Intelligence artificielle pour l'astrophysique à l'époque du big-data

Marc Huertas-Company

PRACTICAL INFO

HYBRID - ZOOM LINK OPEN

https://rediris.zoom.us/j/99594075110?pwd=Y0dFcTVZWFFqb0xUQm41NWpweWs5dz09

Meeting ID: 995 9407 5110
Password: 842832

GITHUB REPO WITH SLIDES AND TUTORIALS:

https://github.com/mhuertascompany/DL_ED127_2023

Internet Access: eduoram

MORNING ~[9h30-12h30]: THEORY / LECTURES

AFTERNOON ~[14H30-17H30]: HANDS-ON SESSION

PRACTICAL INFO

FULL ONLINE COURSE VIA ZOOM:

https://rediris.zoom.us/j/89315397296?pwd=Y4rX4hZ5y4nw2srCPbQ5sCwvH0q6we.1

Meeting ID: 893 1539 7296
Passcode: 299742

SOME RULES DURING THE ZOOM MEETING:

Keep your Microphones Muted

If need to ask questions (which I recommend!) use the “raise hand” option

PRACTICAL INFO

OTHER AVAILABLE RESSOURCES:

GITHUB REPO WITH SLIDES AND TUTORIALS:

https://github.com/mhuertascompany/DL_ED127_2021

Slack Channel for interactive discussions / questions:

https://join.slack.com/t/iaed127/shared_invite/zt-f5sohajk-uP00KQtWdaPIC_Znw6H2EA

wonder.me space for hands-on:

https://www.wonder.me/r?id=33135bca-0b6f-4a19-9041-bbdb9aa36ed2

SOME PRELIMINARY NOTES

I AM NOT A MACHINE LEARNING RESEARCHER

ONLY AN ASTRONOMER WHO HAS BEEN USING MACHINE LEARNING FOR THE LAST ~20 YEARS FOR MY RESEARCH

THESE LECTURES ARE INTENDED TO PROVIDE A **GLOBAL** UNDERSTANDING OF HOW AI TECHNIQUES WORK AND ESPECIALLY **HOW TO USE THEM FOR YOUR RESEARCH**

What is your previous experience with Machine Learning?

No experience at all

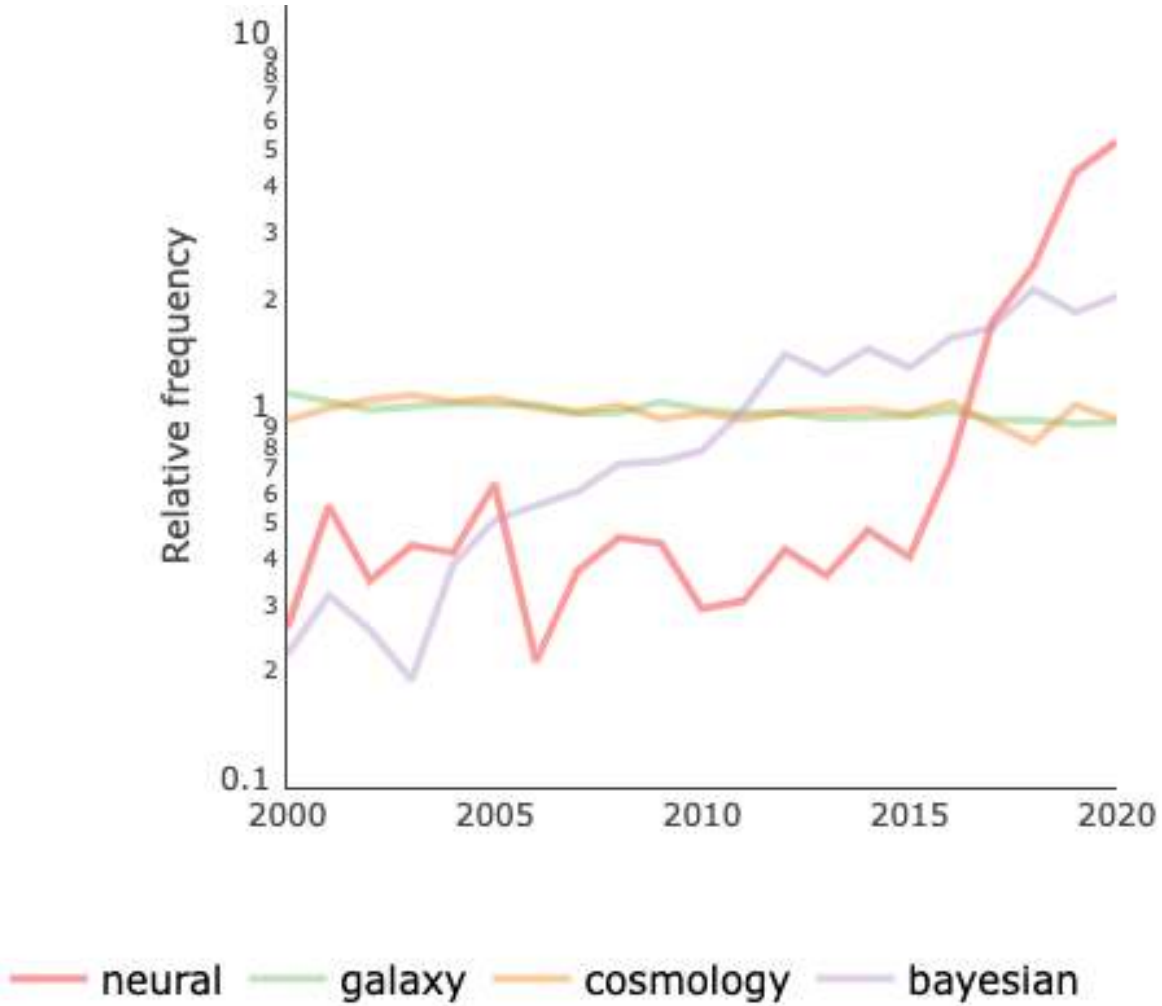
Beginner

Intermediate

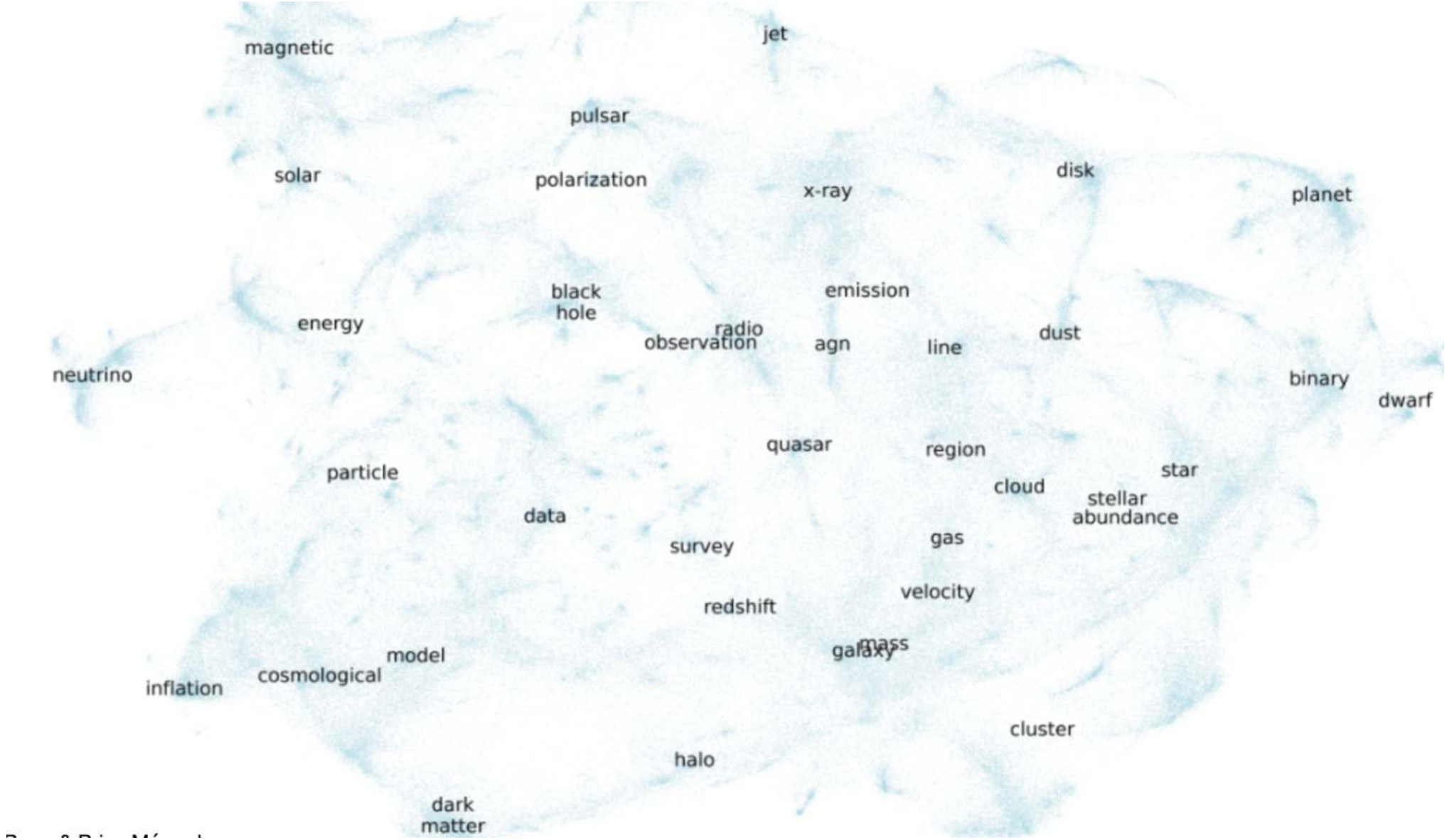
Advanced

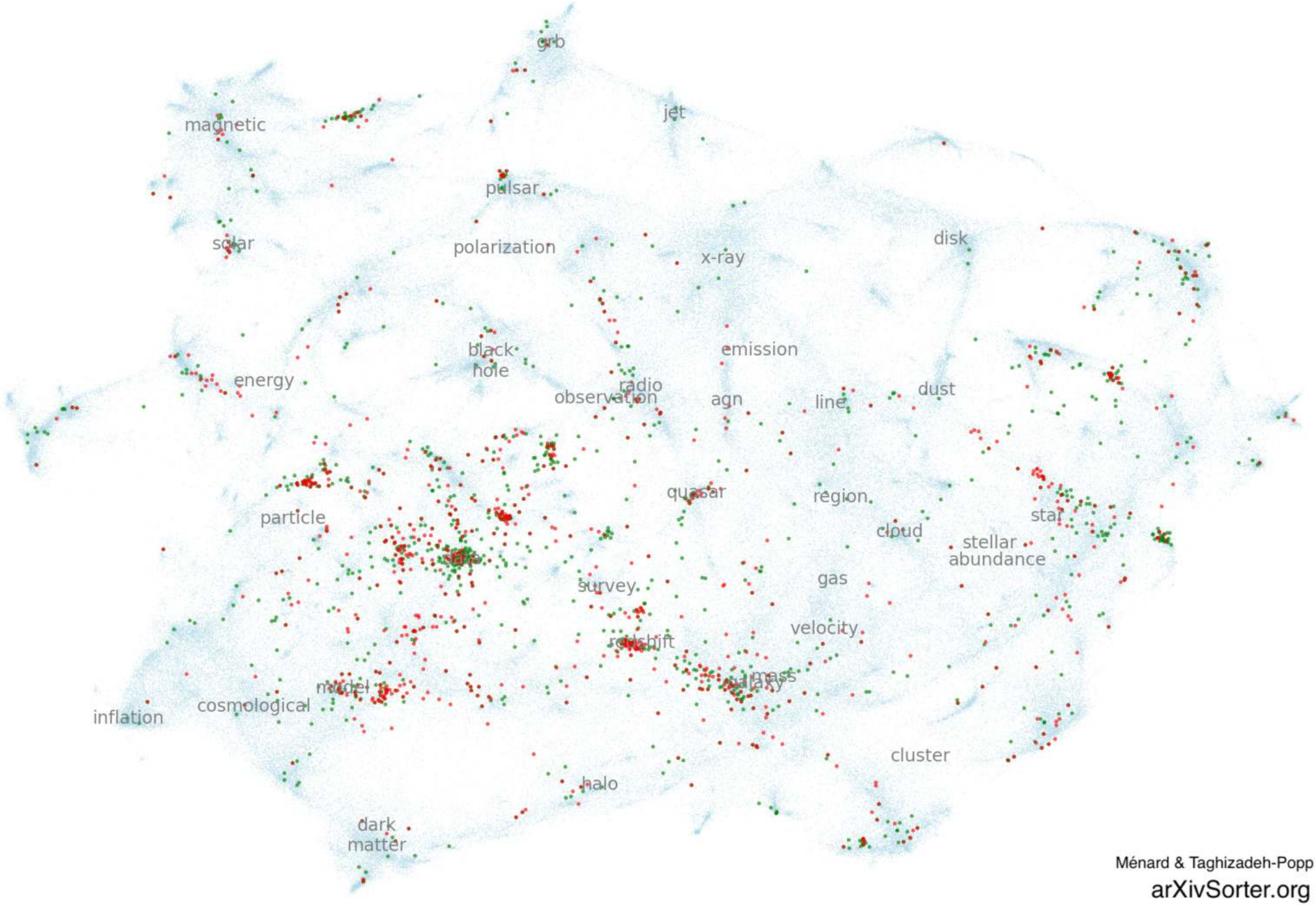
None of the above

Relative change of the number of papers on arXiv/astro-ph



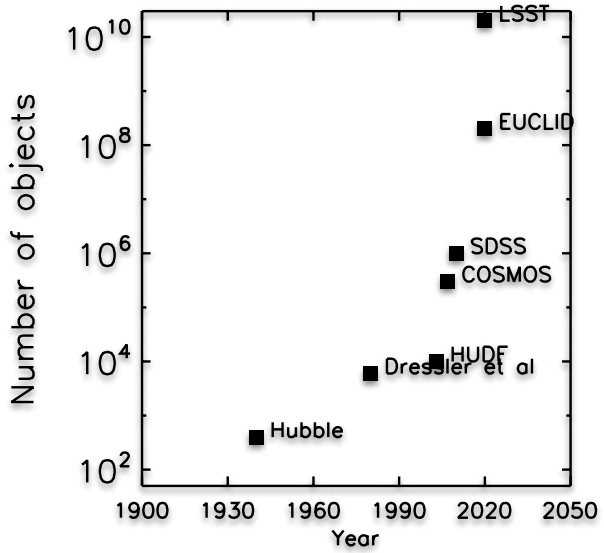
Huertas-Company+23




grb
jet
magnetic
pulsar
solar
polarization
x-ray
disk
black
hole
emission
energy
radio
observation
agn
line
dust
quasar
region
star
particle
cloud
stellar
abundance
gas
survey
velocity
redshift
mass
galaxy
model
cosmological
inflation
cluster
halo
dark
matter
Ménard & Taghizadeh-Popp
arXivSorter.org

WHY DO WE NEED THESE TOOLS IN ASTRONOMY?

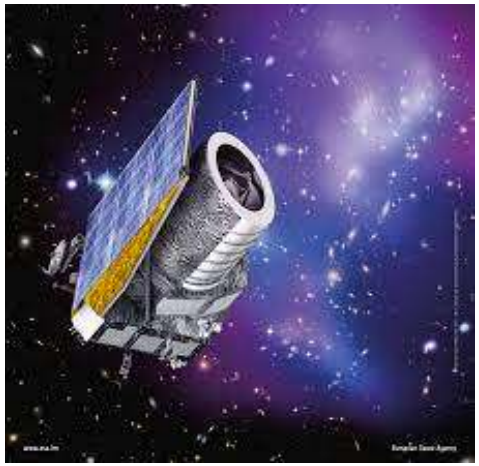
AS IN MANY OTHER DISCIPLINES THE BIG-DATA REVOLUTION HAS ARRIVED TO ASTRONOMY TOO



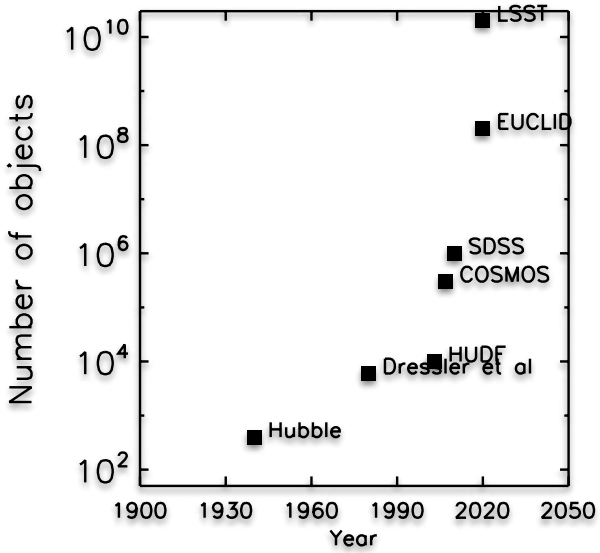
← BIG-DATA REVOLUTION

← we are here

EXTREMELY LARGE IMAGING SURVEYS DELIVERING BILLIONS OF OBJECTS IN 2-5 YEARS



LSST simulation



← BIG-DATA REVOLUTION

← we are here

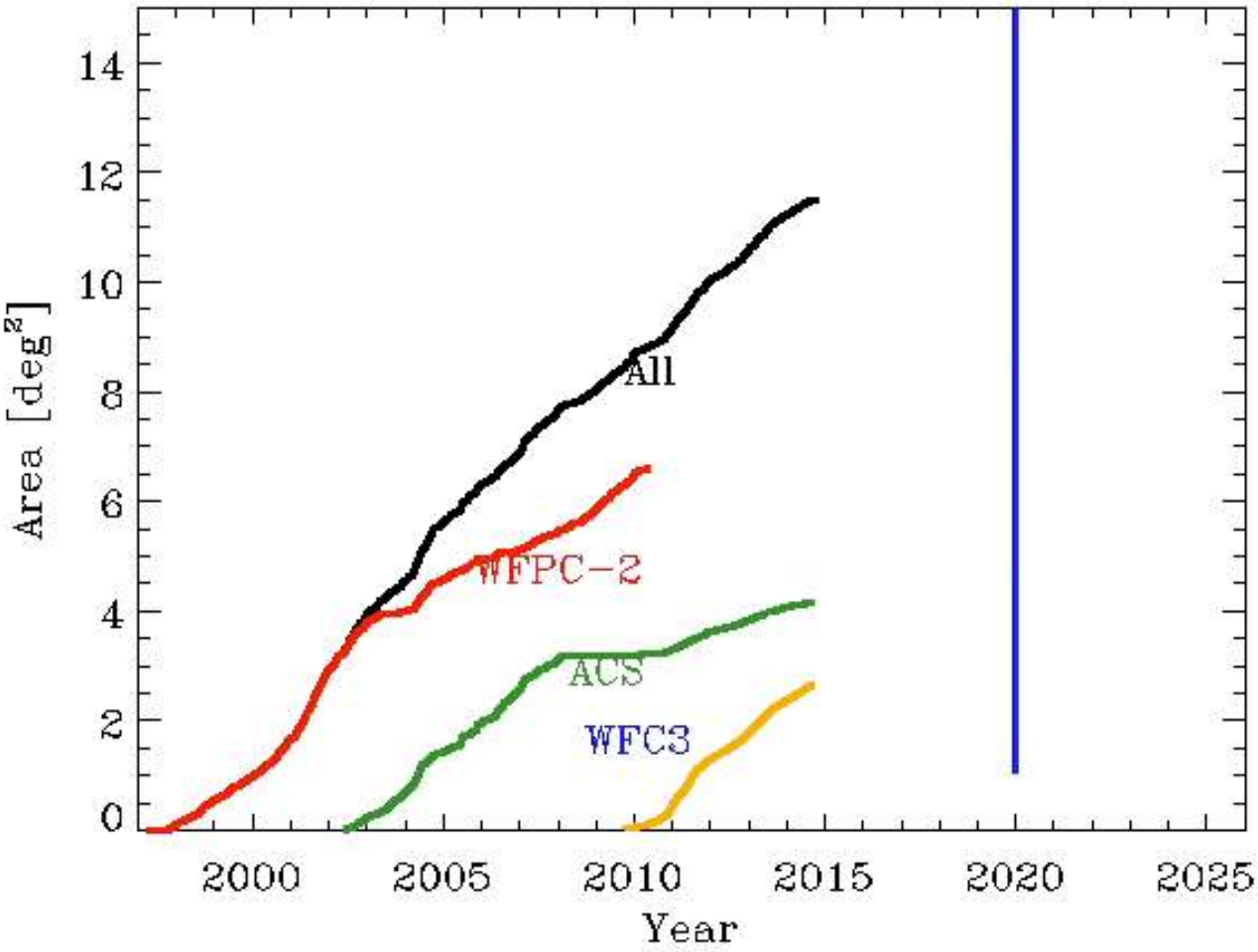
EXTREMELY LARGE IMAGING SURVEYS DELIVERING BILLIONS OF OBJECTS IN 2-5 YEARS

LSST Large Synoptic Survey Telescope

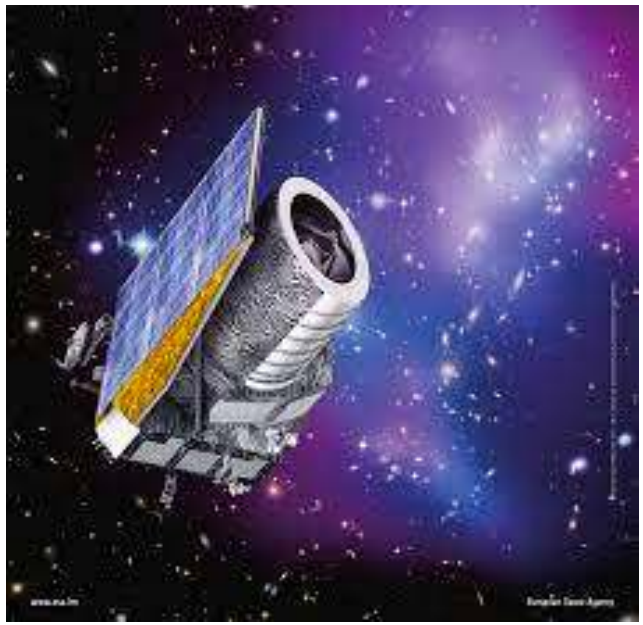
euclid consortium

WIDE-FIELD INFRARED SURVEY TELESCOPE
WFIRST
ASTROPHYSICS • DARK ENERGY • EXOPLANETS

LSST simulation

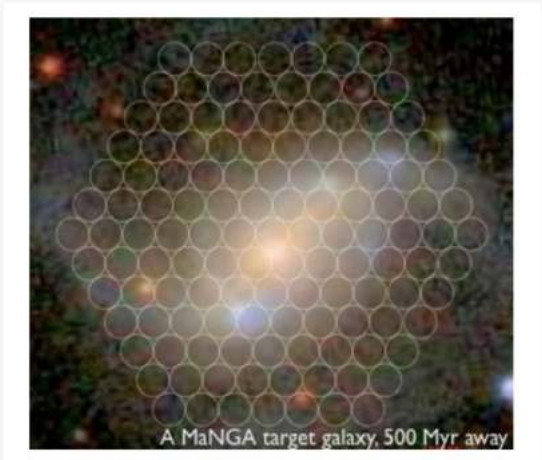


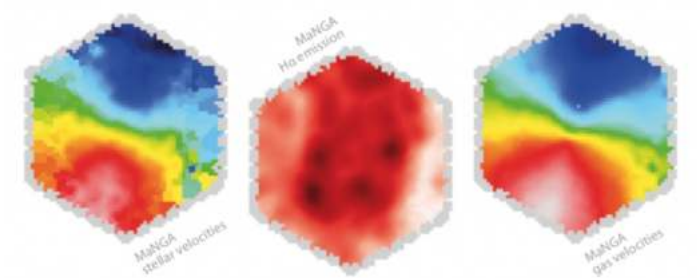
Launch July 2023!



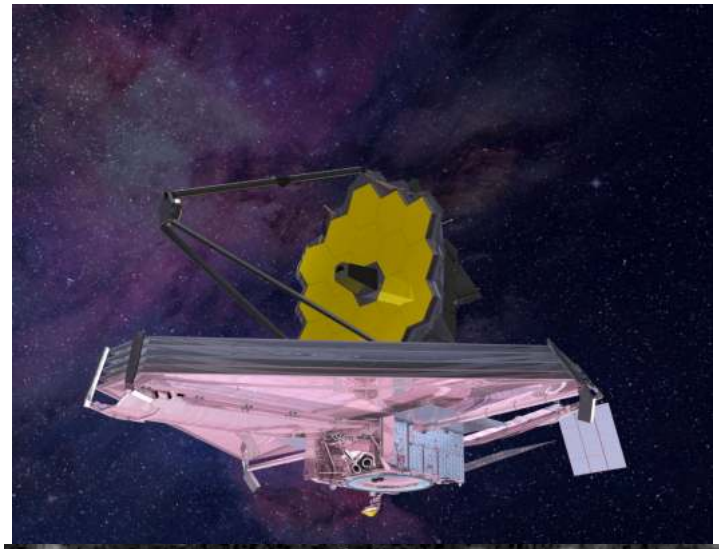
(Thanks to J. Brinchmann)

NOT ONLY VOLUME: AN INCREASING COMPLEXITY OF DATA

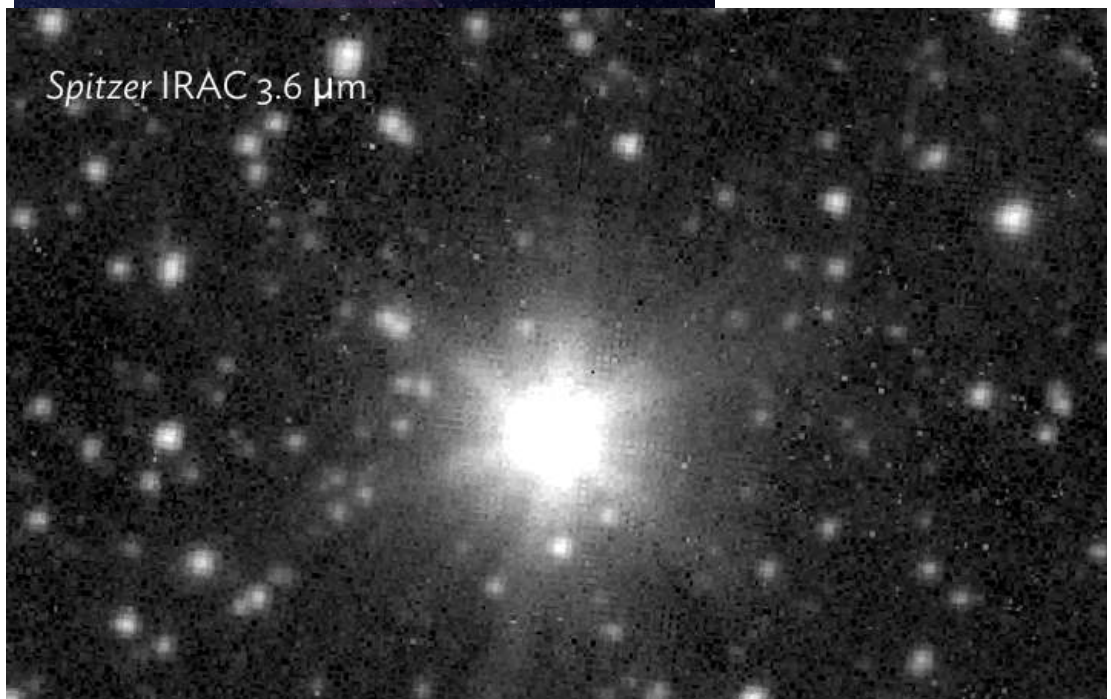




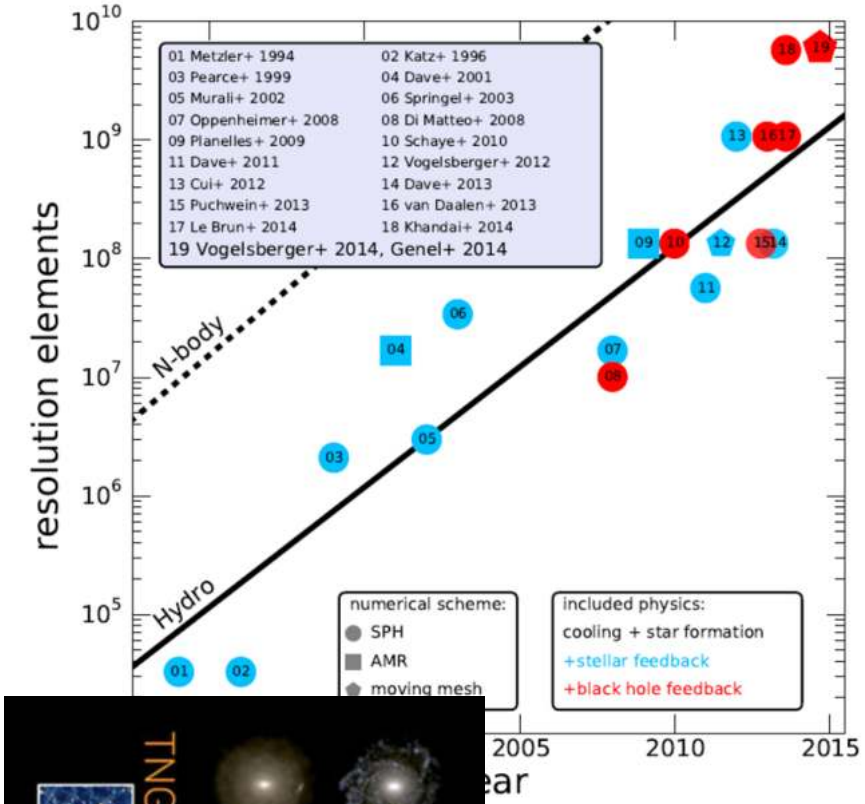
MANGA Survey

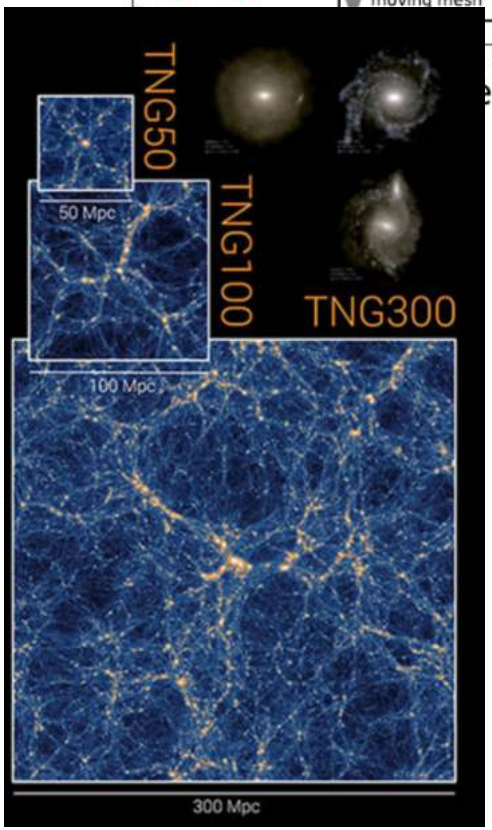


JWST

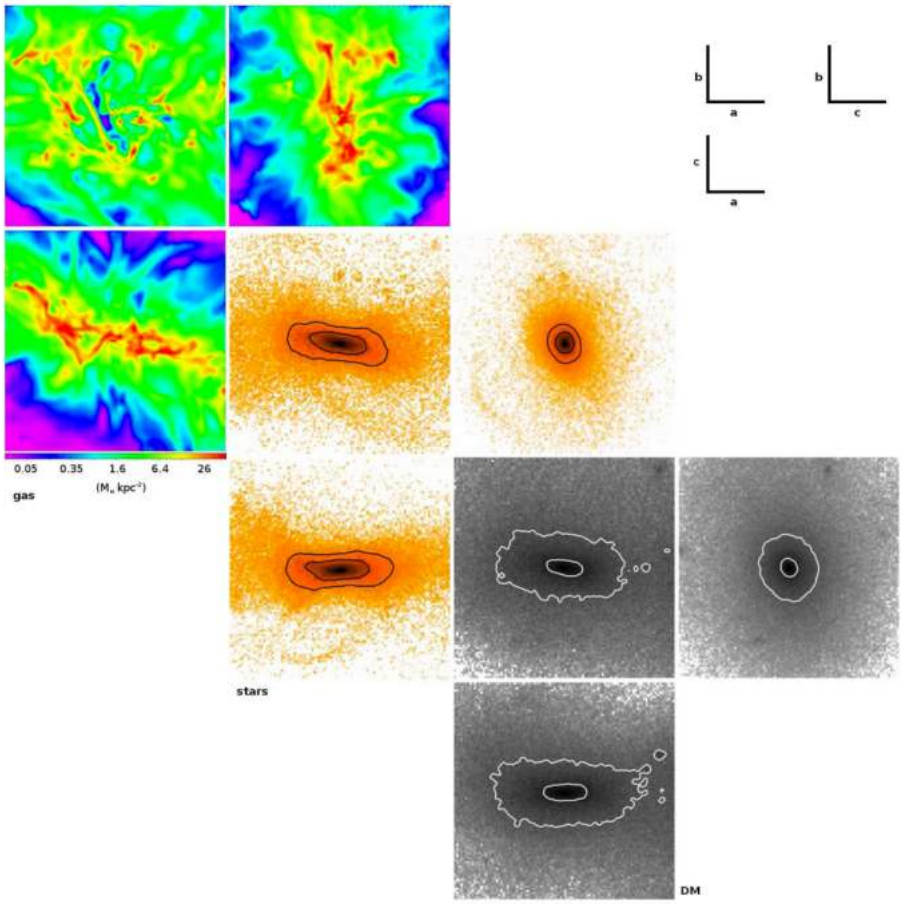


Genel+14





AND ALSO SIMULATIONS!



Ceverino+15

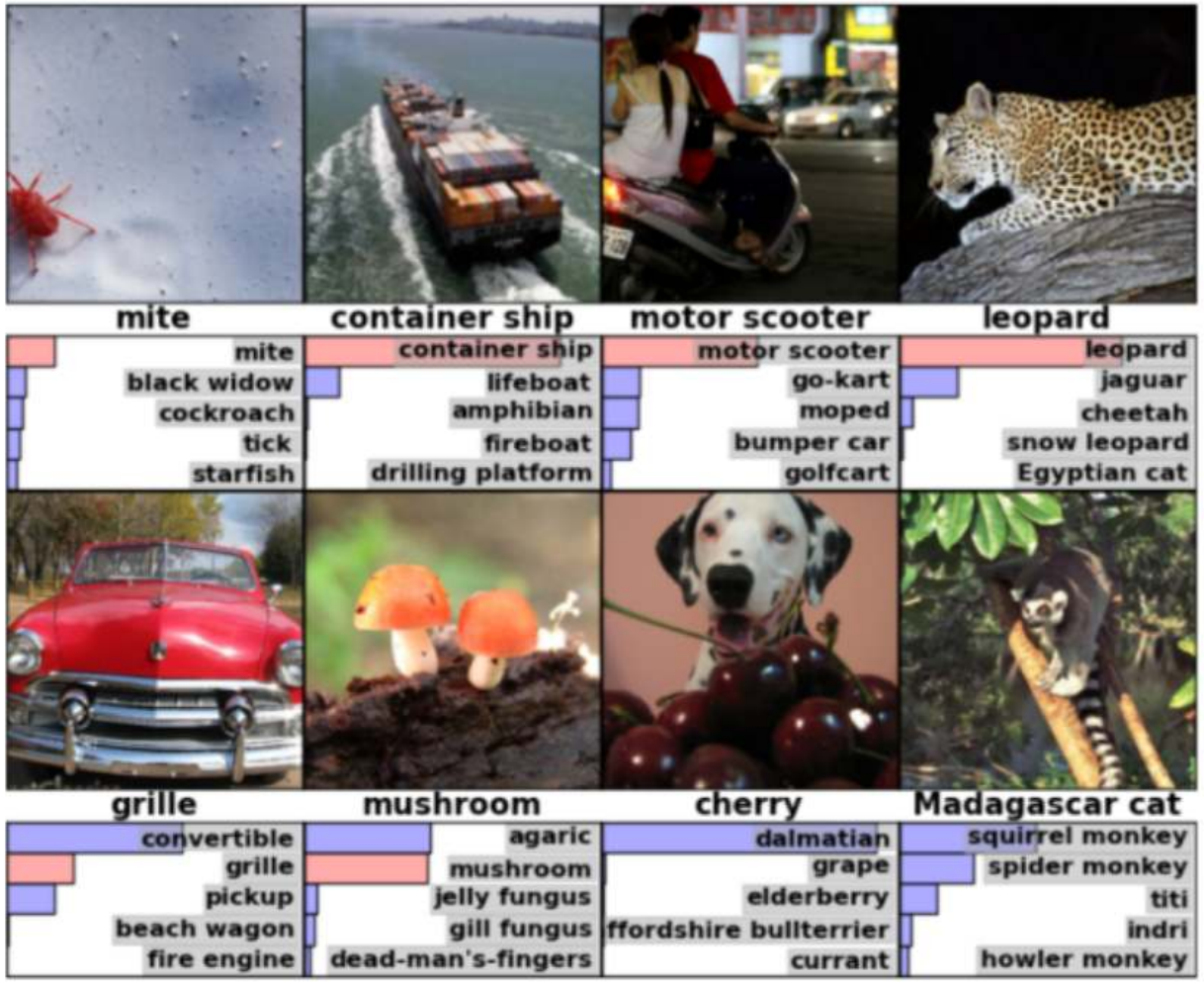
BEFORE 2012….

CAT?

DOG?

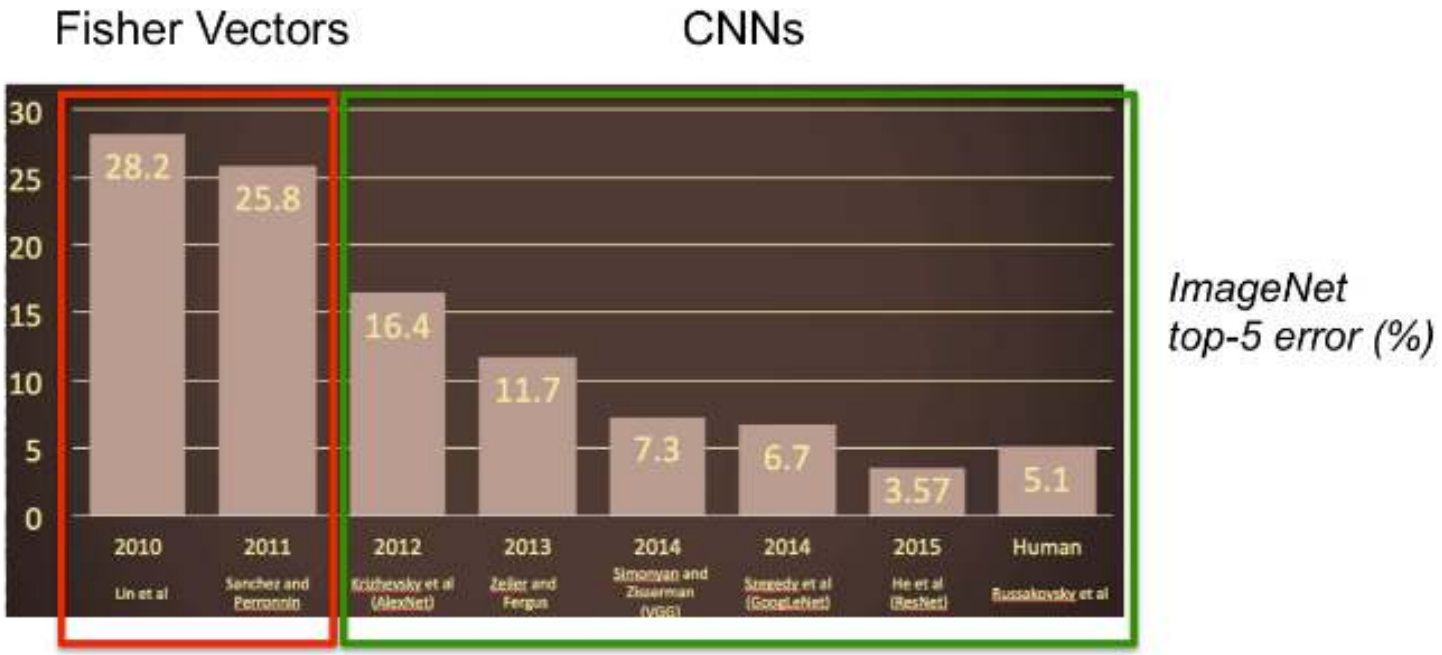
TRIVIAL HUMAN TASKS REMAINED CHALLENGING FOR COMPUTERS

AFTER 2012



IT HAS BECOME TRIVIAL….

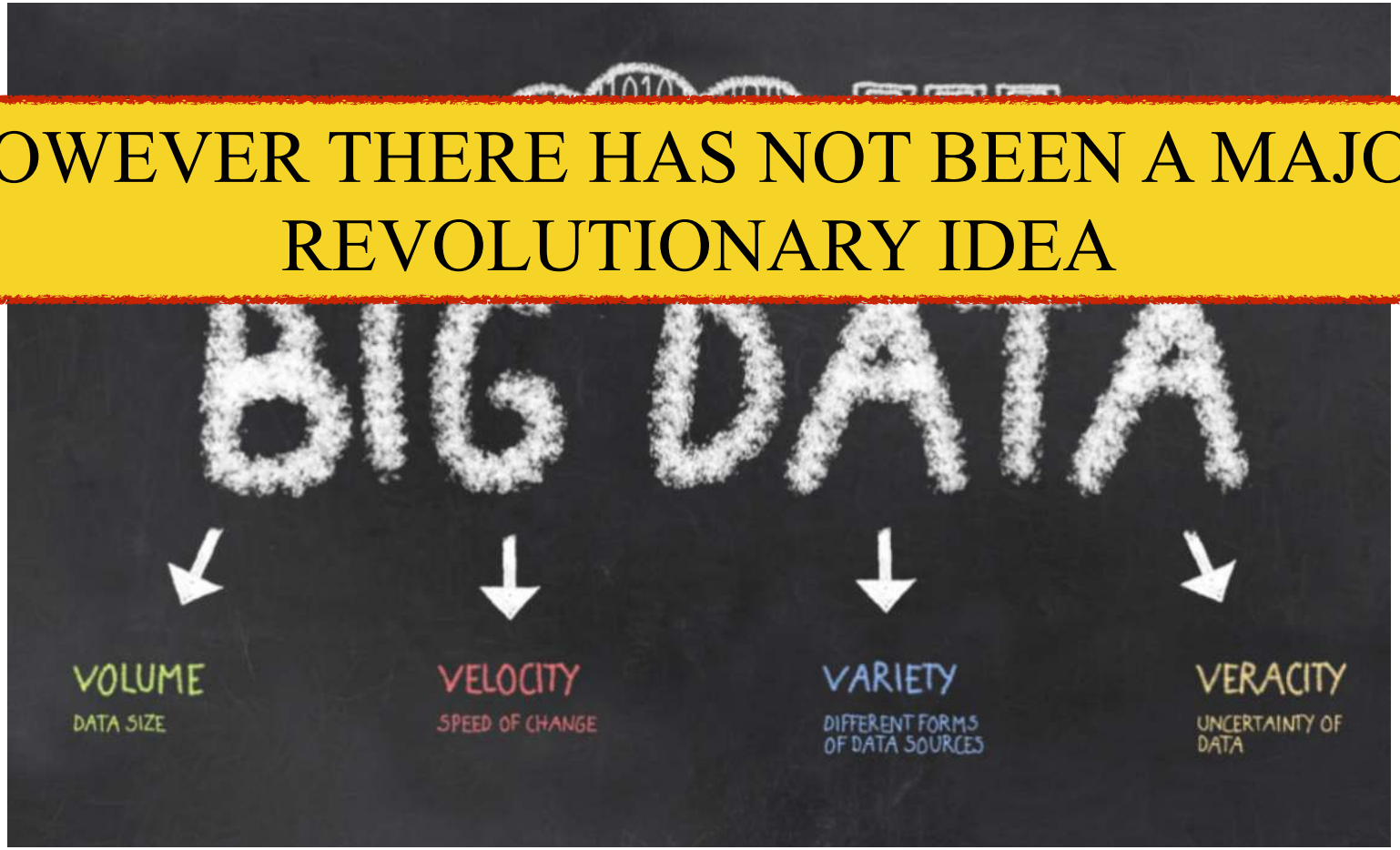
THIS IS A CHANGE OF PARADIGM!



ONE OF THE MAIN REASONS OF THIS BREAKTHROUGH IS THE AVAILABILITY OF VERY LARGE DATASETS TO LEARN

COMBINED WITH THE TECHNOLOGY TO PROCESS ALL THIS DATA

ONE OF THE MAIN REASONS OF THIS BREAKTHROUGH IS THE AVAILABILITY OF VERY LARGE DATASETS TO LEARN



HOWEVER THERE HAS NOT BEEN A MAJOR REVOLUTIONARY IDEA

WHAT IS THESE SERIES OF LECTURES ABOUT?

BASICS OF CLASSICAL MACHINE LEARNING

BASICS OF DEEP LEARNING
(BOTH SUPERVISED AND UNSUPERVISED)

PRACTICAL EXAMPLES IN ASTROPHYSICS WITH TENSORFLOW [with an extragalactic bias]

HOPING THAT THIS WOULD BE USEFUL FOR YOUR RESEARCH!

Have you ever used ML for your work?

Yes

No

I don't know

Planning to

PROGRAM FOR THE WEEK

- PART I: A VERY QUICK INTRODUCTION TO 'CLASSICAL' SUPERVISED MACHINE LEARNING
  - DEFINITION OF SUPERVISED
  - GENERAL STEPS TO "TRAIN A MACHINE LEARNING MODEL"
  - "CLASSICAL" ALGORITHMS EXCLUDING NEURAL NETWORKS: RFs, KERNEL MACHINES

PROGRAM FOR THE WEEK

- PART II: FOUNDATIONS OF ARTIFICIAL NEURAL NETWORKS
  - PERECPTRON, NEURON DEFINITION
  - LAYER OF NEURONS, HIDDEN LAYERS
  - ACTIVATION FUNCTIONS
  - OPTIMIZATION [GRADIENT DESCENT, LEARNING RATES]
  - BACKPROPAGATION
  - LOSS FUNCTIONS

PROGRAM FOR THE WEEK

- PART III: DEEP LEARNING FOR COMPUTER VISION
  - CONVOLUTIONS AS NEURONS
  - CNNs [POOLING, DROPOUT]
  - VANISHING GRADIENT / BATCH NORMALIZATION
  - CNN VISUALIZATION
  - FCNNs (IMAGE2IMAGE NETWORKS)
  - INSTANCE AND SEMANTIC SEGMENTATION

PROGRAM FOR THE WEEK

- PART IV: DEEP LEARNING FOR SEQUENCE MODELLING
  - RECURSIVE NEURAL NETWORKS
  - BACK PROPAGATION IN TIME
  - SELF ATTENTION
  - TRANSFORMERS

PROGRAM FOR THE WEEK

- PART V: INTRODUCTION TO UNSUPERVISED MACHINE (DEEP) LEARNING
  - CLUSTERING ALGORITHMS
  - SELF-SUPERVISED LEARNING, FOUNDATION MODELS
  - DEEP GENERATIVE MODELS
  - DEEP PROBABILISTIC MODELS

HANDS-ON SESSIONS

WE WILL TRY TO PRACTICALLY IMPLEMENT SOME OF THE NOTIONS LEARNED

TUTORIALS WILL USE **GOOGLE COLAB**. YOU WILL NEED A GOOGLE ACCOUNT AND SOME SPACE IN YOUR GDRIVE. WORKS ALSO BETTER ON CHROME.

NOTEBOOKS ARE GENERALLY SELF EXPLANATORY. YOU WILL BE ASKED TO FILL PIECES OF CODE AT DIFFERENT STAGES.

LET'S TRY TO DISCUSS AS MUCH AS POSSIBLE!

REFERENCES

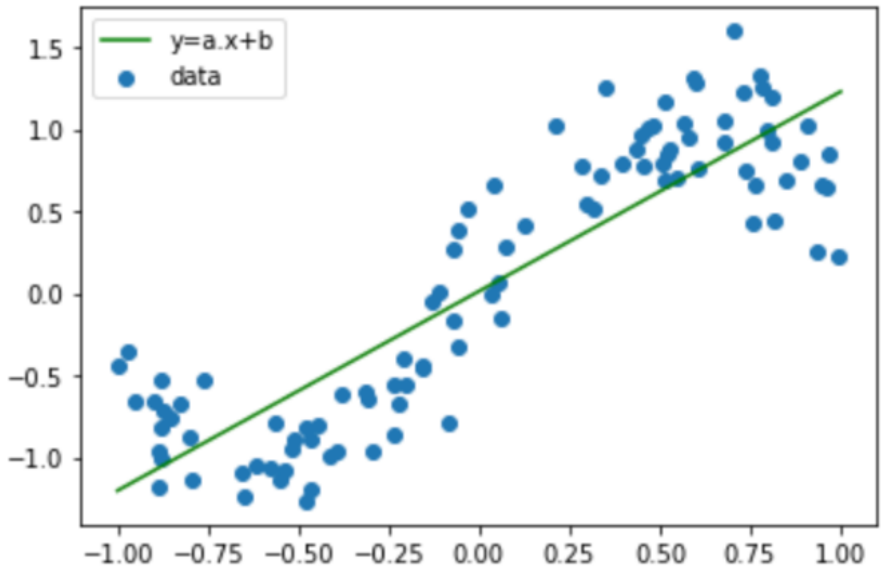
SEVERAL SLIDES / INFOS SHOWN HERE ARE INSPIRED/ TAKEN FROM OTHER WORKS / COURSES FOUND ONLINE

- Deep Learning: Do-It-Yourself! [Bursuc, Krzakala, Lelarge]
- DEEPLEARNING.AI [COURSERA, Ng, Bensouda, Katanforoosh]
- MACHINE LEARNING LECTURES [Keck]
- EPFL DEEP LEARNING COURSE [Fleuret]
- DEEPMIND / UCL Lecture series [DeepMind]
- https://www.deeplearningbook.org/

+ many others!

Thanks to all of them!

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

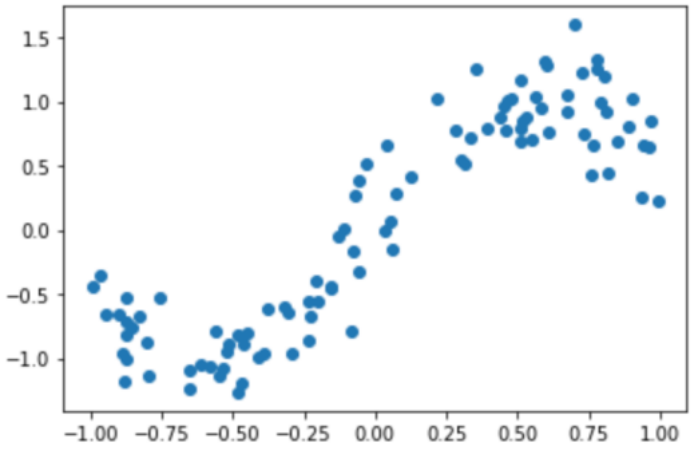
Is this linear fit machine learning?

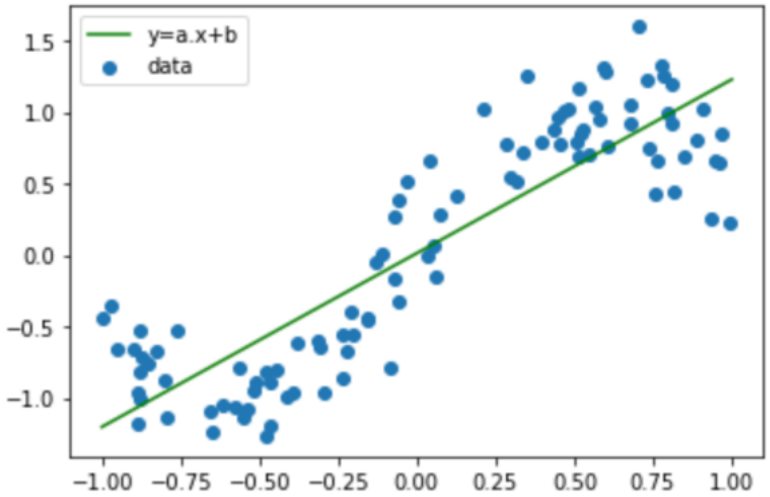
- Yes
- No
- It depends
- I don't know

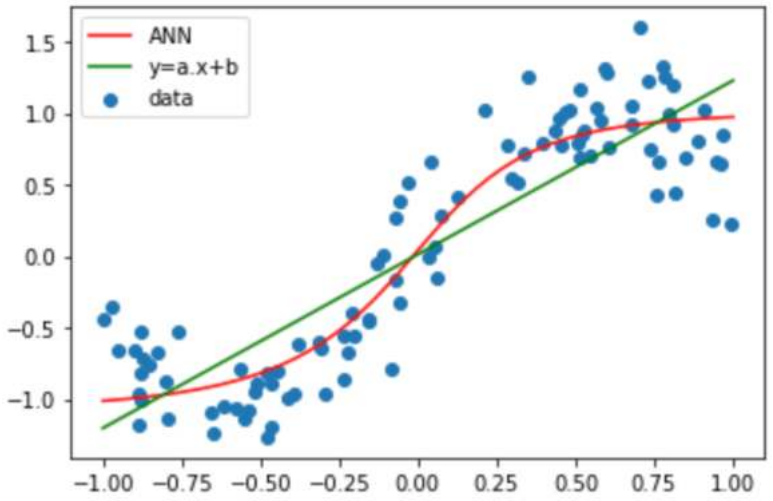
Powered by Poll Everywhere

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

y=F(x,p)?

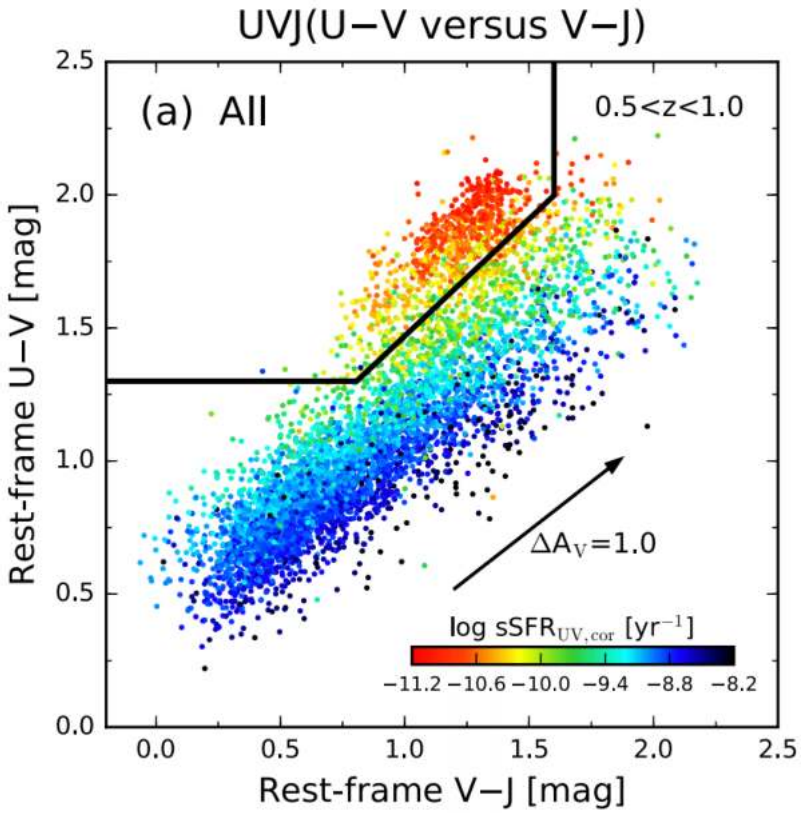






p is derived from physical insight

p is data driven, no physics



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

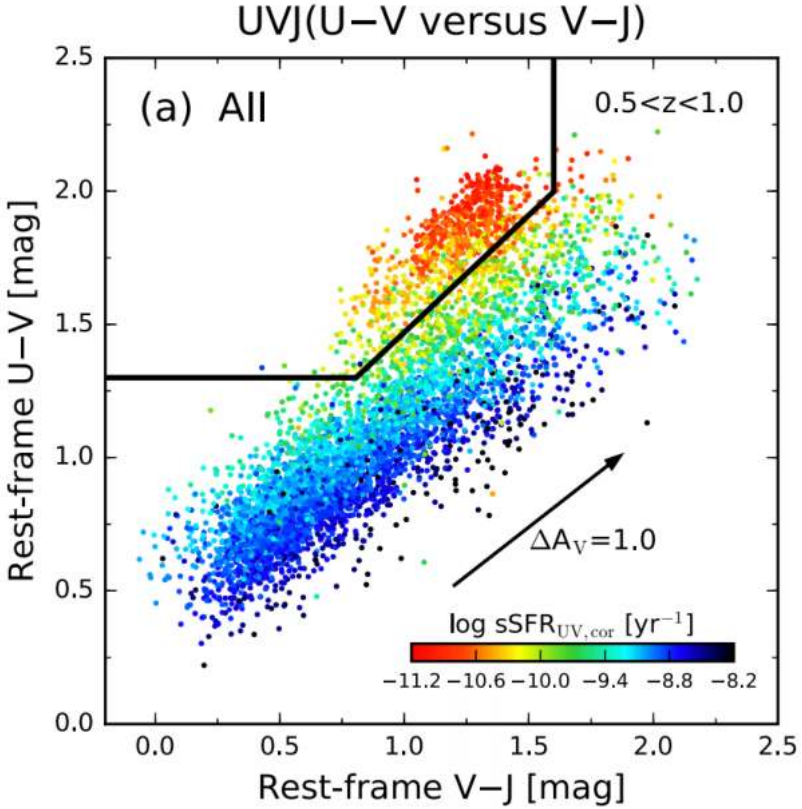
Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

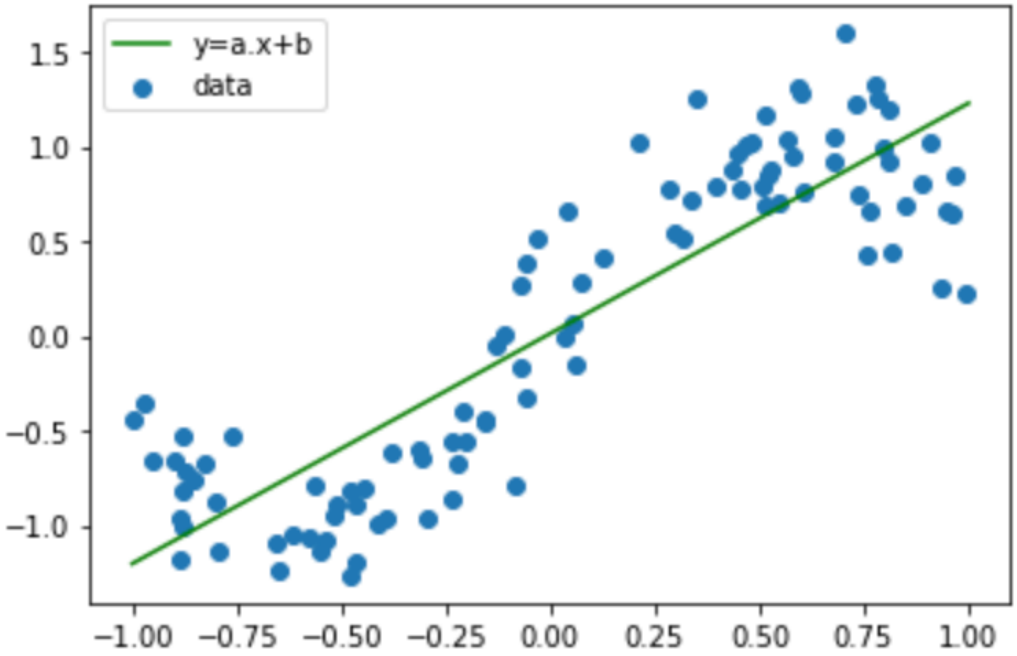
Liu+18

Supervised Machine Learning:

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

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





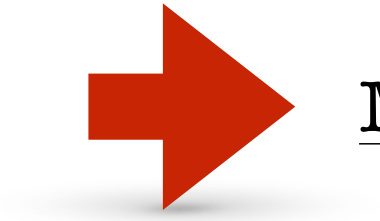
Liu+18

Why data driven?

- No suitable physical model available: **accuracy**
- Physical Model too complex or dataset too large, minimisation difficult: **speed**
- There might be hidden information in the data (beyond usual summary statistics): **discovery**

Why data driven?

- No suitable physical model available: **accuracy**
- Physical Model too complex or dataset too large, minimisation difficult: **speed**
- There might be hidden information in the data (beyond usual summary statistics): **discovery**

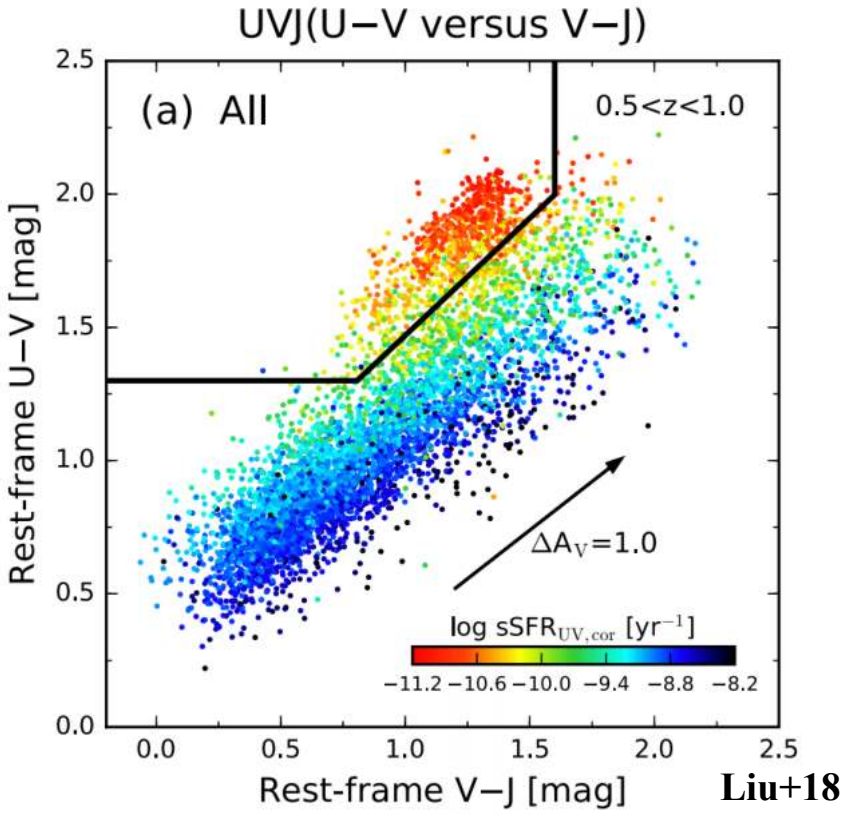


Motivated by large and complex datasets

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

LABEL

Q , SF



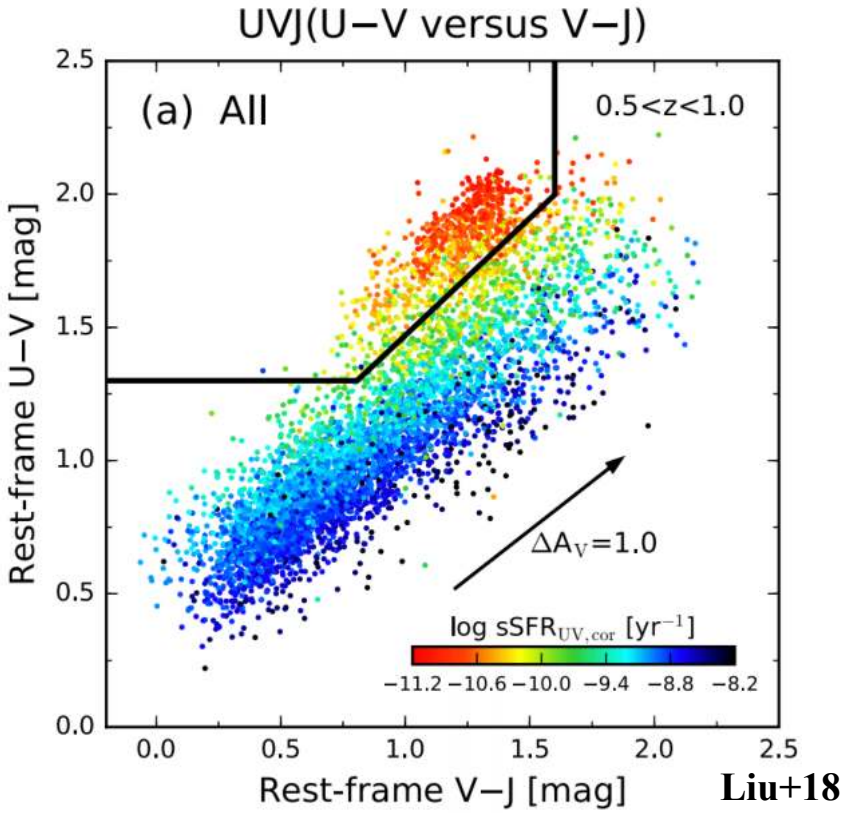
Liu+18

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

LABEL

Q(0) , SF(1)

(U-V, V-J) FEATURES



Liu+18

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

LABEL

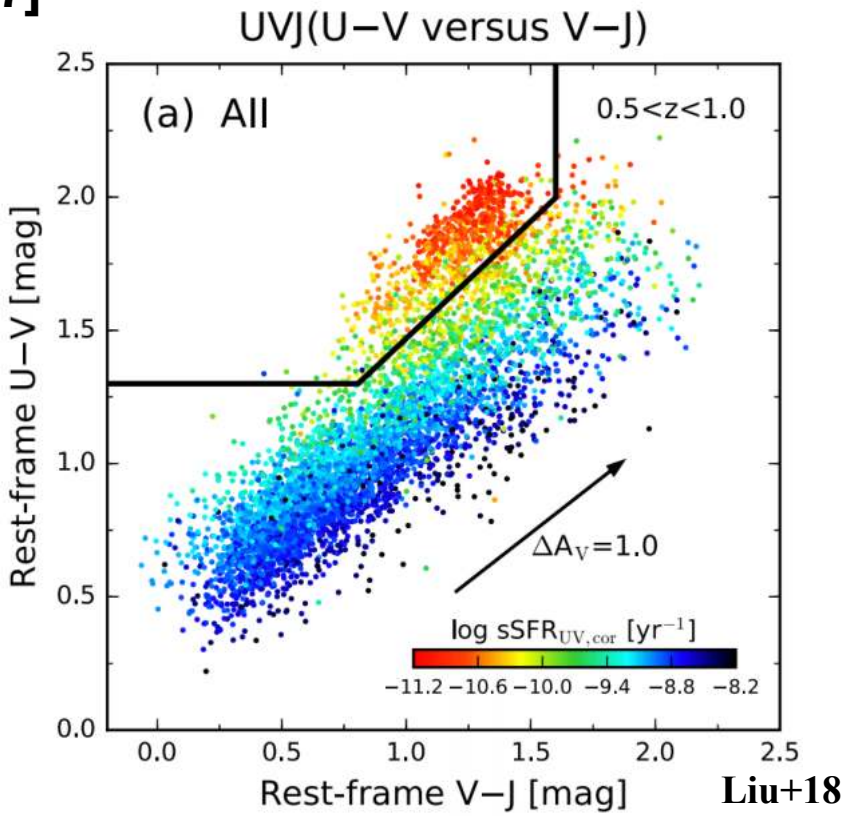
Q(0) , SF(1)

NETWORK FUNCTION

(U-V, V-J) FEATURES

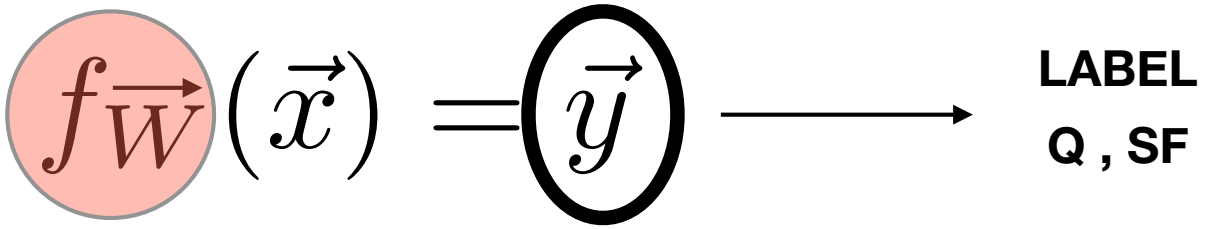
sgn[(u-v)-0.8*(v-j)-0.7]

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

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

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

Features
(colors, fluxes, spectra indices…)

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

Label
(morphology, object type, redshift …)

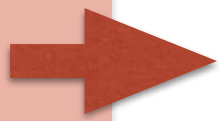
SUPERVISED LEARNING

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

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

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



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

?

SUPERVISED LEARNING

Unlabelled set

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

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

Training set

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

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

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

?

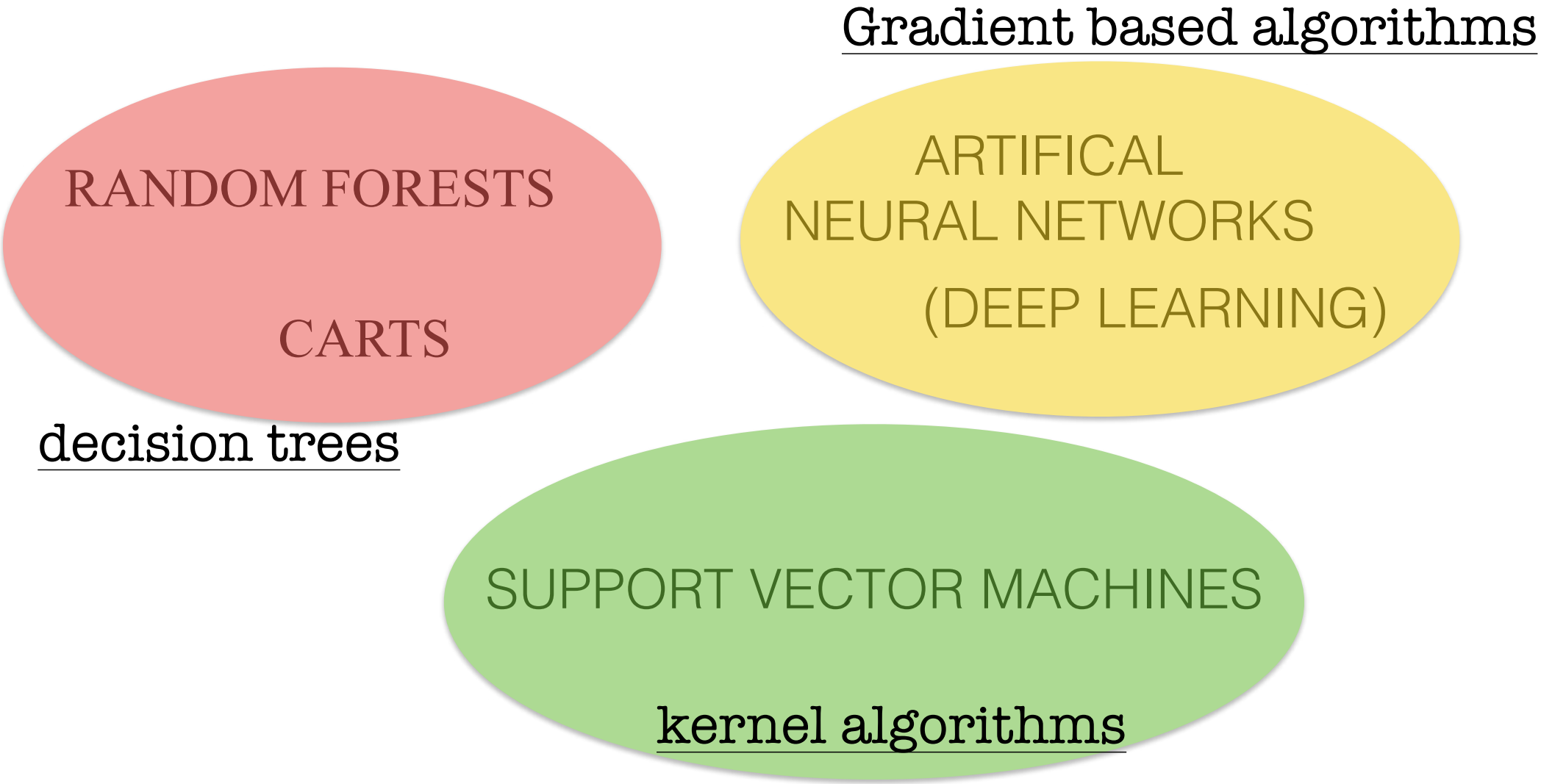
$$(\vec{x_1}, \vec{x_2}, \vec{x_3}, \ldots, \vec{x_n}) \qquad \vec{x} \in \mathbb{R}^d$$

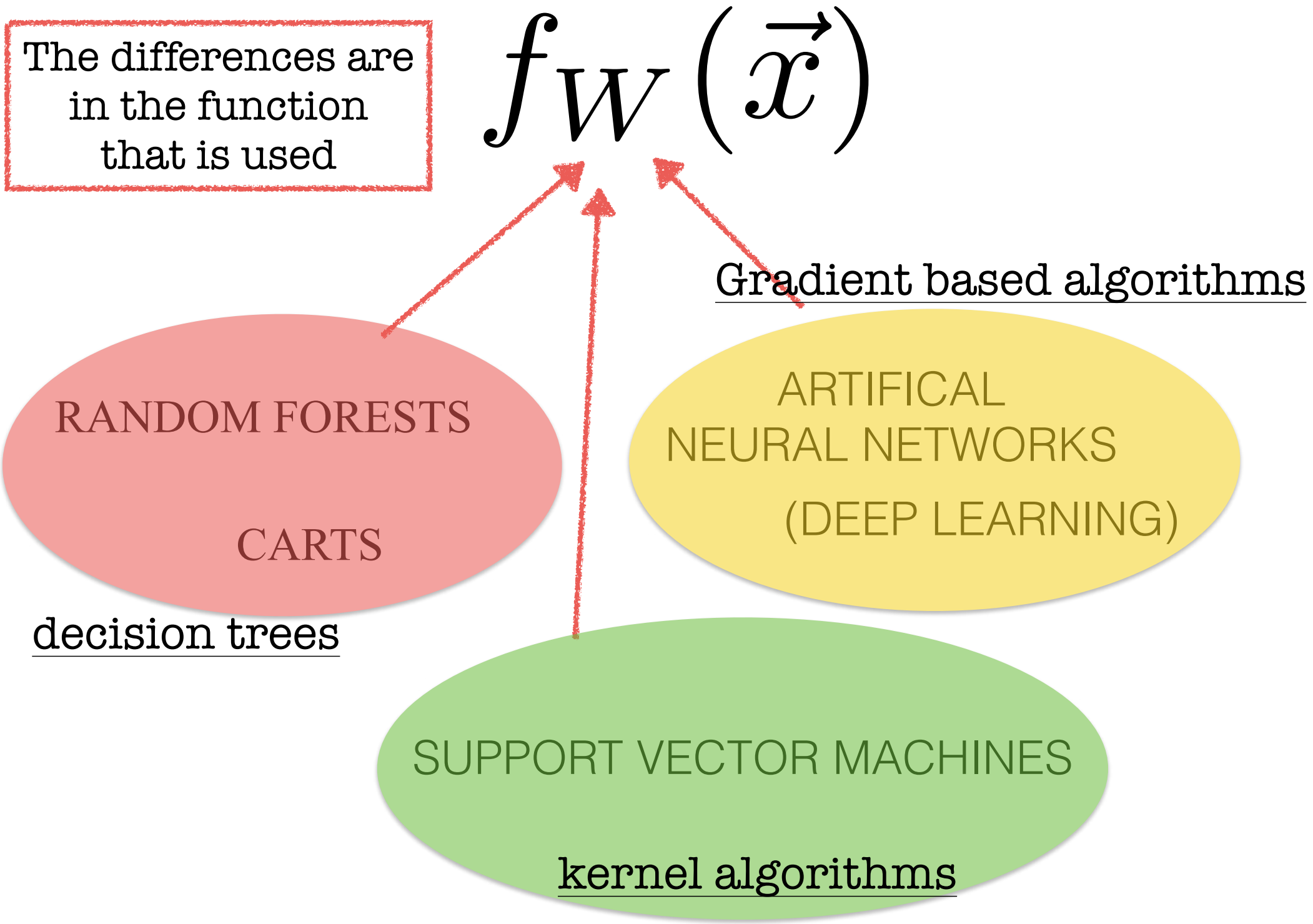
$$(\vec{y_1}, \vec{y_2}, \vec{y_3}, \ldots, \vec{y_n}) \qquad \vec{y} \in \mathbb{R} \qquad \vec{y} \in \mathbb{N}$$

<u>GENERAL GOAL:</u> 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 W such as the prediction error is minimal <u>over all unseen vectors</u>

Different types of supervised machine learning methods




The differences are in the function that is used
$f_W(\vec{x})$
Gradient based algorithms
RANDOM FORESTS
CARTS
decision trees
ARTIFICAL NEURAL NETWORKS
(DEEP LEARNING)
SUPPORT VECTOR MACHINES
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

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

For example: $(F_W(\vec{x_i}) - \vec{y_i})^2$ (MSE LOSS FUNCTION)

2. MINIMISE THE EMPIRICAL RISK WITH OPTIMISATION

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

MINIMISE THE RISK

EMPIRICAL RISK?

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

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

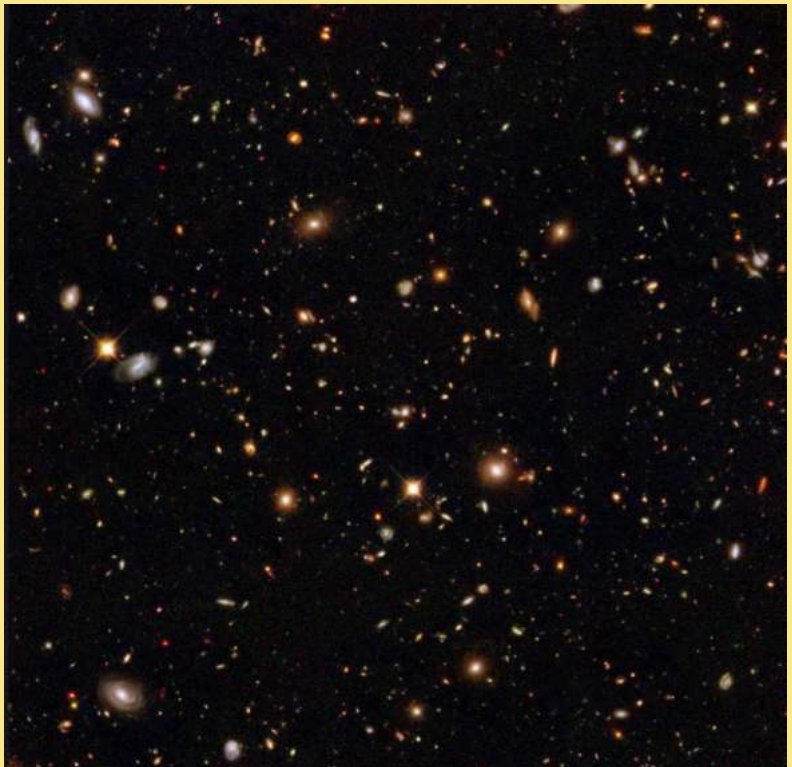
OBSERVED DATASET

EMPIRICAL RISK?

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

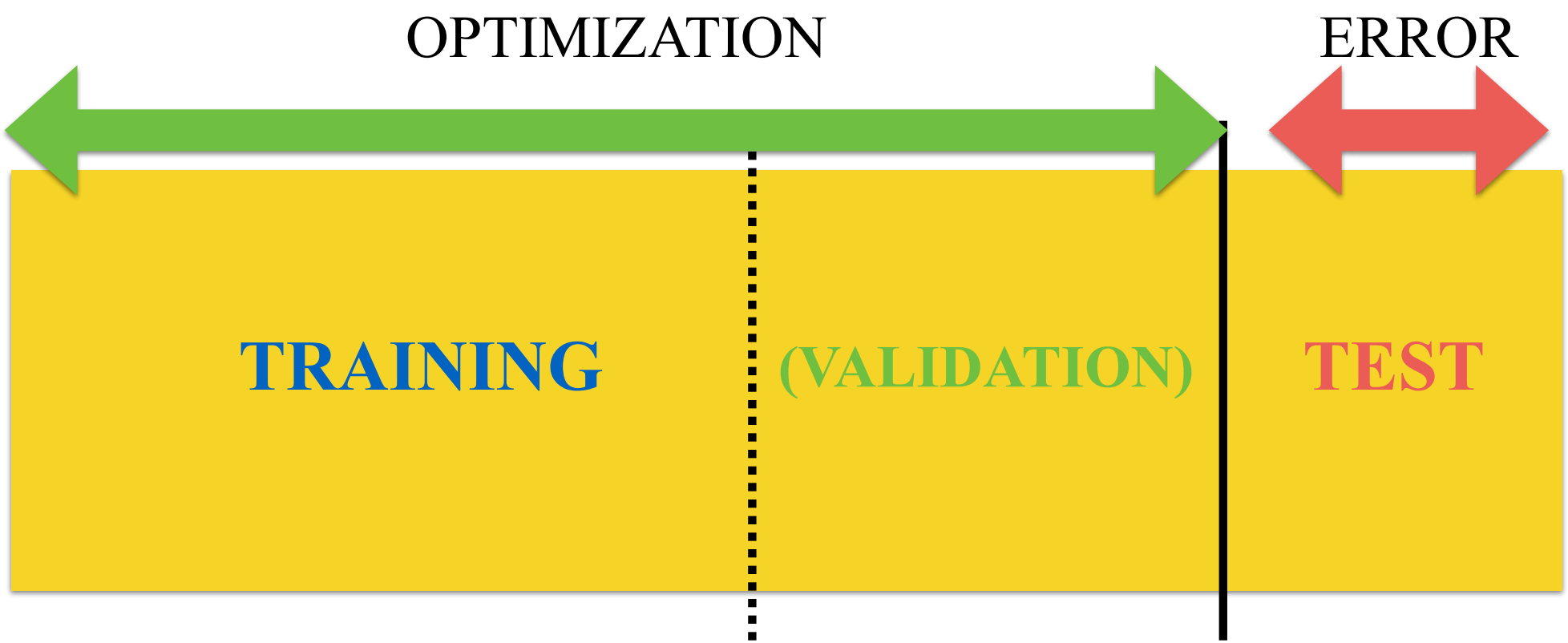
WE ARE MINIMISING WITH RESPECT TO A FINITE NUMBER OF OBSERVED EXAMPLES

ALL “GALAXIES IN THE UNIVERSE”



OBSERVED DATASET

IN PRACTICE

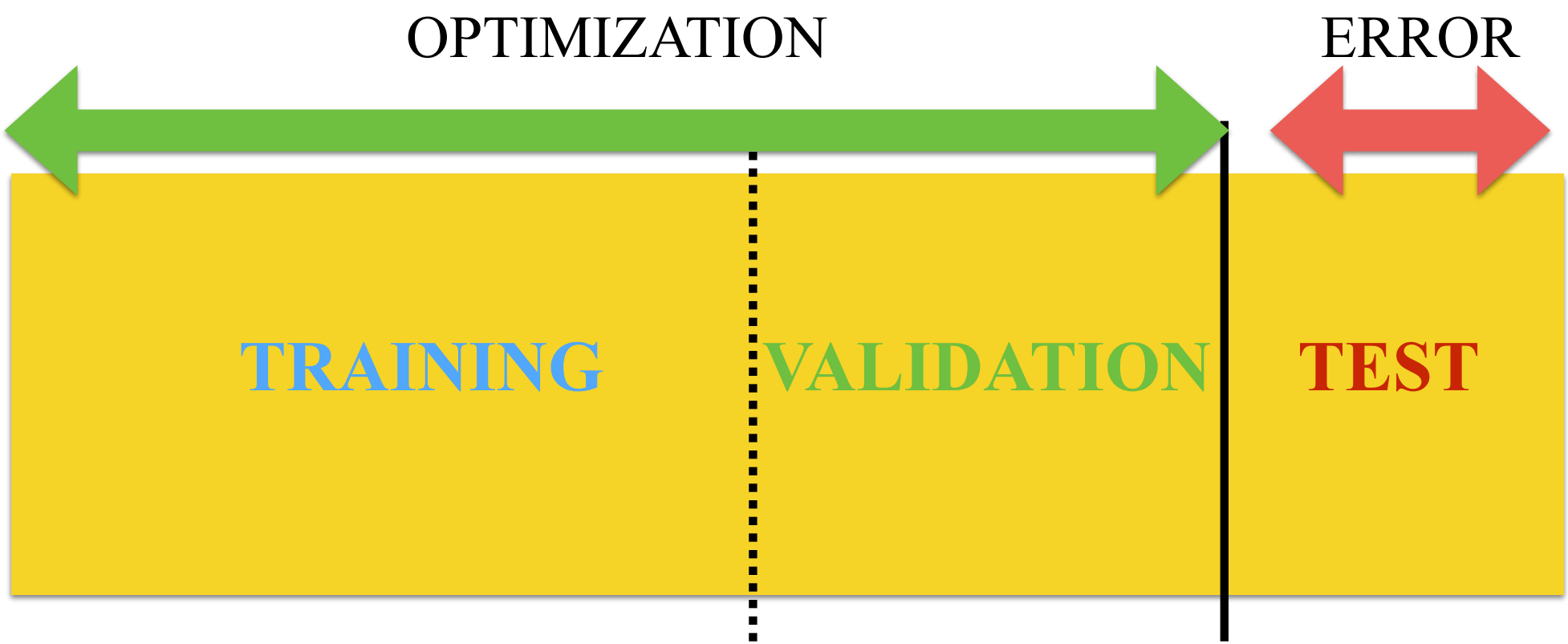


training set: use to train the classifier

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

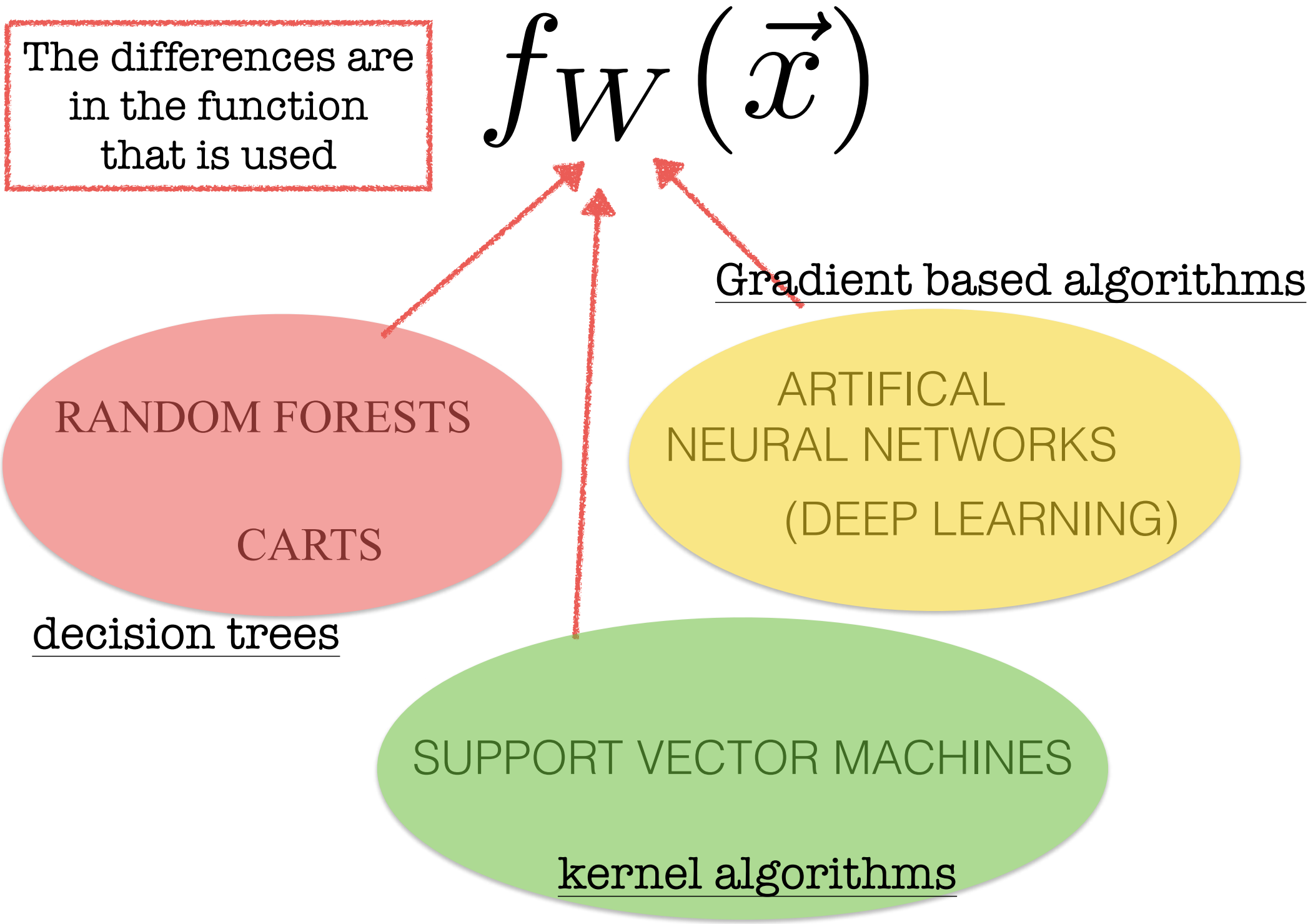
NEURAL NETWORKS USE THE GRADIENT DESCENT AS WE WILL SEE LATER

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

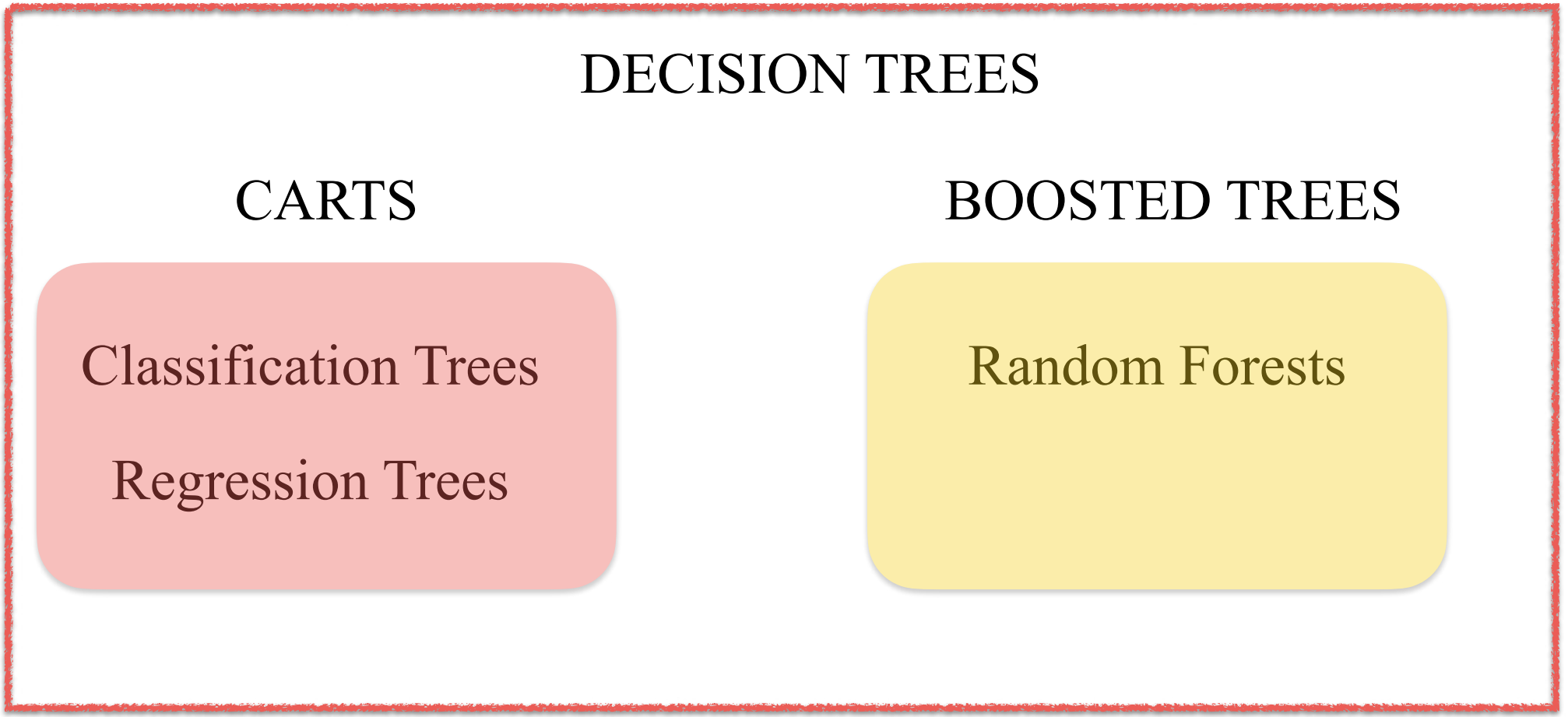
weights to be learned

epoch

learning rate
(hyper parameter)

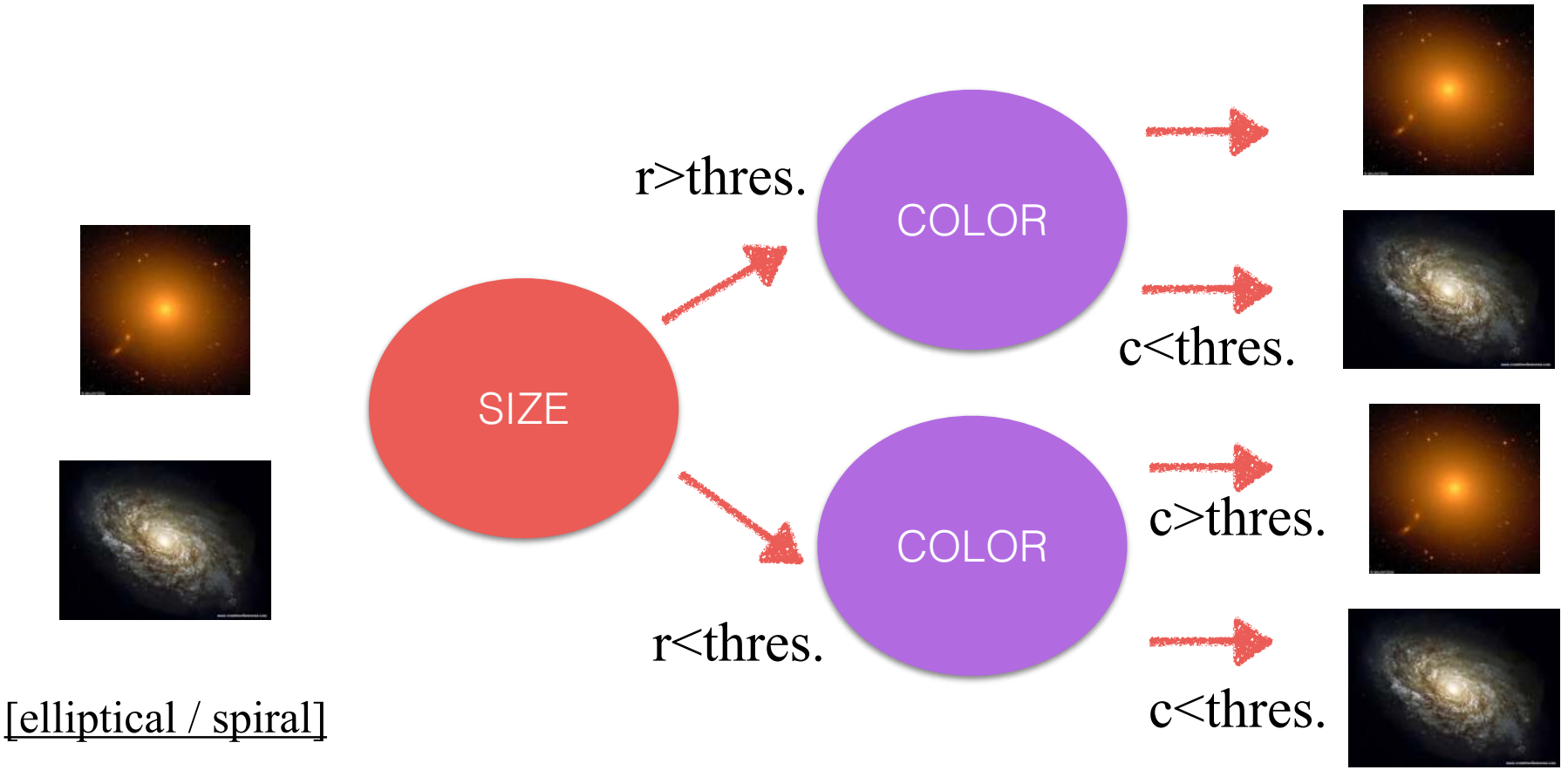

The differences are in the function that is used
$f_W(\vec{x})$
Gradient based algorithms
RANDOM FORESTS
CARTS
decision trees
ARTIFICAL NEURAL NETWORKS
(DEEP LEARNING)
SUPPORT VECTOR MACHINES
kernel algorithms

DECISION TREES ALGORITHMS



CLASSIFICATION AND REGRESSION TREES (CARTS)

THIS IS THE SIMPLEST AND MORE INTUITIVE MACHINE LEARNING ALGORITHM



r>thres.

SIZE

r<thres.

[elliptical / spiral]

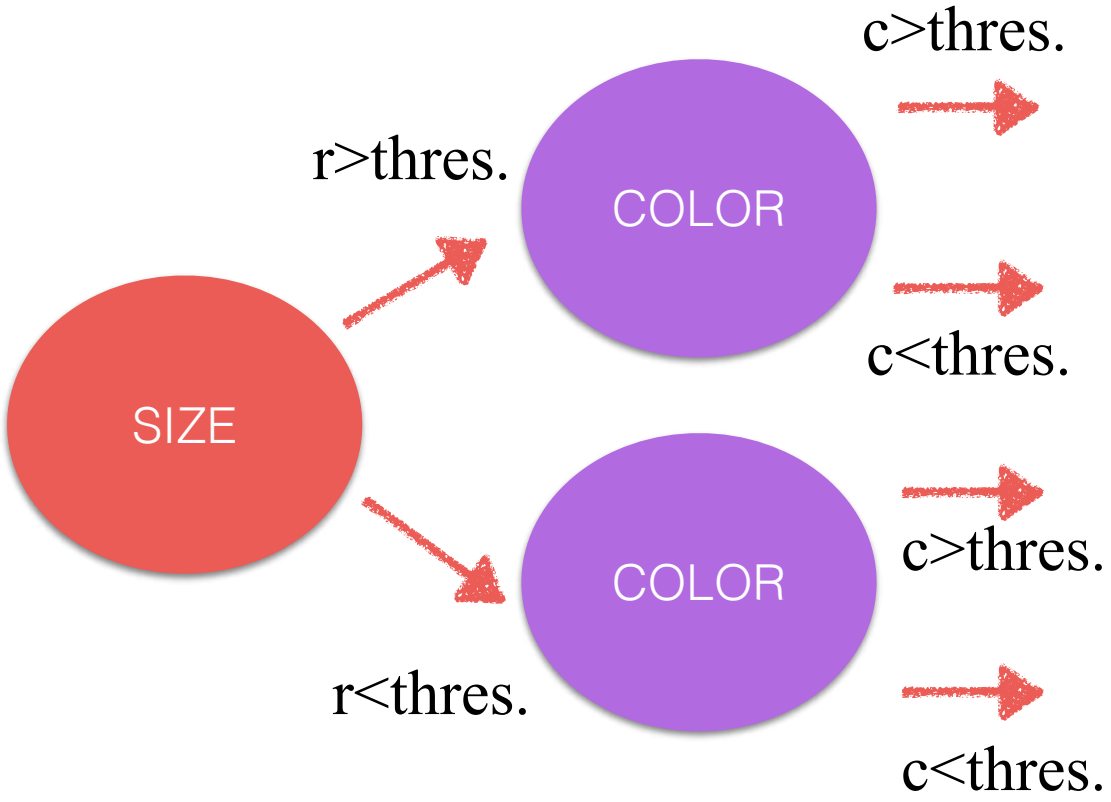
THE MINIMISATION ALGORITHM IS AS FOLLOWS:

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]

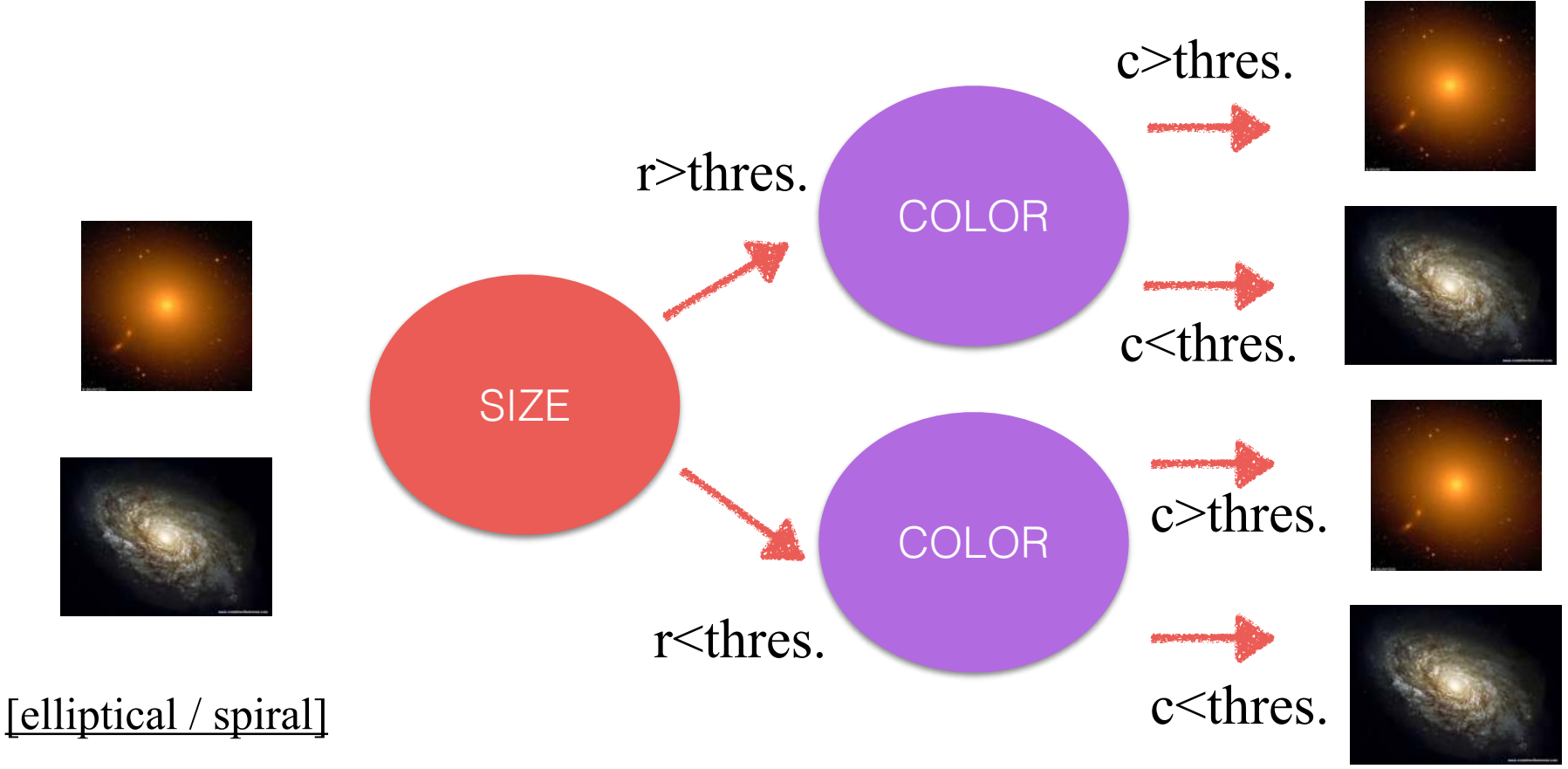


THE MINIMISATION ALGORITHM IS AS FOLLOWS:

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)

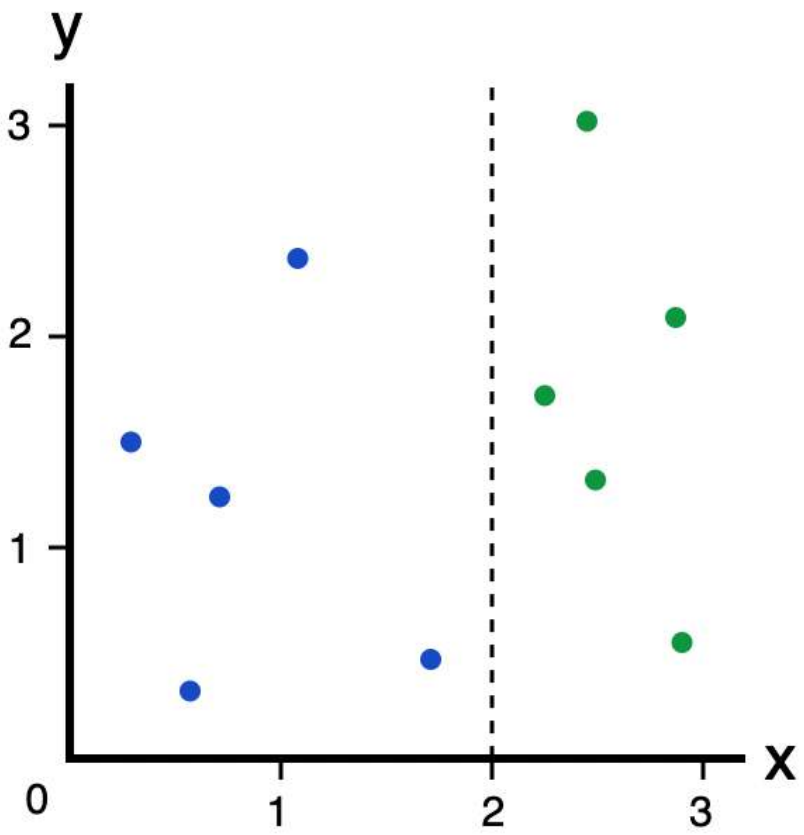
TYPICAL LOSS FUNCTION USED:

GINI IMPURITY

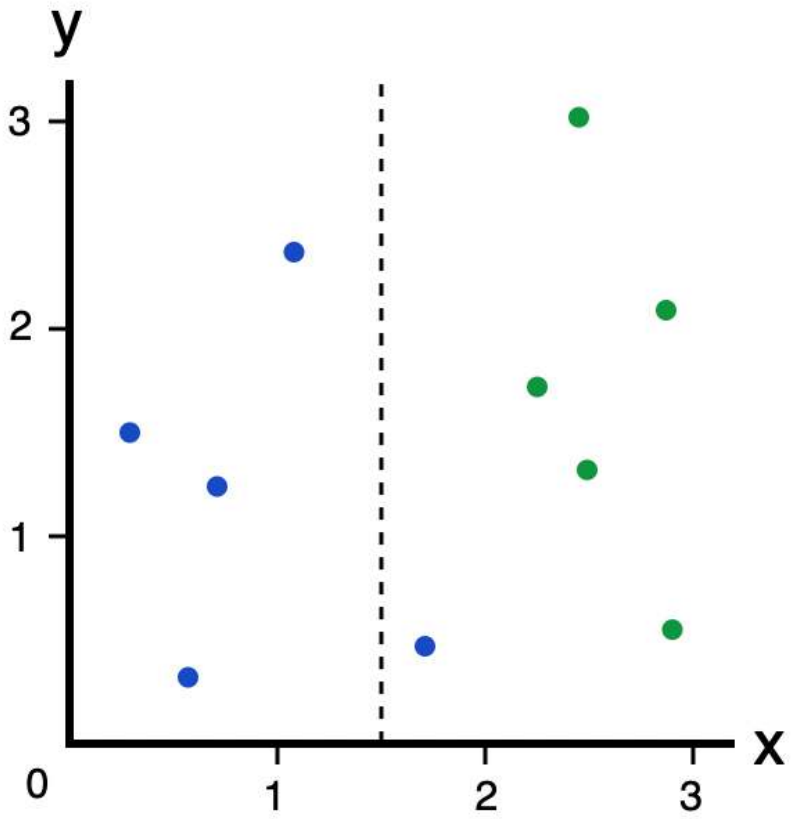
$$\sum_{j=1}^{c} p_j \times (1 - p_j)$$

fraction of
objects in each class given
a split value

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



A Perfect Split



An Imperfect Split

DECISION TREES (CARTS)

[b/a, color, size, SFR …]

TRAINING DATA

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

(parameters)

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

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

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

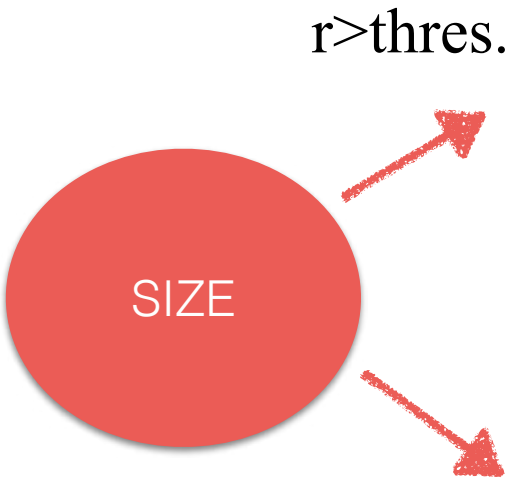
$\vec{y_i} = (y_1, y_2, ..., y_n)$

(classes)

[elliptical / spiral]

DECISION TREES (CARTS)

[elliptical / spiral]



LOOK FOR THRES. SO THAT
GINI IMPURITY IS MINIMAL

DECISION TREES (CARTS)

[elliptical / spiral]

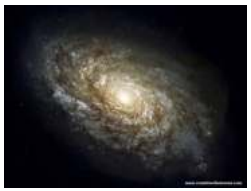
SIZE

r>thres.

COLOR

c>thres.

c<thres.

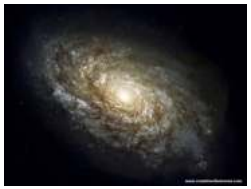


r<thres.

COLOR

c>thres.

c<thres.



LOOK FOR THRES. SO THAT GINI IMPURITY IS MINIMAL

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

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

- 5/9
- 0
- 1
- 3/8
- 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**

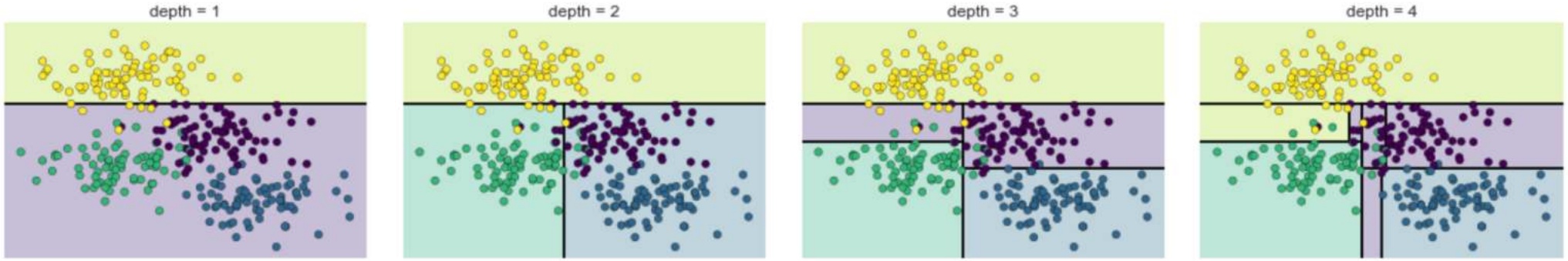
What would be the Gini Impurity with a split value of 6 kpc

- 4/9
- 0
- 1
- 3/8
- 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

[b/a, color, size, SFR …]

TRAINING DATA

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

(parameters)

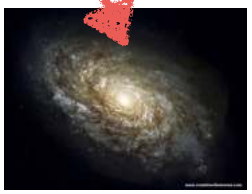
$(\vec{x_2}, \vec{y_2})$

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

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

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

(classes)



[elliptical / spiral]

RANDOM FORESTS

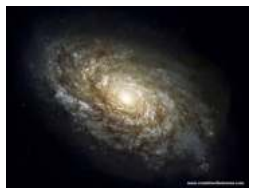
$$(\vec{x_2}, \vec{y_2})$$
$$(\vec{x_1}, \vec{y_1})$$
$$(\vec{x_3}, \vec{y_3}) \quad (\vec{x_4}, \vec{y_4})$$

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

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]

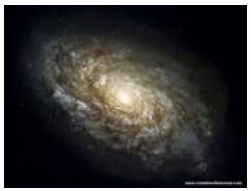
SIZE

r>thres.

COLOR

c>thres.

c<thres.

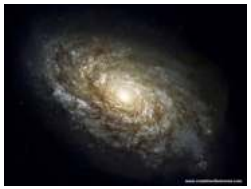


r<thres.

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.

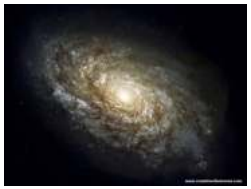
s<thres.

r<thres.

Sersic

s>thres.

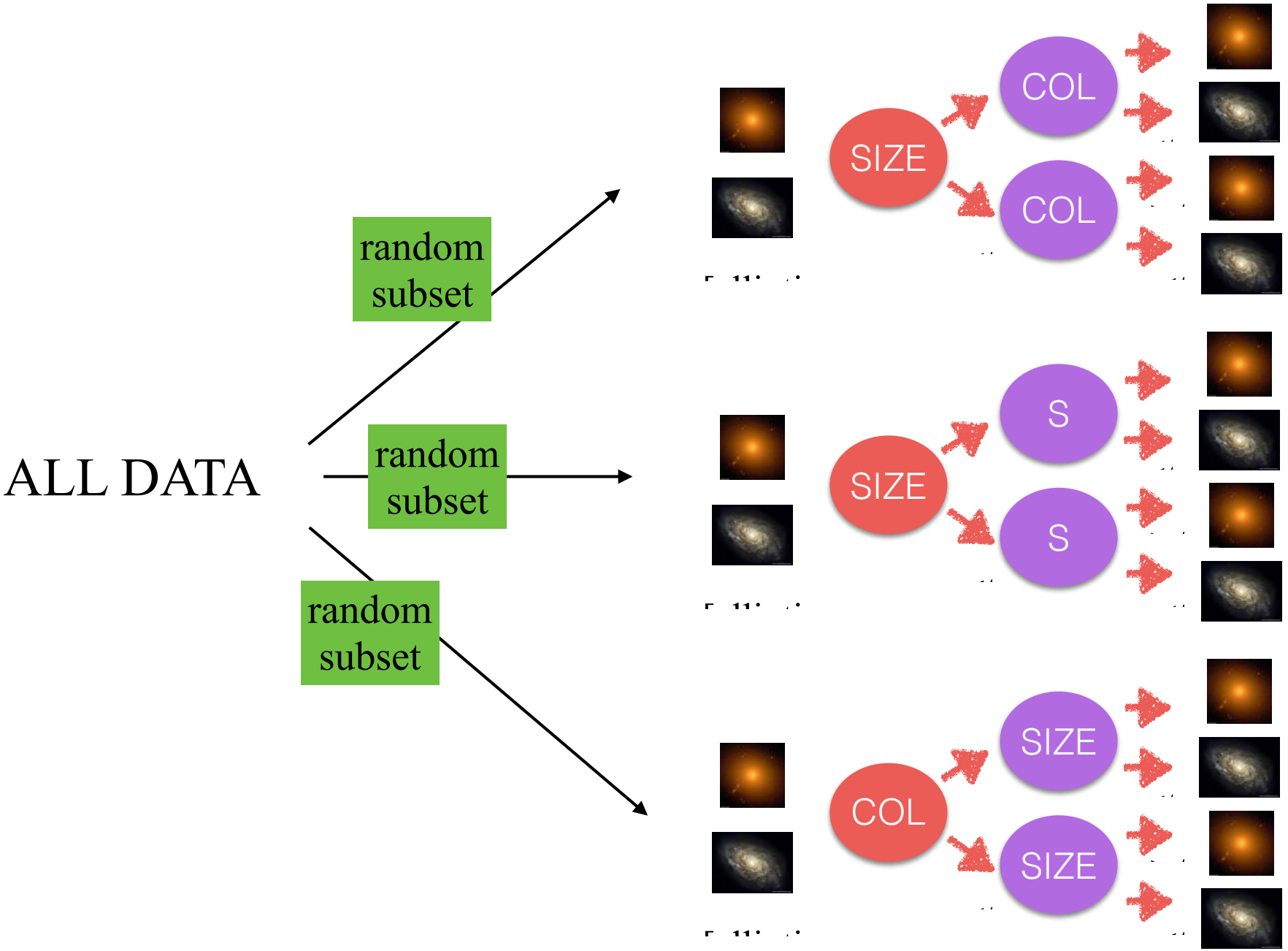
s<thres.



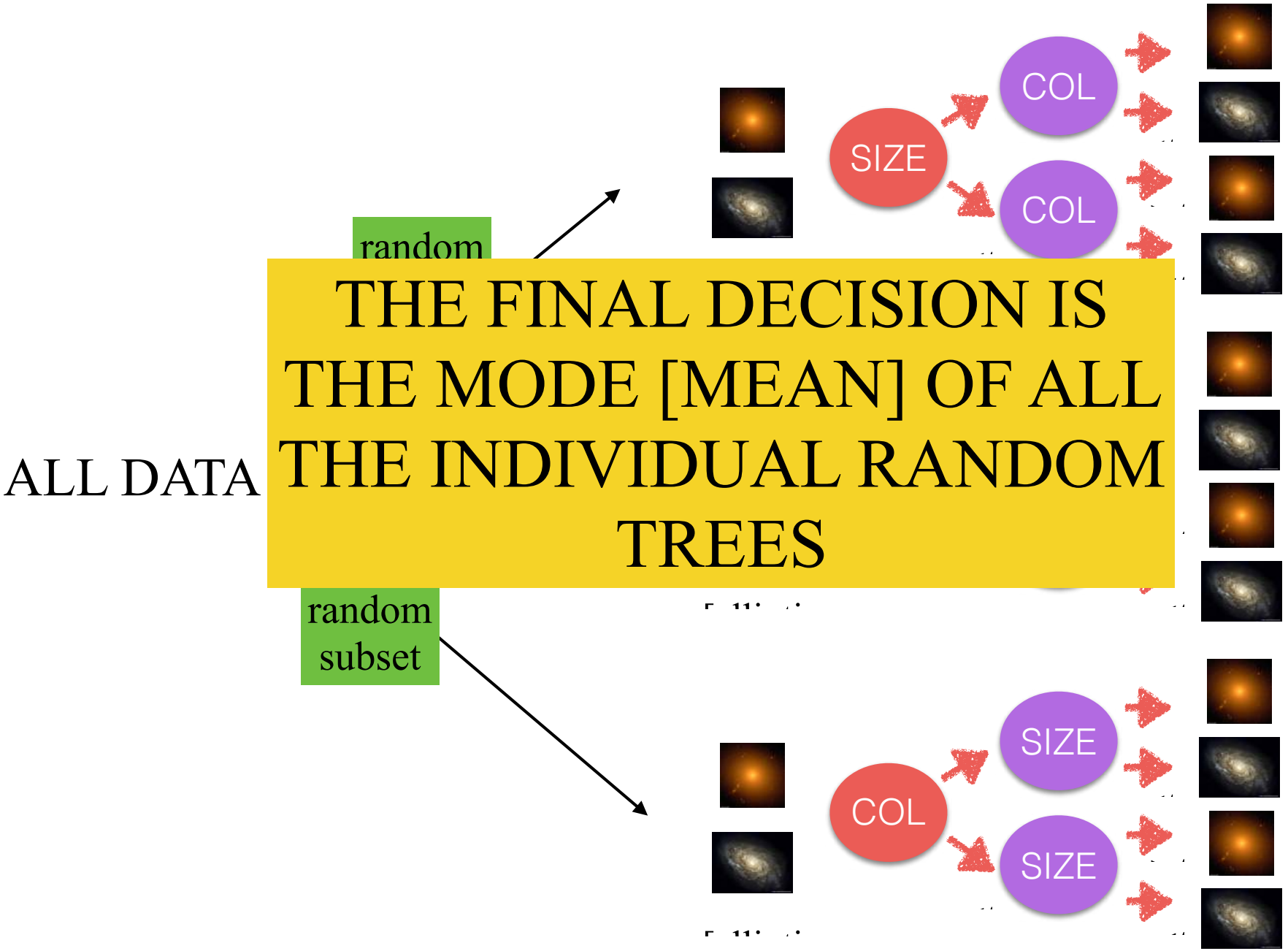
LOOK FOR THRES. SO THAT GINI IMPURITY IS MINIMAL

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

RANDOM FORESTS



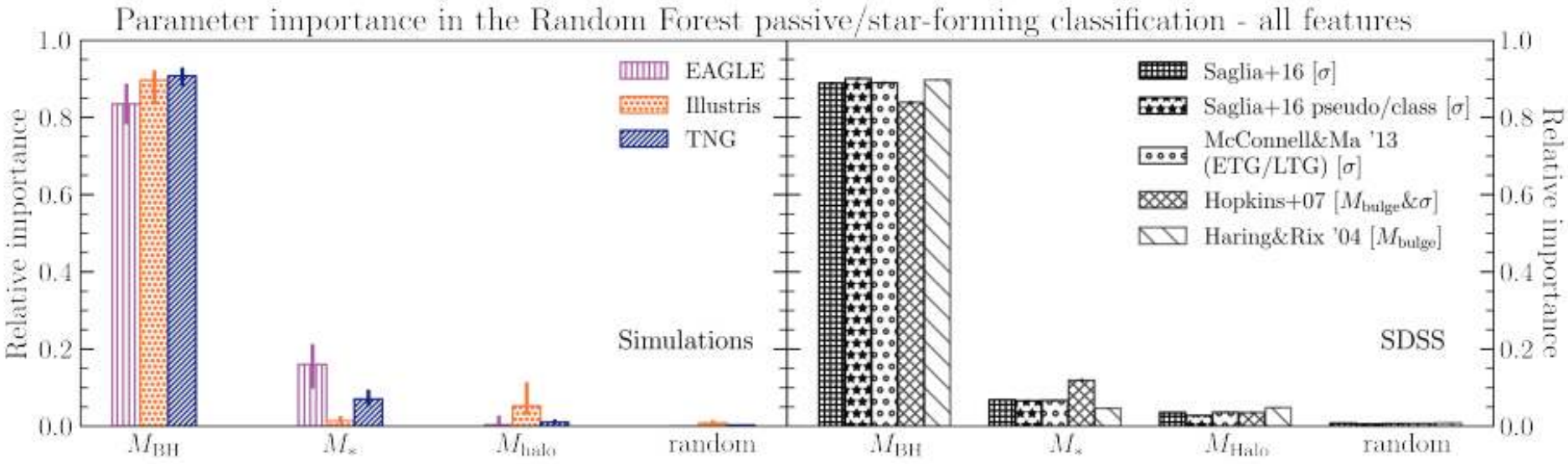
RANDOM FORESTS



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



Piotrowksa+22

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]
	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 TO PREDICT IF A GALAXY IS QUENCHED

Teimiroonia+16

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]

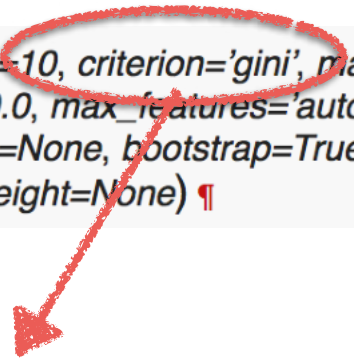
NUMBER OF RANDOM TREES

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]



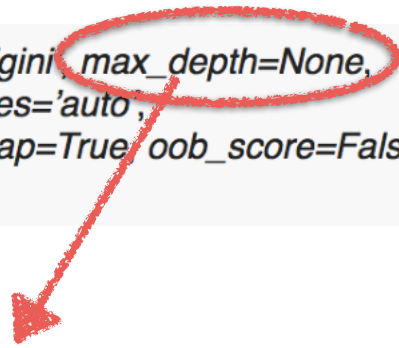
METRIC TO MINIMIZE

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

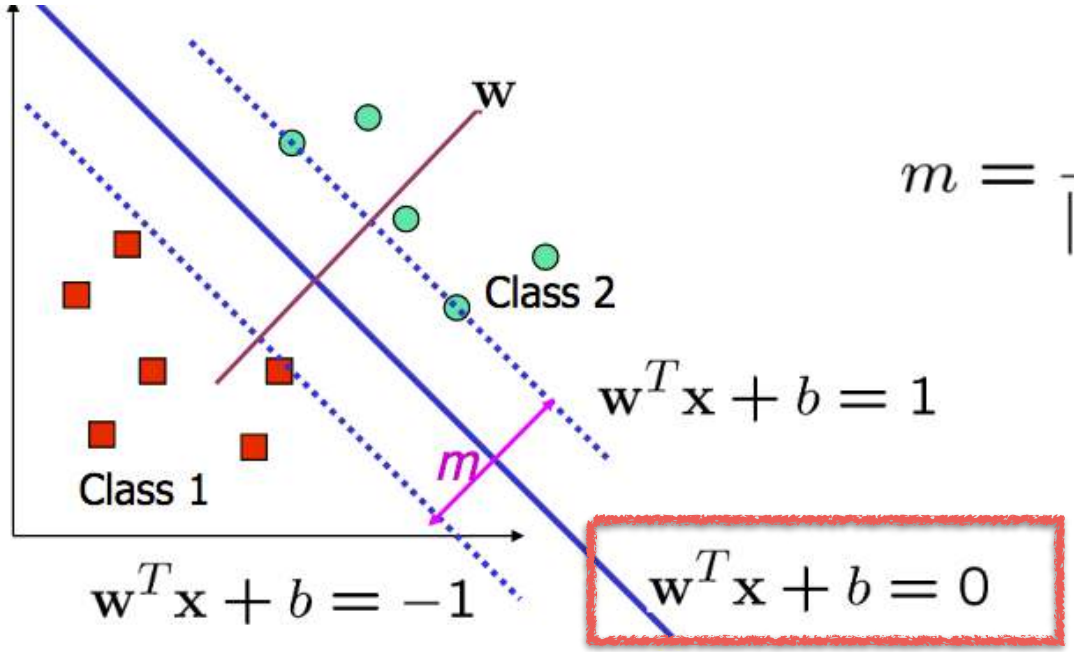
class sklearn.ensemble. **RandomForestClassifier** (*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, 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,
>>> print(clf.predict([[0, 0, 0, 0]]))
[1]
```

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

Support Vector Machines

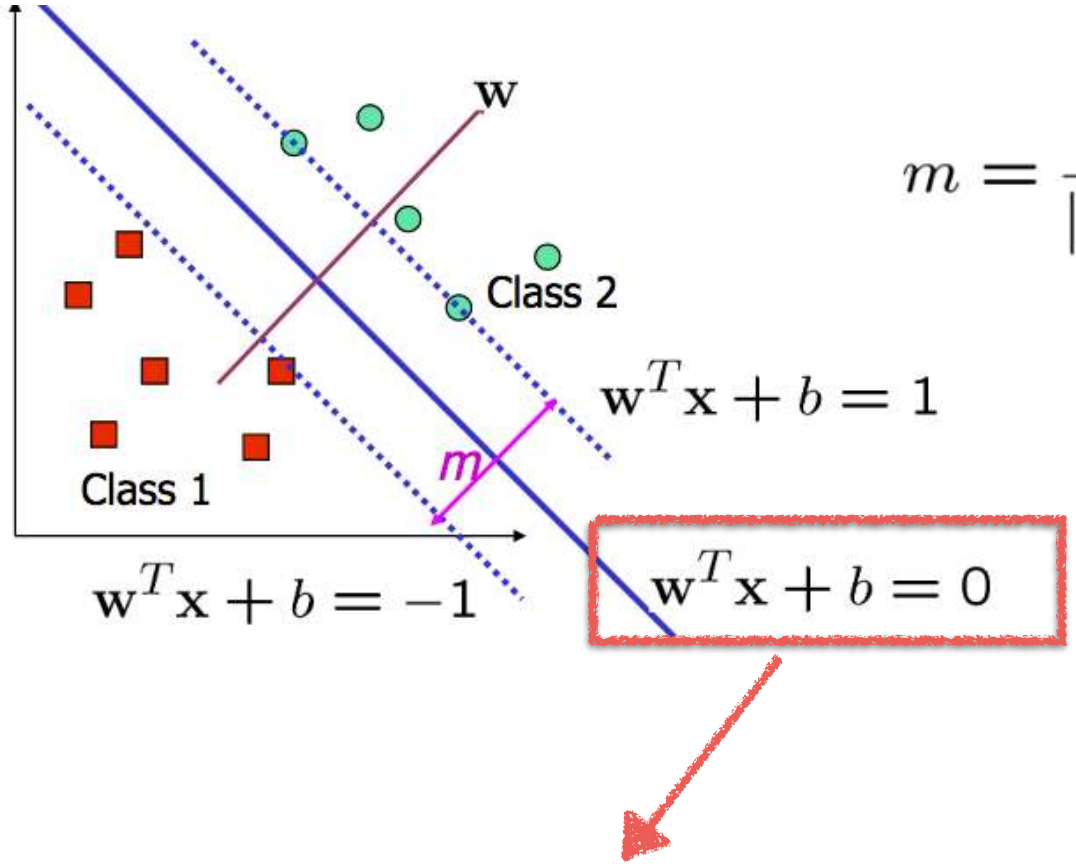


$$m = \frac{2}{||\mathbf{w}||}$$

equation of the hyperplane

[Credit: M. Law]

Support Vector Machines



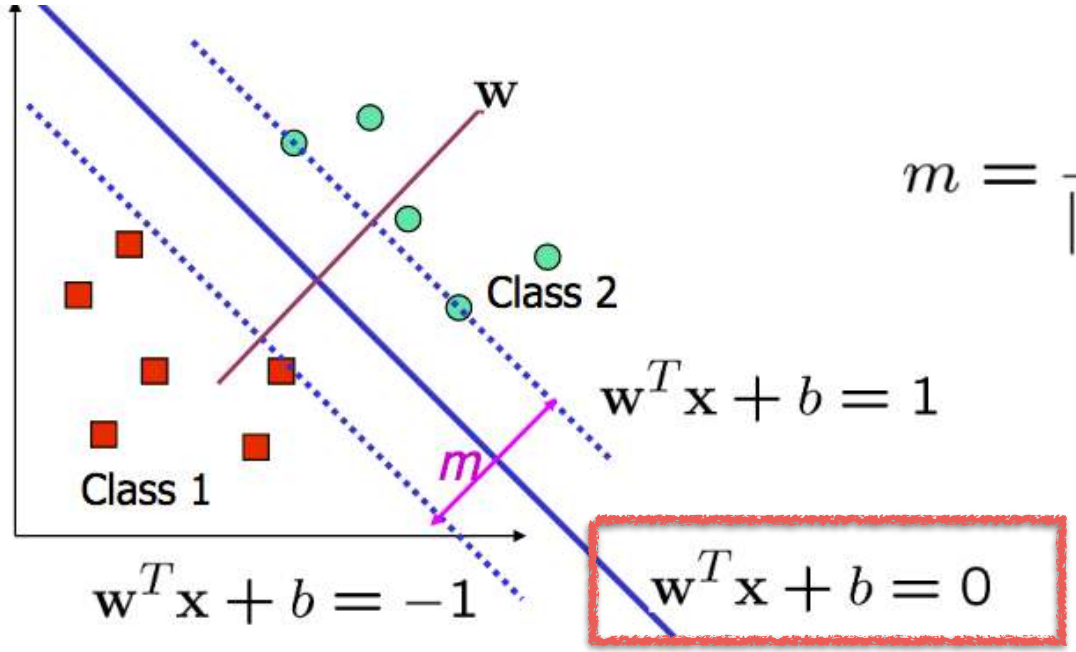
$$m = \frac{2}{||\mathbf{w}||}$$

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



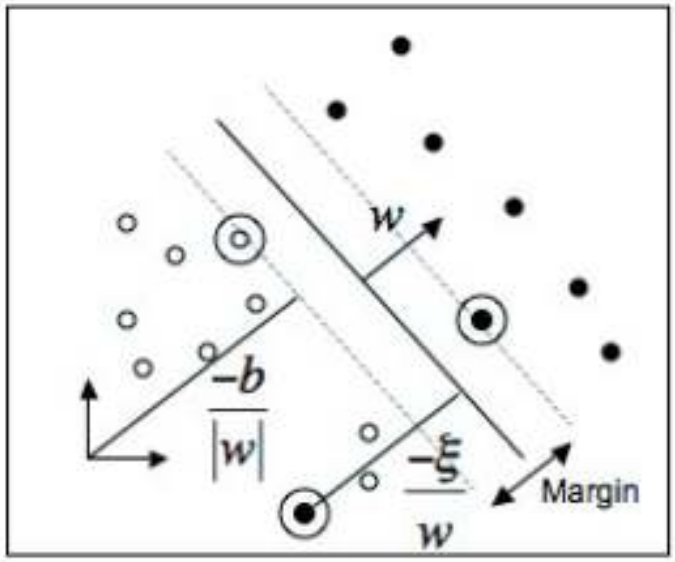
$$m = \frac{2}{||\mathbf{w}||}$$

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

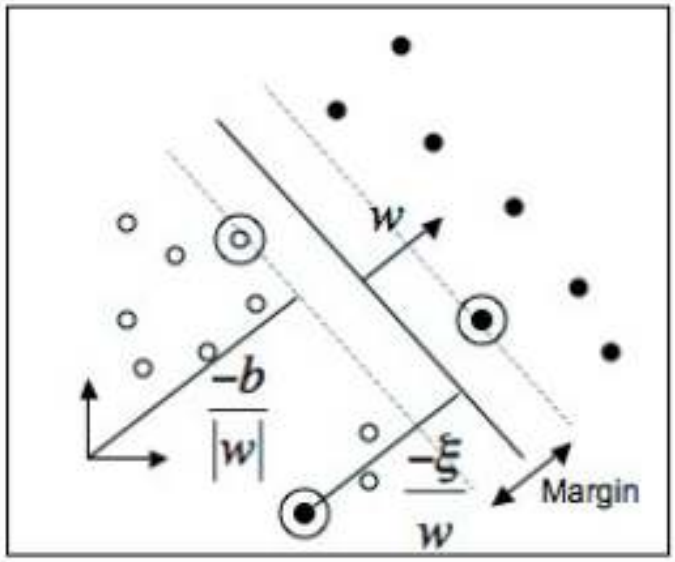
$$\mathbf{x_i}.\mathbf{w} + b \geq +1 - \epsilon_i$$

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

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

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

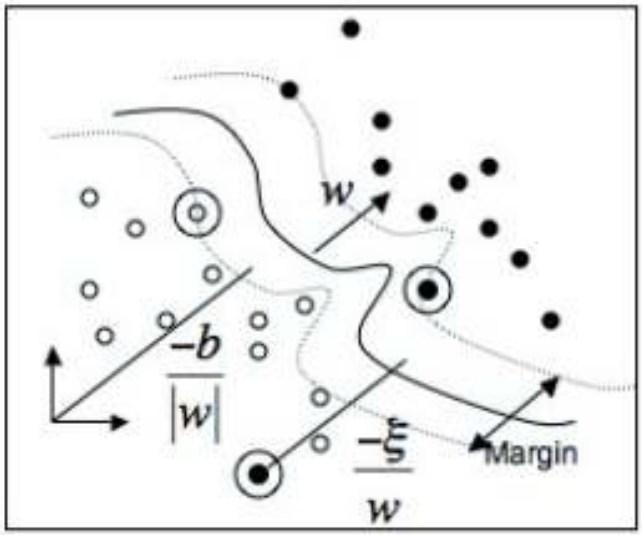
$$\mathbf{x_i}.\mathbf{w} + b \geq +1 + \epsilon_i$$

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

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

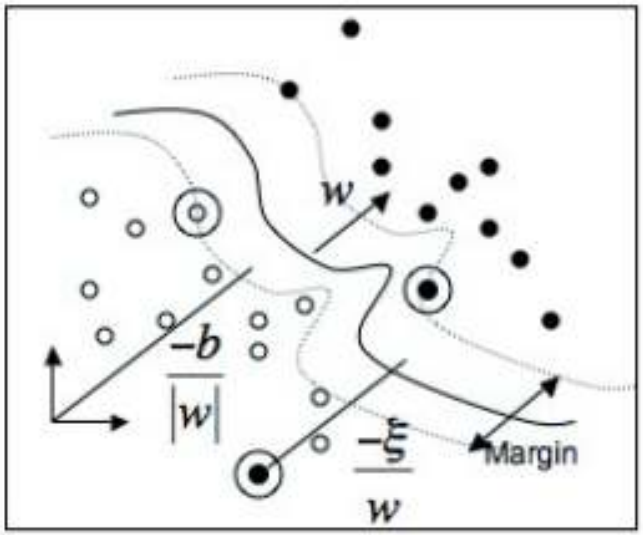
SINCE WE ONLY NEED TO COMPUTE INNER PRODUCTS IN THIS SPACE

$$x_i . x_j$$

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

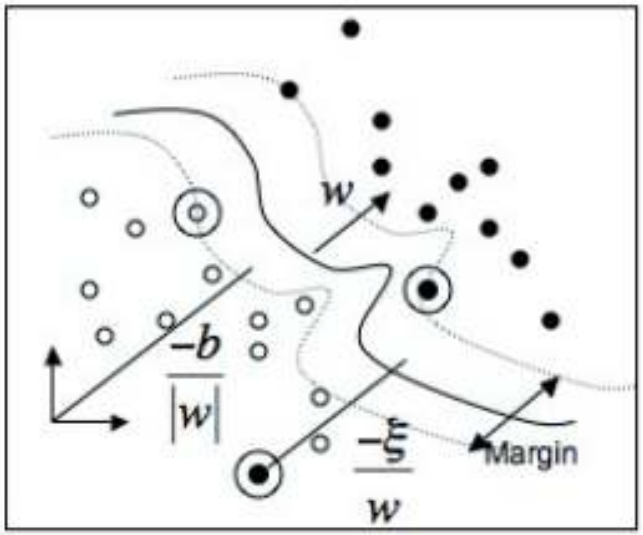
THE FINAL TRICK IS TO FIND NON-LINEAR BOUNDARIES

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

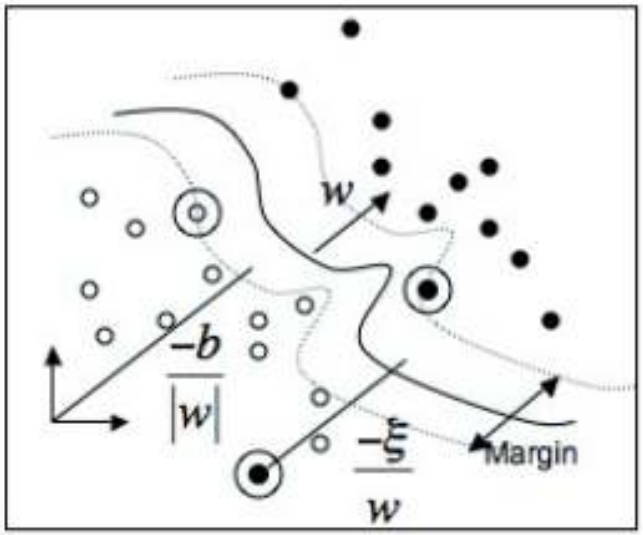
IF THERE IS A **KERNEL FUNCTION** SO THAT:

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

THEN THERE IS NO NEED TO EXPLICITLY KNOW Φ

SVM IS CALLED **A KERNEL METHOD**

Support Vector Machines

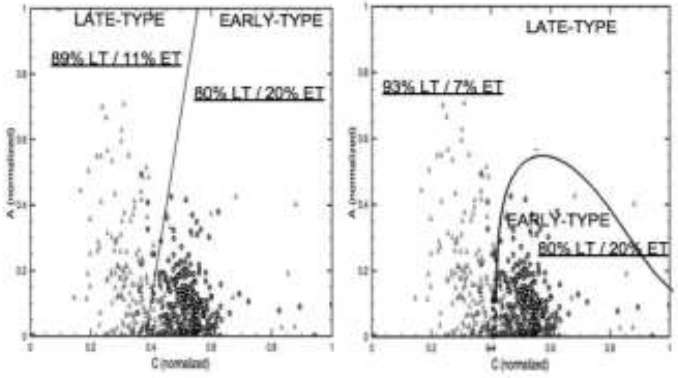


THE FINAL TRICK IS TO FIND NON-LINEAR BOUNDARIES

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

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

Huertas-Company-



IS NO NEED TO EXPLICITLY KNOW Φ

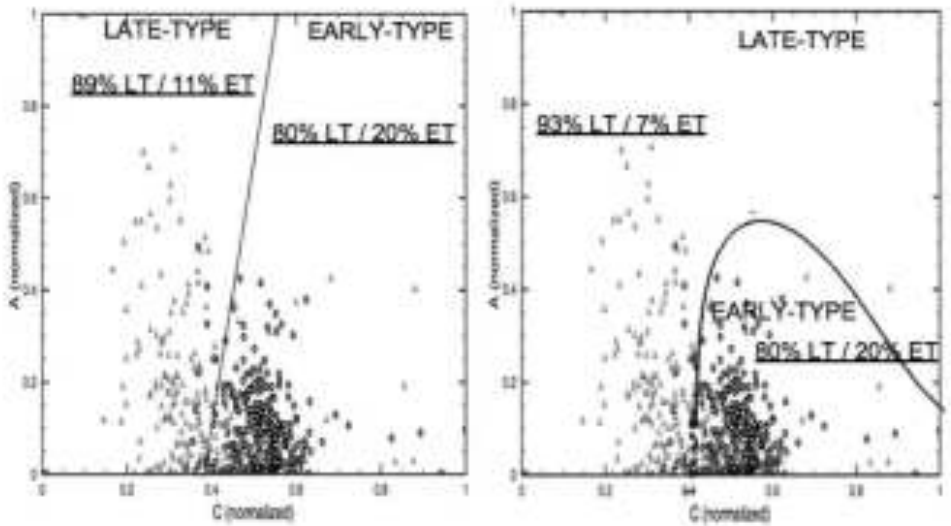
SVM IS CALLED

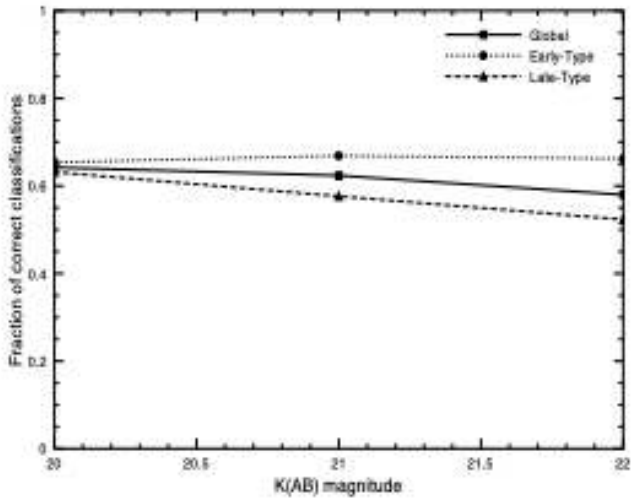
EXAMPLE OF KERNEL:

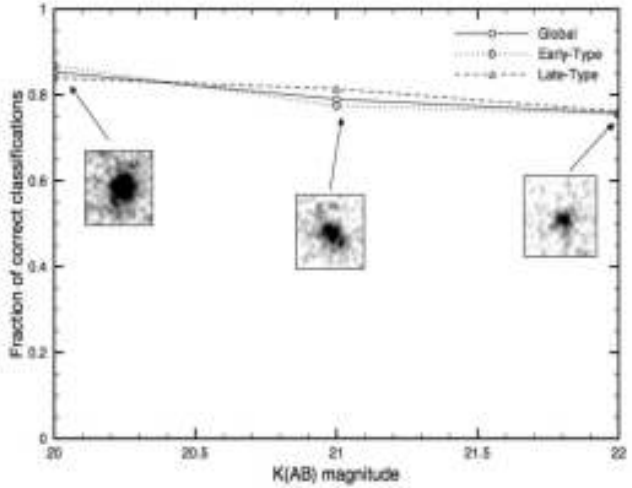
$$K(x, y) = e^{-g||x-y||^2}$$

[GAUSSIAN RBF]

Support Vector Machines





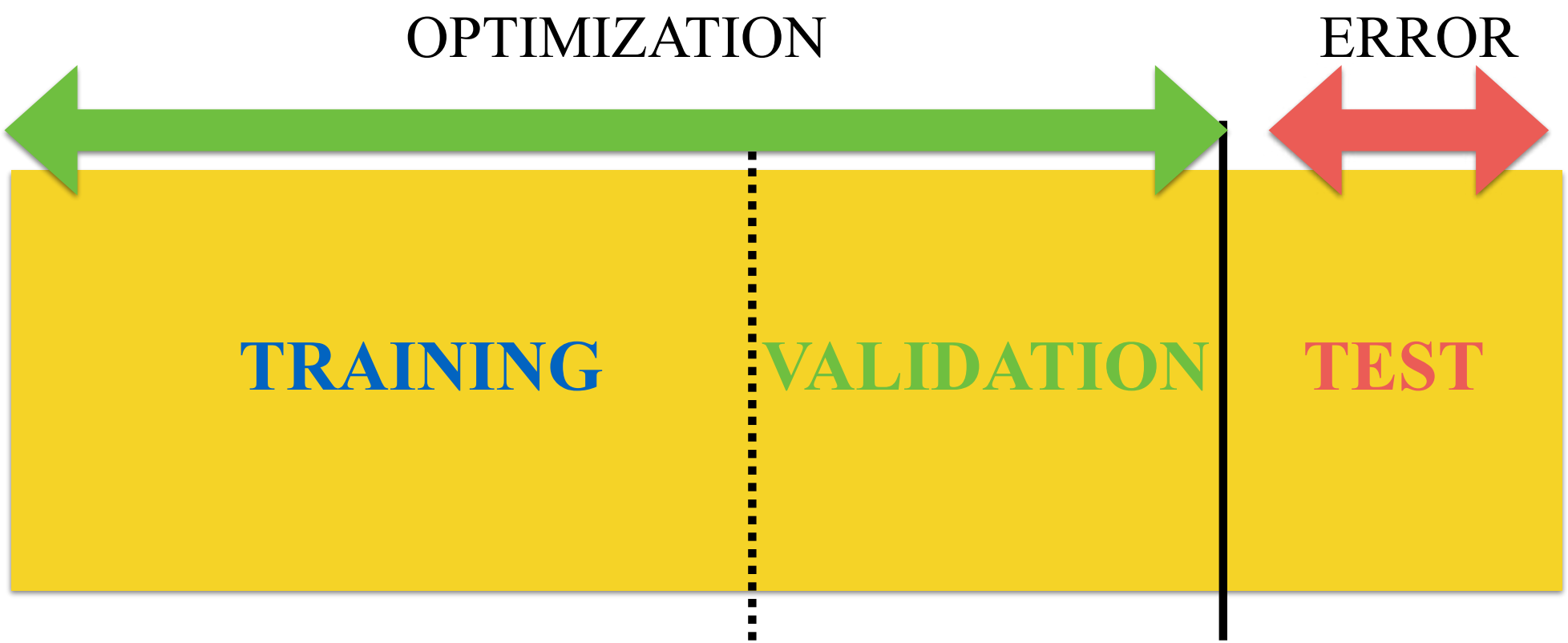


Huertas-Company+08

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

true positives

true negatives

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

MEASURES HOW MANY OBJECTS ARE CORRECTLY CLASSIFIED

Evaluation of results [binary class.]

THE MOST STRAIGHTFORWARD WAY IS TO EVALUATE THE TOTAL ACCURACY

NOT VERY INFORMATIVE IF UNBALANCED CLASSES

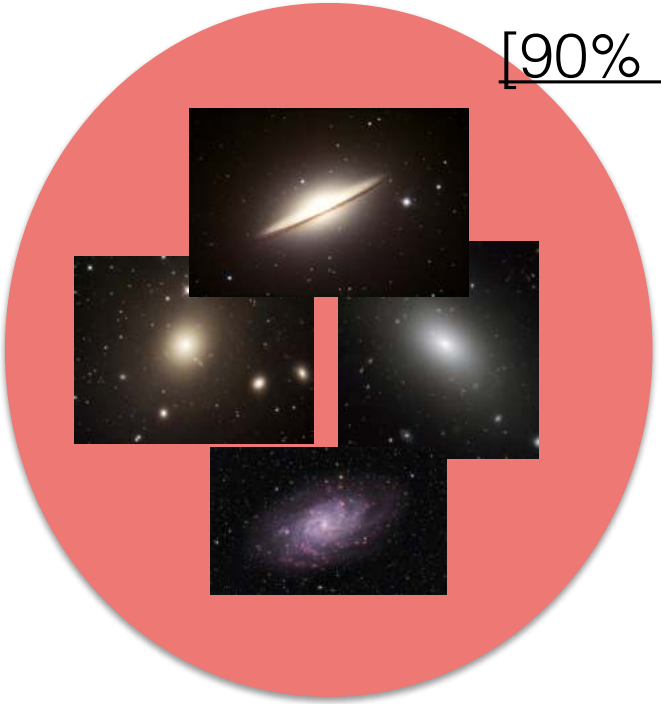
true po ue negatives

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

MEASURES HOW MANY OBJECTS ARE CORRECTLY CLASSIFIED

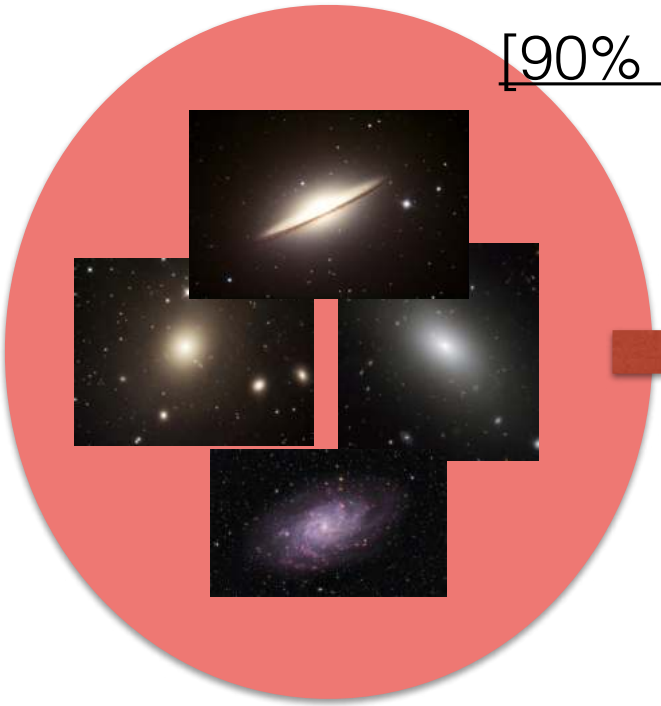
IMAGINE AN EXTREME CASE WITH VERY UNBALANCED DATA…

[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.]

- 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

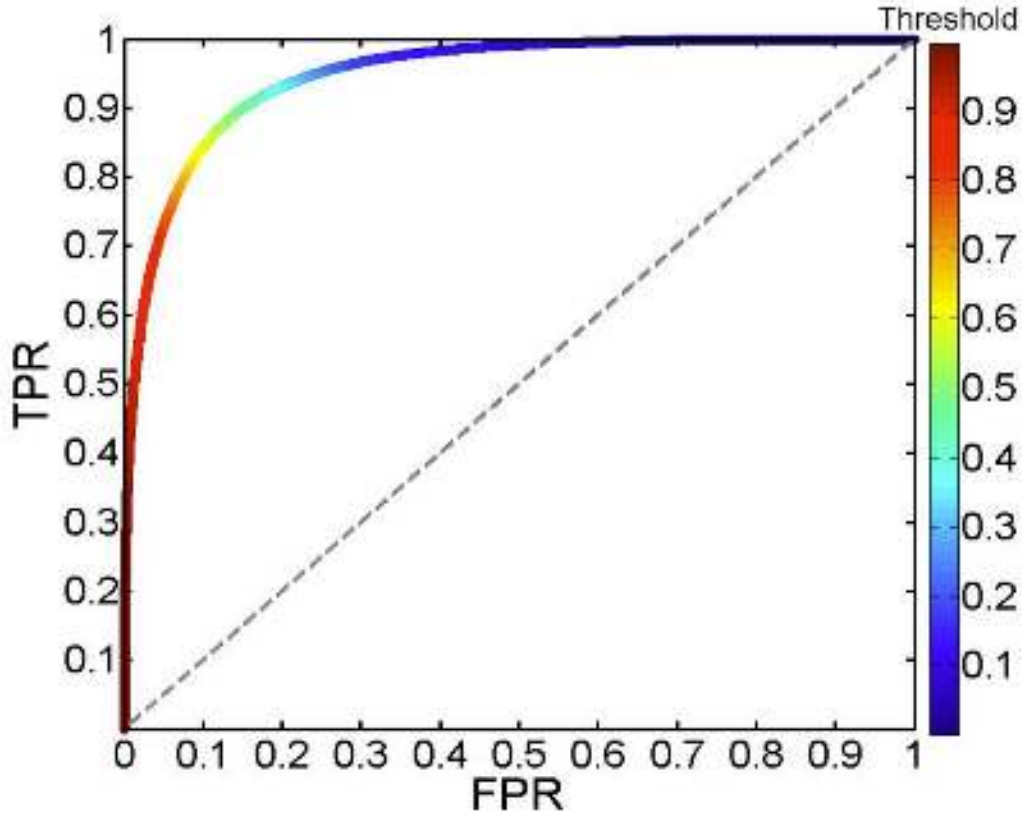
"Fraction of positive examples classified correctly"

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

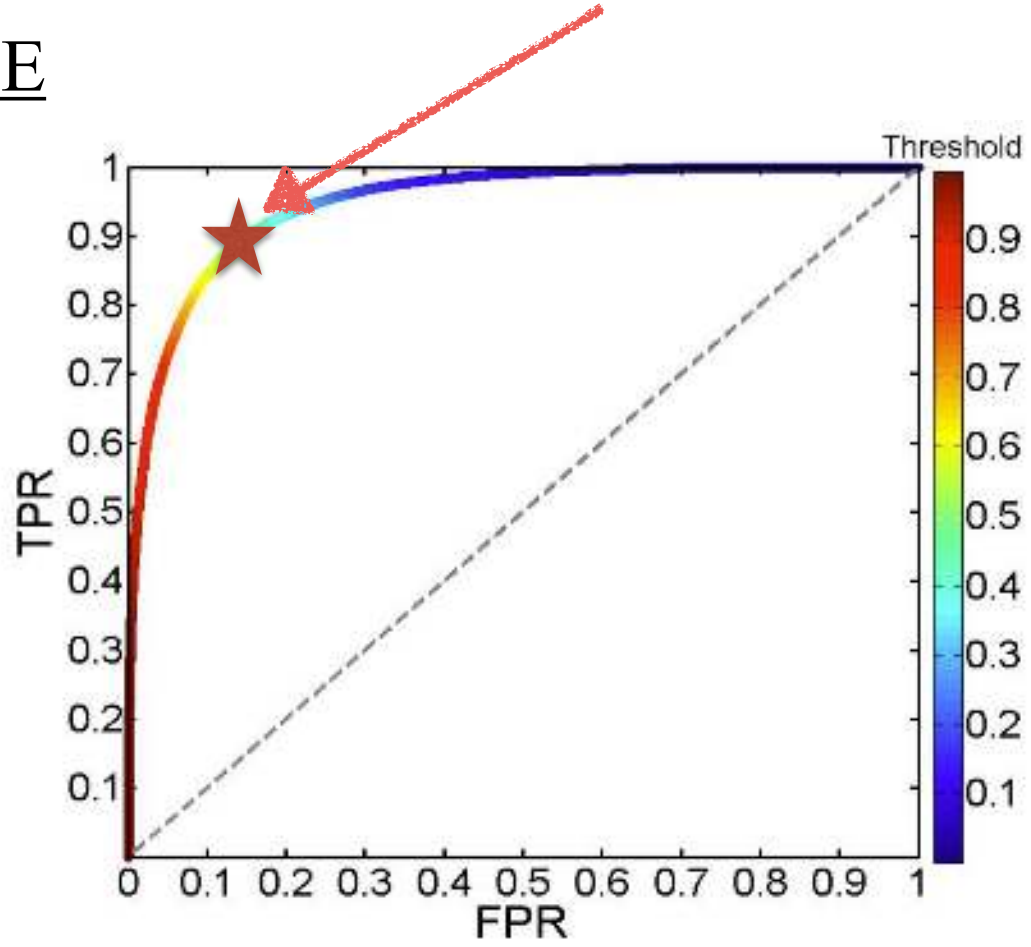


Teimiroonia+16

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

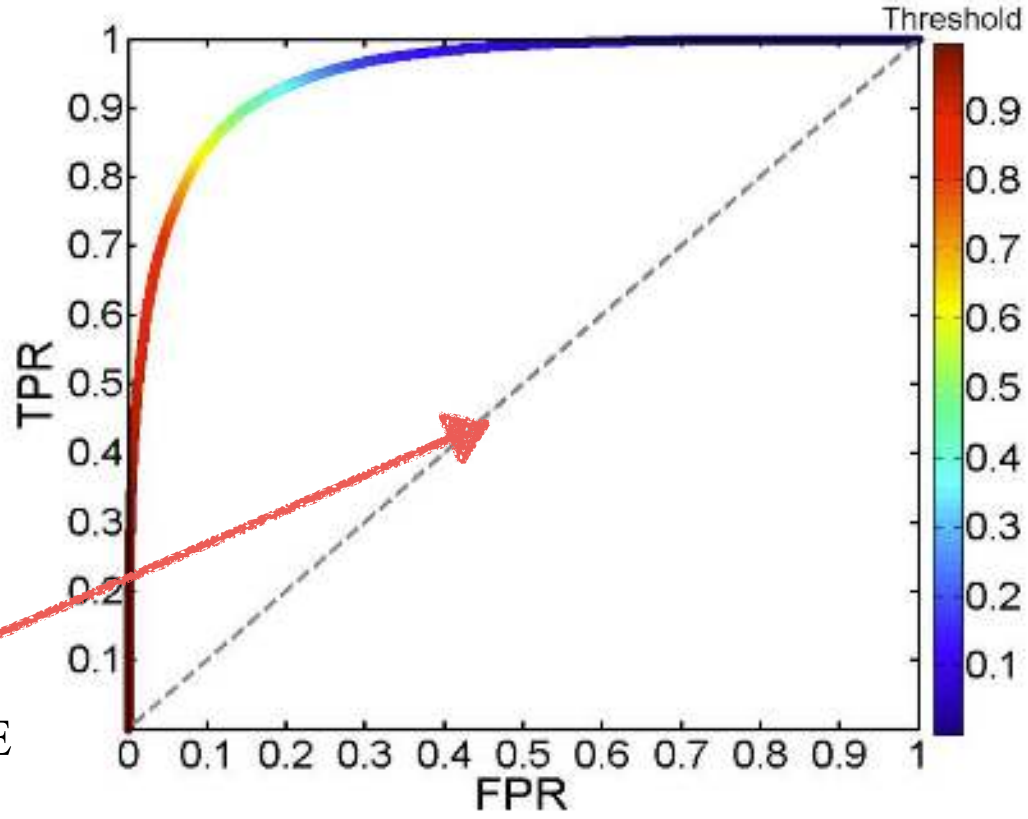
ROC CURVE



Teimiroonia+16

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

ROC CURVE



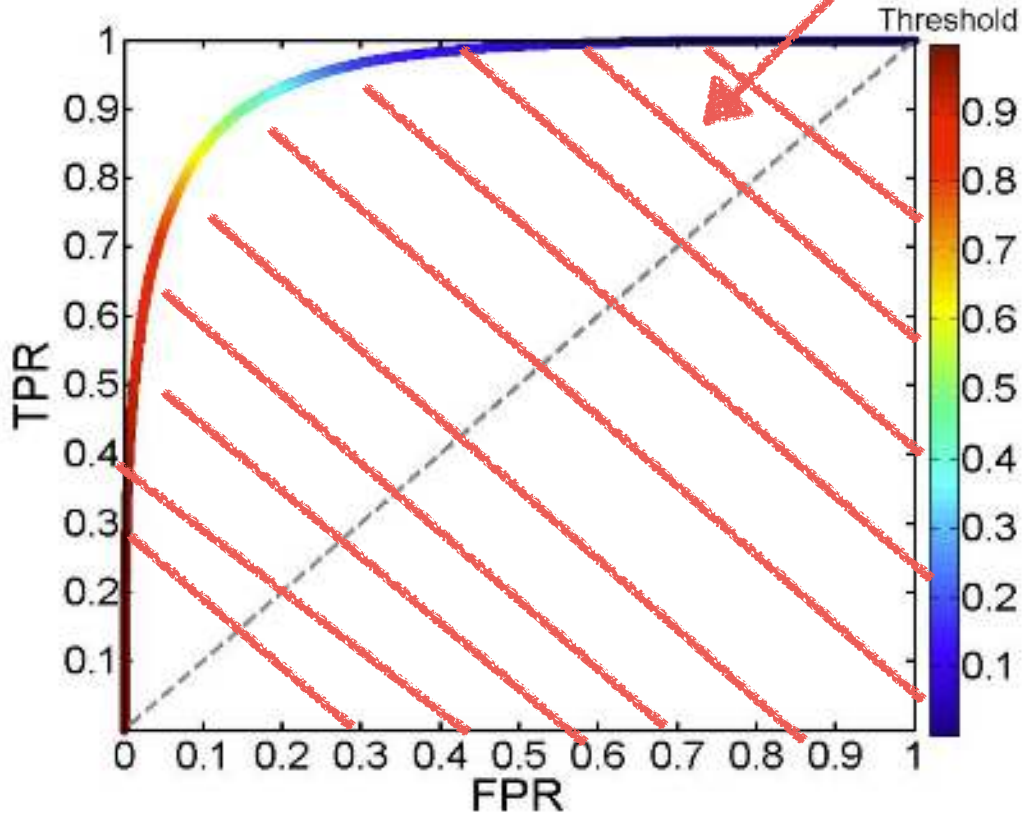
THE ONE-TO-ONE LINE IS A RANDOM CLASSIFICATION

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

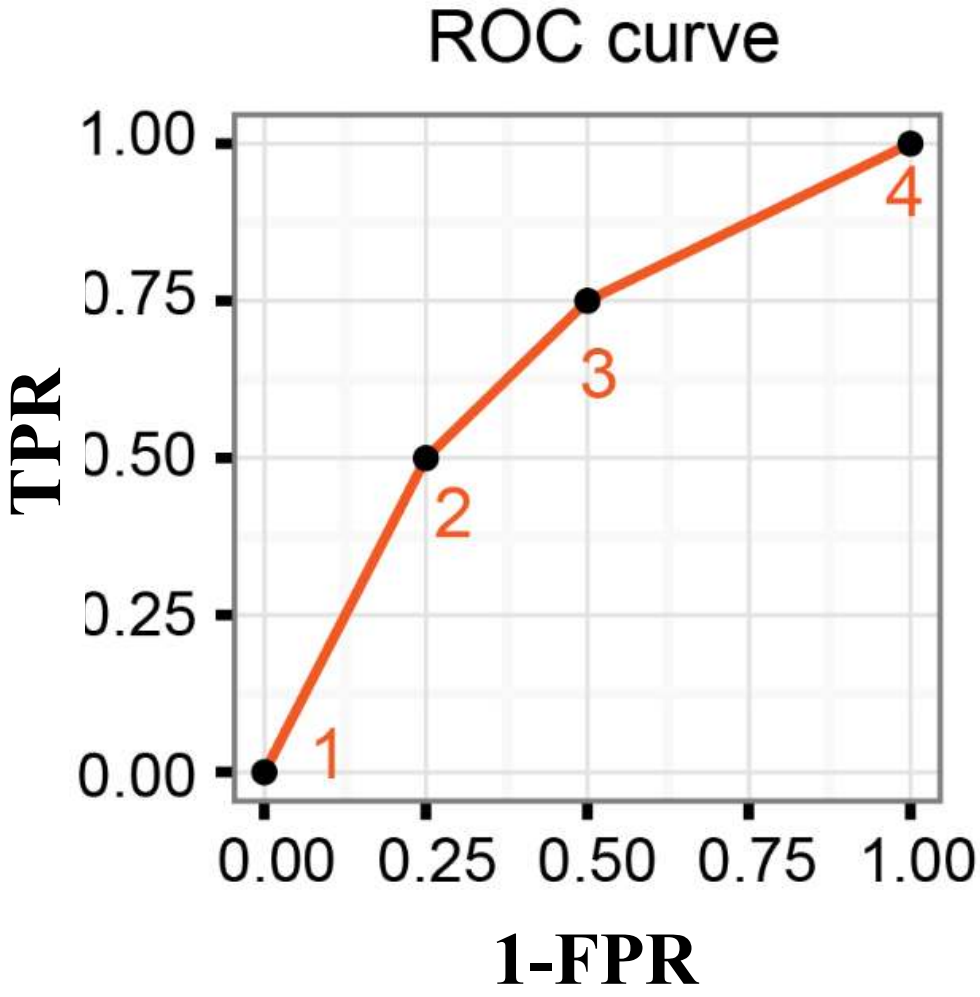
THE P-R CURVE (Precision - Recall)

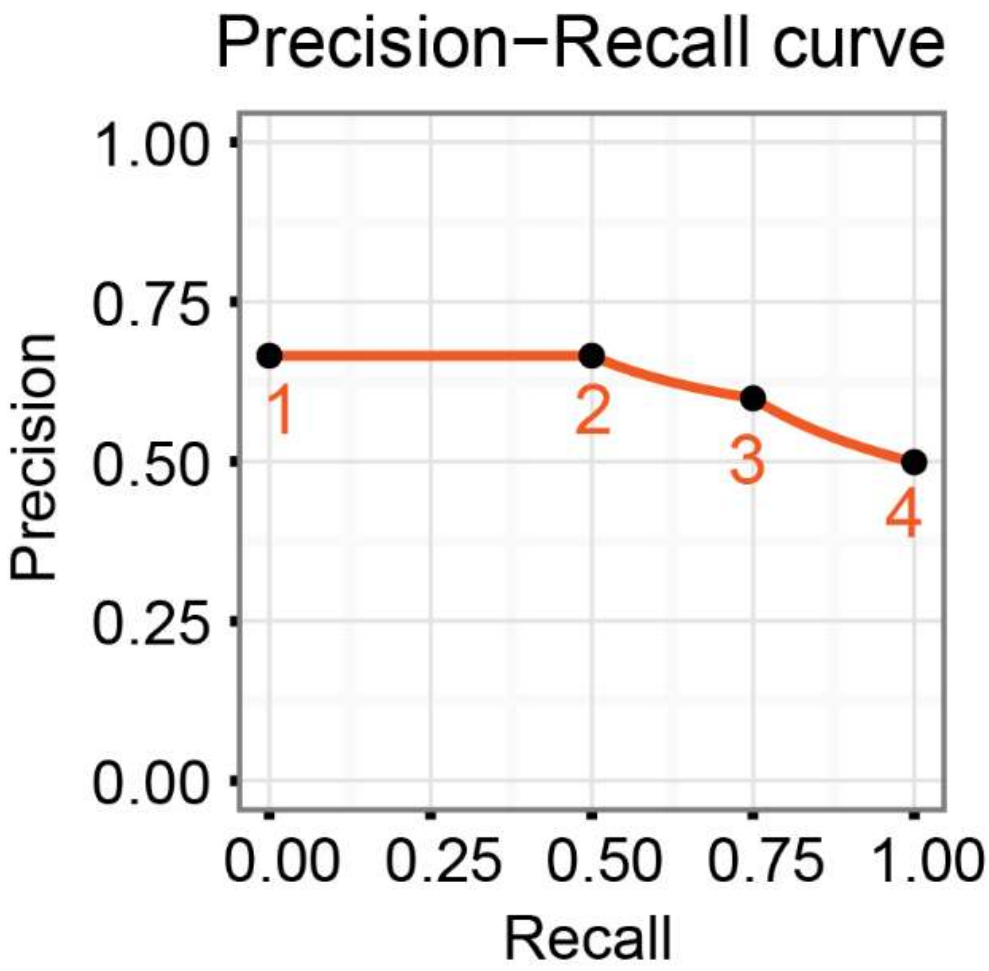
$$Recall = \frac{TP}{TP + FN} = TPR$$ [completeness]

$$Precision = \frac{TP}{TP + FP}$$ [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 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.

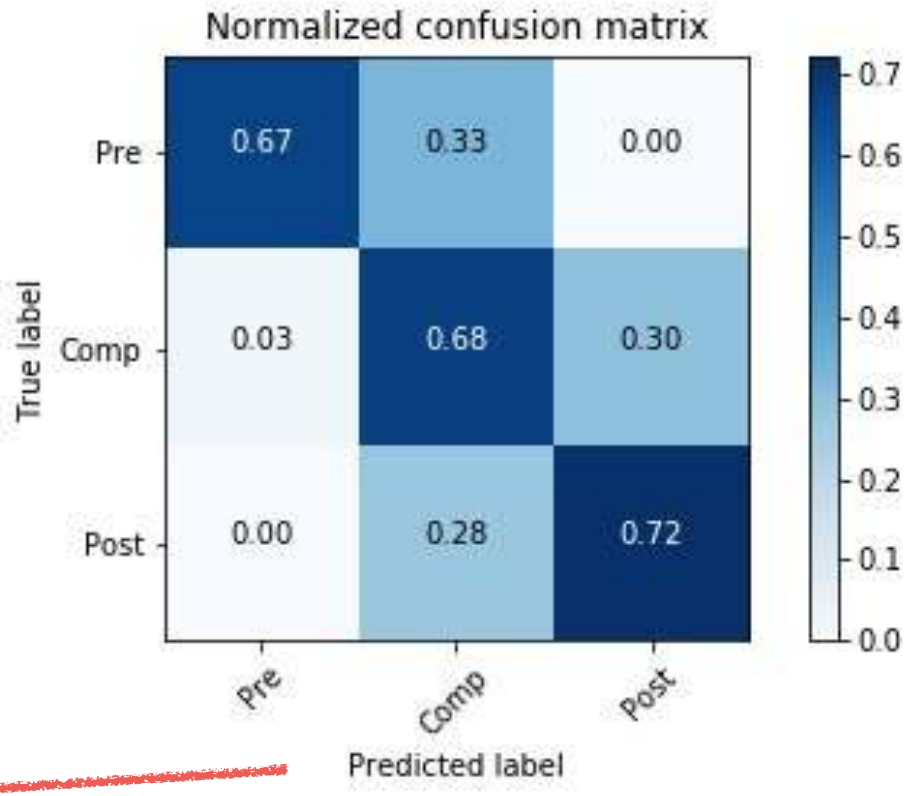
The precision is the ratio `tp / (tp + fp)` where `tp` is the number of true positives and `fp` the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

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]

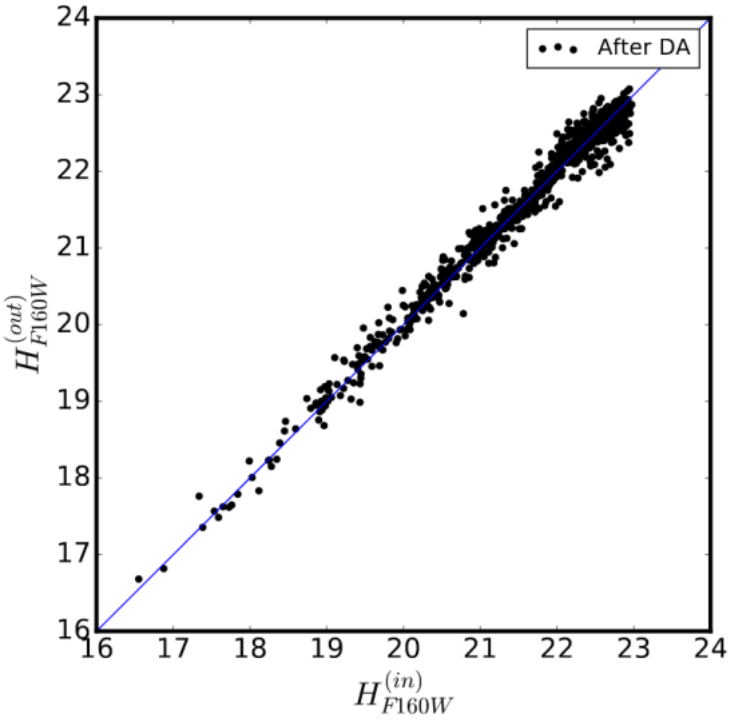
CONFUSION MATRIX



CORRESPONDENCE BETWEEN THE PREDICTED AND TRUE LABEL

Huertas-Company+18

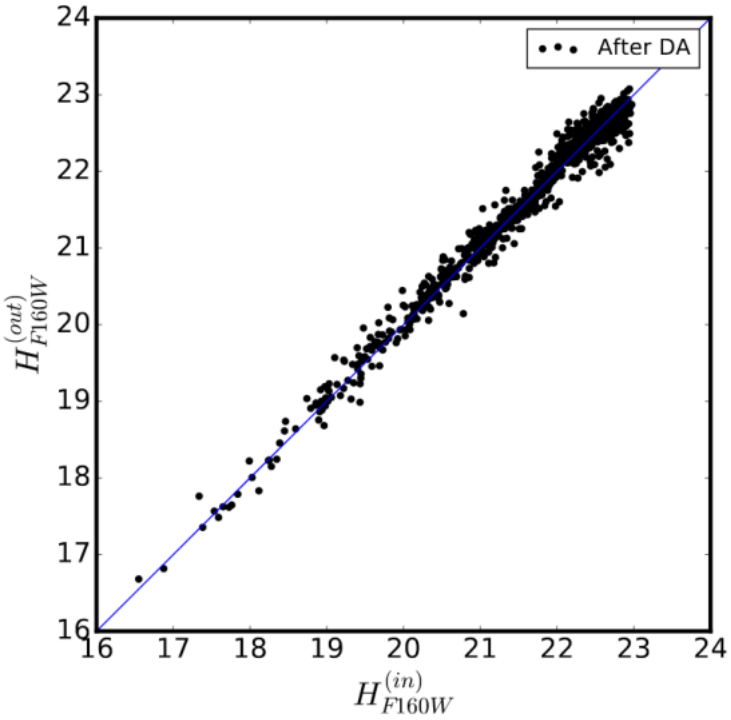
Evaluation of results [regression]



FOR REGRESSIONS, SIMPLY USE
“STANDARD ACCURACY MEASUREMENTS

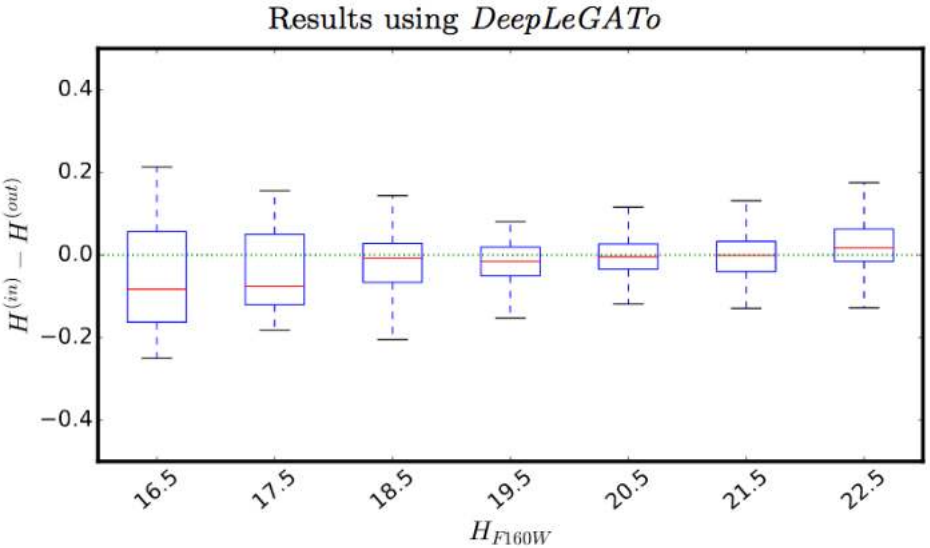
Tuccillo+18

Evaluation of results [regression]



FOR REGRESSIONS, SIMPLY USE
"STANDARD ACCURACY MEASUREMENTS

BIAS, SCATTER …YOU KNOW!



Tuccillo+18