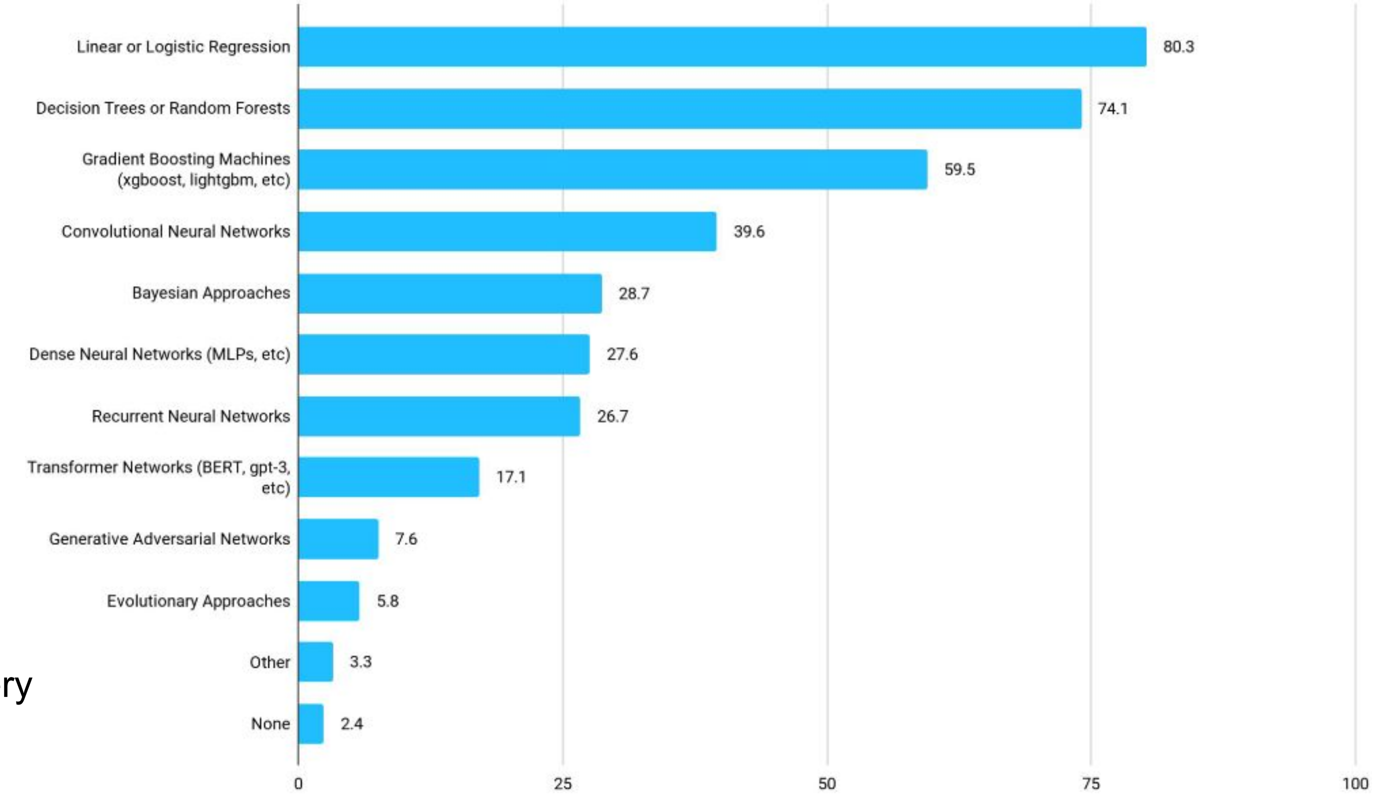
# Decision trees

# Most popular methods in Kaggle (survey 2021)




**Motivation 1:** very efficient method

# Introduction

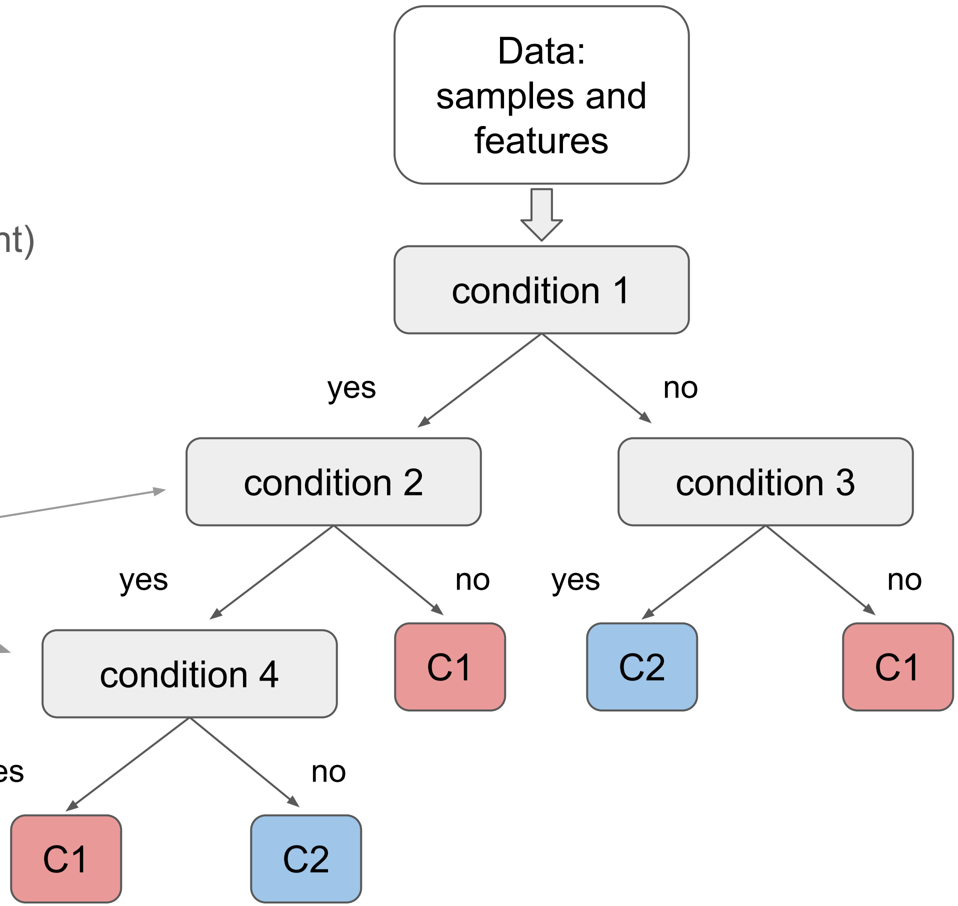
- Supervised learning
- rule-based model
- non-parametric model (no gradient descent)
- hierarchical model (binary tree)

Idea: find rules / conditions on features

Decision nodes

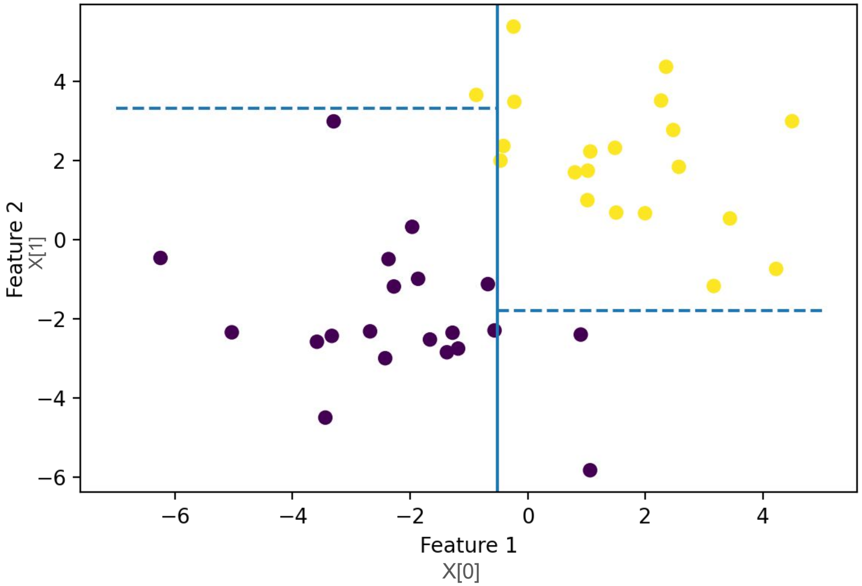
Leaf node

**Motivation 2:** interpretable method “white box”

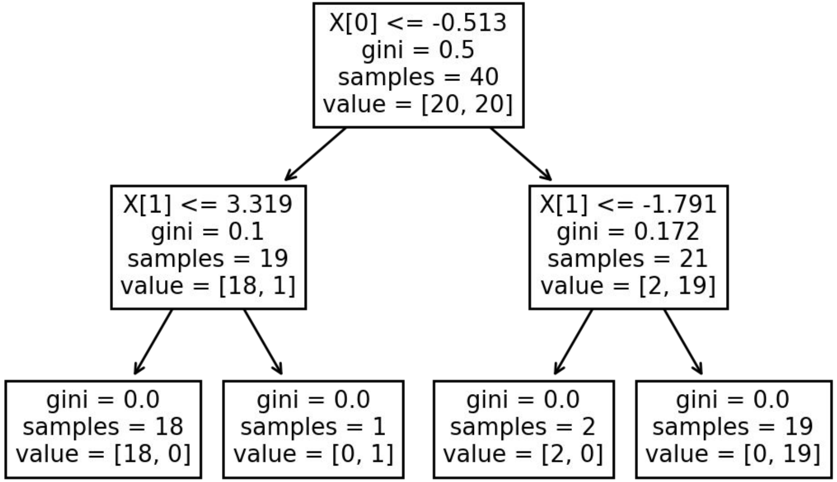


# How does it work?

Example on a dataset of 2 classes with 20 samples each. Each sample has 2 features.




Decision tree




Drawing from sklearn

How do we start the tree? how do we find the conditions? what is “gini”?

# Building the decision tree

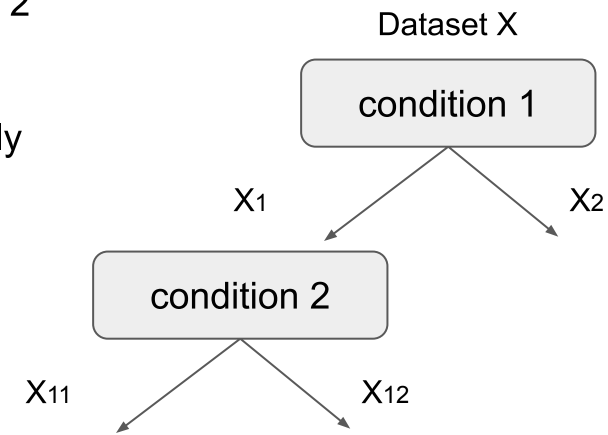
## The decision tree is built recursively

- Each decision node act on a set and split it into 2 subsets
- it calls the decision function on each subset
- until the subset is “pure enough”: it contains only or mostly one class

The procedure repeat itself at each subset:
It can be efficiently coded with **recursion**

```
GenerateTree(X)
  if PureEnough(X):
    GiveMajorityClassLabelTo(X)
    return
  (X1,X2) = MakeDecision(X)
  GenerateTree(X1)
  GenerateTree(X2)
```

A function calls itself!!



# Recursion

Ingredients:

- A function with a call to itself
- some more actions
- a stopping condition (avoid infinite call!)

Example:

```
def factorial(n):
    if n > 0:
        return n * factorial(n - 1)
    else:
        return 1
```

Recursion rules:

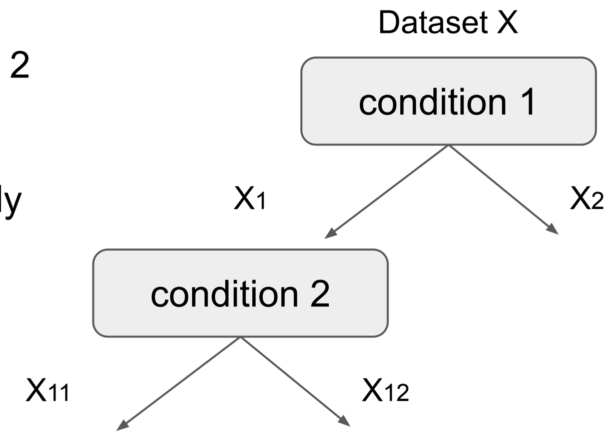
- Needs a base case (where it stops),
- Ensure each recursive call makes a step toward the base case

# Back to the decision tree

## The decision tree is built recursively

- Each decision node act on a set and split it into 2 subsets
- it calls the decision function on each subset
- until the subset is “pure enough”: it contains only or mostly one class

```
GenerateTree(X)
  if PureEnough(X):
    GiveMajorityClassLabelTo(X)
    return
  (X1,X2) = MakeDecision(X)
  GenerateTree(X1)
  GenerateTree(X2)
```

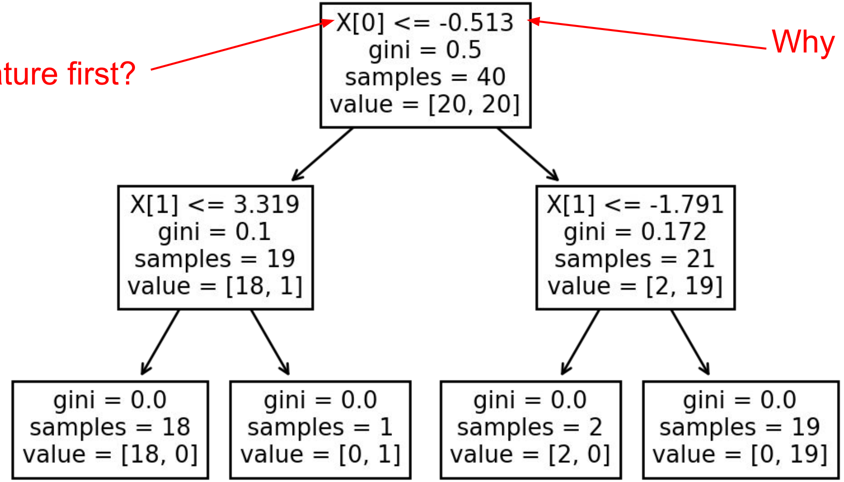


# Decision node

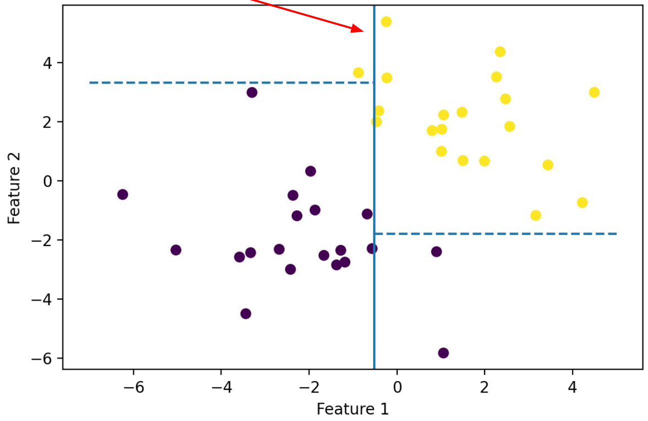
To take a decision, we need:

- a feature: the decision is on a single feature
- a purity or impurity measure: to find the class separation

Why this feature first?

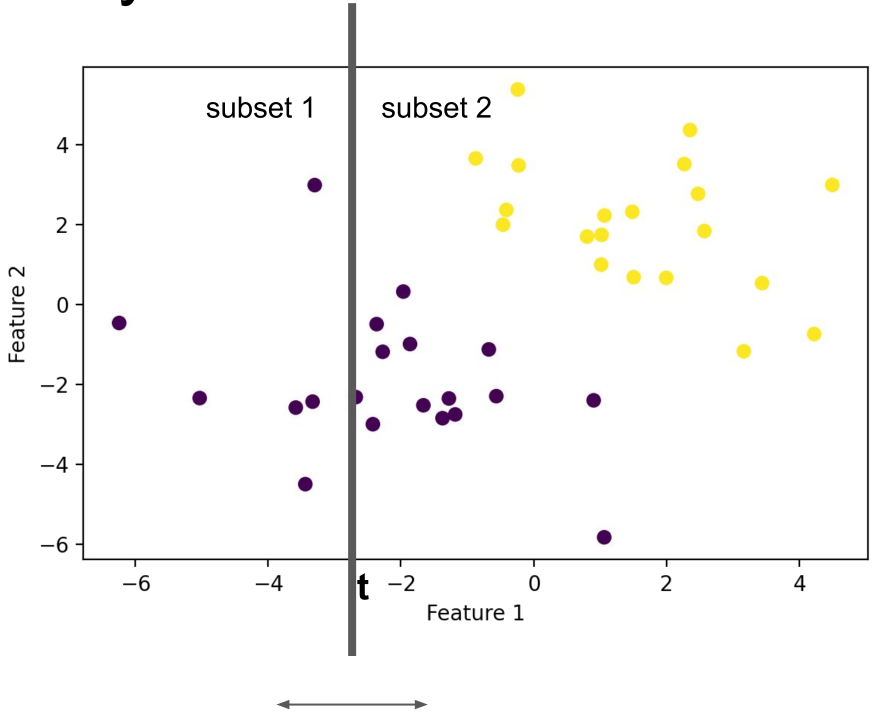



Why this separation value?




Let us first assume we have one feature and let us look at the impurity measure

# Purity / impurity




- A feature is selected
- A threshold value **t** separate into 2 subsets (below and above **t**)

Where to set the cut such that the 2 subsets are as pure as possible?

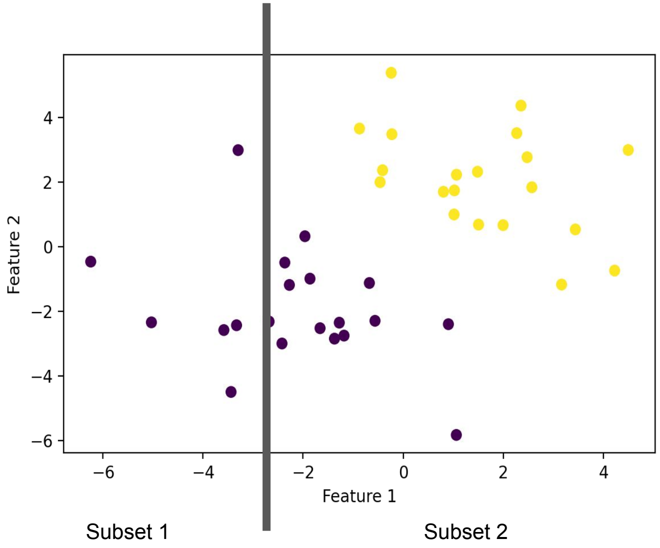
**Note for later:** the feature is selected such that classes are well separated with a threshold.

# Entropy and Gini index

## Two mathematical definitions of purity (or impurity)

2 classes (0 and 1), we look at one of the 2 subsets
In the subset,
Probability or ratio $p_0 = \frac{n_0}{n_0 + n_1}$ $p_1 = \frac{n_1}{n_0 + n_1}$

$$p_0 + p_1 = 1 \qquad p_1 = 1 - p_0$$




Gini

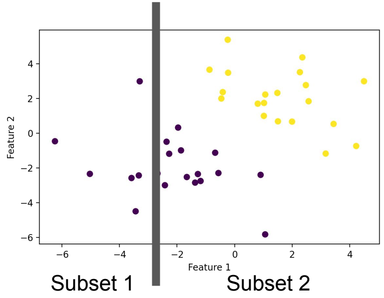
$$G = 2p_0p_1 = 2p_0(1 - p_0)$$

Entropy

$$H = -p_0 \log_2 p_0 - p_1 \log_2 p_1$$

$$p_0 = 0\ ?\ , p_1 = 0\ ?$$

# Entropy and Gini index




## Two mathematical definitions of purity (or impurity)

2 classes (0 and 1), we look at one of the 2 subsets
In the subset,
Probability or ratio $p_0 = \frac{n_0}{n_0 + n_1}$ $p_1 = \frac{n_1}{n_0 + n_1}$

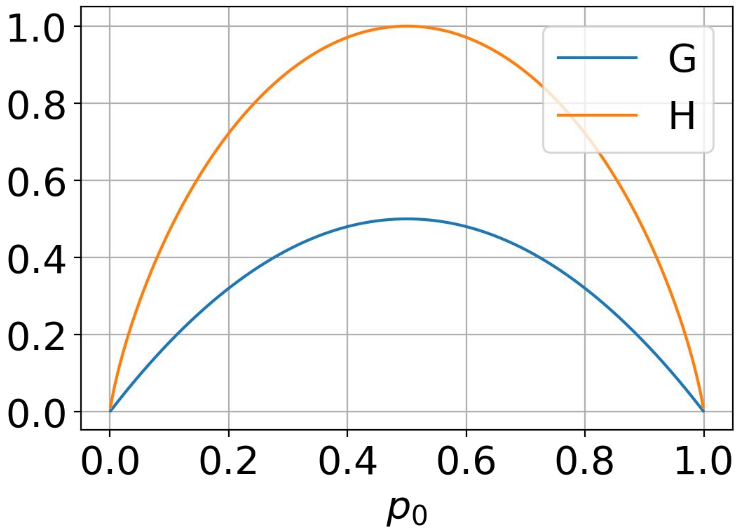
$$p_0 + p_1 = 1 \qquad p_1 = 1 - p_0$$

Gini

$$G = 2p_0p_1 = 2p_0(1 - p_0)$$

Entropy

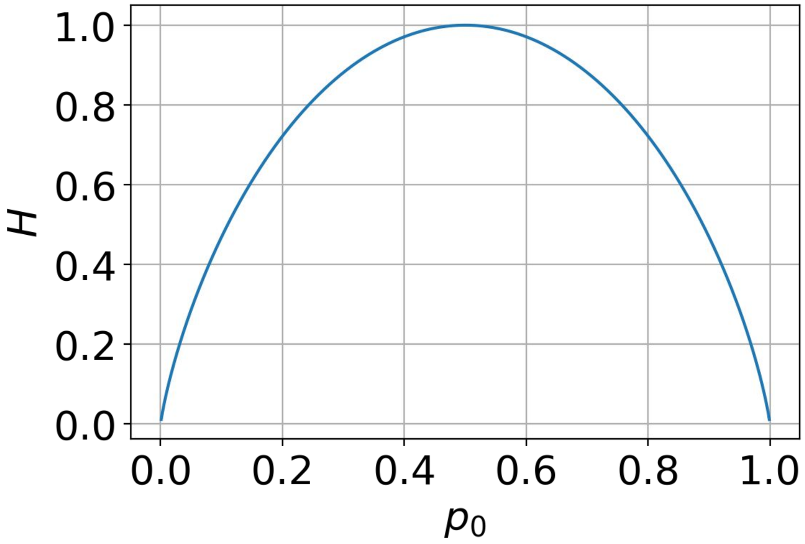
$$H = -p_0 \log_2 p_0 - p_1 \log_2 p_1$$




- for 2 classes (0 and 1)

Probability or ratio $p_0 = \frac{n_0}{n_0 + n_1}$ $p_1 = \frac{n_1}{n_0 + n_1}$

$$p_0 + p_1 = 1$$




$$H(p) = -p_0 \ln p_0 - p_1 \ln p_1$$

Lowest entropy when the set has a single class

- More generally for k classes

$$H(f) = -\sum_k f_k \ln f_k \quad \text{with} \quad \sum_k f_k = 1$$

$$G = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{N} p_k^2$$

# Entropy - general point of view

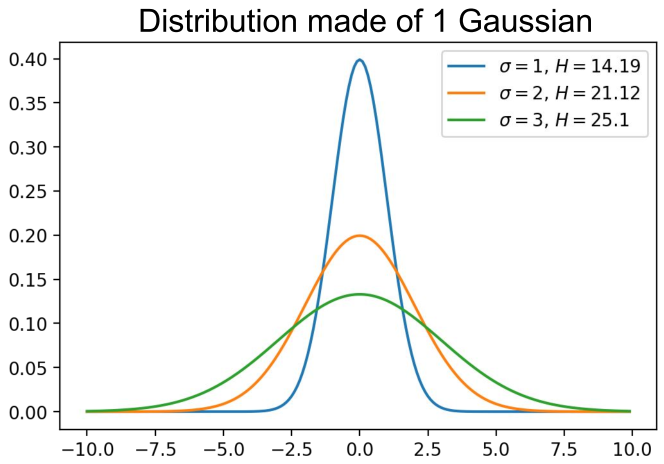
Entropy measures the spreading of a distribution
It measures the variety, the disorder or the “mess”

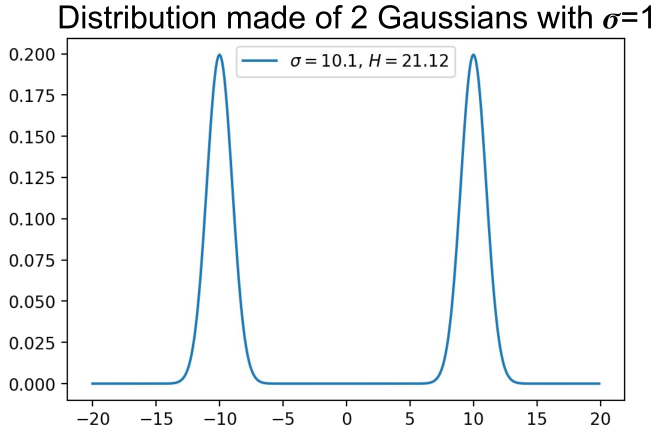
$$H(f) = -\int f(x) \ln f(x) dx$$

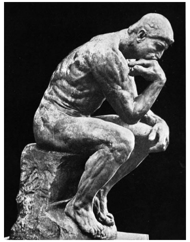
for f probability distribution with:

$\int f(x)dx = 1$ and $f(x) \geq 0$

Entropy increase with variance



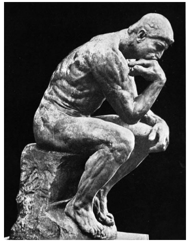



variance is not a good measure of the spreading here, but entropy is

# Entropy

This concept can be found in

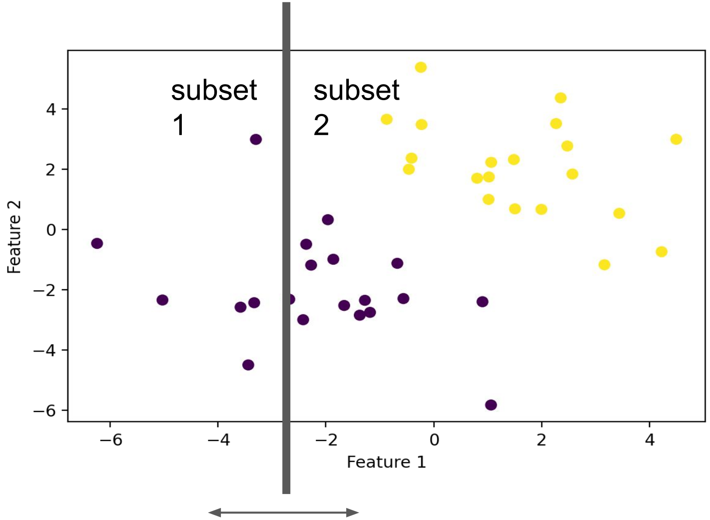
- Physics Second law of thermodynamics (Entropy cannot decrease)
- computer science (information theory). How many bits are needed to encode information (Shannon entropy)



# Is it a mess of classes in my subsets?

For each separation threshold:

- compute the entropy or Gini coeff. for both subsets
- compute the total disorder:




$$H = \frac{N_0}{N} H_0 + \frac{N_1}{N} H_1$$

subset 1

subset 2

N total number of samples

Choose the threshold with the lowest score

# Choice of feature

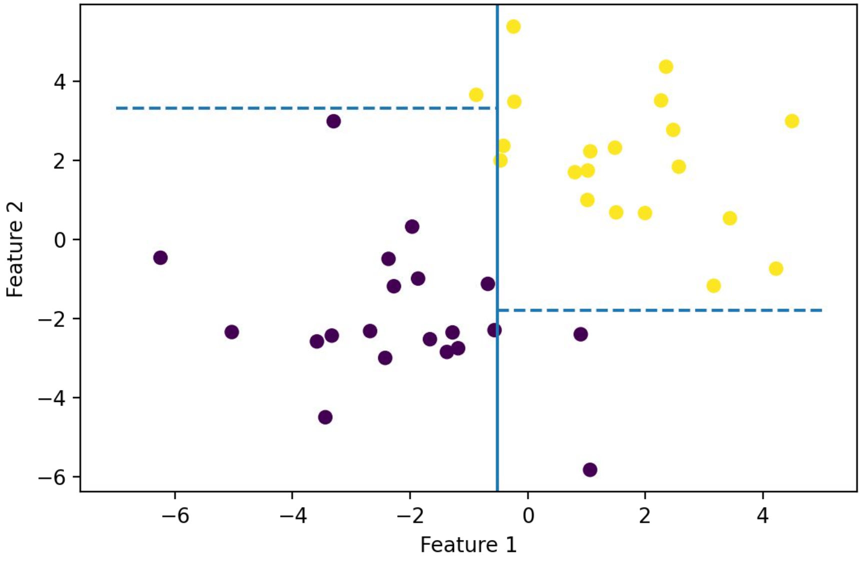
- For each feature, compute the best impurity score: the best split
- Compare impurity scores across features and select the feature with the best one

The feature with the best split is selected for the node decision

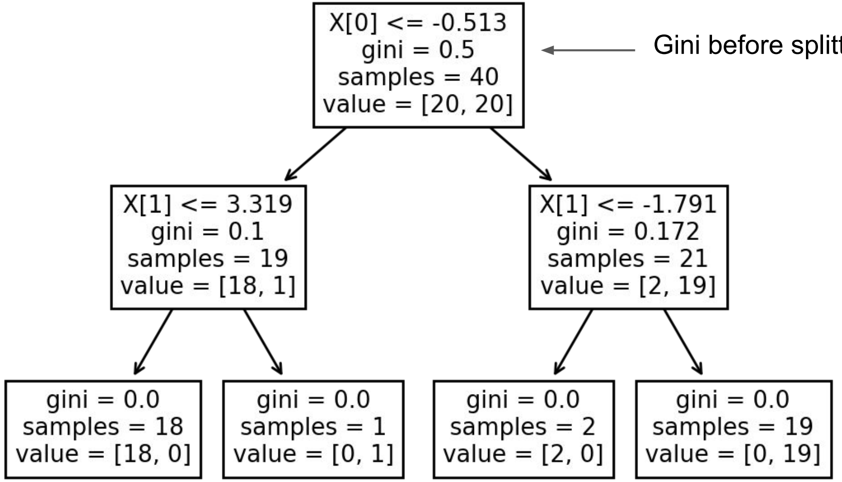
2 loops: loop over the possible cut values and loop over the features

# Back to our example

Example on a dataset of 2 classes with 20 samples each. Each sample has 2 features.




Decision tree



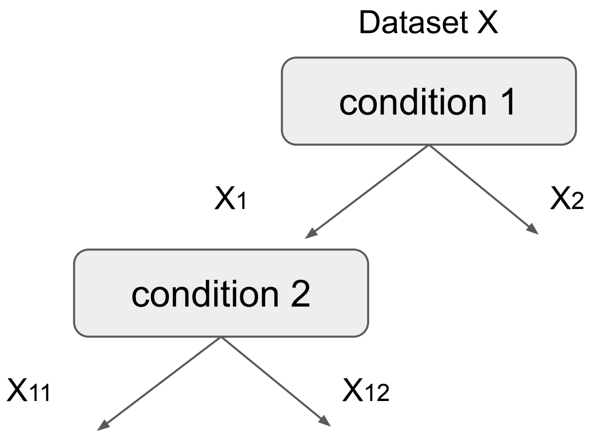

Gini before splitting

Drawing from sklearn

# Back to the decision tree architecture

```
GenerateTree(X)
  if PureEnough(X):
    GiveMajorityClassLabelTo(X)
    return
  (X1,X2) = MakeDecision(X)
  GenerateTree(X1)
  GenerateTree(X2)
```

Now we know what to put in the stop condition and in the “Makedecision()”

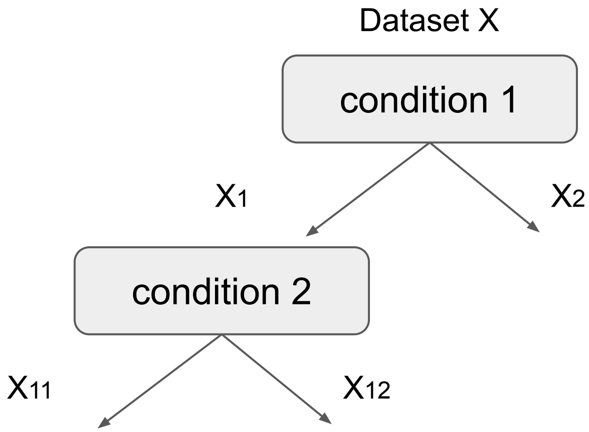



**Stop conditions:**

- if entropy H(X)=0 or H(x)< $\boldsymbol{\varepsilon}$
- if X too small
- optional: if tree has too many leaves (to prevent overfitting)

# Back to the decision tree architecture

```
GenerateTree(X)
  if PureEnough(X):
    GiveMajorityClassLabelTo(X)
    return
  (X1,X2) = MakeDecision(X)
  GenerateTree(X1)
  GenerateTree(X2)
```




**MakeDecision():**

- loop over all features
  - loop over all possible thresholds
  - compute entropy for each case
- Save feature and threshold with the best overall entropy (smallest)
- return the 2 subsets with best split


**Note:** GenerateTree need to save the information for the decision at each node (not shown)

# Variations of decision trees

# Features with discrete values

The splitting process has to be modified

If we have **n** discrete values, we split into **n** subsets
We compute the entropy for this split:

$$H = -\sum_{i=1}^{n} \frac{N_i}{N} \sum_{k=1}^{K} p_k(i) \ln p_k(i)$$

- K is the number of classes
- N number of samples
- $N_i$ number of samples in subset i

# Regression with decision trees

Decision trees can be used for regression

- approximate the label by a piecewise constant function
- the impurity is replaced by the mean squared error

For each subset Q, compute:

$$E = \frac{1}{N} \sum_{i \in Q} (y_i - \bar{y})^2$$

Mean value inside the subset

The best cut minimize E in the subsets: the points in the subset should be close to a mean value in the subset

# Stopping criteria & overfitting

To avoid overfitting

- Almost pure is often enough
- splitting size
- Tree depth should not be too high

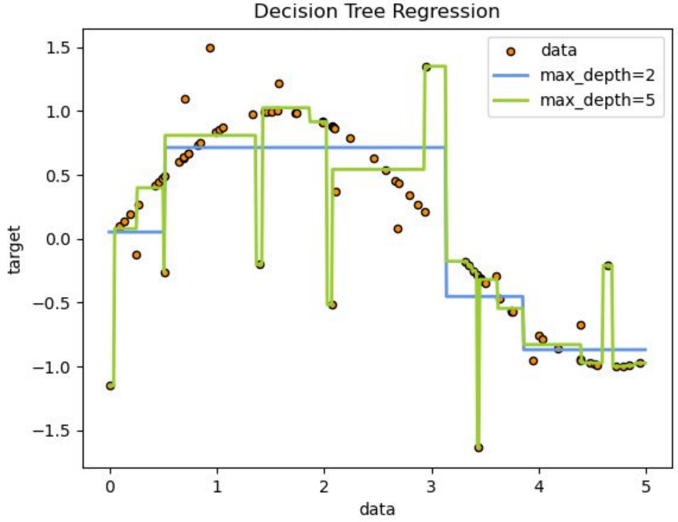
# Different decision trees

- We have seen the CART (Classification and Regression Trees) model, with its purity measure. This is the one used in sklearn
- For categorical features, a branch per value (called ID3 or C4.5)
- Multivariate trees: variants taking into account several features at the same time. Example: replacing separating with entropy by doing logistic regression to get the class separation inside a decision node.

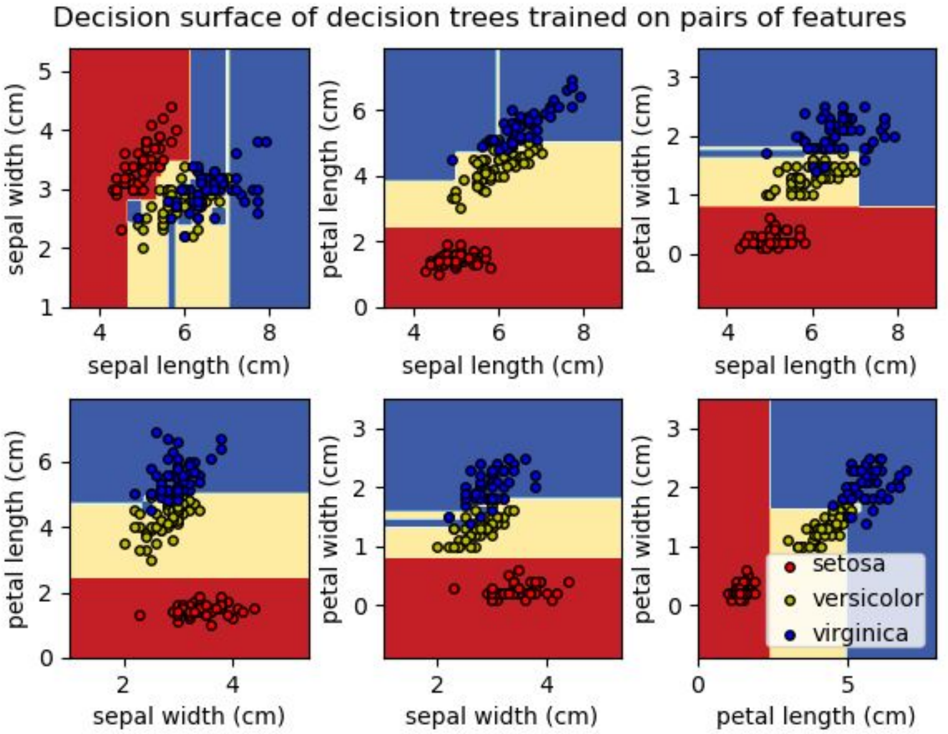
# Limits & overfitting

- Piecewise approximation
- Prone to overfitting

1d example (1 feature)




2d example (2 features)
classification with 3 classes




Do you see the overfitting problem?

Pictures from https://scikit-learn.org/stable/modules/tree.html

# Interpretability & explainability

What are the most important features for the classification/regression task?

-> just look at the features selected in the decision tree!

Why a sample is classified in a particular class?

-> just follow the decision tree! The rules are simple to understand