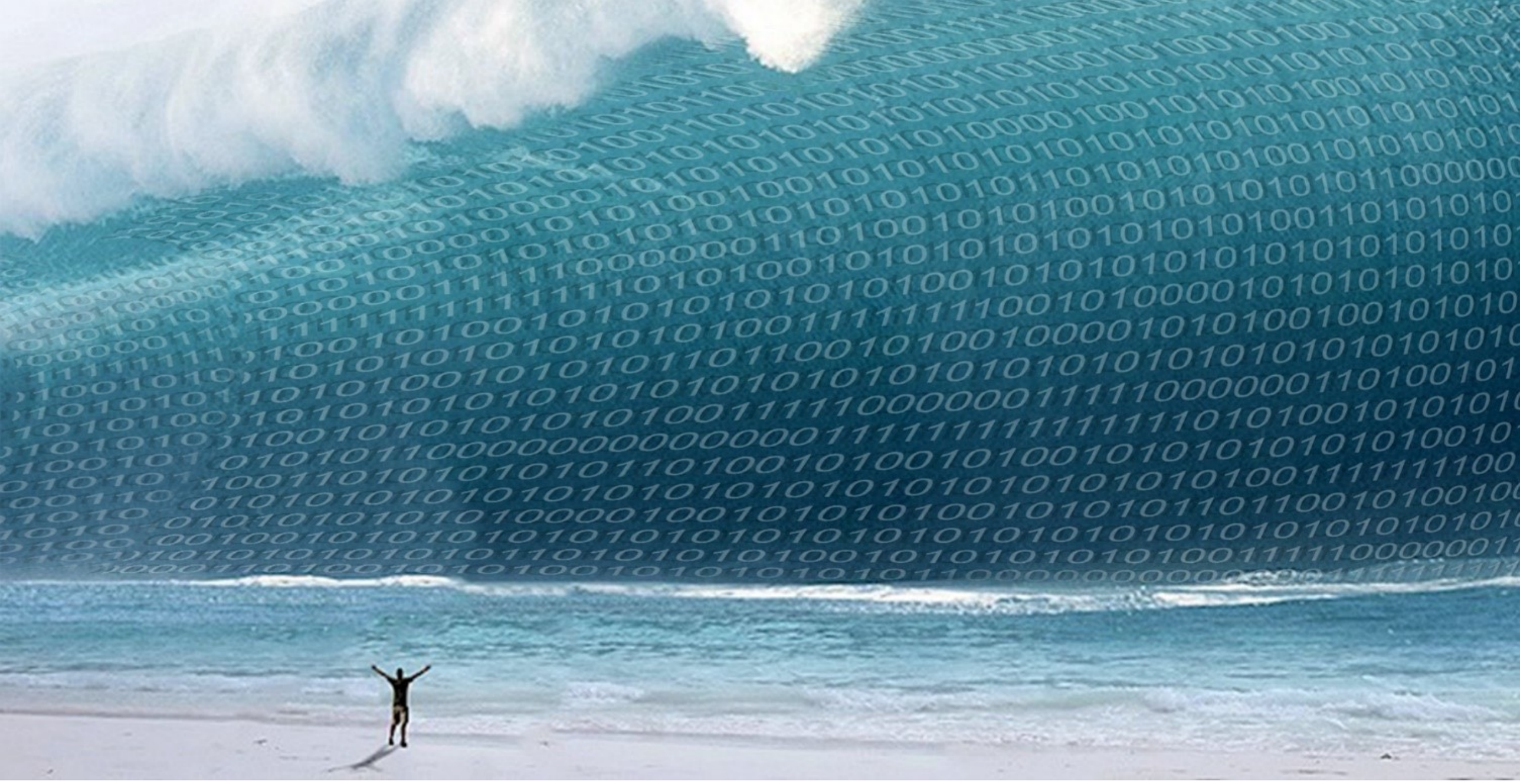
F5611 Machine Learning for Astronomers
by Martin Topinka

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

Guest lecture: 25 Nov 2021 Nima Sedaghat (ESO)

Machines Learn to Infer Stellar Parameters Just by Looking at a Large Number of Spectra

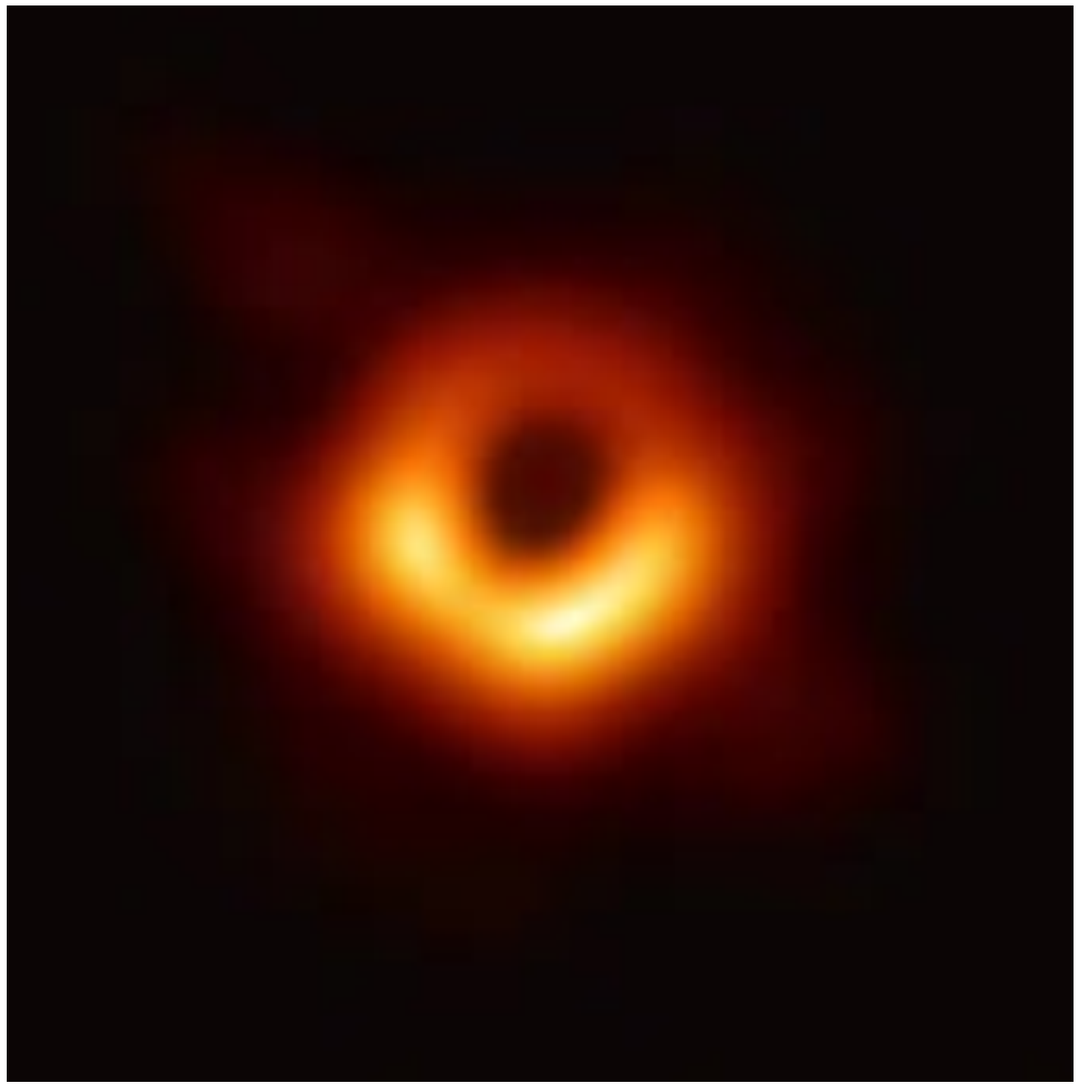
Nima Sedaghat,[1]★ Martino Romaniello,[1] Jonathan E. Carrick[2] and FranÃğois-Xavier Pineau[3]
[1]*European Southern Observatory (ESO), Karl-Schwarzschild-Str., 85748 Garching, Germany*
[2]*Physics Department, Lancaster University, Bailrigg, Lancaster, LA1 4YW, UK*
[3]*UniversitÃł de Strasbourg, CNRS, Observatoire astronomique de Strasbourg, UMR 7550, F-67000 Strasbourg, France*



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

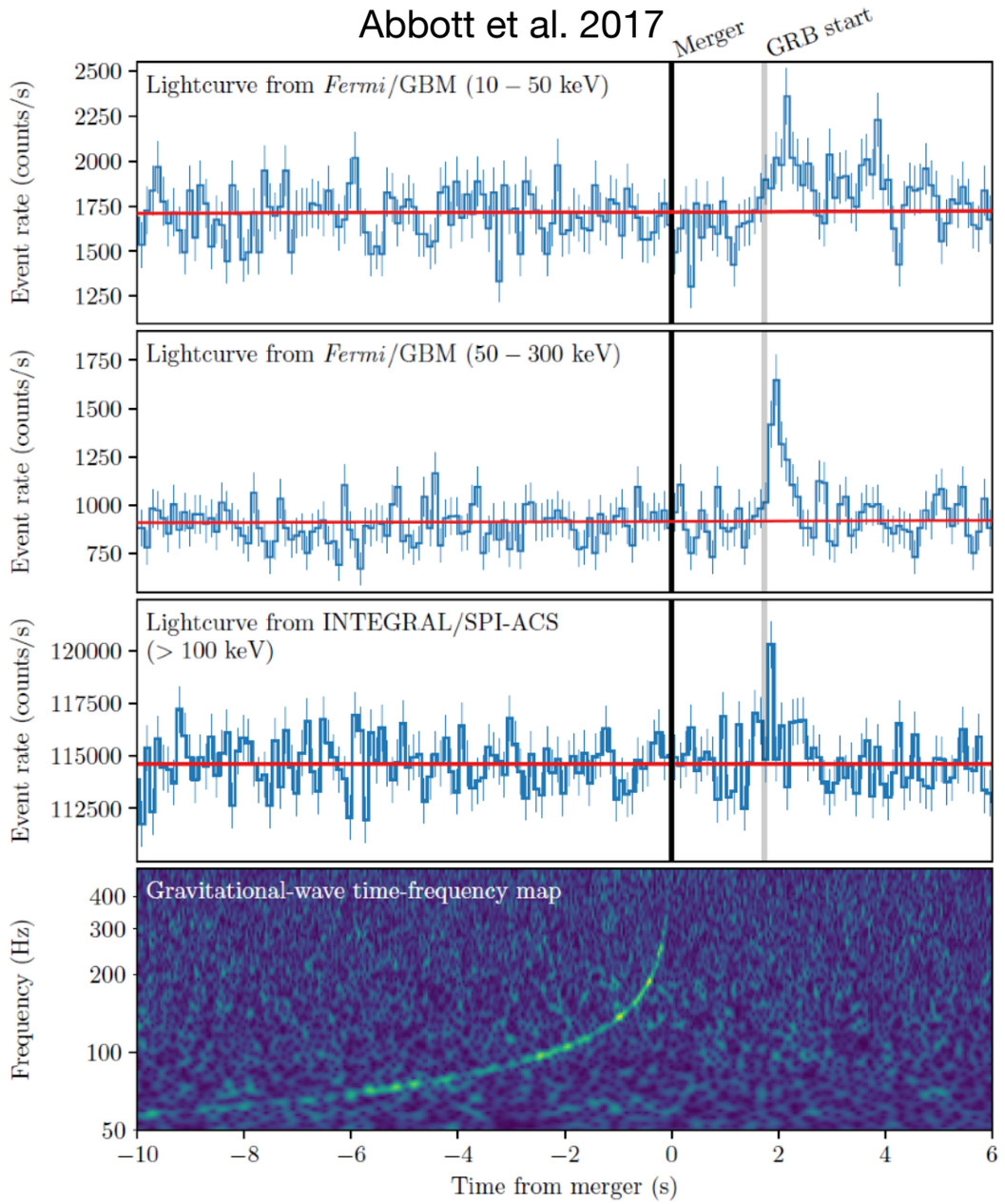
Increase in the data information content ⇒ Data-driven vs hypothesis driven science

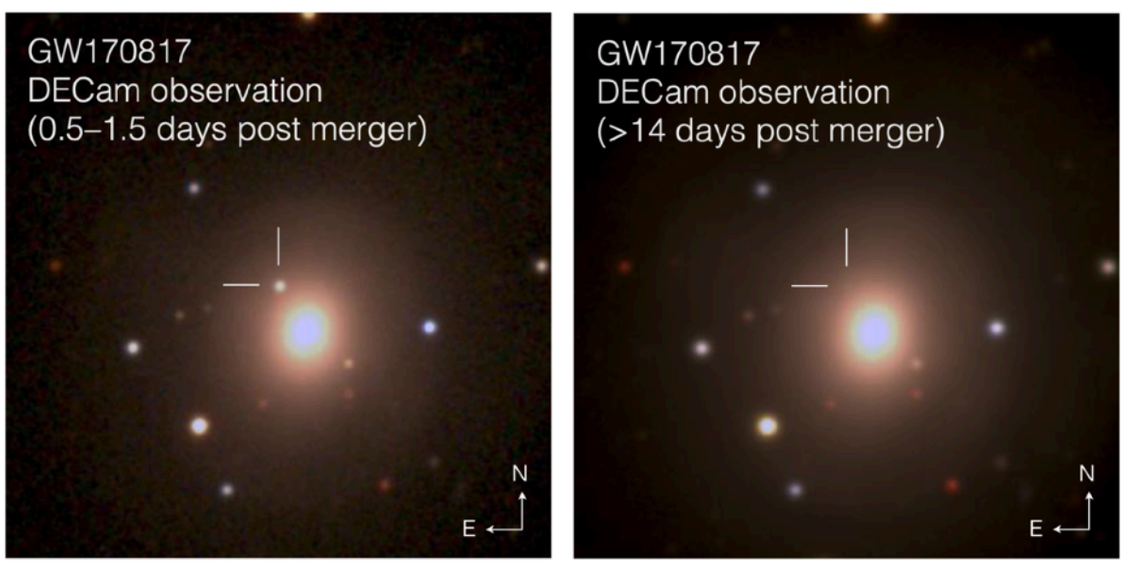
Increase in the information complexity ⇒ There are patterns in the data that cannot be comprehended by human directly



Credit: EHT Collaboration

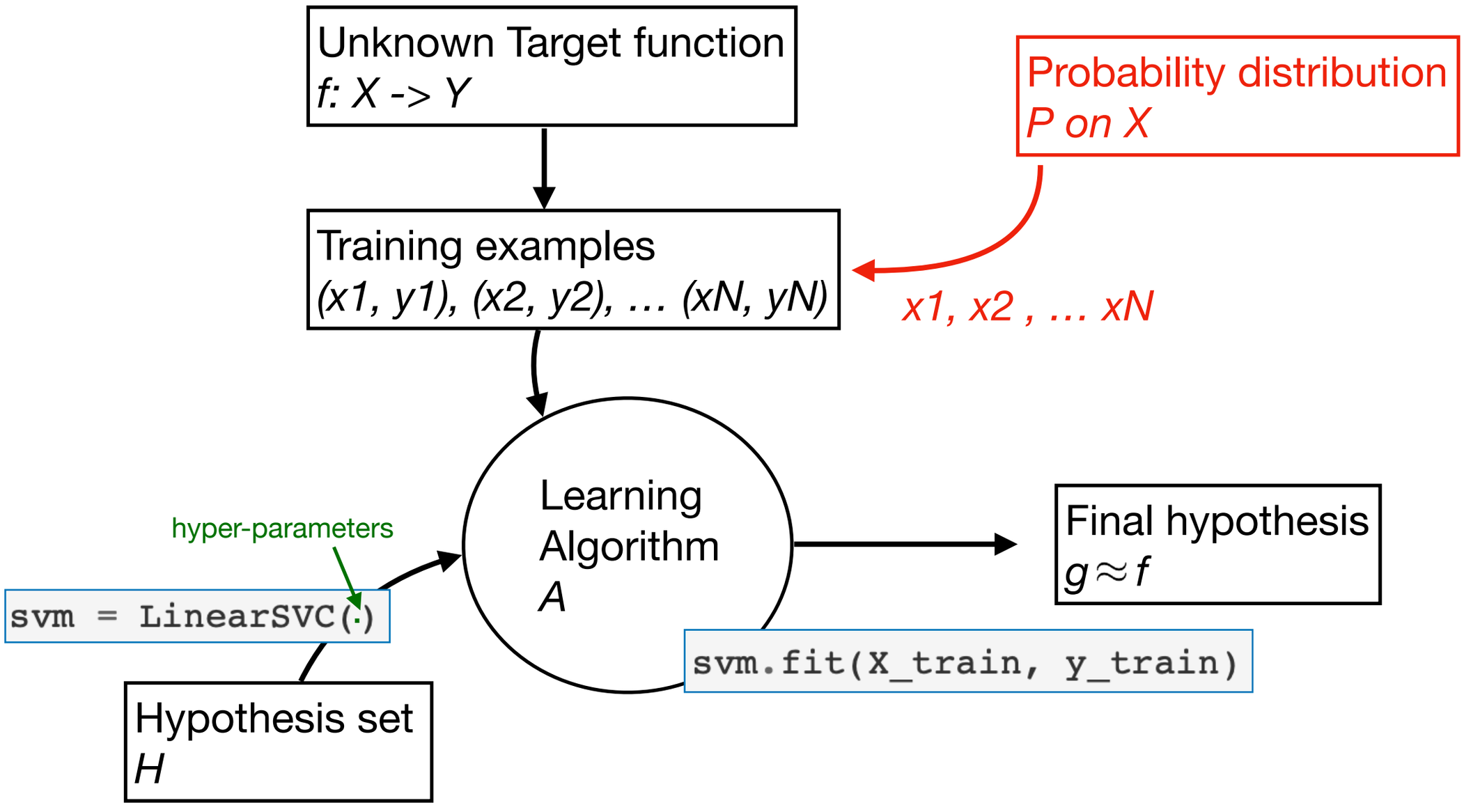
Abbott et al. 2017





Soares-Santos et al. 2017

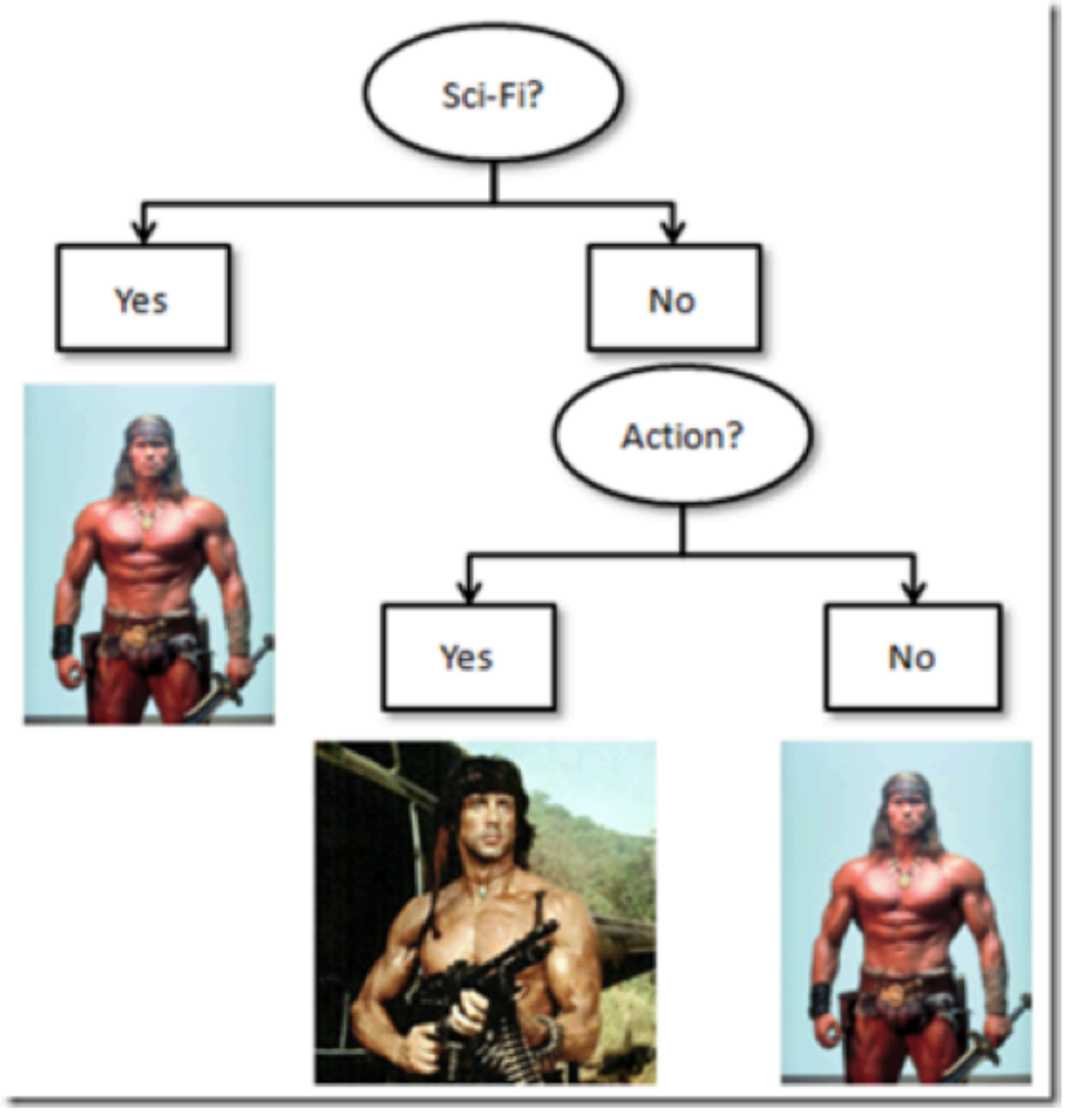
Learning Diagram

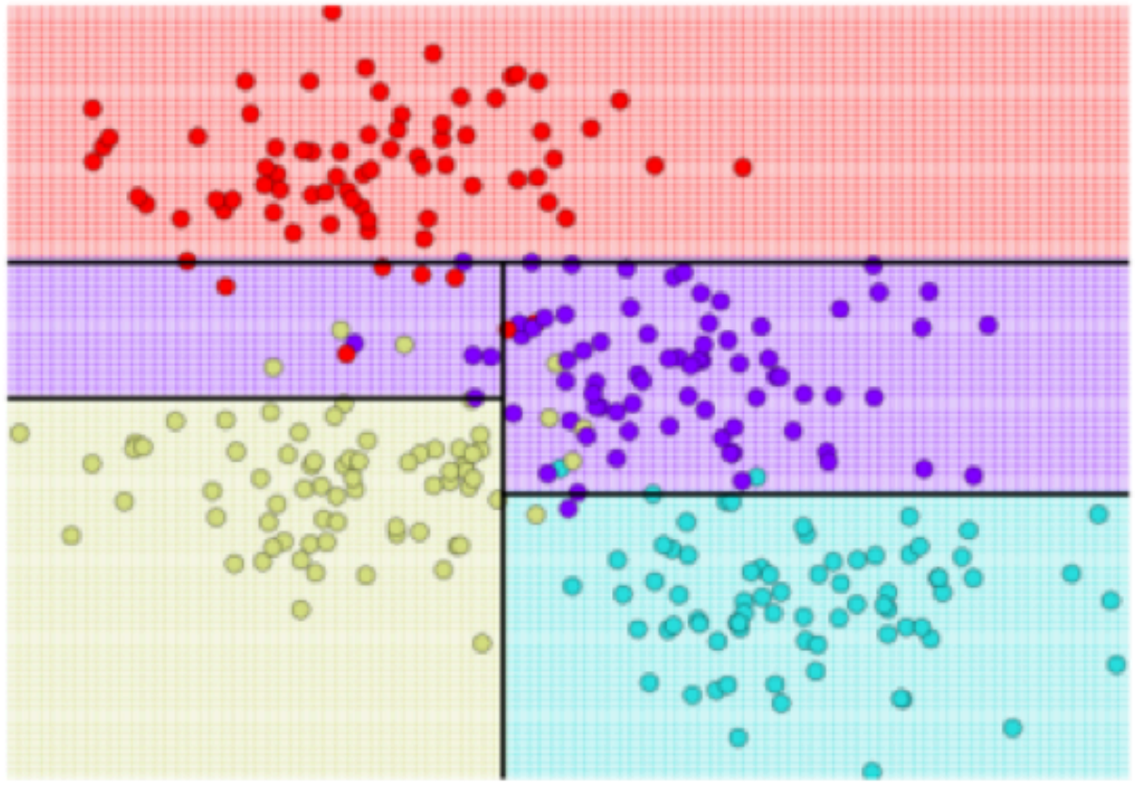


More advanced algorithms

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

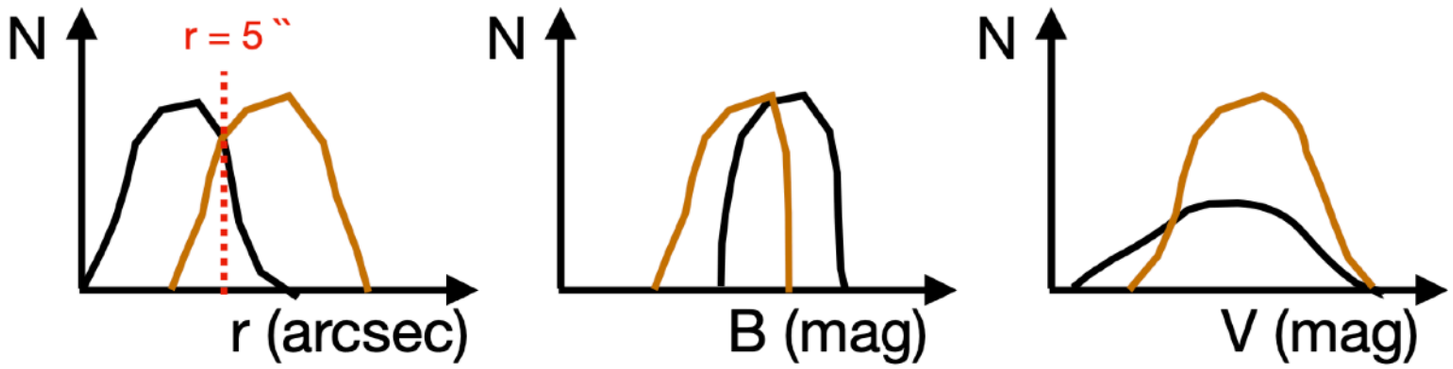
Random Forest (Ensemble of Decision Trees)

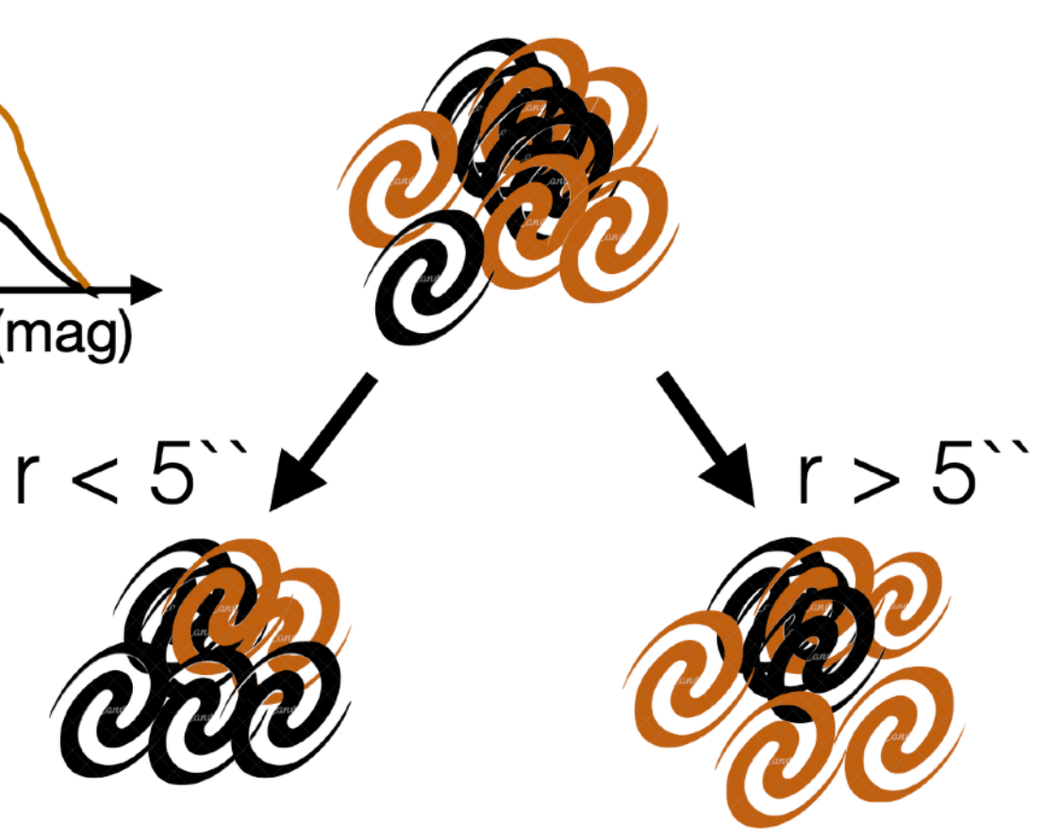




Construction of Decision Tree

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

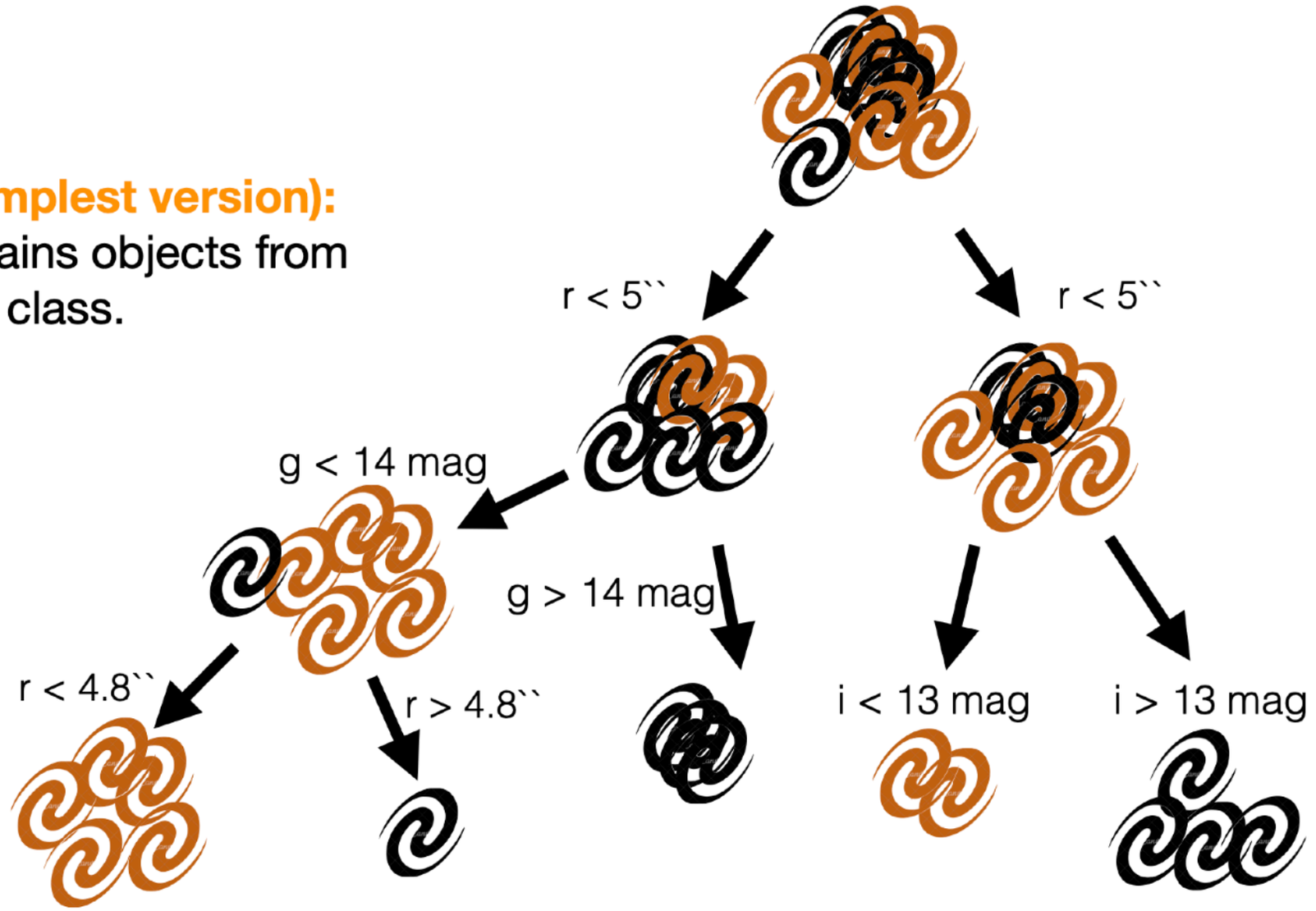




Construction of Decision Tree

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

Stop criterion (simplest version):
each terminal contains objects from a single class.



$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)

= Entropy (0.36, 0.64)

= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)

= 0.94

$$Information\ Gain\ =\ Entropy(before) - \sum_{j=1}^{K} Entropy(j,\ after)$$

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Advantages:

(1) Non-linear model, which is constructed during training.
(2) In its simplest version, very few free parameters.
(3) Handles numerous features and numerous objects.
(4) No need to scale the feature values to the same “units”.
(5) Produces classification probability (in its more complex version).
(6) Produces feature importance.

Disadvantages:

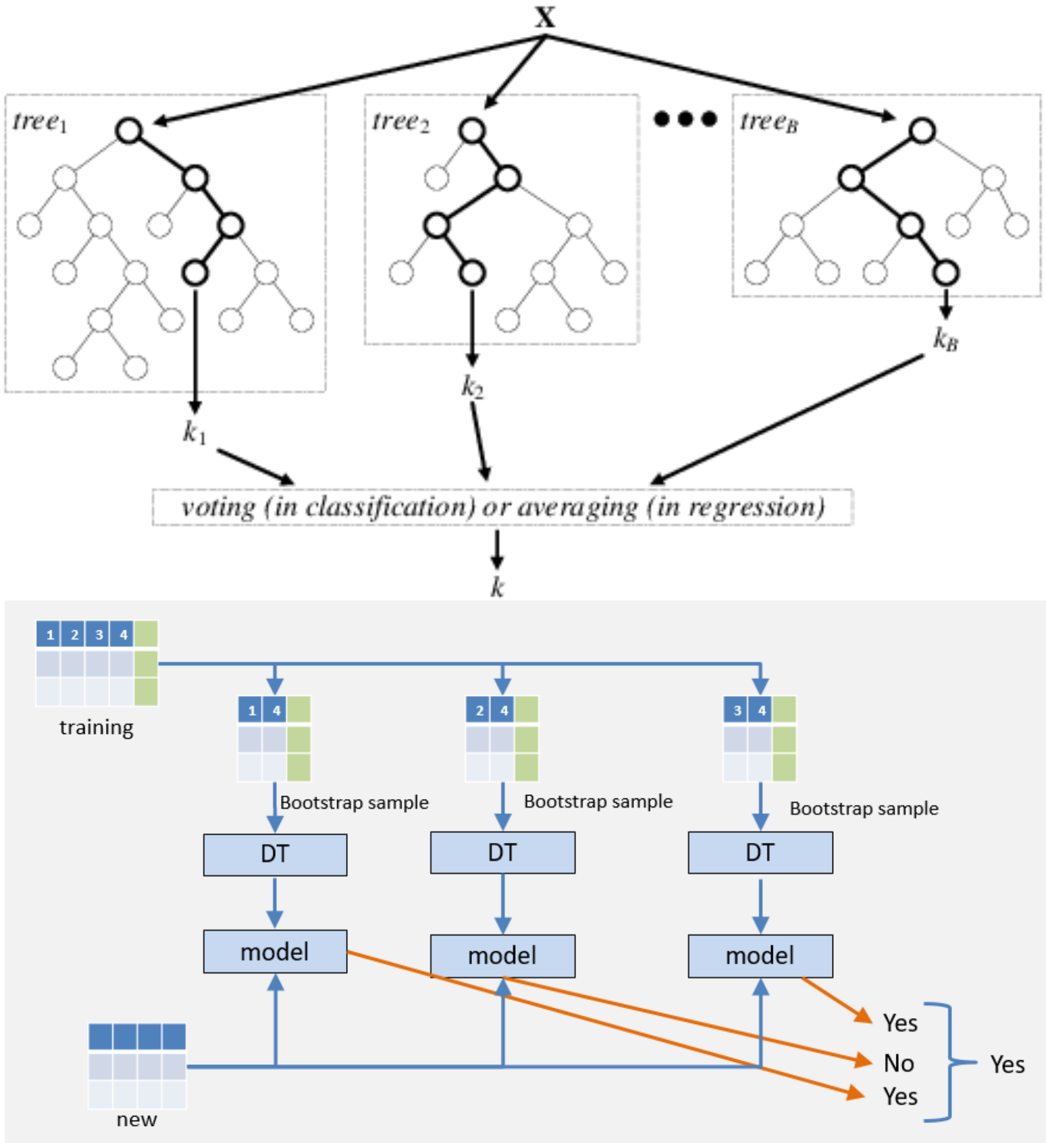
(1) Tends to overfit!

Random Forest

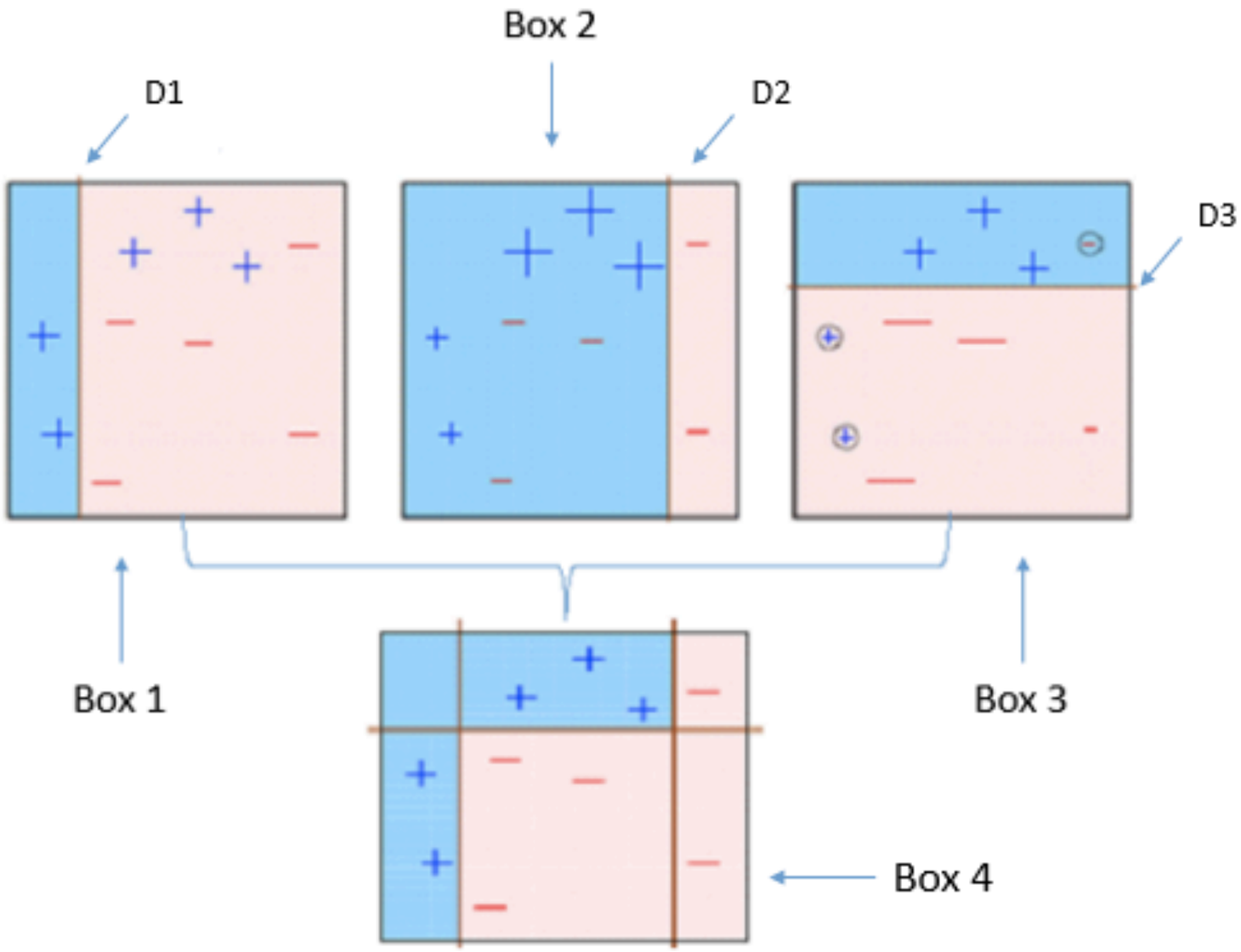
=

collection of randomised trees

(random data subsample
"bootstrapping",
random feature "de-correlate")



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

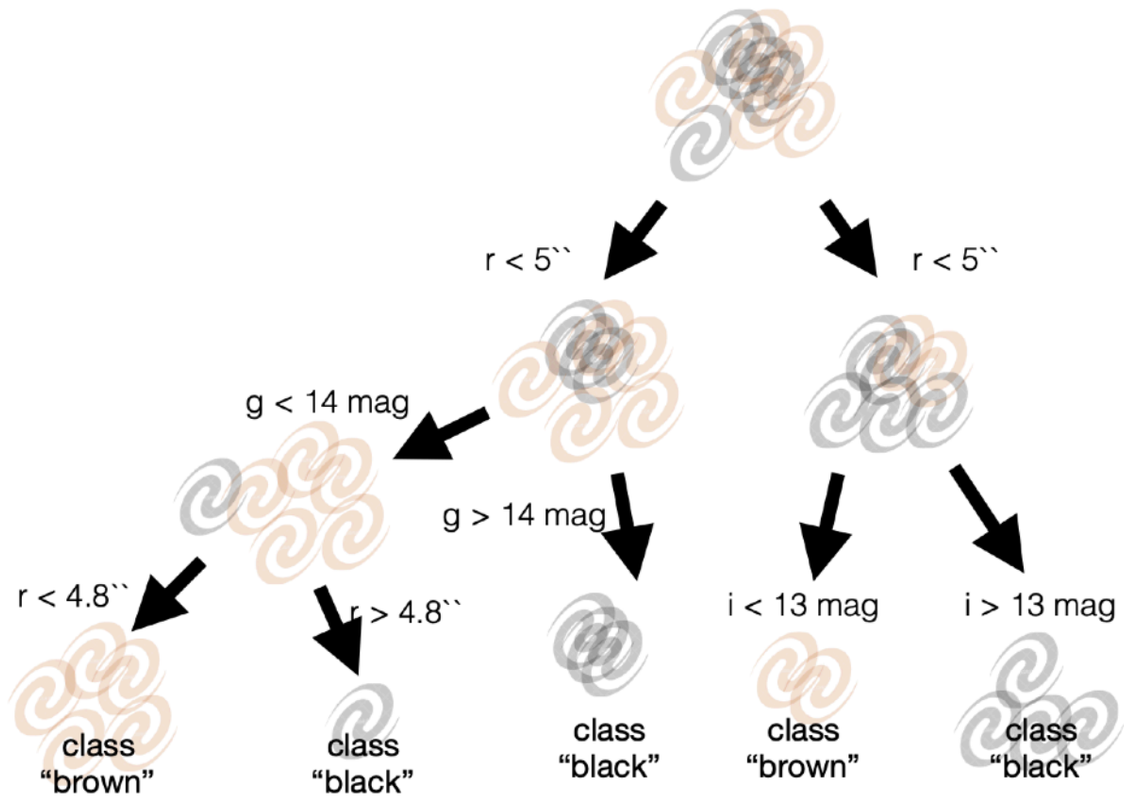


Feature importance & feature selection

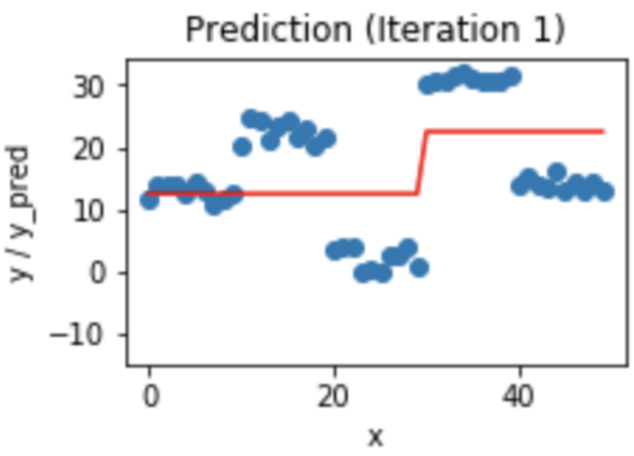
DT: The higher a feature is in the tree, the more important it is for classification.

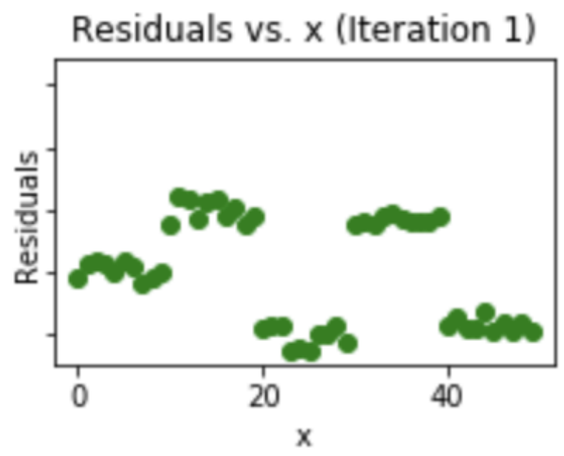
RF: Omitting a feature makes separates data faster, feature has low importance and vice versa.

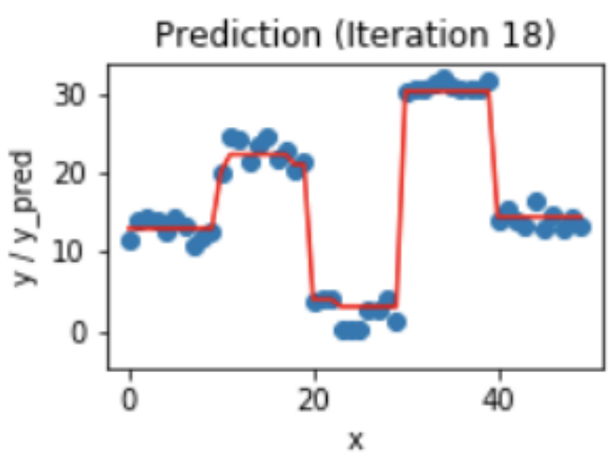
Trick: add a non-informative dummy feature, e.g. random or a constant. If your physical feature is ranked lower, remove it.

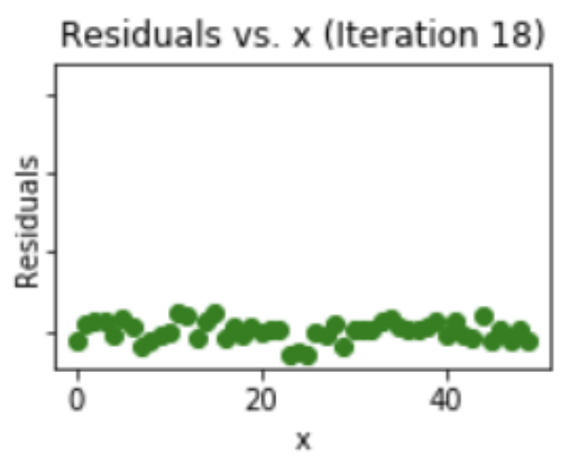


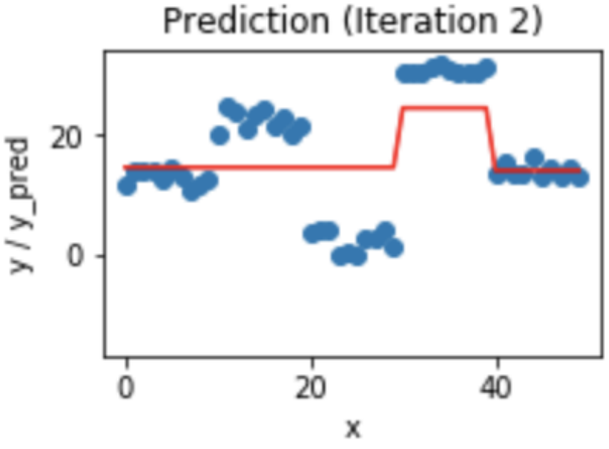
Gradient Boosted Trees

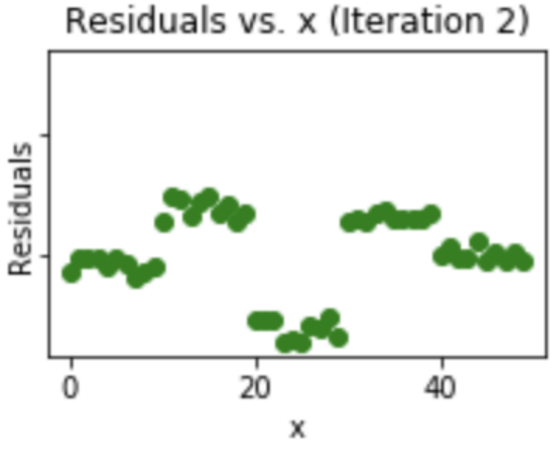


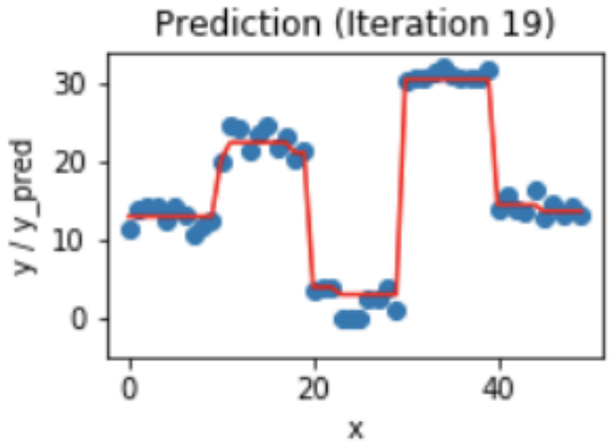


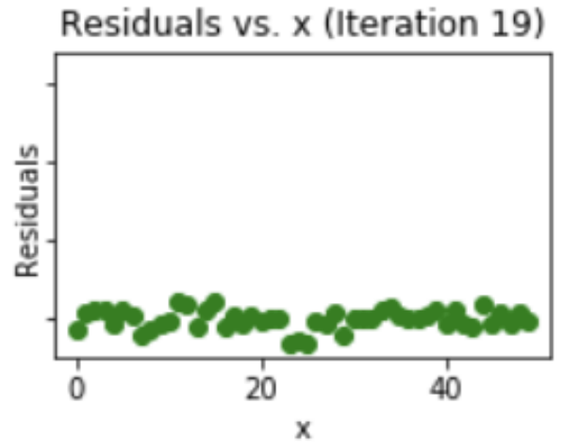


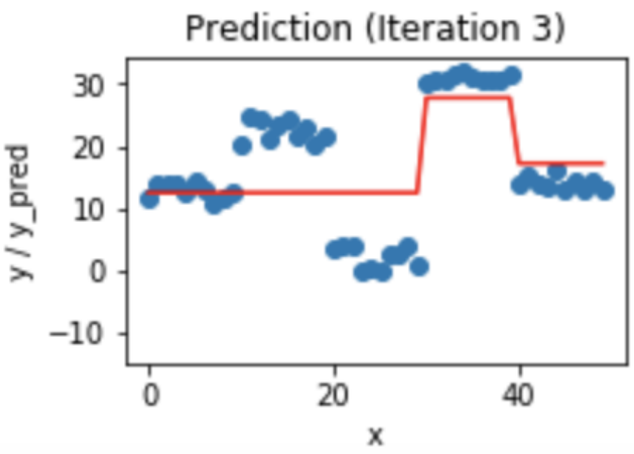


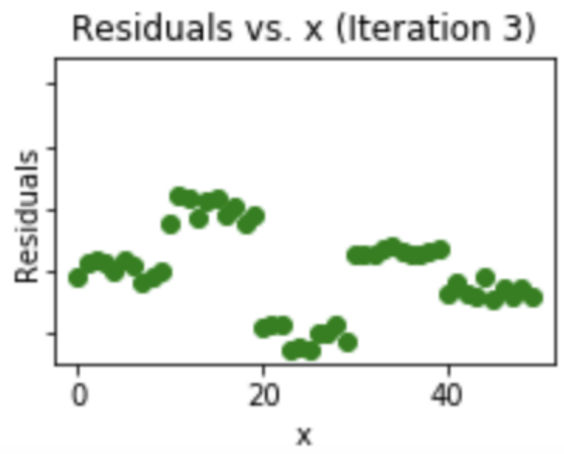


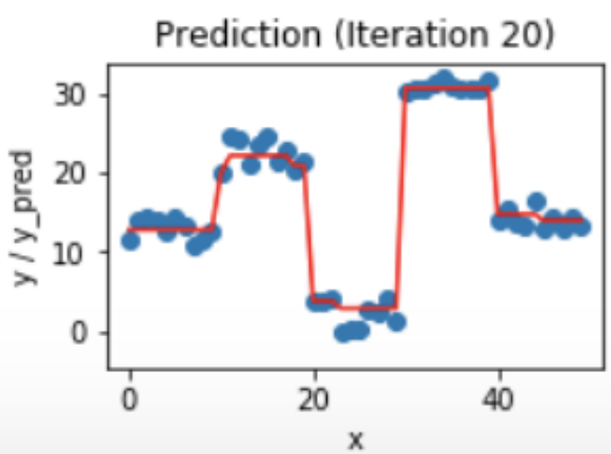


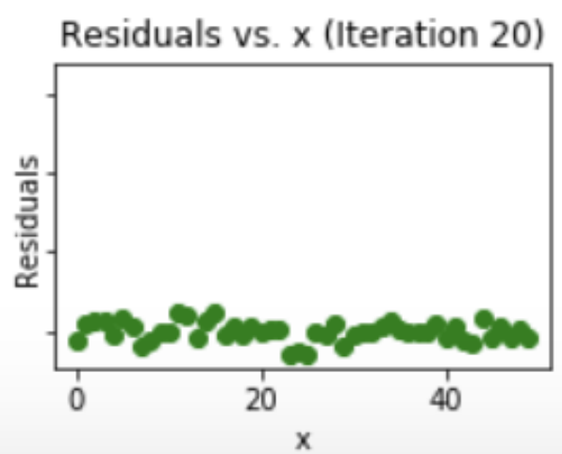












```
# Dataset X, Y

mu = mean(Y) # often starts with constant "mean" predictor
dY = Y - mu
for k=1..N:
  Learner[k] = Decision_Tree_Regressor(X, dY)
  alpha[k] = 1 # "learning rate" or "step size"

  # compute residuals
  dY = dY - alpha[k] * Learner[k].predict(X)

# test how it works
predict = zeros(len(Xtest))
for k=1..N:
  predict = predict + alha(k)*Learner[k].predict(Xtest)
```

kind of gradient descent

- weighted combination of weak learners (often stumps), built sequentially
- in random forest each tree has the same weight, in boosted trees not

Decision Tree

Random Forest

AdaBoost

.75

.82

.85

Decision Tree

Random Forest

AdaBoost

.82

.85

.85

Decision Tree

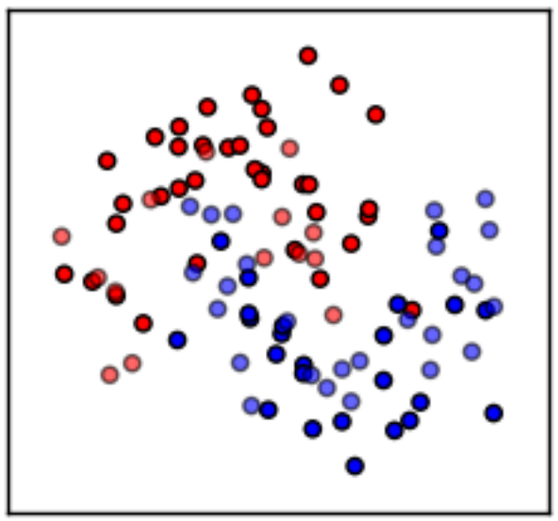
Random Forest

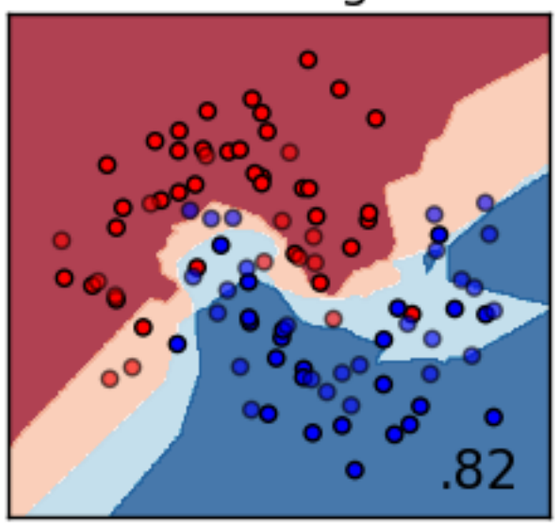
AdaBoost

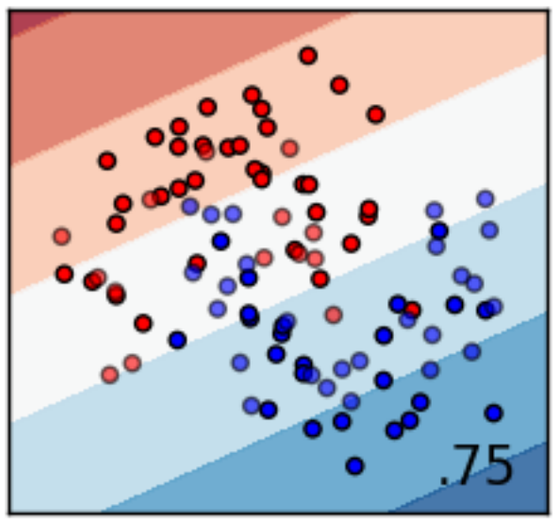
.88

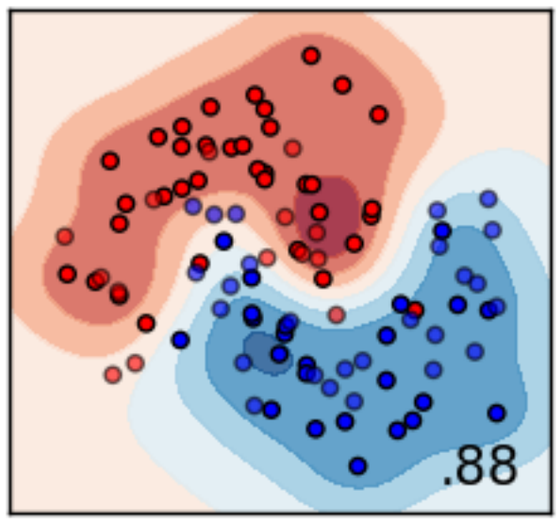
.88

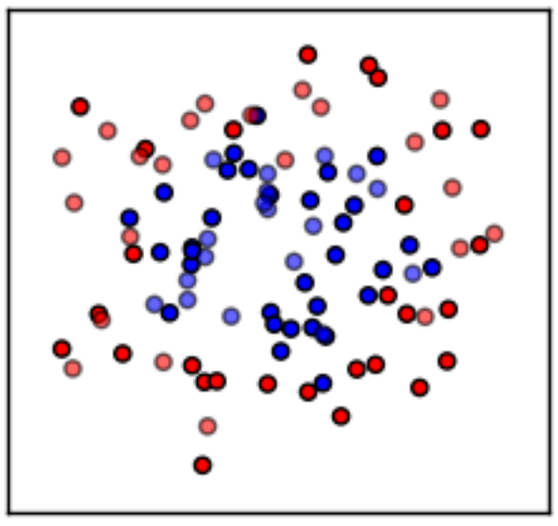
.90

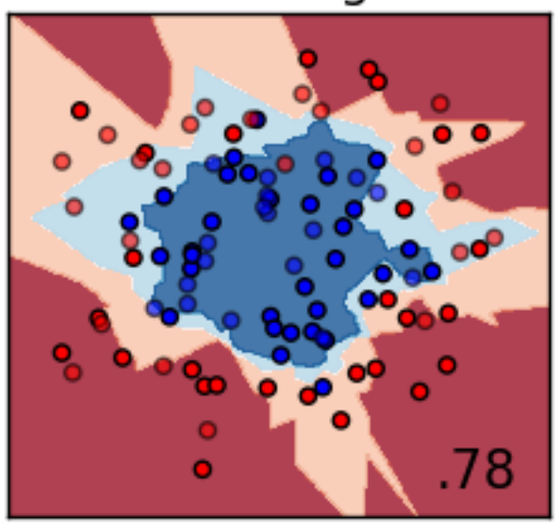


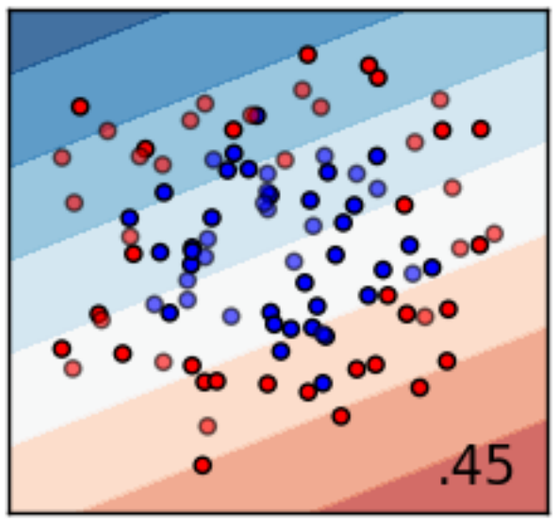

Nearest Neighbors
.82

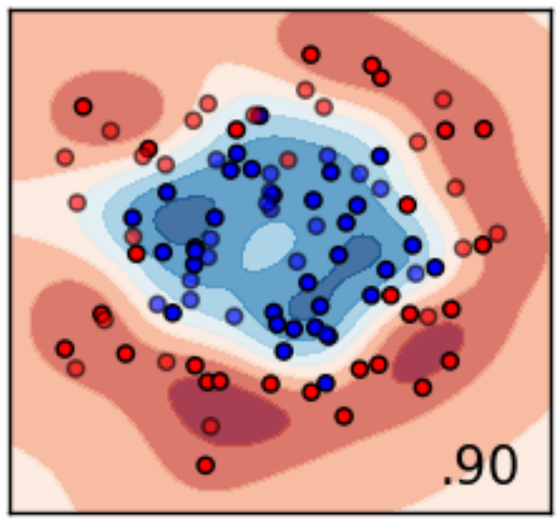

Linear SVM
.75

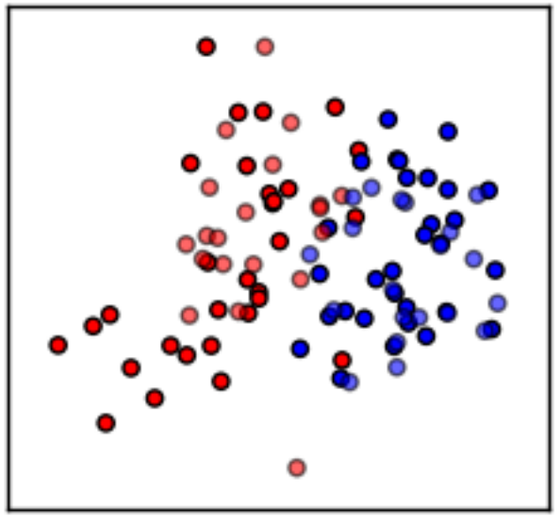

RBF SVM
.88

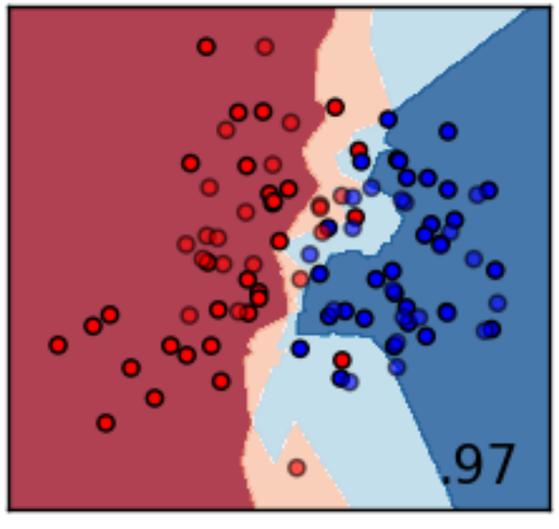


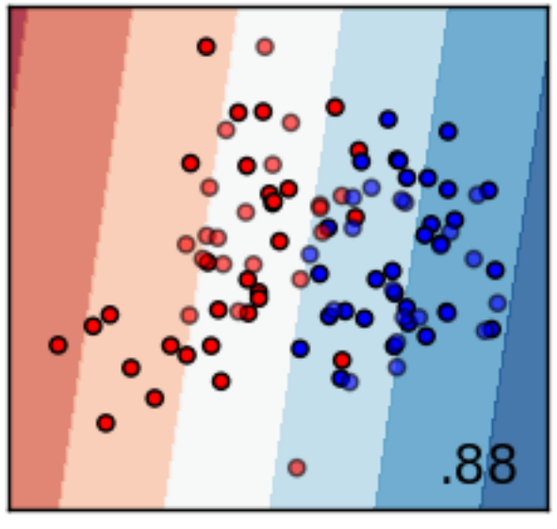

Nearest Neighbors
.78

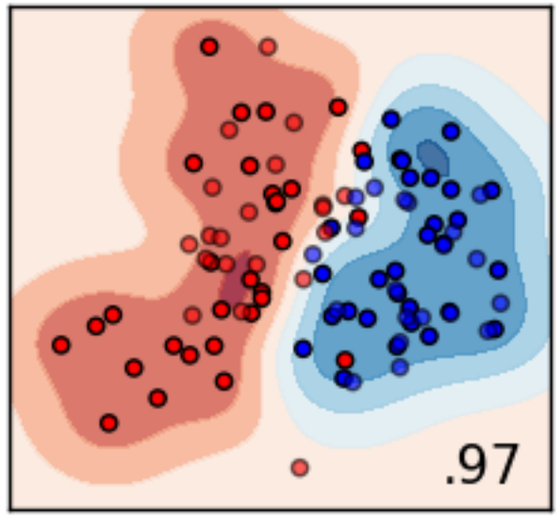

Linear SVM
.45


RBF SVM
.90




Nearest Neighbors
.97

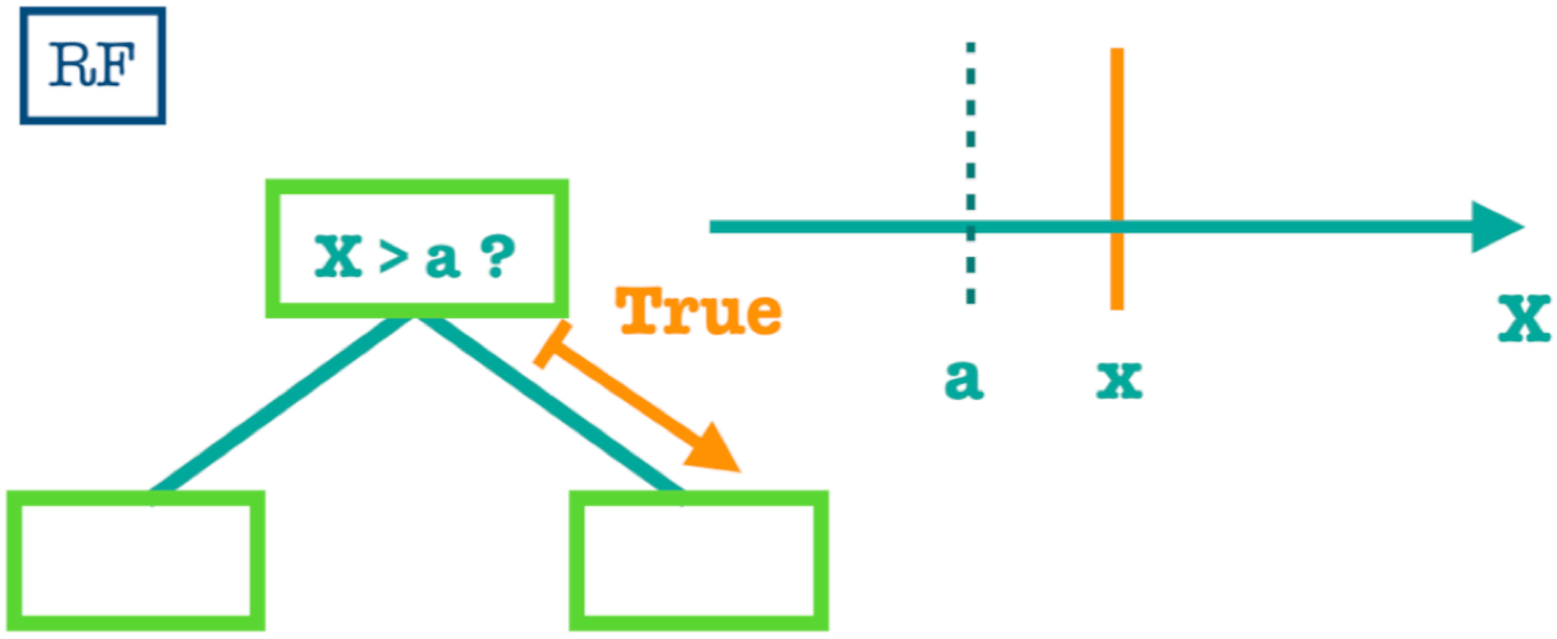

Linear SVM
.88


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

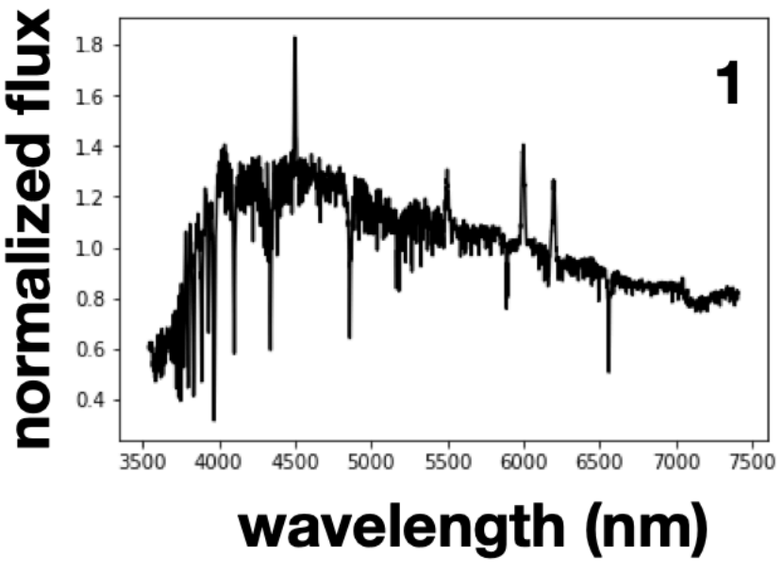


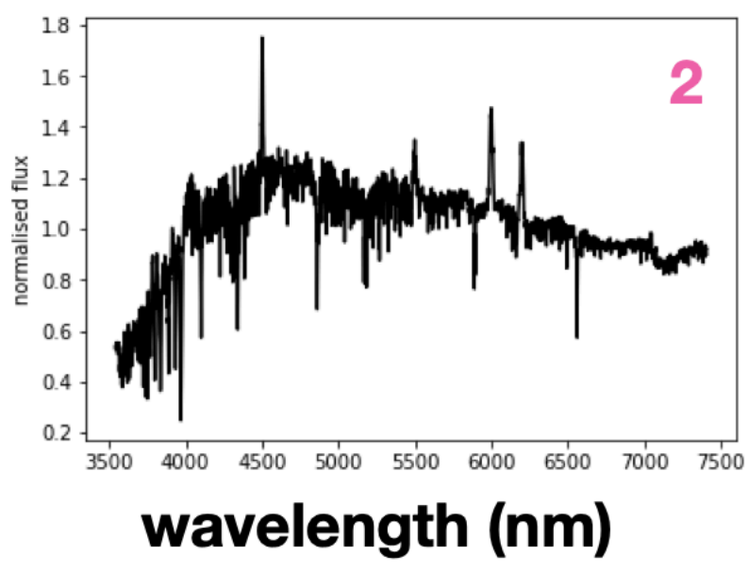
Reis et al. 2018

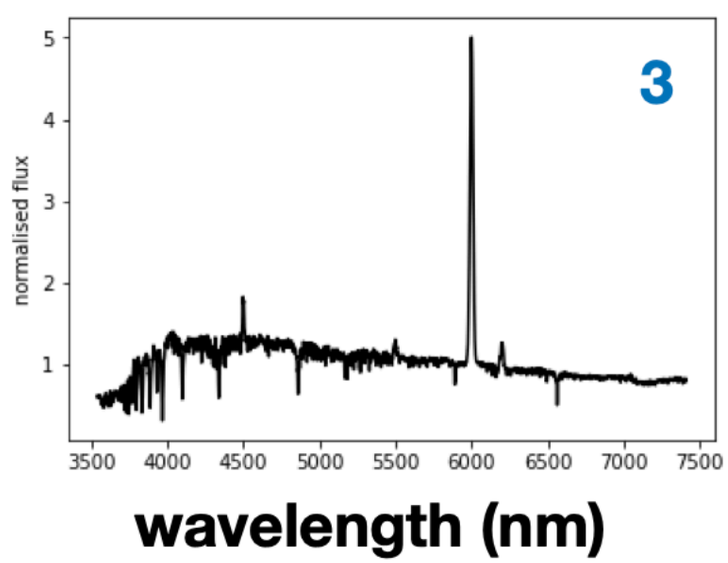
Unsupervised Random Forest

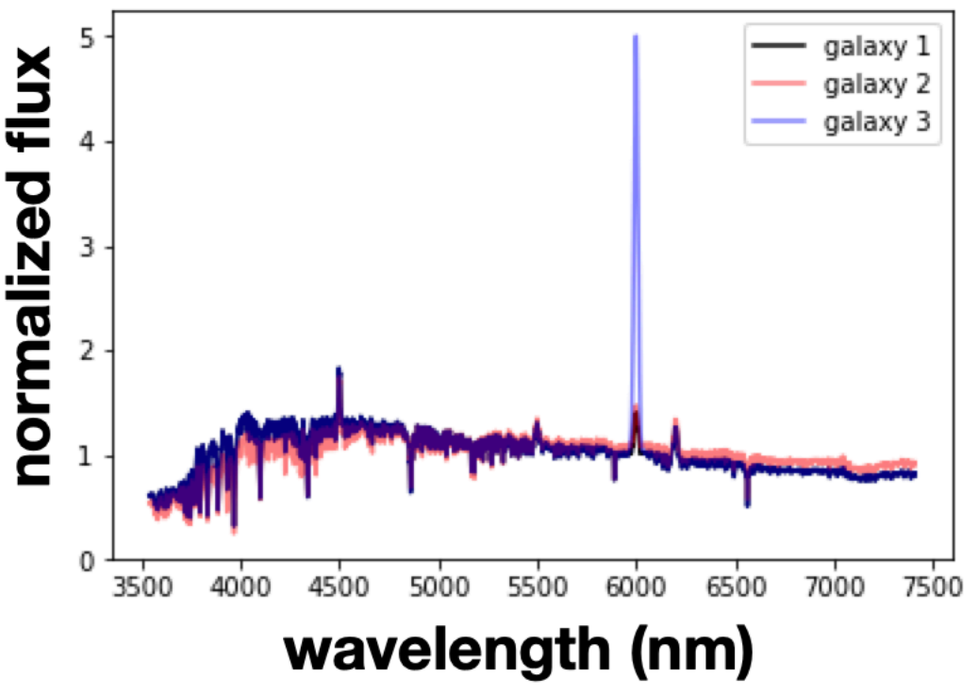
Random Forest can be used as an unsupervised algorithm, to produce pair-wise similarity for the objects in our sample.

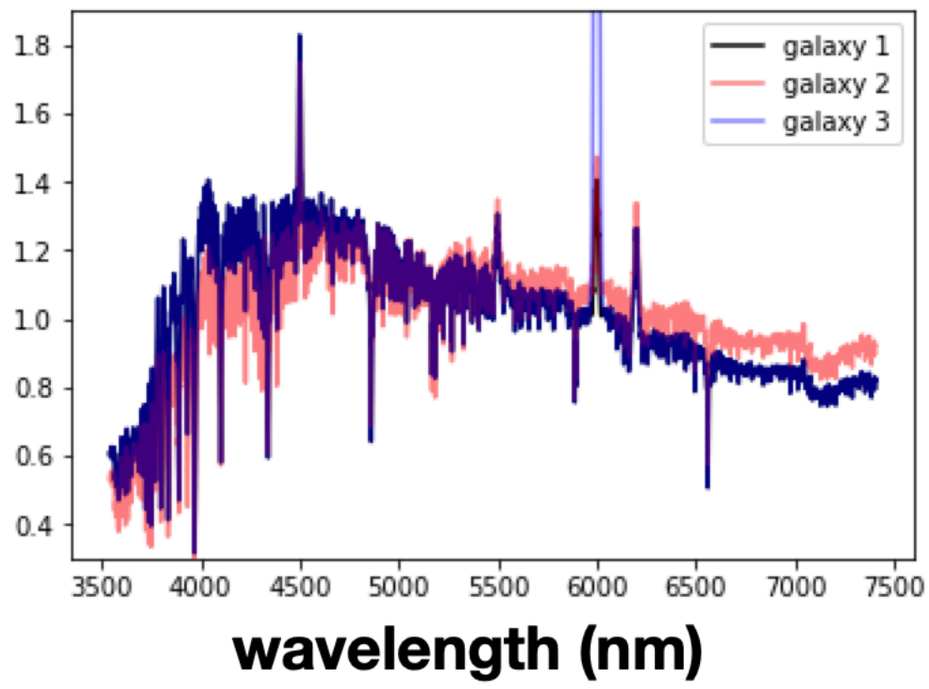
Why do we need to measure distances between objects?









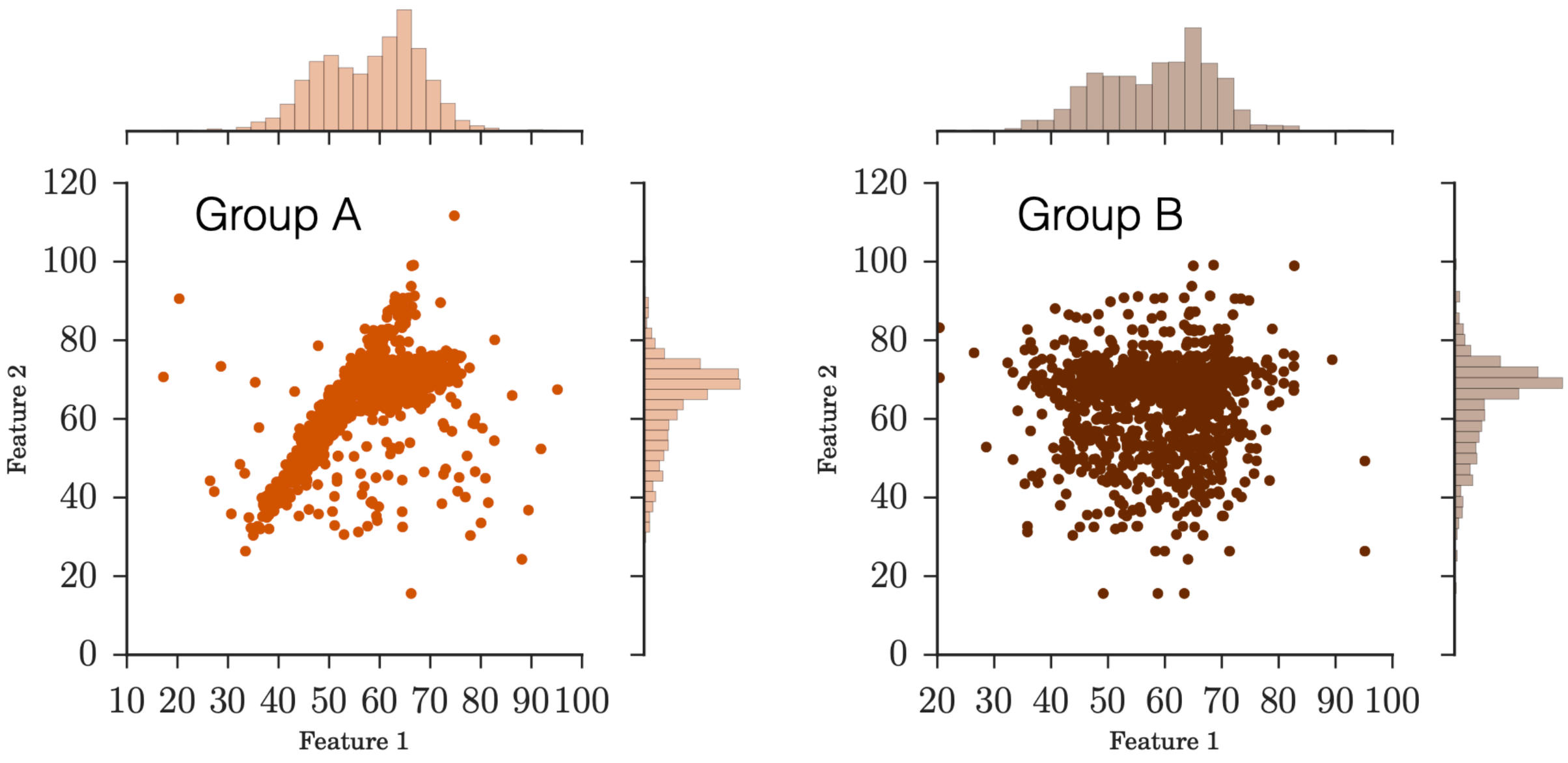


Unsupervised Random Forest

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!

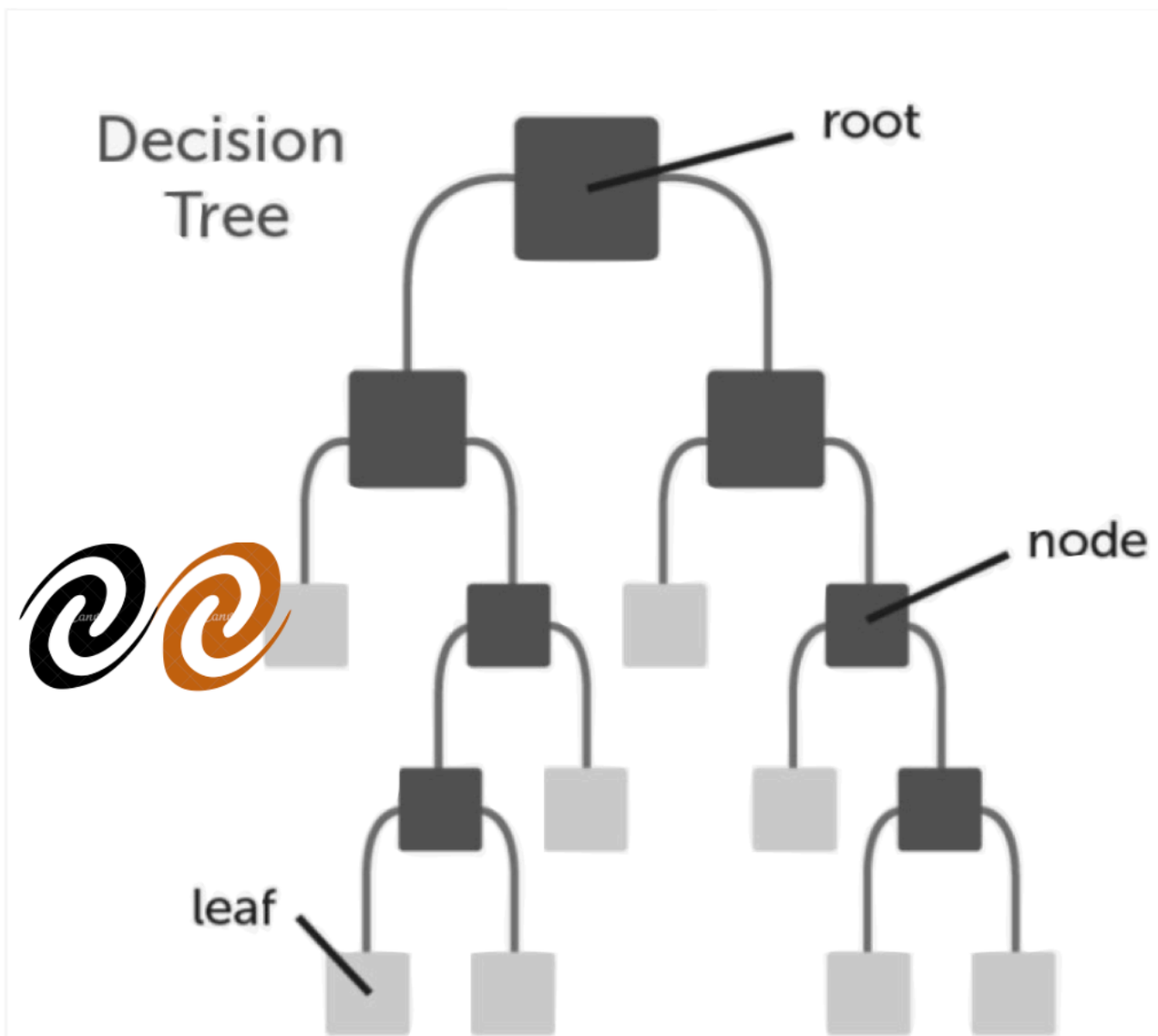
Random Forest is trained to distinguish between real and synthetic datasets.

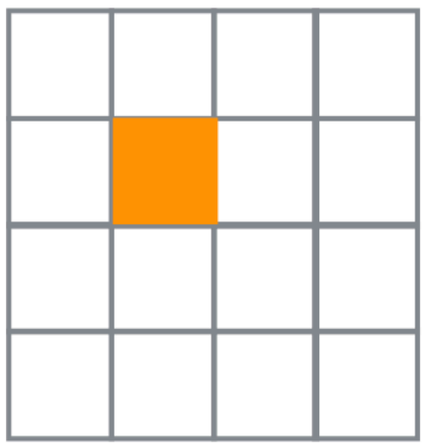


Unsupervised Random Forest

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.

"how many times spectrum[i] and spectrum[j] end up in the same node"





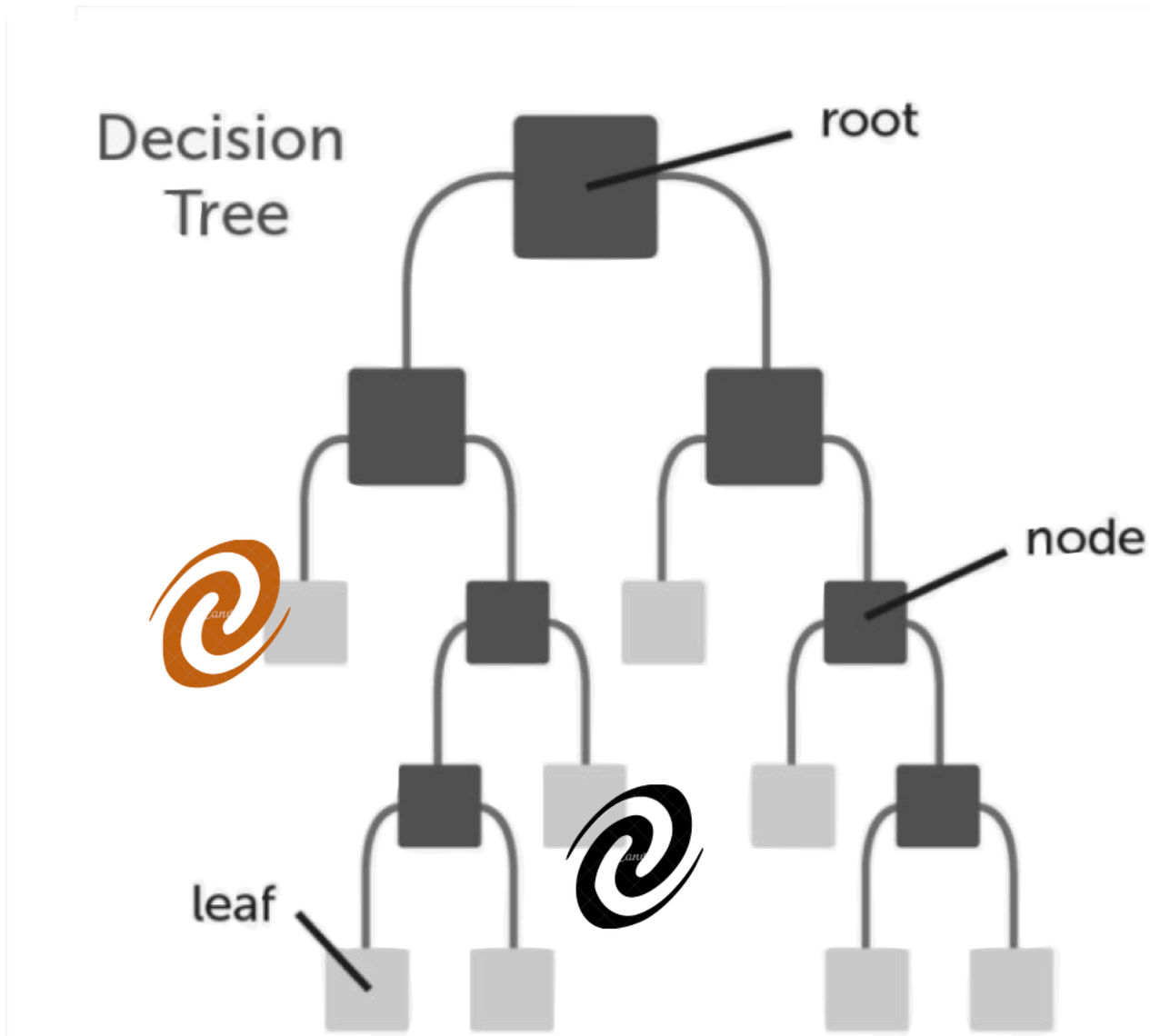
similarity matrix

similarity += 1

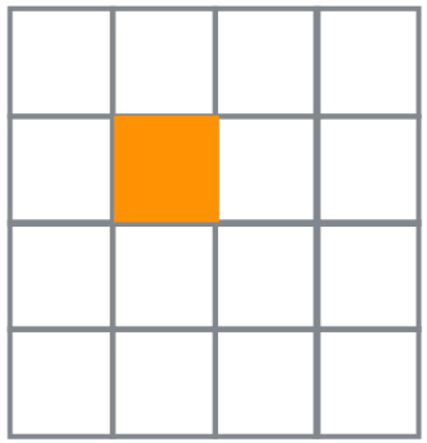
Unsupervised Random Forest

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.

"how many times spectrum[i] and spectrum[j] end up in the same node"



similarity += 0



similarity matrix

The process is repeated for all the trees in the forest.
Therefore, the similarity ranges from 0 to N, the number of trees in the forest.

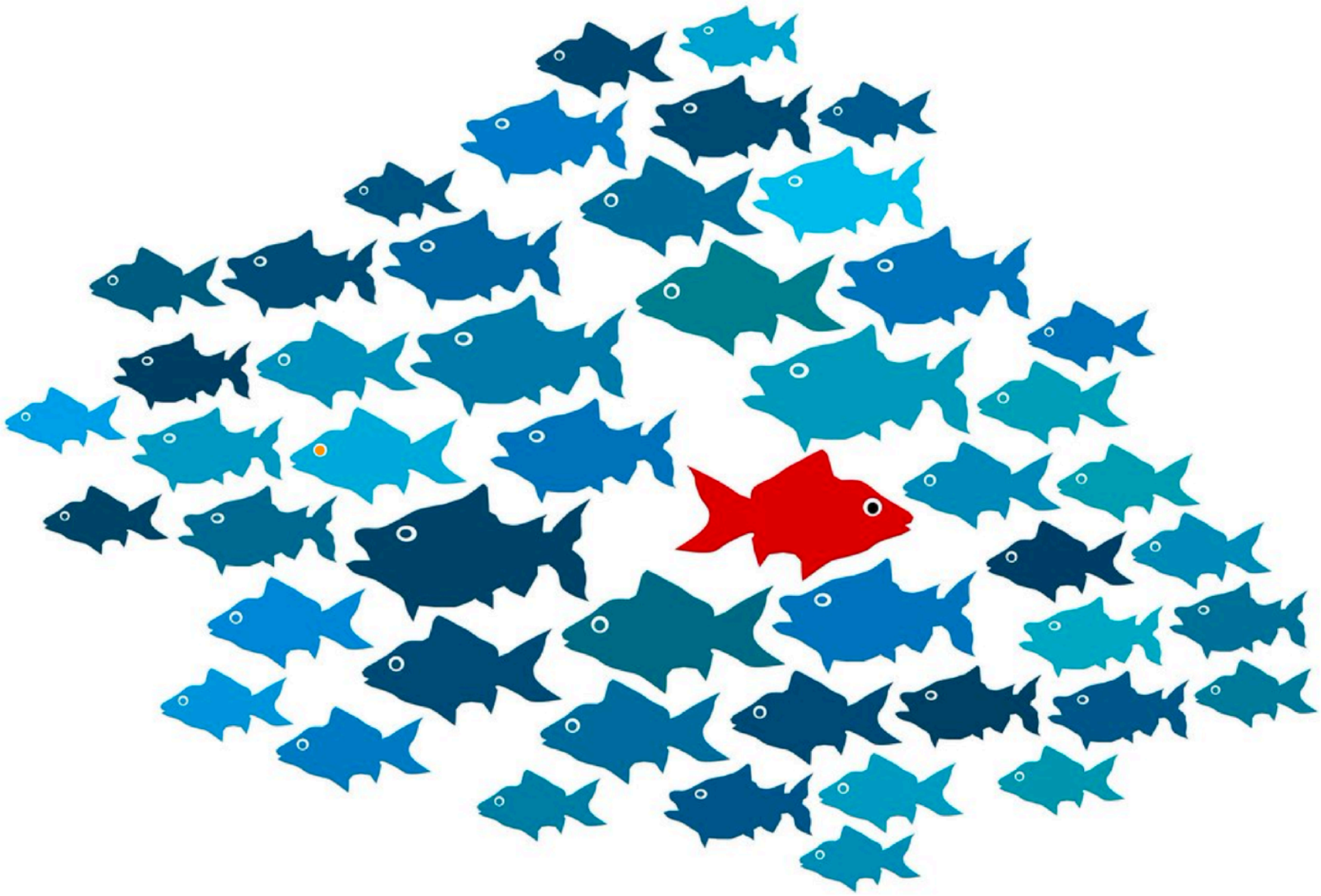
More details on unsupervised random forest use case:

The weirdest SDSS galaxies: results from an outlier detection algorithm

Dalya Baron[1⋆], Dovi Poznanski[1†]
[1] *School of Physics and Astronomy, Tel-Aviv University, Tel Aviv 69978, Israel.*

2016

Unsupervised learning (with metric) ⇒ outlier detection



Outliers

“Bad” object: artefacts, cosmic rays, bad reduction

Misclassified object: star classified as QSO, variable star classified as SN.

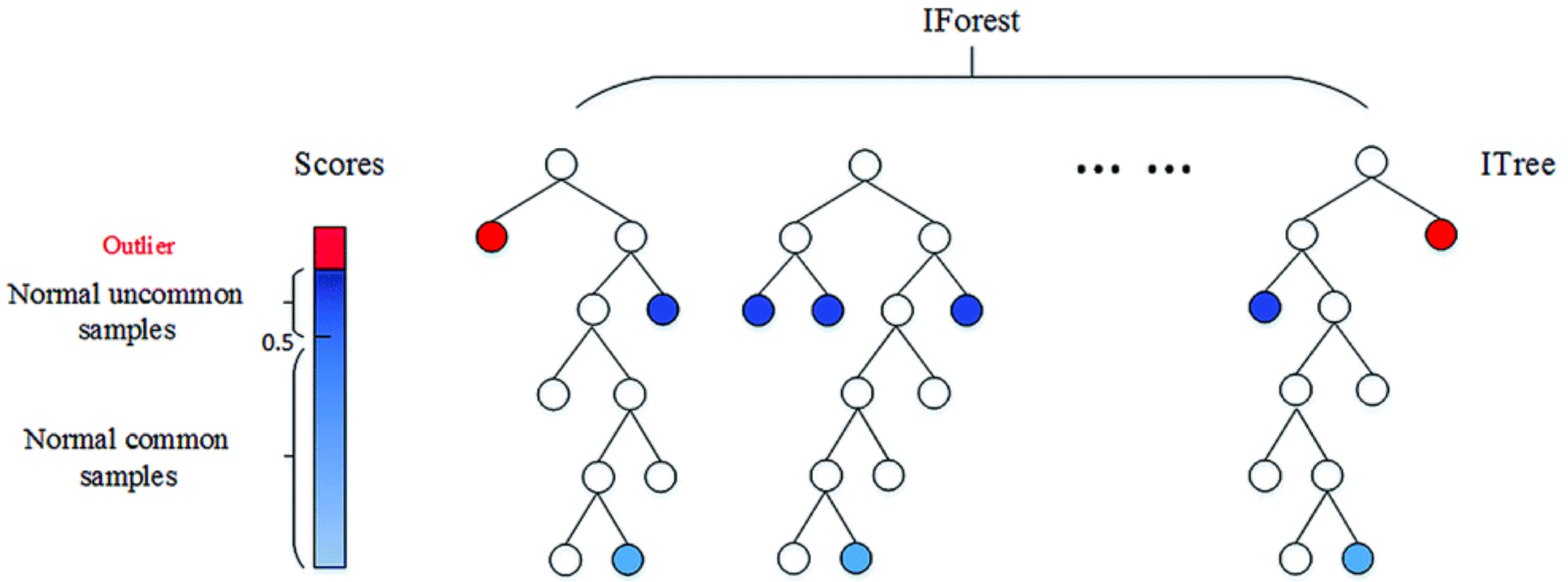
Tail of a distribution: most luminous SN, fastest accreting BH.

Unknown unknowns: completely new objects we did not know we should be looking for.

In astronomy: processes which happen on shorter time scales

Isolation forests

A fast algorithm that does not require distance measurements.
Outliers are objects that are separated from the rest of the dataset higher in the tree.



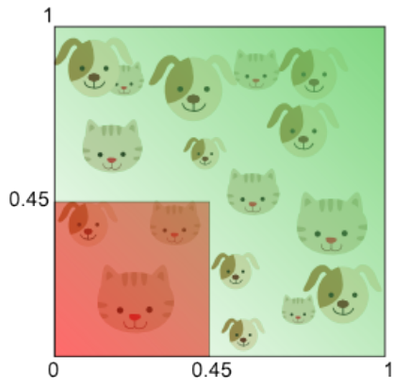
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

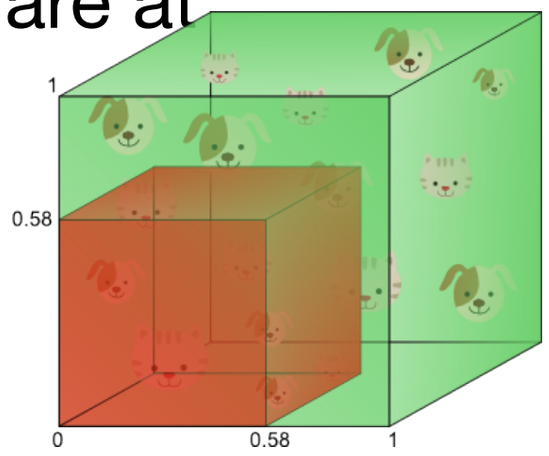
Curse of Dimensionality

- 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**

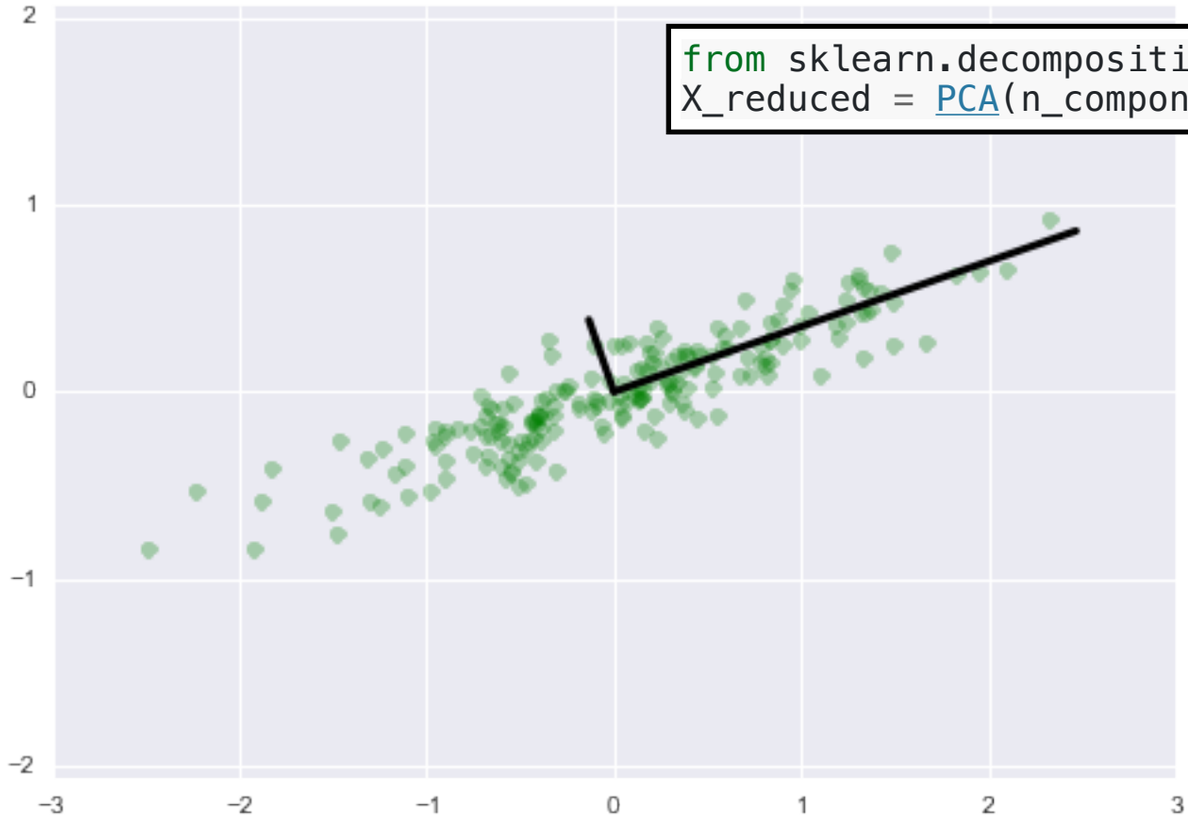
- *“If people can see in multi-dimensions we would not need machine learning”*
- More and more data needed to fill the same % of space
- Distance measure degenerates, most of the points are at the surface of a sphere (distance r)

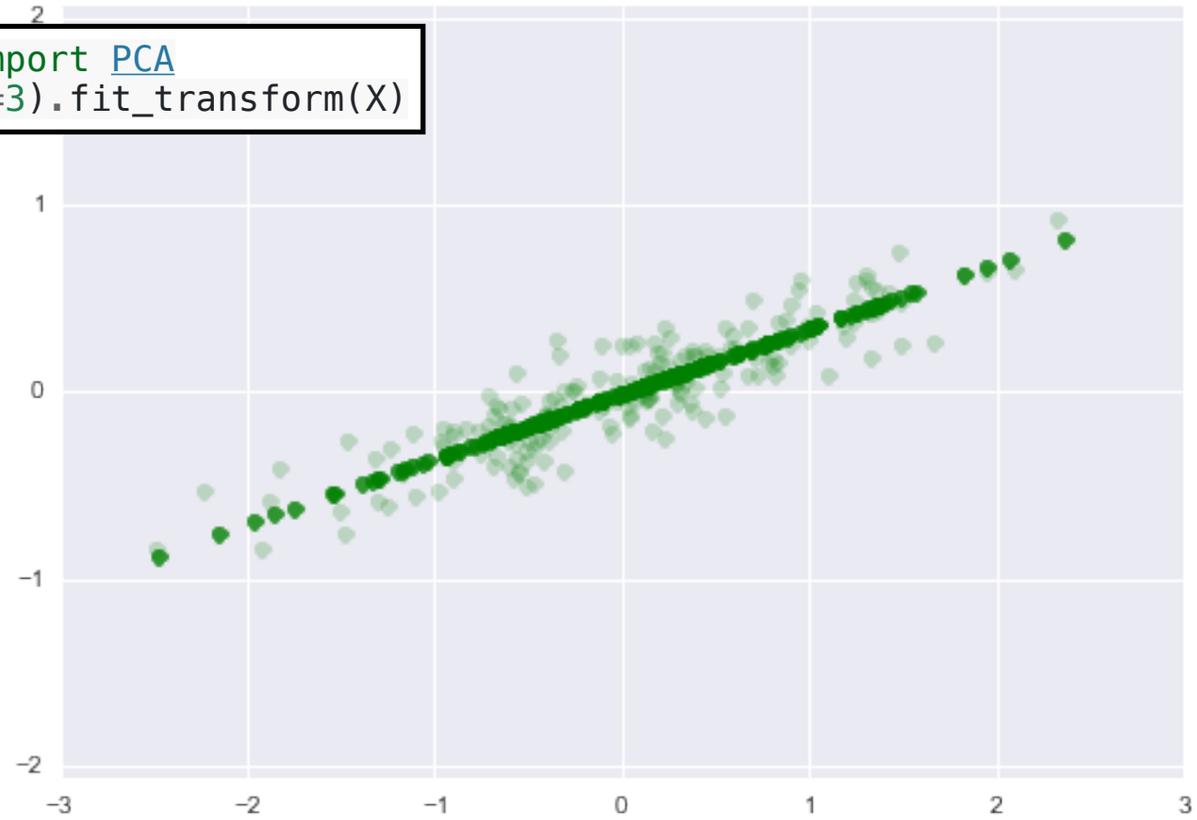
$$\frac{V_{\text{hypersphere}}}{V_{\text{hypercube}}} = \frac{\pi^{d/2}}{d2^{d-1}\Gamma(d/2)} \rightarrow 0 \text{ as } d \rightarrow \infty.$$





PCA (unsupervised)

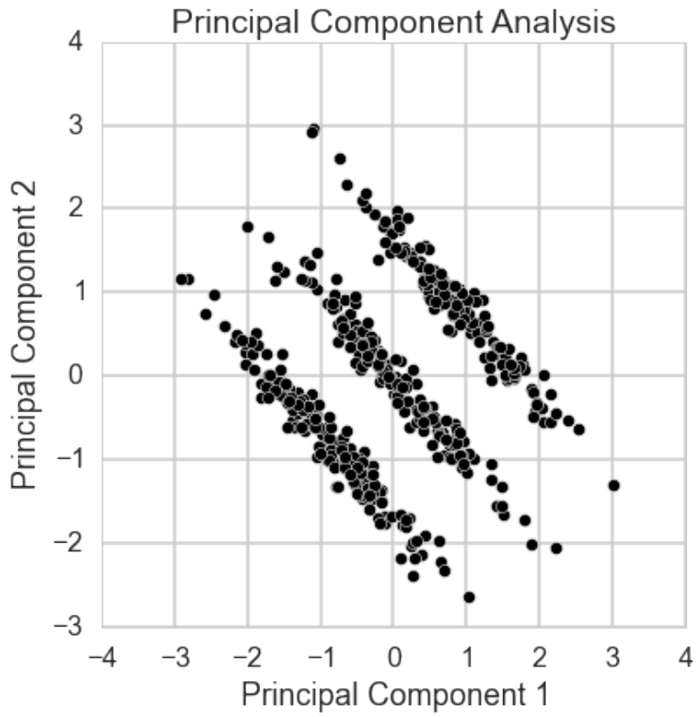




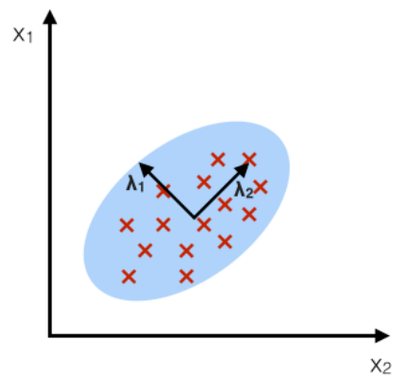
```
from sklearn.decomposition import PCA
X_reduced = PCA(n_components=3).fit_transform(X)
```

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

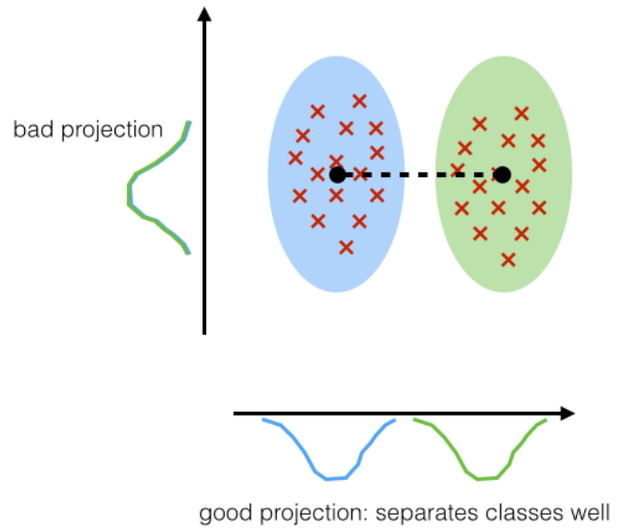
$$cov(x,y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N}$$



PCA:
component axes that maximize the variance

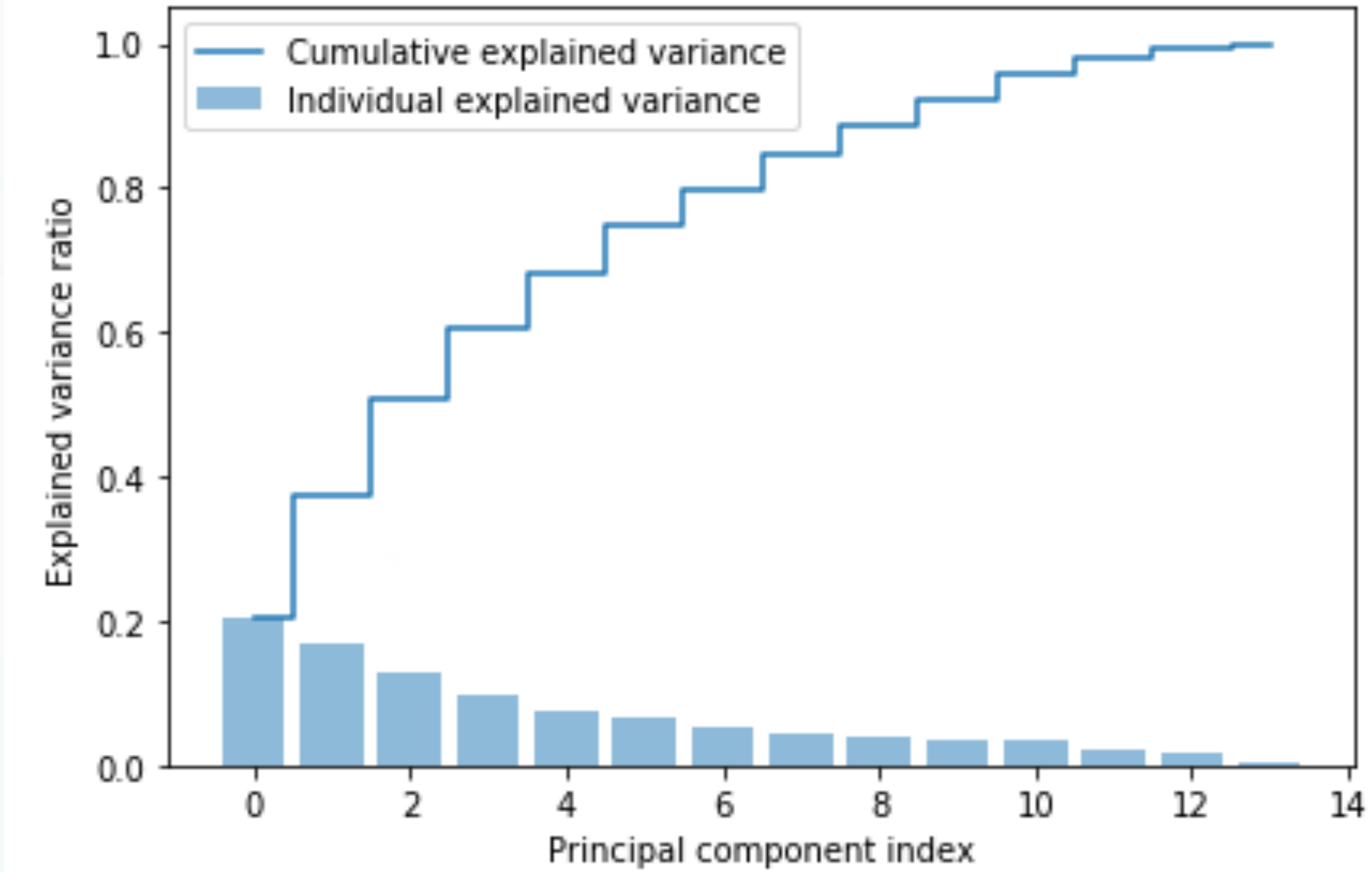


LDA:
maximizing the component axes for class-separation



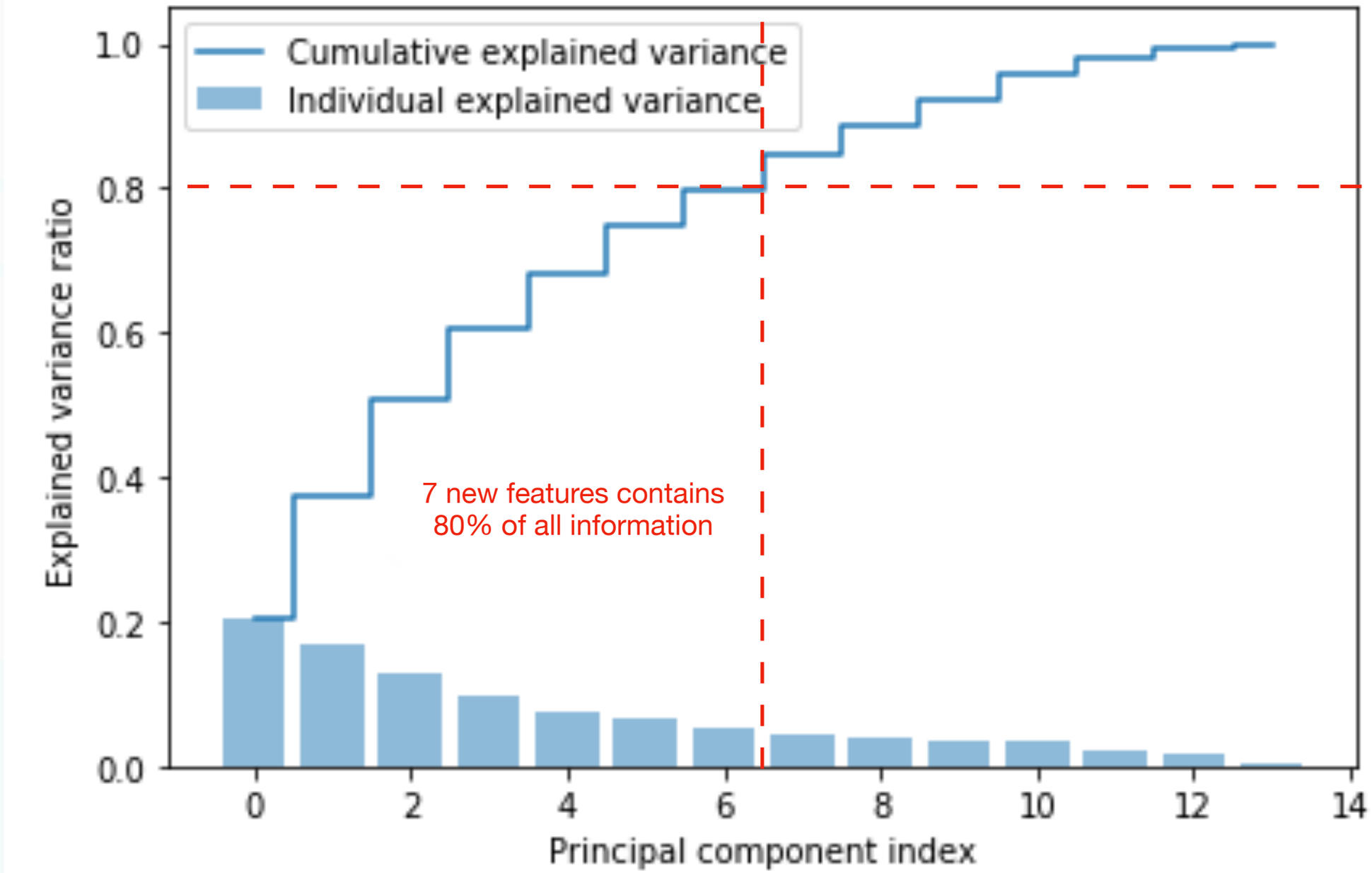
Principle Component Analysis (PCA)

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



Principle Component Analysis (PCA)

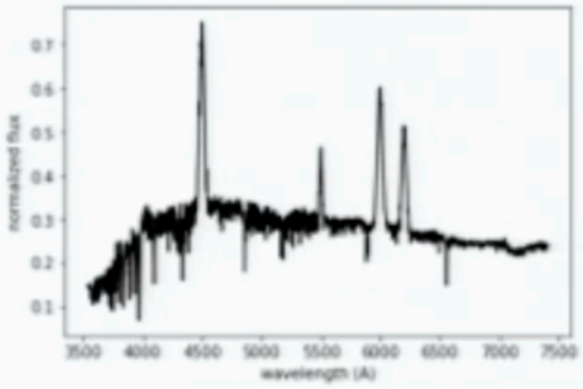
PCA allows us to compress the data, by representing each object as a projection on the first principle components.



Principle Component Analysis (PCA)

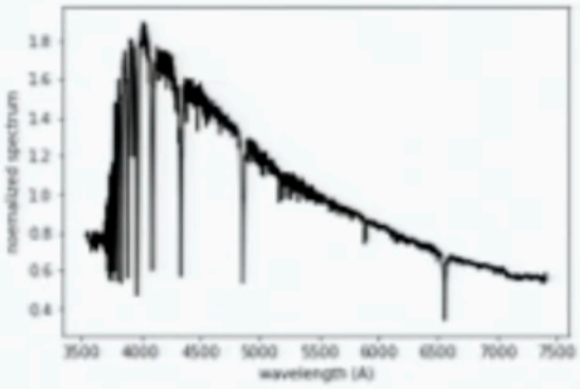
The principle components **may** represent the true **building blocks** of the objects in our dataset.

observed object



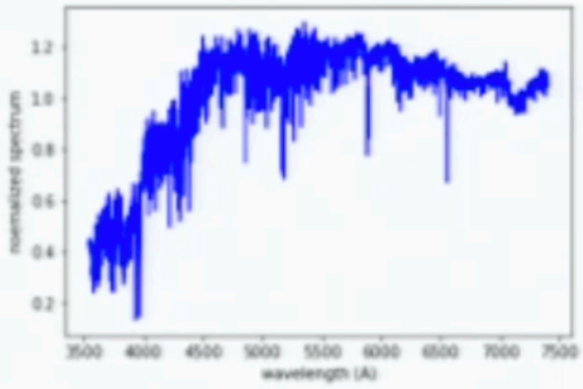
= A *

principle comp. 1



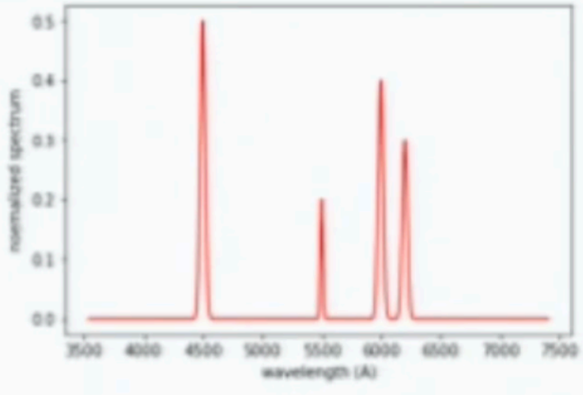
+ B *

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"

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

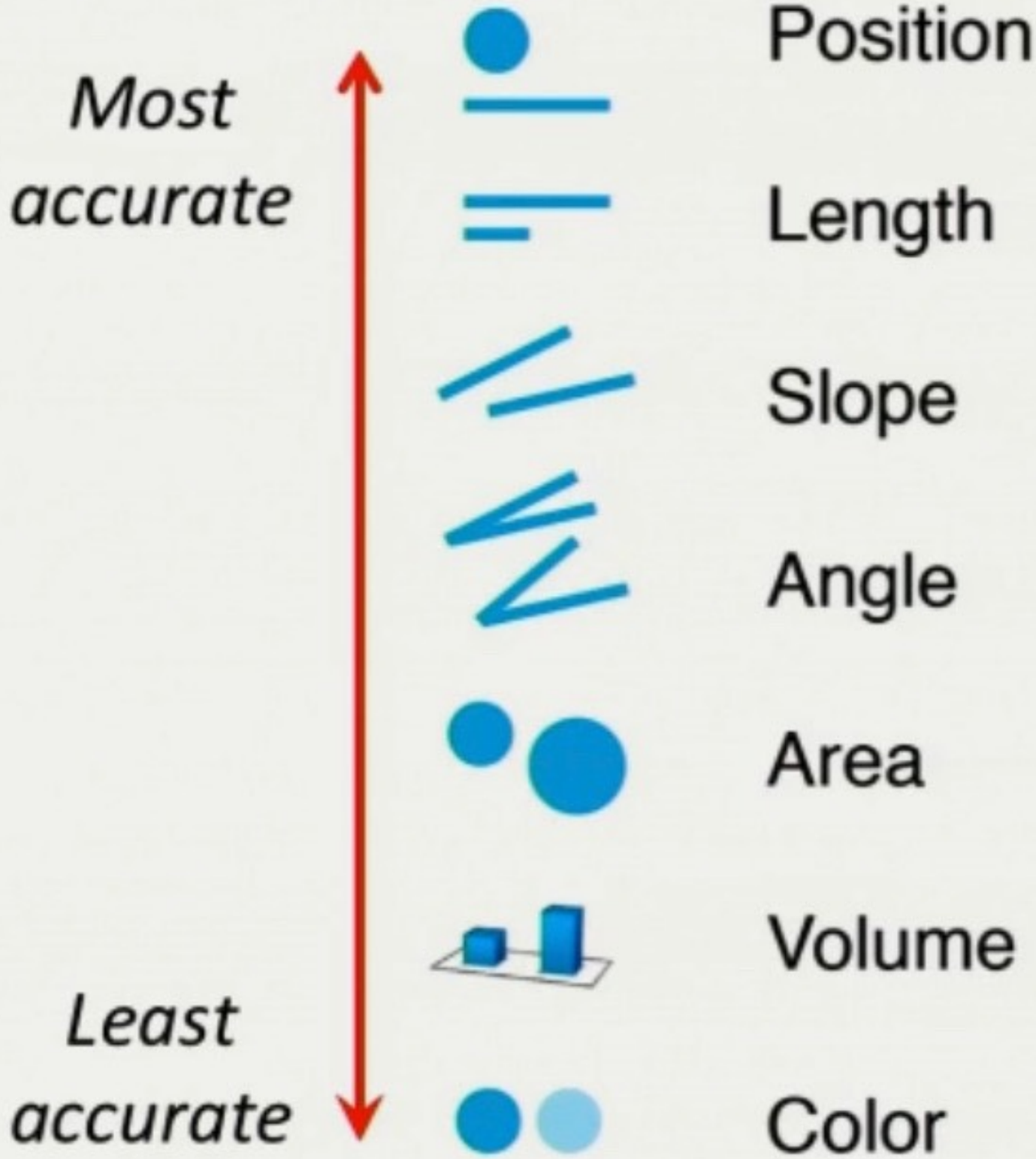
Value encodes continuous variables (less well)

How Many Colors?

Hue encodes nominal variables

(from S. Davidoff)

Relative accuracy of the visualisation space axes:

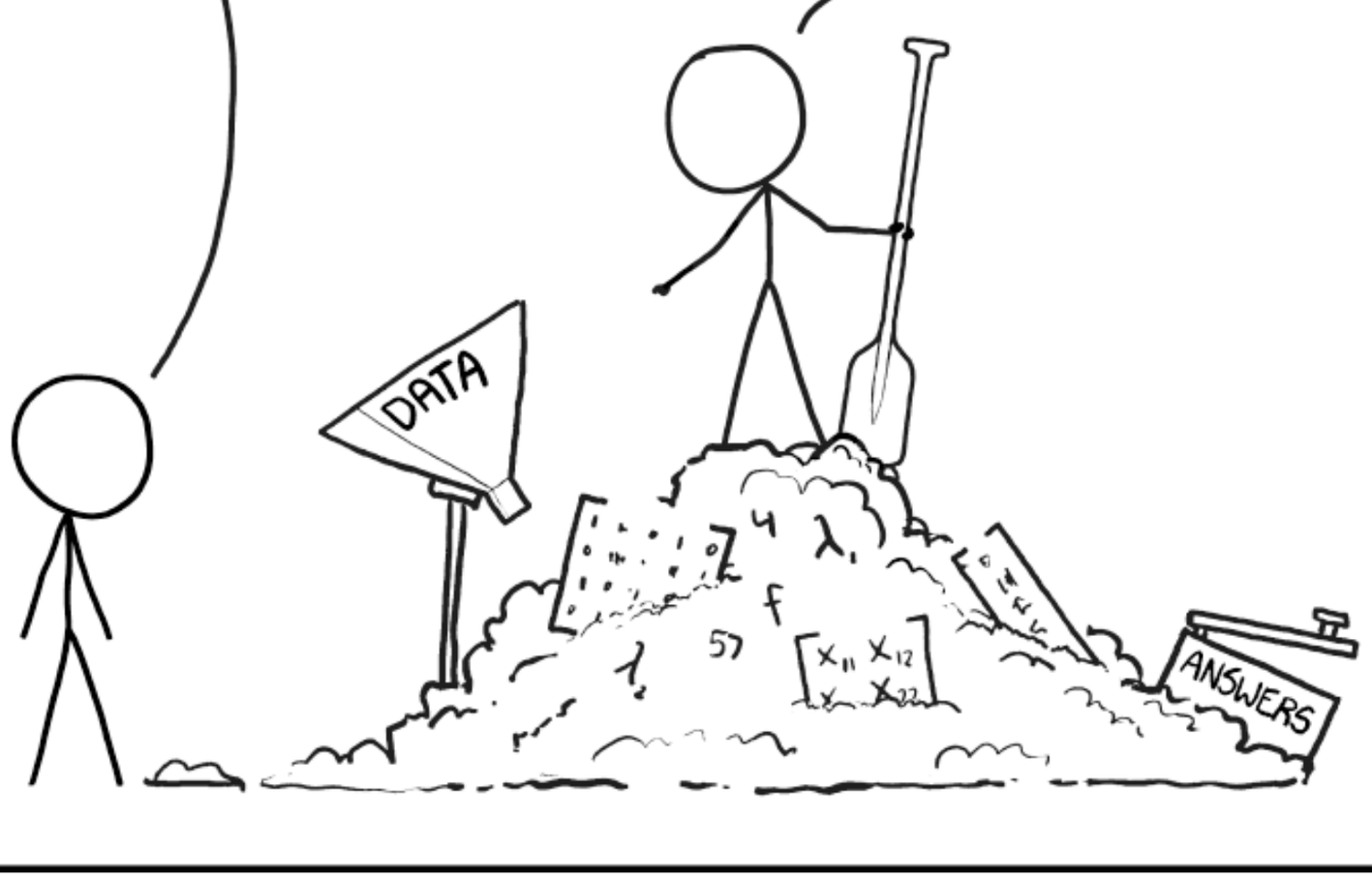


THIS IS YOUR MACHINE LEARNING SYSTEM?

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.



Imbalanced Classes

Uneven Classes…

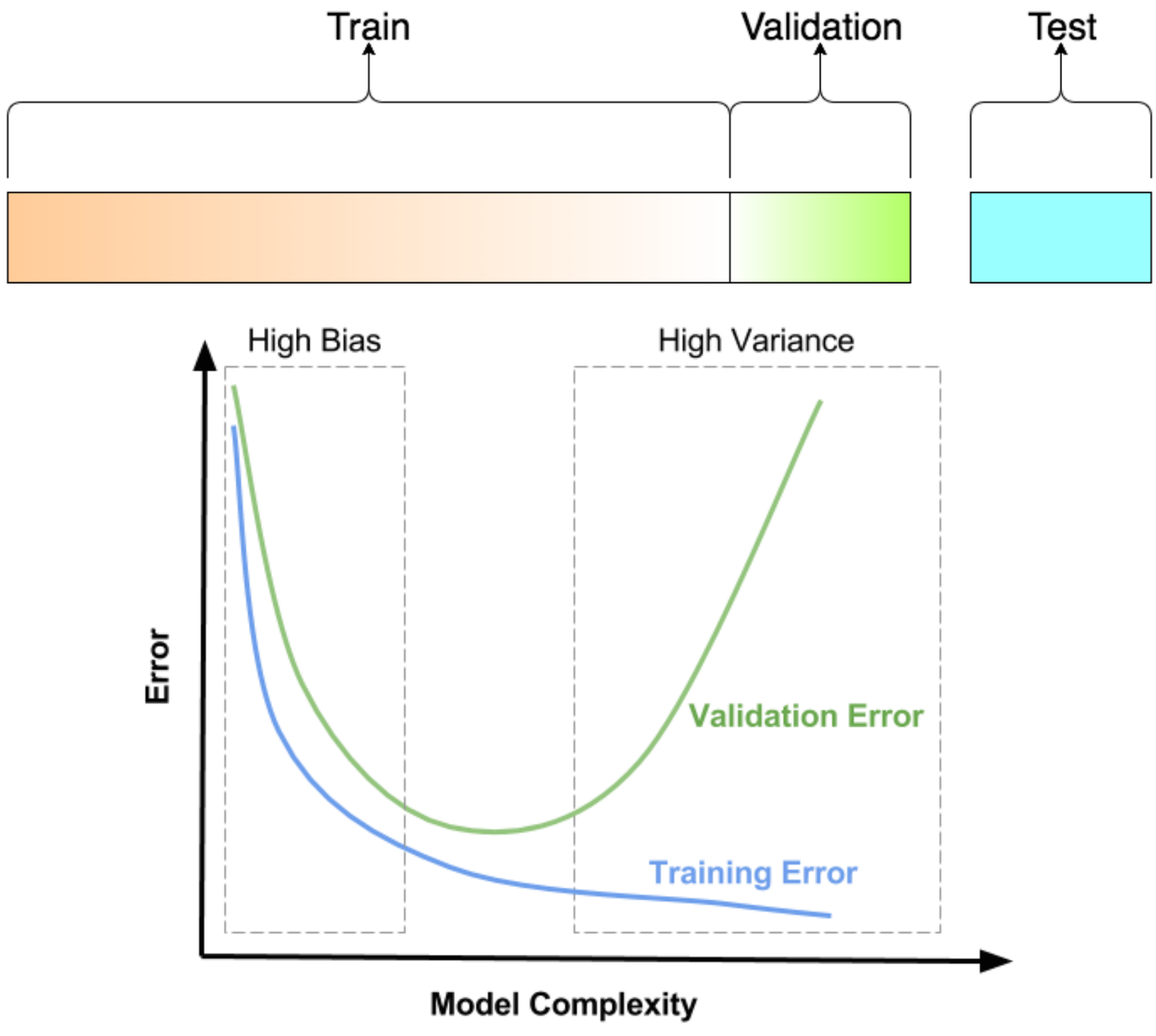
Modify the sets to look highly uneven. The accuracy increases. Why?

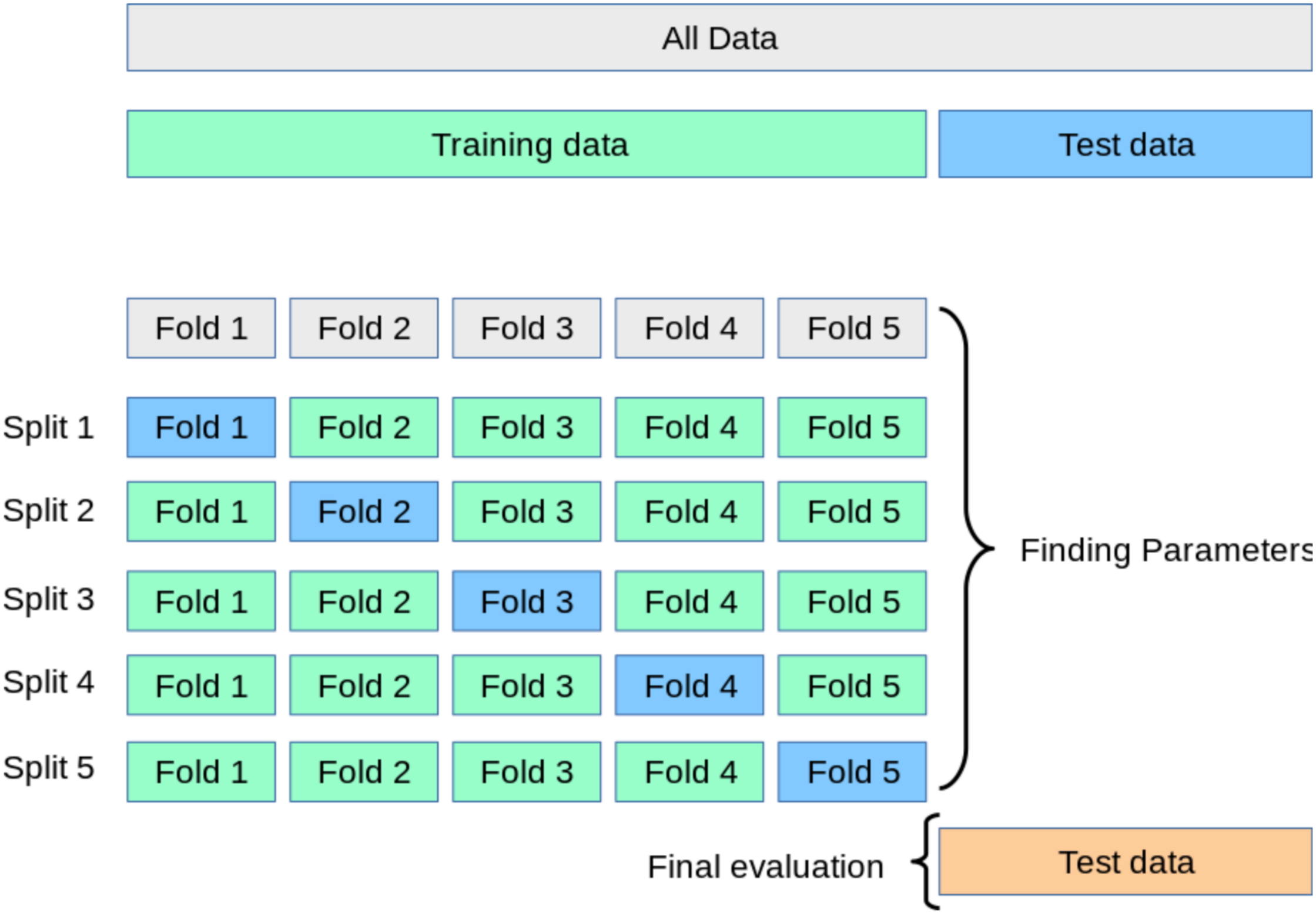
Check the weights keyword

More advanced techniques:

- weights
- sampling & averaging (smote)
- generating artificial rare cases

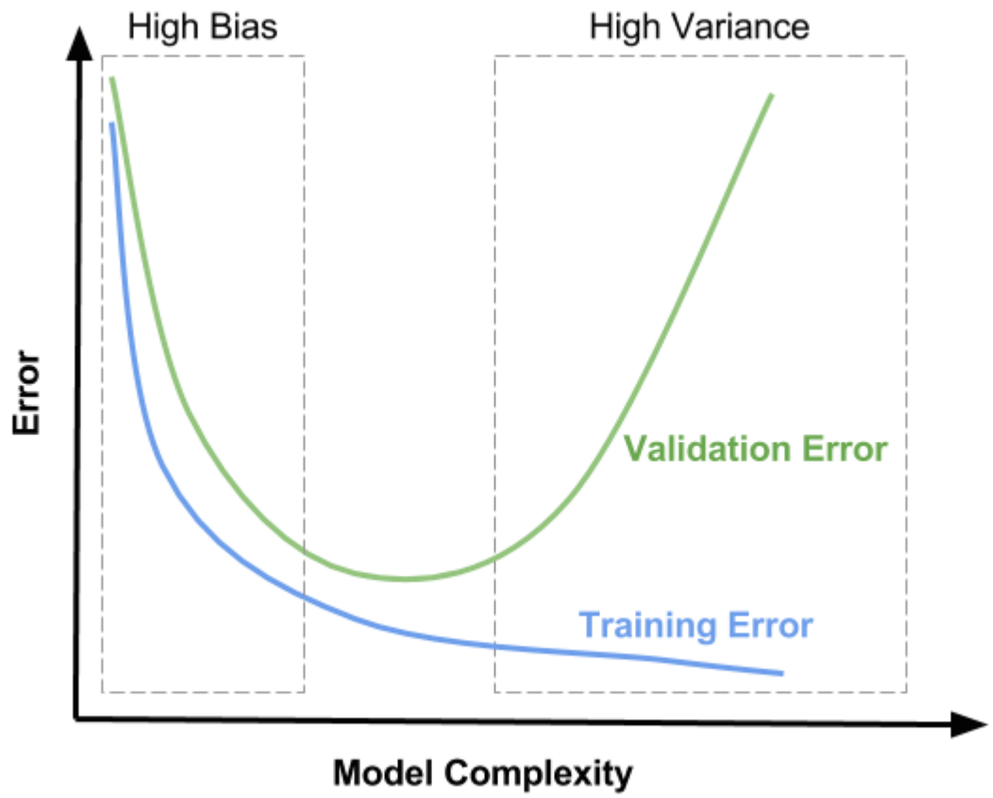
(cross)-Validation




All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5
Finding Parameters
Final evaluation
Test data

Fighting over-fitting

- Regularisation
- Dropout
- Early stopping
- Simplify the architecture

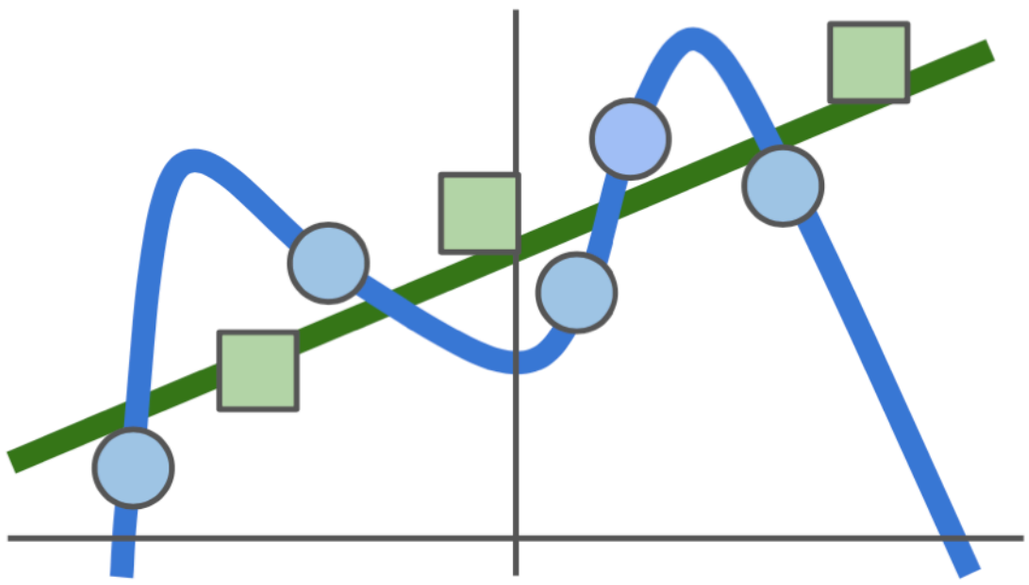


- More data (even if you have to make it up: translate, rotate, flip, crop, lighten/darken, add noise)

# of weights ~ VC dimension ~ # degrees of freedom

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

Data loss: Model predictions should match training data



Regularization: Model should be "simple", so it works on test data

Occam's Razor:
"Among competing hypotheses, the simplest is the best"
William of Ockham, 1285 - 1347

Regularization

λ = regularization strength (hyperparameter)

$$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq y_i}\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \boxed{\lambda R(W)}$$

In common use:

L2 regularization $R(W) = \sum_k \sum_l W_{k,l}^2$ "weight decay"

L1 regularization $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2) $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$


2/17
After 40 years as a teacher, Gary McAdams finally manages to bore a student to death
© 2015 Wulffmorgenthaler/Dist. by Universal Uclick www.gocomics.com