ML101

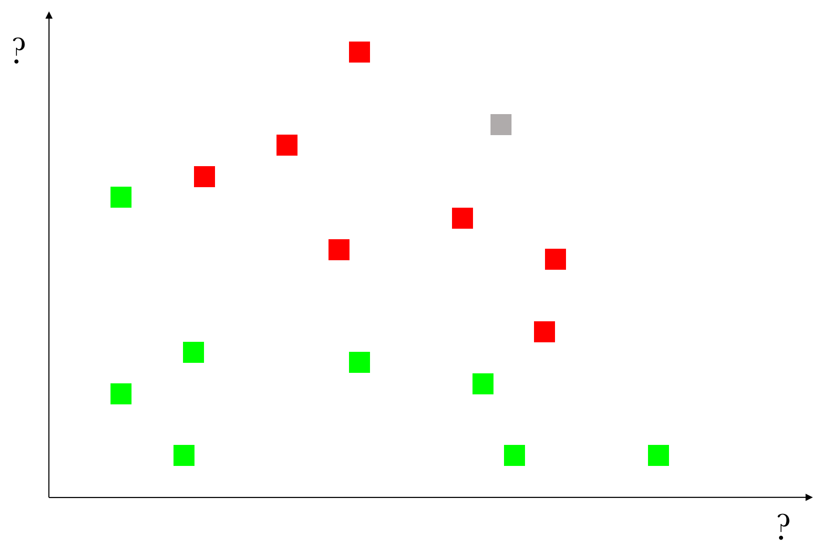
Viviana Acquaviva
(CUNY)

vacquaviva@citytech.cuny.edu

What is machine learning?

**THE ART OF
TEACHING A MACHINE
TO MAKE DECISIONS**
**(e.g. recognize objects,
similarities and differences,
patterns, signal vs noise)**

Our brain machine learns



(of course) ML is not the only way

I CAN ALSO WRITE A FORMULA
(MAKE A MODEL)
TO PREDICT COLOR BASED ON COORDINATES

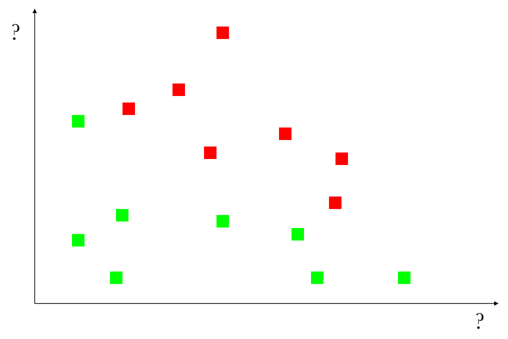
MACHINE LEARNING ALGORITHMS
PROVIDE AN “IMPLICIT MODEL”
AND PERHAPS RESEMBLE MORE THE WAY
WE (HUMANS) SOLVE PROBLEMS

MACHINE LEARNING JARGON

Features are observable quantities known for all objects

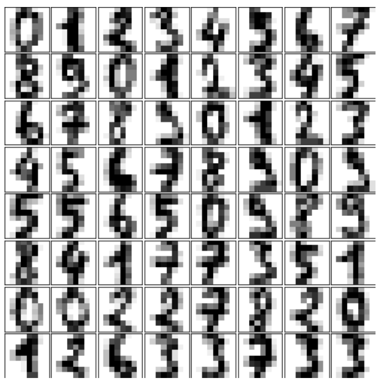
Label is the target property that we want to predict

SUPERVISED ML ASSUMES THAT WE HAVE A SET OF OBJECTS WITH KNOWN LABELS, called the LEARNING SET



Regression vs. classification

Usually we talk about classification when the target is a discrete variable (or class). For example in this image recognition problem:



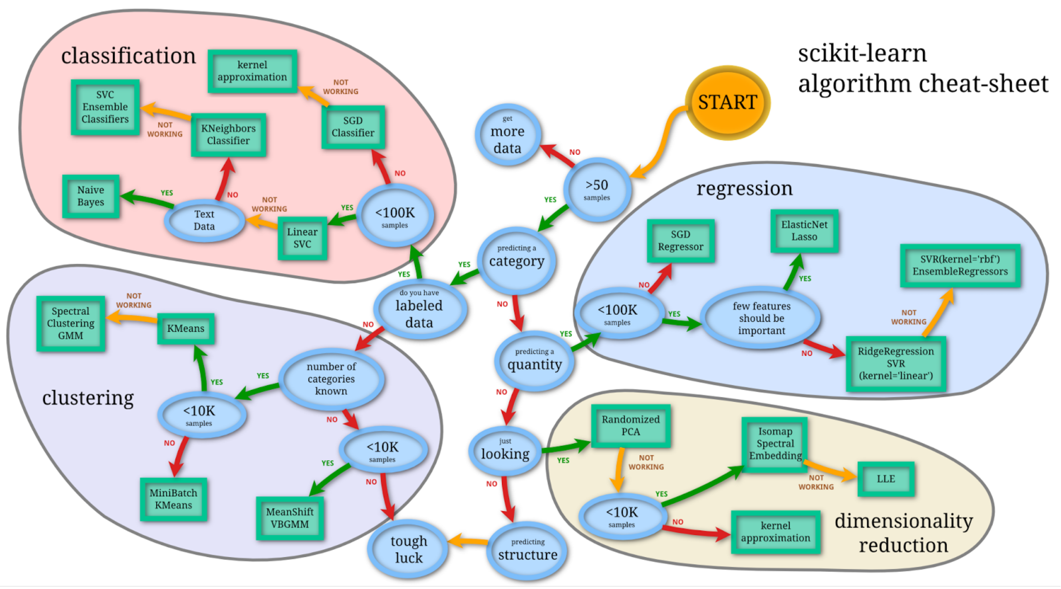
There are a finite (10) numbers of possible outcomes.

Regression vs. classification

- Vice versa, if we are trying to predict, say, the probability that it will rain in half a hour based on the current weather conditions, the outcome (target) is a continuous variable that can have all values between 0 and 1.

What if I was trying to decide whether or not I should bring an umbrella?

ALGORITHMS ABOUND

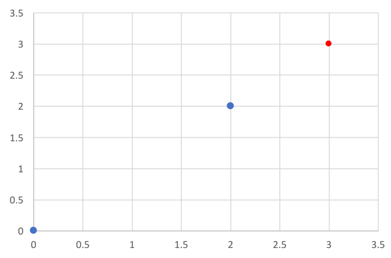
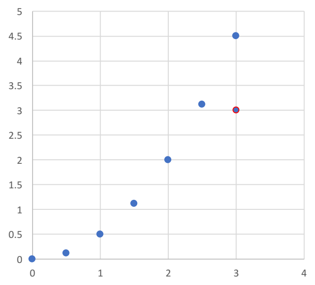


Out of the box, they typically perform terribly.

Performance is limited by size and quality of the learning set.

0 , 0
2 , 2

3 , 3 ?

Data representation (sometimes called feature engineering) and determining whether you have enough data to create a good model are crucial.

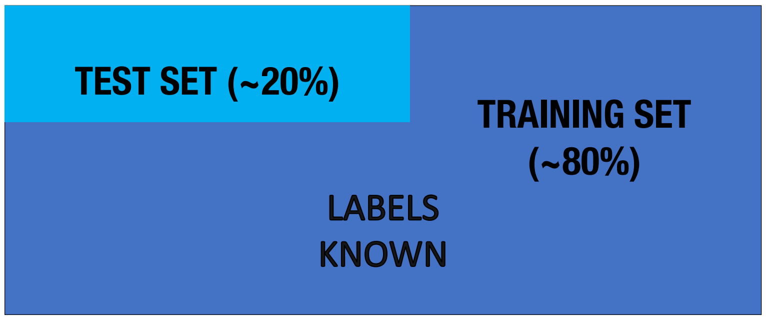
Machine Learning vs Model Fitting

ML	MF
• Data-driven (only as good as the data)	• Intuition or model-driven (only as good as the scientist :))
• Usually generalizes poorly (model derived using some data can't be applied blindly to different data)	• Generalizes well if model (physics) is well understood
• Interpretation is possible but might be non-trivial	• Easier to interpret
• Fast(er)	• Might be computationally intensive
• More robust/accommodating of mixed and missing data	• Dealing with heterogenous data often a pain in the neck
• Allows serendipitous discoveries	• Leads to loss of information if models are too simplistic

Synergy is often the best strategy

Important:
you shouldn't use all of your learning set to build your model.

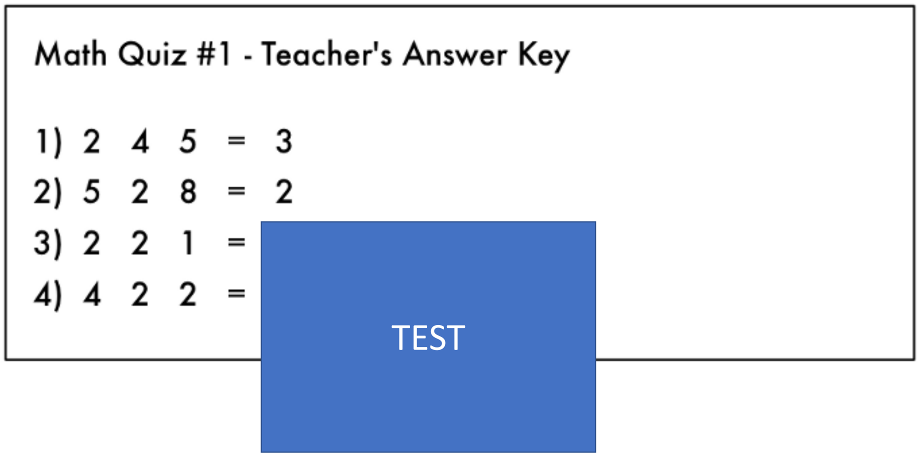
It is customary to split the learning set into a training set and test set



By building a model on the training set and applying it to the test set, you "mimic" what happens when your model sees new data for which the labels are not known.
Otherwise you would be too optimistic!

Note:
You still are.

For example…



Math Quiz #1 - Teacher's Answer Key

1) 2 4 5 = 3
2) 5 2 8 = 2
3) 2 2 1 =
4) 4 2 2 =

Diagnosing and Improving Machine Learning algorithm performance: cross validation, performance metrics, diagnostics

The goal of the training set and test set split is to be able to evaluate performance on new objects.

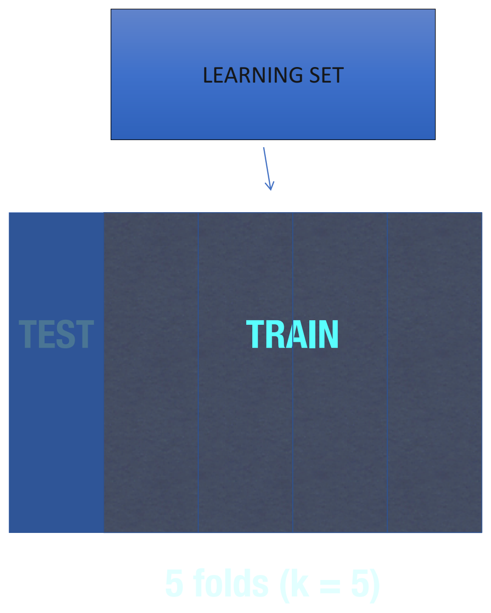
The test set “mimics” new data.

However, it might be better to pick more than one test split.
Why would this be a good idea?

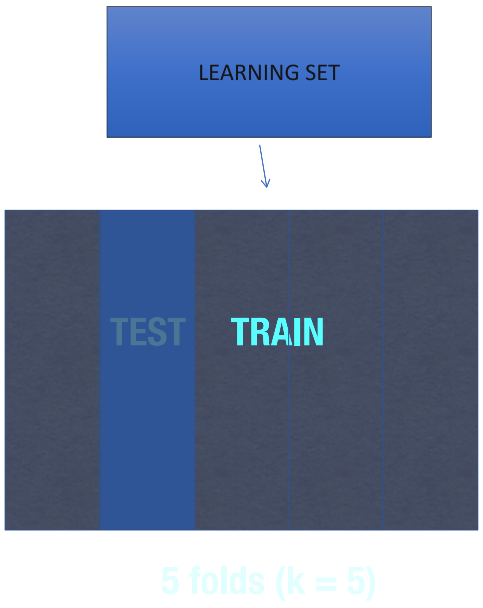
1. We use all the training data for training!

2. We avoid the risk of under/overestimating performance because of a “weird” pick of train/test split.

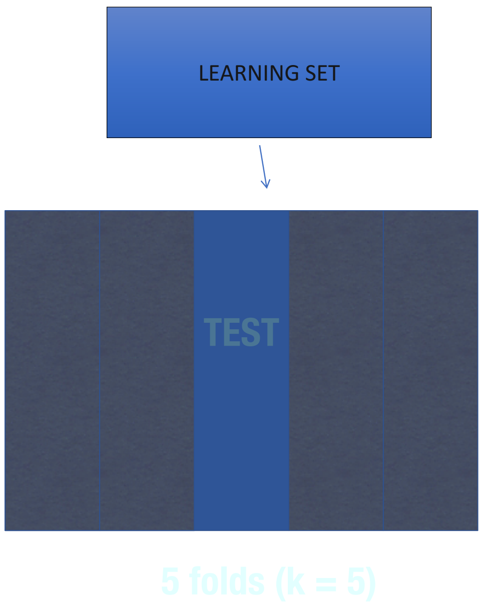
k-FOLD CROSS VALIDATION



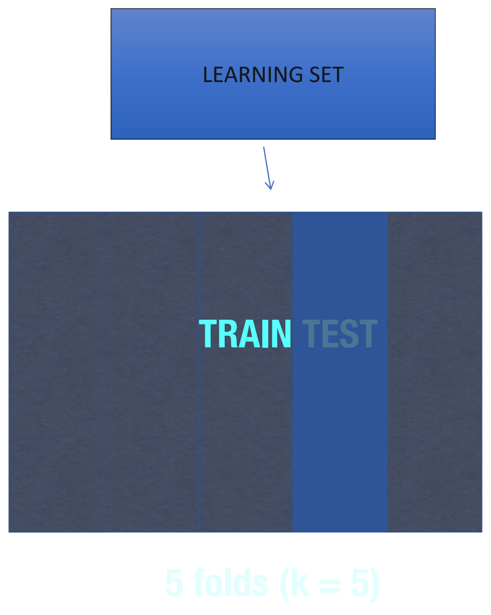
k-FOLD CROSS VALIDATION



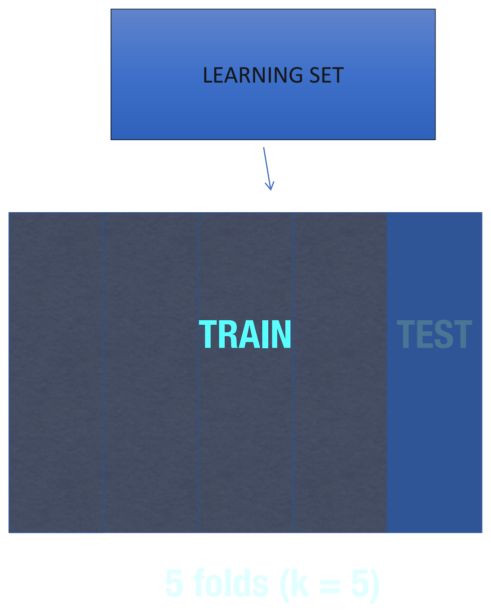
k-FOLD CROSS VALIDATION



k-FOLD CROSS VALIDATION



k-FOLD CROSS VALIDATION

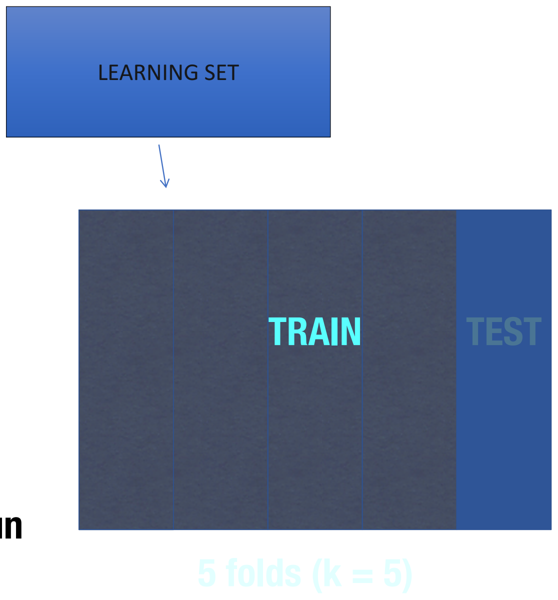


k-FOLD CROSS VALIDATION

☺ We use all learning data

☺ The spread
of scores vector
gives us an idea of average
performance + uncertainty

☹ It takes 5x more time to run



Diagnosing a ML algorithm

BIAS

Algorithm
can't capture
complexity
of rule
connecting
input and output

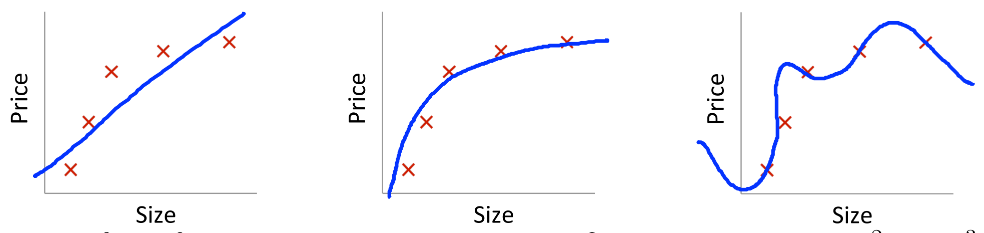
UNDERFITTING

VARIANCE

Algorithm
is excessively
tailored
to training set
and generalizes
poorly

OVERFITTING

Bias/variance



$\theta_0 + \theta_1 x$ | $\theta_0 + \theta_1 x + \theta_2 x^2$ | $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High bias (underfit)

$d=1$

"Just right"

$d=2$

High variance (overfit)

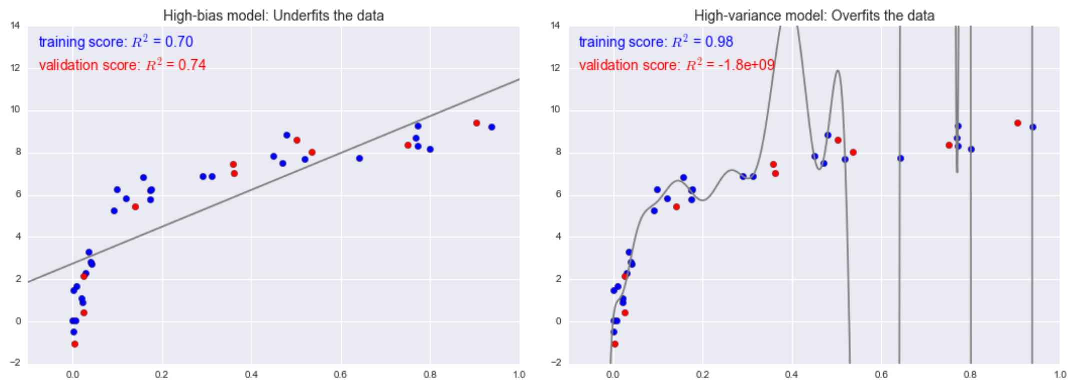
$d=4$

Andrew Ng

slide from Andrew Ng's Coursera ML class

How can we diagnose high variance vs high bias?

figure from Jake VanderPlas' book



High bias: test and train error are similar but high

High variance: there is a gap between test and train error because algorithm does not generalize well

Improving high bias

1. Try using different features.
2. Try engineering new features.
3. Try a more complex algorithm.

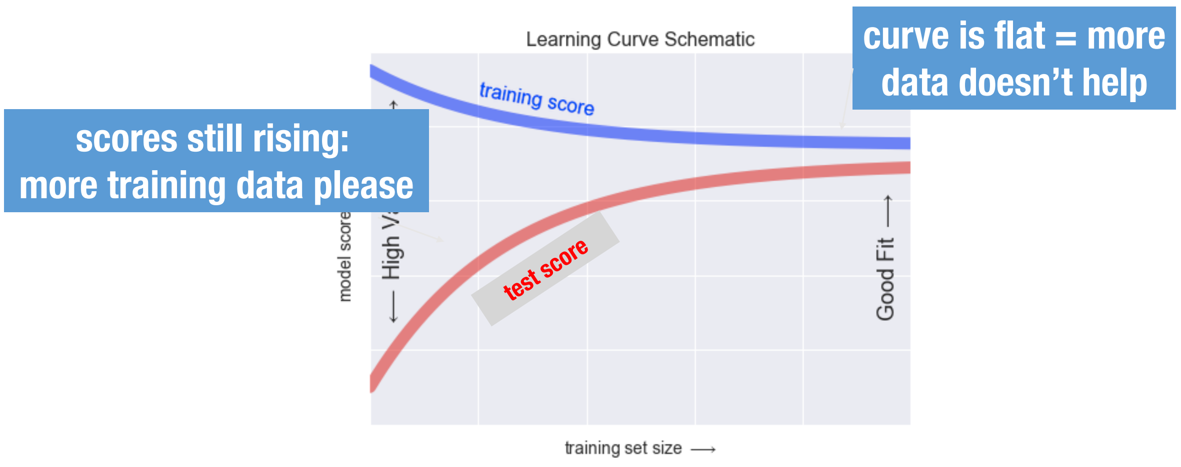
Improving high variance

1. Try reducing the number of features.
2. Try a less complex algorithm/chance parameters.

Also: Check if you need more training data

Useful diagnostics: learning curves

plot performance of algorithm for train and test set
as a function of size of training set



Note: which algorithm is the best?

- High bias low variance
- High variance low bias
- Lowest gap between train/test
- Highest test scores

Theorem. For the squared error loss, the bias-variance decomposition of the expected generalization error at $X = \mathbf{x}$ is

$$E_L\{Err(\varphi_L(\mathbf{x}))\} = \text{noise}(\mathbf{x}) + \text{bias}^2(\mathbf{x}) + \text{var}(\mathbf{x})$$

If we are willing to take a hit in bias,
we can reduce variance,
and still have an improved model.
(ensemble/bagging methods)

DECISION TREES

- Work by splitting data on different values of features
- If categorical features, the split would be on yes/no
- If numerical, the split would be on a certain value (e.g. x > 100 or x < 100)

Example: Look at this 2-feature data set. How should we split?

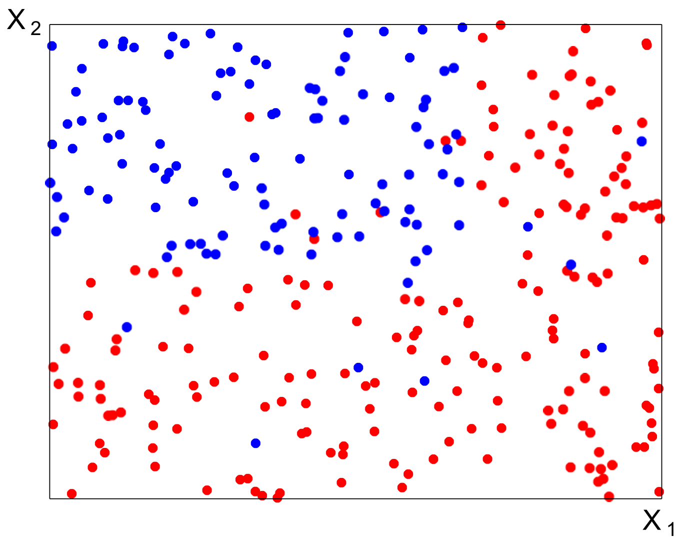


Figure credit:
Gilles Louppes

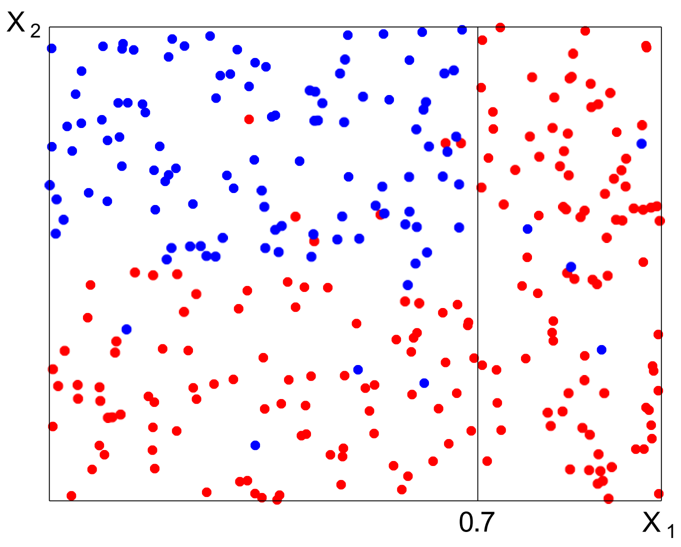


Figure credit:
Gilles Louppes

Should we stop?

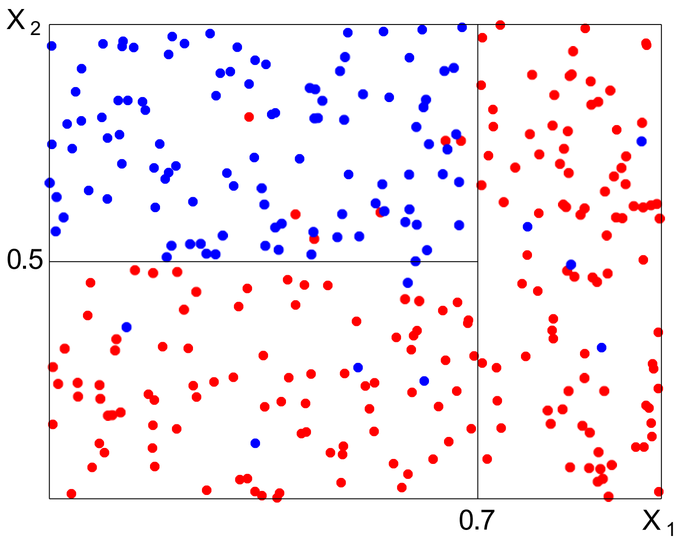
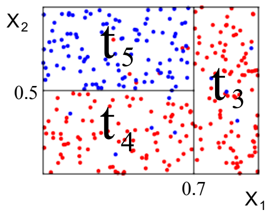
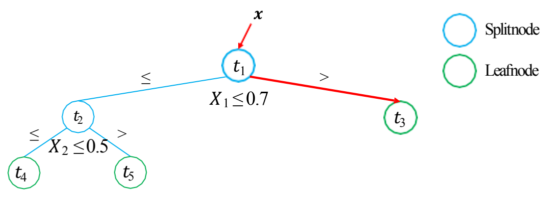


Figure credit:
Gilles Louppes

Decision trees: defined by splits and leaves





How many splits in this tree?
How many leaves?
How do we decide whether we should keep splitting?

Figures credit:
Gilles Louppes

Pseudo code for decision trees

```
function BuildDecisionTree(L)
   Create node $t$ from the learning sample $L_t$ = L;
   calculate (im)purity
   if the stopping criterion is met for $t$ then
     $\hat{y}_t$ = some constant value/class  (MAKE PREDICTION)
   else
      Find the split on $L_t$ that maximizes impurity
      decrease
       $s^* = \arg\max_{s \in Q} \Delta i(s, t)$
       Partition $L_t$ into $L_{t_L} \cup L_{t_R}$ according to $s^*$
             $t_L$ = BuildDecisionTree($L_L$)
             $t_R$ = BuildDecisionTree($L_R$ )
   end if
   return t
end function
```

Code adapted from Gilles Louppes

stopping criterion
Gini (im)purity = 0

Gini (node L) =

$1 - \sum f(i)^2$

where f(i) is the frequency of the i-th class

Gini (splits Lt and Lr) =

$L_L/L * (1 - \sum f(i)^2) +$
$L_R/L * (1 - \sum f(i)^2)$

where f(i) is the frequency of the i-th class

Note: **splits happen along features!**