PART I: AN INTRODUCTION TO "CLASSICAL" MACHINE LEARNING

WHAT DOES MACHINE LEARNING DO?

the machine is told what to look for

SUPERVISED

the machine is NOT told what to look for

UN-SUPERVISED

TWO BIG TYPES OF MACHINE LEARNING ALGORITHMS

SUPERVISED LEARNING

CAT

CAT

the machine is told what to look for

DOG

HUMAN LABELLING

DOG

SUPERVISED LEARNING

CAT

CAT

the machine is told what to look for

DOG

HUMAN LABELLING

TRAINING SET OF LABELED EXAMPLES

DOG

CAT

CAT

DOG

DOG

SUPERVISED LEARNING

ML

CAT

SUPERVISED LEARNING

CAT

CAT

DOG

DOG

ML

DOG

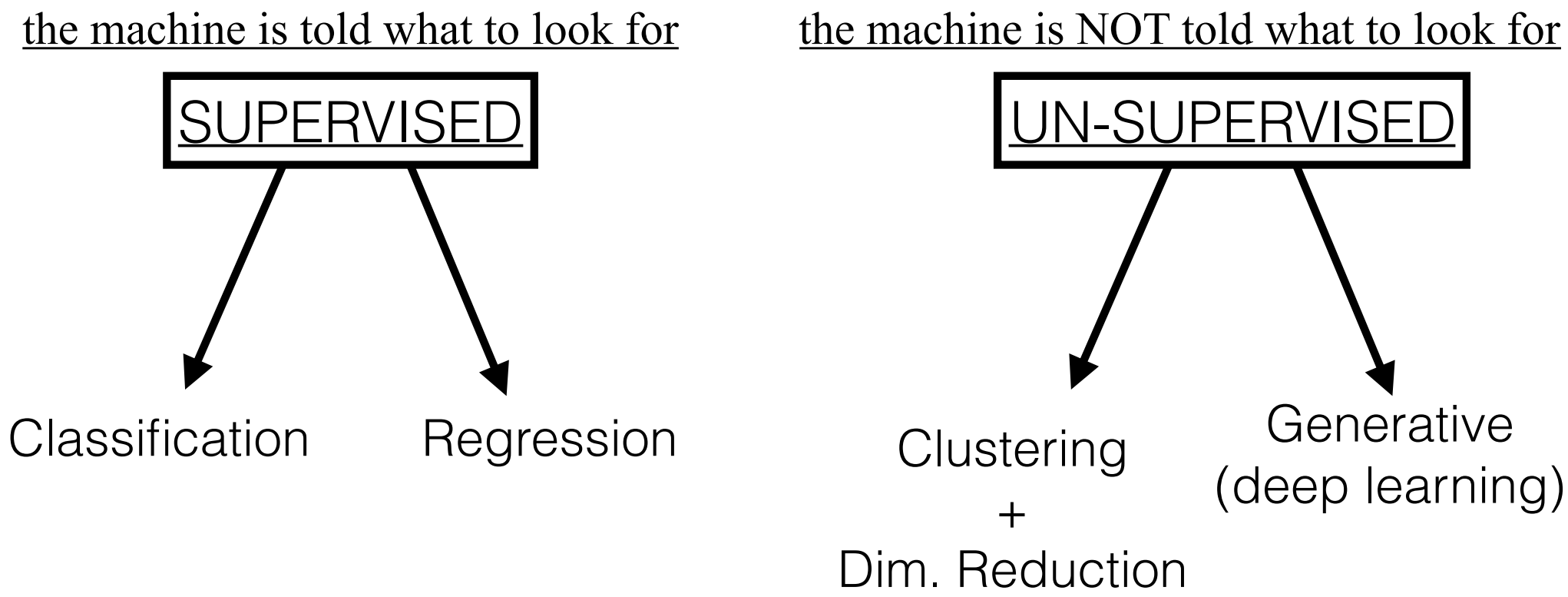
UNSUPERVISED LEARNING

UNSUPERVISED LEARNING

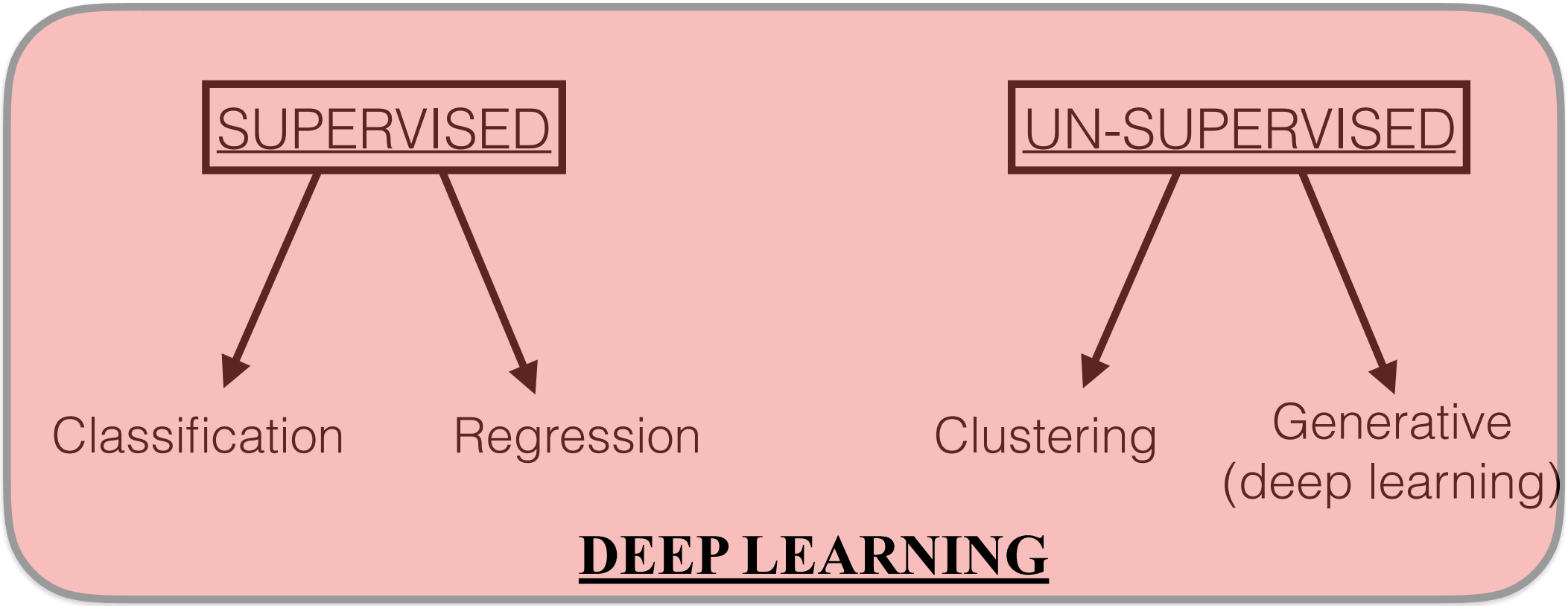
UNSUPERVISED LEARNING

THE DEFINITION OF CLASSES IS SOMETIMES NOT OBVIOUS

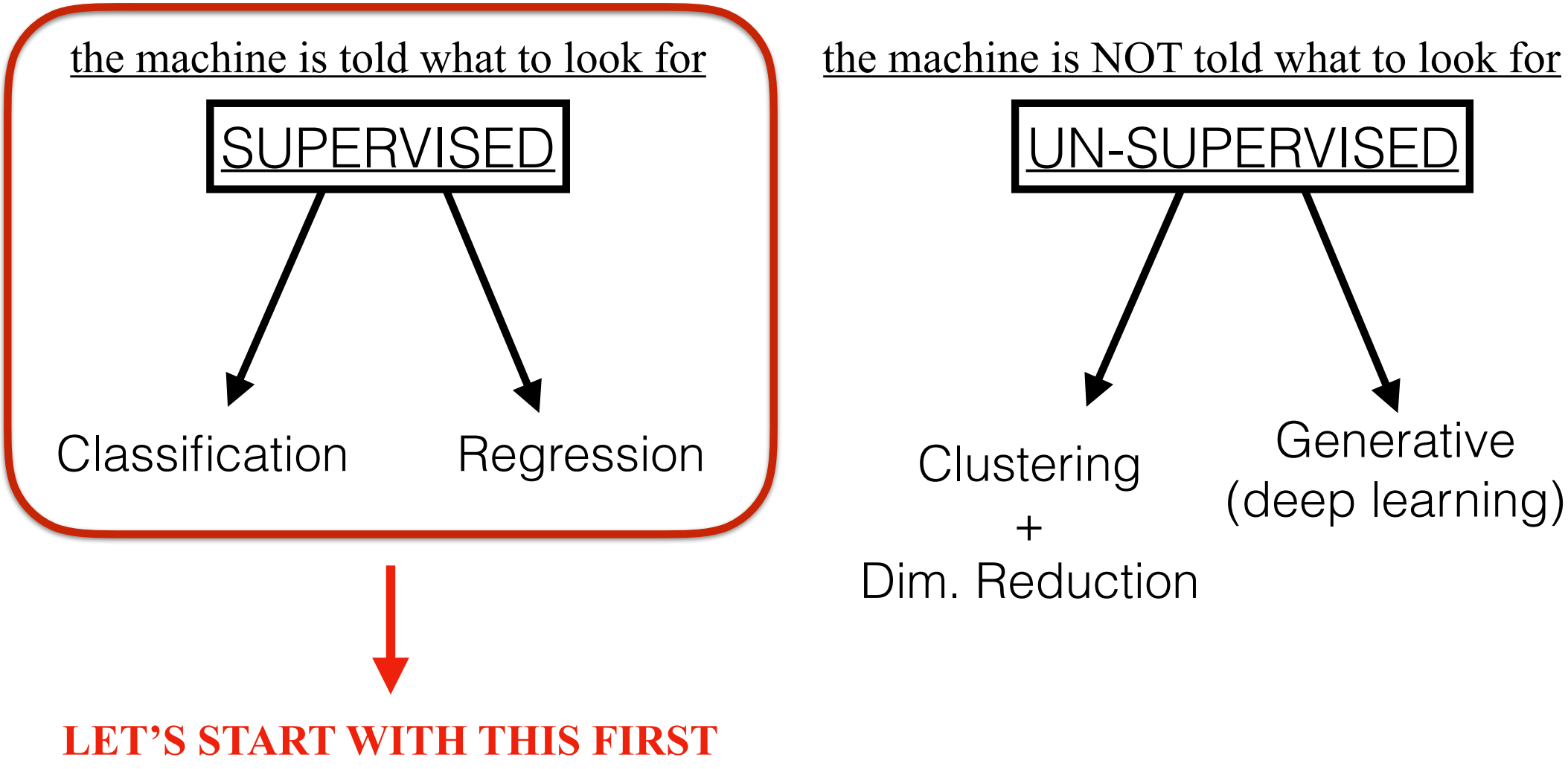
WHAT DOES MACHINE LEARNING DO?



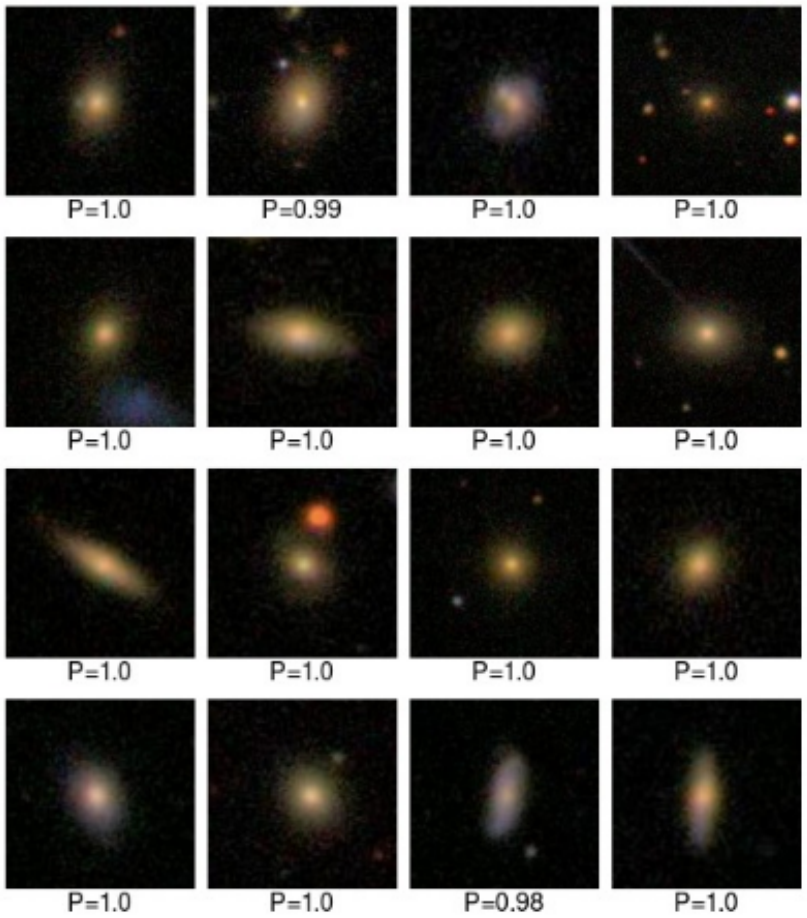
WHAT DOES MACHINE LEARNING DO?



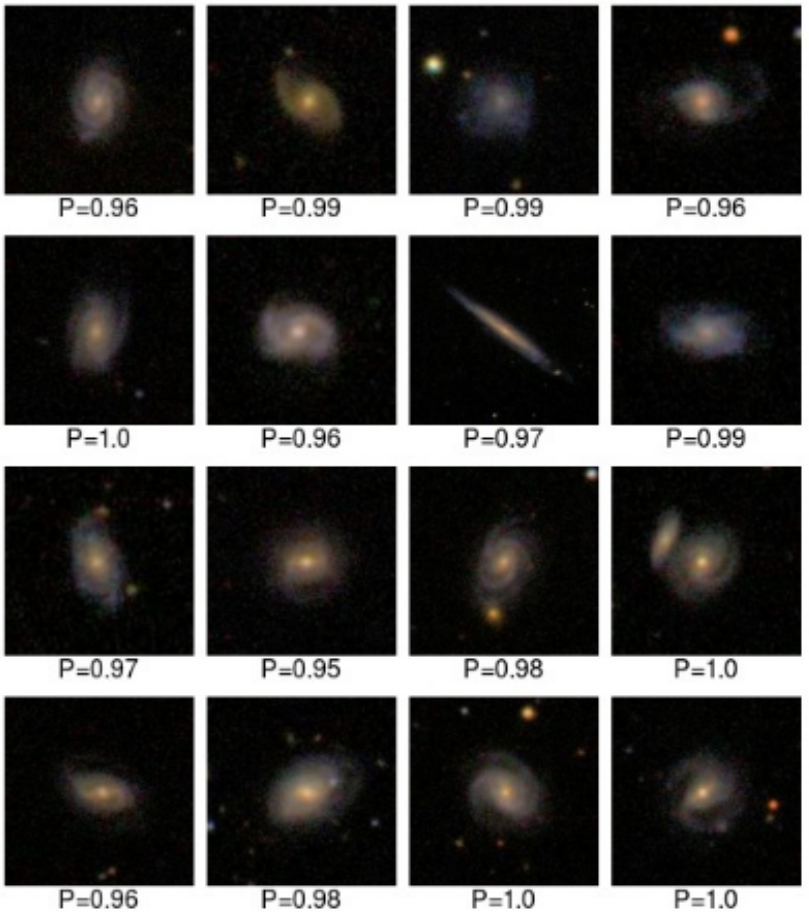
WHAT DOES MACHINE LEARNING DO?



SUPERVISED CLASSIFICATION



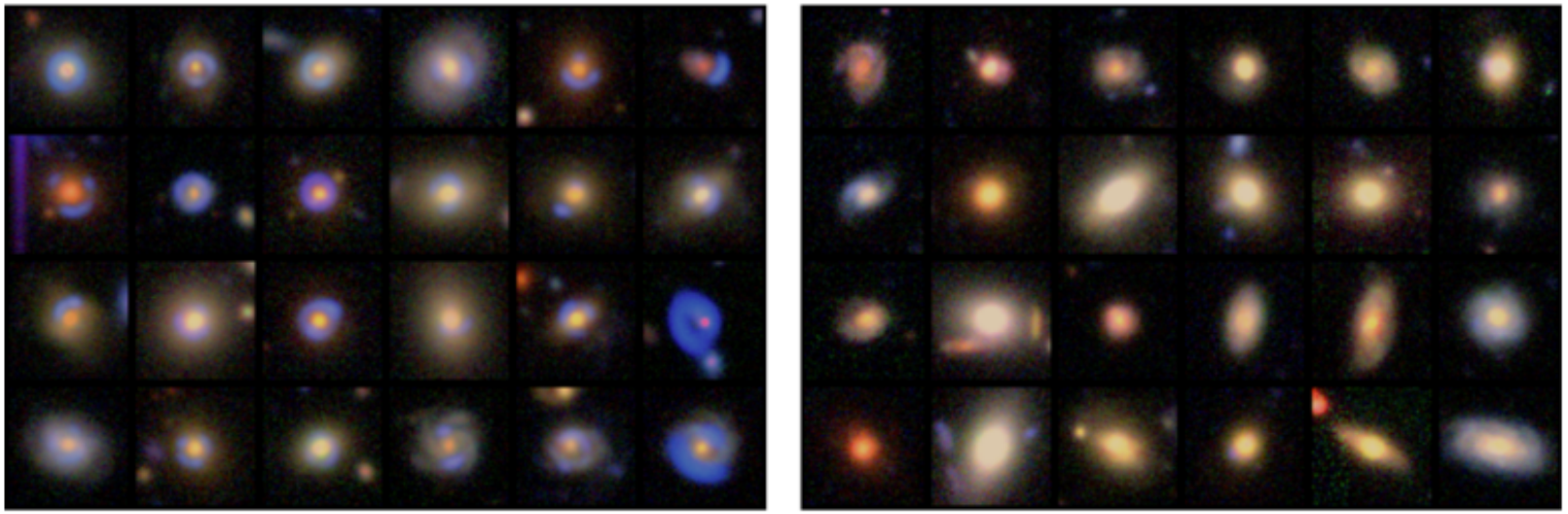
EARLY-TYPE



LATE-TYPE

Dominguez-Sanchez+18

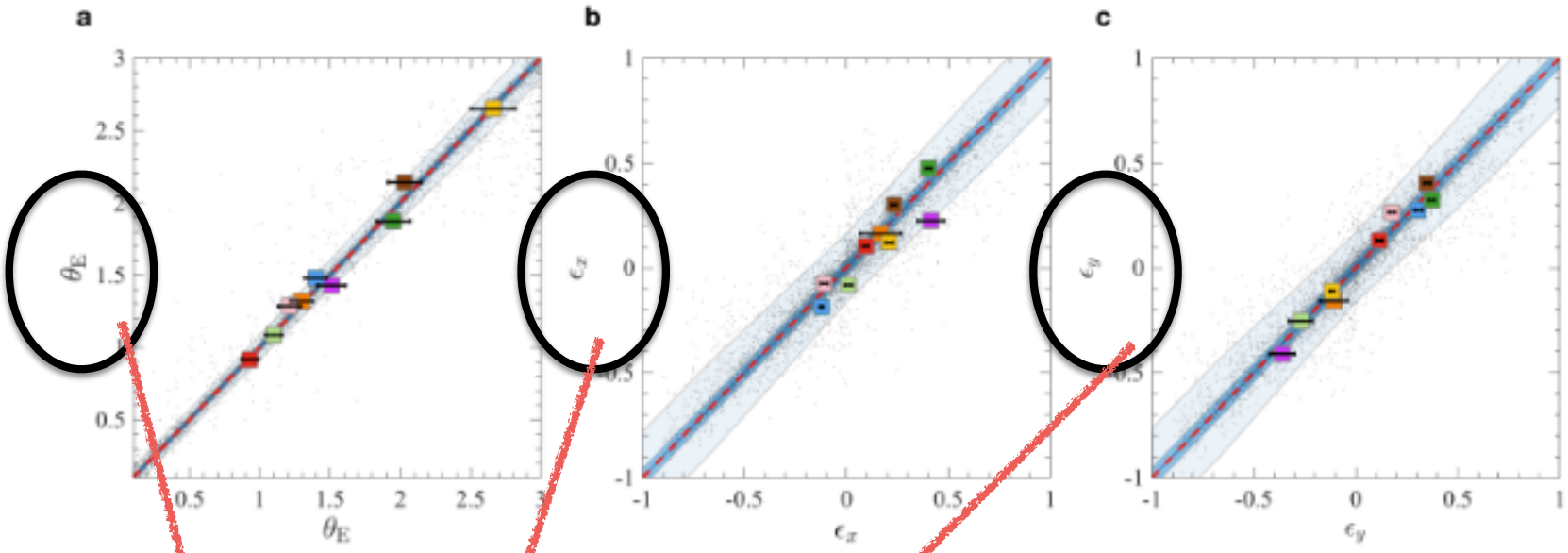
ANOTHER EXAMPLE OF SUPERVISED CLASSIFICATION



FINDING LENSED GALAXIES

Jacobs+17

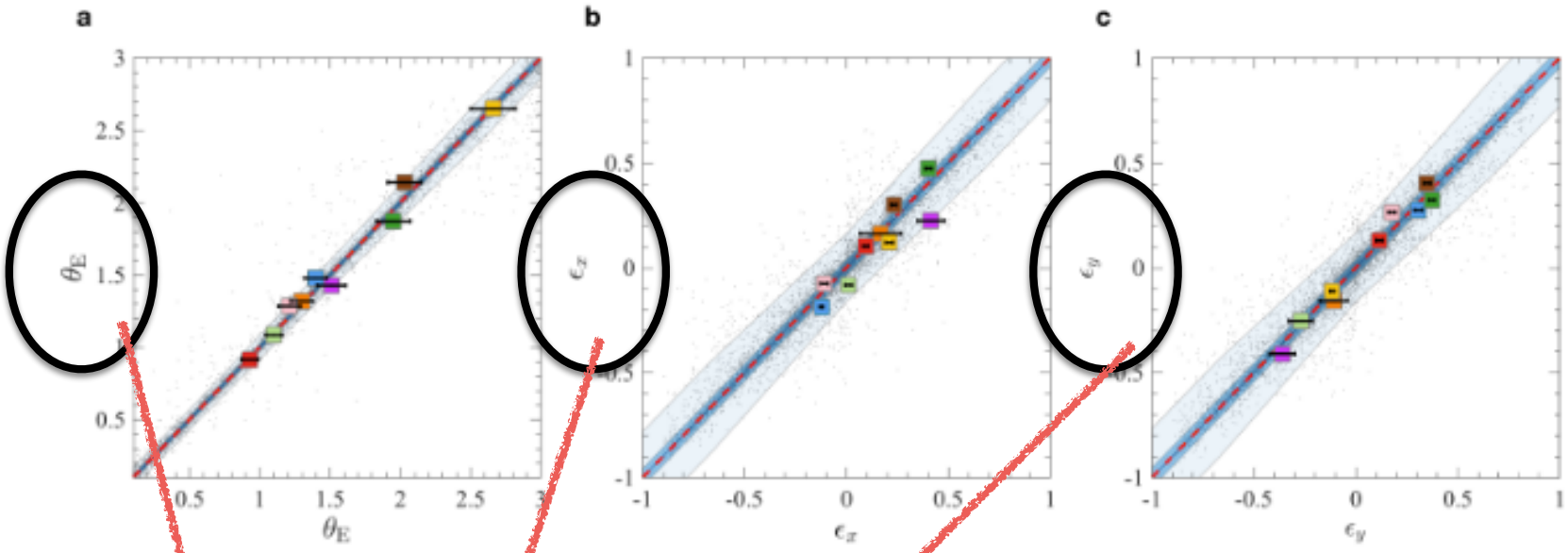
SUPERVISED REGRESSION



Hezaveh+17, Nature

REGRESSION ON
STRONG LENSES PARAMETERS

SUPERVISED REGRESSION

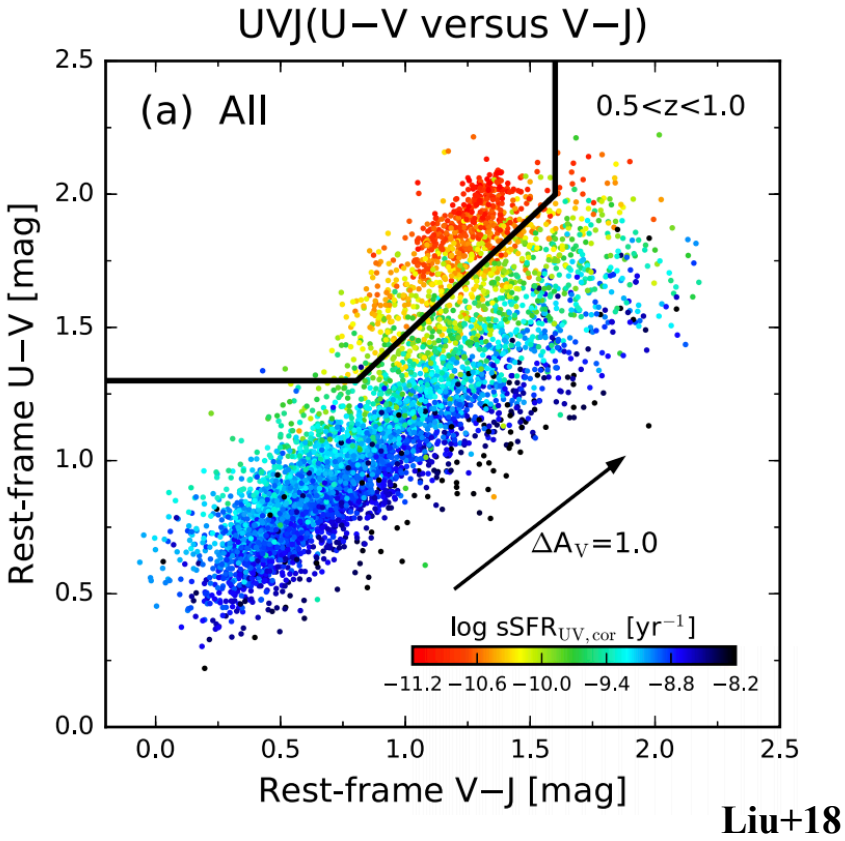


Hezaveh+17, Nature

REGRESSION ON
STRONG LENSES PARAMETERS

THRE IS NO MAGIC IN MACHINE LEARNING, AND IT IS ACTUALLY PRETTY SIMPLE

SEPARATE
STAR-FORMING
FROM
PASSIVE GALAXIES

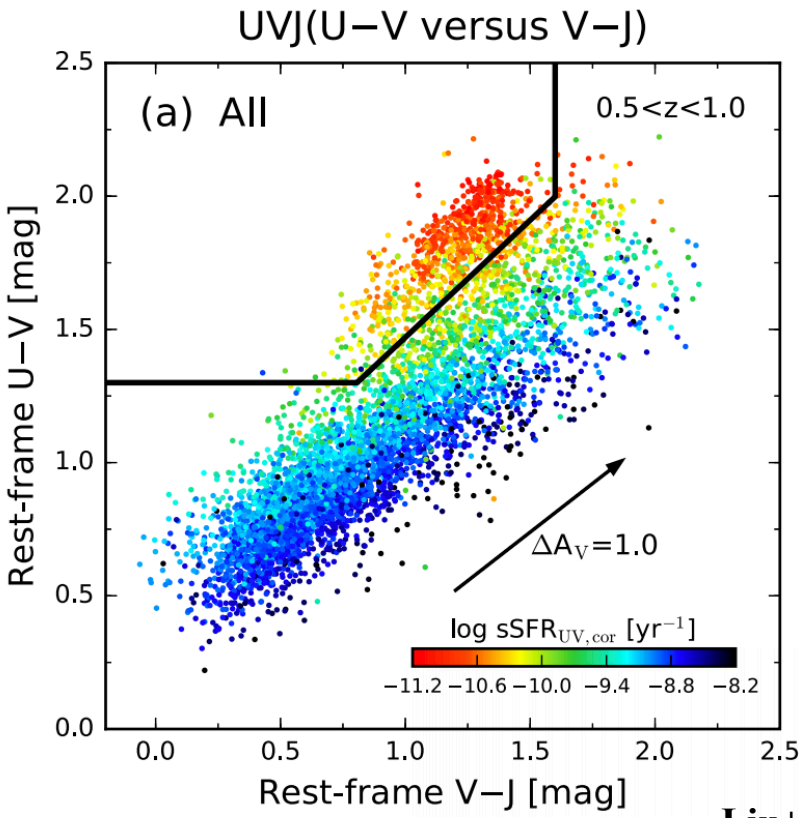


Liu+18

COLOR-COLOR PLOT

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

SEPARATE
STAR-FORMING
FROM
PASSIVE GALAXIES



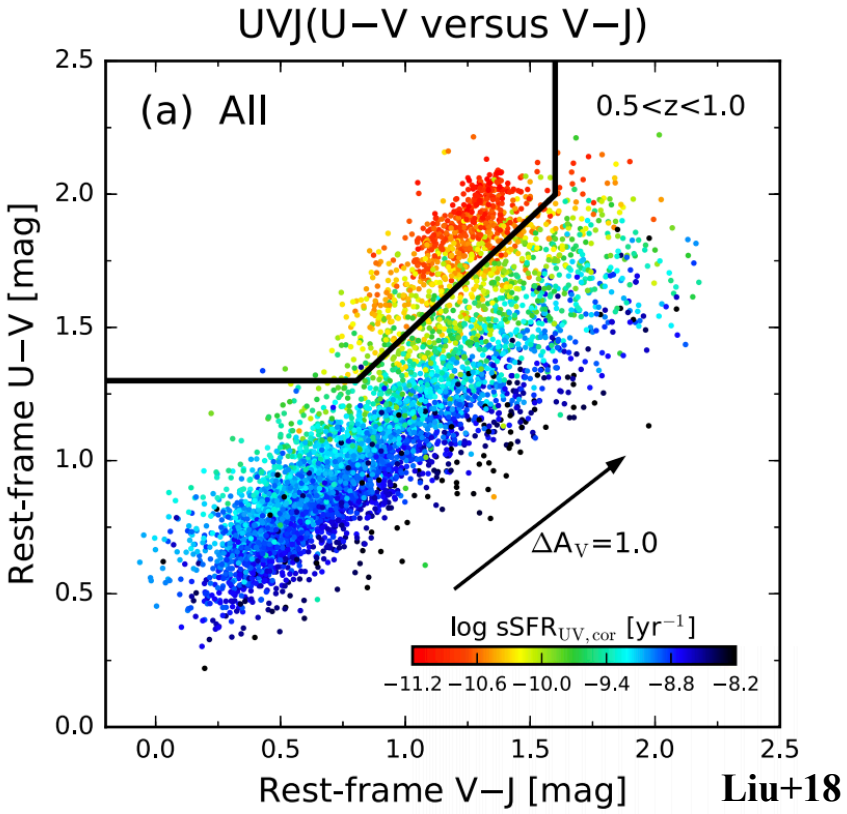
Liu+18

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

LABEL

Q(0) , SF(1)

SEPARATE
STAR-FORMING
FROM
PASSIVE GALAXIES

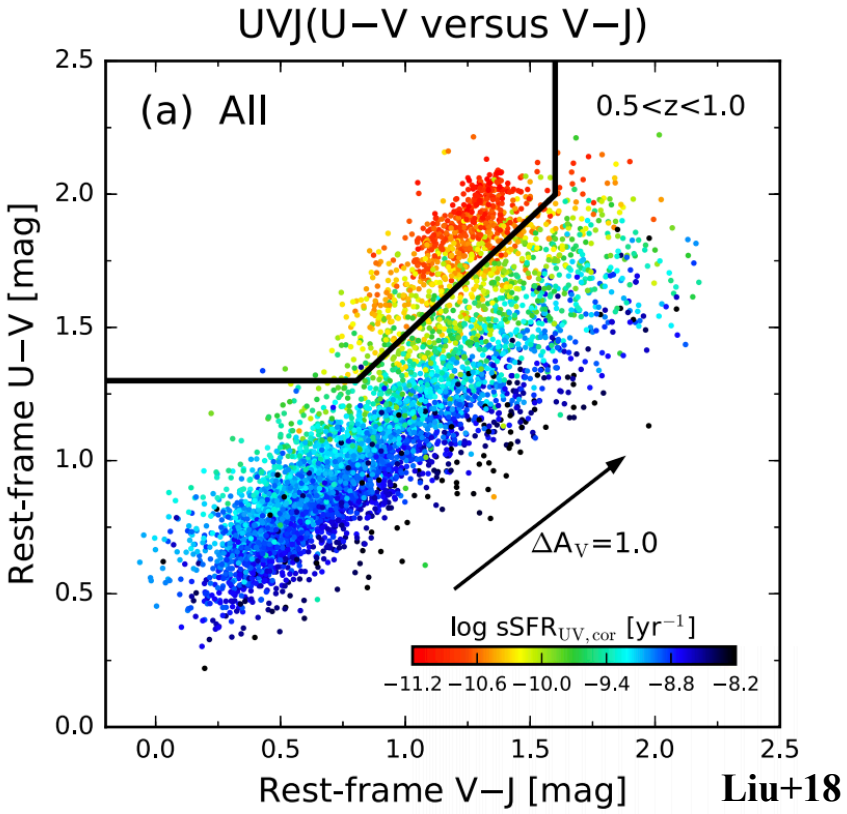


$$f_{\vec{W}}(\vec{x}) = \vec{y}$$

LABEL

Q(0) , SF(1)

(U-V, V-J) FEATURES



Liu+18

$$f_{\vec{W}}(\vec{x}) = \vec{y}$$

LABEL

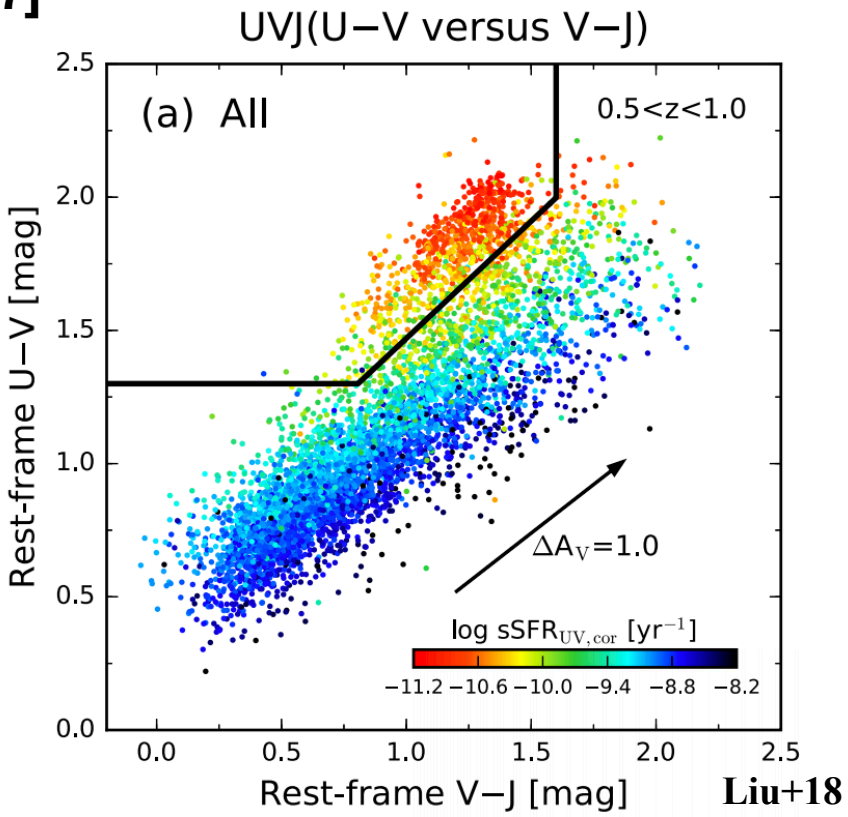
Q(0) , SF(1)

NETWORK FUNCTION

(U-V, V-J) FEATURES

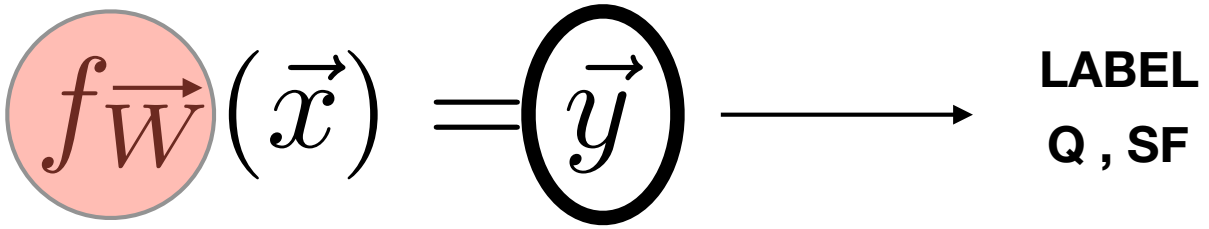
sgn[(u-v)-0.8*(v-j)-0.7]

WEIGHTS



Liu+18

“CLASSICAL”
MACHINE LEARNING



sgn[(u-v)-W1*(v-j)-W2]

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

SUPERVISED LEARNING: SOME FORMALISM

Given a dataset with known labels (measurements) - find a function that can assign (predict) measurements for an unlabeled dataset

Training set

$(\vec{x_1}, \vec{x_2}, \vec{x_3}, ..., \vec{x_n})$ → Measurements (colors, fluxes, spectra indices…)

$(\vec{y_1}, \vec{y_2}, \vec{y_3}, ..., \vec{y_n})$ → Label (morphology, object type, transit …)

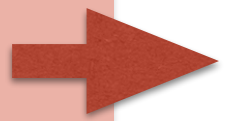
SUPERVISED LEARNING: SOME FORMALISM

Given a dataset with known labels (measurements) - find a function that can assign (predict) measurements for an unlabeled dataset

Training set

$(\vec{x_1}, \vec{x_2}, \vec{x_3}, ..., \vec{x_n})$

$(\vec{y_1}, \vec{y_2}, \vec{y_3}, ..., \vec{y_n})$



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

?

SUPERVISED LEARNING: SOME FORMALISM

Unlabeled set

$(\vec{x_1}', \vec{x_2}', \vec{x_3}', \ldots, \vec{x_n}')$

$(\vec{y_1}', \vec{y_2}', \vec{y_3}', \ldots, \vec{y_n}')$

Training set

$(\vec{x_1}, \vec{x_2}, \vec{x_3}, \ldots, \vec{x_n})$

$(\vec{y_1}, \vec{y_2}, \vec{y_3}, \ldots, \vec{y_n})$

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

?

$$(\vec{x_1}, \vec{x_2}, \vec{x_3}, \ldots, \vec{x_n}) \qquad \vec{x} \in \mathbb{R}^d$$

$$(\vec{y_1}, \vec{y_2}, \vec{y_3}, \ldots, \vec{y_n}) \qquad \vec{y} \in \mathbb{R} \qquad \vec{y} \in \mathbb{N}$$

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

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

Number of parameters - can be large

IT IS TRANSLATED INTO A MINIMIZATION PROBLEM : FIND F SUCH AS THE PREDICTION ERROR IS MINIMAL OVER ALL UNSEEN VECTORS

We need two key elements

A LOSS FUNCTION

A MINIMIZATION OR OPTIMIZATION ALGORITHM

We need two key elements

A LOSS FUNCTION

A MINIMIZATION OR OPTIMIZATION ALGORITHM

THIS IS COMMON TO ALL MACHINE LEARNING ALGORITHMS

1. DEFINE A LOSS FUNCTION

$$loss(F_W(.), \vec{x_i}, \vec{y_i})$$

Typically: $(F_W(\vec{x_i}) - \vec{y_i})^2$ Quadratic loss function

2. MINIMIZE THE EMPIRICAL RISK

$$\Re_{empirical}(W) = \frac{1}{N} \sum_{i}^{N} [loss(W, \vec{x}, \vec{y})]$$

MINIMIZE THE RISK

EMPIRICAL RISK?

$$\Re_{empirical}(W) = \frac{1}{N}\sum_{i}^{N}[loss(W, \vec{x}, \vec{y})]$$

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

EMPIRICAL RISK?

$$\Re_{empirical}(W) = \frac{1}{N}\sum_{i}^{N}[loss(W, \vec{x}, \vec{y})]$$

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

OBSERVED DATASET

EMPIRICAL RISK?

$$\Re_{empirical}(W) = \frac{1}{N}\sum_{i}^{N}[loss(W, \vec{x}, \vec{y})]$$

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

ALL “GALAXIES IN THE UNIVERSE”

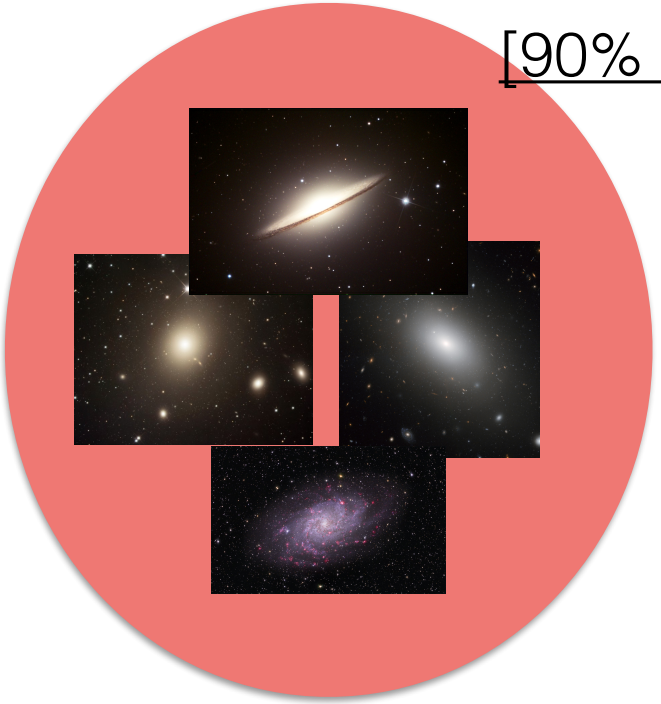
OBSERVED DATASET

THIS IS A VERY IMPORTANT ASSUMPTION. EVERY MACHINE WILL BE OPTIMIZED TO MINIMIZE THE RISK IN THE OBSERVED DATASET

THIS IS A VERY IMPORTANT ASSUMPTION. EVERY MACHINE WILL BE OPTIMIZED TO MINIMIZE THE RISK IN THE OBSERVED DATASET

IMAGINE AN EXTREME CASE WITH VERY UNBALANCED DATA…

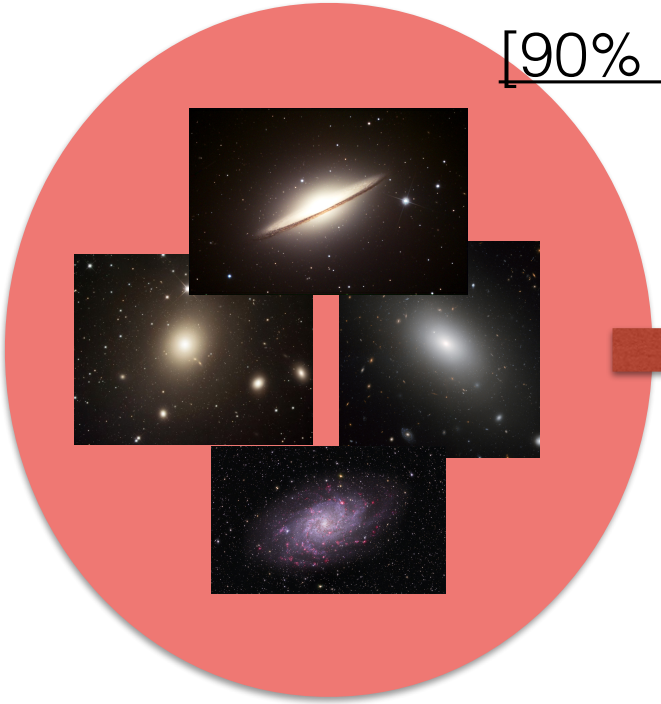
[90% E, 10% S]



THIS IS A VERY IMPORTANT ASSUMPTION. EVERY MACHINE WILL BE OPTIMIZED TO MINIMIZE THE RISK IN THE OBSERVED DATASET

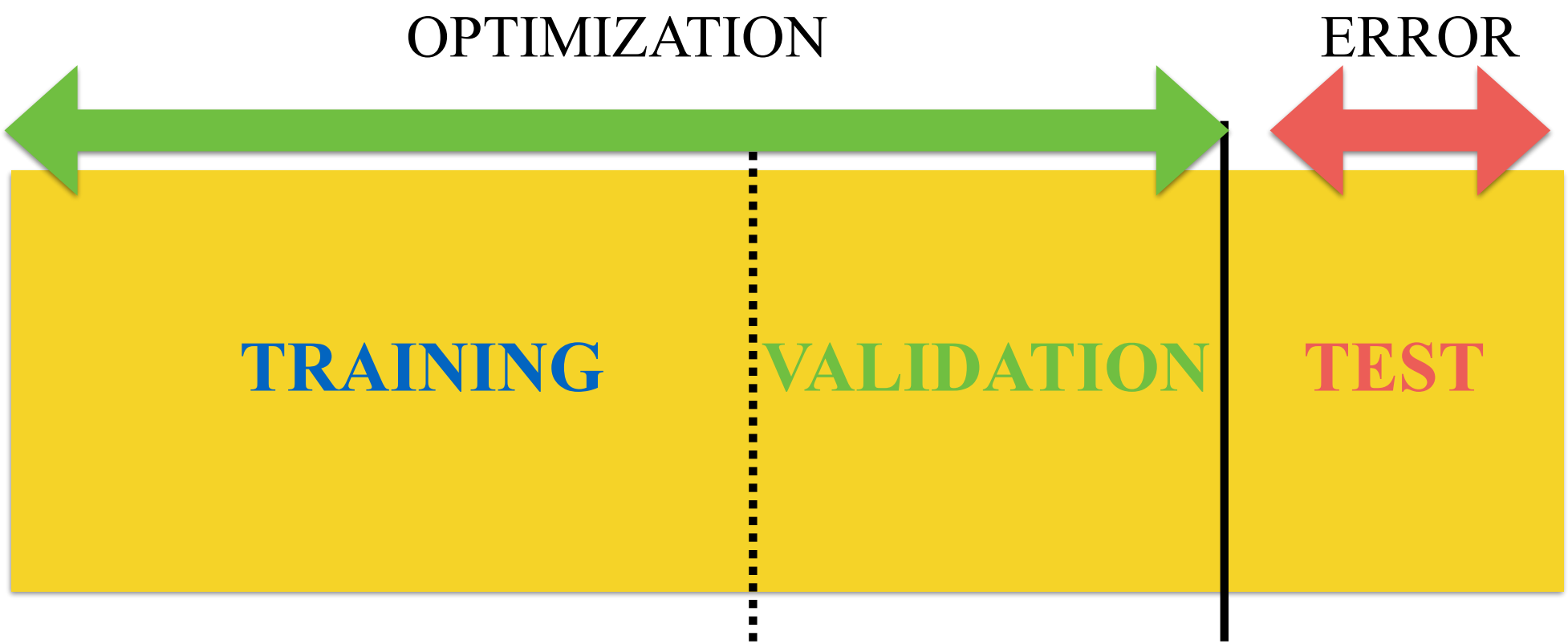
IMAGINE AN EXTREME CASE WITH VERY UNBALANCED DATA…

[90% E, 10% S]



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

In practice

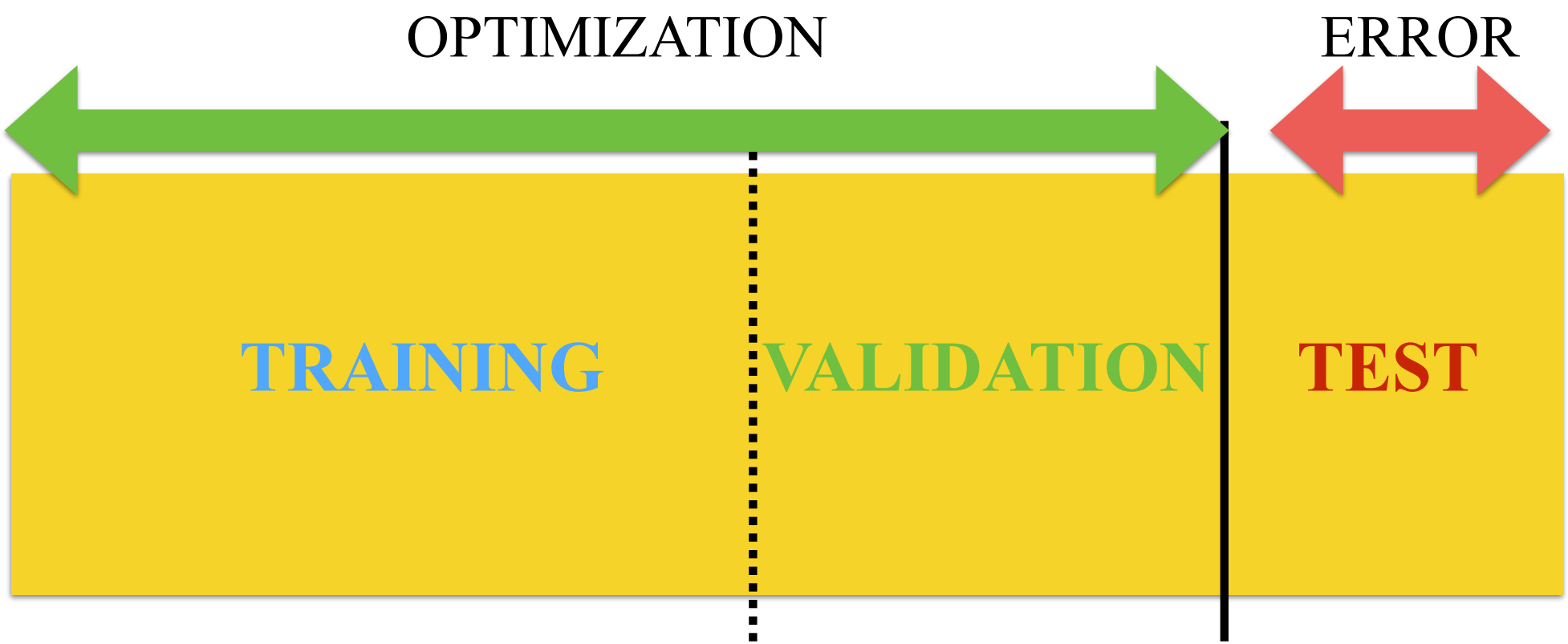


training set: use to train the classifier

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 minimize is called OPTIMIZATION

THERE ARE SEVERAL OPTIMIZATION TECHNIQUES

OPTIMIZATION

THERE ARE SEVERAL OPTIMIZATION TECHNIQUES

THEY DEPEND ON THE MACHINE LEARNING ALGORITHM

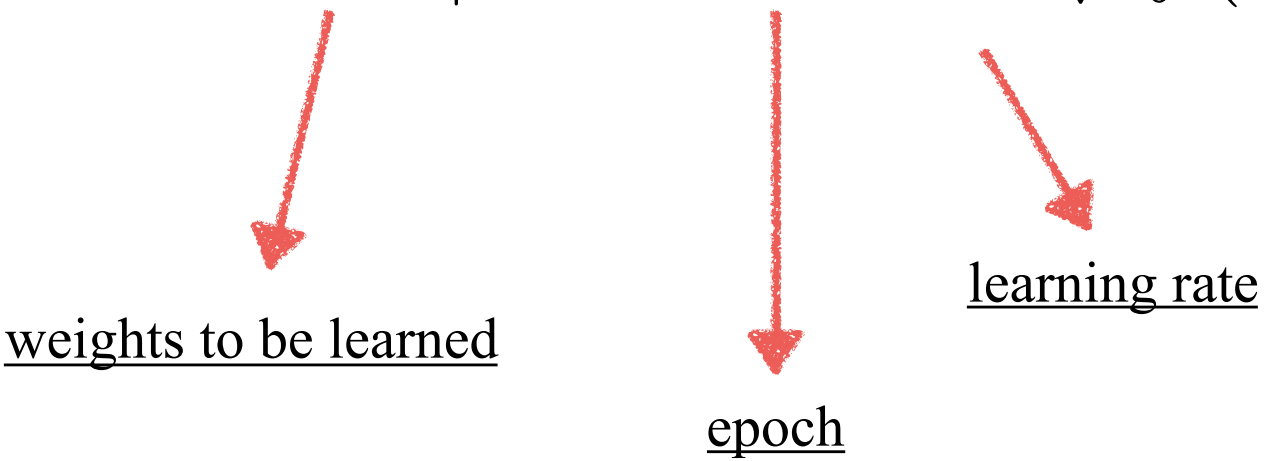
OPTIMIZATION

THERE ARE SEVERAL OPTIMIZATION TECHNIQUES

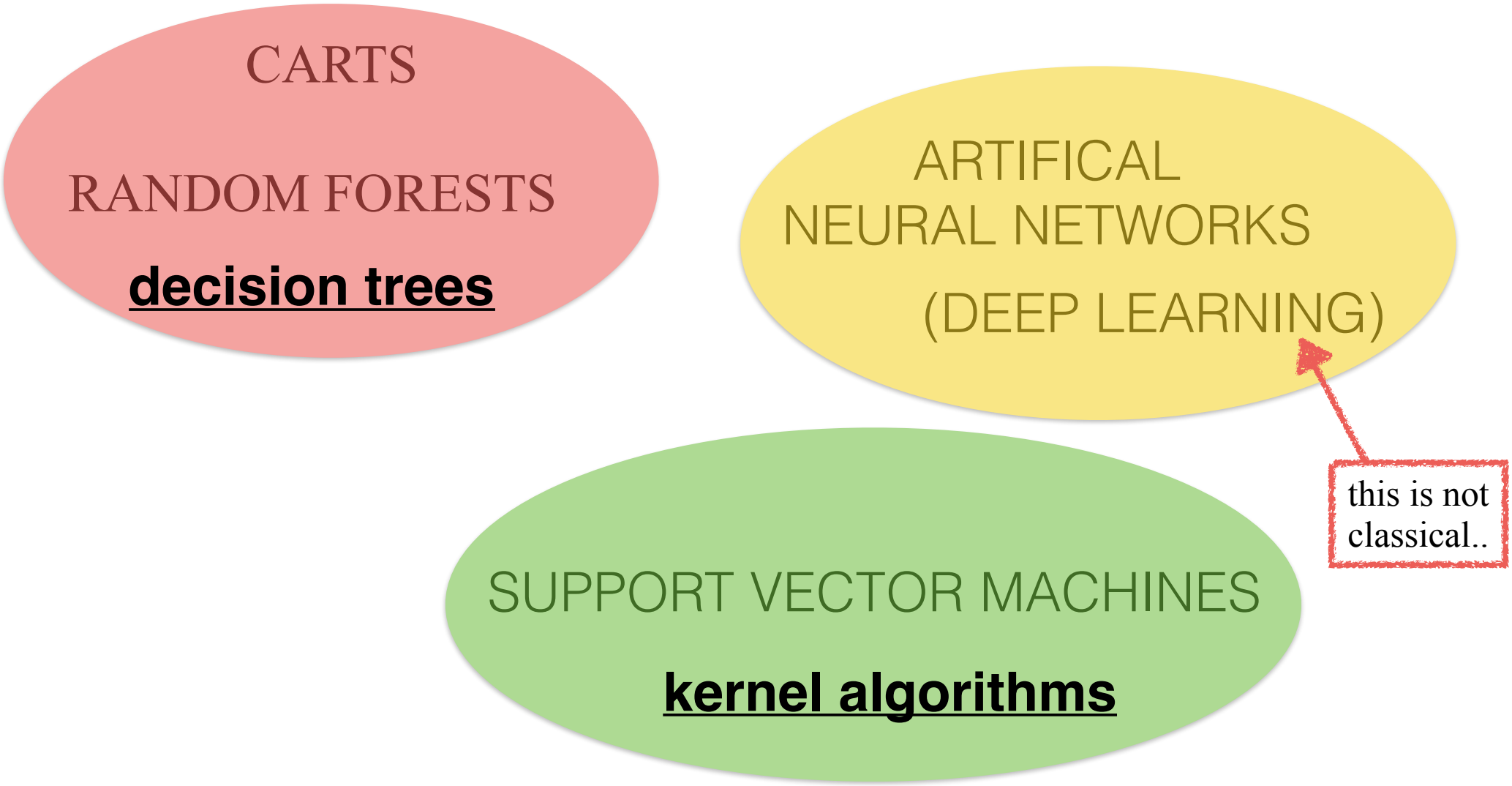
THEY DEPEND ON THE MACHINE LEARNING ALGORITHM

NEURAL NETWORKS USE THE GRADIENT DESCENT AS WE WILL SEE LATER

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



DIFFERENT "CLASSICAL" SUPERVISED MACHINE LEARNING METHODS



THE DIFFERENCES
ARE
IN THE FUNCTION
THAT IS USED

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

CARTS
RANDOM FORESTS

decision trees

ARTIFICAL
NEURAL NETWORKS

(DEEP LEARNING)

SUPPORT VECTOR MACHINES

kernel algorithms

THE DIFFERENCES ARE IN THE FUNCTION THAT IS USED

Standard Chi2 model fitting is a type of machine learning … with a priori known parameters

All algorithms minimize a loss function

RANDOM FORESTS

CARTS

decision trees

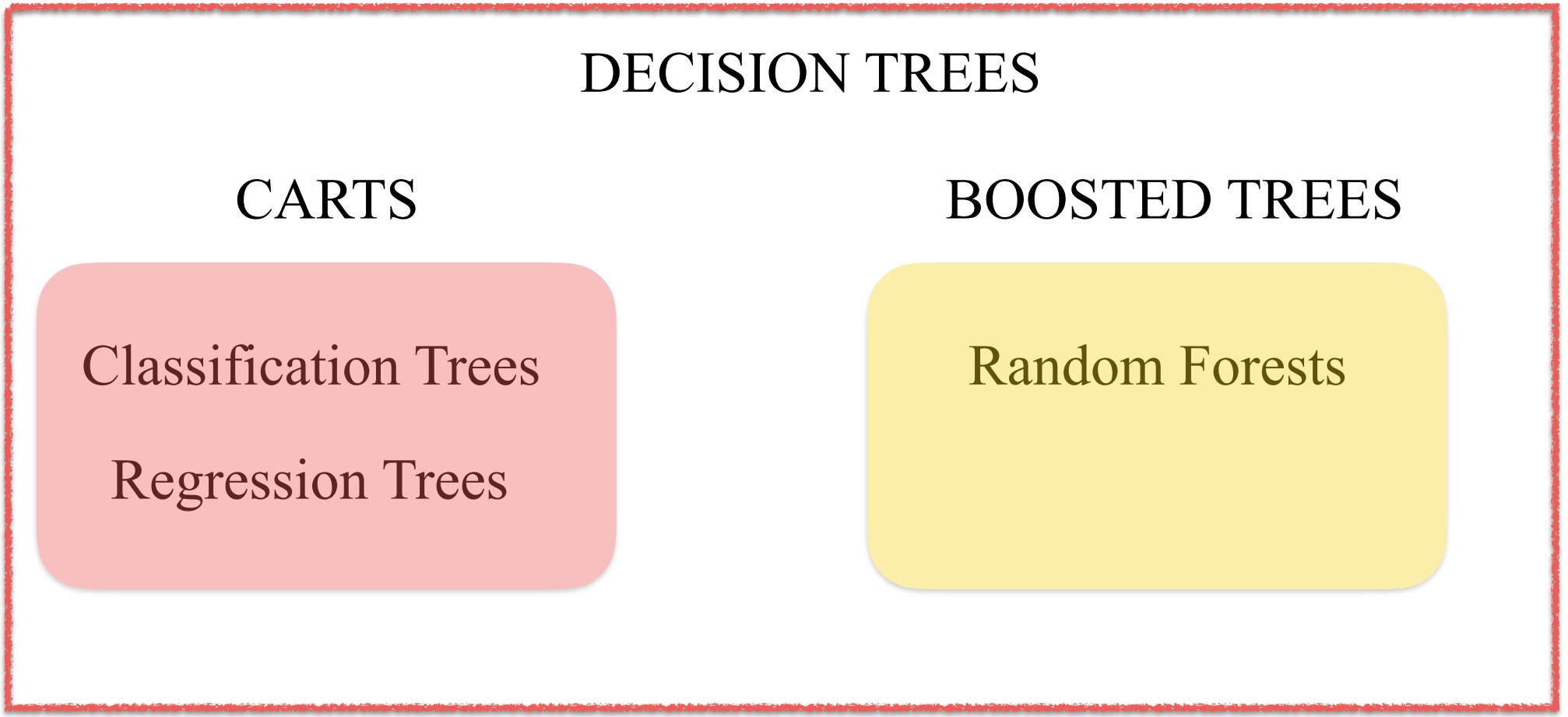
ARTIFICAL NEURAL NETWORKS

(DEEP LEARNING)

SUPPORT VECTOR MACHINES

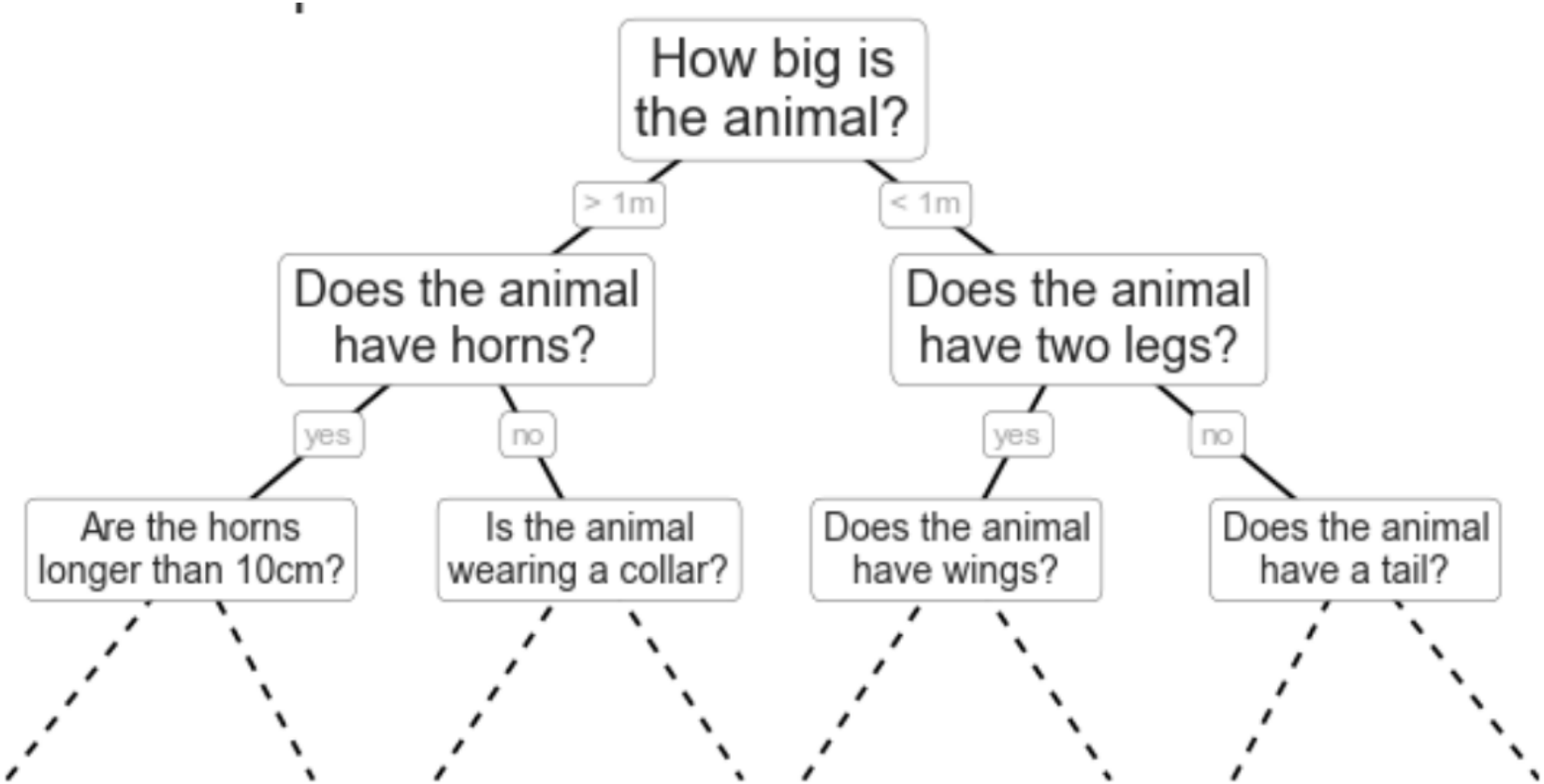
kernel algorithms

DECISION TREES ALGORITHMS

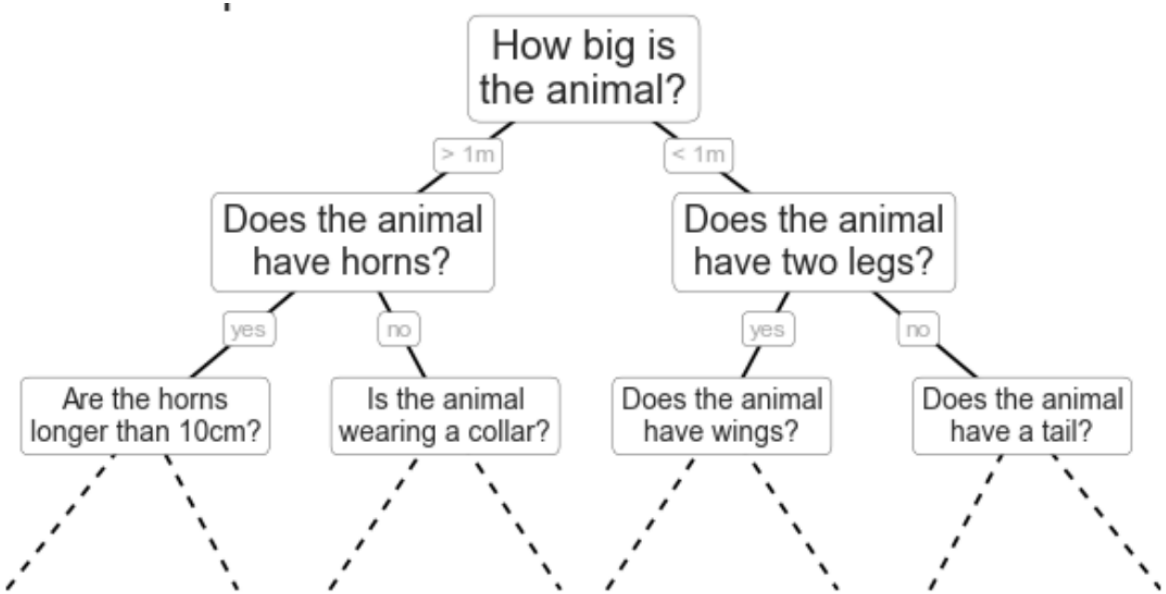


CLASSIFICATION AND REGRESSION TREES (CARTS)

THIS IS THE SIMPLEST AND MORE INTUITIVE MACHINE LEARNING ALGORITHM



DECISION TREES (CARTS)



IT IS BUILT IN AN ITERATIVE WAY

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)

TRAINING DATA

$$\vec{x_i} = (p_1, p_2, ..., p_n)$$

(parameters)

[b/a, color, size, SFR …]

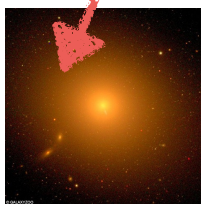
$(\vec{x_2}, \vec{y_2})$

$(\vec{x_1}, \vec{y_1})$

$(\vec{x_3}, \vec{y_3})$ $(\vec{x_4}, \vec{y_4})$

$$\vec{y_i} = (y_1, y_2, ..., y_n)$$

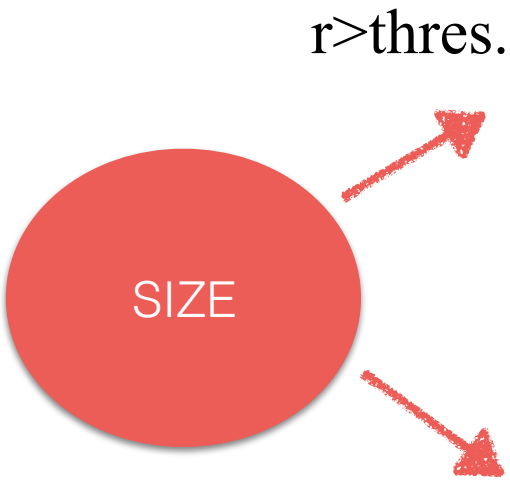
(classes)



[elliptical / spiral]

DECISION TREES (CARTS)

[elliptical / spiral]



LOOK FOR “OPTIMAL” THRESHOLD

DECISION TREES (CARTS)

[elliptical / spiral]

SIZE

r>thres.

COLOR

c>thres.

c<thres.

r<thres.

COLOR

c>thres.

c<thres.

LOOK FOR "OPTIMAL" THRESHOLD

LOOK FOR 2nd "OPTIMAL" THRESHOLD

WHAT IS AN OPTIMAL THRESHOLD?

[OR WHAT LOSS FUNCTION TO MINIMIZE]

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

DECISION TREES (CARTS)

TYPICAL METRIC USED:

GINI IMPURITY $$G = 1 - \sum_{j=1}^{c} p_j^2$$

DECISION TREES (CARTS)

TYPICAL METRIC USED:

GINI IMPURITY $$G = 1 - \sum_{j=1}^{c} p_j^2$$

fraction of
objects in each class given
a split value

DECISION TREES (CARTS)

TYPICAL METRIC USED:

GINI IMPURITY

$$G = 1 - \sum_{j=1}^{c} p_j^2$$

IF THERE IS ONLY ONE CLASS
THE GINI IMPURITY GOES TO 0

fraction of
objects in each class given
a split value

RANDOM FORESTS

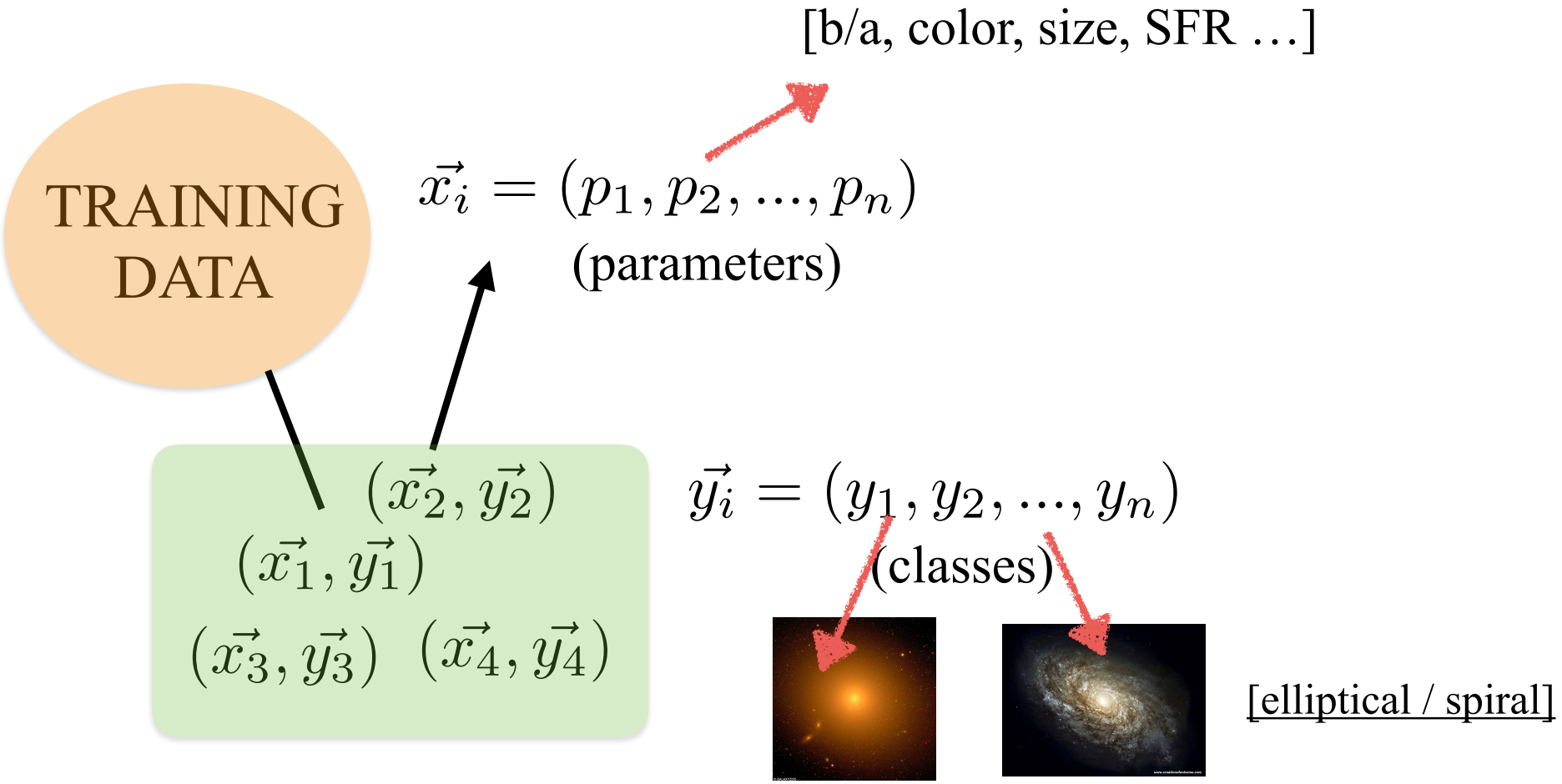
ONE PROBLEM WITH CLASSIFICATION TREES IS THAT THEY CAN EASILY OVERFIT

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

ALSO DEPENDENT ON THE ORDER FEATURES ARE CONSIDERED

RANDOM FORESTS

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



RANDOM FORESTS

$(\vec{x_2}, \vec{y_2})$

$(\vec{x_1}, \vec{y_1})$

$(\vec{x_3}, \vec{y_3})$ $(\vec{x_4}, \vec{y_4})$

$\vec{x_i} = (p_1, p_2, ..., p_n)$

RANDOM FORESTS

$(\vec{x_2}, \vec{y_2})$
$(\vec{x_1}, \vec{y_1})$
$(\vec{x_3}, \vec{y_3})$ $(\vec{x_4}, \vec{y_4})$

$\vec{x_i} = (p_1, p_2, .., p_n)$

[elliptical / spiral]

SIZE

r>thres.

COLOR

c>thres.

c<thres.

r<thres.

COLOR

c>thres.

c<thres.

LOOK FOR THRES. SO THAT GINI IMPURITY IS MINIMAL

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

RANDOM FORESTS

$(\vec{x_2}, \vec{y_2})$

$(\vec{x_1}, \vec{y_1})$

$(\vec{x_3}, \vec{y_3})$ $(\vec{x_4}, \vec{y_4})$

$\vec{x_i} = (p_1, p_2, ..., p_n)$

[elliptical / spiral]

SIZE

r>thres.

Sersic

s>thres.

s<thres.

r<thres.

Sersic

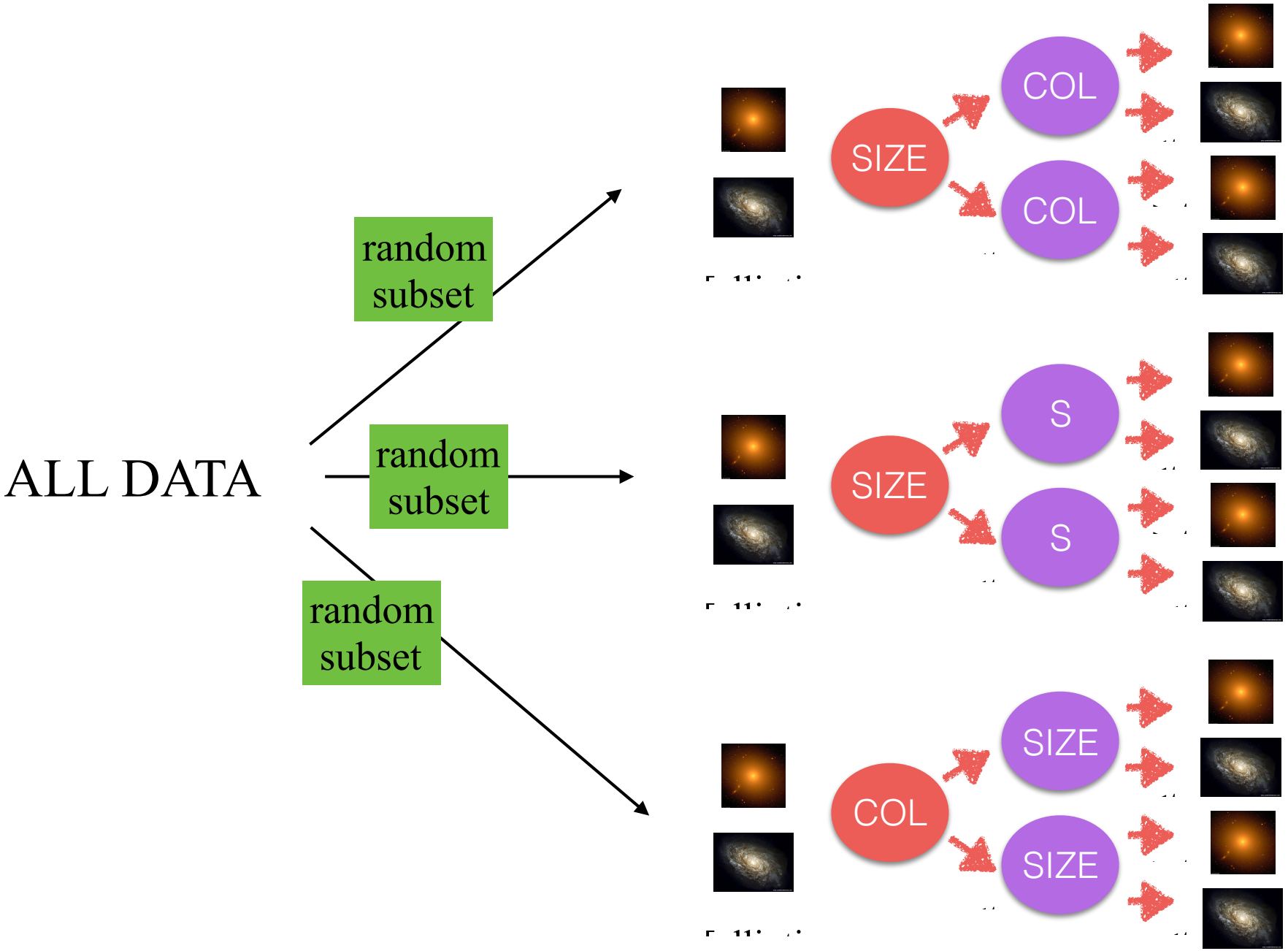
s>thres.

s<thres.

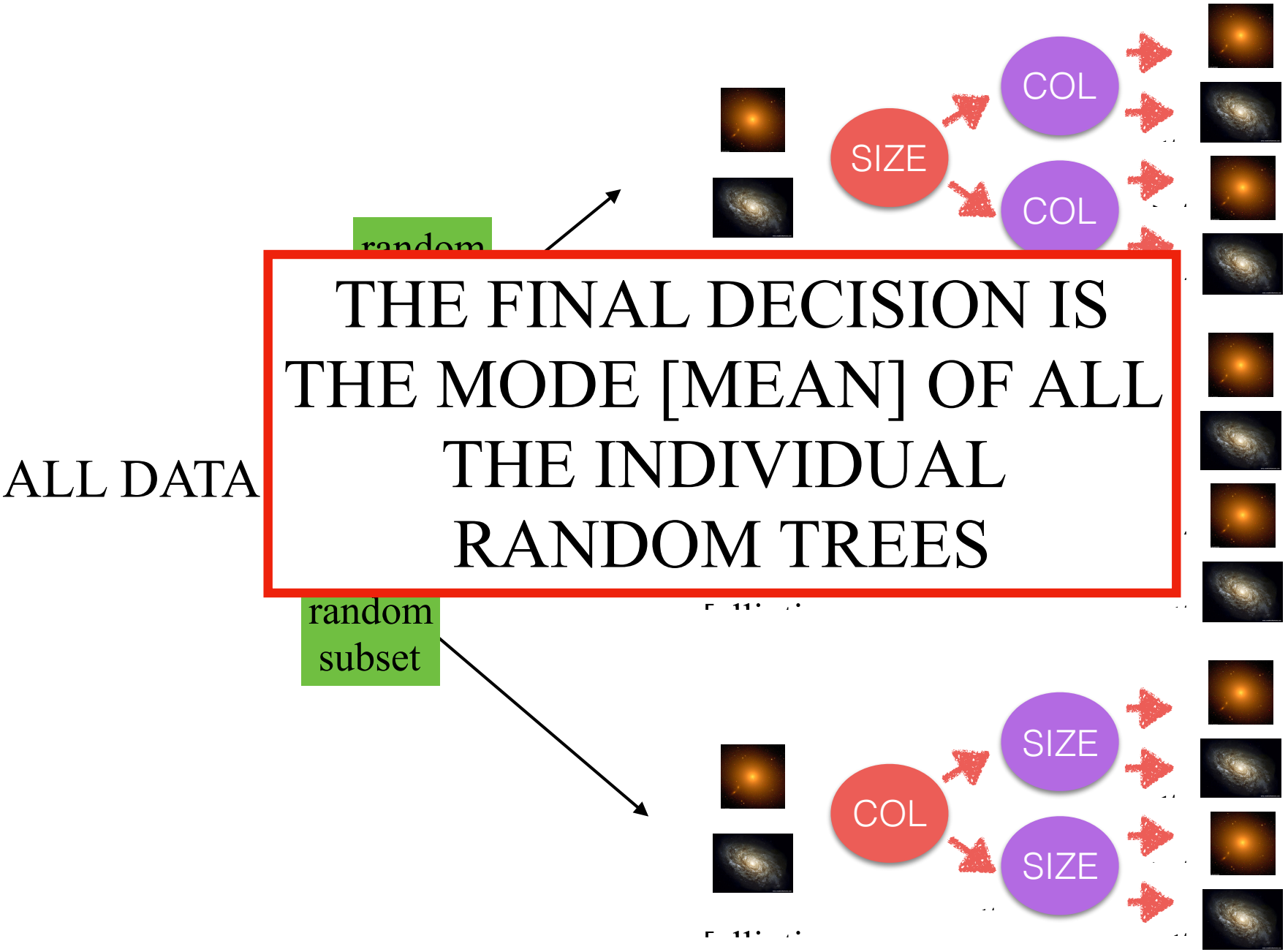
LOOK FOR THRES. SO THAT GINI IMPURITY IS MINIMAL

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

ONE CAN EASILY DETERMINE THE MOST IMPORTANT FEATURES TO TAKE DECISIONS

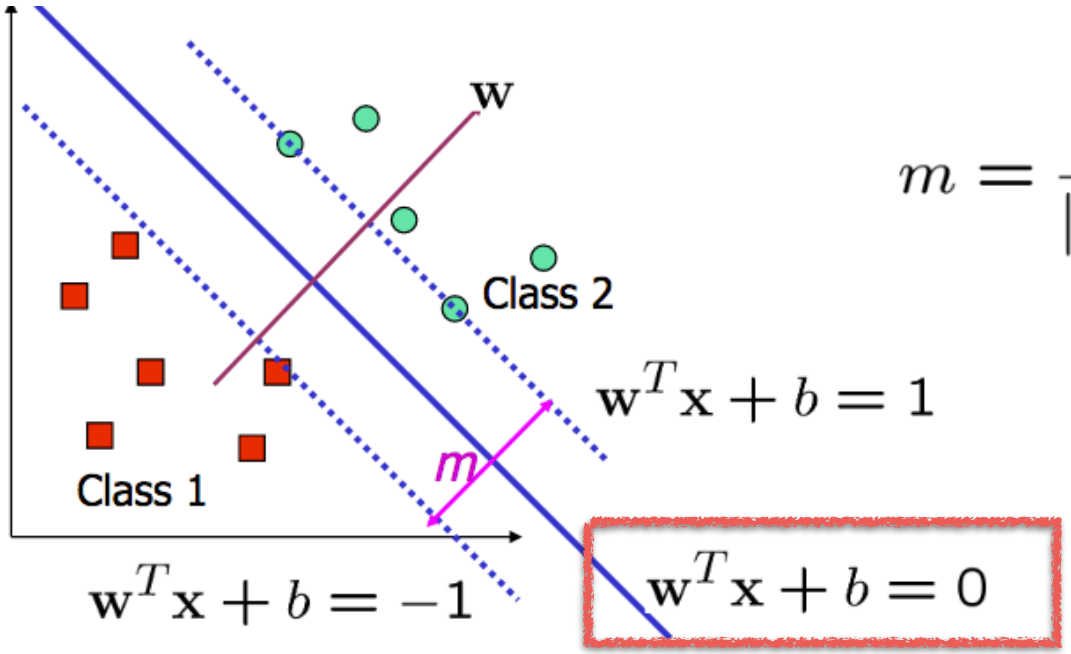
Rank	Property	AUC	Success label[a]
	ALL	0.9074 ± 0.0106	Outstanding
1	CVD	0.8559 ± 0.0039	Excellent
2	M_{bulge}	0.8335 ± 0.0060	Excellent
3	B/T	0.8267 ± 0.0028	Excellent
4	M_{halo}	0.7983 ± 0.0045	Acceptable
5	M_*	0.7819 ± 0.0025	Acceptable
6	M_{disc}	0.7124 ± 0.0016	Acceptable
7	δ_5	0.5894 ± 0.0015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

IMPORTANCE OF PARAMETERS T TO PREDICT IF A GALAXY IS QUENCHED

(BE CAREFUL WITH CORRELATIONS)

Teimiroonia+16

Support Vector Machines

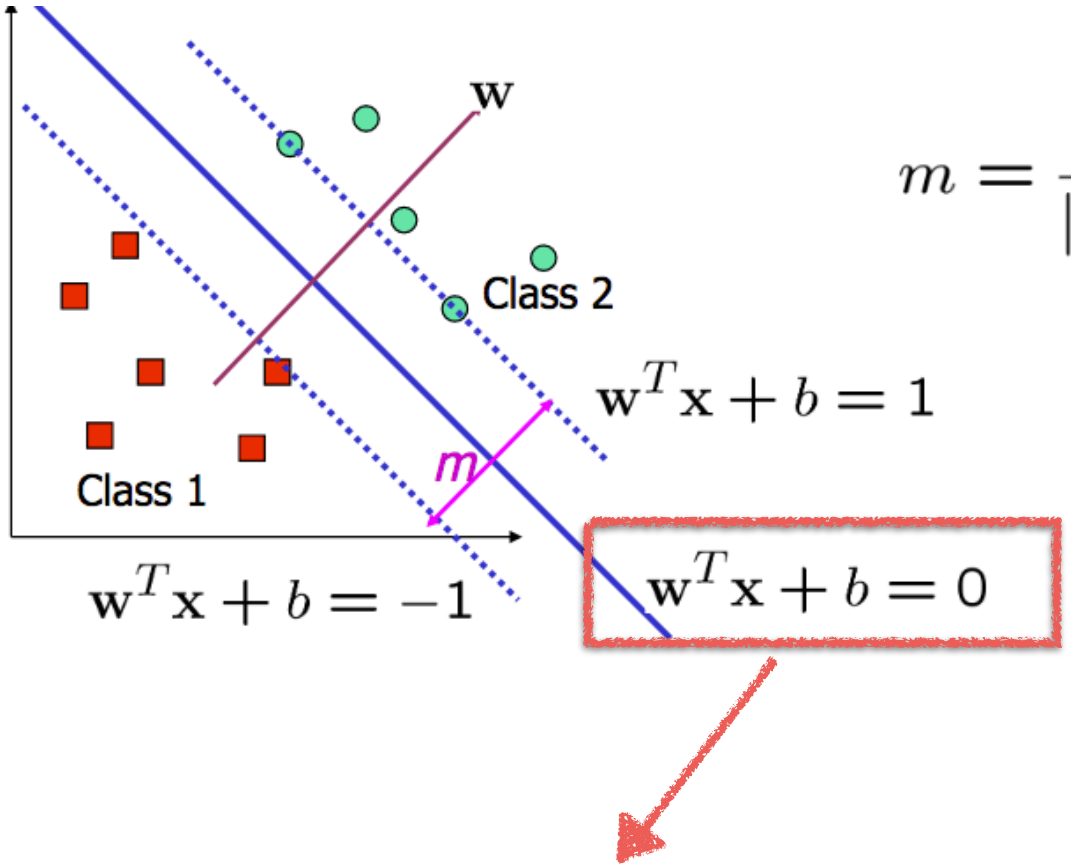


$$m = \frac{2}{||\mathbf{w}||}$$

equation of the hyperplane

[Credit: M. Law]

Support Vector Machines



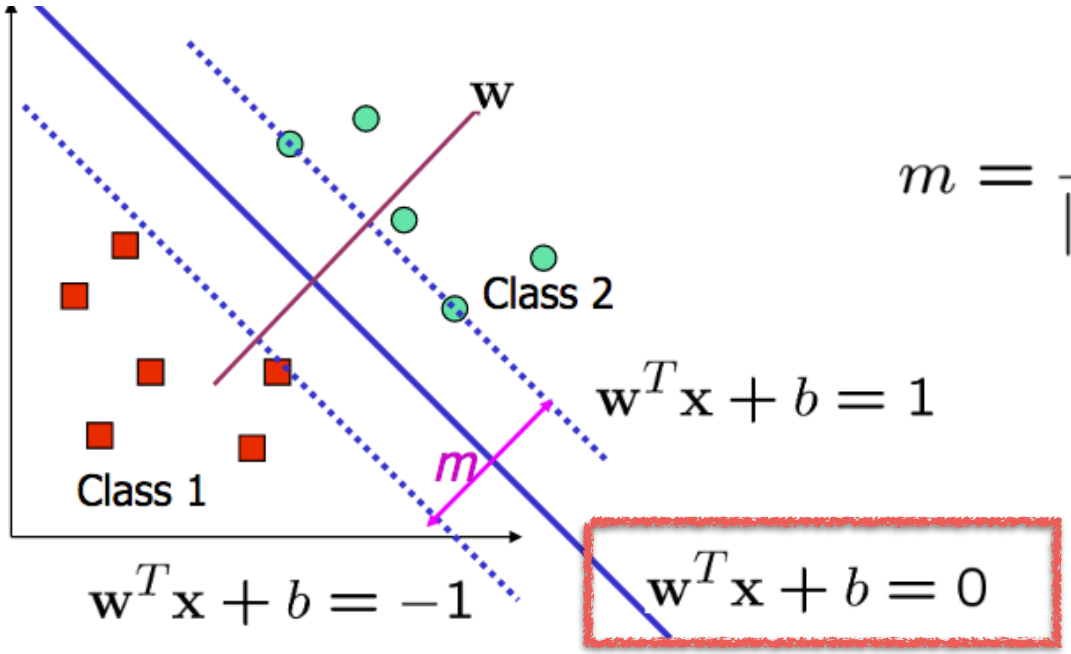
$$m = \frac{2}{||\mathbf{w}||}$$

equation of the hyperplane

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

[Credit: M. Law]

Support Vector Machines



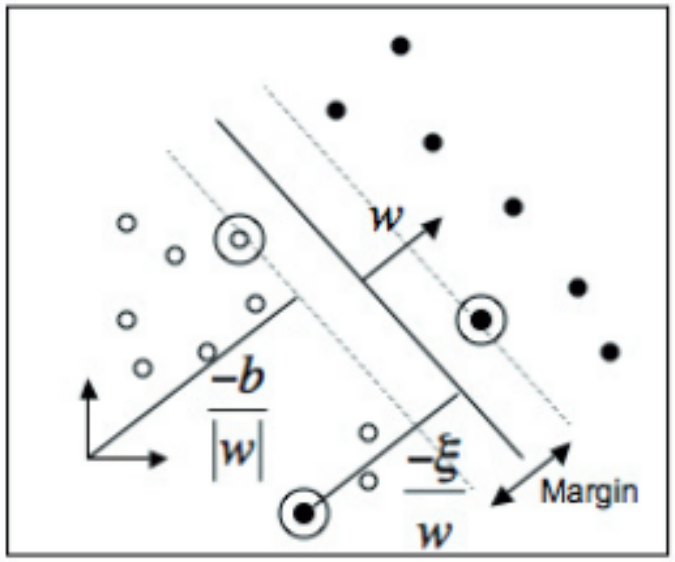
$$m = \frac{2}{||\mathbf{w}||}$$

equation of the hyperplane

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

[Credit: M. Law]

Support Vector Machines



Huertas-Company+08

IN MOST CASES THE DATA ARE NON LINEARLY SEPARABLE

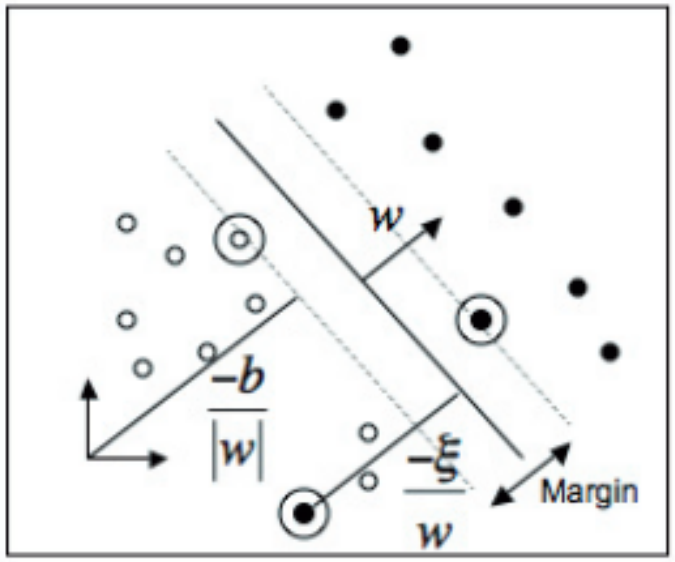
$$\mathbf{x_i}.\mathbf{w} + b \geq +1 - \epsilon_i$$

$$\mathbf{x_i}.\mathbf{w} + b \geq +1 + \epsilon_i$$

THE CONDITION IS RELAXED BY ADDING A POSITIVE STACK VARIABLE

Huertas-Company+08

Support Vector Machines



Huertas-Company+08

IN MOST CASES THE DATA ARE NON LINEARLY SEPARABLE

$$\mathbf{x_i}.\mathbf{w} + b \geq +1 - \epsilon_i$$

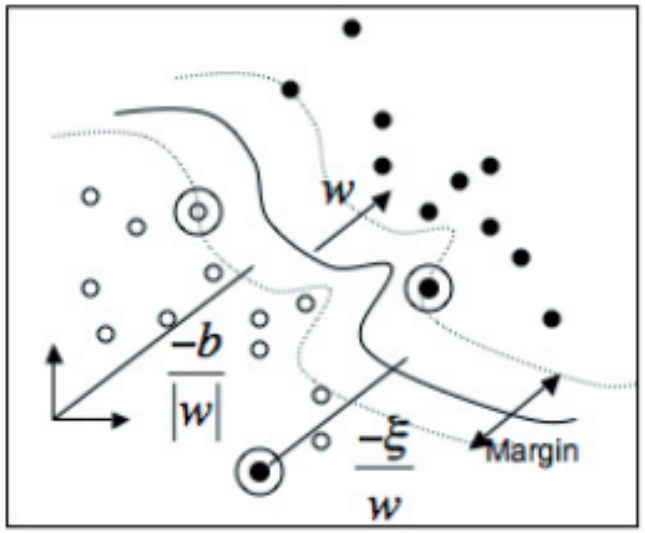
$$\mathbf{x_i}.\mathbf{w} + b \geq +1 + \epsilon_i$$

THE CONDITION IS RELAXED BY ADDING A POSITIVE STACK VARIABLE

THE FUNCTION TO MINIMIZE: $\frac{||w^2||}{2} + T[\sum \epsilon_i]$

THE LARGER T, THE MORE ERRORS ARE PENALIZED

Support Vector Machines



Huertas-Company+08

THE FINAL TRICK IS TO FIND NON-LINEAR BOUNDARIES

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

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

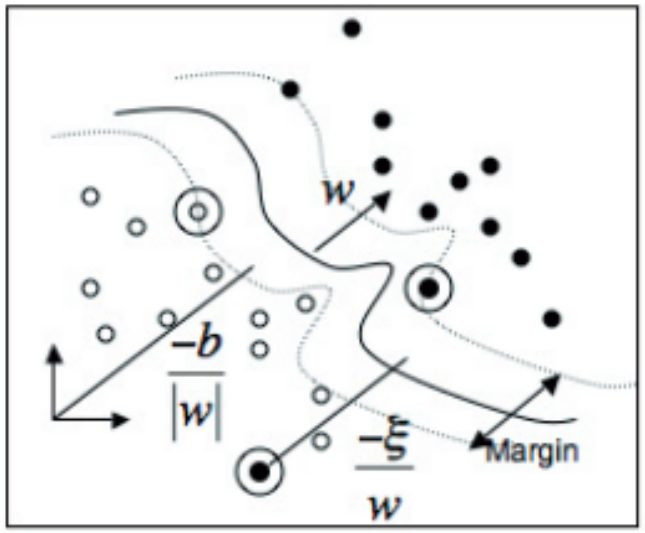
SINCE WE ONLY NEED TO COMPUTE INNER PRODUCTS IN THIS SPACE

$$x_i . x_j$$

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

$$\Phi(x_i).\Phi(x_j)$$

Support Vector Machines



Huertas-Company+08

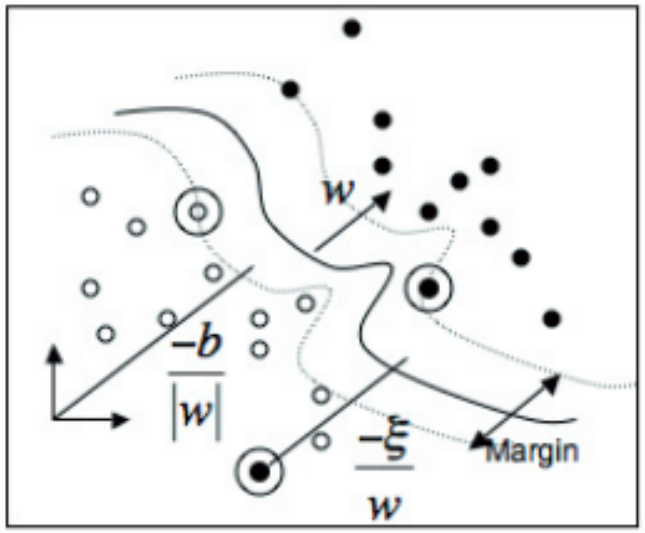
THE FINAL TRICK IS TO FIND NON-LINEAR BOUNDARIES

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

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

THEN THERE IS NO NEED TO EXPLICITLY KNOW Φ

Support Vector Machines



Huertas-Company+08

THE FINAL TRICK IS TO FIND NON-LINEAR BOUNDARIES

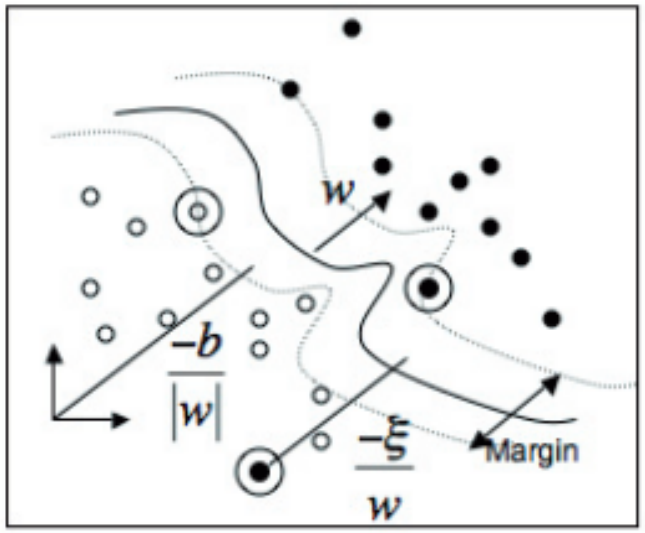
IF THERE IS A **KERNEL FUNCTION** SO THAT:

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

THEN THERE IS NO NEED TO EXPLICITLY KNOW Φ

SVM IS CALLED **A KERNEL METHOD**

Support Vector Machines



Huertas-Company+08

THE FINAL TRICK IS TO FIND NON-LINEAR BOUNDARIES

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

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

THEN THERE IS NO NEED TO EXPLICITLY KNOW Φ

SVM IS CALLED **A KERNEL METHOD**

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

[GAUSSIAN RBF]

In practice: *Python scikit learn*

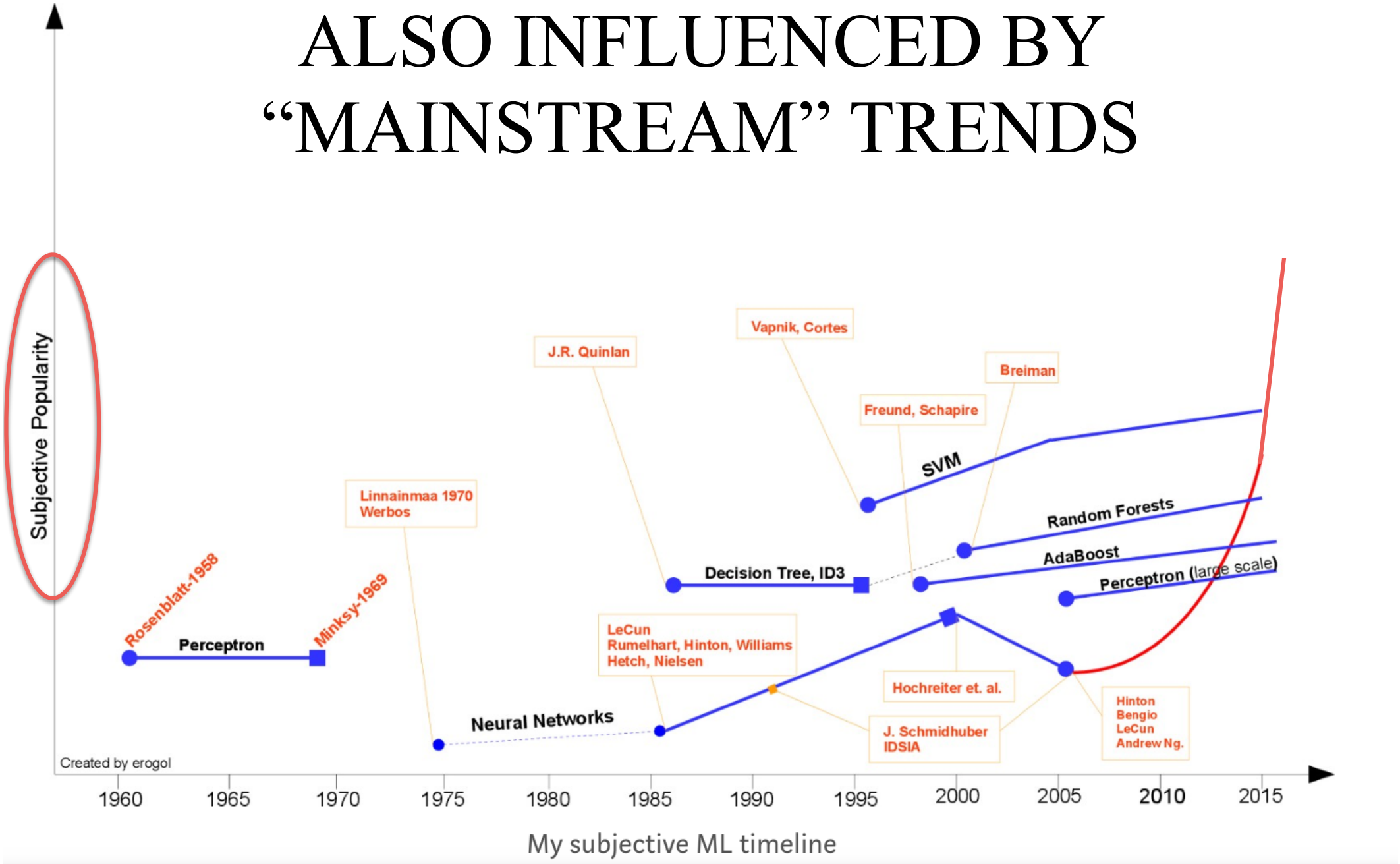
- All these different methods are standard and available in Python
- Very easy to use
- All info here

How to choose your classical classifier?

NO RULE OF THUMB - REALLY DEPENDS ON APPLICATION

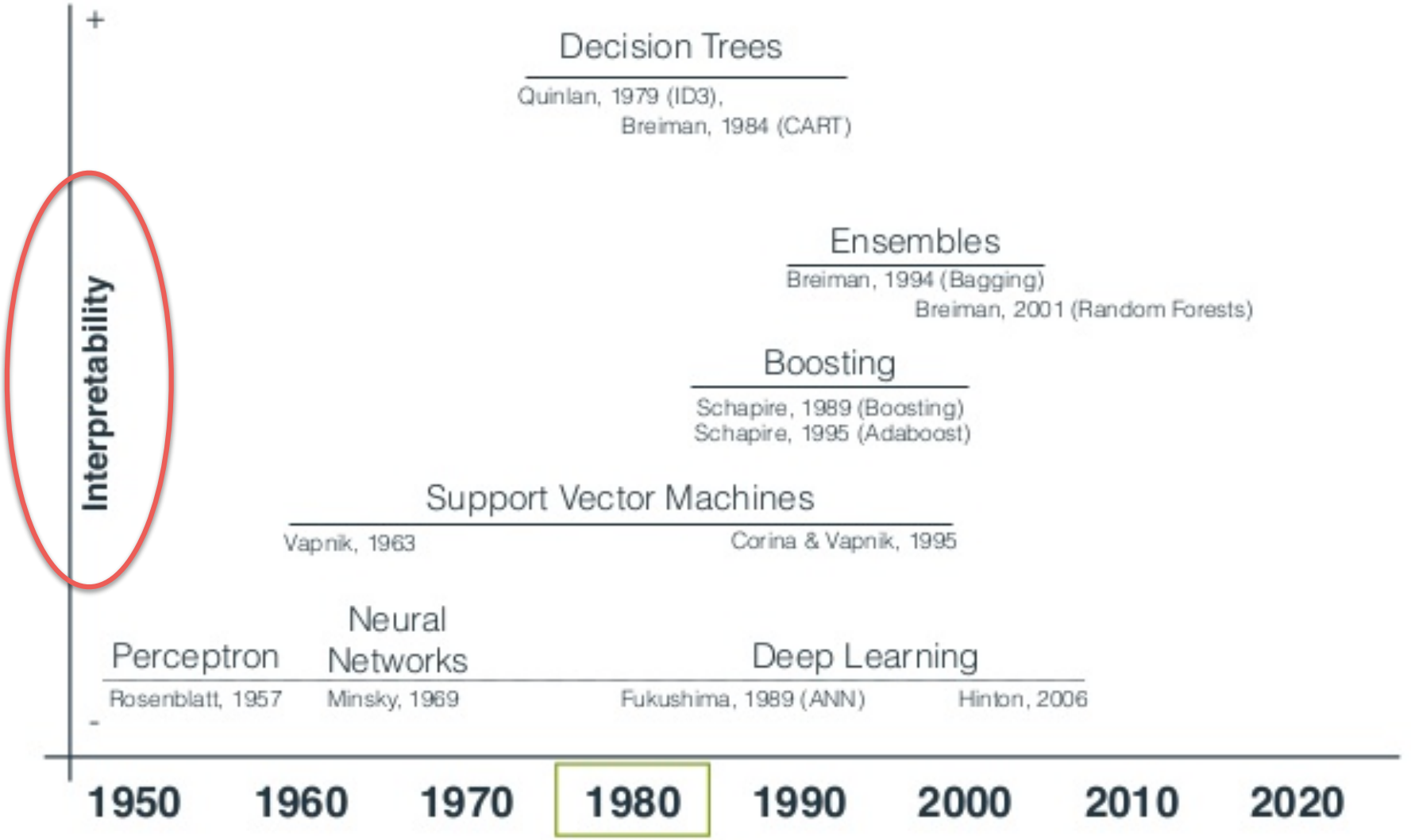
ML METHOD	++	—	Python
CARTS / RANDOM FOREST	Easy to interpret ("White box") Litte data preparation Both numerical + categorical	Over-complex trees Unstable Biased tress if some classes dominate	sklearn.ensemble.RandomForestClassifier sklearn.ensemble.RandomForestRegressor
SVM	Easy to interpret + Fast Kernel trick allows no linear problems	not very well suited to multi-class problems	sklearn.svm sklearn.svc
ANN	seed of deep-learning very efficient with large amount of data as we will see	more difficult to interpret computing intensive	sklearn.neural_network.MPL_CLassifier sklearn.neural_network.MPL_Regressor

ALSO INFLUENCED BY "MAINSTREAM" TRENDS



Source

CAN DEPEND ON YOUR MAIN INTEREST



credit

PRACTICAL NOTE: *Python scikit learn*

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

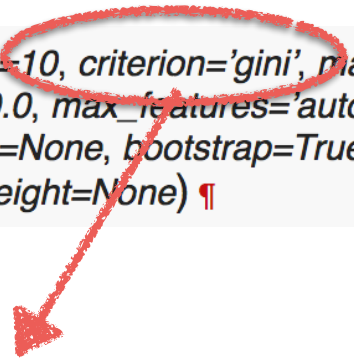
sklearn.ensemble.RandomForestClassifier

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

sklearn.ensemble.RandomForestClassifier

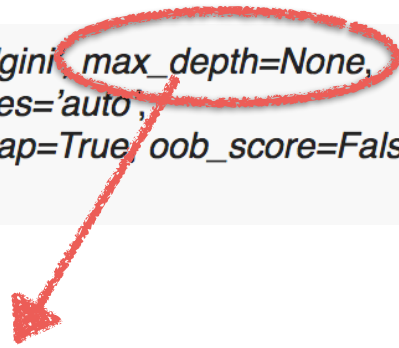
class sklearn.ensemble. **RandomForestClassifier** (*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False,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



METRIC TO MINIMIZE

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

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..

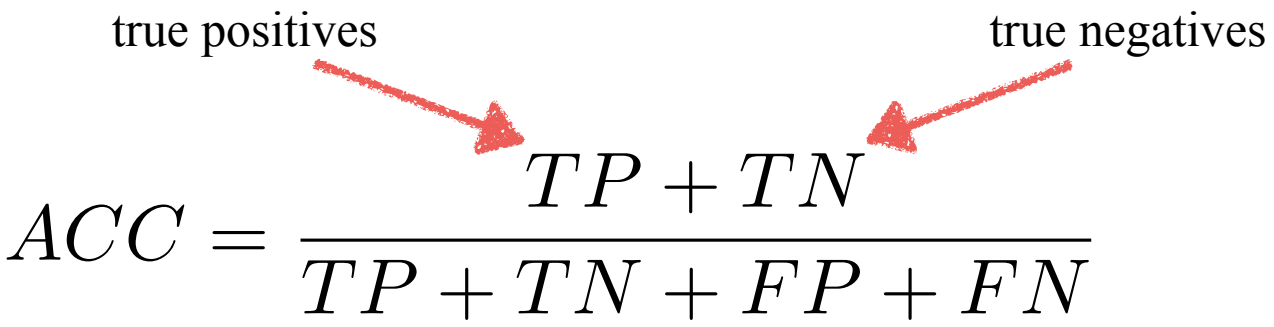
```
>>> print(clf.predict([[0, 0, 0, 0]]))
[1]
```

EVALUATION OF RESULTS

- 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..

EVALUATION: BINARY CLASSIFIERS

THE MOST STRAIGHTFORWARD WAY IS TO EVALUATE THE TOTAL ACCURACY



MEASURES HOW MANY OBJECTS ARE CORRECTLY CLASSIFIED

EVALUATION: BINARY CLASSIFIERS

THE MOST STRAIGHTFORWARD WAY IS TO EVALUATE THE TOTAL ACCURACY

NOT VERY INFORMATIVE IF UNBALANCED CLASSES

true p ue negatives

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

MEASURES HOW MANY OBJECTS ARE CORRECTLY CLASSIFIED

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.]

ver Operating Characteristic)

YOU WANT THIS TO BE AS BIG AS POSSIBLE

IT IS BASED ON TWO VERY SIMPLE PARAMETERS

1. $TPR = \frac{TP}{TP +$ TRUE POSITIVE RATE Completeness]

YOU WANT THIS TO BE AS SMALL AS POSSIBLE

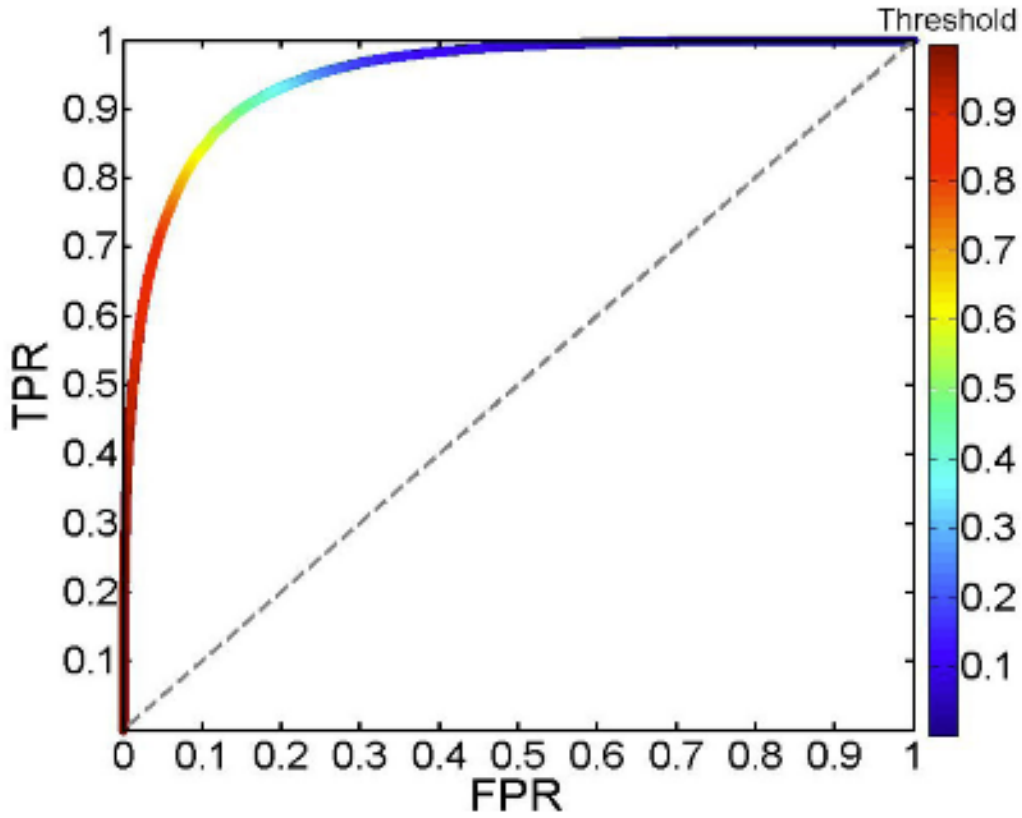
"Fraction of positive examples classified correctly"

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

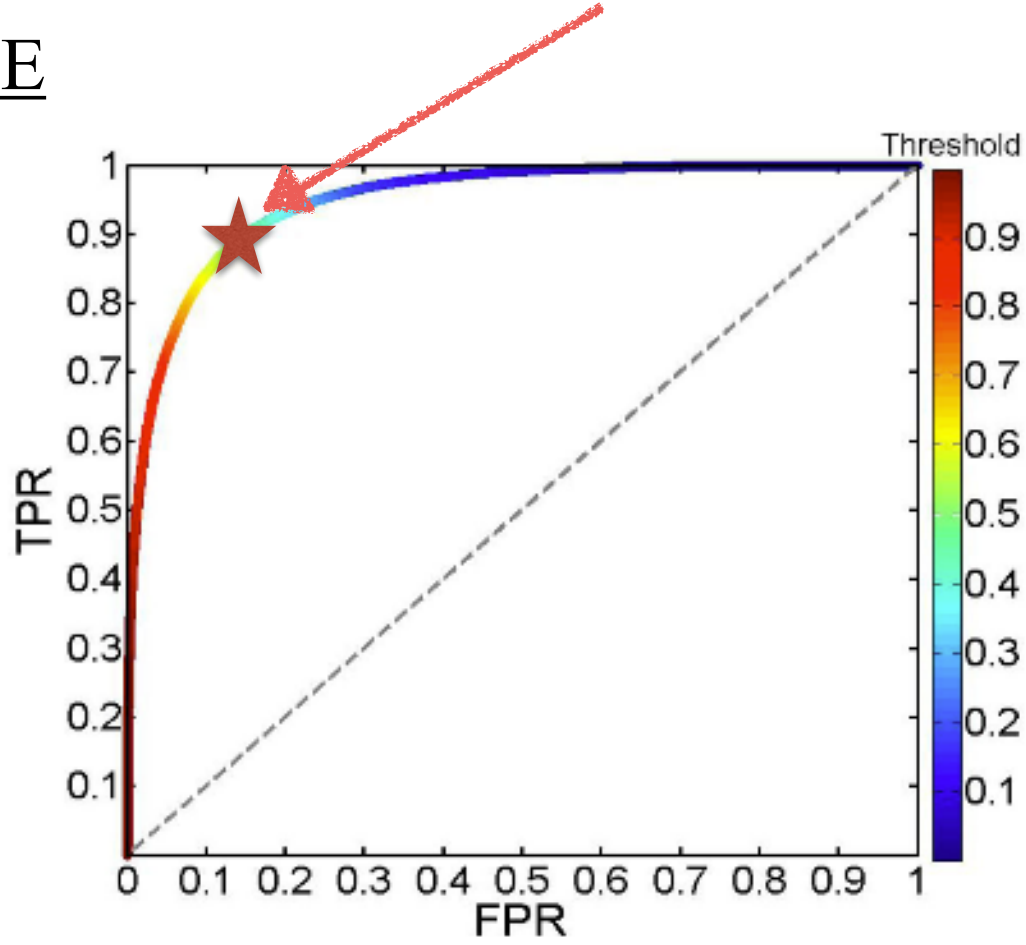


Teimiroonia+16

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 FOR FOR A GIVEN THRESHOLD

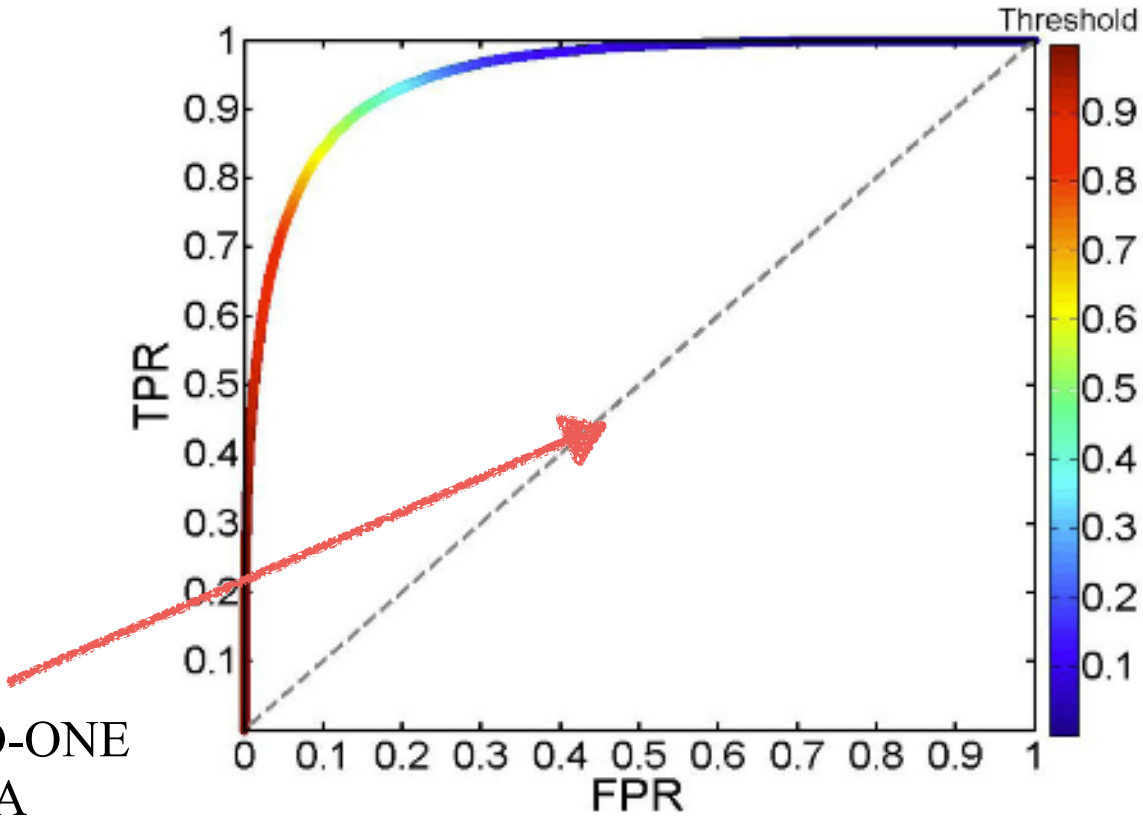
ROC CURVE



Teimiroonia+16

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

ROC CURVE



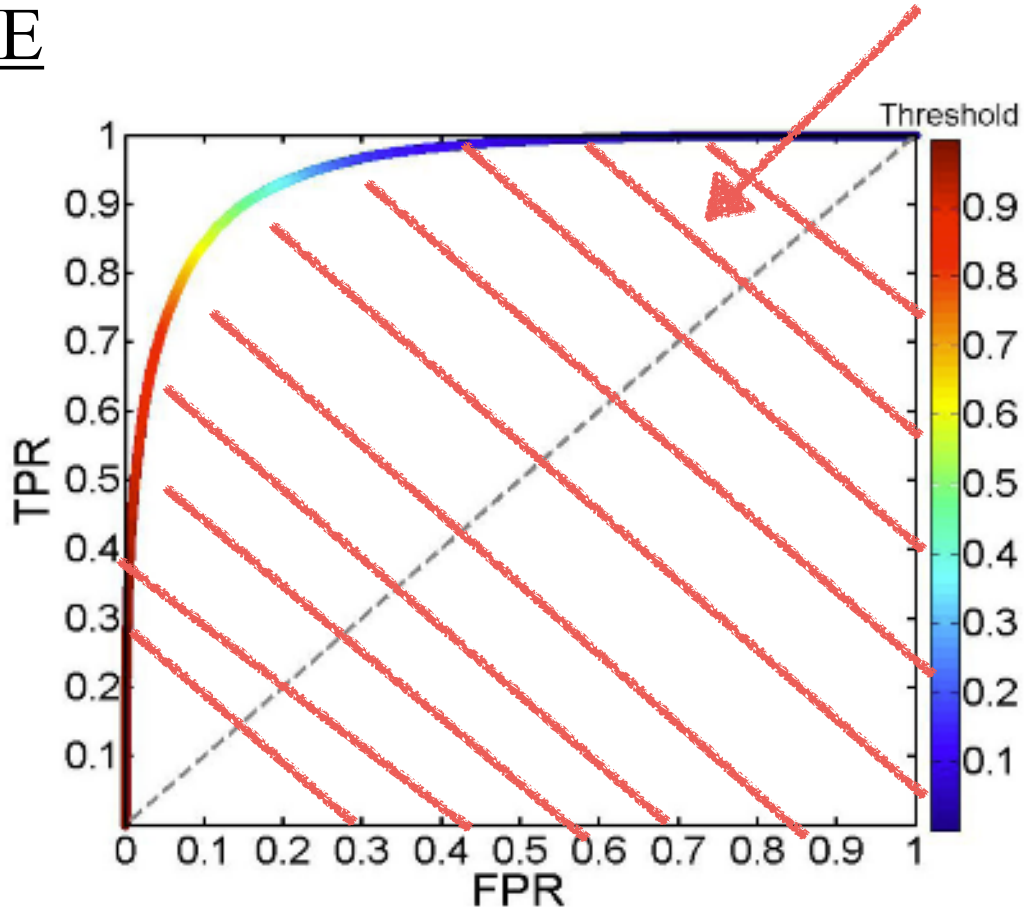
THE ONE-TO-ONE LINE IS A RANDOM CLASSIFICATION

Teimiroonia+16

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

THE AREA UNDER THE CURVE **[AUC]** ALSO MEASURES THE GLOBAL ACCURACY

ROC CURVE



Teimiroonia+16

Evaluation of results

THE P-R CURVE (Precision - Recall)

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

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

Evaluation of results

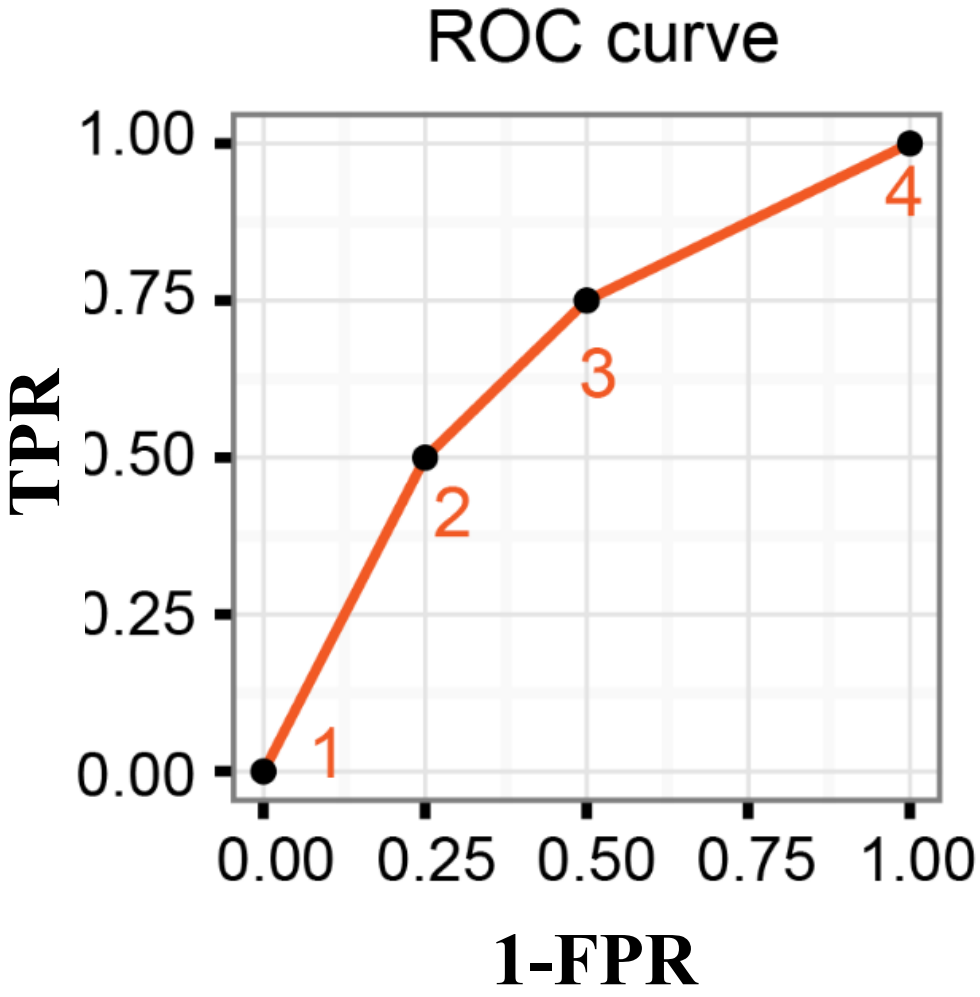
THE P-R CURVE (Precision - Recall)

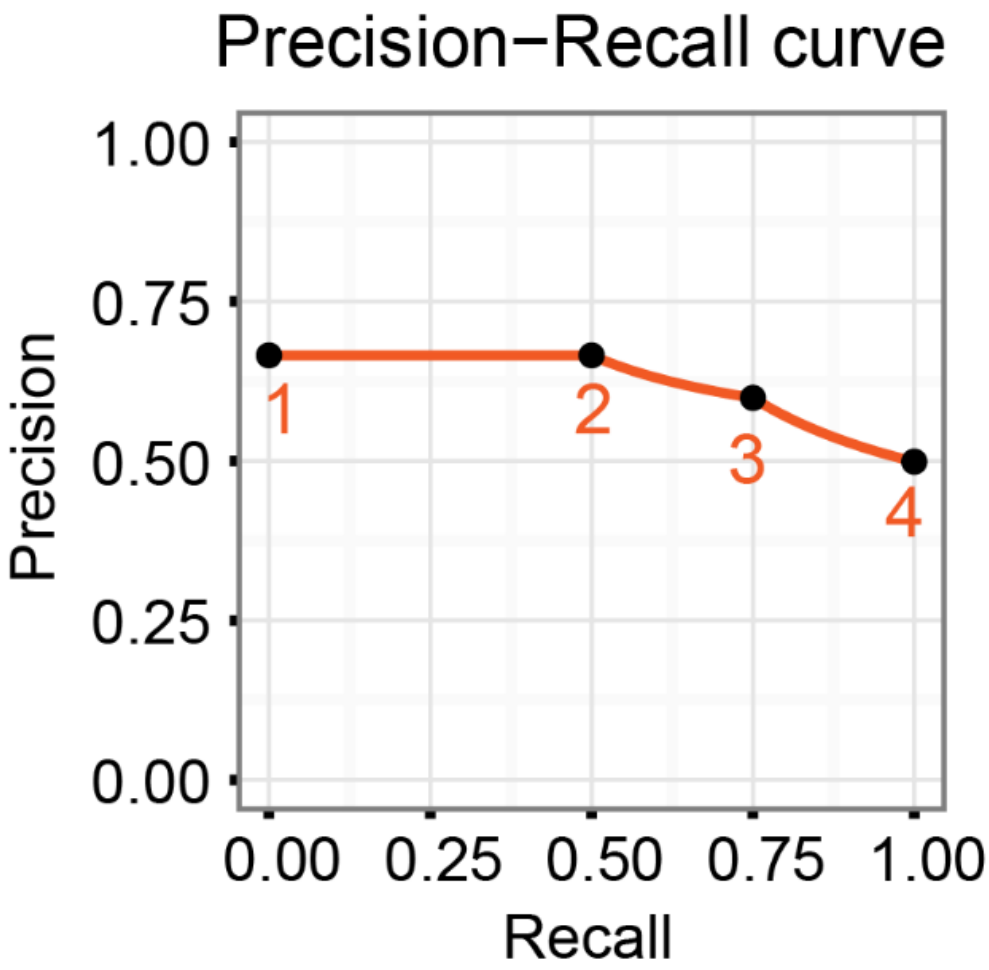
$$Recall = \frac{TP}{TP + FN} = TPR \qquad \text{[completeness]}$$

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

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

Evaluation of results





SUMMARY OF DIFFERENT ACCURACY TRACERS

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

SOURCE

SUMMARY OF DIFFERENT ACCURACY TRACERS

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

F_1 score = $\frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$

THE F1 SCORE
COMBINES BOTH INFORMATIONS IN ONE VALUE

SOURCE

ALL THESE ARE INCLUDED IN SKLEARN

AND ARE VETY 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.

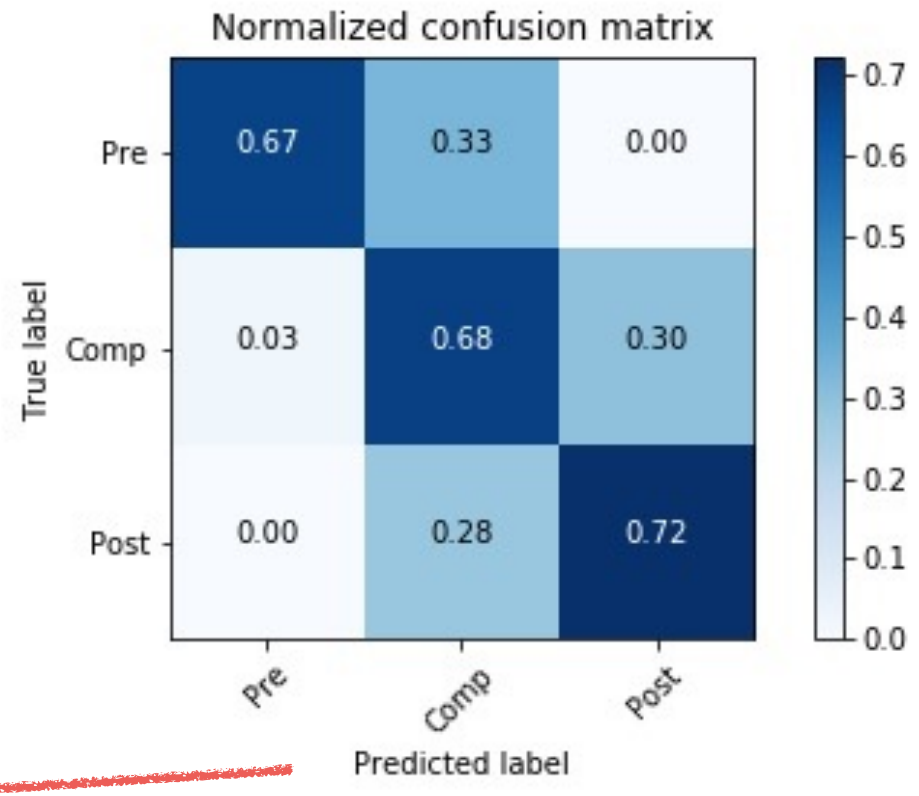
The precision is the ratio `tp / (tp + fp)` where `tp` is the number of true positives and `fp` the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

The recall is the ratio `tp / (tp + 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]

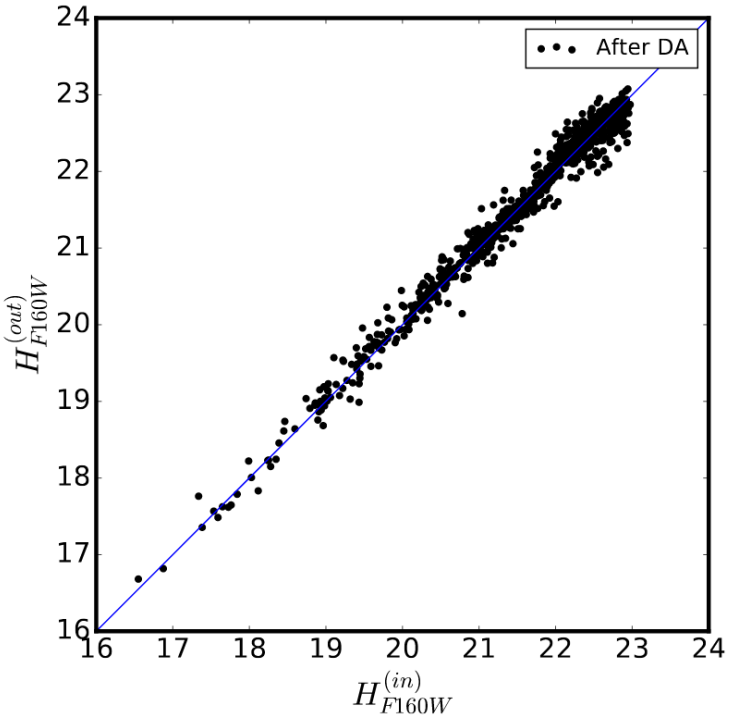
CONFUSION MATRIX



CORRESPONDANCE BETWEEN THE PREDICTED AND TRUE LABEL

Huertas-Company+18

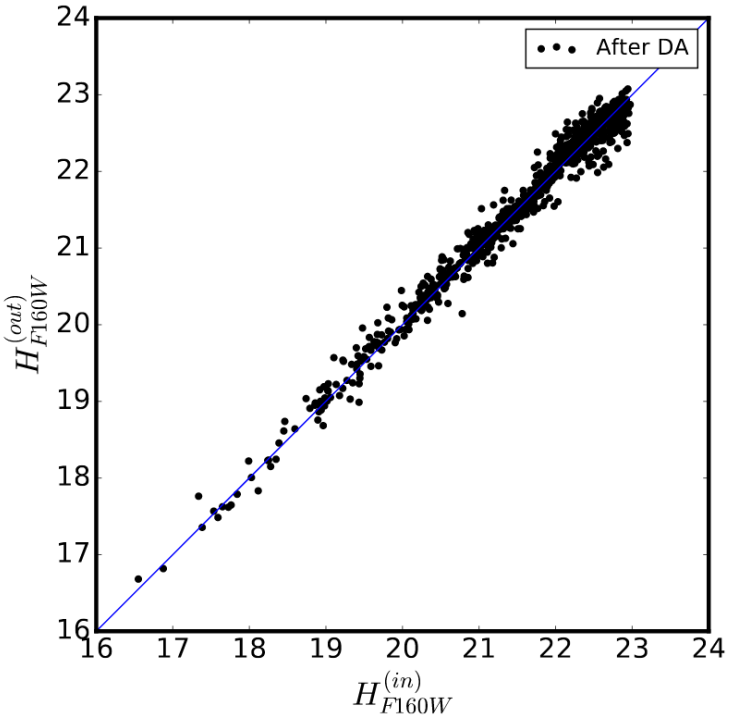
Evaluation of results [regression]



FOR REGRESSIONS, SIMPLY USE
“STANDARD ACCURACY MEASUREMENTS

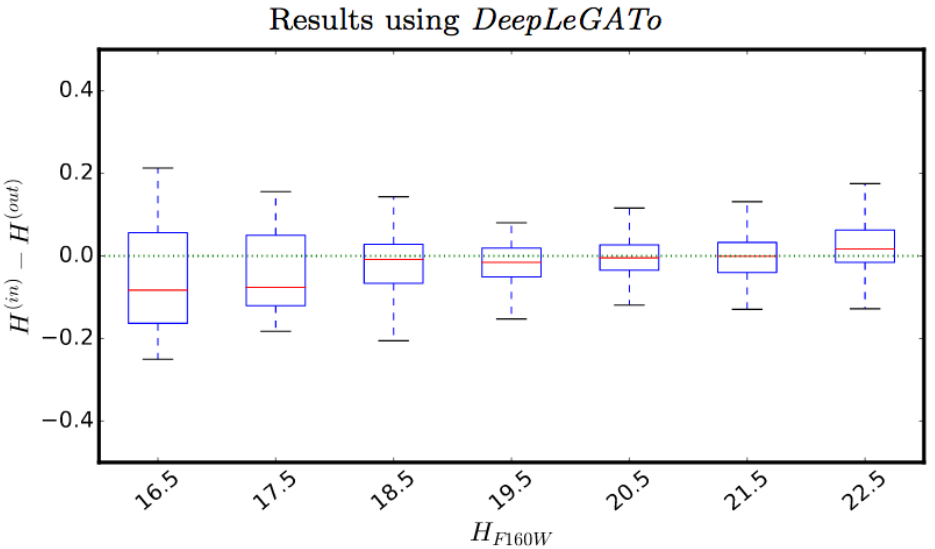
Tuccillo+18

Evaluation of results [regression]



FOR REGRESSIONS, SIMPLY USE
“STANDARD ACCURACY MEASUREMENTS

BIAS, SCATTER …YOU KNOW!



Tuccillo+18