Something about machine learning Episode 2

Viviana Acquaviva
(CUNY)
vacquaviva@citytech.cuny.edu

Let us focus on supervised learning for now.

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 =

TEST

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 unseen examples.

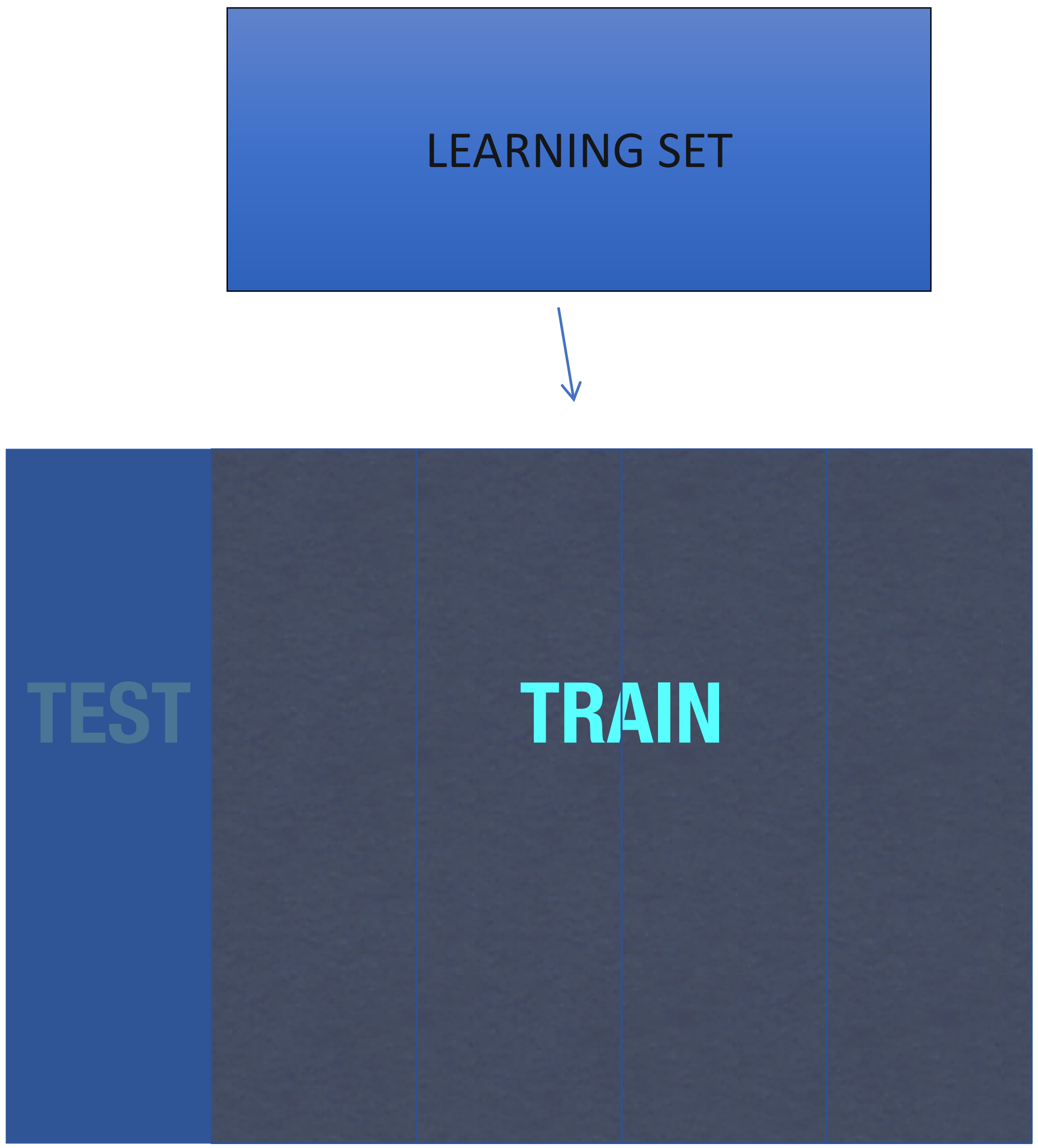
The test set "mimics" new data.

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!

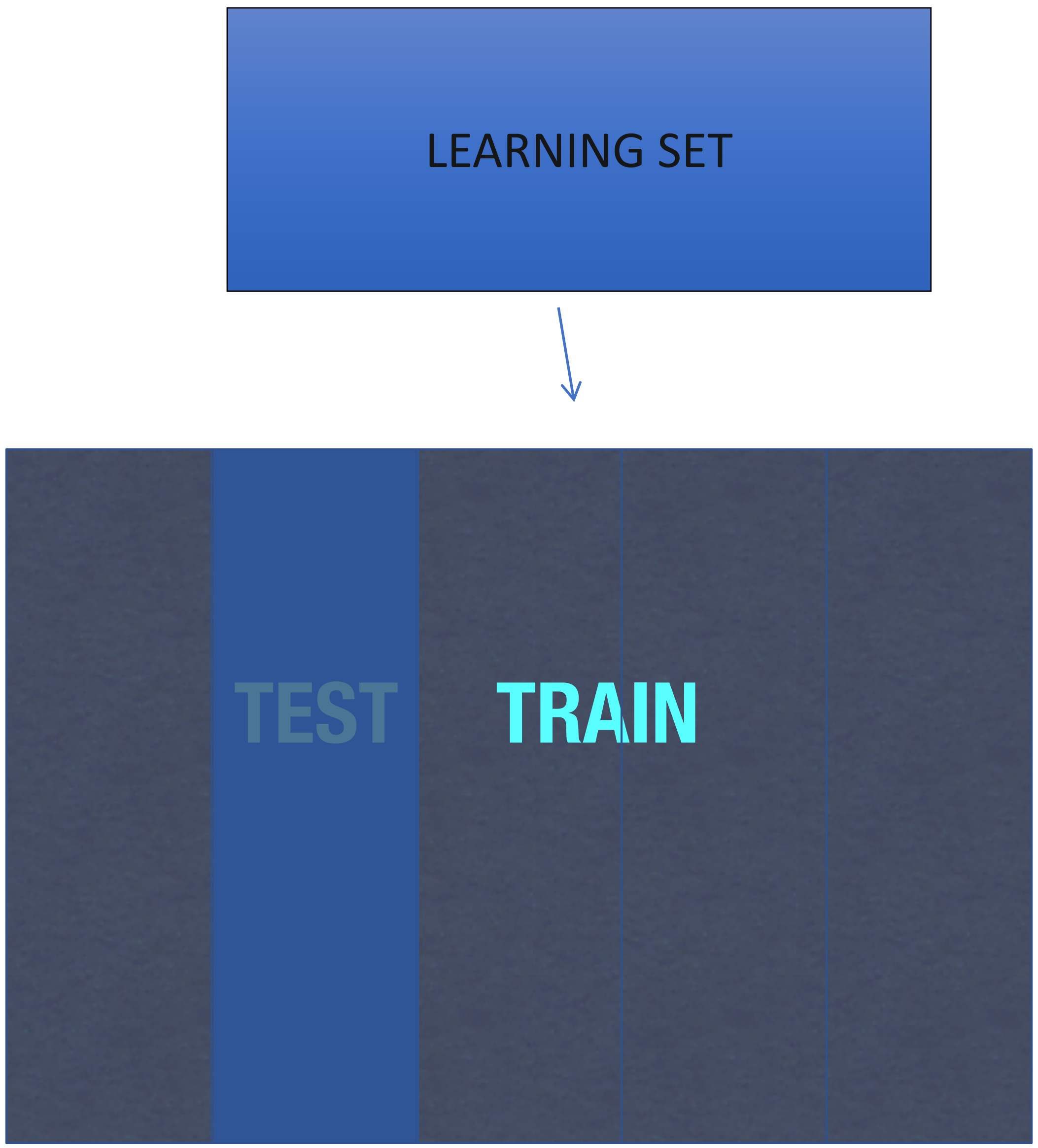
2. We avoid the risk of under/overestimating performance because of a “weird” pick of train/test split.

k-FOLD CROSS VALIDATION



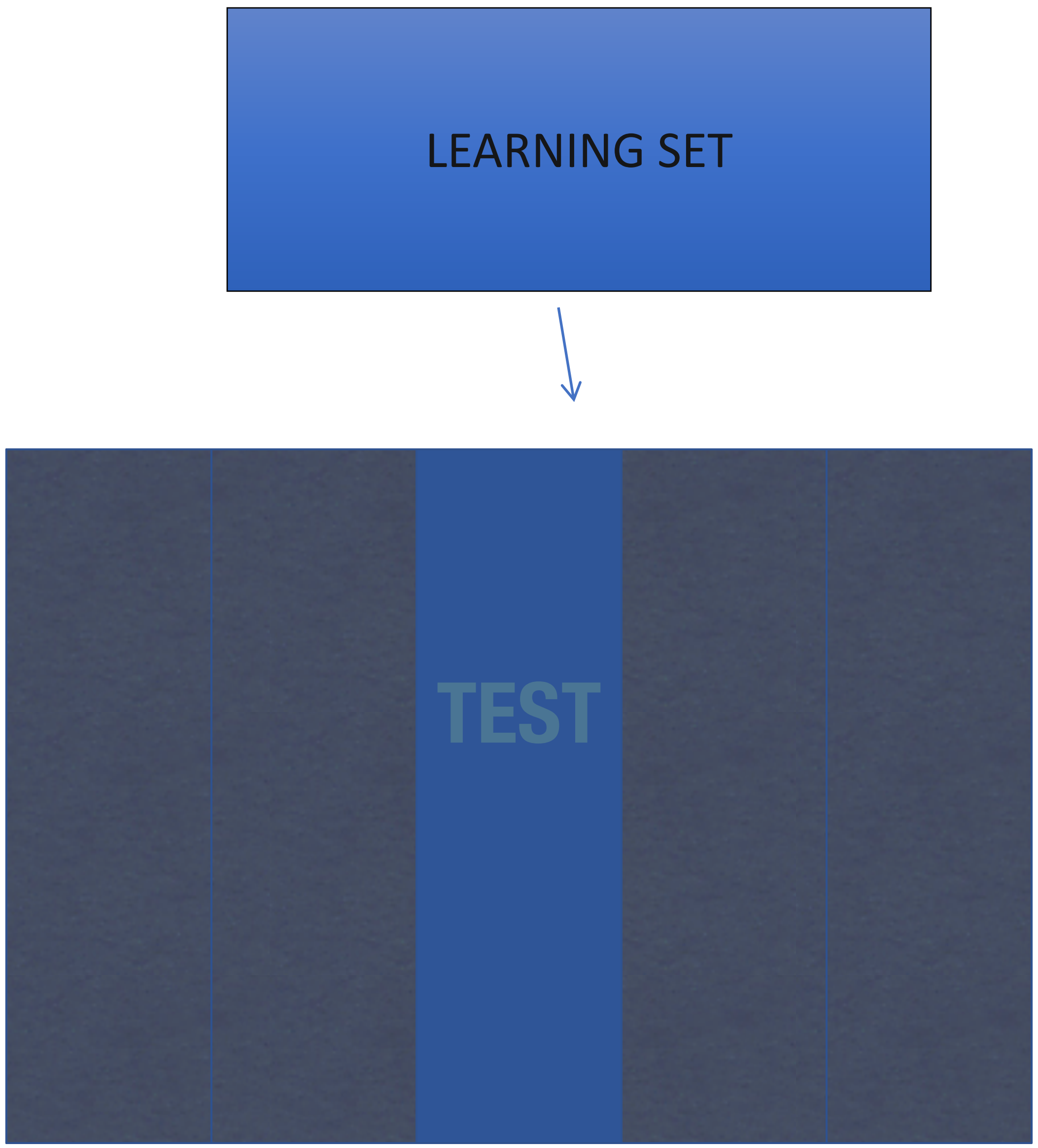
5 folds (k = 5)

k-FOLD CROSS VALIDATION



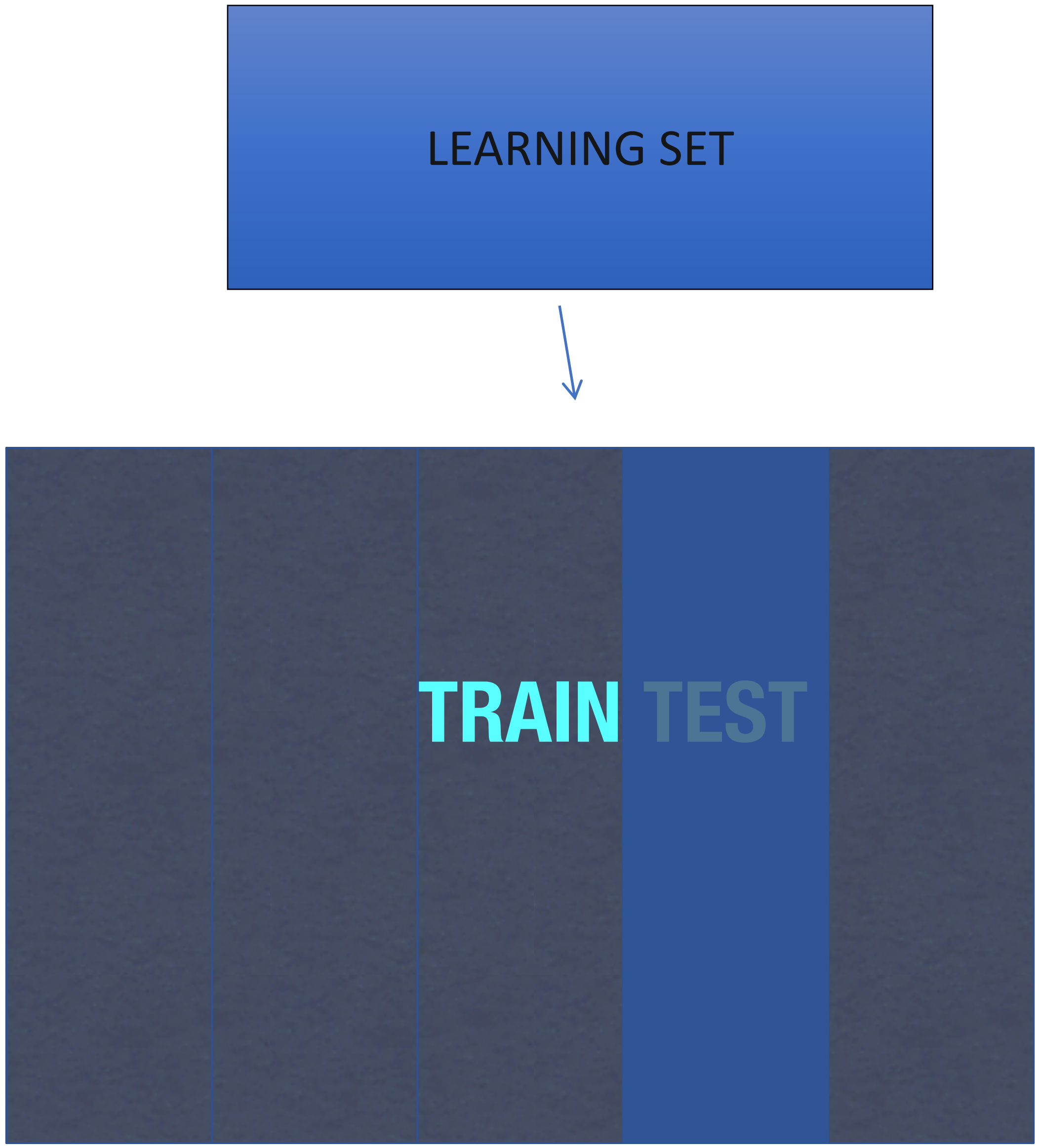
5 folds (k = 5)

k-FOLD CROSS VALIDATION



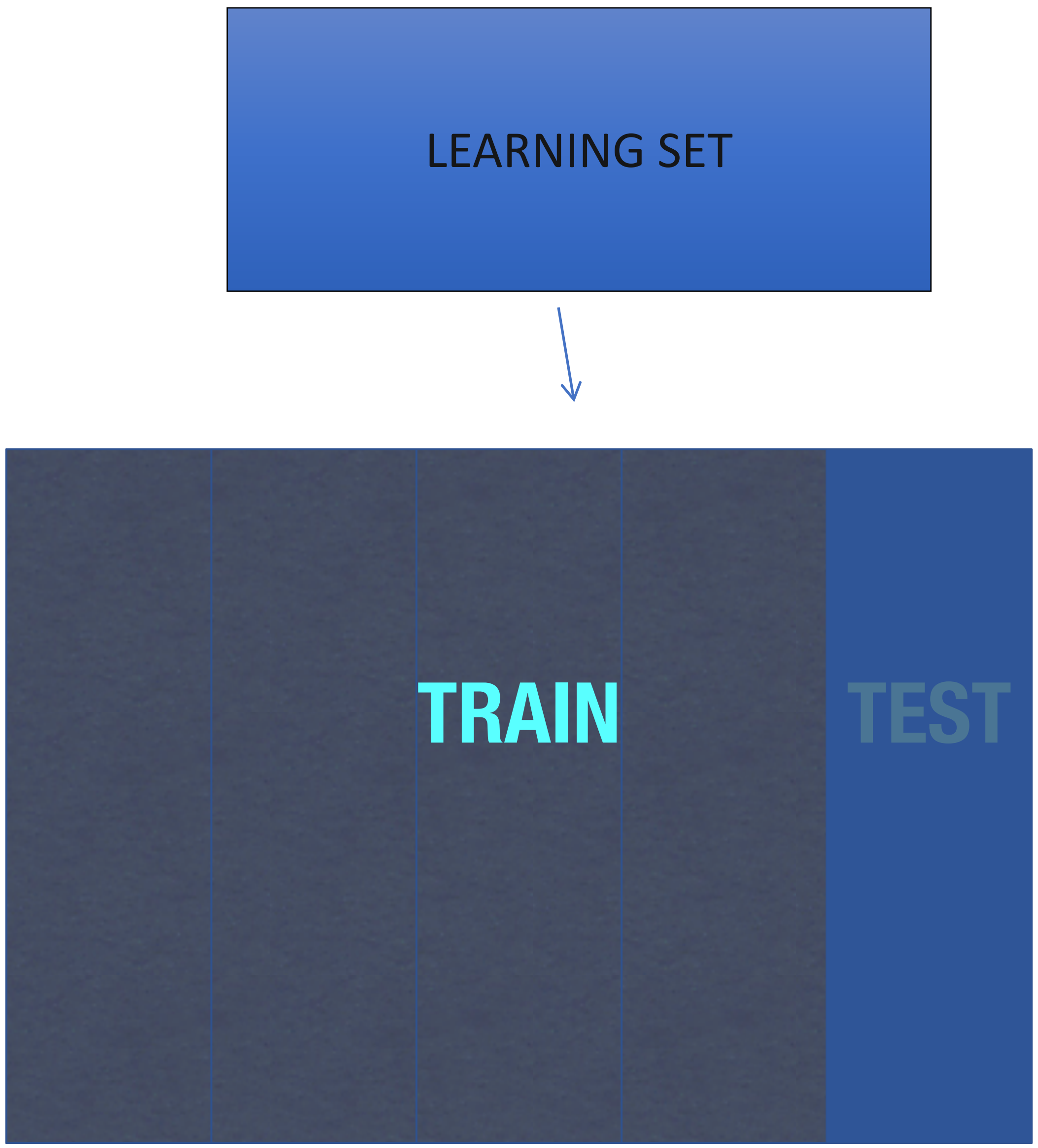
5 folds (k = 5)

k-FOLD CROSS VALIDATION



5 folds (k = 5)

k-FOLD CROSS VALIDATION



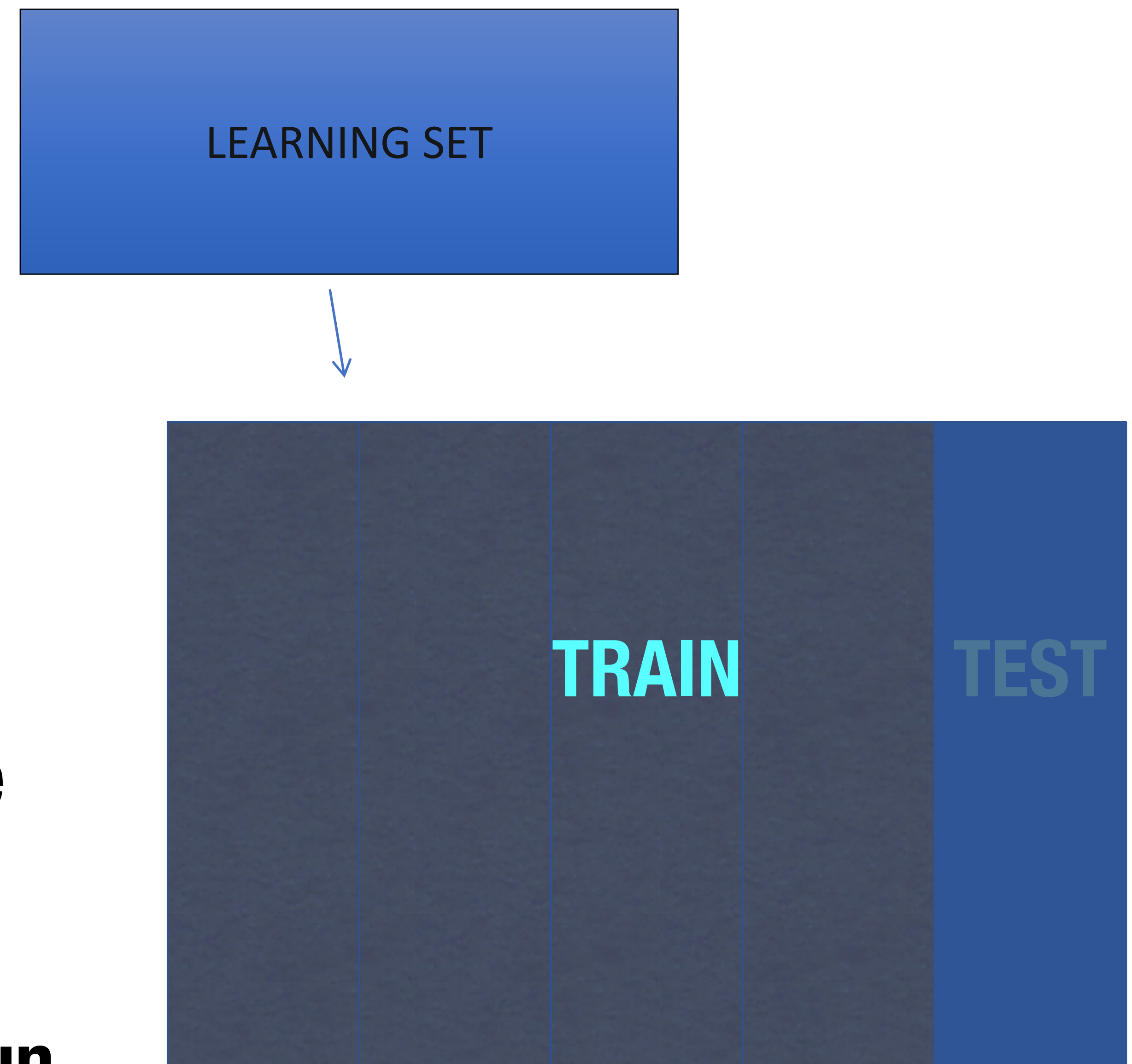
5 folds (k = 5)

k-FOLD CROSS VALIDATION

☺ We use all learning data

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

☹ It takes 5x more time to run



5 folds (k = 5)

Diagnosing a ML algorithm

BIAS

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

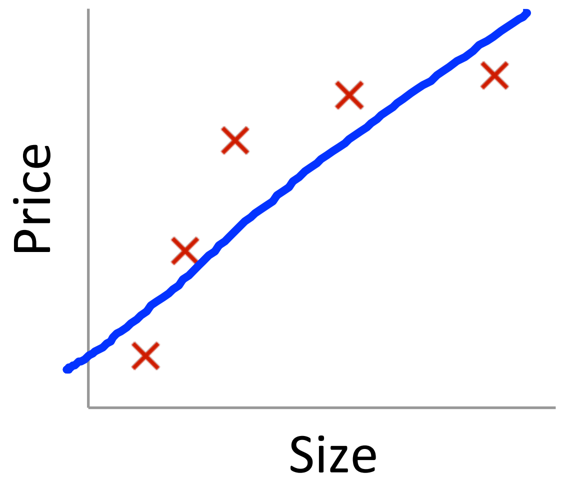
UNDERFITTING

VARIANCE

Algorithm
is excessively
tailored
to training set
and generalizes
poorly

OVERFITTING

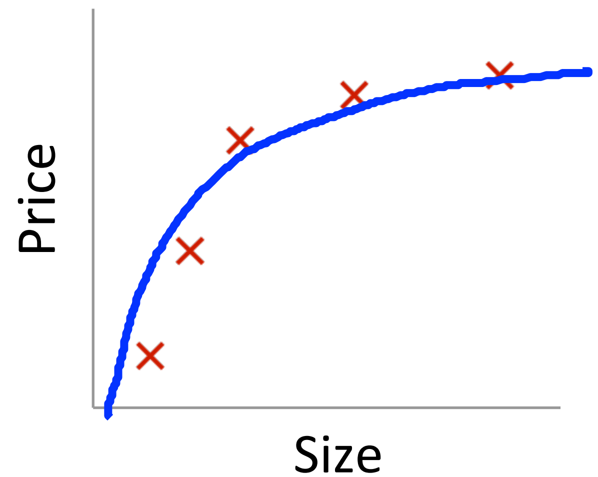
Bias/variance



$$\theta_0 + \theta_1 x$$

High bias
(underfit)

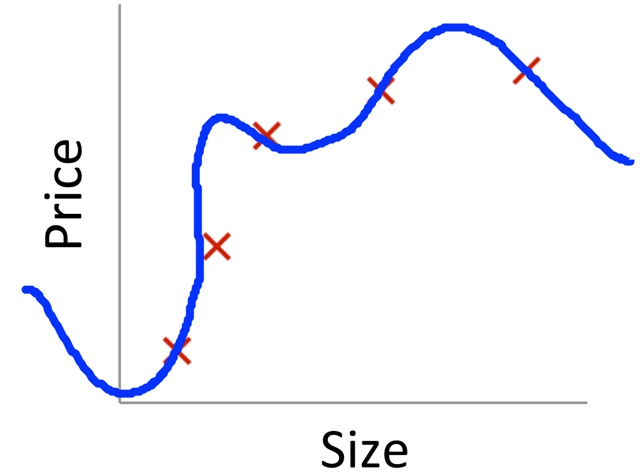
$d=1$



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

"Just right"

$d=2$



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

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

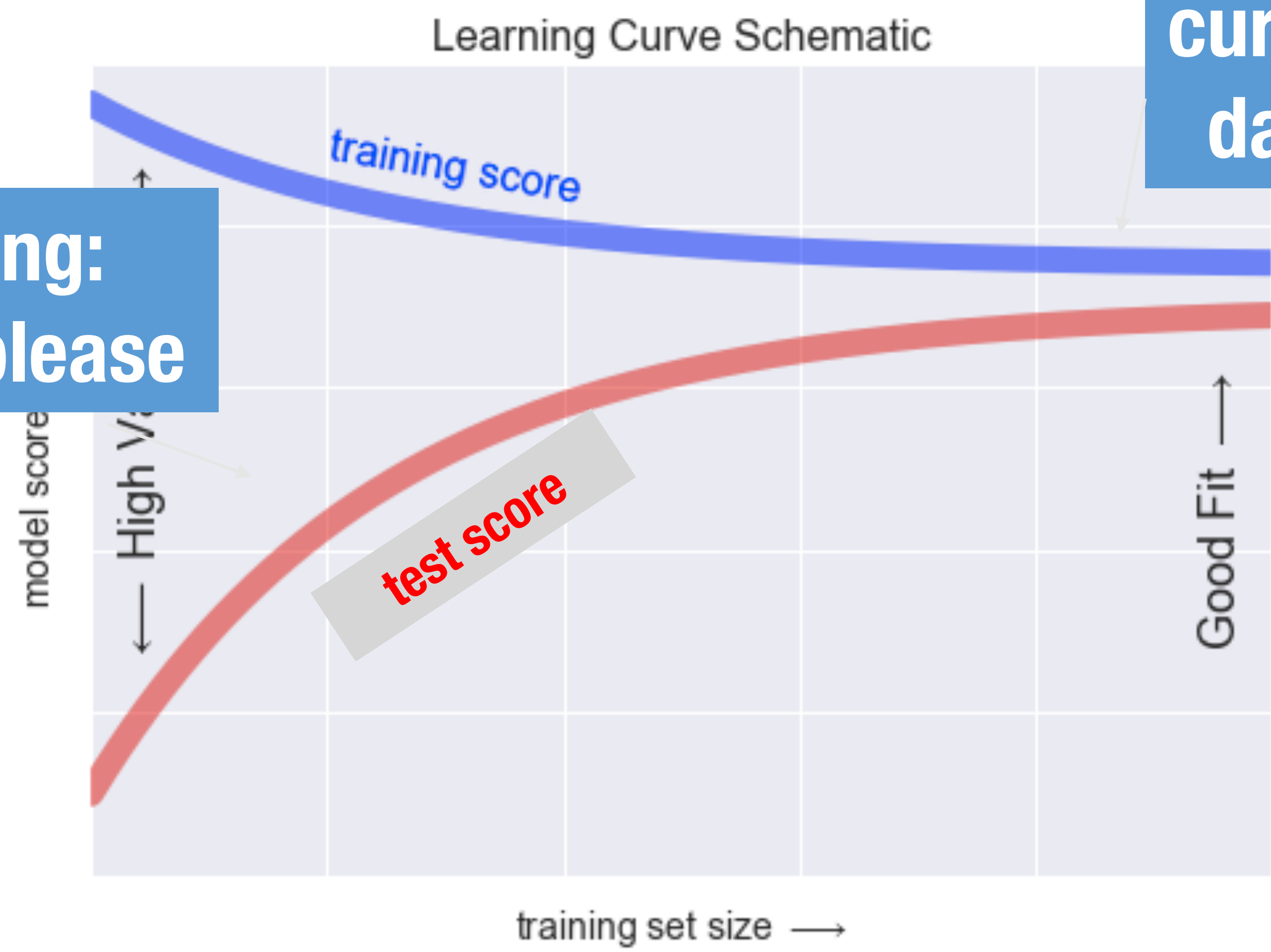
1. Try reducing the number of features.
2. Try a less complex algorithm/change parameters.

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

scores still changing: more training data please



curve is flat = more data doesn’t help

Note: which algorithm is the best?

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

Note: which algorithm is the best?

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

DECISION TREES

- Work by splitting data on different values of features
- Simplest trees are binary trees
- 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?

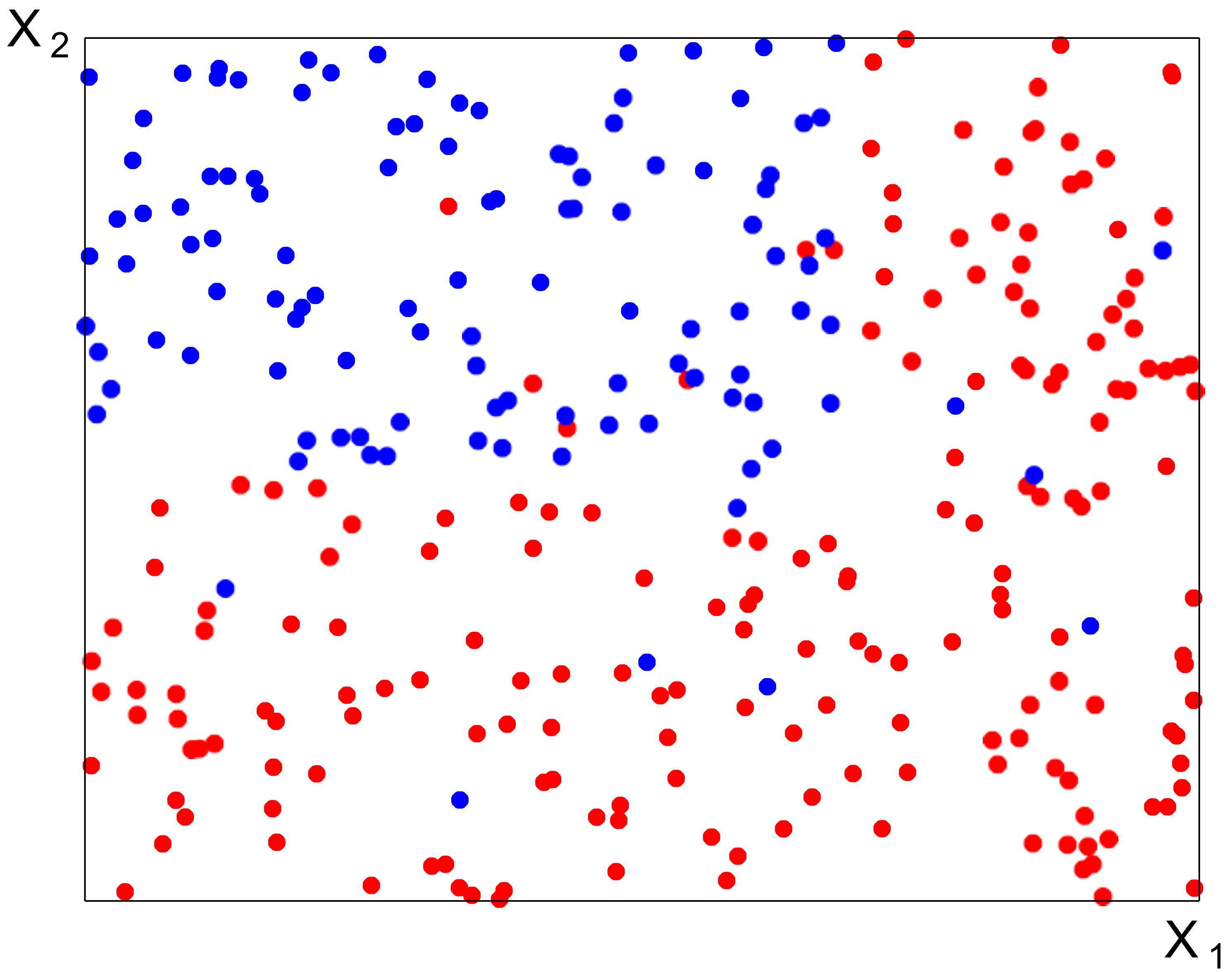


Figure credit:
Gilles Louppes

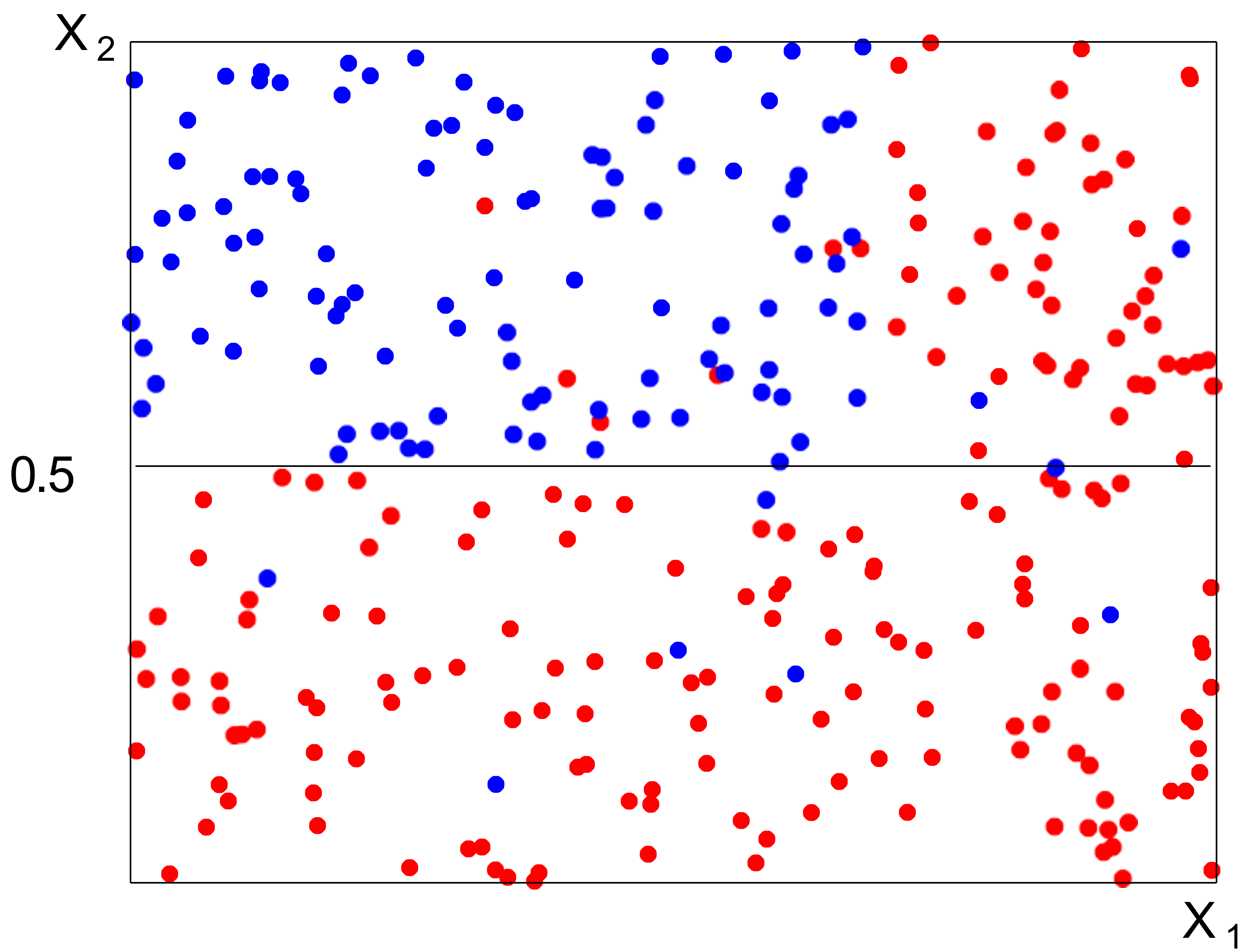


Figure credit:
Gilles Louppes

Should we stop?

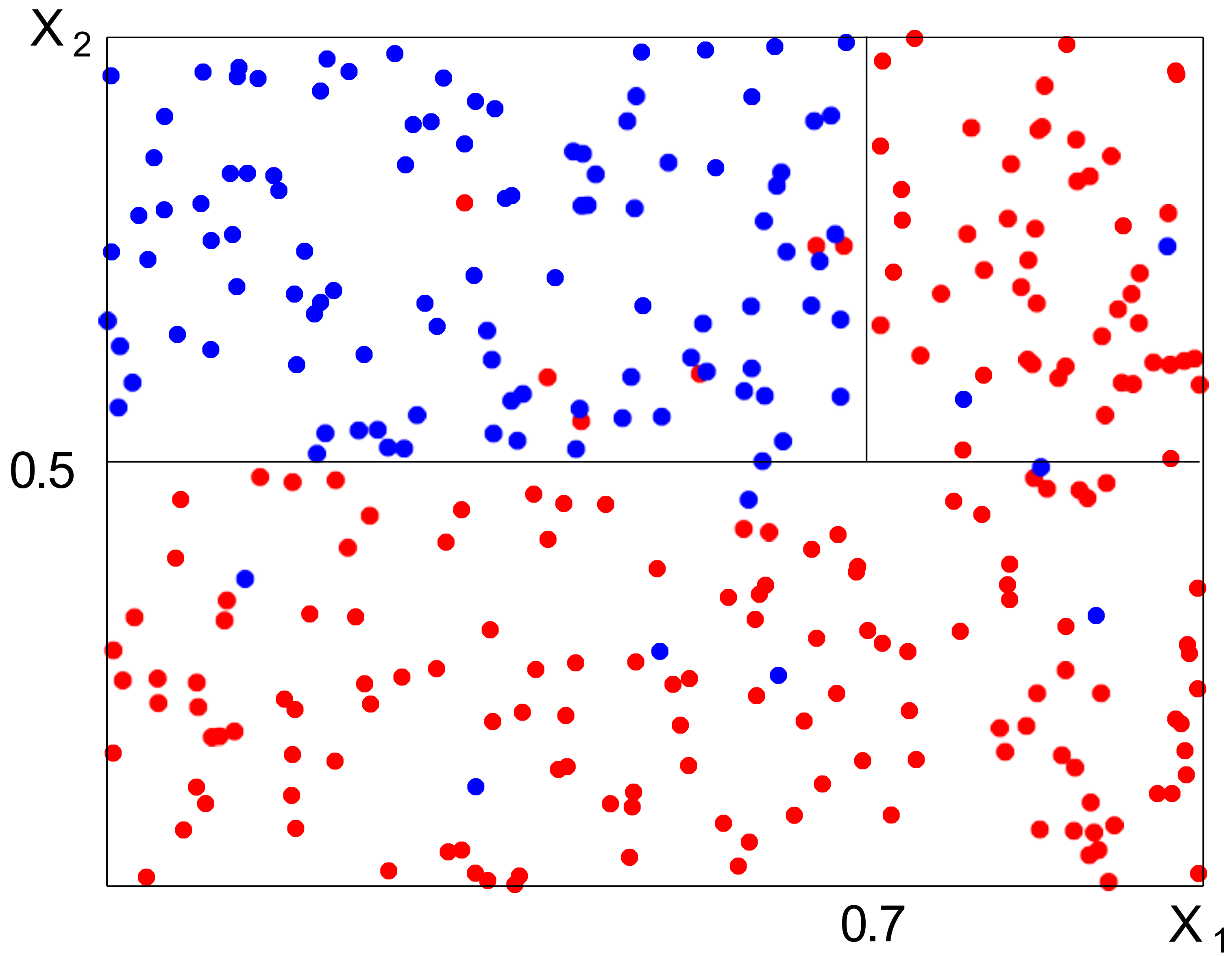
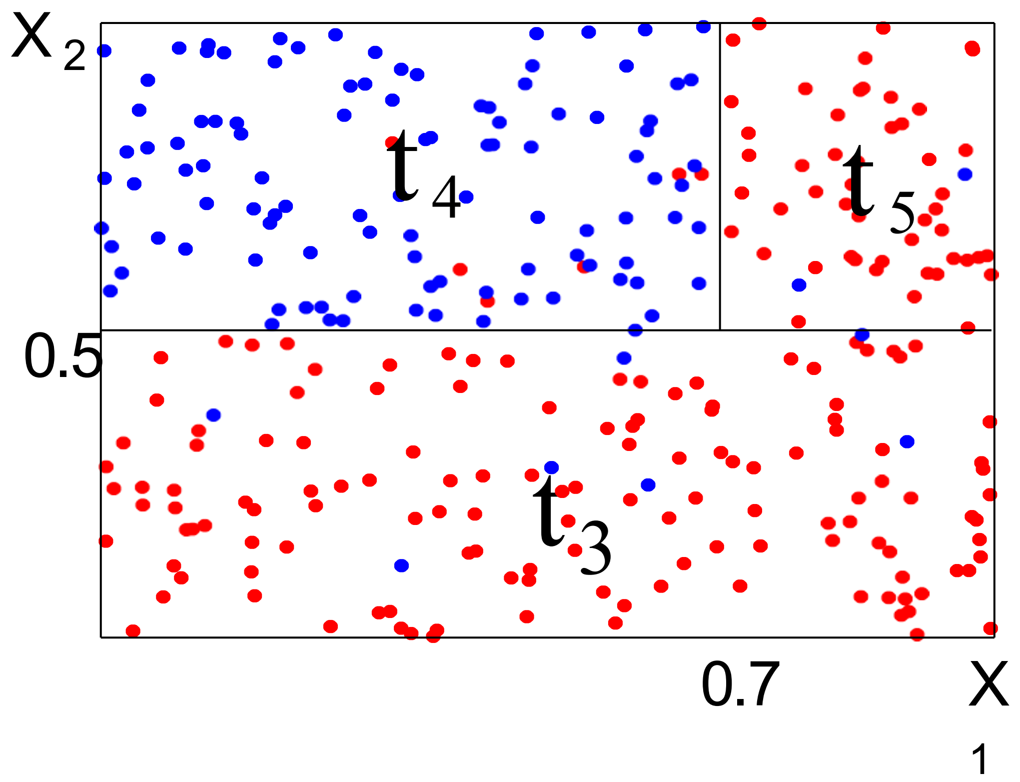
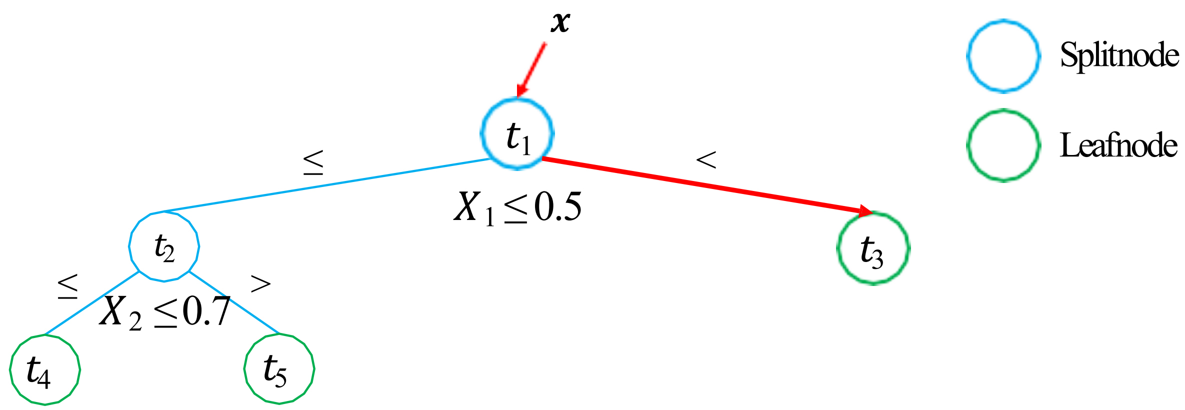


Figure credit:
Gilles Louppes

Decision trees: defined by splits and leaves





Figures credit:
Gilles Louppes

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

Pseudo code for decision trees

```
function BuildDecisionTree(L)
   Create node t from the learning sample Lt = L;
   calculate (im)purity
   if the stopping criterion is met for t then
     y^t = some constant value/class  (MAKE PREDICTION)
   else
      Find the split on Lt that maximizes impurity
      decrease
       s* = arg max Δi (s, t)
                 s∈Q
       Partition Lt into LtL ∪ LtR according to s*
             tL = BuildDecisionTree(LL)
             tR = BuildDecisionTree(LR )
   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

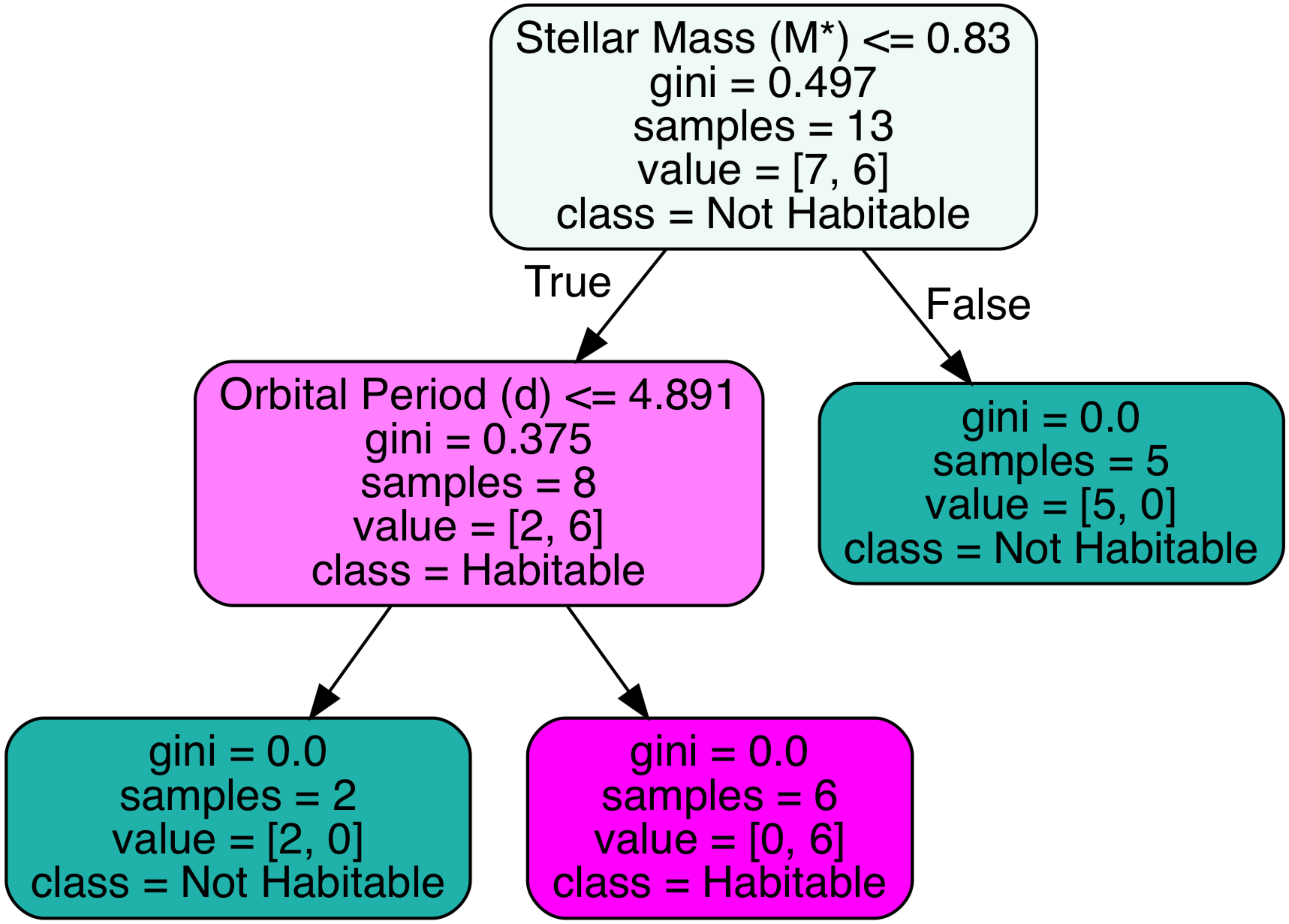
Note: **splits happen along features!**

Our first example will be a supervised classification problem, in which we are trying to decide if a planet is habitable, based on its distance from parent star, mass of parent star, and orbital period.

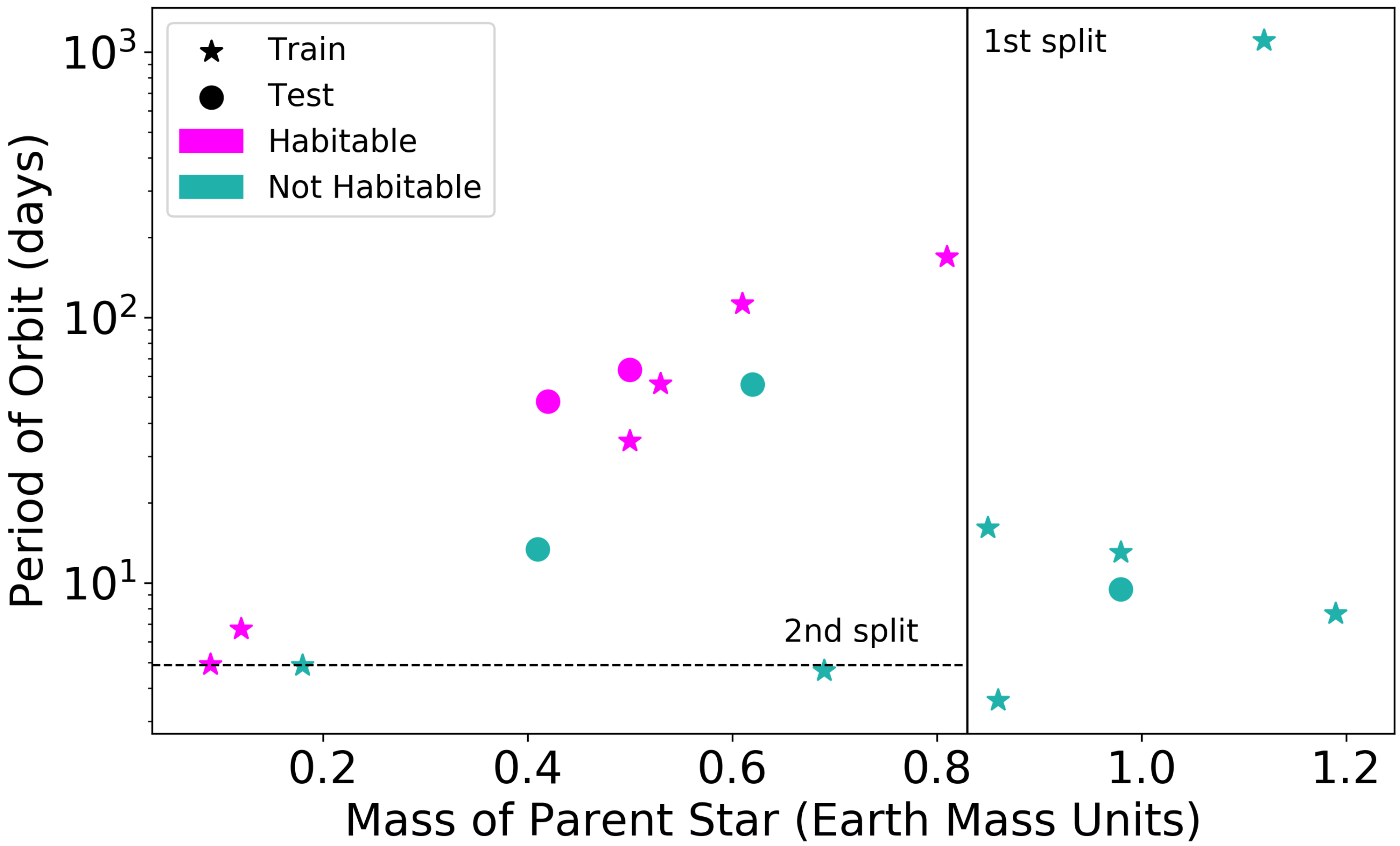
Name	Stellar Mass ($M_\odot$)	Orbital Period (days)	Distance (AU)	Habitable?
Kepler-736 b	0.86	3.60	0.0437	0
Kepler-636 b	0.85	16.08	0.1180	0
Kepler-887 c	1.19	7.64	0.0804	0
Kepler-442 b	0.61	112.30	0.4093	1
Kepler-772 b	0.98	12.99	0.1074	0
Teegarden's Star b	0.09	4.91	0.0252	1
K2-116 b	0.69	4.66	0.0481	0
GJ 1061 c	0.12	6.69	0.035	1
HD 68402 b	1.12	1103	2.1810	0
Kepler-1544 b	0.81	168.81	0.5571	1
Kepler-296 e	0.5	34.14	0.1782	1
Kepler-705 b	0.53	56.06	0.2319	1
Kepler-445 c	0.18	4.87	0.0317	0
HD 104067 b	0.62	55.81	0.26	[redacted]
GJ 4276 b	0.41	13.35	0.0876	[redacted]
Kepler-296 f	0.5	63.34	0.2689	[redacted]
Kepler-63 b	0.98	9.43	0.0881	[redacted]
GJ 3293 d	0.42	48.13	0.1953	[redacted]

Table 2.1: Learning set for the habitable planets problem.

Let’s see what sklearn says

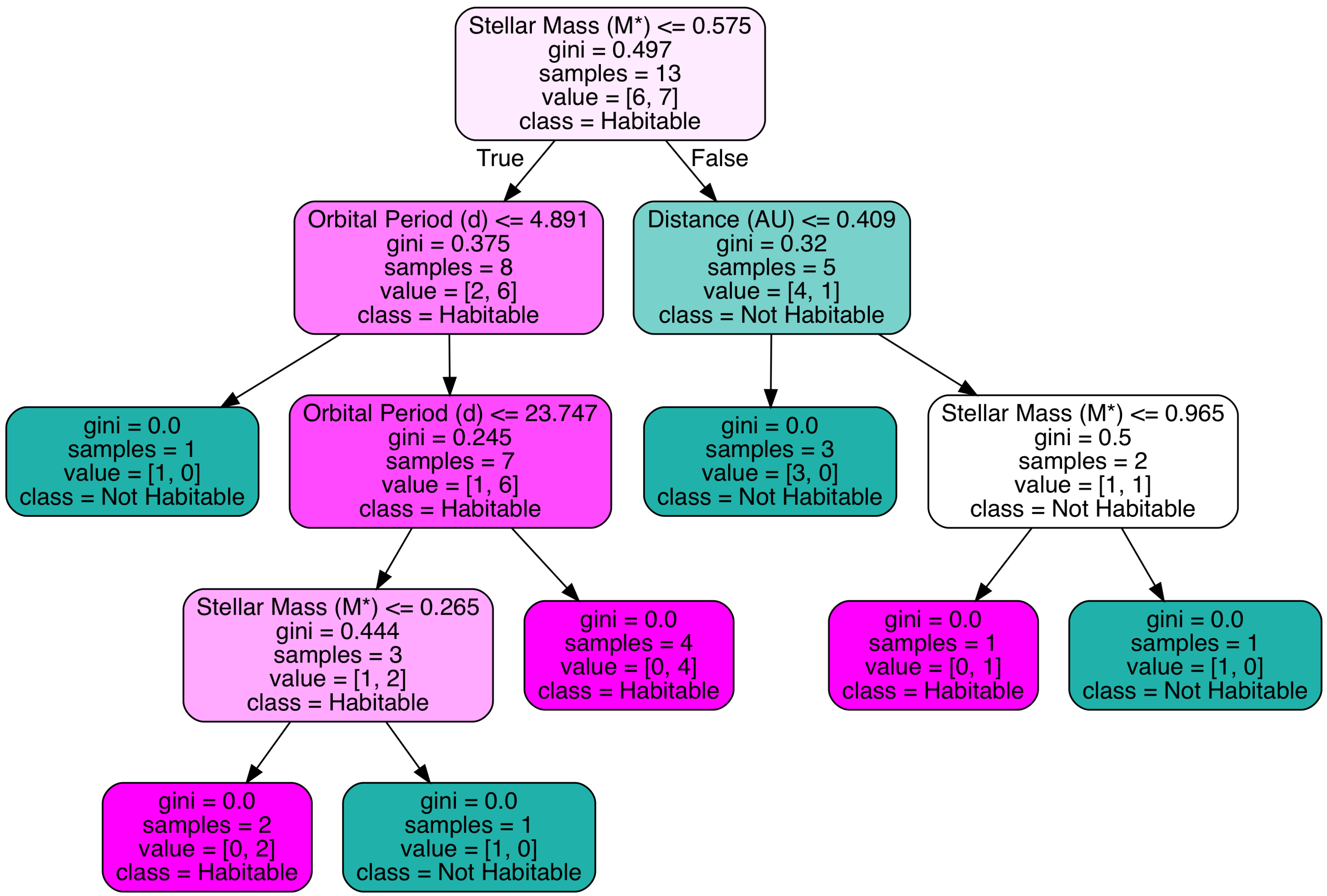


And visualize the criteria that we found!



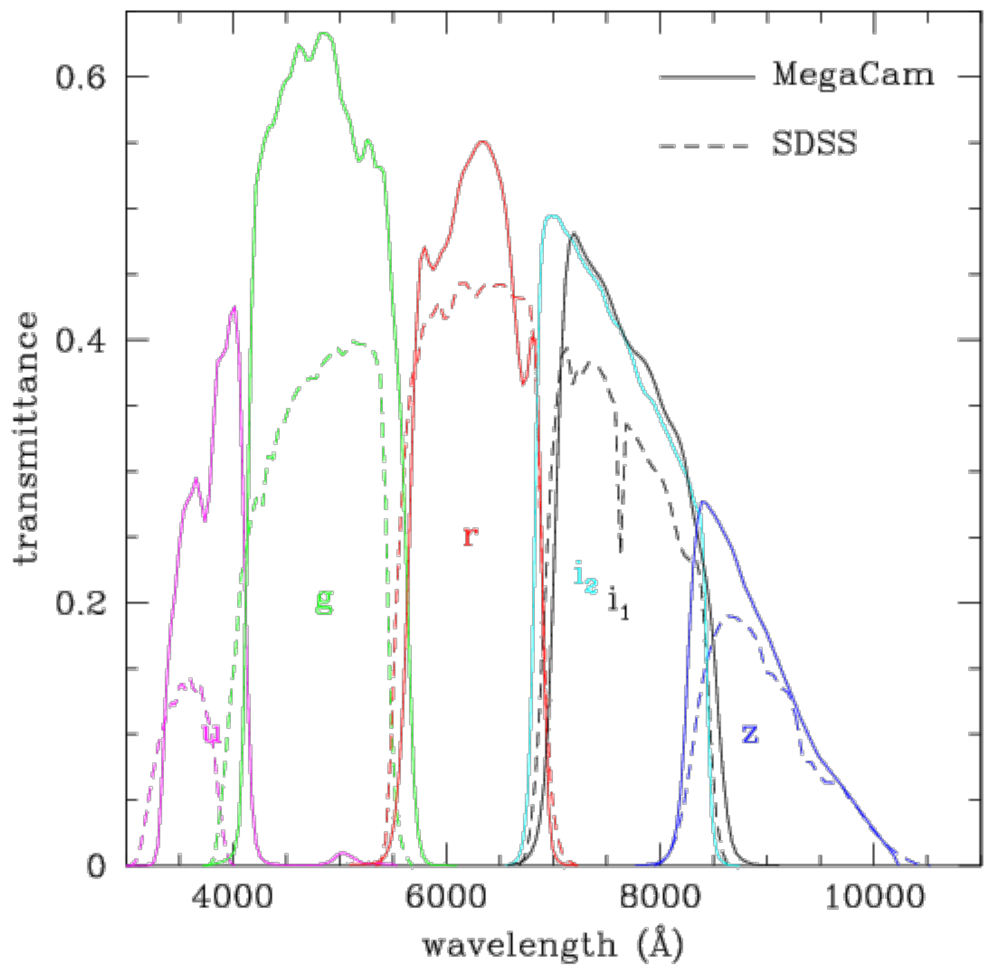
Can you figure out the accuracy on the test set?

Note (but you have to take my word for it): if you use the last 13 rows for training and the first 5 for testing, you get this tree:

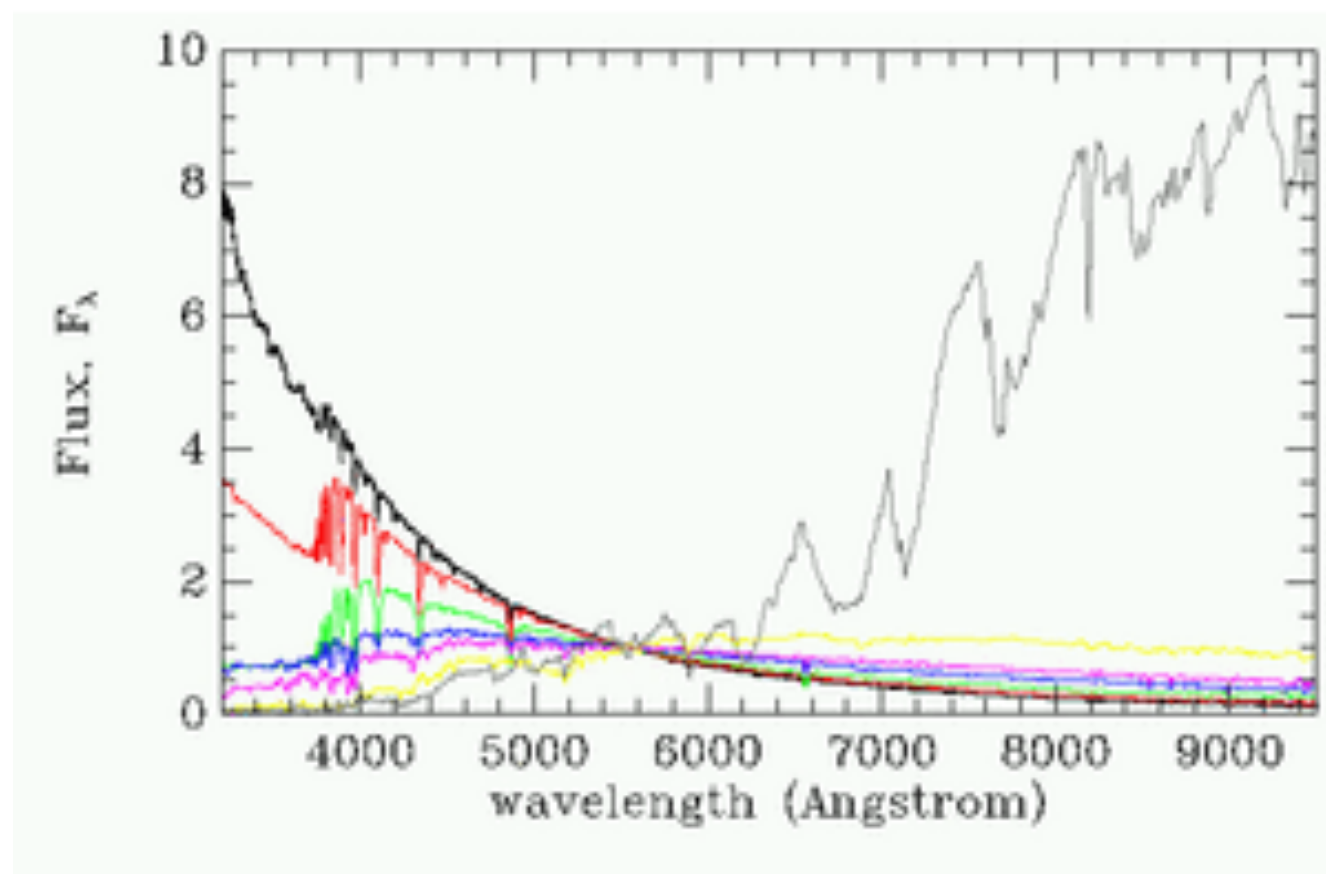


and 100% accuracy on test set (remember our instability problem?)

Coding exercise: supervised classification problem, in which we are trying to decide if a star is a variable star (RR Lyrae), based on imaging data in 5 bands (u, g, r, i, z) and four colors (u-g, g-r, r-i, i-z)



range of observed brightness



spectra of different stellar types