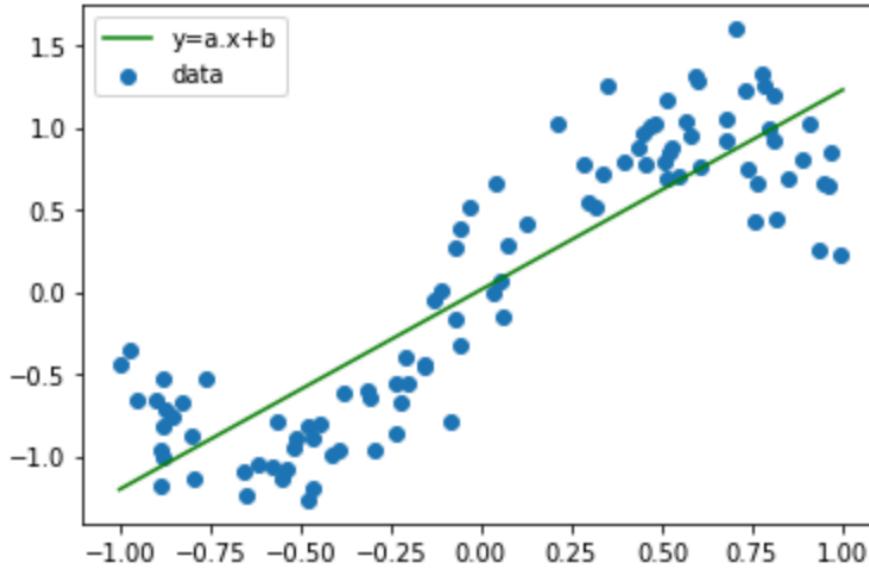


PART I: AN INTRODUCTION TO
“CLASSICAL” SUPERVISED MACHINE
LEARNING



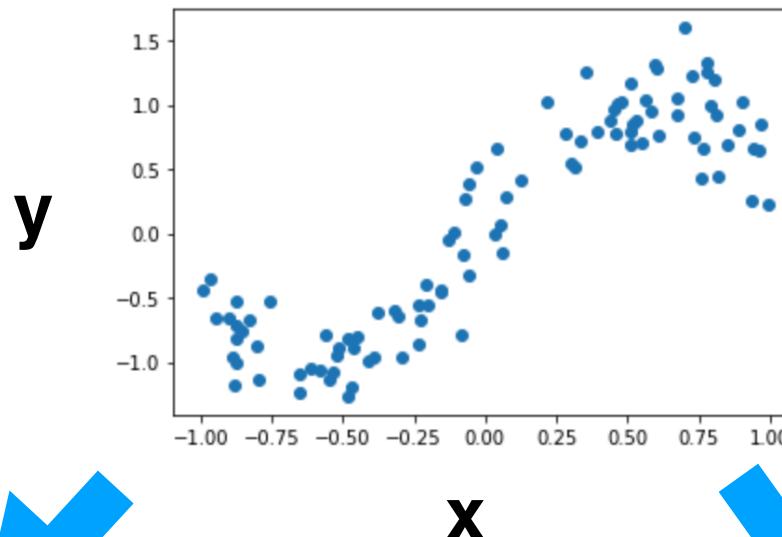
When poll is active, respond at pollev.com/marchuertasc257

Text **MARCHUERTASC257** to **22333** once to join

Is this linear fit machine learning?

- Yes
- No
- It depends
- I don't know

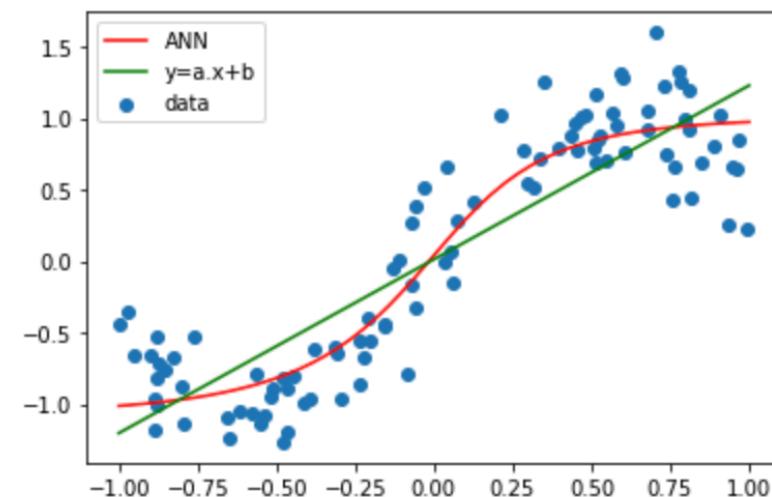
$$y=F(x,p)?$$



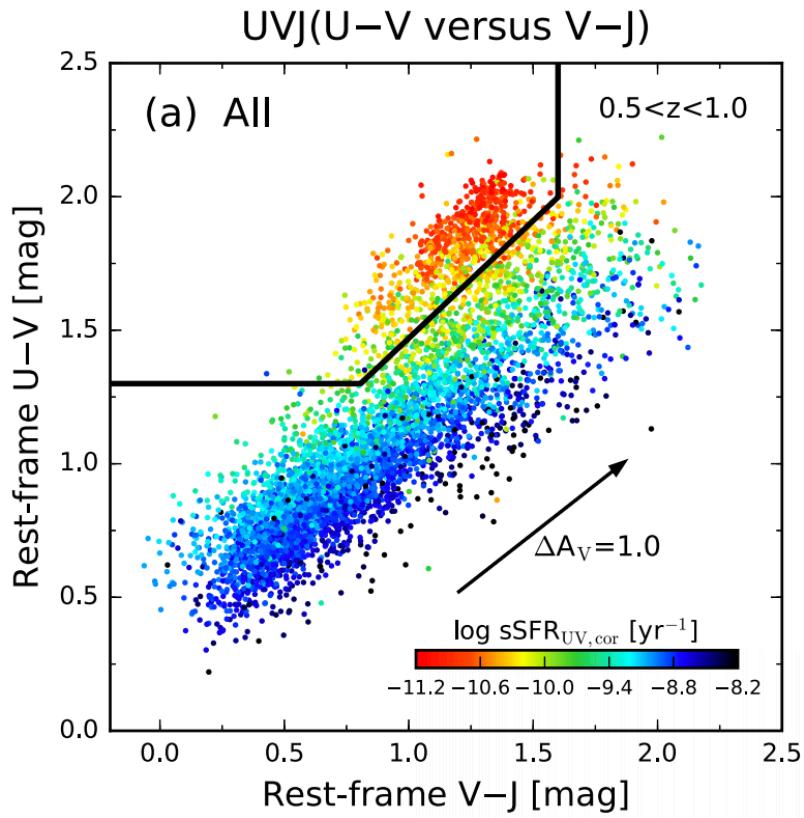
x



p is derived from physical insight



p is data driven, no physics



Liu+18

When poll is active, respond at pollev.com/marchuertasc257
Text **MARCHUERTASC257** to **22333** once to join

Is this color-color plot machine learning?

Yes

No

It depends

I don't know

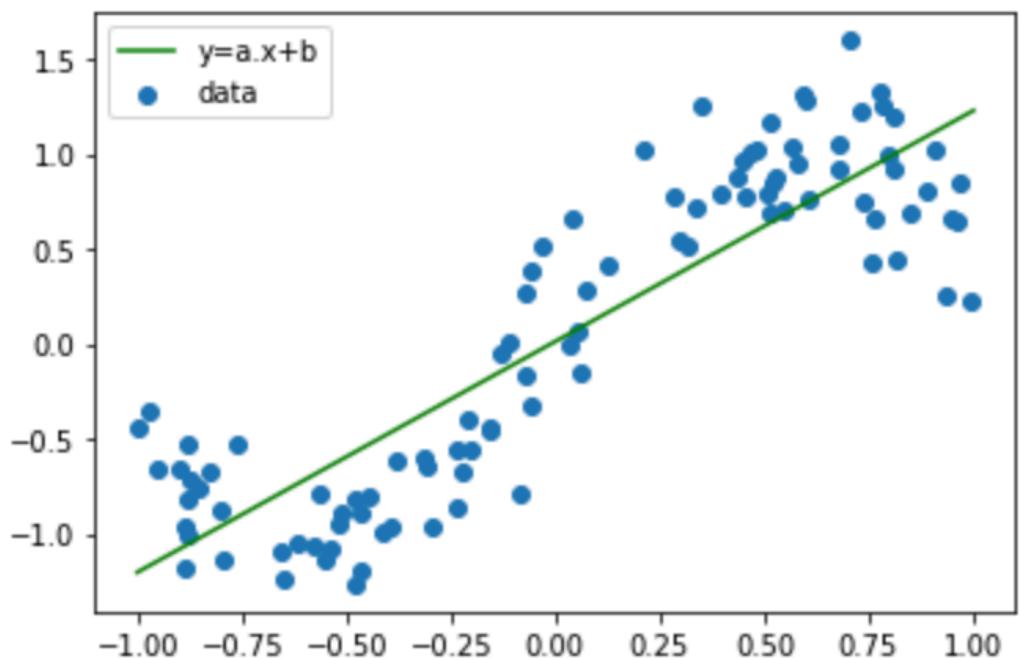
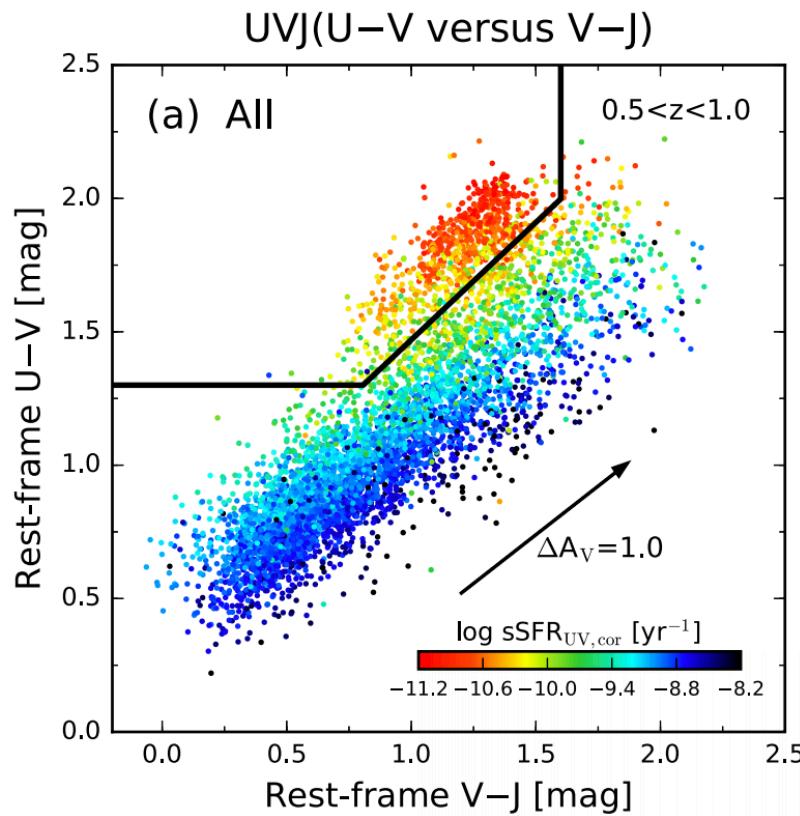
Powered by Poll Everywhere
Start the presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app

Supervised Machine Learning:

$$f_W(\vec{x}) = \vec{y}$$

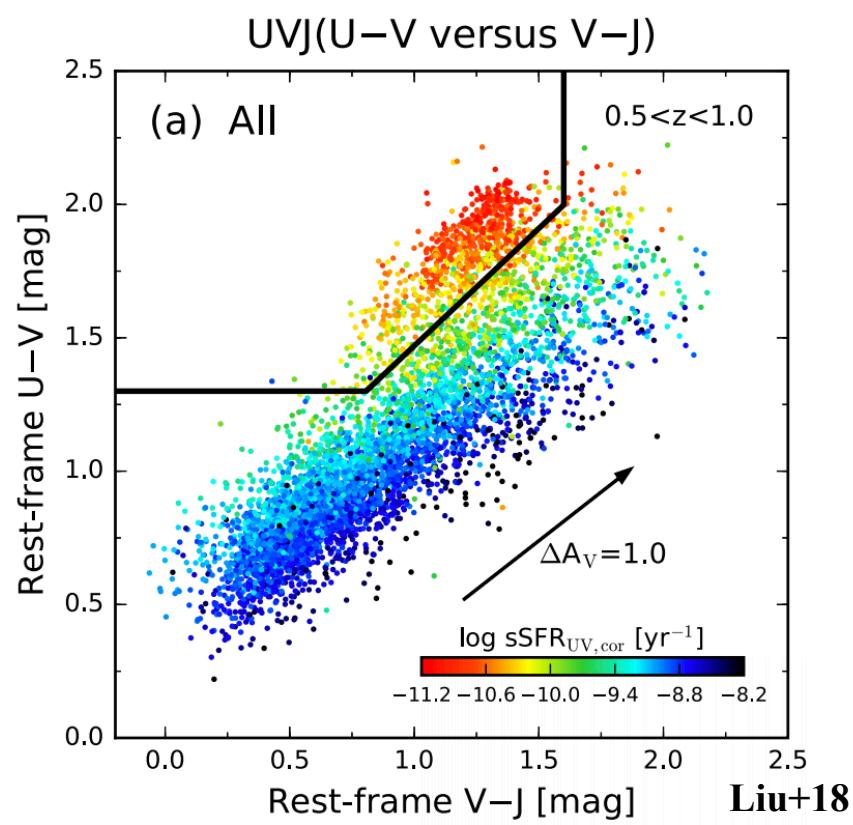


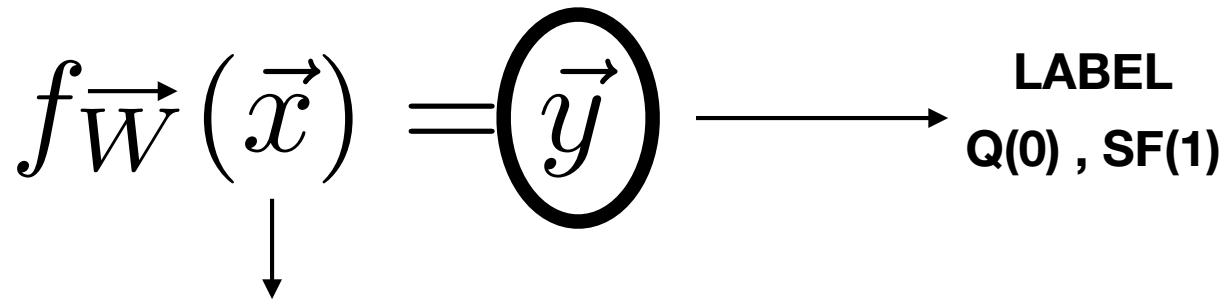
If y is discrete: **classification**
If y is a real number: **regression**



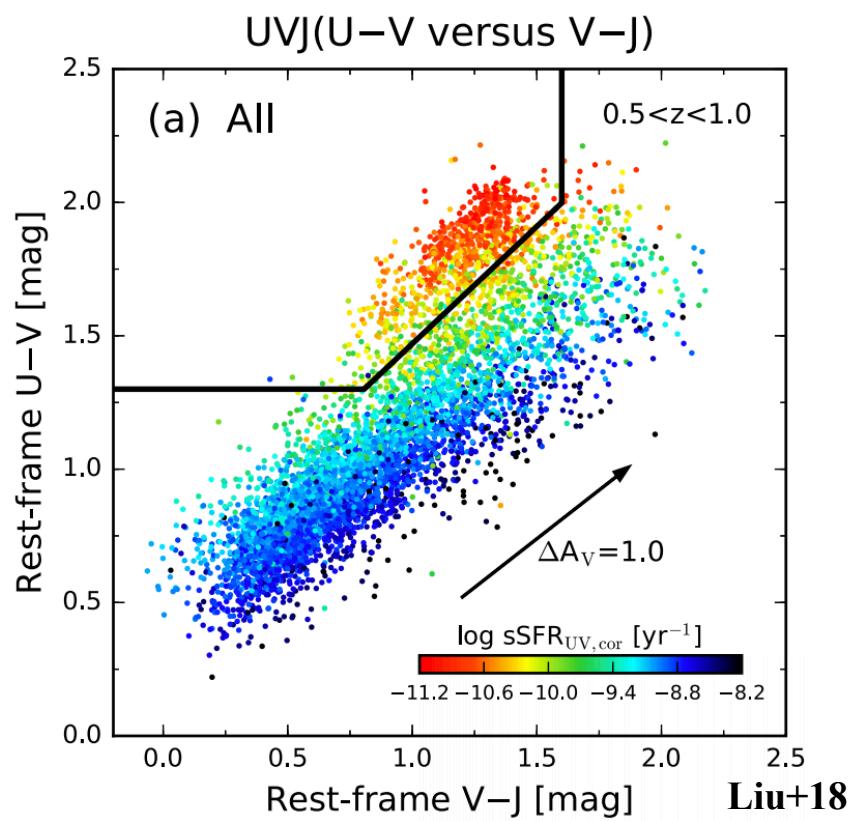
$$f_W(\vec{x}) = \vec{y}$$

LABEL
Q , SF





(U-V, V-J) FEATURES



$$f_W(\vec{x}) = \vec{y} \longrightarrow \text{LABEL}$$

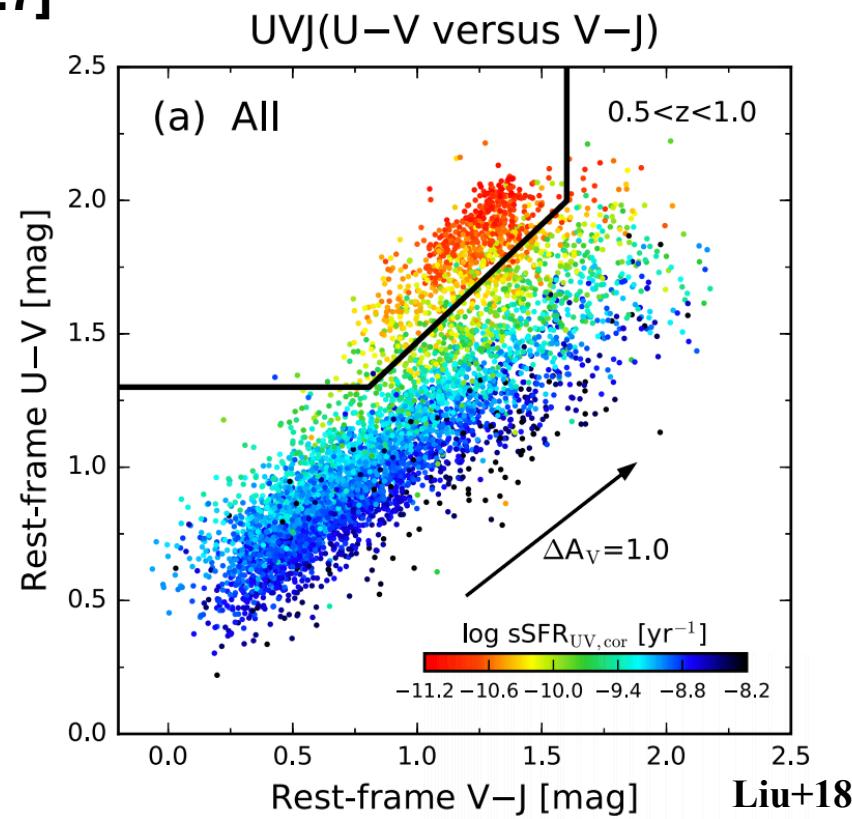
Q(0) , SF(1)

NETWORK FUNCTION

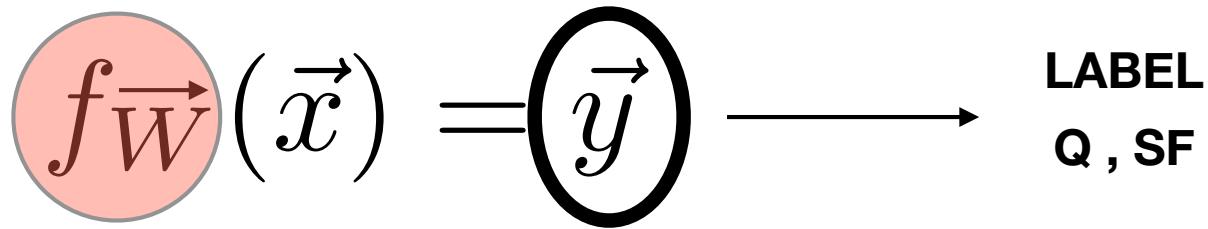
(U-V, V-J) FEATURES

$$\text{sgn}[(u-v)-0.8*(v-j)-0.7]$$

WEIGHTS



“CLASSICAL”
MACHINE LEARNING



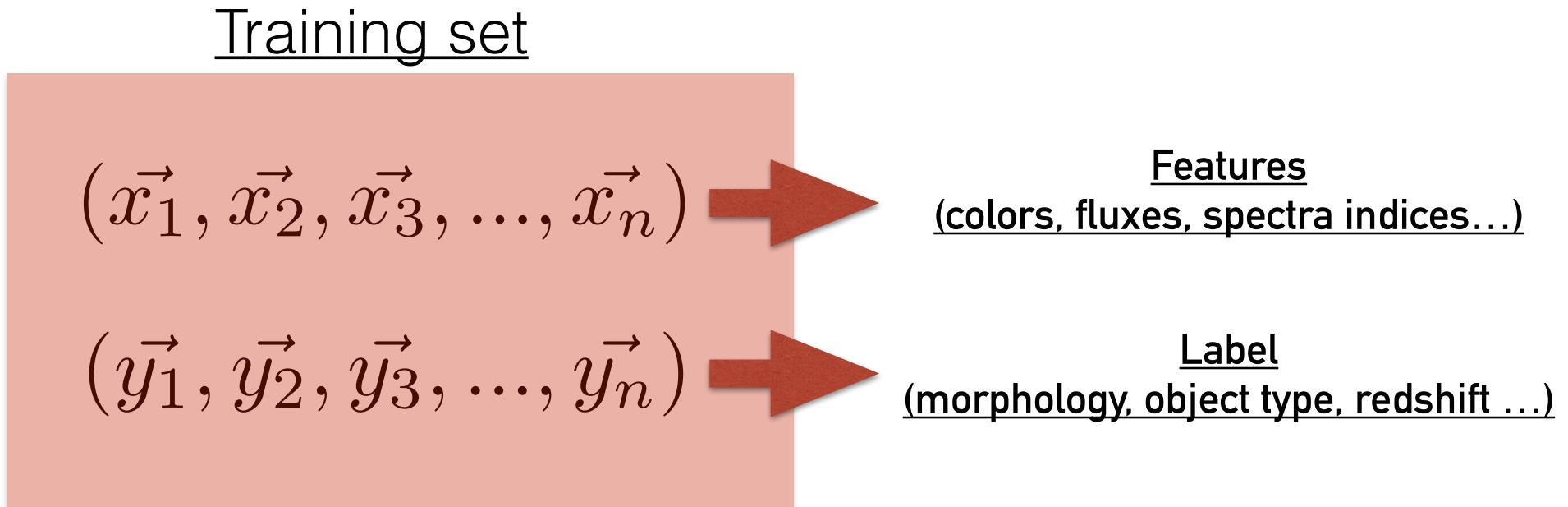
$\text{sgn}[(u-v)-W_1*(v-j)-W_2]$



**REPLACE THIS BY A GENERAL
NON LINEAR FUNCTION WITH SOME PARAMETERS W**

SUPERVISED LEARNING

Given a dataset with known labels - find a function that can assign (predict) labels for an unlabelled dataset using a set of features (measurements)



SUPERVISED LEARNING

Given a dataset with known labels - find a function that can assign (predict) labels for an unlabelled dataset using a set of features (measurements)

Training set

$$(\vec{x}_1, \vec{x}_2, \vec{x}_3, \dots, \vec{x}_n)$$

$$(\vec{y}_1, \vec{y}_2, \vec{y}_3, \dots, \vec{y}_n)$$

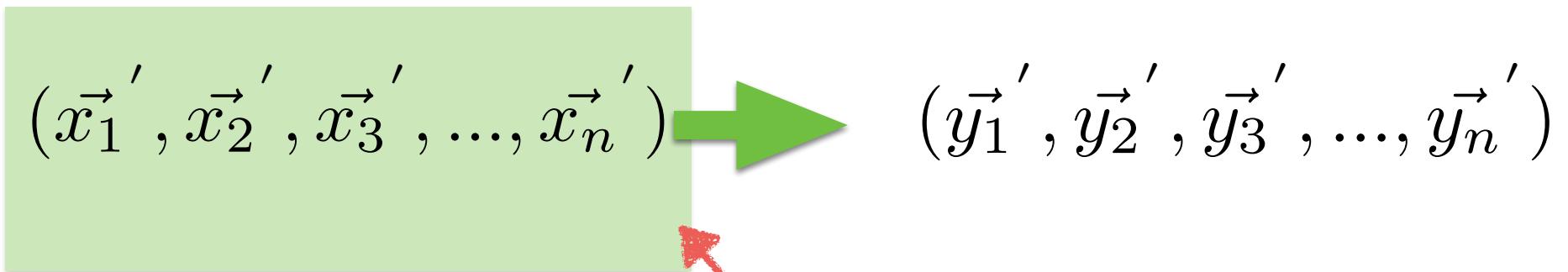


$$f_W(\vec{x}) = \vec{y}$$

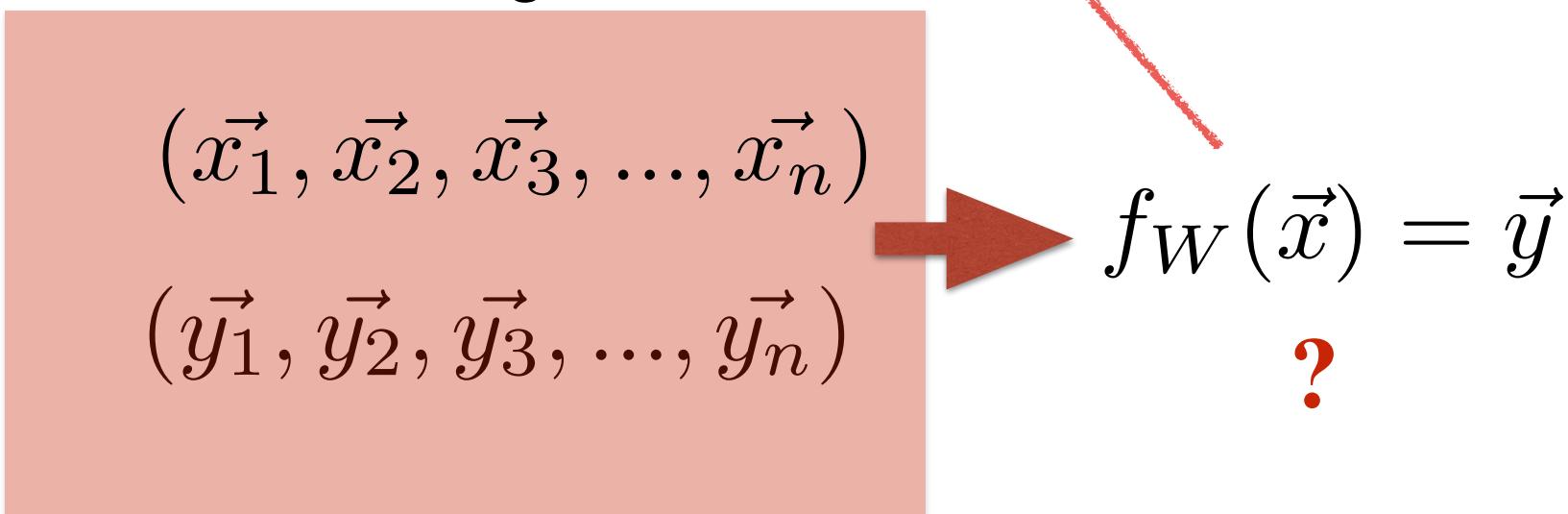
?

SUPERVISED LEARNING

Unlabelled set



Training set



$$(\vec{x}_1, \vec{x}_2, \vec{x}_3, \dots, \vec{x}_n)$$

$$\vec{x} \in \mathbb{R}^d$$

$$(\vec{y}_1, \vec{y}_2, \vec{y}_3, \dots, \vec{y}_n)$$

$$\vec{y} \in \mathbb{R} \quad \vec{y} \in \mathbb{N}$$

GENERAL GOAL: Find a (non-linear) function that outputs the correct class / label (y) for a given input object:



It is translated into a minimization problem : find \mathbf{W} such as the prediction error is minimal over all unseen vectors

Different types of supervised machine learning methods

RANDOM FORESTS

CARTS

decision trees

Gradient based algorithms

ARTIFICAL
NEURAL NETWORKS
(DEEP LEARNING)

SUPPORT VECTOR MACHINES

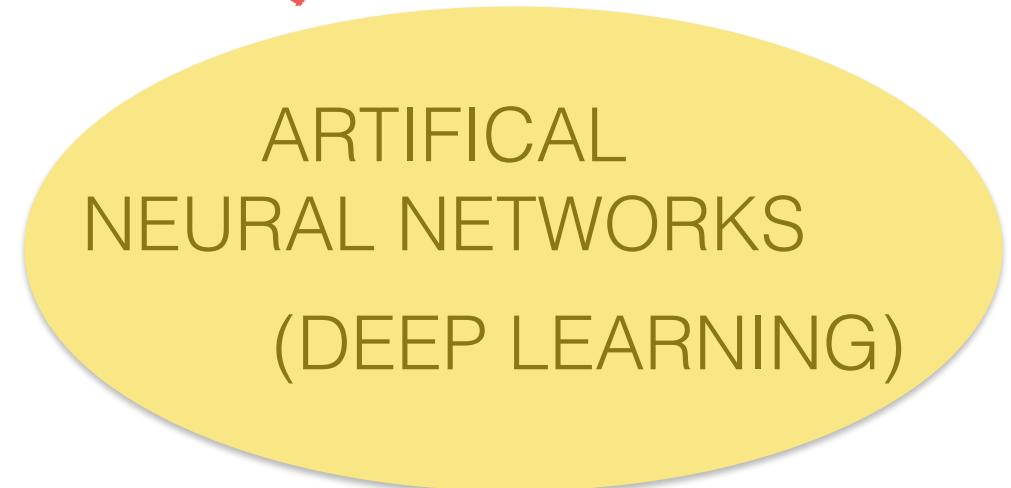
kernel algorithms

The differences are
in the function
that is used

$$f_W(\vec{x})$$



decision trees



Gradient based algorithms



kernel algorithms

TO SUMMARISE: we need two key elements

1. **A LOSS FUNCTION**
(OBJECTIVE FUNCTION TO MINIMISE)
2. **A MINIMISATION OR OPTIMISATION**
ALGORITHM

TO SUMMARISE: we need
two key elements

1. A LOSS FUNCTION

2. A MINIMISATION OR OPTIMISATION
ALGORITHM

THIS IS COMMON TO ALL MACHINE LEARNING
ALGORITHMS

FOR EXAMPLE:

1. DEFINE A LOSS FUNCTION

$$loss(F_W(\cdot), \vec{x}_i, \vec{y}_i)$$

For example: $(F_W(\vec{x}_i) - \vec{y}_i)^2$ (MSE LOSS FUNCTION)

2. MINIMISE THE EMPIRICAL RISK WITH OPTIMISATION

$$\mathfrak{R}_{empirical}(W) = \frac{1}{N} \sum_i^N [loss(W, \vec{x}, \vec{y})]$$



MINIMISE THE RISK

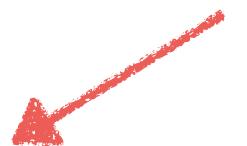
EMPIRICAL RISK?

$$\mathfrak{R}_{\text{empirical}}(W) = \frac{1}{N} \sum_i^N [\text{loss}(W, \vec{x}, \vec{y})]$$

WE ARE MINIMISING WITH RESPECT TO A FINITE NUMBER OF OBSERVED EXAMPLES

EMPIRICAL RISK?

$$\mathfrak{R}_{\text{empirical}}(W) = \frac{1}{N} \sum_i^N [\text{loss}(W, \vec{x}, \vec{y})]$$



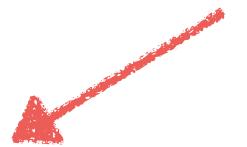
WE ARE MINIMISING WITH RESPECT TO A FINITE NUMBER OF OBSERVED EXAMPLES

OBSERVED DATASET



EMPIRICAL RISK?

$$\mathfrak{R}_{\text{empirical}}(W) = \frac{1}{N} \sum_i^N [\text{loss}(W, \vec{x}, \vec{y})]$$



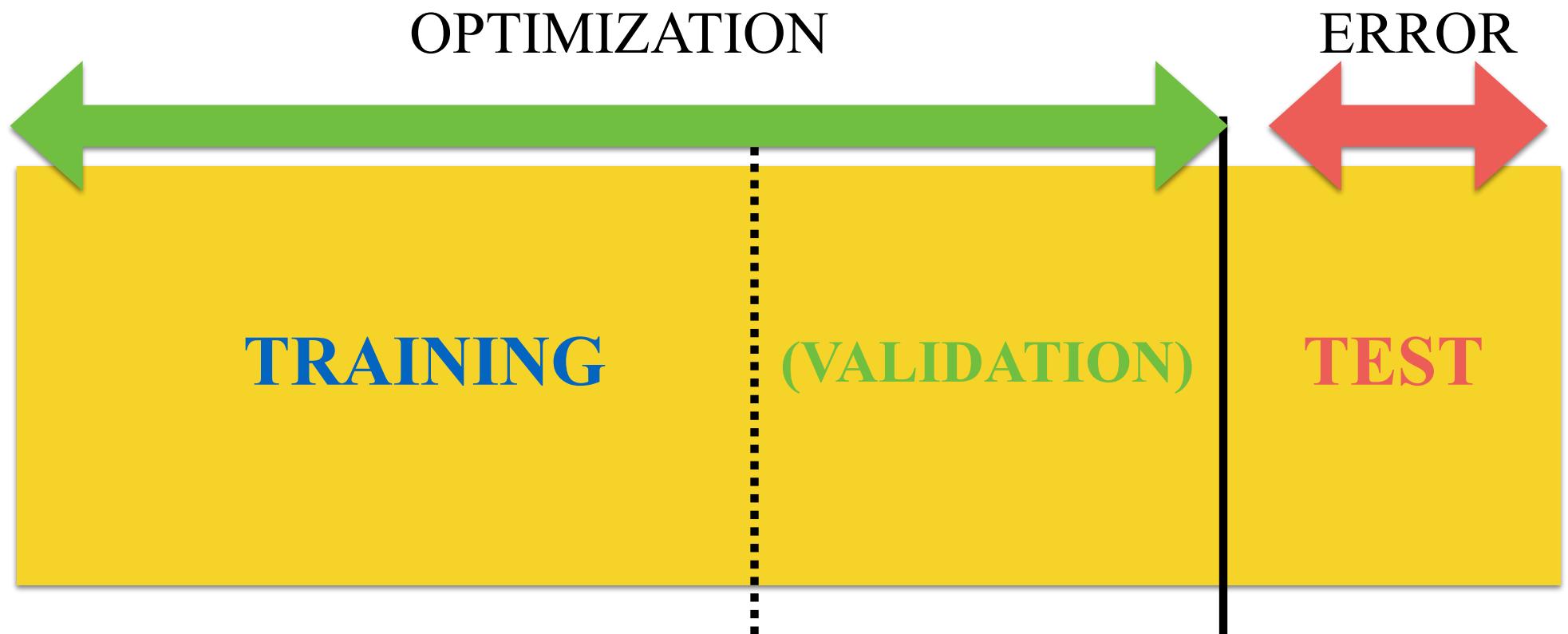
WE ARE MINIMISING WITH RESPECT TO A FINITE NUMBER OF OBSERVED EXAMPLES

ALL “GALAXIES IN THE UNIVERSE”

OBSERVED DATASET



IN PRACTICE

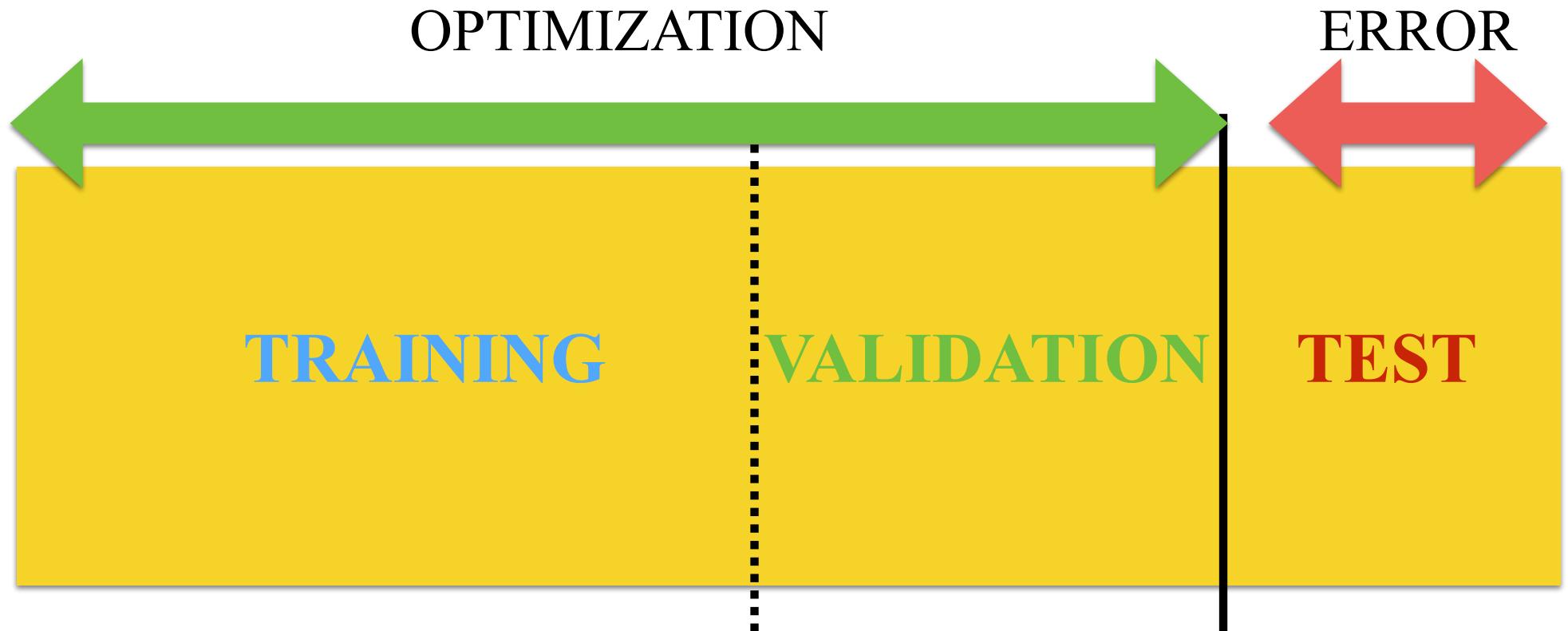


training set: use to train the classifier

validation set: use to monitor performance in real time - check
for overfitting

test set: use to train the classifier

IN PRACTICE



NO CHEATING! NEVER USE TRAINING TO VALIDATE
YOUR ALGORITHM!

The algorithm used to minimise is
called OPTIMISATION

THERE ARE SEVERAL OPTIMISATION TECHNIQUES

Optimisation

THERE ARE SEVERAL OPTIMISATION TECHNIQUES

THEY DEPEND ON THE MACHINE LEARNING ALGORITHM

Optimisation

THERE ARE SEVERAL OPTIMISATION TECHNIQUES

THEY DEPEND ON THE MACHINE LEARNING ALGORITHM

NEURAL NETWORKS USE THE GRADIENT DESCENT AS WE
WILL SEE LATER

$$W_{t+1} = W_t - \lambda_h \nabla f(W_t)$$

weights to be learned



epoch

learning rate
(hyper parameter)

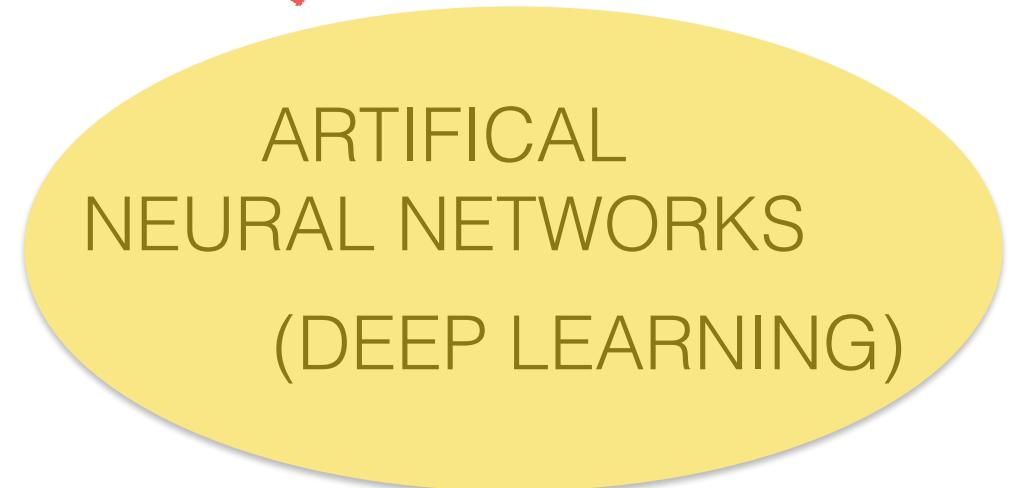


The differences are
in the function
that is used

$$f_W(\vec{x})$$



decision trees



Gradient based algorithms



kernel algorithms

DECISION TREES ALGORITHMS

DECISION TREES

CARTS

Classification Trees

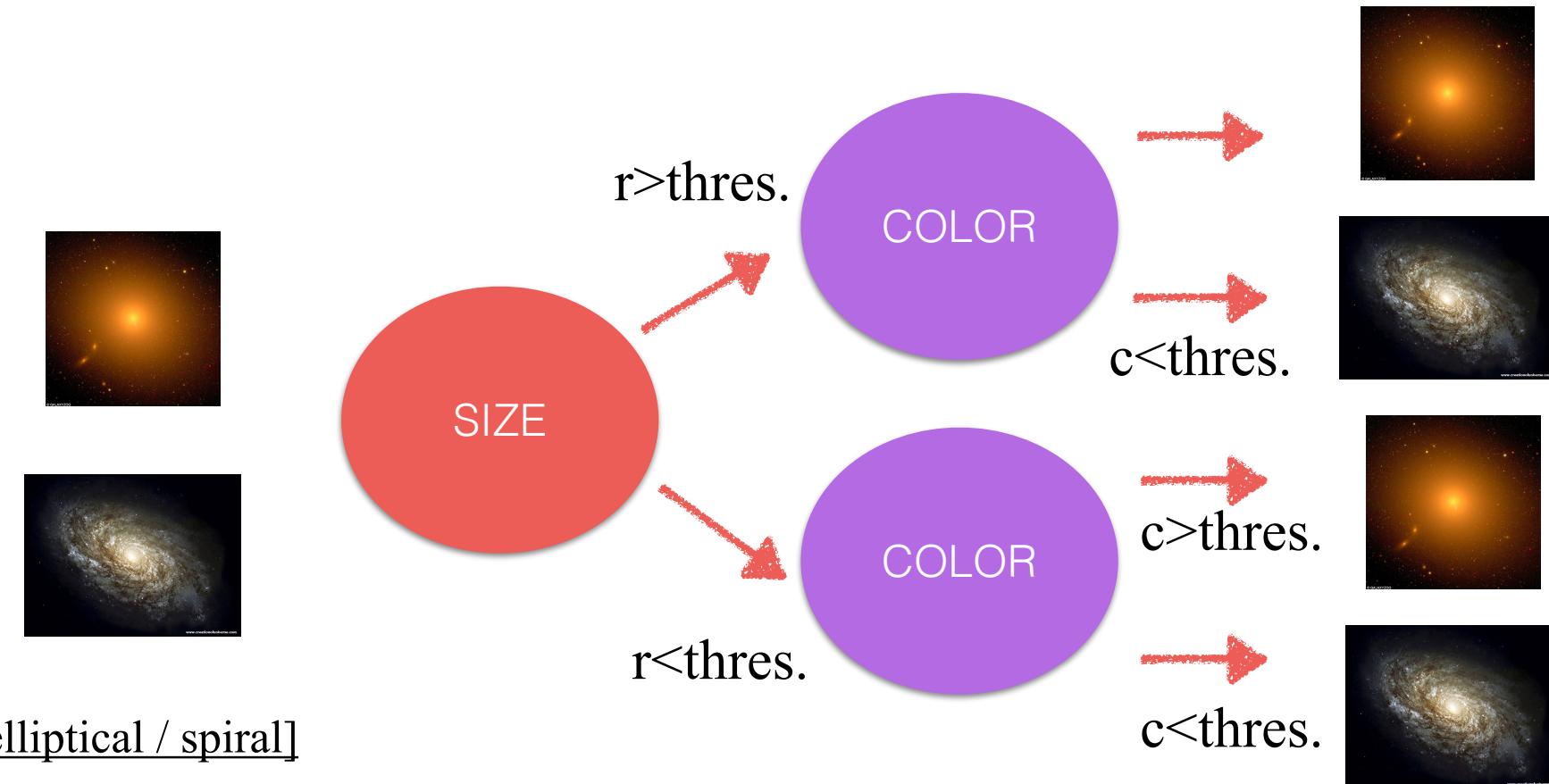
Regression Trees

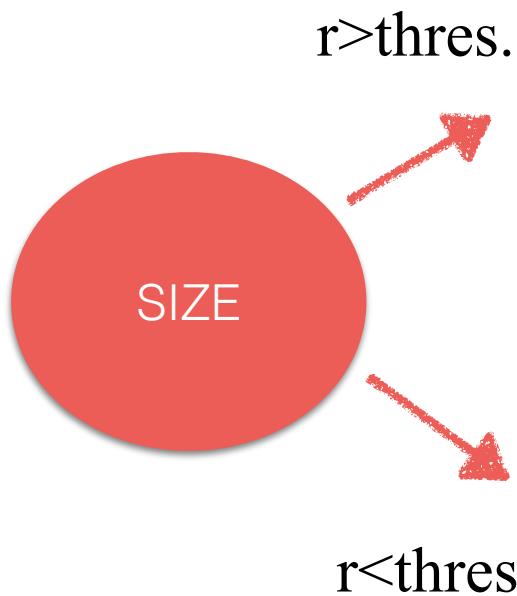
BOOSTED TREES

Random Forests

CLASSIFICATION AND REGRESSION TREES (CARTS)

THIS IS THE SIMPLEST AND MORE INTUITIVE MACHINE LEARNING ALGORITHM





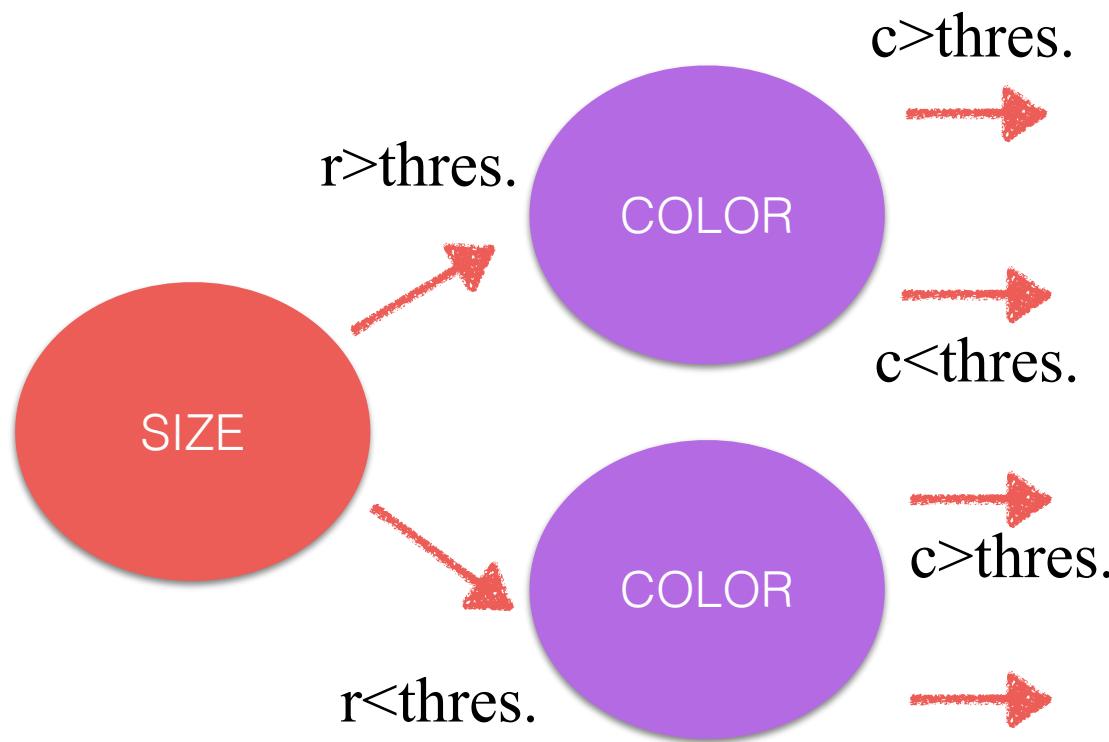
[elliptical / spiral]

THE MINIMISATION ALGORITHM IS AS FOLLOWS:

- 1. FROM THE INPUT PARAMETERS, FIRST FIND THE PROPERTY THAT BEST SPLITS INTO 2 GROUPS [I.E. MINIMIZES SOME LOSS FUNCTION]**
- 2. REPEAT STEP 1 WITH ANOTHER PARAMETER**
- 3. AT THE END THERE IS A TREE WHERE, AT EACH POINT, ONE OF TWO DECISIONS CAN BE MADE**

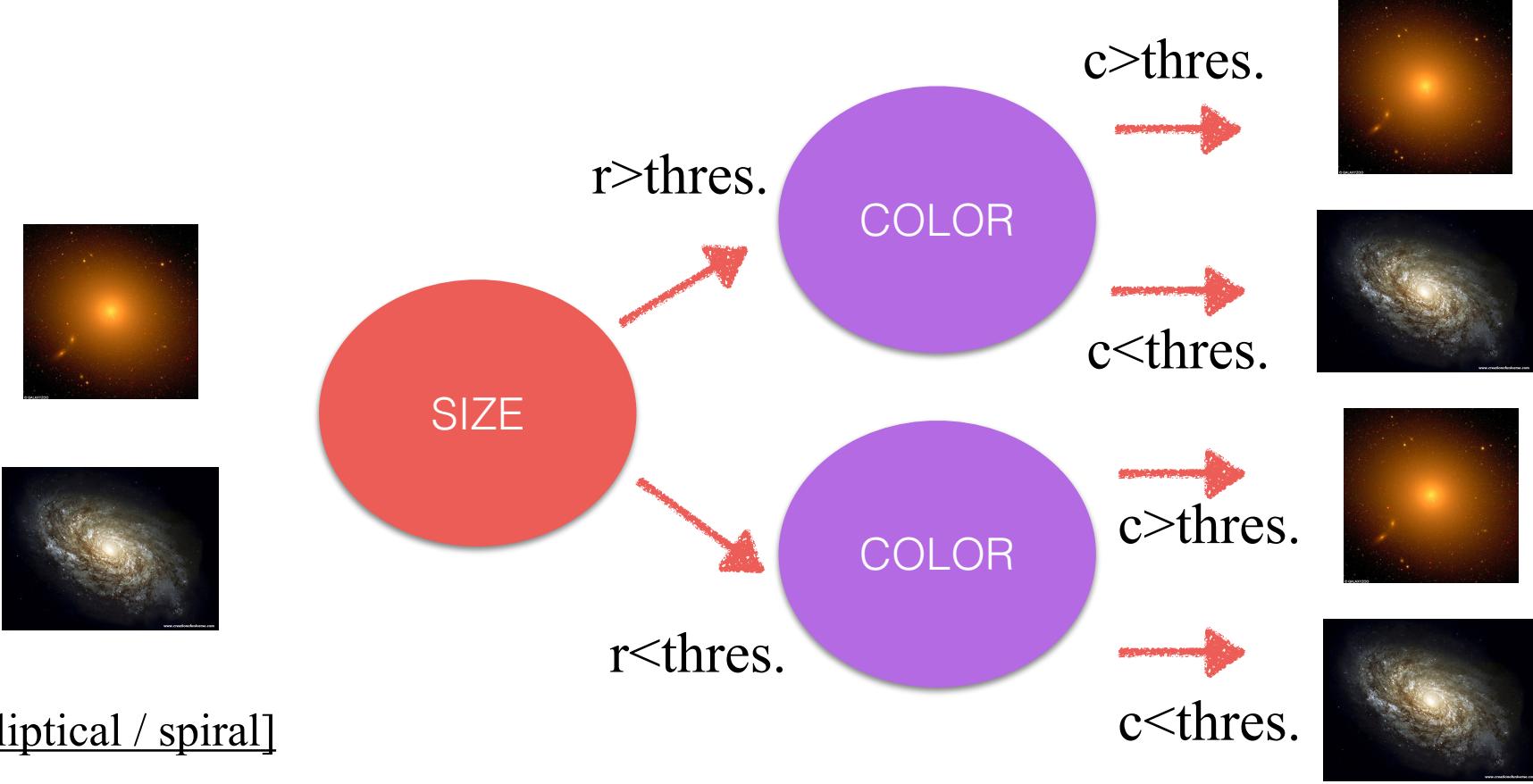


[elliptical / spiral]



THE MINIMISATION ALGORITHM IS AS FOLLOWS:

1. FROM THE INPUT PARAMETERS, FIRST FIND THE PROPERTY THAT BEST SPLITS INTO 2 GROUPS [I.E. **MINIMIZES SOME LOSS FUNCTION**]
- 2. REPEAT STEP 1 WITH ANOTHER PARAMETER**
3. AT THE END THERE IS A TREE WHERE, AT EACH POINT, ONE OF TWO DECISIONS CAN BE MADE



THE MINIMISATION ALGORITHM IS AS FOLLOWS:

1. FROM THE INPUT PARAMETERS, FIRST FIND THE PROPERTY THAT BEST SPLITS INTO 2 GROUPS [I.E. **MINIMIZES SOME LOSS FUNCTION**]
2. REPEAT STEP 1 WITH ANOTHER PARAMETER
3. AT THE END THERE IS A TREE WHERE, AT EACH POINT, ONE OF TWO DECISIONS CAN BE MADE

DECISION TREES (CARTS)

TYPICAL METRICS USED:

[THE IDEA IS TO FIND THE SPLITTING VALUE THAT PUTS ALL OBJECTS OF A GIVEN CLASS IN ONE LEAF]

Decision trees (CARTS)

TYPICAL LOSS FUNCTION USED:

GINI IMPURITY

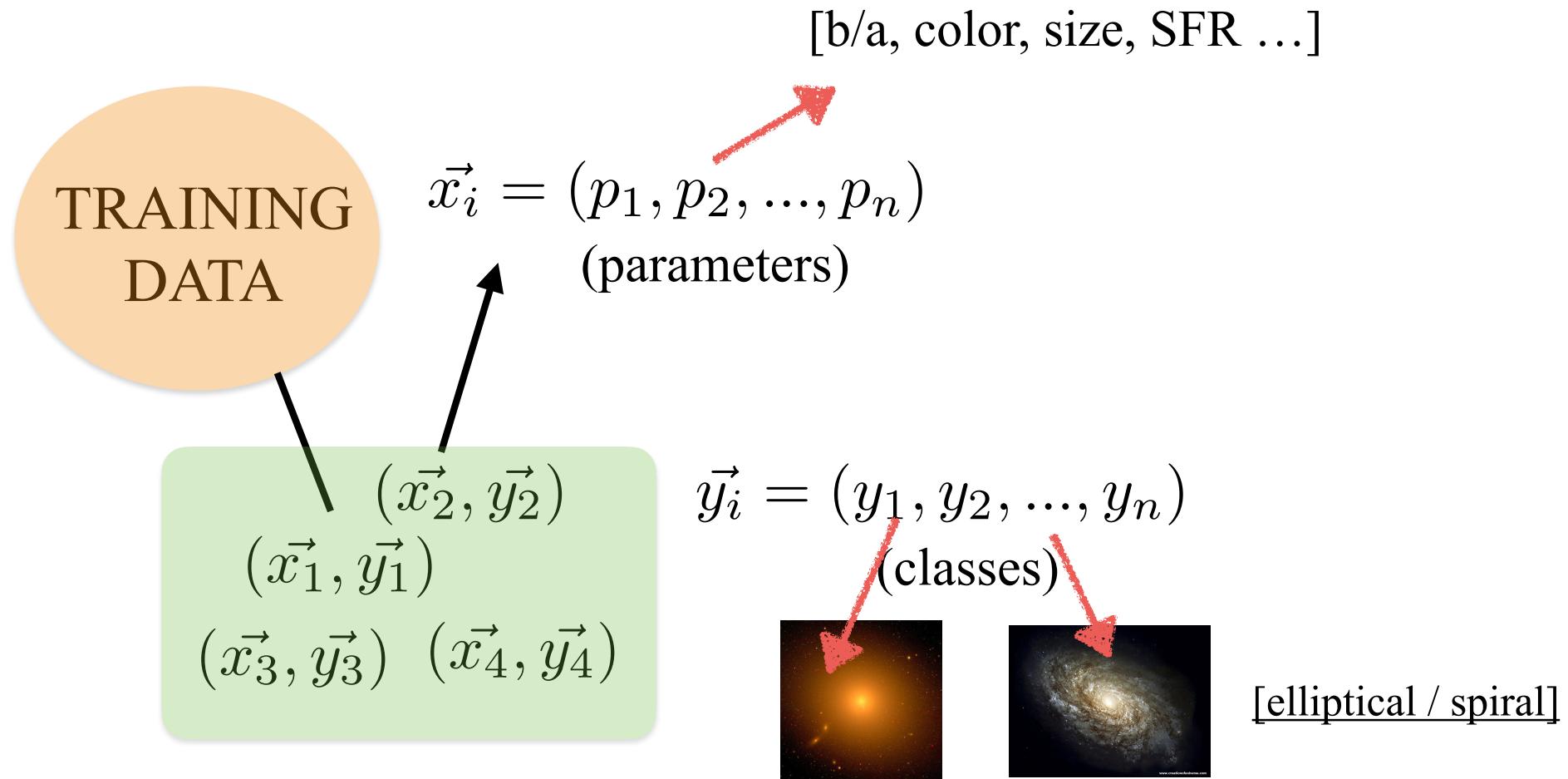
A PERFECT SPLIT GIVES $G=0$,
SO WE WANT TO MINIMISE G

$$\sum_{j=1}^c p_j \times (1 - p_j)$$

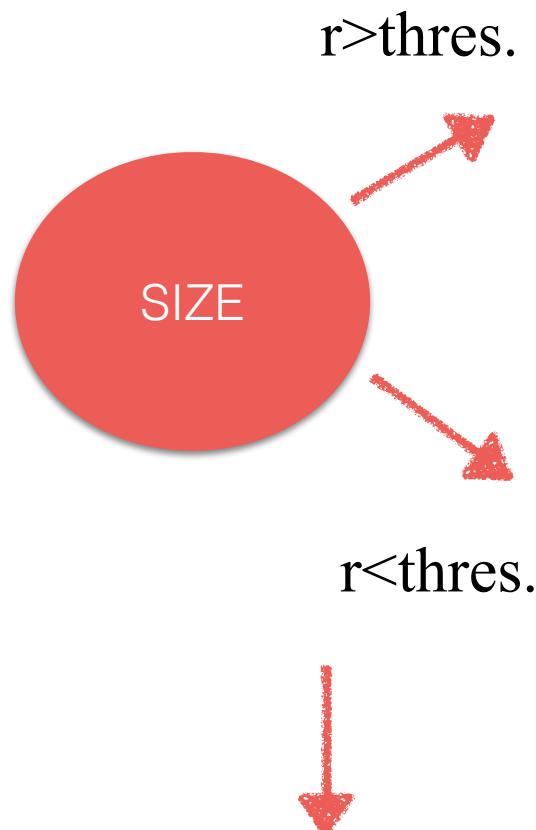


fraction of
objects in each class given
a split value

DECISION TREES (CARTS)



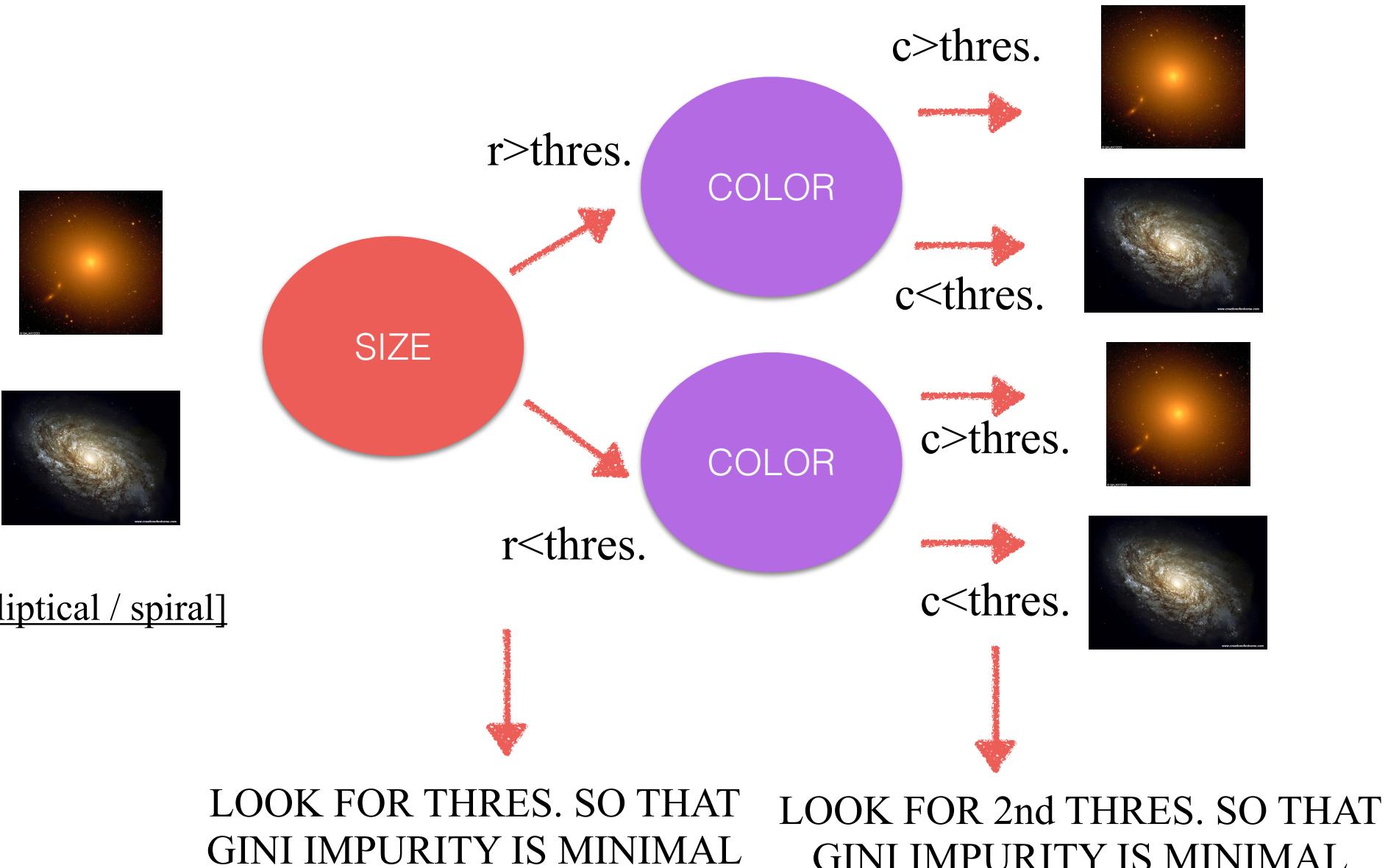
DECISION TREES (CARTS)



[elliptical / spiral]

LOOK FOR THRES. SO THAT
GINI IMPURITY IS MINIMAL

DECISION TREES (CARTS)



Let's assume the following sample of galaxies:

Label

Size



3 kpc



10 kpc



4 kpc



8 kpc



4 kpc



9 kpc

When poll is active, respond at pollev.com/marchuertasc257

What would be the Gini Impurity with a split value of 6 kpc



None of the above

Powered by  Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app

Label

Size



3 kpc



10 kpc



4 kpc



8 kpc



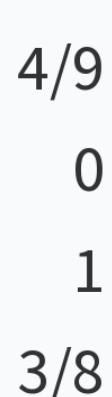
4 kpc



9 kpc

When poll is active, respond at pollev.com/marchuertasc257

What would be the Gini Impurity with a split value of 6 kpc

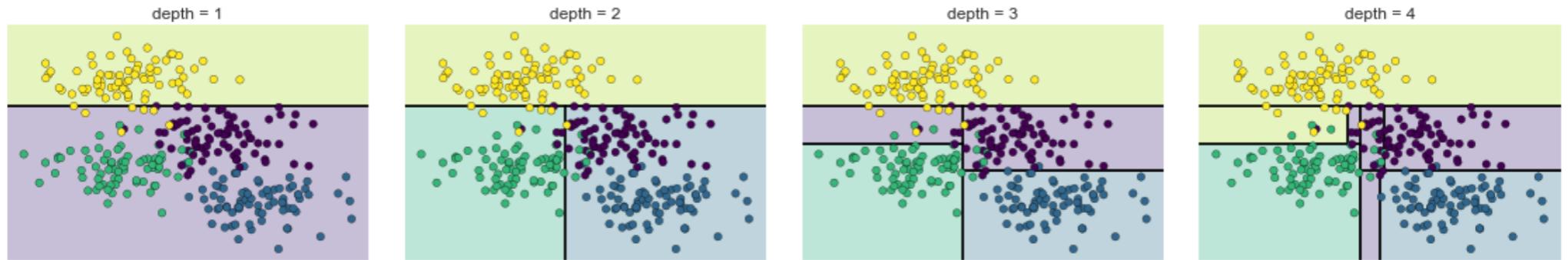


None of the above

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app

DECISION TREES (CARTS)



IT IS SIMPLY A PARTITION OF THE PARAMETER SPACE WITH CONSTANT BOUNDARIES - THE NUMBER OF BOUNDARIES DEPENDS ON THE DEPTH OF THE ALGORITHM (HYPER-PARAMETER)

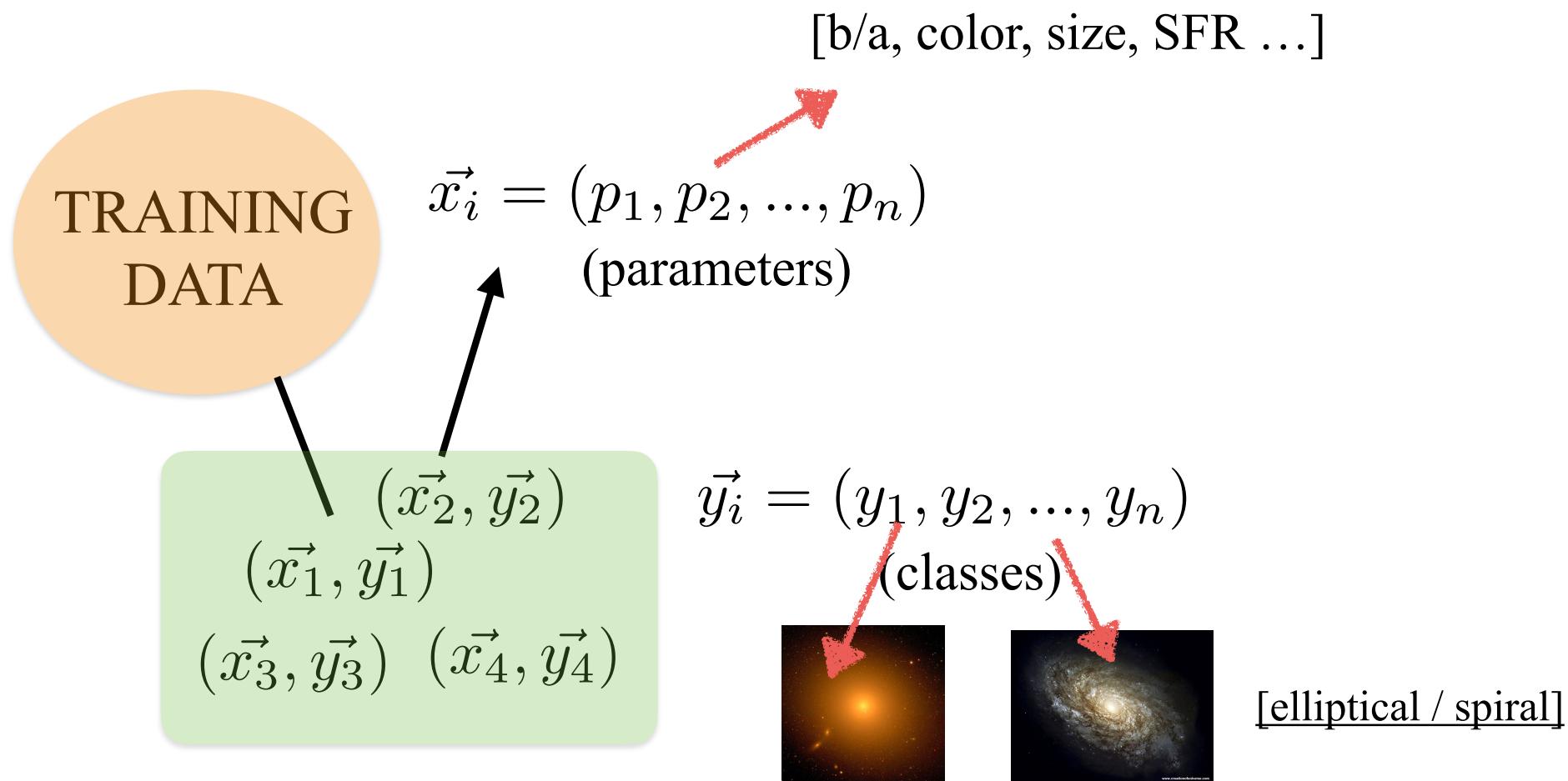
RANDOM FORESTS

ONE PROBLEM WITH CLASSIFICATION TREES IS THAT
THEY CAN EASILY OVERFIT

THE DECISIONS ARE VERY SPECIFIC TO THE TRAINING
SET AND NOT REPRESENTATIVE OF THE FULL
POPULATION

RANDOM FORESTS

RANDOM FORESTS TRY TO SOLVE THIS PROBLEM BY INTRODUCING SOME RANDOM INFORMATION IN THE TRAINING PROCESS



RANDOM FORESTS

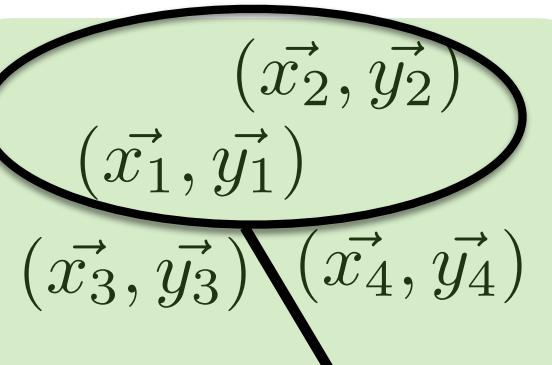
(\vec{x}_2, \vec{y}_2)

(\vec{x}_1, \vec{y}_1)

(\vec{x}_3, \vec{y}_3) (\vec{x}_4, \vec{y}_4)

$\vec{x}_i = (p_1, p_2, \dots, p_n)$

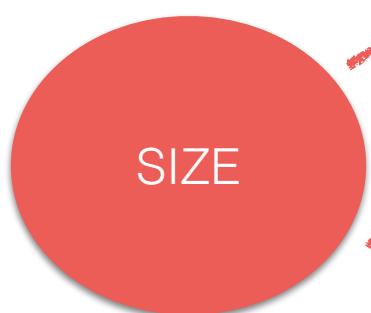
RANDOM FORESTS



$$\vec{x}_i = (p_1, p_2, \dots, p_n)$$



[elliptical / spiral]



$r > \text{thres.}$

$r < \text{thres.}$



$c > \text{thres.}$

$c < \text{thres.}$



$c > \text{thres.}$

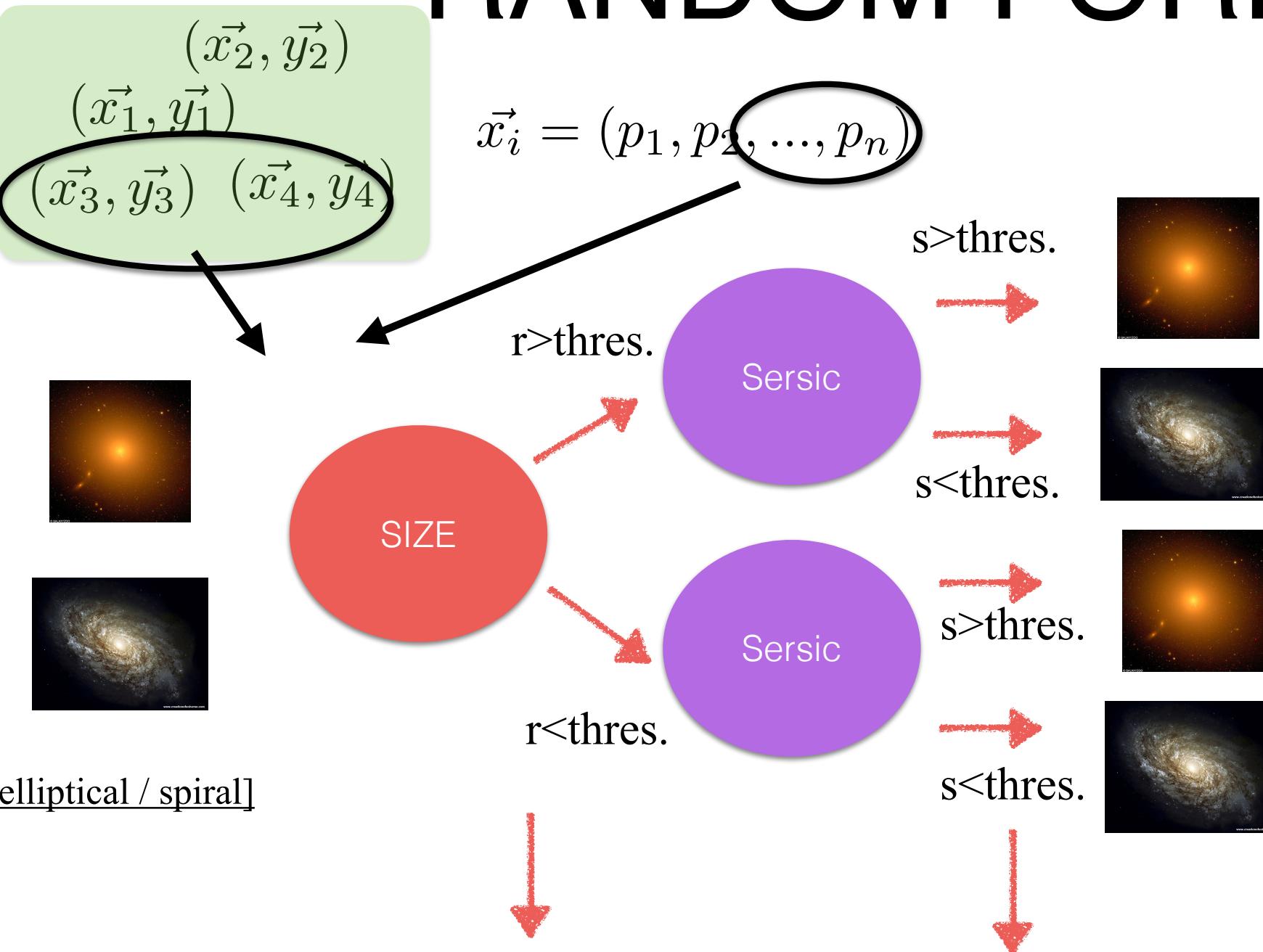
$c < \text{thres.}$



LOOK FOR THRES. SO THAT
GINI IMPURITY IS MINIMAL

LOOK FOR 2nd THRES. SO THAT
GINI IMPURITY IS MINIMAL

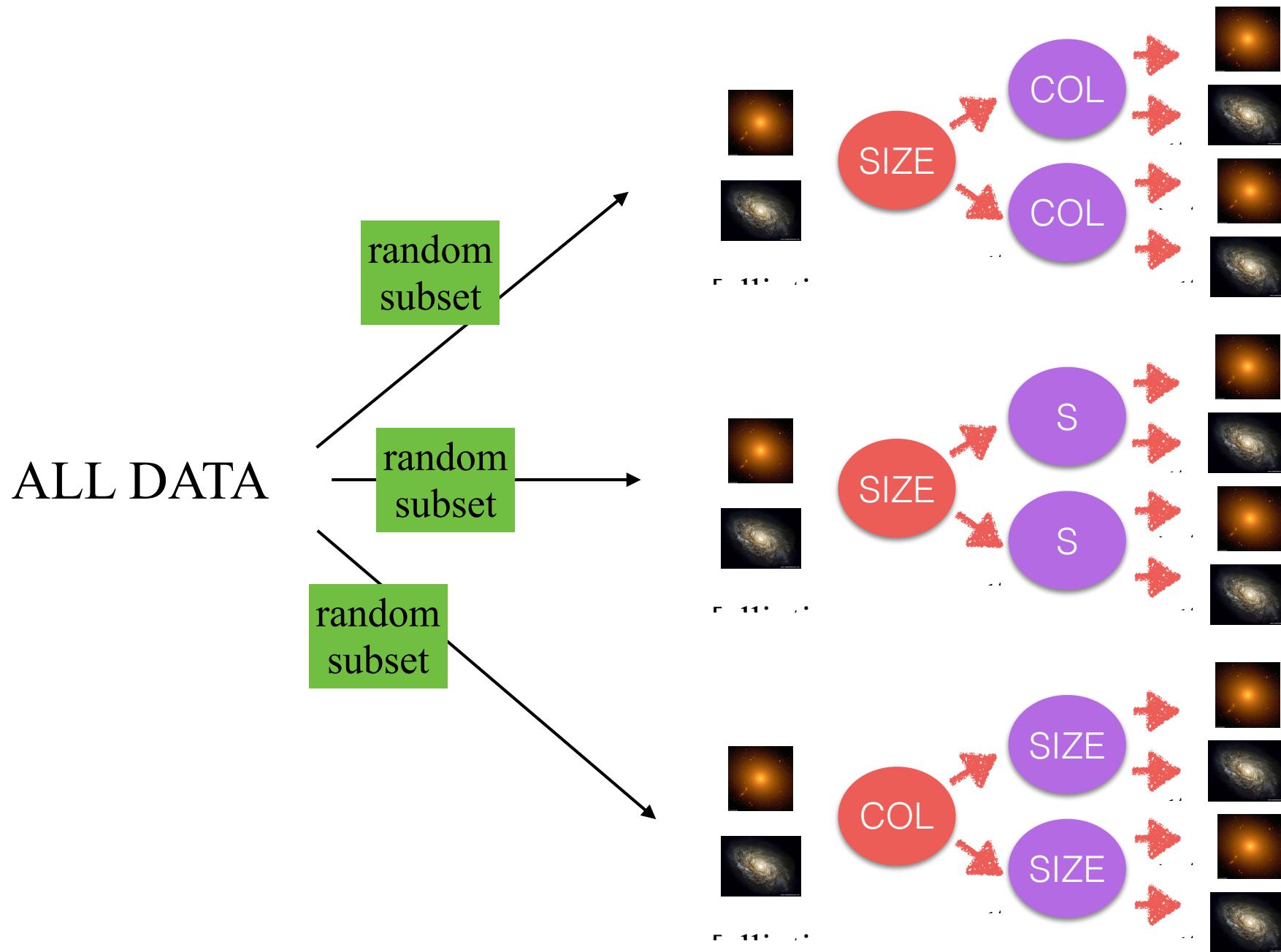
RANDOM FORESTS



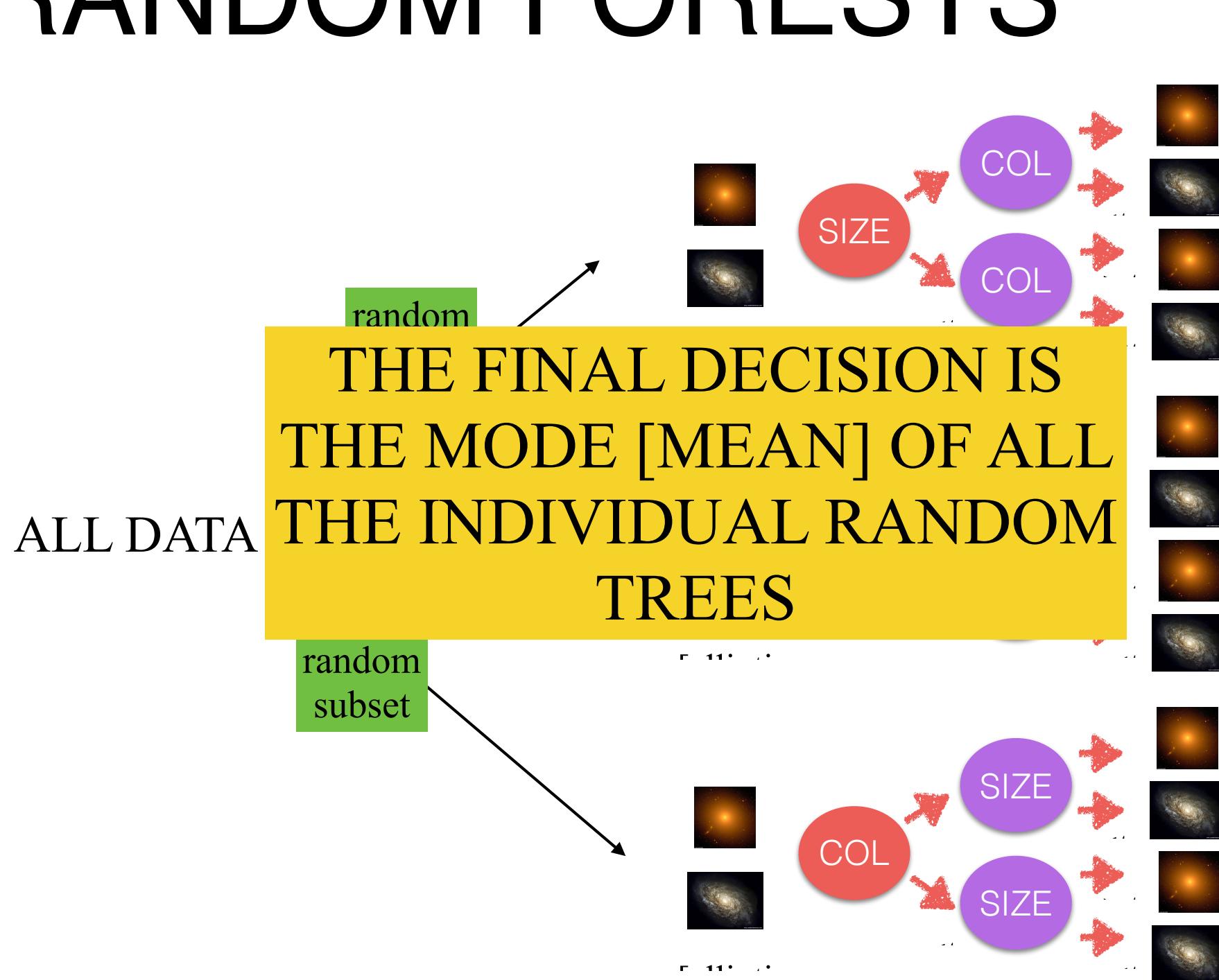
LOOK FOR THRES. SO THAT
GINI IMPURITY IS MINIMAL

LOOK FOR 2nd THRES. SO THAT
GINI IMPURITY IS MINIMAL

RANDOM FORESTS



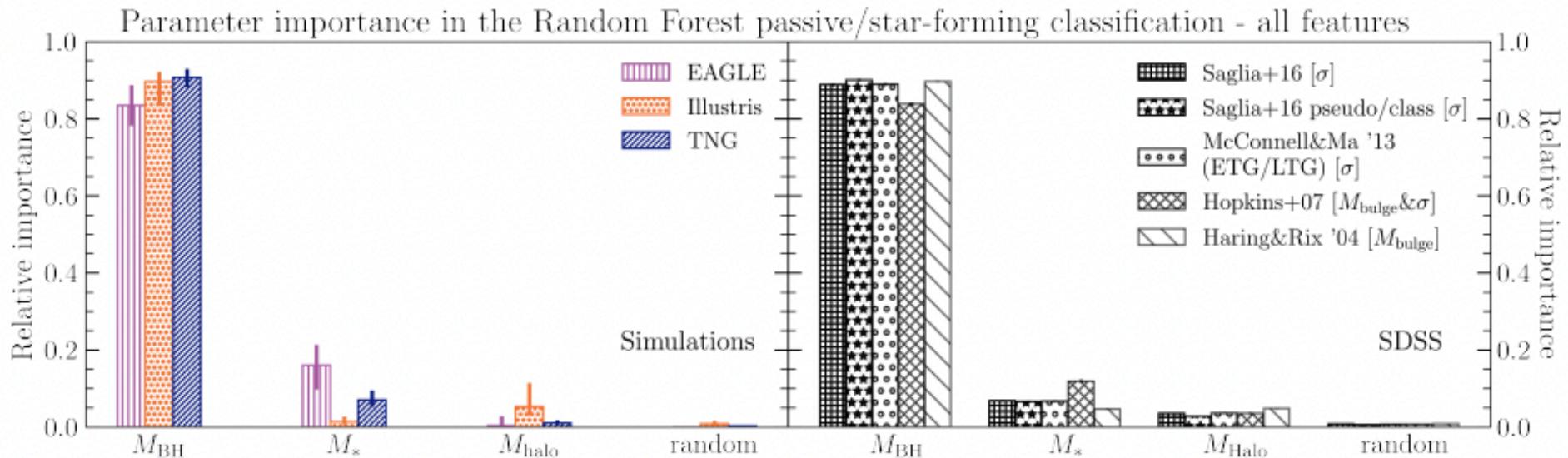
RANDOM FORESTS



Random Forests

ONE KEY ADVANTAGE OF DECISION TREE ALGORITHMS IS THAT
THEY ARE VERY EASY TO INTERPRET

ONE CAN EASILY DETERMINE THE MOST IMPORTANT FEATURES
TO TAKE DECISIONS



RANDOM FORESTS

ONE KEY ADVANTAGE OF DECISION TREE ALGORITHMS IS THAT THEY ARE VERY EASY TO INTERPRET (INTERPRETABILITY)

ONE CAN “EASILY” DETERMINE THE MOST IMPORTANT FEATURES TO TAKE DECISIONS

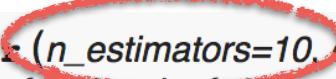
Rank	Property	AUC	Success label ^a
1	ALL	0.9074 ± 0.0106	Outstanding
2	CVD	0.8559 ± 0.0039	Excellent
3	M_{bulge}	0.8335 ± 0.0060	Excellent
4	B/T	0.8267 ± 0.0028	Excellent
5	M_{halo}	0.7983 ± 0.0045	Acceptable
6	M_{*}	0.7819 ± 0.0025	Acceptable
7	M_{disc}	0.7124 ± 0.0016	Acceptable
8	δ_5	0.5894 ± 0.0015	Unacceptable
—	Re	0.5599 ± 0.0013	Unacceptable

IMPORTANCE OF
PARAMETERS TO PREDICT IF
A GALAXY IS QUENCHED

PRACTICAL NOTE: *Python scikit learn*

- All these different methods are standard and available in Python
- Very easy to use
- All info here

`sklearn.ensemble.RandomForestClassifier`

```
class sklearn.ensemble. RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',  
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False,  
n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None)  [source]
```

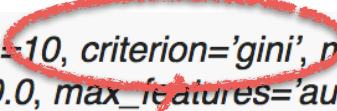


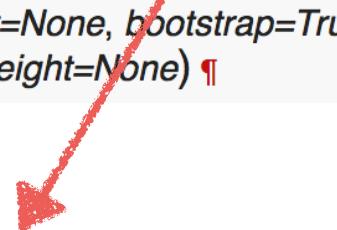
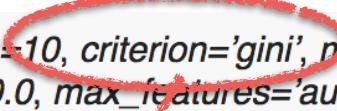
NUMBER OF RANDOM
TREES

EVERY ML ALGORITHM HAS A NUMBER OF HYPER
PARAMETERS WHICH NEED TO BE TUNED

EXCEPT FOR VERY FEW CASES, THERE ARE NO PRE-
DEFINED RULES

`sklearn.ensemble.RandomForestClassifier`

```
class sklearn.ensemble. RandomForestClassifier (n_estimators=10, criterion='gini', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',  
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False,  
n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None)  [source]
```



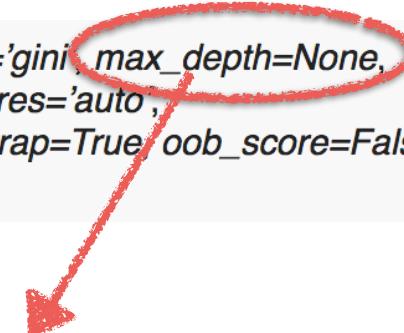
METRIC TO MINIMIZE

EVERY ML ALGORITHM HAS A NUMBER OF HYPER
PARAMETERS WHICH NEED TO BE TUNED

EXCEPT FOR VERY FEW CASES, THERE ARE NO PRE-
DEFINED RULES

`sklearn.ensemble.RandomForestClassifier`

```
class sklearn.ensemble. RandomForestClassifier (n_estimators=10, criterion='gini', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',  
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False,  
n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None) ¶ [source]
```



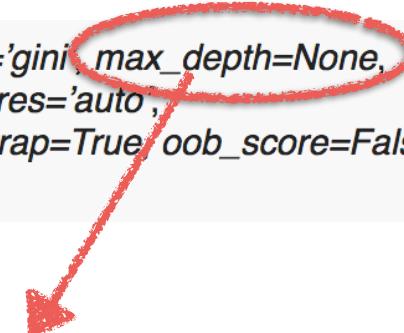
MAXIMUM NUMBER OF SPLITS

EVERY ML ALGORITHM HAS A NUMBER OF HYPER PARAMETERS WHICH NEED TO BE TUNED

EXCEPT FOR VERY FEW CASES, THERE ARE NO PRE-DEFINED RULES

sklearn.ensemble.RandomForestClassifier

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',  
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False,  
n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None) ¶ [source]
```

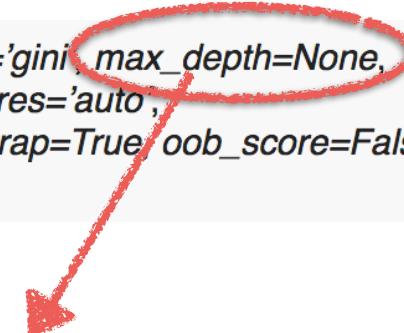


MAXIMUM NUMBER OF SPLITS

```
>>> from sklearn.ensemble import RandomForestClassifier  
>>> from sklearn.datasets import make_classification  
>>>  
>>> X, y = make_classification(n_samples=1000, n_features=4,  
...                                n_informative=2, n_redundant=0,  
...                                random_state=0, shuffle=False)  
>>> clf = RandomForestClassifier(max_depth=2, random_state=0)  
>>> clf.fit(X, y)  
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',  
                      max_depth=2, max_features='auto', max_leaf_nodes=None,  
                      min_impurity_decrease=0.0, min_impurity_split=None,  
                      min_samples_leaf=1, min_samples_split=2,  
                      min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,  
                      oob_score=False, random_state=0, verbose=0, warm_start=False)  
>>> print(clf.feature_importances_)  
[ 0.17287856  0.80608704  0.01884792  0.00218648]  
>>> print(clf.predict([[0, 0, 0, 0]]))  
[1]
```

sklearn.ensemble.RandomForestClassifier

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',  
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False,  
n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None) ¶ [source]
```



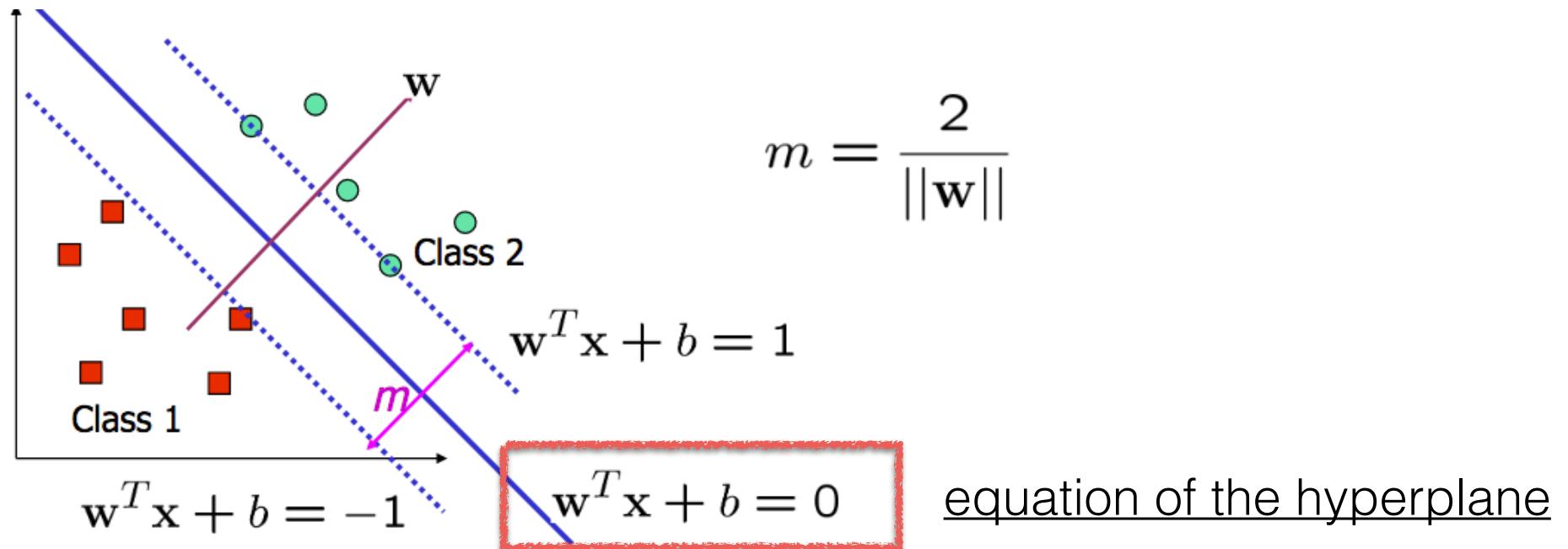
MAXIMUM NUMBER OF SPLITS

```
>>> from sklearn.ensemble import RandomForestClassifier  
>>> from sklearn.datasets import make_classification  
>>>  
>>> X, y = make_classification(n_samples=1000, n_features=4,  
...                                n_informative=2, n_redundant=0,  
...                                random_state=0, shuffle=False)  
>>> clf = RandomForestClassifier(max_depth=2, random_state=0)  
>>> clf.fit(X, y)  
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',  
                      max_depth=2, max_features='auto', max_leaf_nodes=None,  
                      min_impurity_decrease=0.0, min_impurity_split=None,
```

DIFFERENT CLASSIFIERS ARE OBJECTS WITH METHODS TO
FIT, PREDICT ETC..

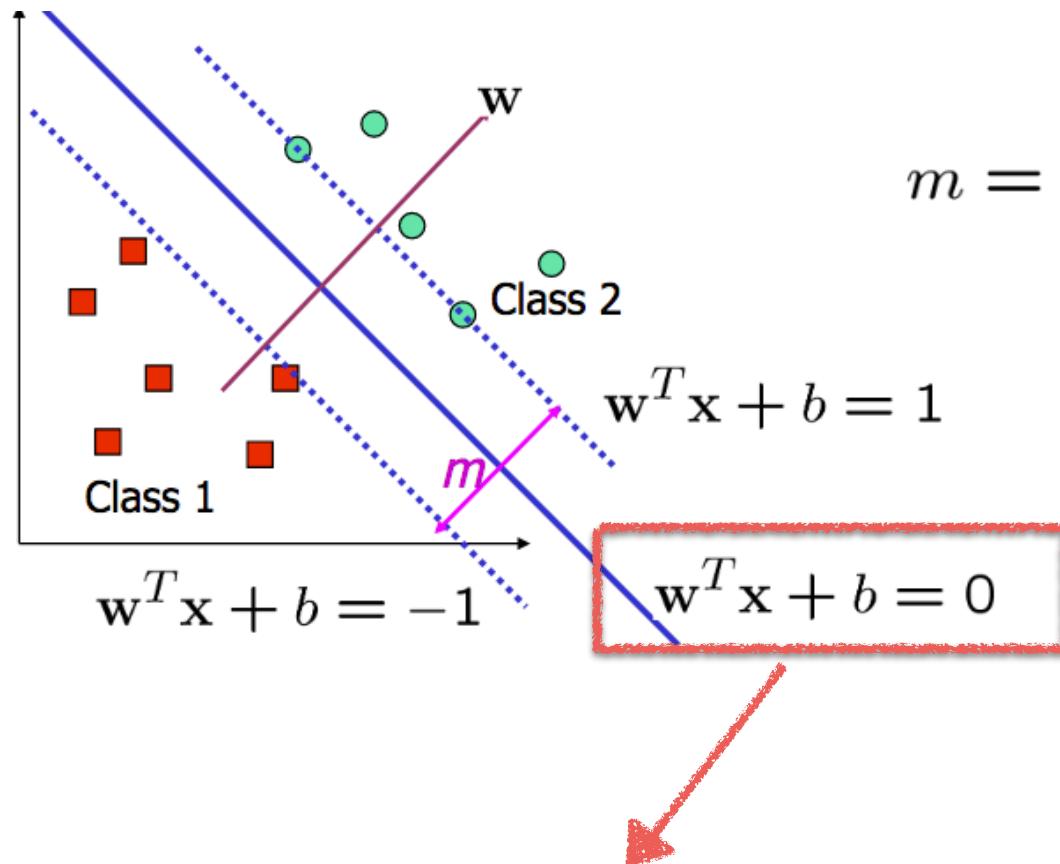
```
>>> print(clf.predict([[0, 0, 0, 0]]))  
[1]
```

Support Vector Machines



[Credit: M. Law]

Support Vector Machines

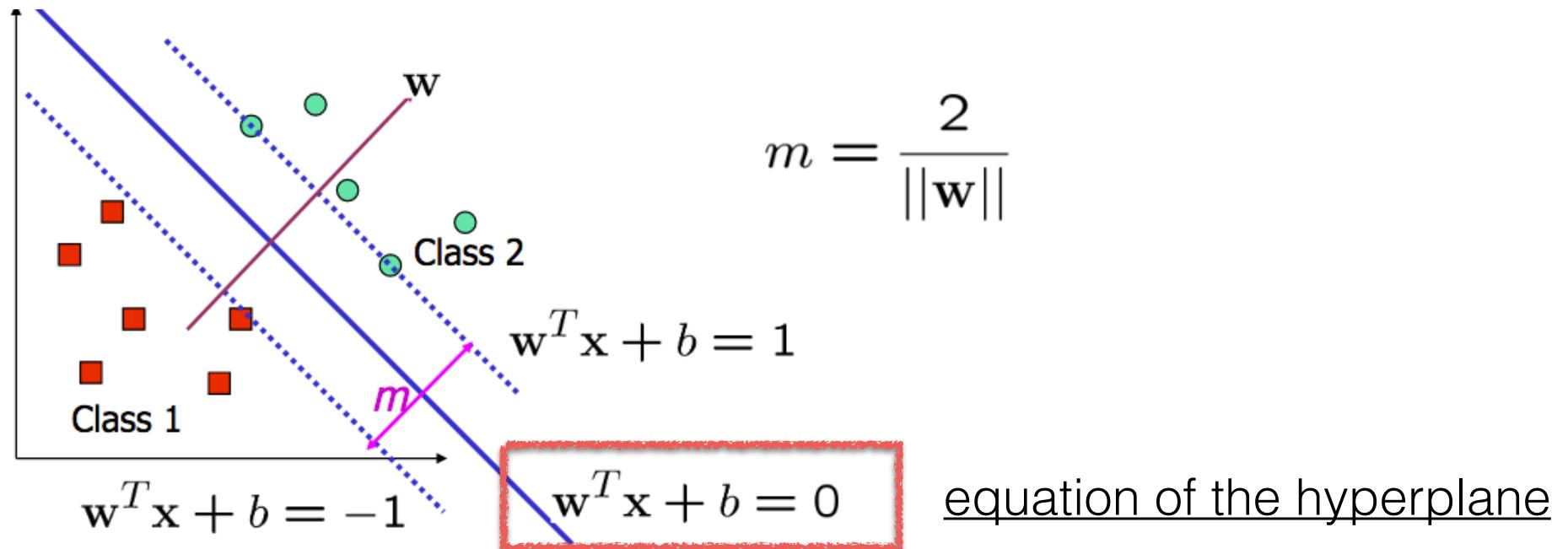


$$m = \frac{2}{\|w\|}$$

equation of the hyperplane

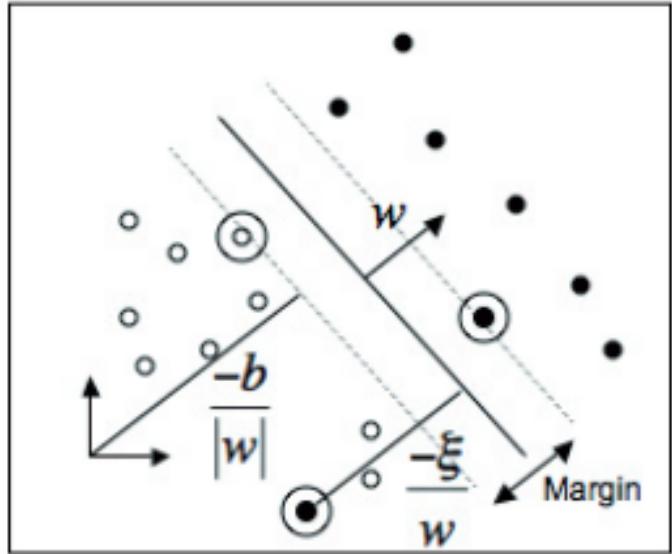
THE GAME HERE IS TO FIND THE W THAT
MAXIMIZES THE MARGIN [MINIMIZE $\|w\|/2$]

Support Vector Machines



IF THE PROBLEM IS LINEARLY SEPARABLE, THERE IS
AN IDEAL SOLUTION [see figure]

Support Vector Machines



IN MOST CASES THE DATA
ARE NON LINEARLY
SEPARABLE

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 - \epsilon_i$$

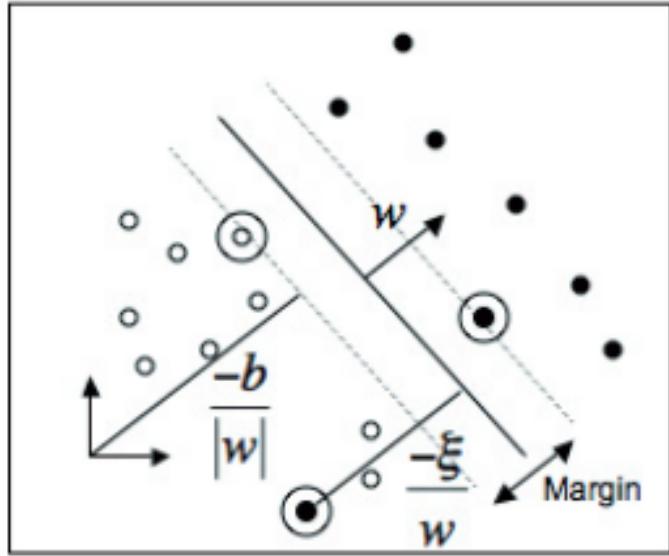
$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 + \epsilon_i$$

Huertas-Company+08

THE CONDITION IS RELAXED BY ADDING A POSITIVE
STACK VARIABLE

Huertas-Company+08

Support Vector Machines



IN MOST CASES THE DATA
ARE NON LINEARLY
SEPARABLE

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 - \epsilon_i$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 + \epsilon_i$$

Huertas-Company+08

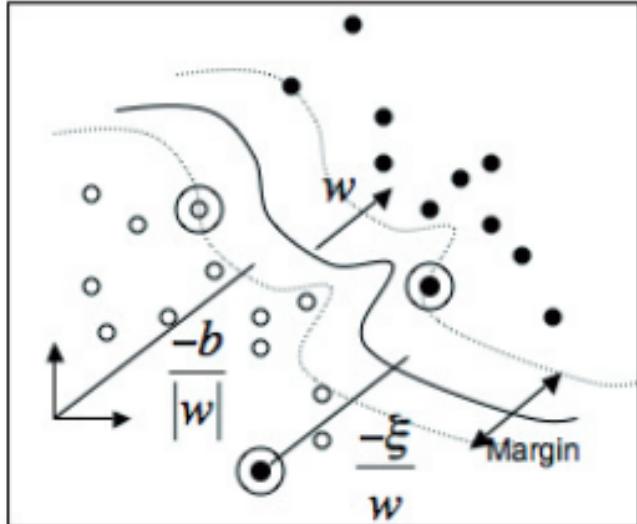
THE CONDITION IS RELAXED BY ADDING A POSITIVE
STACK VARIABLE

THE FUNCTION TO MINIMIZE:

$$\frac{\|\mathbf{w}\|^2}{2} + T \left[\sum \epsilon_i \right]$$

THE LARGER T, THE
MORE ERRORS
ARE PENALIZED

Support Vector Machines



THE FINAL TRICK IS TO FIND
NON-LINEAR BOUNDARIES

WE MAP THE PLANE INTO
A NEW EUCLIDIAN SPACE
WHERE THE POINTS ARE
SEPARABLE

$$\Phi : \mathbb{R} \rightarrow H$$

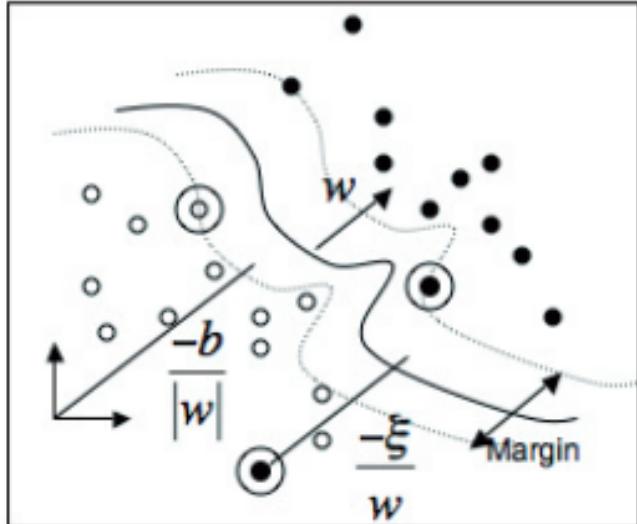
SINCE WE ONLY NEED
TO COMPUTE INNER
PRODUCTS IN THIS
SPACE

$$x_i \cdot x_j$$

THE TRAINING
ALGORITHM WOULD
ONLY DEPEND ON THE
DATA THROUGH THE
PRODUCTS IN H

$$\Phi(x_i) \cdot \Phi(x_j)$$

Support Vector Machines



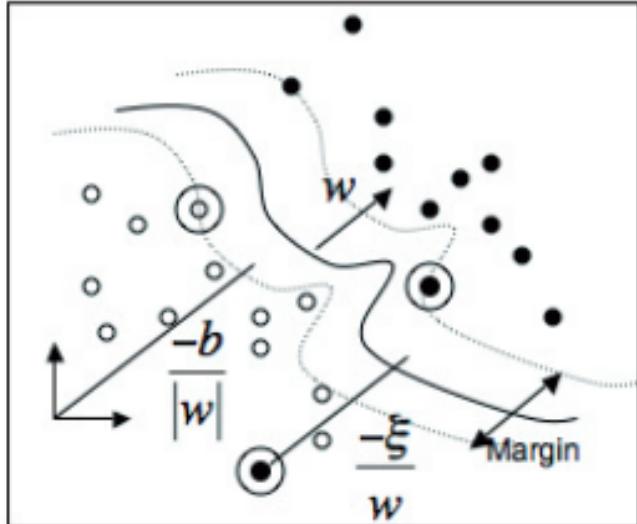
THE FINAL TRICK IS TO FIND
NON-LINEAR BOUNDARIES

IF THERE IS A **KERNEL FUNCTION** SO THAT:

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$$

THEN THERE IS NO NEED TO EXPLICITLY
KNOW Φ

Support Vector Machines



THE FINAL TRICK IS TO FIND
NON-LINEAR BOUNDARIES

IF THERE IS A **KERNEL FUNCTION** SO THAT:

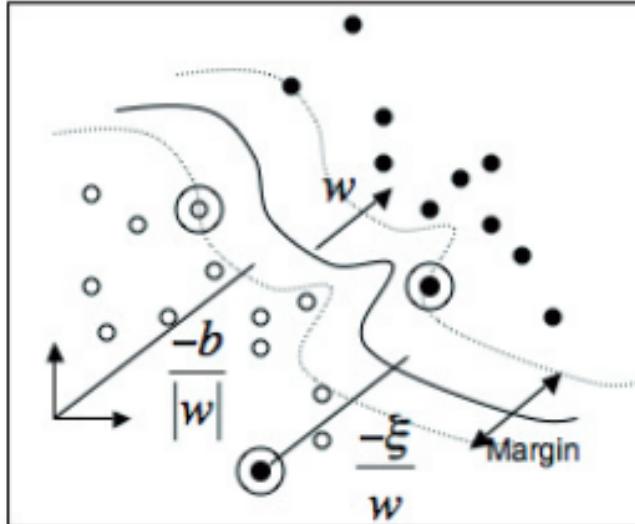
$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$$

THEN THERE IS NO NEED TO EXPLICITLY
KNOW Φ

SVM IS CALLED **A KERNEL METHOD**

Huertas-Company+08

Support Vector Machines



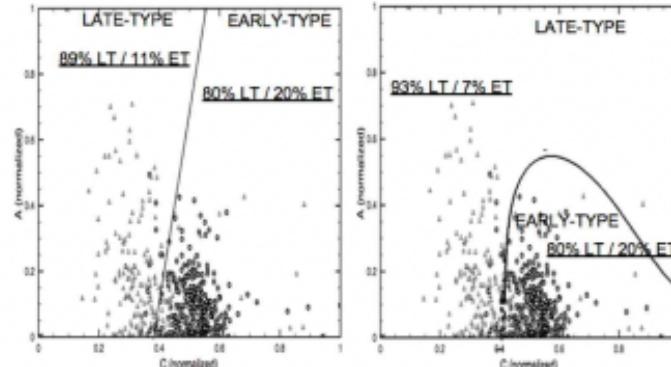
THE FINAL TRICK IS TO FIND
NON-LINEAR BOUNDARIES

IF THERE IS A **KERNEL FUNCTION** SO THAT:

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$$

Huertas-Company-

SVM IS CALLED



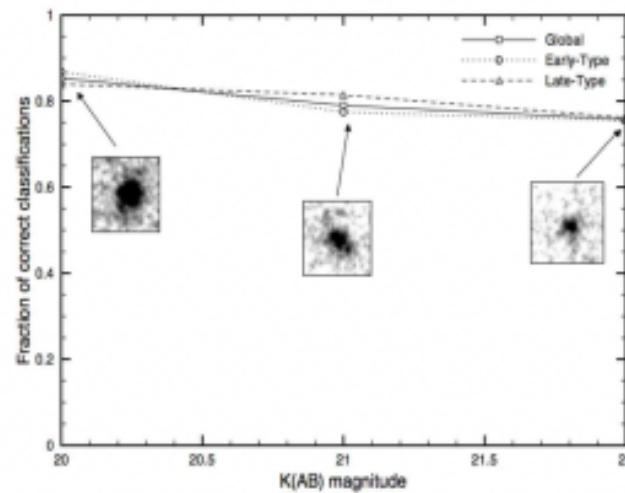
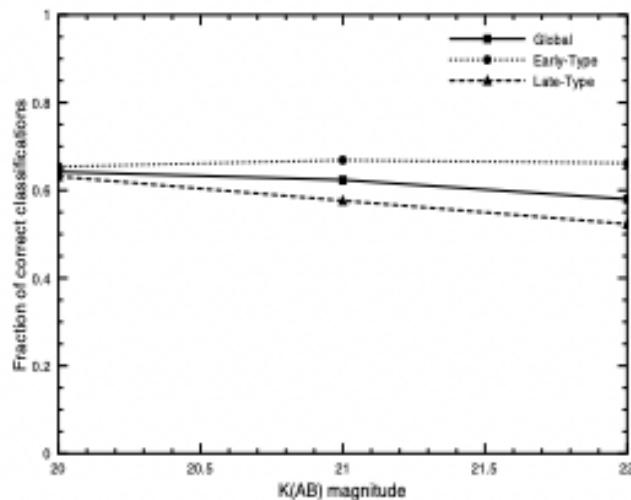
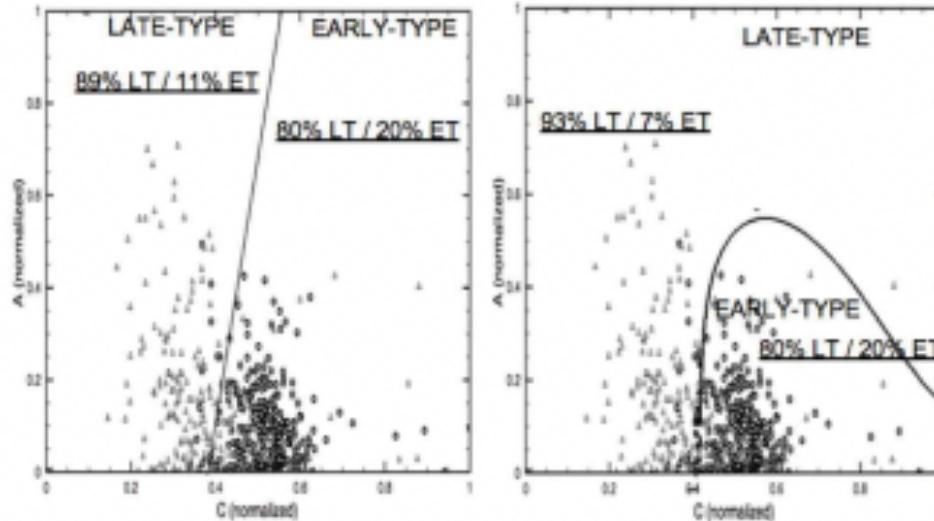
IS NO NEED TO EXPLICITLY
KNOW Φ

EXAMPLE OF KERNEL:

$$K(x, y) = e^{-g||x-y||^2}$$

[GAUSSIAN RBF]

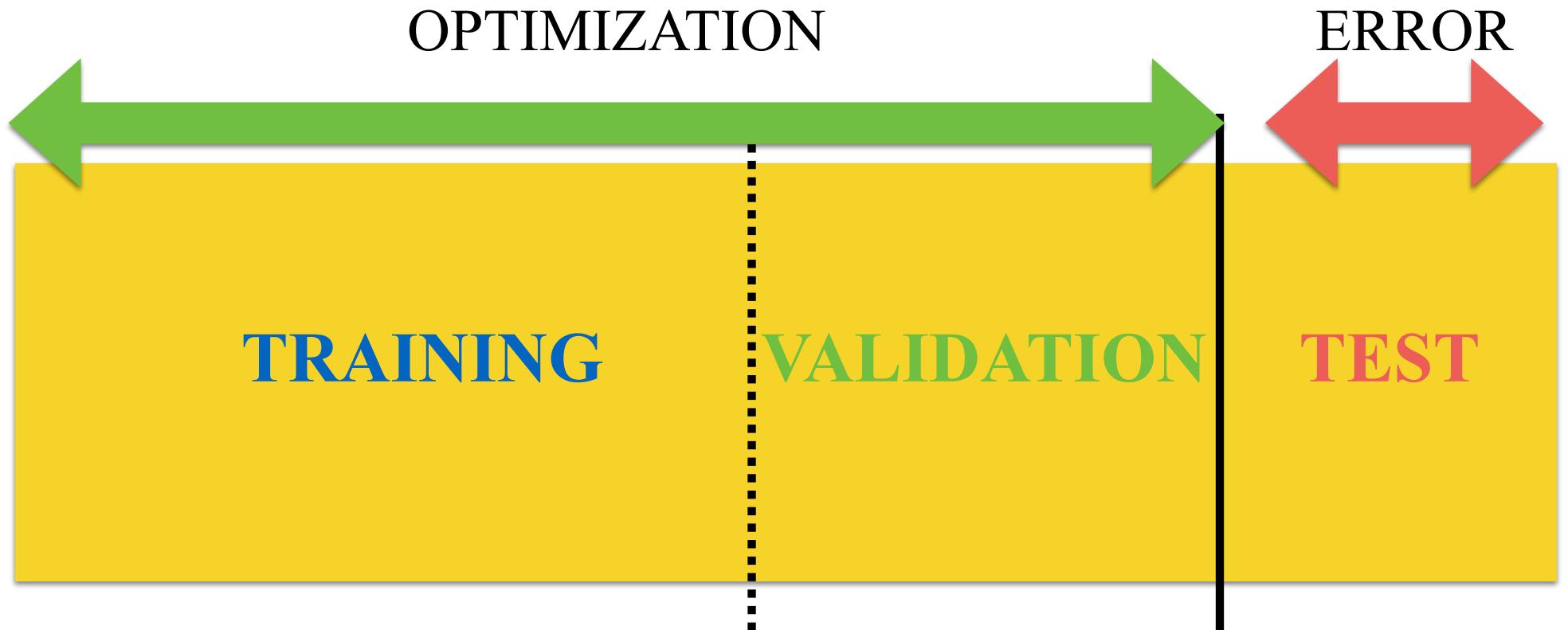
Support Vector Machines



HOW DO I KNOW THAT MY MACHINE LEARNING ALGORITHM IS WORKING?

- The way results are evaluated depends strongly on the type of problem
 - Binary Classification: ACC, ROC, P-C
 - Multi-Class: Confusion Matrix
 - Regression: RMSE, Bias, Scatter etc..

REMEMBER



training set: use to train the classifier

validation set: use to monitor performance in real time - check
for overfitting

test set: use to train the classifier

Evaluation of results [binary class.]

THE MOST STRAIGHTFORWARD WAY IS TO EVALUATE THE
TOTAL ACCURACY

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

true positives true negatives



MEASURES HOW MANY OBJECTS ARE CORRECTLY
CLASSIFIED

Evaluation of results [binary class.]

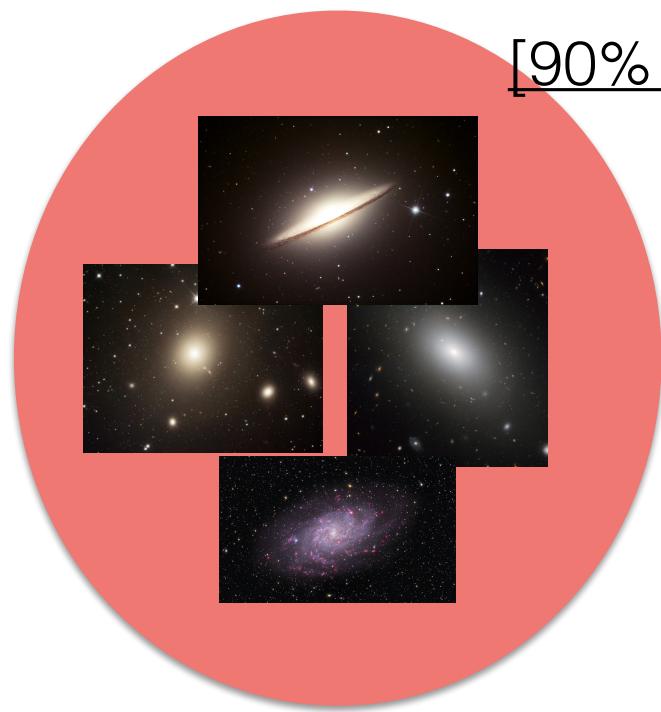
THE MOST STRAIGHTFORWARD WAY IS TO EVALUATE THE
TOTAL ACCURACY

$$ACC = \frac{\text{true positives} + \text{true negatives}}{TP + TN + FP + FN}$$

NOT VERY
INFORMATIVE IF
UNBALANCED
CLASSES

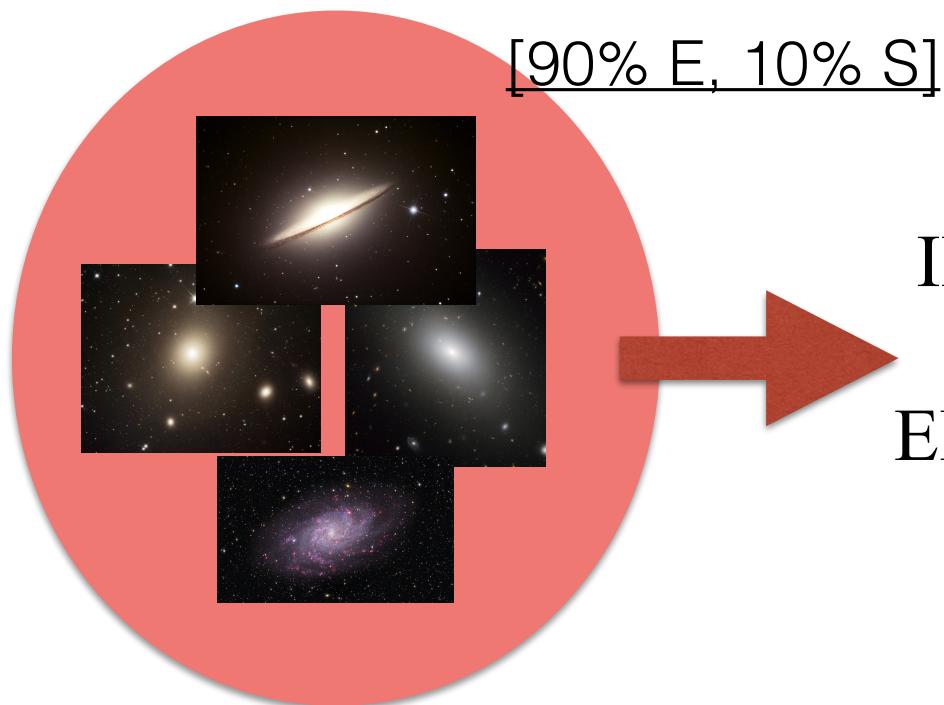
MEASURES HOW MANY OBJECTS ARE CORRECTLY
CLASSIFIED

IMAGINE AN EXTREME CASE WITH VERY UNBALANCED
DATA...

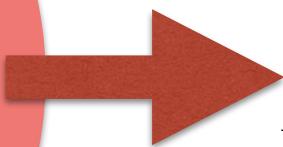


[90% E, 10% S]

IMAGINE AN EXTREME CASE WITH VERY UNBALANCED
DATA...



[90% E, 10% S]



IF I SAY THAT ALL GALAXIES
IN THE UNIVERSE ARE
ELLIPTICALS I WILL BE RIGHT
90% OF THE TIMES

Evaluation of results [binary class.]

THE ROC CURVE (Receiver Operating Characteristic)

IT IS BASED ON TWO VERY SIMPLE PARAMETERS

1.
$$TPR = \frac{TP}{TP + FN}$$
 [Also called Sensitivity, Completeness]

“Fraction of positive examples classified correctly”

Evaluation of results [binary class.]

THE ROC CURVE (Receiver Operating Characteristic)

IT IS BASED ON TWO VERY SIMPLE PARAMETERS

1.

$$TPR = \frac{TP}{TP + FN}$$

TRUE POSITIVE RATE
[Also called Sensitivity, Completeness]

“Fraction of positive examples classified correctly”

2.

$$FPR = \frac{FP}{FP + TN}$$

FALSE POSITIVE RATE
[Also called Specificity, Contamination]

“Fraction of negative examples classified as positive”

Evaluation of results [binary class.]

- YOU WANT THIS TO BE AS BIG AS POSSIBLE
- Over Operating Characteristic)

IT IS BASED ON TWO VERY SIMPLE PARAMETERS

1.

$$TPR = \frac{TP}{TP + FN}$$

TRUE POSITIVE RATE
[Completeness]

YOU WANT THIS TO BE AS SMALL AS POSSIBLE

“Fraction of positive examples classified correctly”

2.

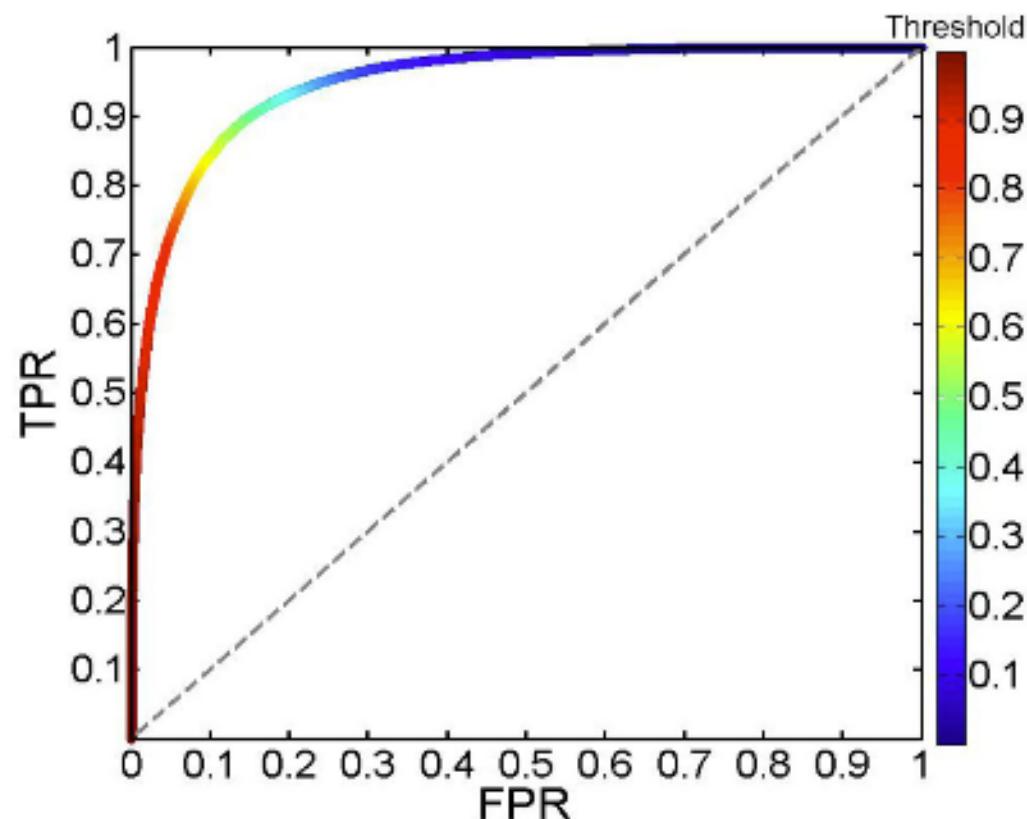
$$FPR = \frac{FP}{FP + TN}$$

FALSE POSITIVE RATE
[Also called Specificity, Contamination]

“Fraction of negative examples classified as positive”

IF YOUR CLASSIFIER OUTPUTS A SORT OF PROBABILITY,
TPR AND FPR CAN BE PLOTTED ONE AGAINST THE OTHER

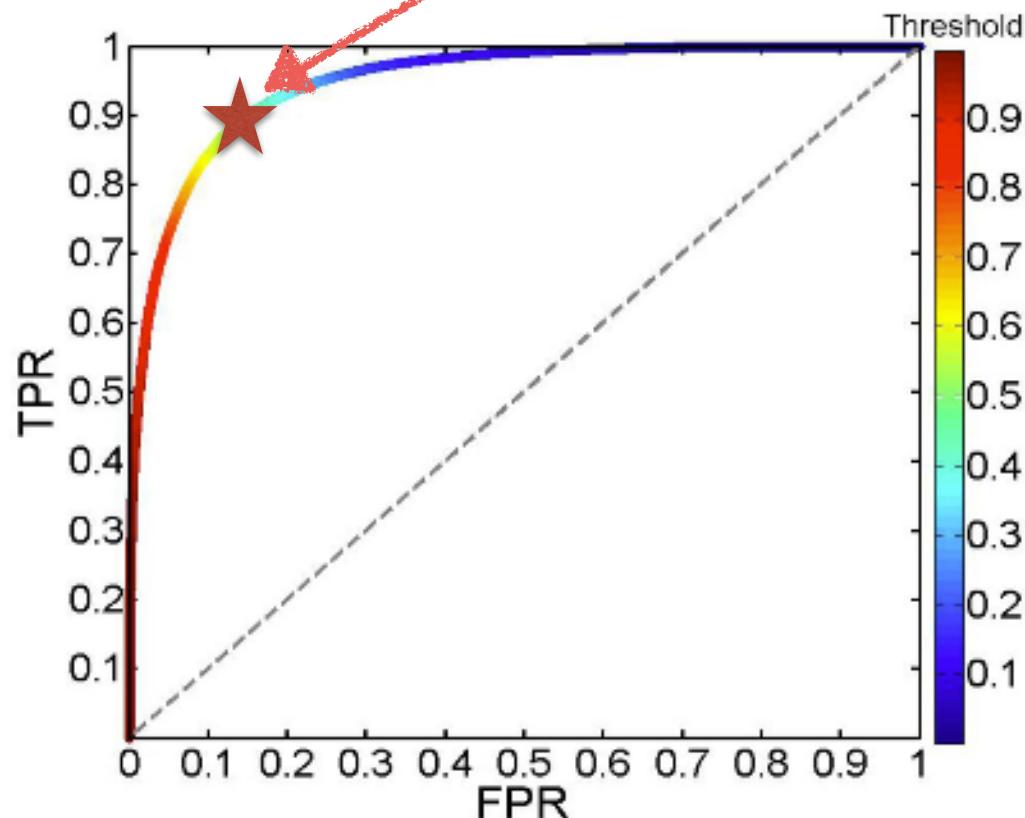
ROC CURVE



IF YOUR CLASSIFIER OUTPUTS A SORT OF PROBABILITY,
TPR AND FPR CAN BE PLOTTED ONE AGAINST THE OTHER

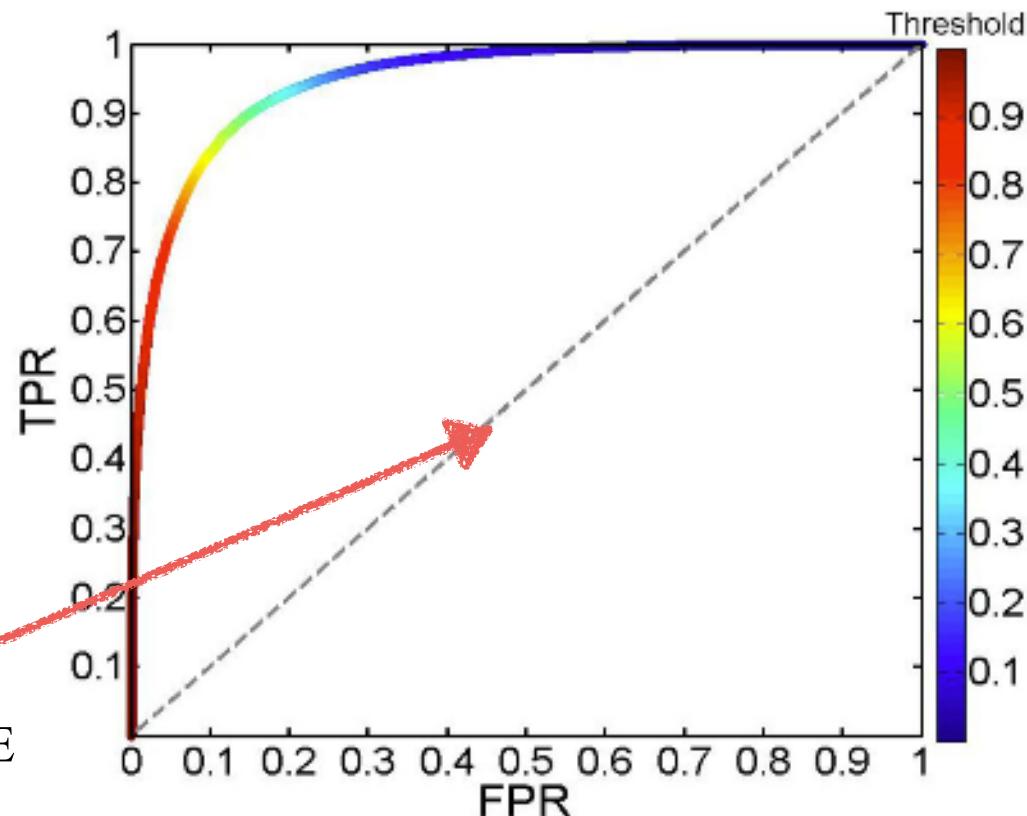
EACH POINT HERE SHOWS THE VALUES OF TPR AND
FPR FOR A GIVEN THRESHOLD

ROC CURVE



IF YOUR CLASSIFIER OUTPUTS A SORT OF PROBABILITY,
TPR AND FPR CAN BE PLOTTED ONE AGAINST THE OTHER

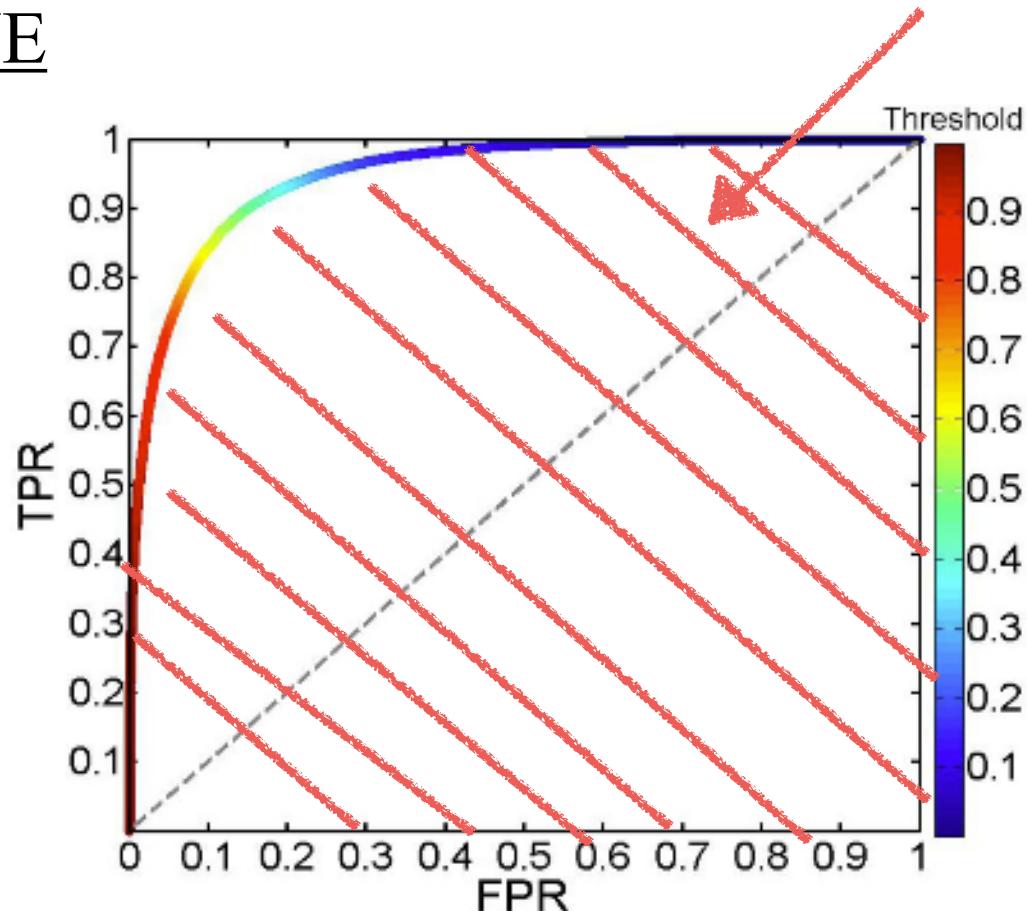
ROC CURVE



THE ONE-TO-ONE
LINE IS A
RANDOM
CLASSIFICATION

IF YOUR CLASSIFIER OUTPUTS A SORT OF PROBABILITY,
TPR AND FPR CAN BE PLOTTED ONE AGAINST THE OTHER

ROC CURVE



THE AREA UNDER THE
CURVE AUC ALSO
MEASURES THE
GLOBAL ACCURACY

Evaluation of results

THE P-R CURVE (Precision - Recall)

$$Recall = \frac{TP}{TP + FN} = TPR \quad [\text{completeness}]$$

$$Precision = \frac{TP}{TP + FP} \quad [\text{purity}]$$

Evaluation of results

THE P-R CURVE (Precision - Recall)

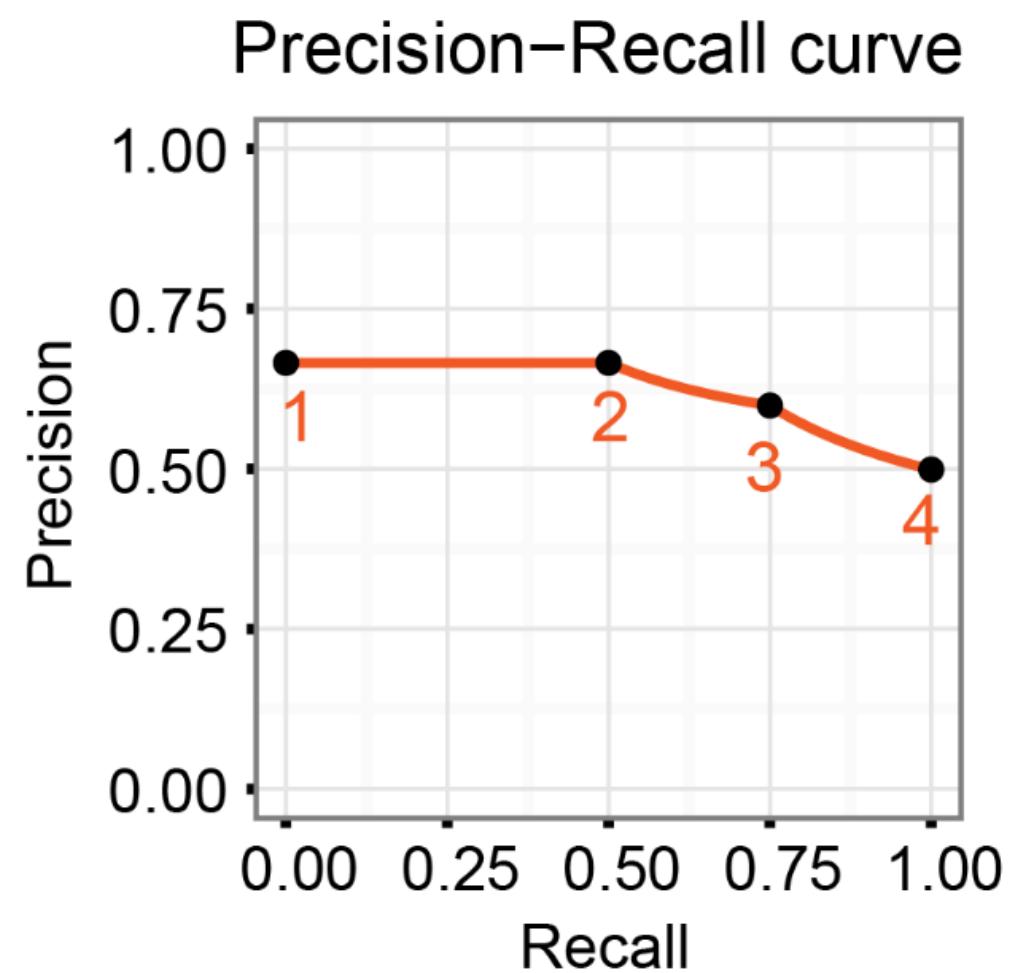
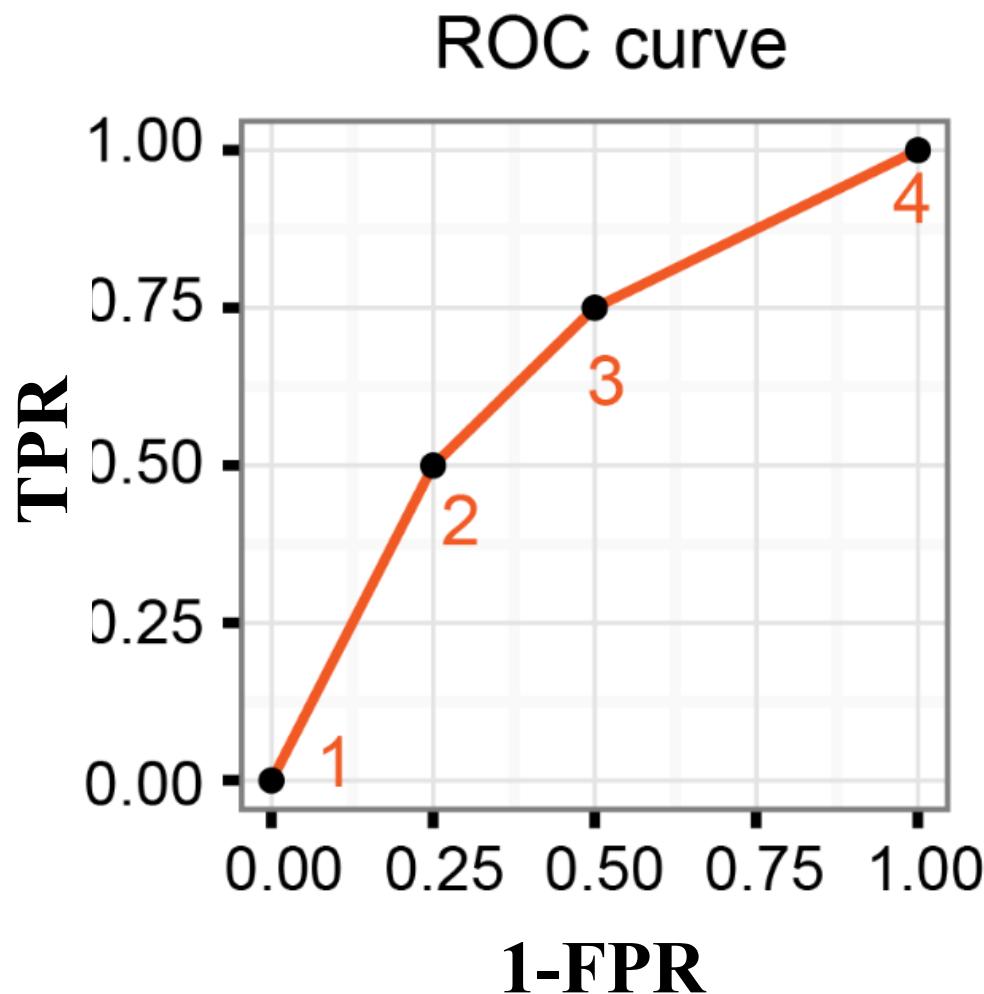
$$Recall = \frac{TP}{TP + FN} = TPR \quad [\text{completeness}]$$

$$Precision = \frac{TP}{TP + FP} \quad [\text{purity}]$$



FOR BALANCED DATA: $Precision \sim 1 - FPR$

Evaluation of results



SUMMARY OF DIFFERENT ACCURACY TRACERS

		True condition			
Total population	Condition positive	Condition negative	Prevalence = $\frac{\sum \text{Condition positive}}{\sum \text{Total population}}$	Accuracy (ACC) = $\frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total population}}$	
Predicted condition	Predicted condition positive	True positive, Power	False positive, Type I error	Positive predictive value (PPV), Precision = $\frac{\sum \text{True positive}}{\sum \text{Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\sum \text{False positive}}{\sum \text{Predicted condition positive}}$
	Predicted condition negative	False negative, Type II error	True negative	False omission rate (FOR) = $\frac{\sum \text{False negative}}{\sum \text{Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\sum \text{True negative}}{\sum \text{Predicted condition negative}}$
	True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\sum \text{False positive}}{\sum \text{Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR-}}$	$F_1 \text{ score} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$
	False negative rate (FNR), Miss rate = $\frac{\sum \text{False negative}}{\sum \text{Condition positive}}$	True negative rate (TNR), Specificity (SPC) = $\frac{\sum \text{True negative}}{\sum \text{Condition negative}}$	Negative likelihood ratio (LR-) = $\frac{\text{FNR}}{\text{TNR}}$		

SOURCE

SUMMARY OF DIFFERENT ACCURACY TRACERS

		True condition			
Total population	Condition positive	Condition negative	Prevalence = $\frac{\sum \text{Condition positive}}{\sum \text{Total population}}$	Accuracy (ACC) = $\frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total population}}$	
Predicted condition	Predicted condition positive	True positive, Power	False positive, Type I error	Positive predictive value (PPV), Precision = $\frac{\sum \text{True positive}}{\sum \text{Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\sum \text{False positive}}{\sum \text{Predicted condition positive}}$
	Predicted condition negative	False negative, Type II error	True negative	False omission rate (FOR) = $\frac{\sum \text{False negative}}{\sum \text{Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\sum \text{True negative}}{\sum \text{Predicted condition negative}}$
	True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\sum \text{False positive}}{\sum \text{Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR-}}$	$F_1 \text{ score} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$
	False negative rate (FNR), Miss rate $= \frac{\sum \text{False negative}}{\sum \text{Condition positive}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\sum \text{True negative}}{\sum \text{Condition negative}}$	Negative likelihood ratio (LR-) = $\frac{\text{FNR}}{\text{TNR}}$		

THE F1 SCORE
COMBINES BOTH INFORMATIONS IN ONE VALUE

[SOURCE](#)

ALL THESE ARE INCLUDED IN SKLEARN

AND ARE VERY EASY TO USE. NO NEED OF CODING THEM AGAIN!

```
sklearn.metrics. precision_recall_curve (y_true, probas_pred, pos_label=None, sample_weight=None) ¶  
[source]
```

Compute precision-recall pairs for different probability thresholds

Note: this implementation is restricted to the binary classification task.

The precision is the ratio $\text{tp} / (\text{tp} + \text{fp})$ where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

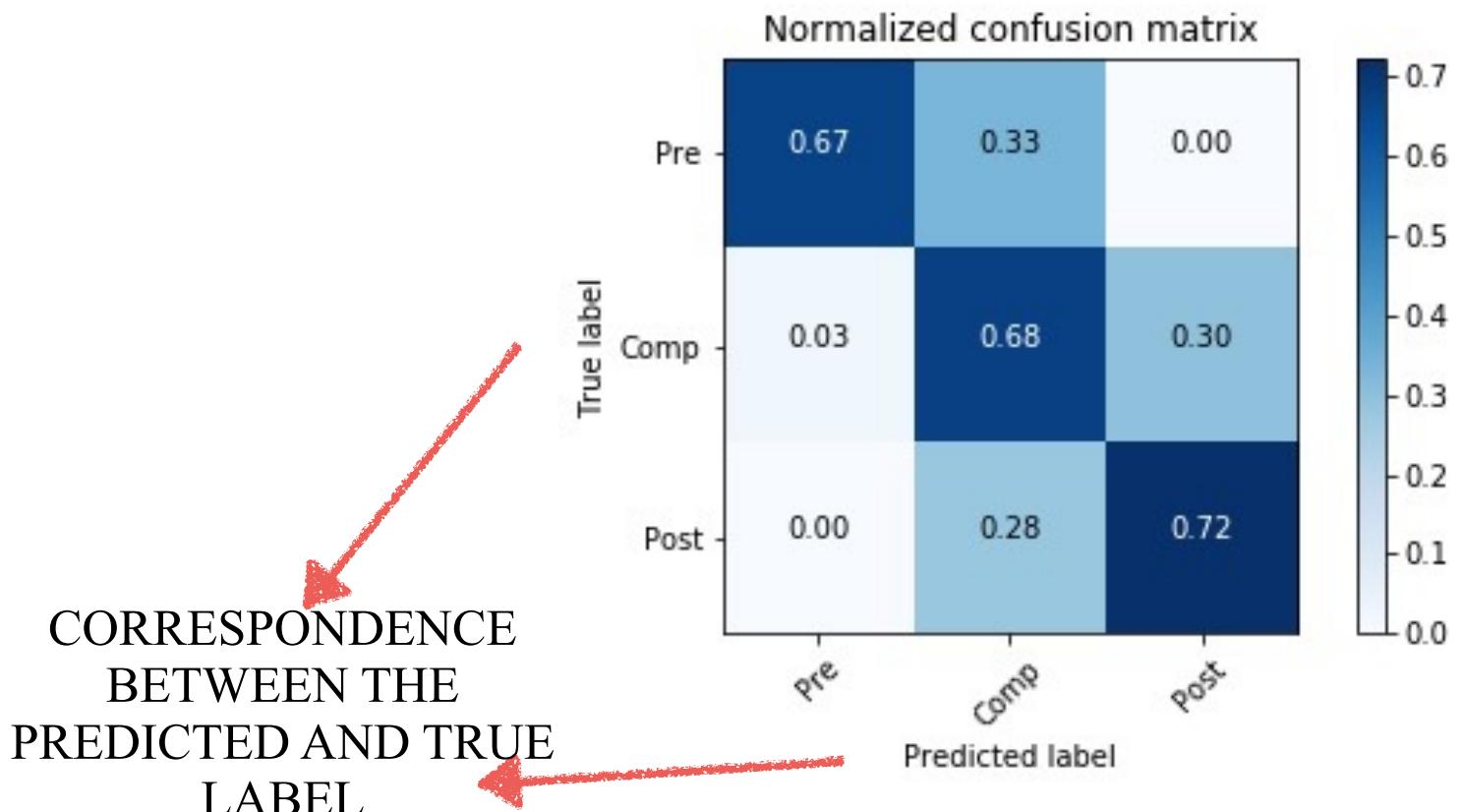
The recall is the ratio $\text{tp} / (\text{tp} + \text{fn})$ where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

The last precision and recall values are 1. and 0. respectively and do not have a corresponding threshold. This ensures that the graph starts on the x axis.

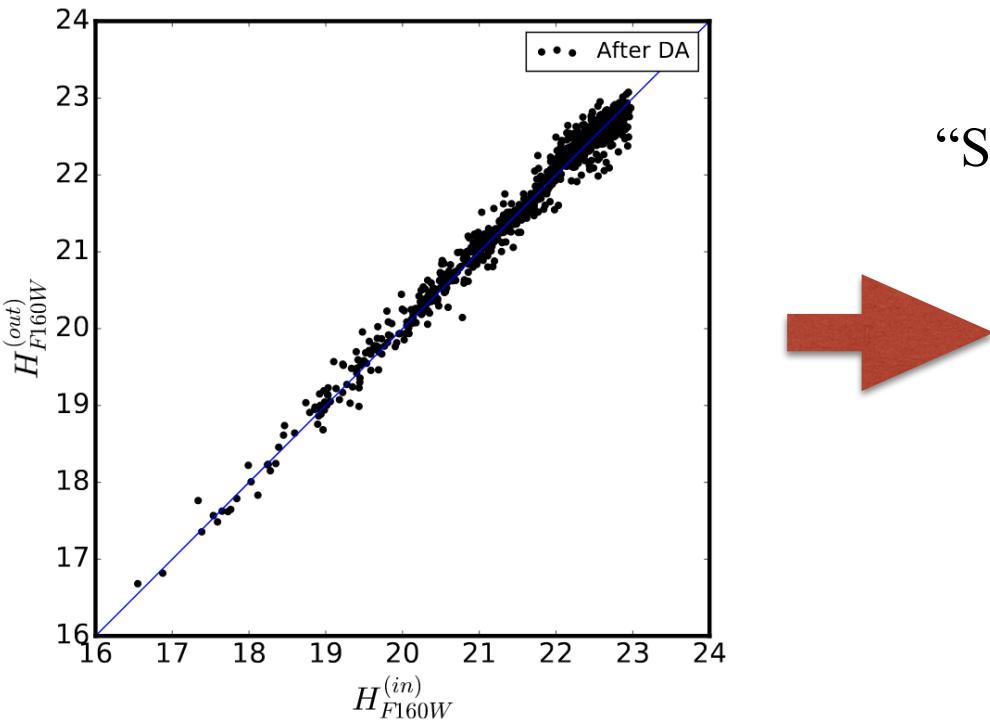
Evaluation of results

[multi-class]

CONFUSION MATRIX

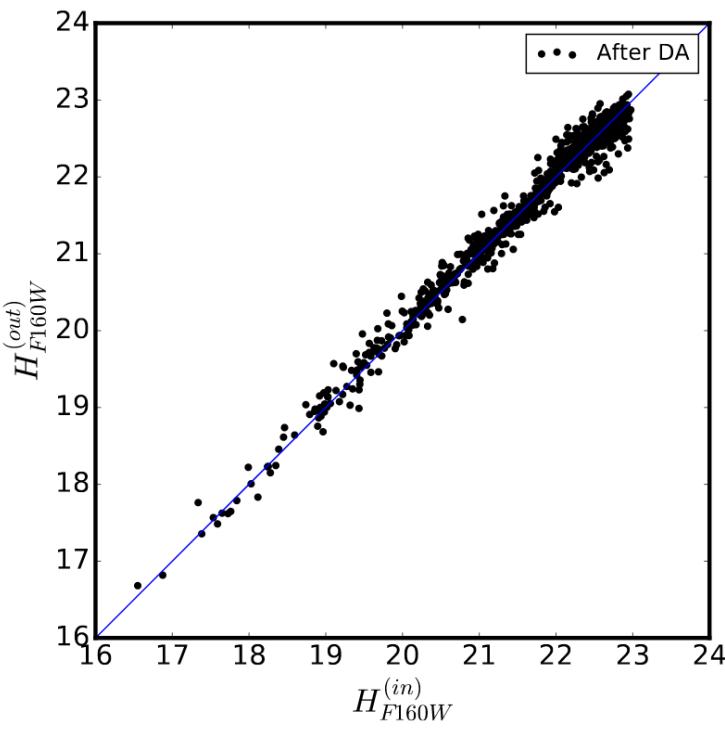


Evaluation of results [regression]



FOR REGRESSIONS, SIMPLY USE
“STANDARD ACCURACY MEASUREMENTS”

Evaluation of results [regression]



FOR REGRESSIONS, SIMPLY USE
“STANDARD ACCURACY MEASUREMENTS”

BIAS, SCATTER ... YOU KNOW!

