Part 3: Random forest and Grid Search

anton.korosov@nersc.no

October 2021

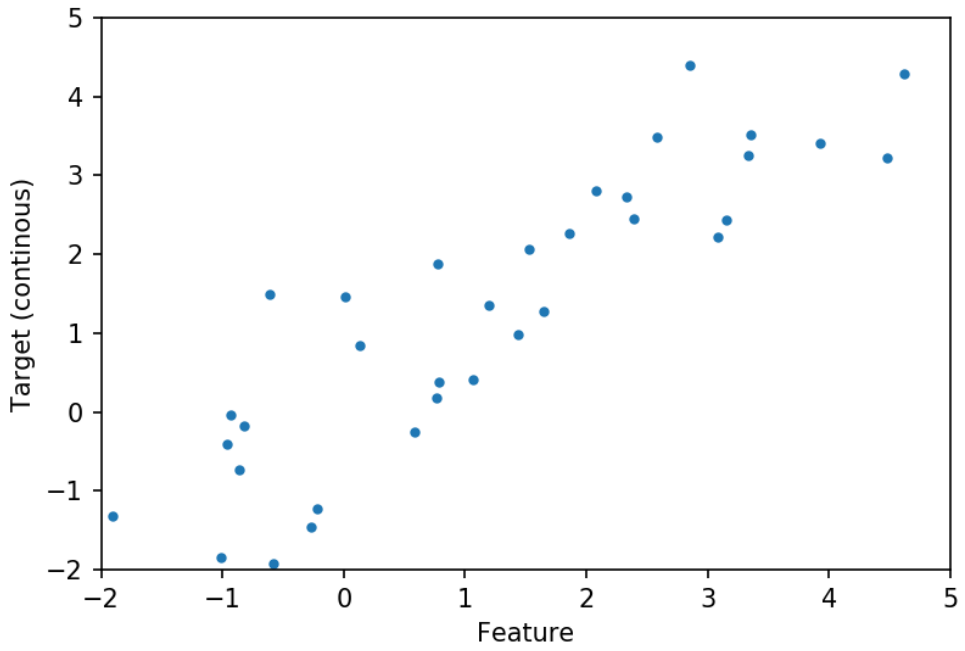
NERSC

slides+notebook:`https://github.com/nansencenter/nersc_ml_course`

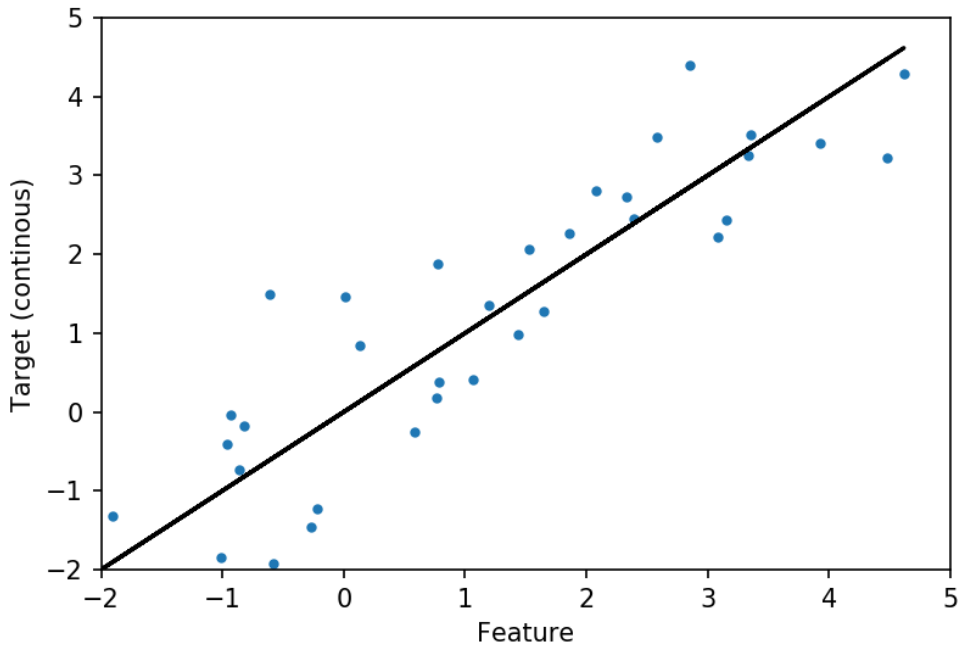
Table of contents i

Regression problem and Classification probelm

Regression



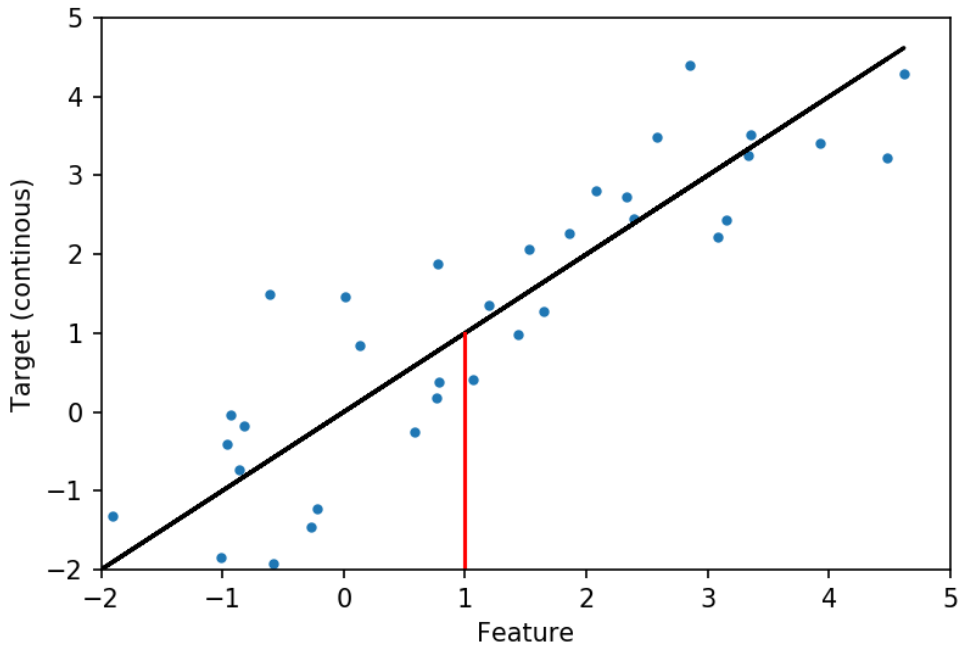
Regression



Regression: the model links feature and a continuous target:

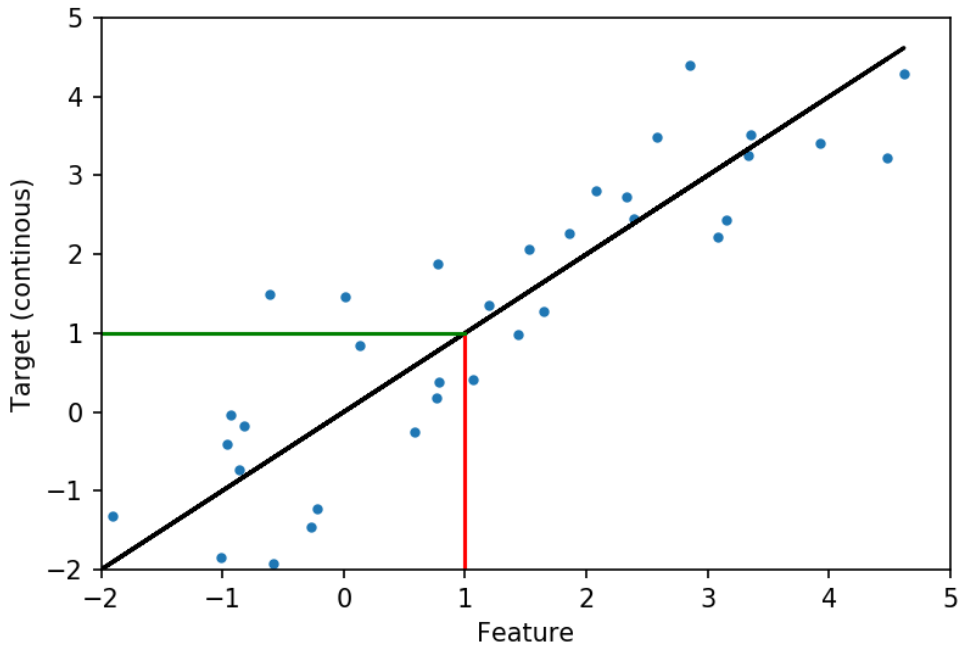
$t = M(f)$

Regression



If we have an unknown feature ($x0 = 1$):

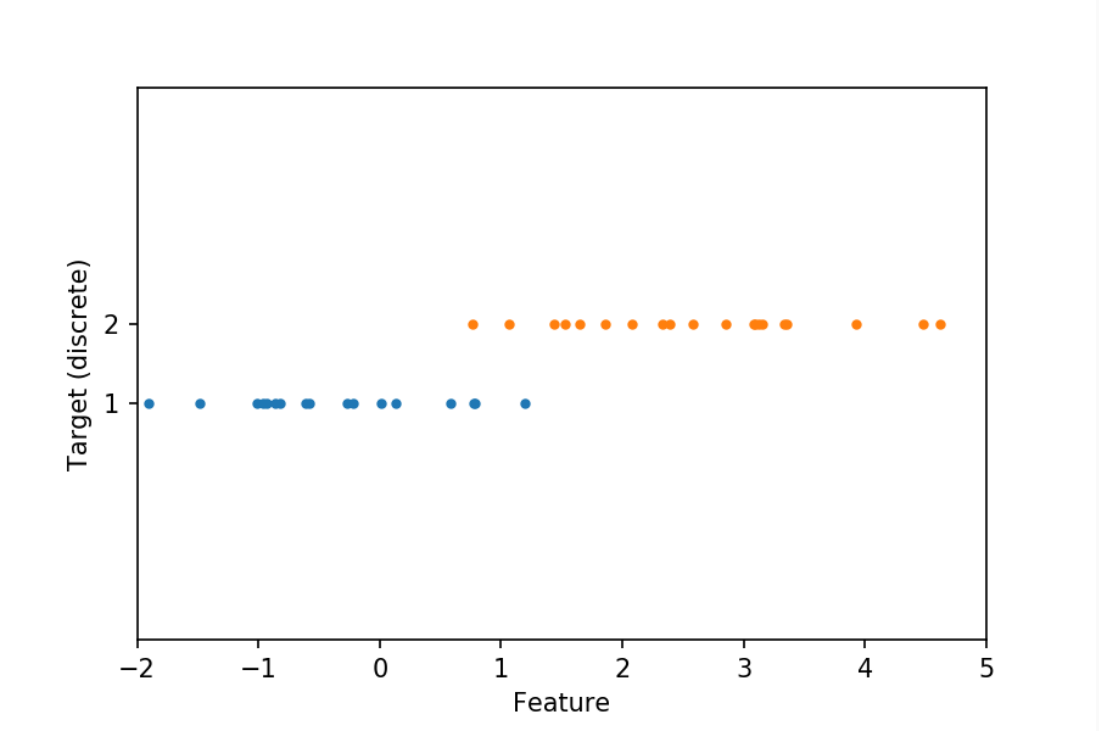
Regression



We can compute a new target:

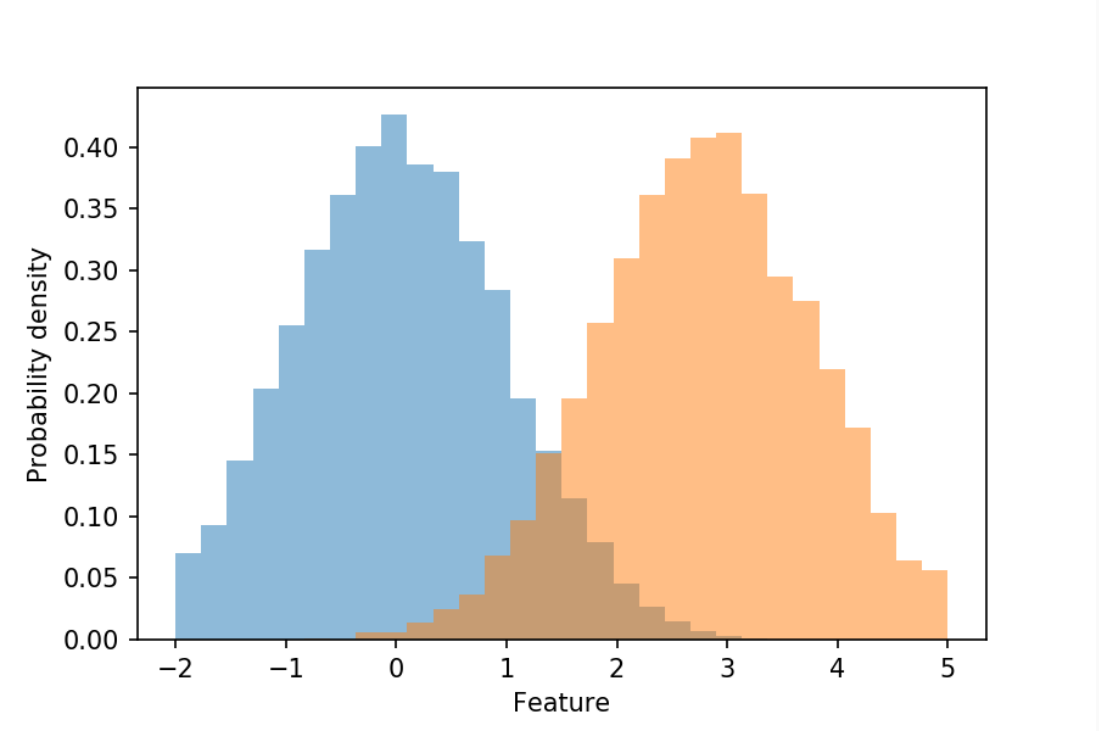
$t0 = M(x0)$

Classification



Classification: for the same feature, the target is discrete. For example, only two classes.

Classification

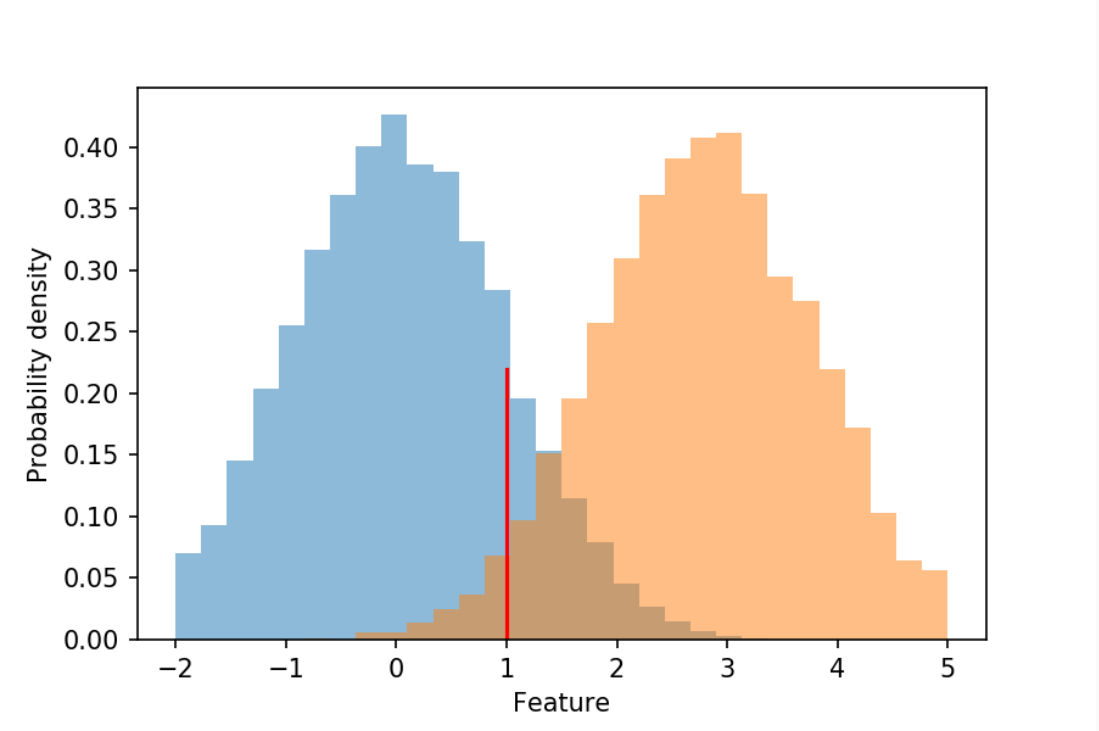


The model links feature and target through probability.

What are the probabilities of feature $f0$ to belong to classes $t0$ and $t1$?

$p_i = M(f)$

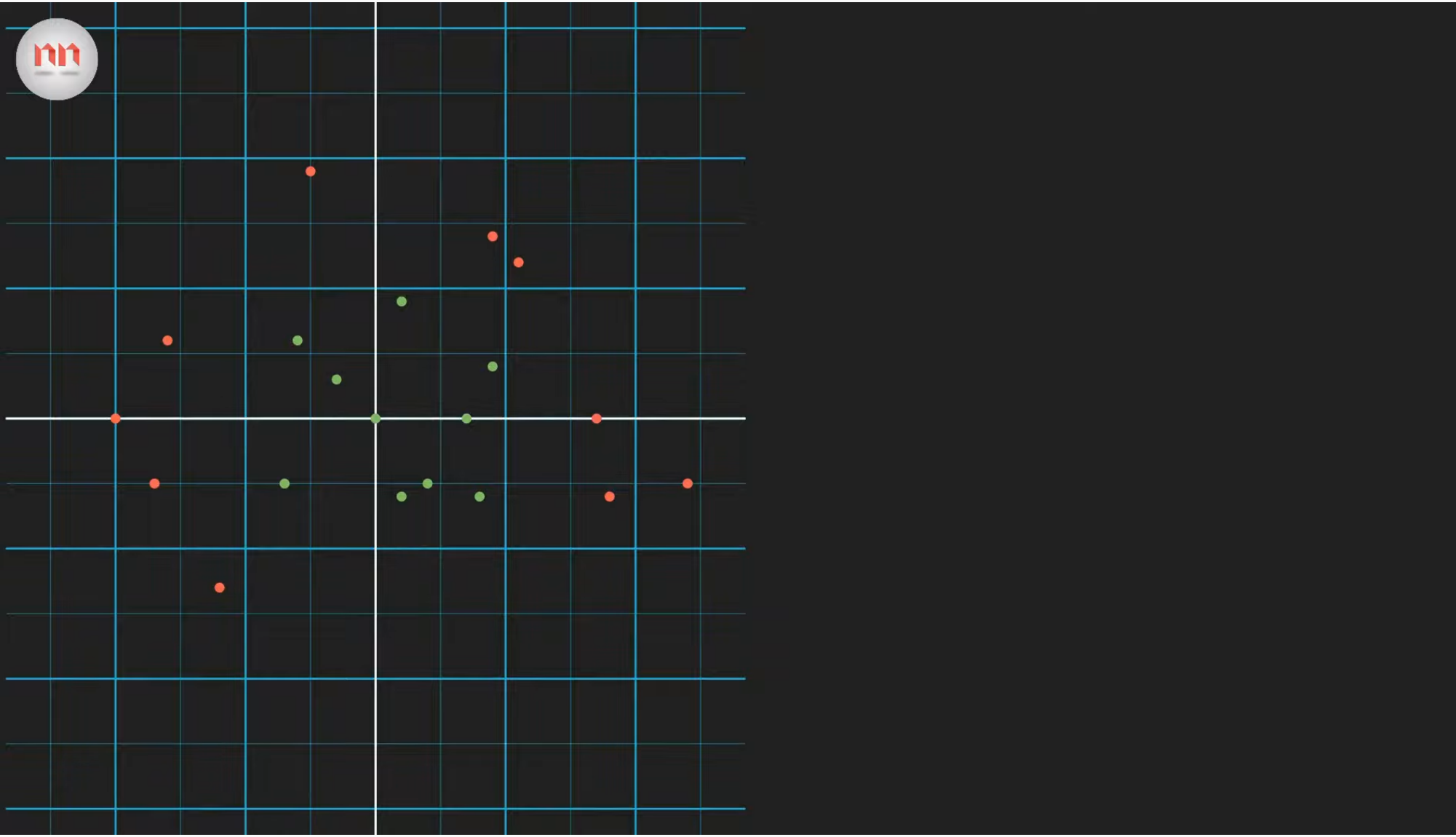
Classification



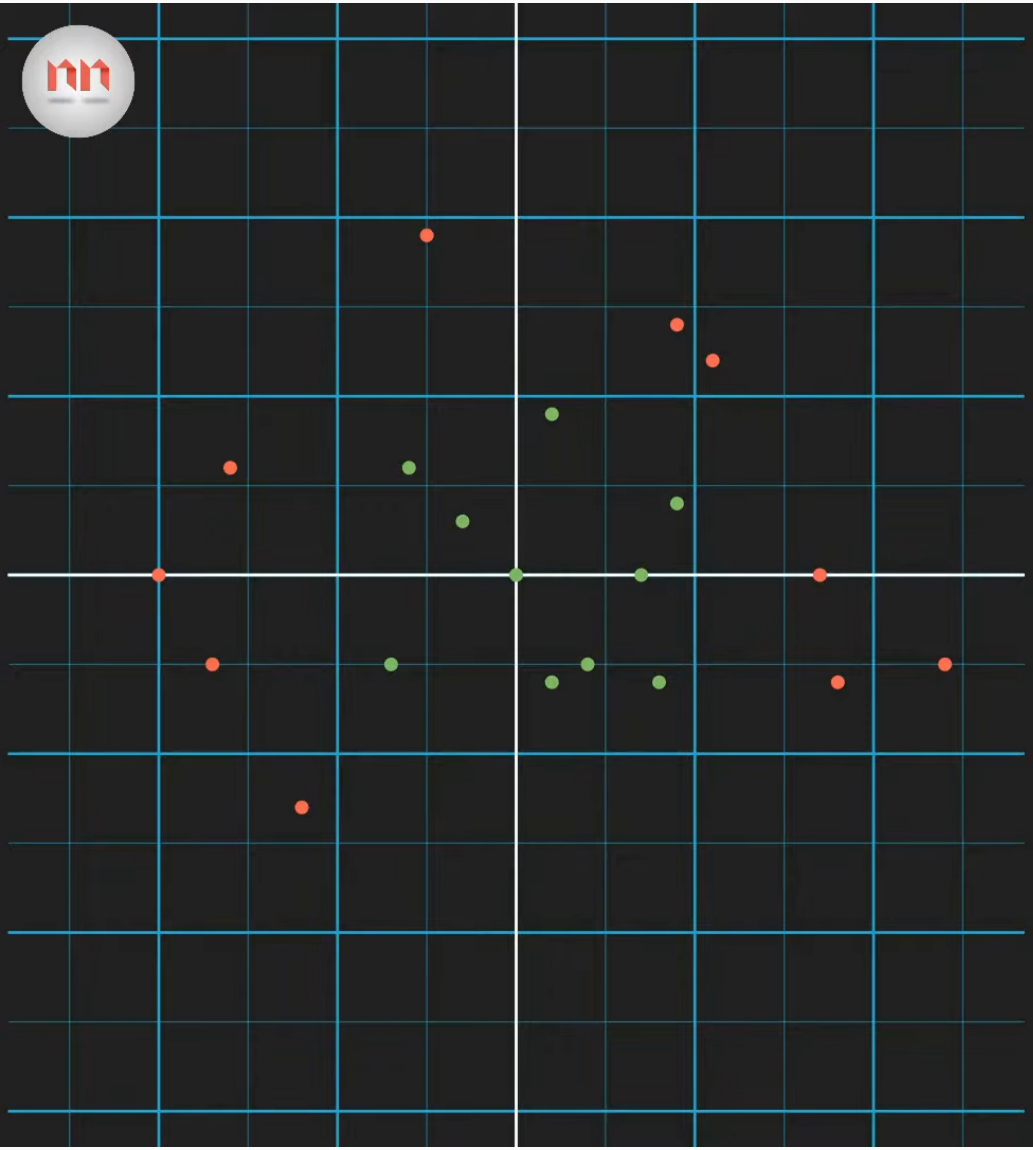
Although the PDFs overlap, the probability that point $f = 1$ belongs to class $p = 1$ is higher. Therefore this point is classified as belonging to class 1.

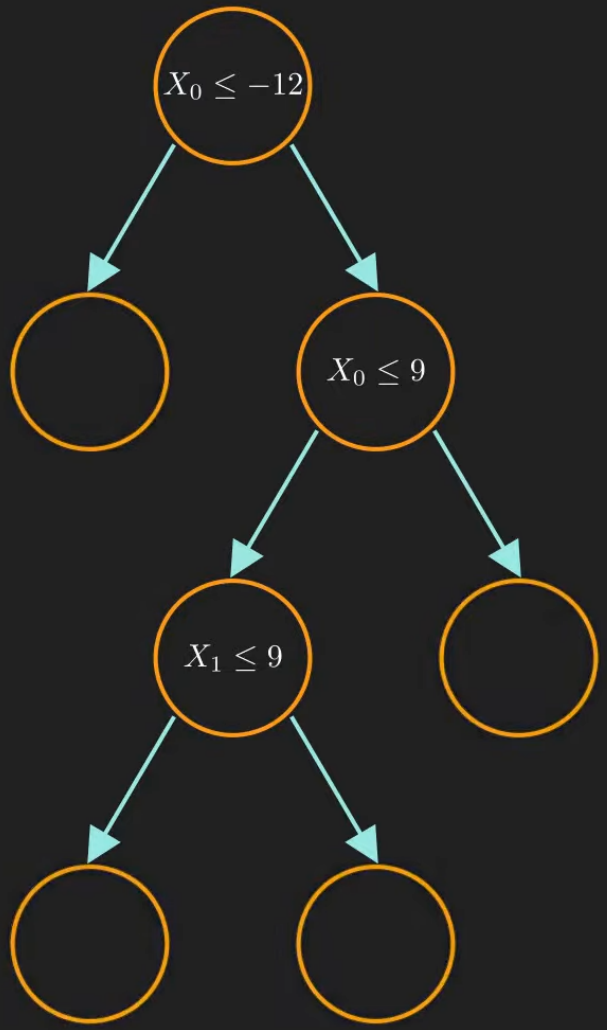
Decision trees

Dataset: 20 points, 2 features x0 and x1, 2 classes

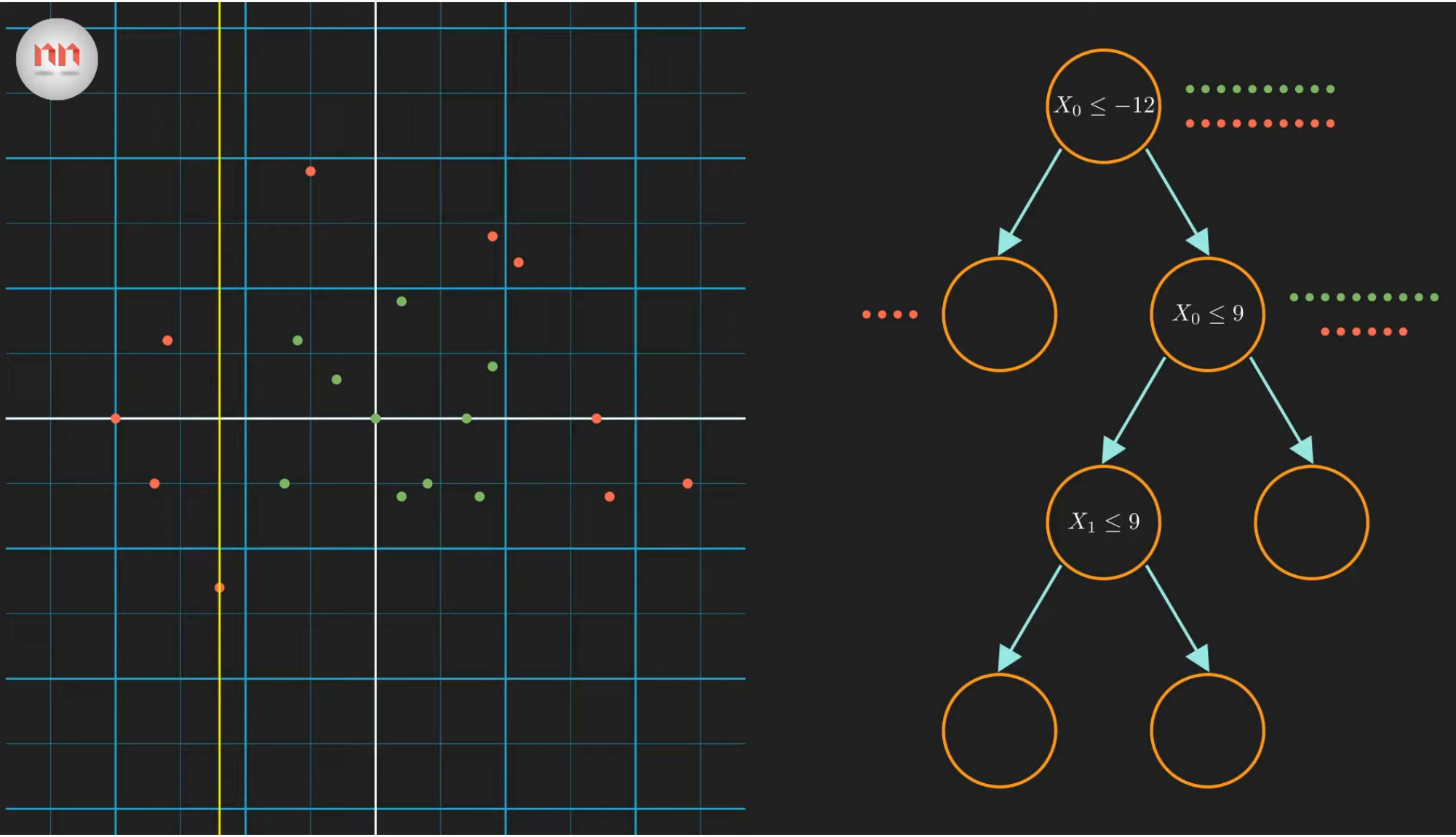


Decision tree ready for classification

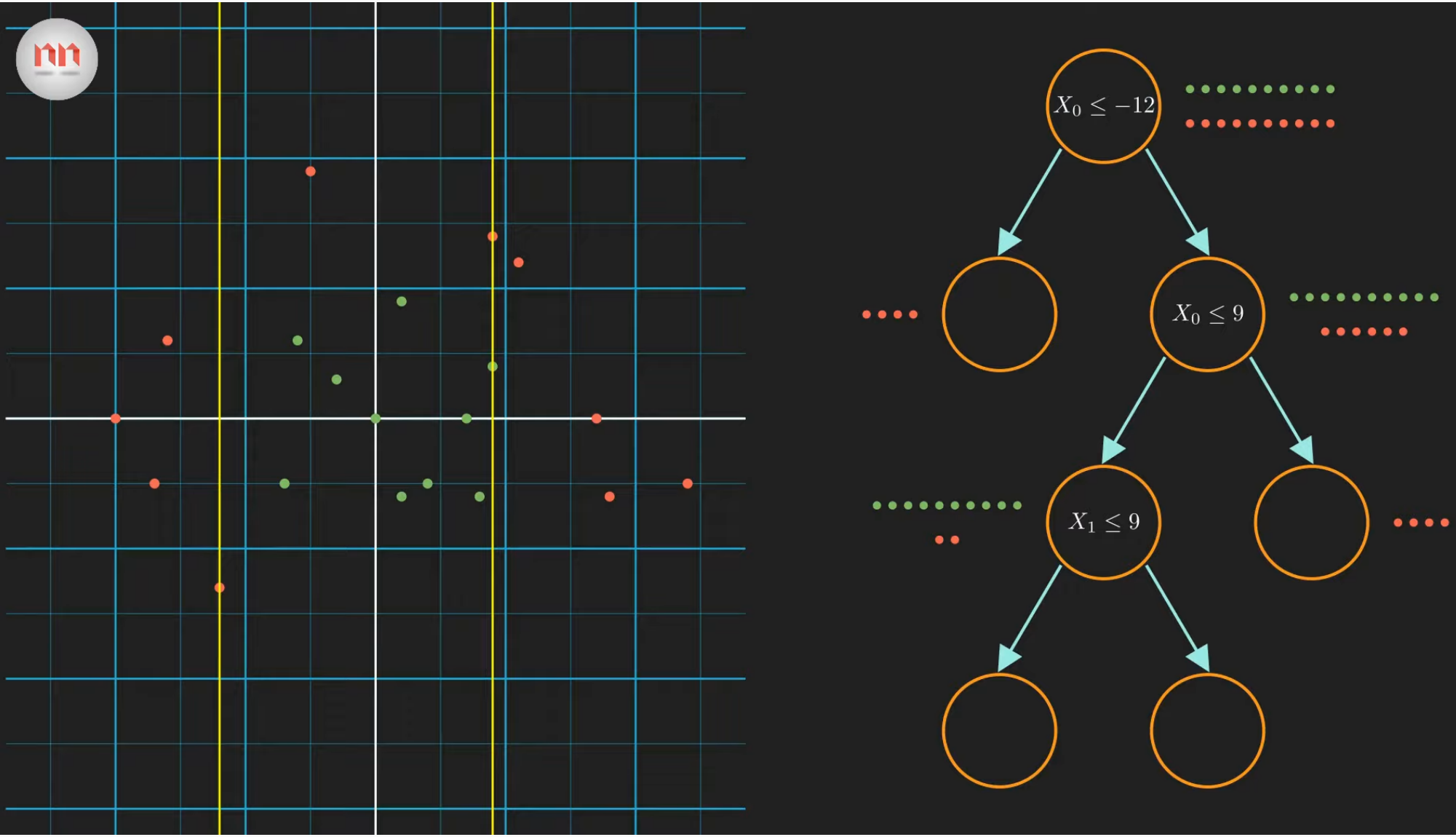




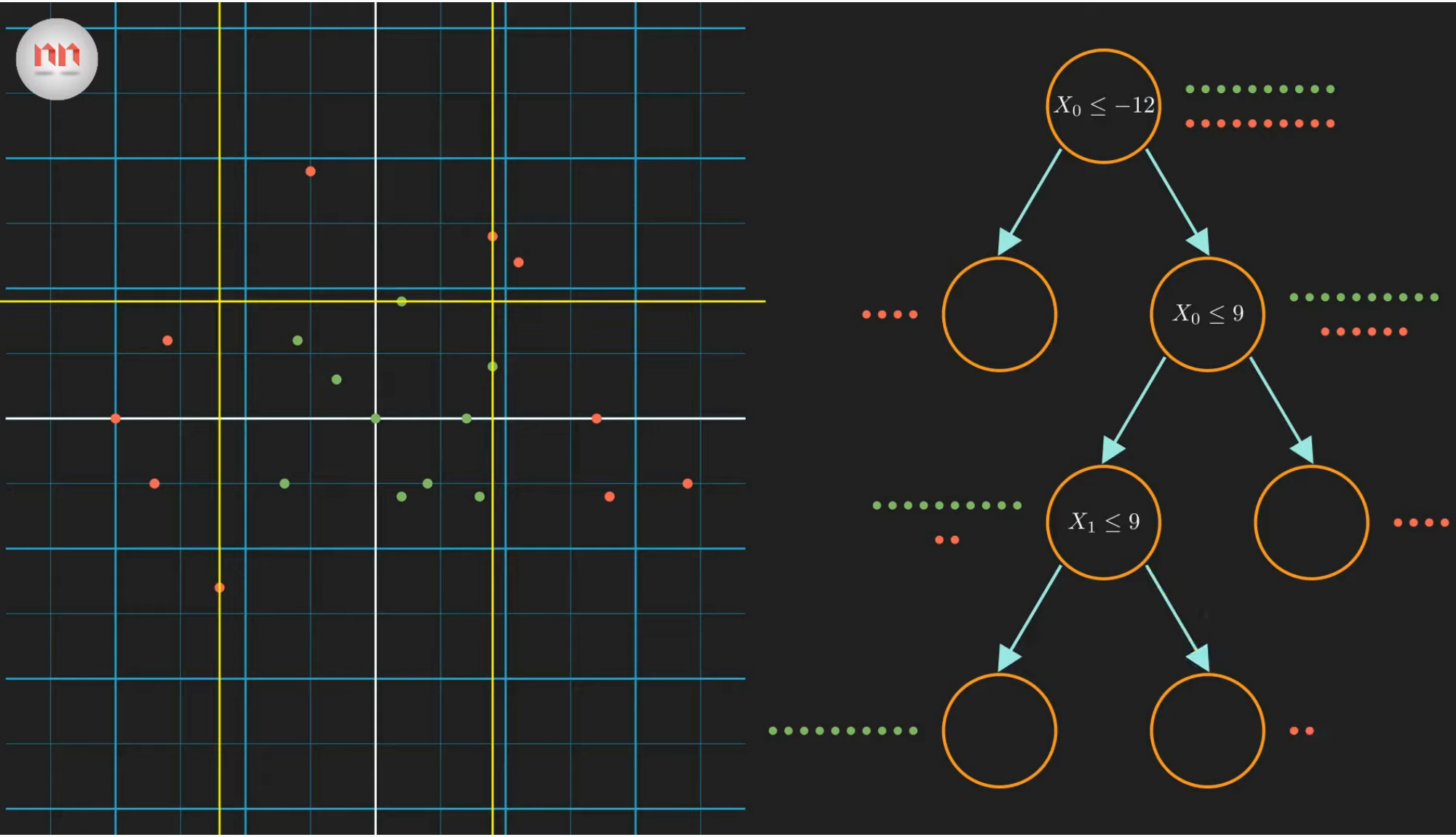
First split of points by x0



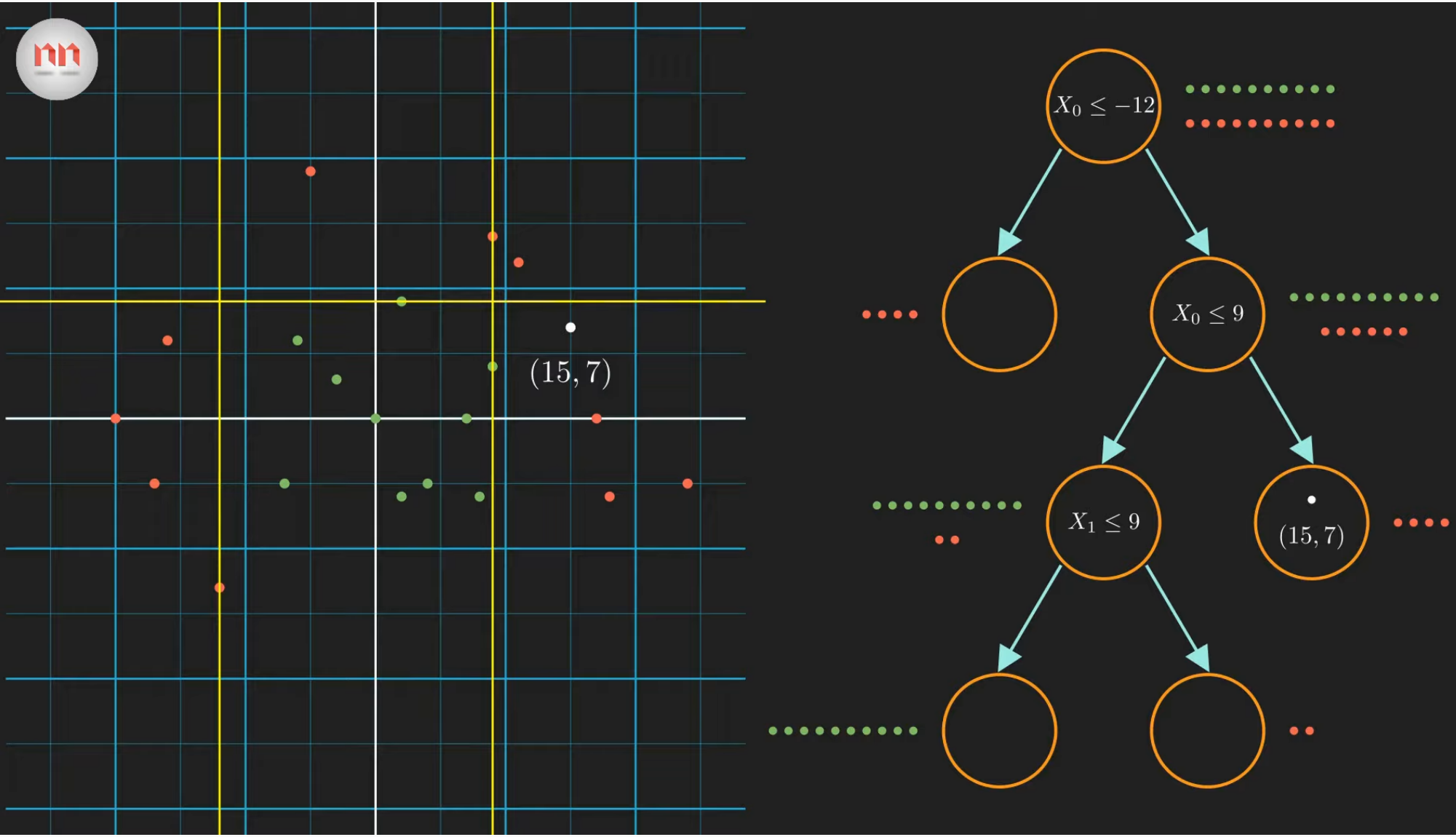
Second split of points by x0



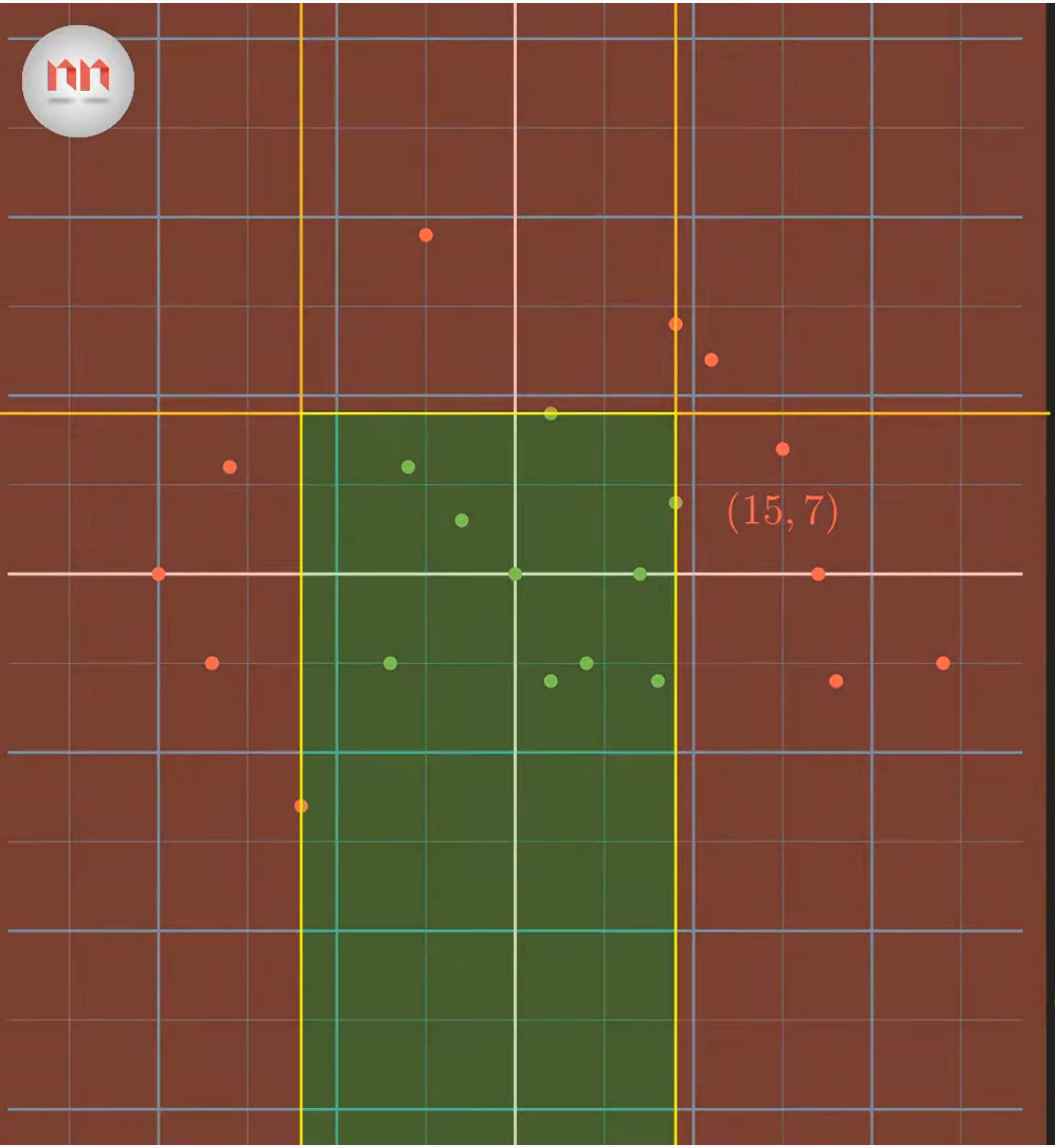
Third split of points by x1

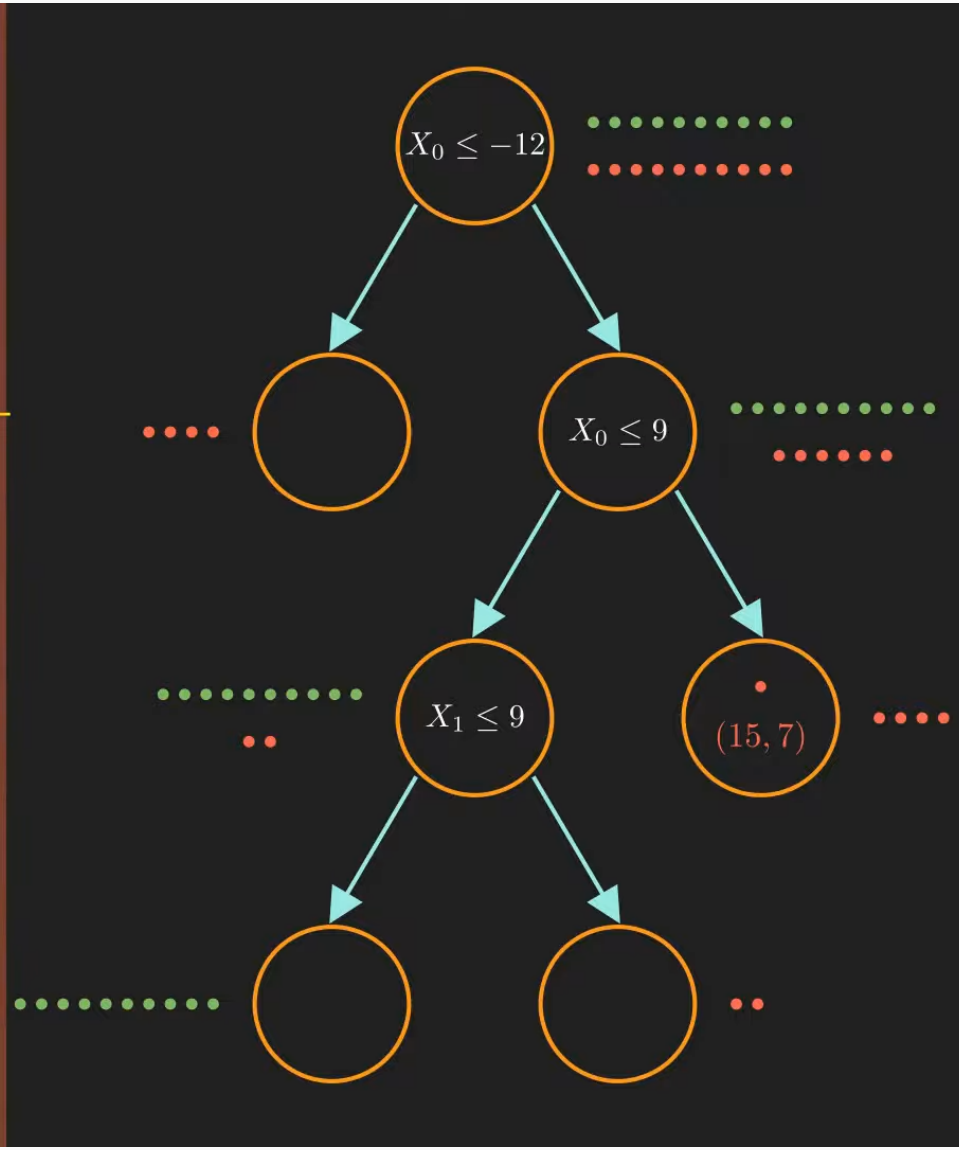


How to use the decision tree to classify point (15,7)

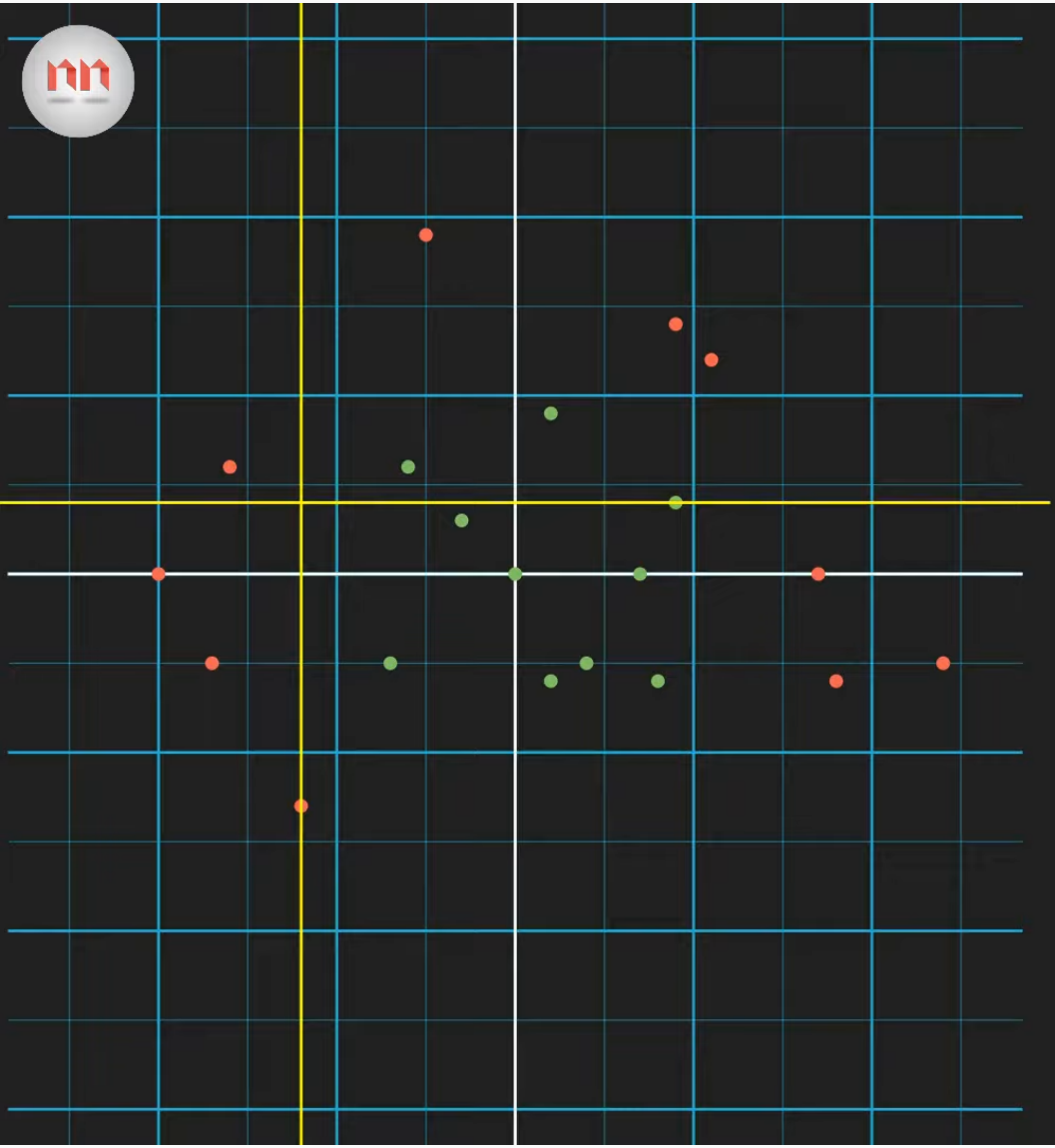


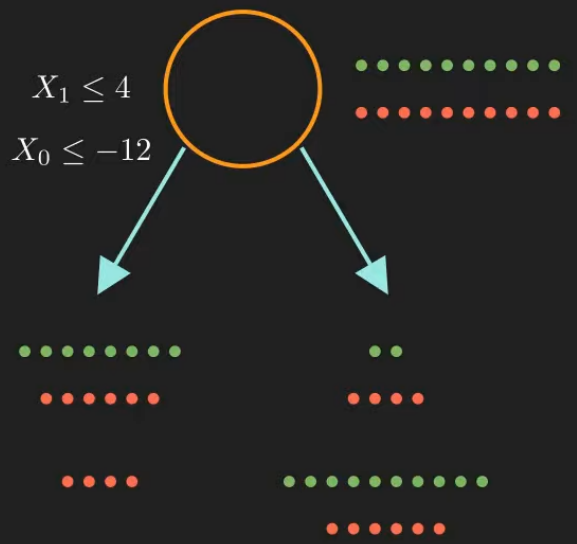
Classification space





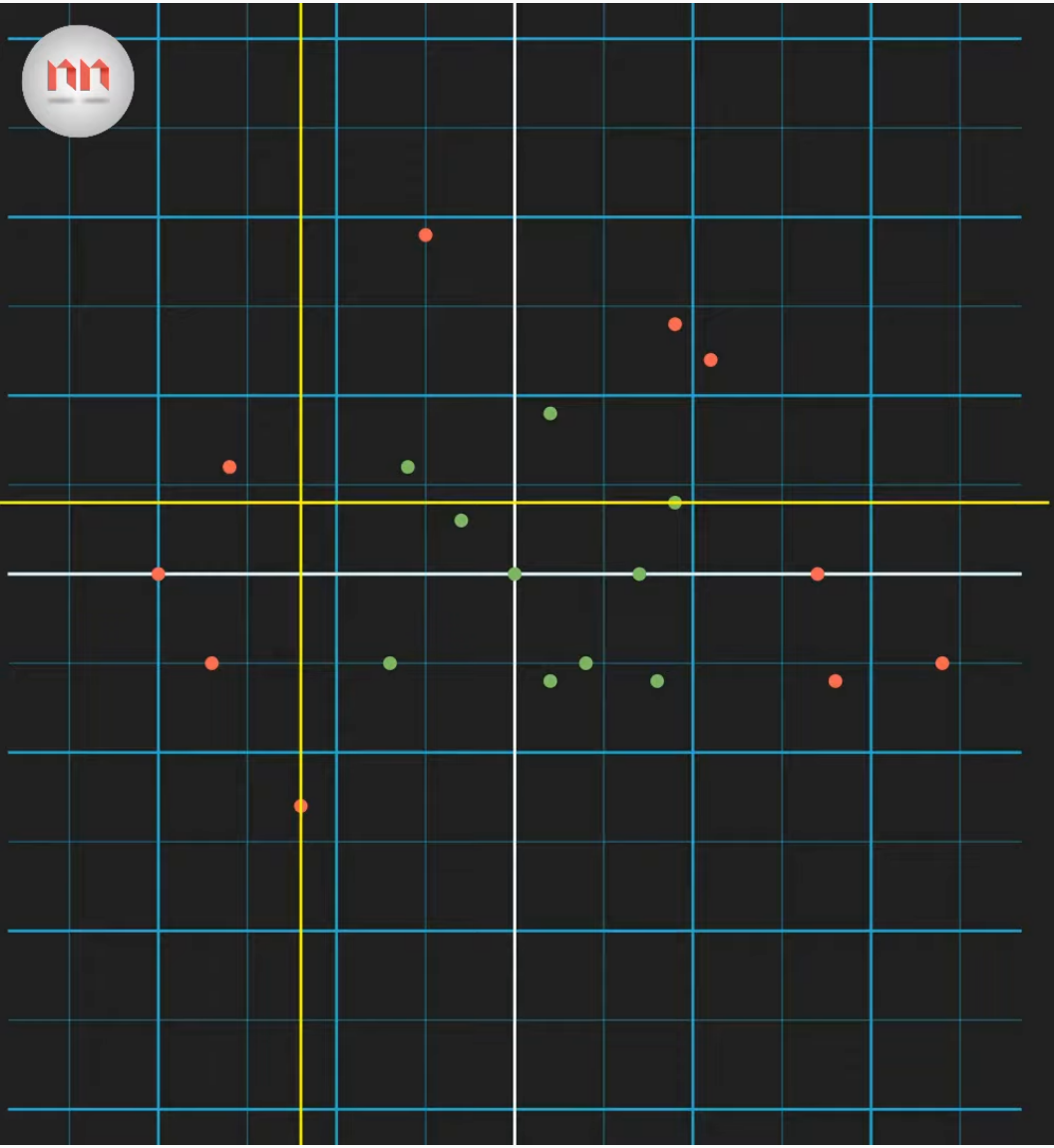
Building tree: how to split

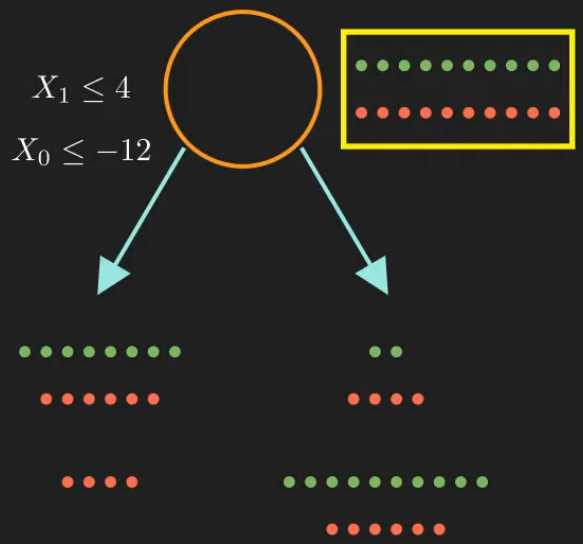




Which split is better?

Building tree: entropy - measure of informativeness

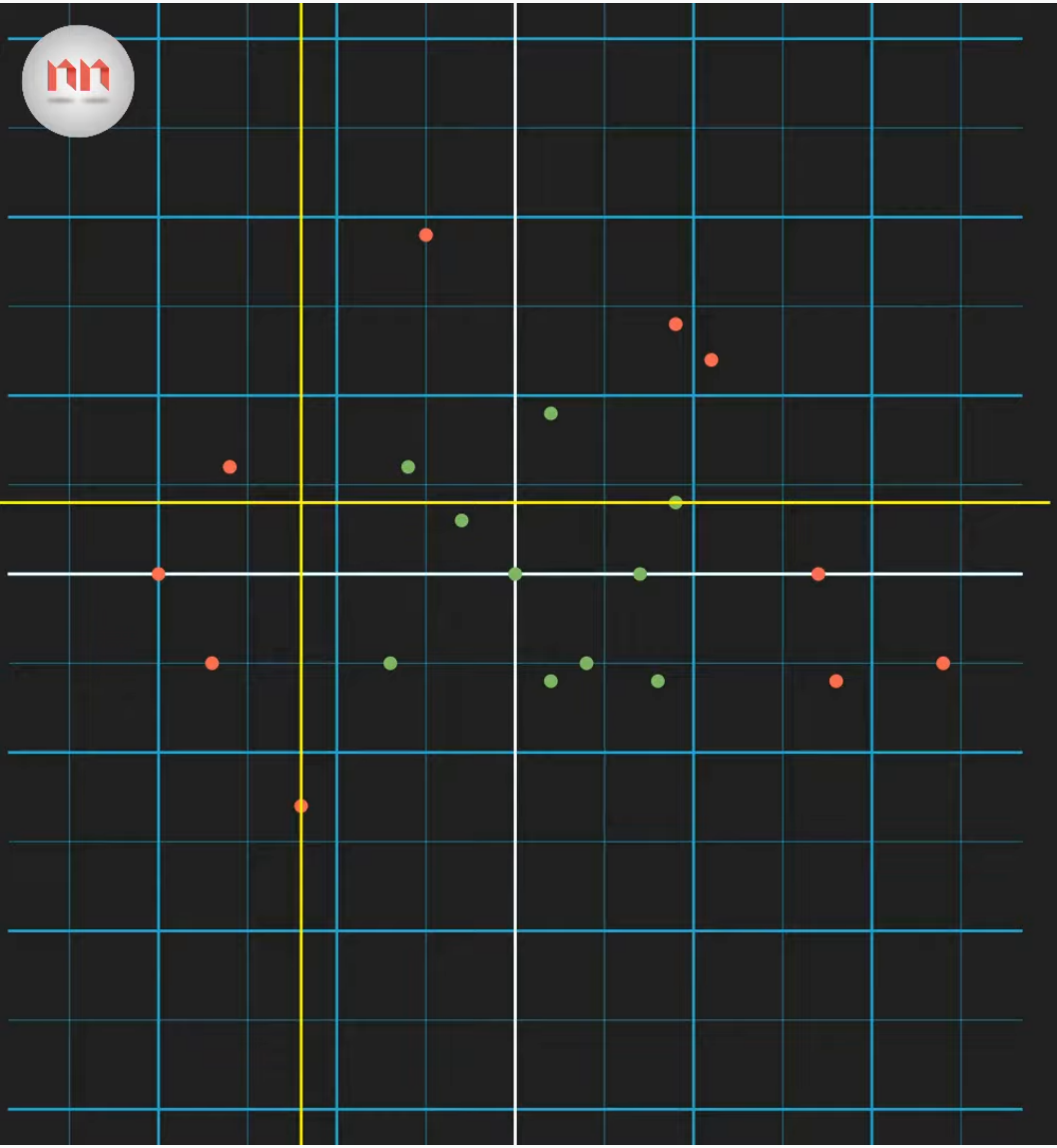


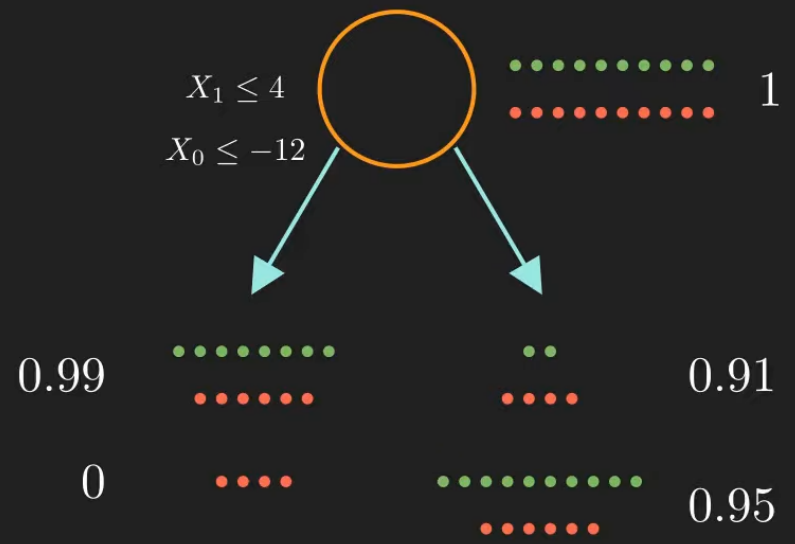


$$Entropy = -\sum p_i \log(p_i)$$

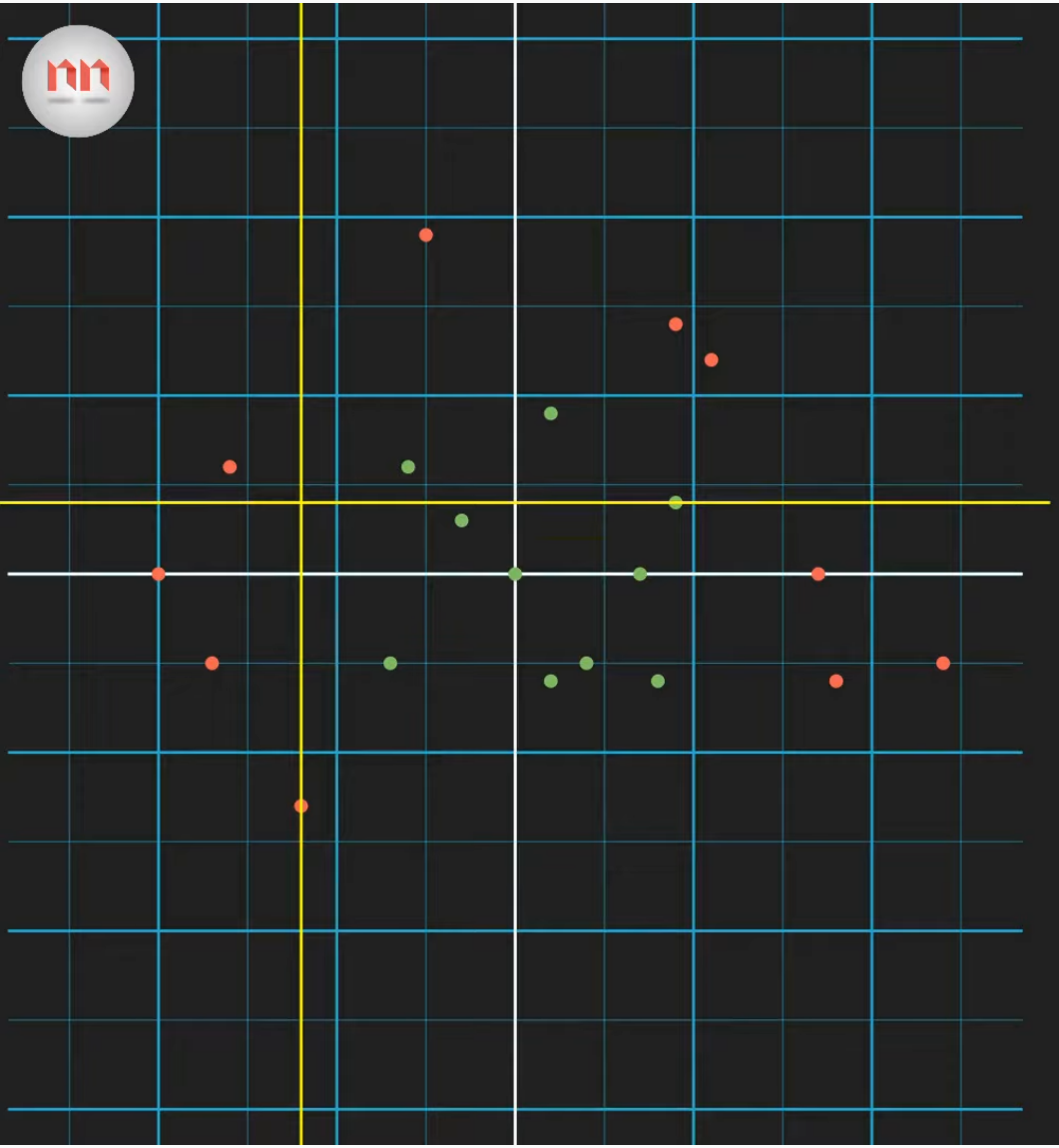
p_i = probability of class i

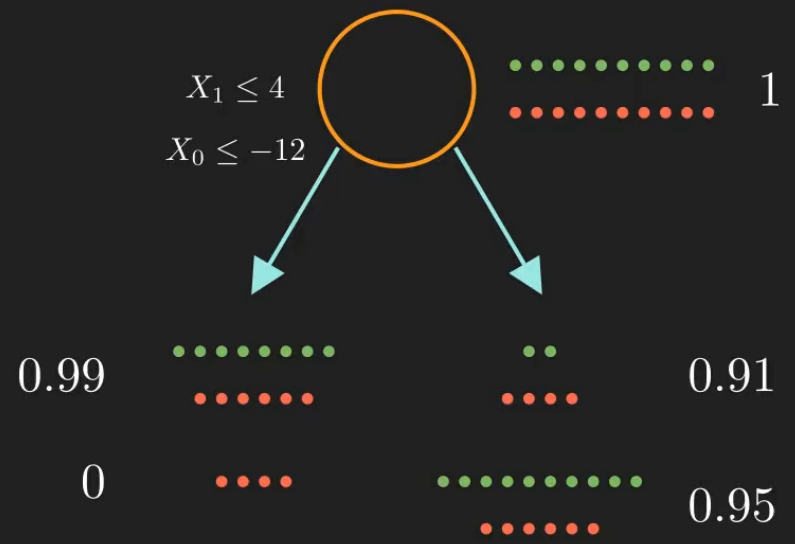
Building tree: entropy - measure of informativeness





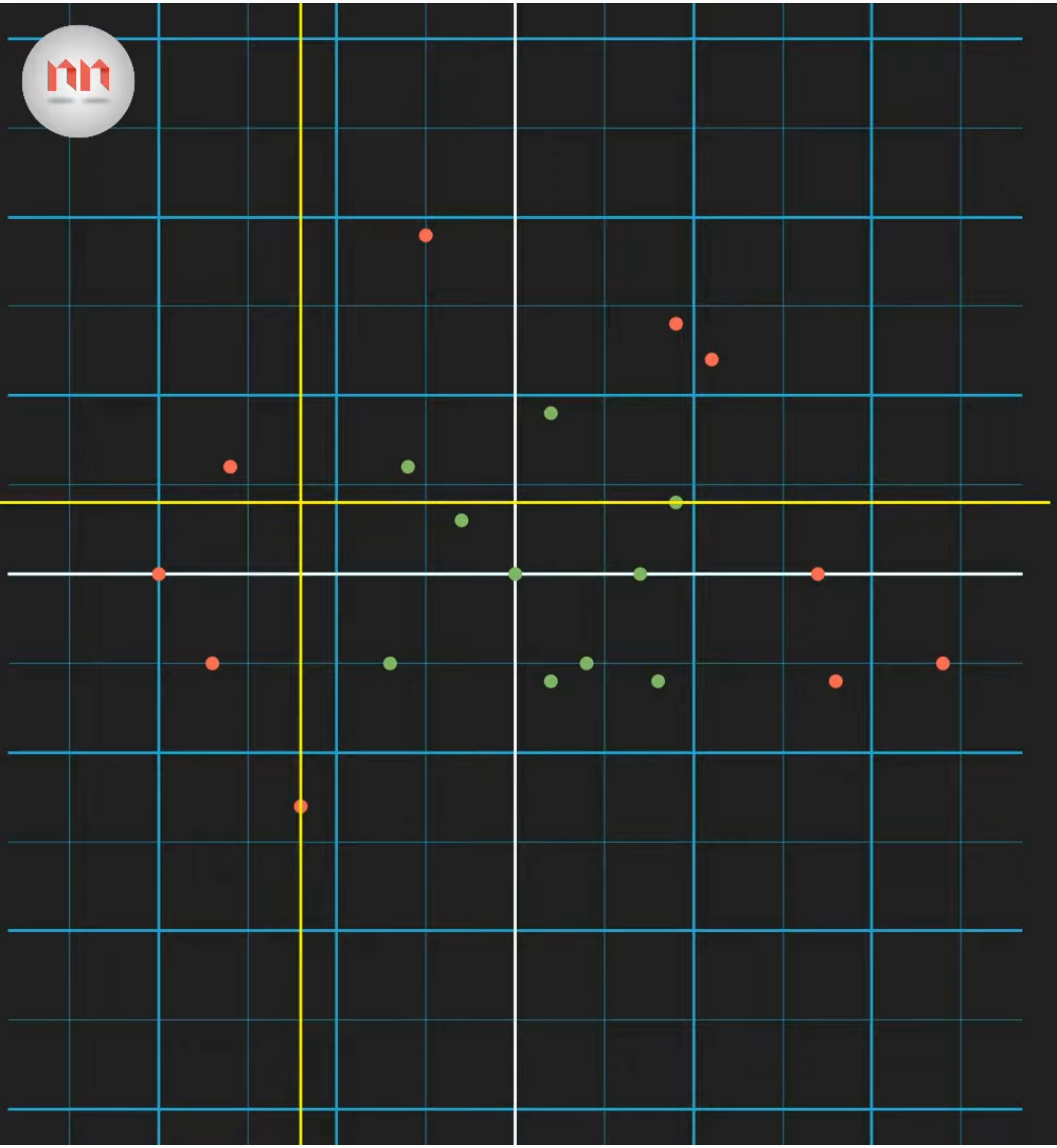
Building tree: information gain (IG)

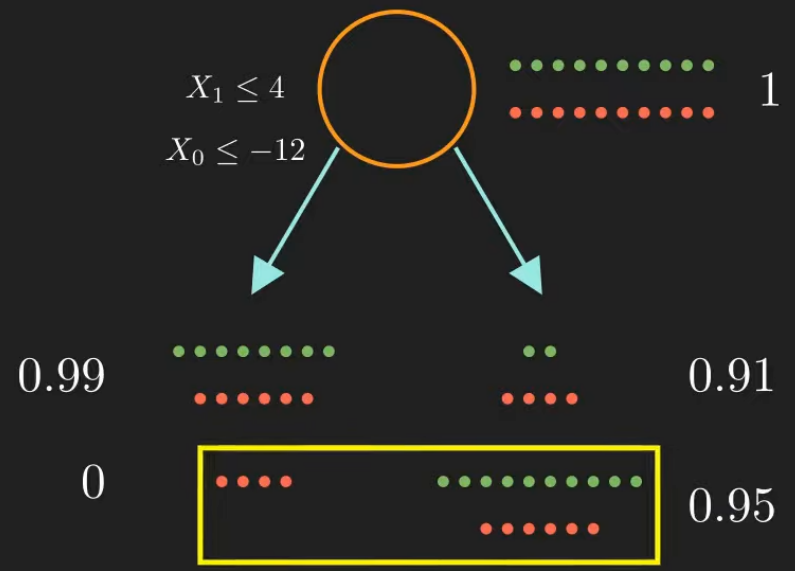




$$\text{IG} = E(parent) - \sum w_i \; E(child_i)$$

Building tree: information gain (IG)





$$\mathrm{IG}_2 > \mathrm{IG}_1$$

Summary: decision tree algorithm

Build a decision tree

- Loop over all points
  - Split the dataset by the point position
  - Compute entropy for left and right sub-datasets
  - Compute information gain (IG)
- Use the highest IG and create a decision node
- Split the dataset into left and right sub-dataset
- Recursively build the tree for the left and right sub-datasets

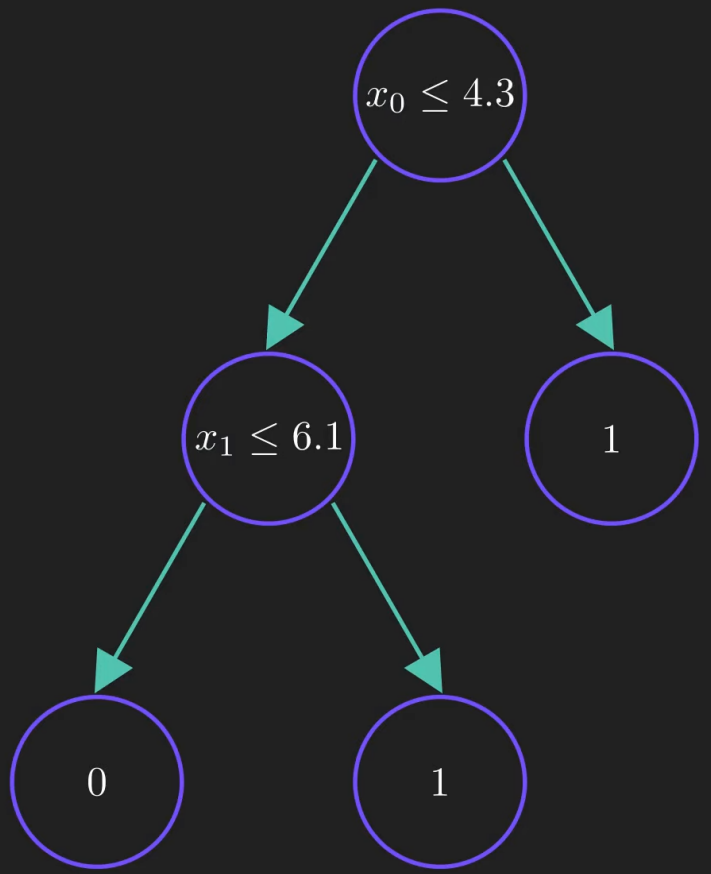
Random forest

Dataset: 6 points, 5 features, 2 classes

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

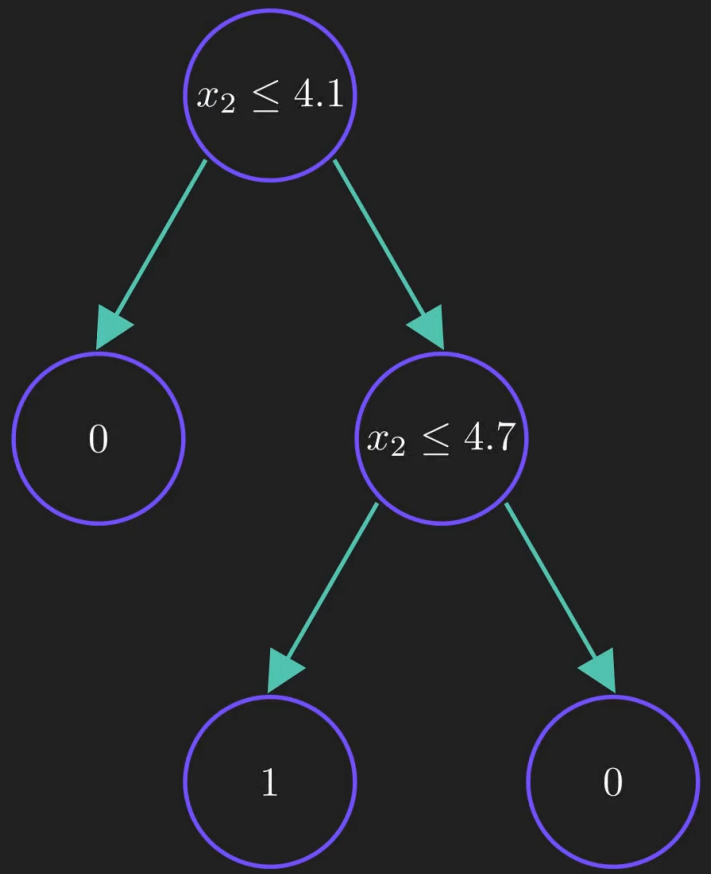
A simple working decision tree

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1



What if values slightly change?

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	6.5	4.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1



4 random datasets: take 6 random point with repetition

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

id
2
0
2
4
5
5

id
2
1
3
1
4
4

id
4
1
3
0
0
2

id
3
3
2
5
1
2

Random sampling with repetition = Bootstrapping

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

id
2
0
2
4
5
5

id
2
1
3
1
4
4

id
4
1
3
0
0
2

id
3
3
2
5
1
2

Bootstrapped Datasets

Randomly select only 2 features for each dataset

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

id
2
0
2
4
5
5

x_0, x_1

id
2
1
3
1
4
4

x_2, x_3

id
4
1
3
0
0
2

x_2, x_4

id
3
3
2
5
1
2

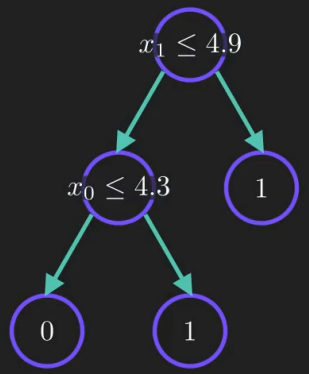
x_1, x_3

Build decision trees for each dataset

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

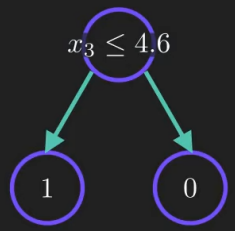
id
2
0
2
4
5
5

x_0, x_1



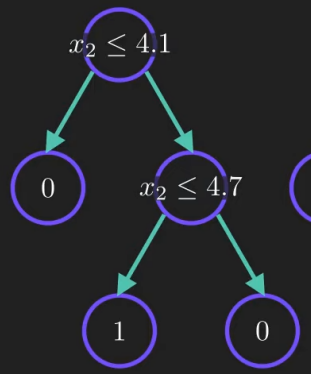
id
2
1
3
1
4
4

x_2, x_3



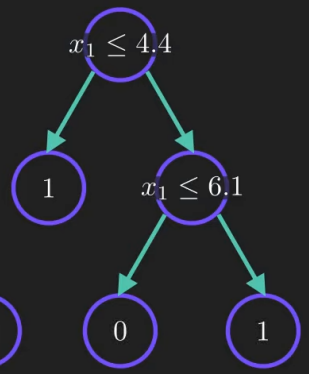
id
4
1
3
0
0
2

x_2, x_4



id
3
3
2
5
1
2

x_1, x_3



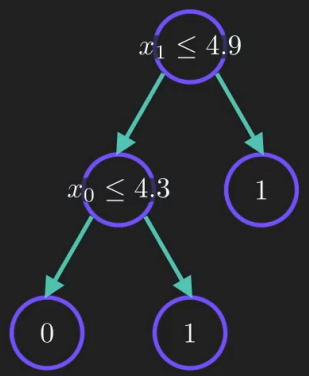
A new point

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

2.8	6.2	4.3	5.3	5.5

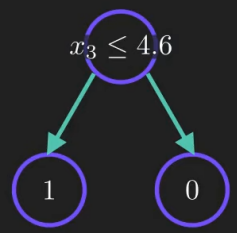
id
2
0
2
4
5
5

x_0, x_1



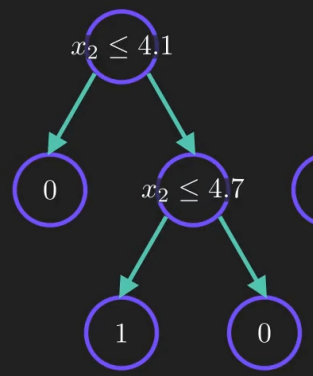
id
2
1
3
1
4
4

x_2, x_3



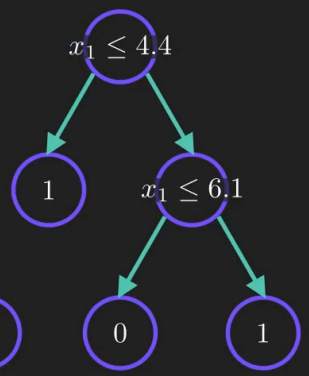
id
4
1
3
0
0
2

x_2, x_4



id
3
3
2
5
1
2

x_1, x_3



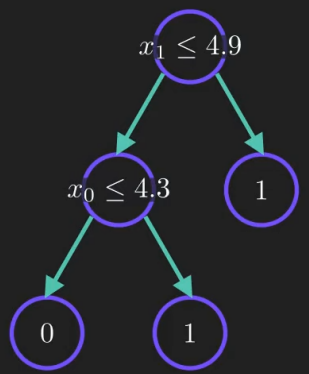
Apply the decision trees for the new point

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

2.8	6.2	4.3	5.3	5.5

id
2
0
2
4
5
5

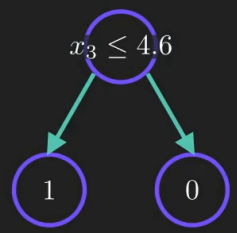
x_0, x_1



1

id
2
1
3
1
4
4

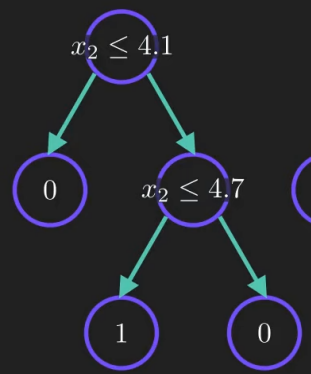
x_2, x_3



0

id
4
1
3
0
0
2

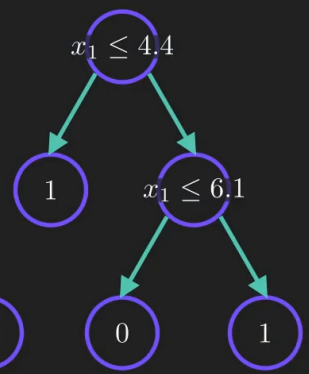
x_2, x_4



1

id
3
3
2
5
1
2

x_1, x_3



1

Aggregate answers

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

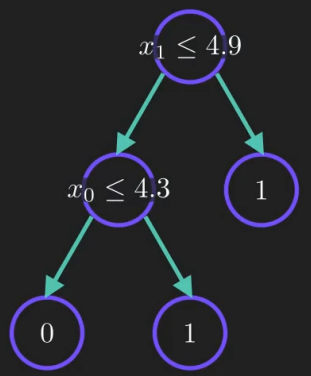
2.8	6.2	4.3	5.3	5.5

Bootstrap + Aggregating

(Bagging)

id
2
0
2
4
5
5

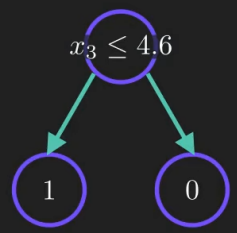
x_0, x_1



1

id
2
1
3
1
4
4

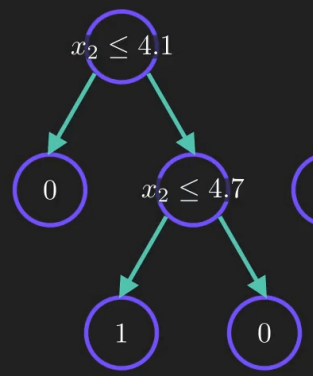
x_2, x_3



0

id
4
1
3
0
0
2

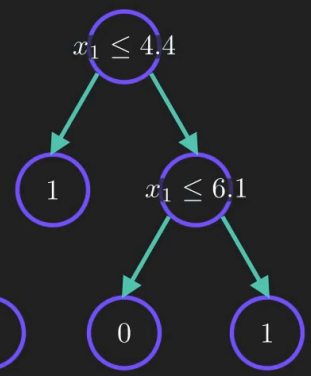
x_2, x_4



1

id
3
3
2
5
1
2

x_1, x_3



1

Summary: random forest algorithm algorithm

Build a random forest

- Bootstrap the dataset
  - Take n random points from the dataset with repetition
  - Take m random features
  - Create k random datasets
- Train K decision trees

Summary: random forest algorithm algorithm

Build a random forest

- Bootstrap the dataset
  - Take n random points from the dataset with repetition
  - Take m random features
  - Create k random datasets
- Train K decision trees

Apply the random forest

- Traverse the original dataset through K decision trees using m features
- Aggregate results from each decision tree (average or maximum probability)

Summary: random forest algorithm algorithm

Build a rand

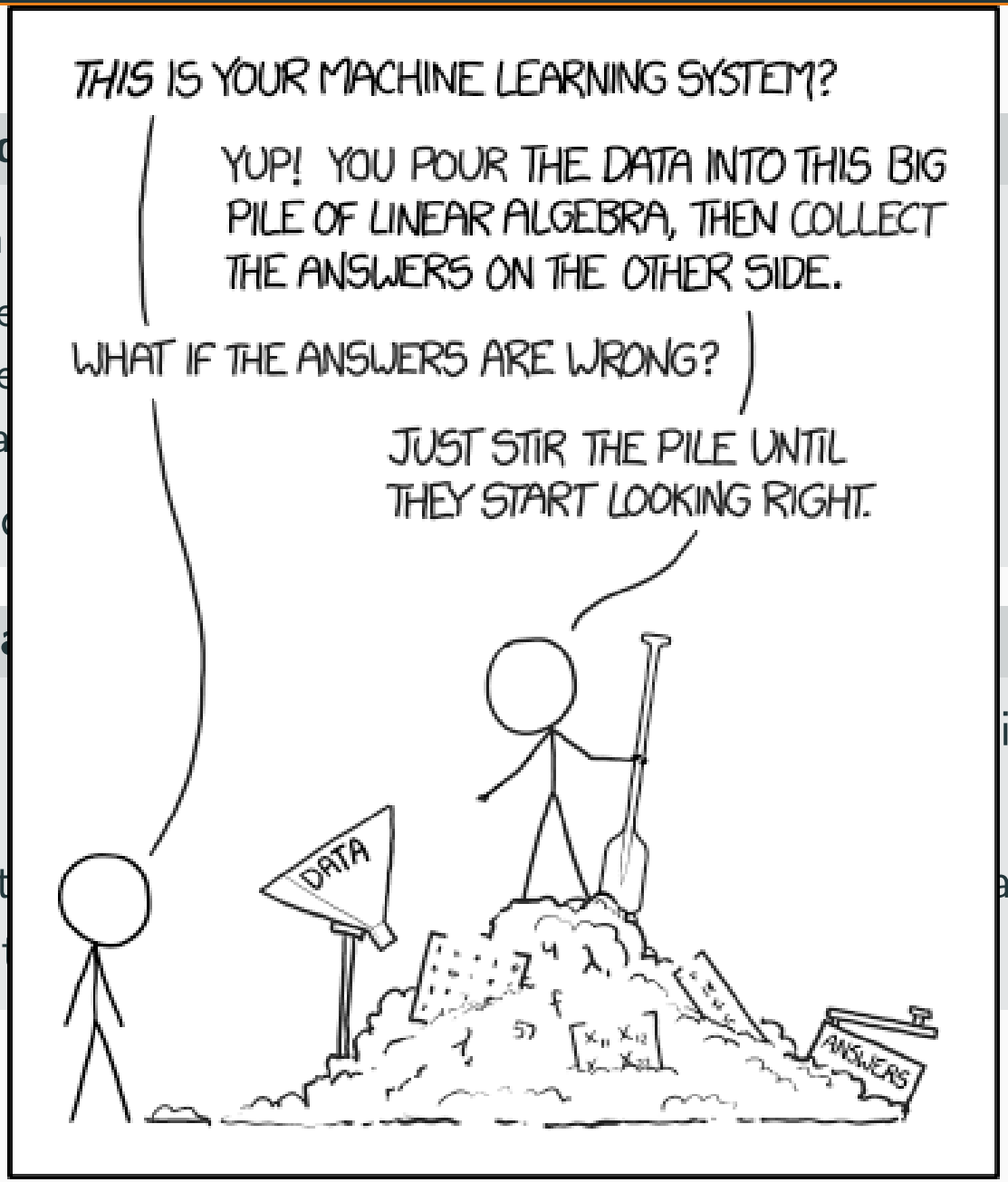
- Bootstra
  - Take
  - Take
  - Crea
- Train K

Apply the ra

- Traverse
  features
- Aggregat
  probabili

ing m

aximum



Grid search

Parameters and hyperparameters

- Parameters in the model that are optimized during training are called parameters (e.g., the criteria on which we split the dataset into left and right sub-datasets)

Parameters and hyperparameters

- Parameters in the model that are optimized during training are called parameters (e.g., the criteria on which we split the dataset into left and right sub-datasets)
- Parameters that are not optimized during the training are called hyperparameters (e.g., k - number of decision trees in a random forest; m - number of random features in each tree)

Parameters and hyperparameters

- Parameters in the model that are optimized during training are called parameters (e.g., the criteria on which we split the dataset into left and right sub-datasets)
- Parameters that are not optimized during the training are called hyperparameters (e.g., k - number of decision trees in a random forest; m - number of random features in each tree)
- Hyperparameters can be determined using a score on the validation dataset or using a cross-validation procedure

The grid search

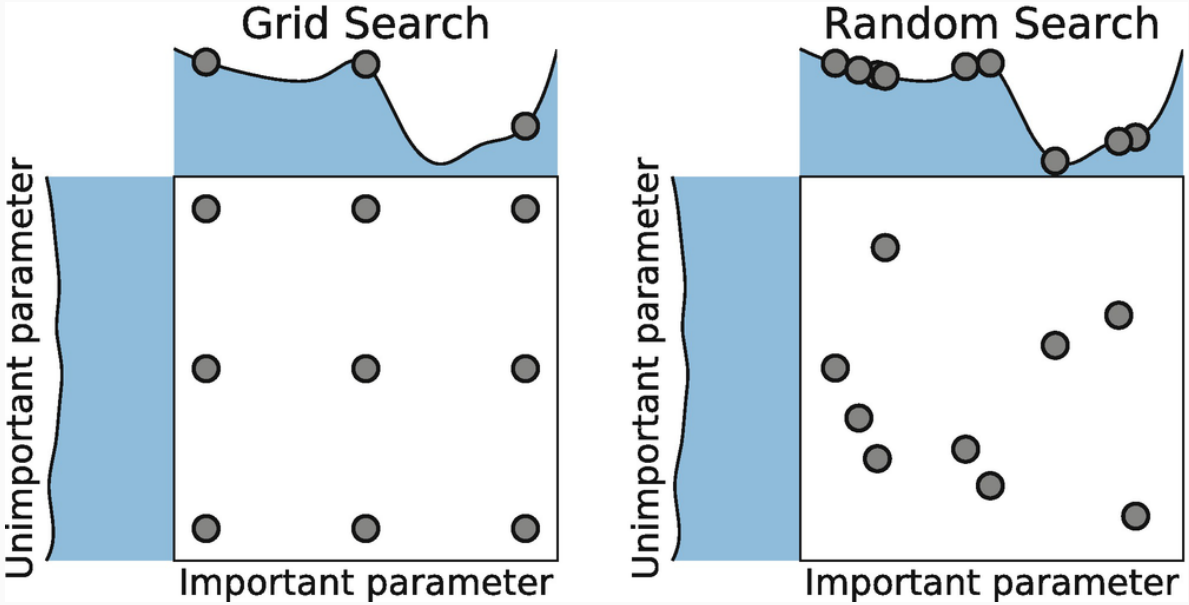
1. Split the dataset into training and validation sub-datasets (e.g., with ratio 4:1)
2. Specify a list of hyperparameters to be tested
3. For each of the parameters, specify a set of values to test
4. Train a model on the training sub-dataset for each of the possible combinations of hyperparameters
5. Validate the model on the validation sub-dataset
6. Retain the best model

Remarks on the grid search procedure

- It make an exhaustive search of the hyperparameters
- The procedure is easy to parallelize
- It is not naturally adapted for quantitative hyperparameters (not continuous).
- It can become very costly. (e.g. 8 hyperparameters with 8 values each to test: $8^8 = 16, 777, 216$ trainings.

The random search

1. Specify a list of hyperparameters to be tested
2. For each of the parameters, specify a set of values to test or a law to draw a random value
3. Draw n combinations of the hyperparameters
4. Train a model for each of the combinations and validate it on the validation sub-dataset
5. Retain the best model



Remarks on the random search procedure

- It does not make an exhaustive search of the hyperparameters
- The procedure is easy to parallelize.
- The cost is predictable (number of draw).

Remarks on the random search procedure

- It does not make an exhaustive search of the hyperparameters
- The procedure is easy to parallelize.
- The cost is predictable (number of draw).

Both grid search and random search are implemented and easy to use in scikit-learn.

Data for the practical exercise

AMSR2 passive microwave radiometer data

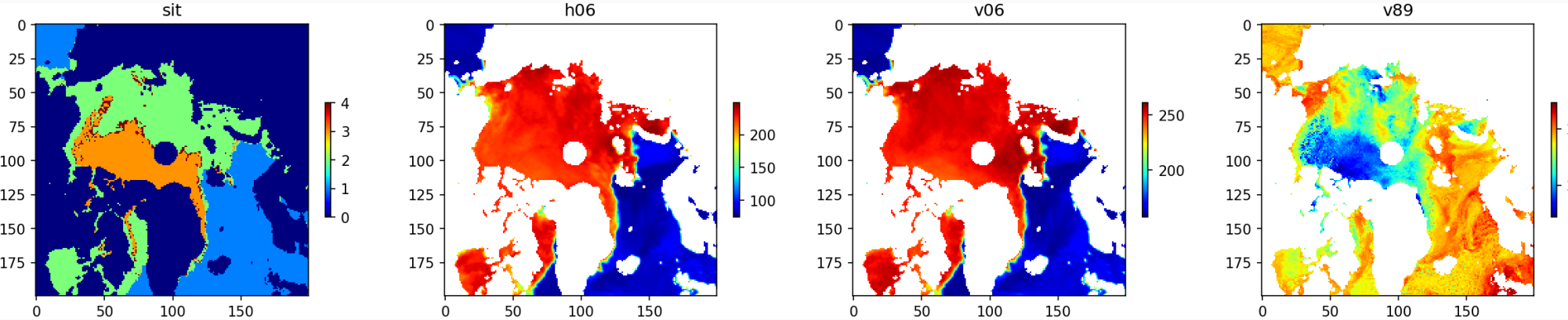
- Advanced Microwave Scanning Radiometer 2 (AMSR2)
- Measures brightness temperature of the Earth surface at 7 frequencies and in 2 polarisations in Kelvins
- Fourteen 2D-arrays covering the Arctic (14 x 200 x 200 pixels)
- A subset from the original image with reduced resolution and coverage

Sea ice type data

- Sea ice type product of the EUMETSAT OSI-SAF
- 3+1 type: Open water, First-year ice, Multi-year ice, Uncertain type
- Only one 2D-array covering the Arctic (200 x 200 pixels)

Sea ice type data

- Sea ice type product of the EUMETSAT OSI-SAF
- 3+1 type: Open water, First-year ice, Multi-year ice, Uncertain type
- Only one 2D-array covering the Arctic (200 x 200 pixels)



Aaboe, S., L.-A. Breivik and S. Eastwood (2014), Improvement of OSI SAF Product of Sea Ice Edge and Sea Ice Type. EUMETSAT Meteorological Satellite Conference, Geneva (Switzerland), September 2014.