Decision Trees and Random Forests
Data science Certificate

Aymeric DIEULEVEUT

May 2021

- why ?
- what they are used for?
- how is it built $\longrightarrow$ training / algorithm.
- how to implement a decision tree in python.
- pros / cons / tips.

1. Decision Trees
2. Random Forests

Outline

A Reminder on Supervised Learning

we have access to : (X_i, Y_i) on the training set

goal: find a fund° f. $f : \mathcal{X} \longrightarrow \mathcal{Y}$

$X_{new} \longrightarrow f(X_{new})$

- Observations: $(X_i)_{i=1,\dots n}$. Each X_i has d components.
- Outputs: $(Y_i)_{i=1,\dots n}$

3 Examples:

- Iris Dataset: Y is 3 possible classes
- MNIST: Y is $\{0, \dots, 9\}$
- House price prediction $Y \in \mathbb{R}_+$ $Y_i \geq 0$

Example: Iris Dataset

Figure: Iris: setosa, versiocolor, virginica

One of the most widely used toy dataset in classification:

- 3 classes : *['setosa', 'versicolor', 'virginica']* $\longrightarrow$ $Y_i \in \{0, 1, 2\}$
- 50 points per class. $n = 3 \times 50 = 150$ obs
- 4 features: *['sepal length', 'sepal width', 'petal length', 'petal width']*

$X_i = [\quad\quad\quad\quad]$

Example: MNIST Dataset

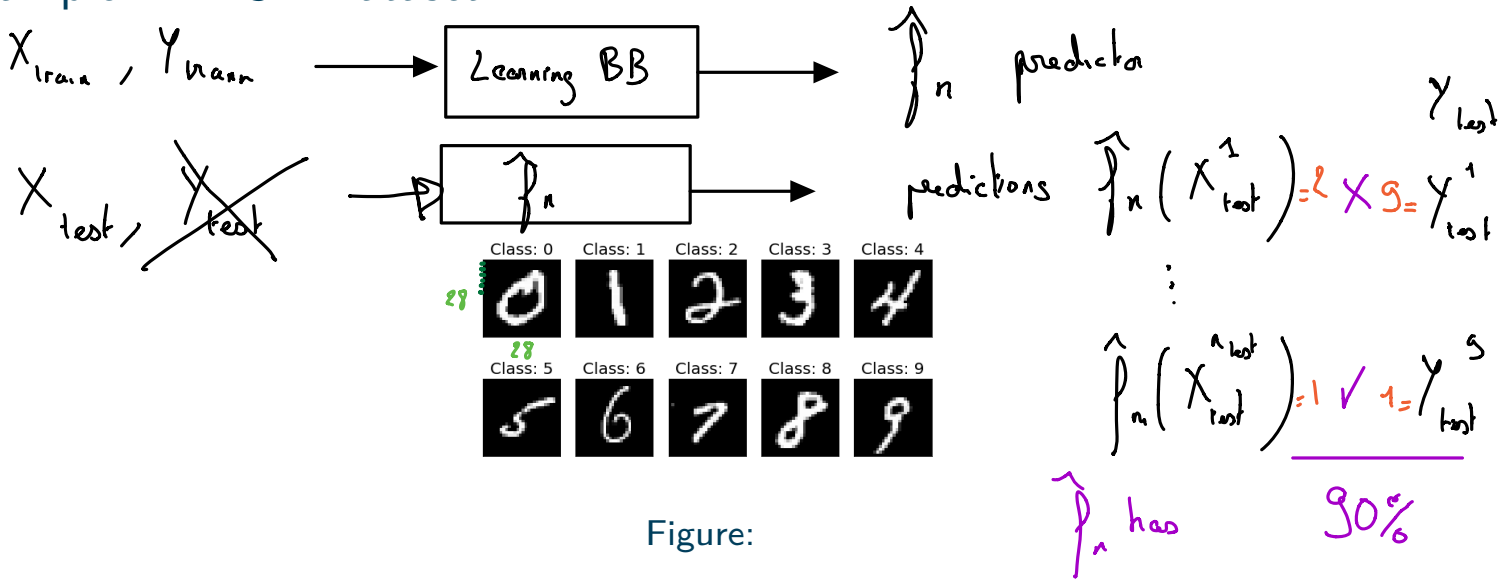
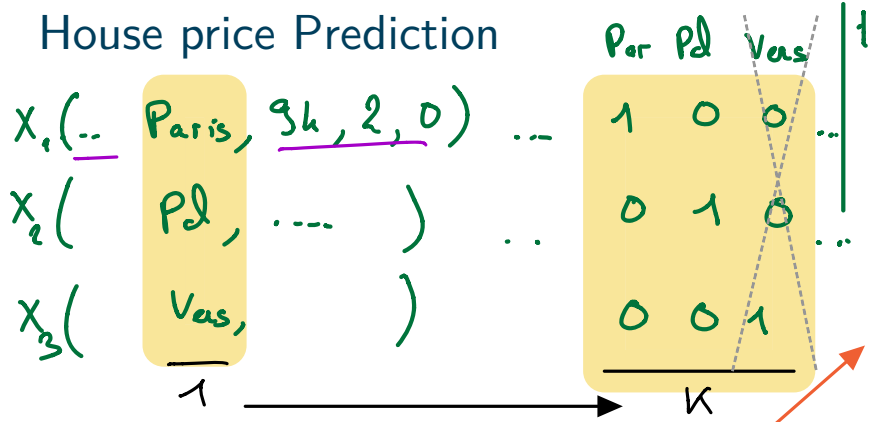


Figure:

Digit recognition from an image:

- 10 classes : $Y_i \in \{0, \dots, 9\}$
- 60k images, (50k train, 10k test), balanced
- 784 features.

House price Prediction



Trick: one-hot-encoding.

get_dummies (pd)

drop_first = True.

quantitative

{Paris, Palaiseau, Vers.}

$\mathbb{R}_+$

- Output : house value $Y \in \mathbb{R}$
- 1500 examples .
- 81 features: Construction year, Neighborhood, Surface, Floor, Balcony: both quantitative and qualitative, some ordinal variables.

{0, 1}

linear link between features

Iris: } all features were quantitative

Mnist: }

d° cdums = 4 + 3

$X =$ (n rows)

drop 1 to avoid linearly dep features!!

Example about linearly dependant features:

Y = salary = k€

X = height

$$Y \simeq 20 + \underbrace{(X - 1m)}_{\text{one feature}} \times \underbrace{10}_{\alpha_1}$$

$$10 \times h_m + 0\, h_{inches} = 0\, h_m \times 300\, h_{inches}$$

Second character. / feature

$$10 \times (\text{height m})$$

$$\Longleftrightarrow \quad 300\ (\text{height inches})$$

X = (height in m, height in inches)

$$\alpha_1 (\text{height m})$$

$$+ \alpha_2 (\text{height inches})$$

$$\hat{\alpha} = (X^T X)^{-1} X Y$$

does not exist

Summary of those three datasets

	Iris	MNIST	House price prediction
Number of examples n	150	60k	1500
Number of the features d	4	784	81
Type of the features	quant	quant	quant + qual.
Space $\mathcal{Y}$	$\{0, 1, 2\}$	$\{0, \dots 9\}$	$\mathbb{R}_+$
Task	Classificat°.		Regression task.

↳ $\mathcal{Y} = \{0 \dots K\}$

Decision Tree

also trees in unsupervised but they are ≠ !

Goals of Decision Trees

1. Supervised Learning: solve the classification or regression task
2. Provide some understanding: what on this example allows me to classify it? What are the important features ?

Age BMI

Y is severe covid risk

if age is 70+ → Y = 1,

if BMI is 35+ → Y = 1

$Y = \mathbb{1}_{0.5 < \frac{1}{40}(age - 50) + \frac{1}{20}(BMI - 25)}$

Decision Tree

Goals of Decision Trees

1. Supervised Learning: solve the classification or regression task
2. Provide some understanding: what on this example allows me to classify it? What are the important features ?

Two questions:

- How to train a decision tree? $\rightarrow$ **training phase (learning algorithm)**
- How to predict with a given decision tree? $\rightarrow$ **inference phase**

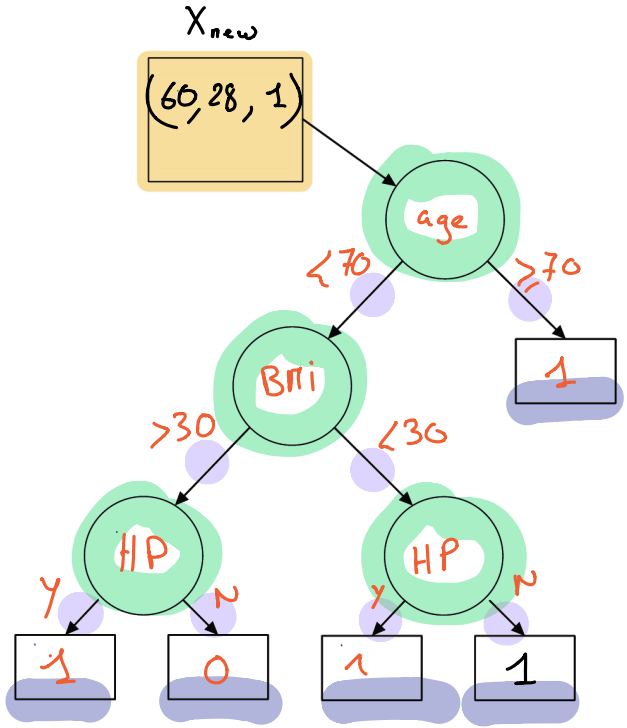
CART.

Inference Phase

$Y = \{0, 1\}$ at risk COVID 19

$X = (\text{age}, \text{BMI}, \text{"have-precondit"})$

- Root = observation = input
- Nodes = tests
- Branches = possible outcomes
- Leaves = final decision = output



orange = result of the training phase

↳ later

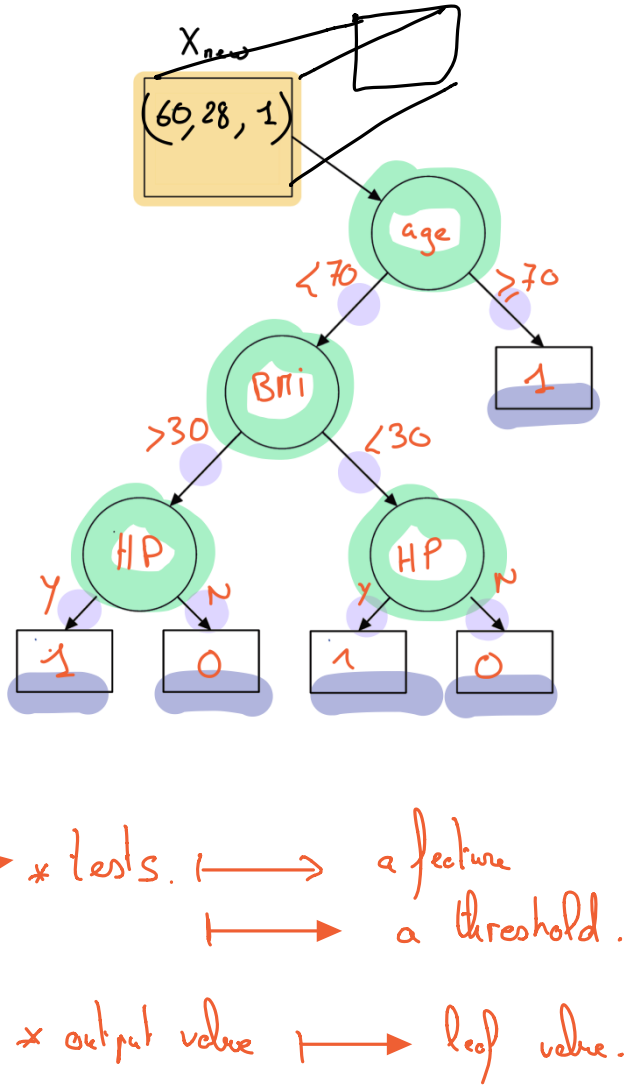
already trained

Training: how to build a tree.

Key question !

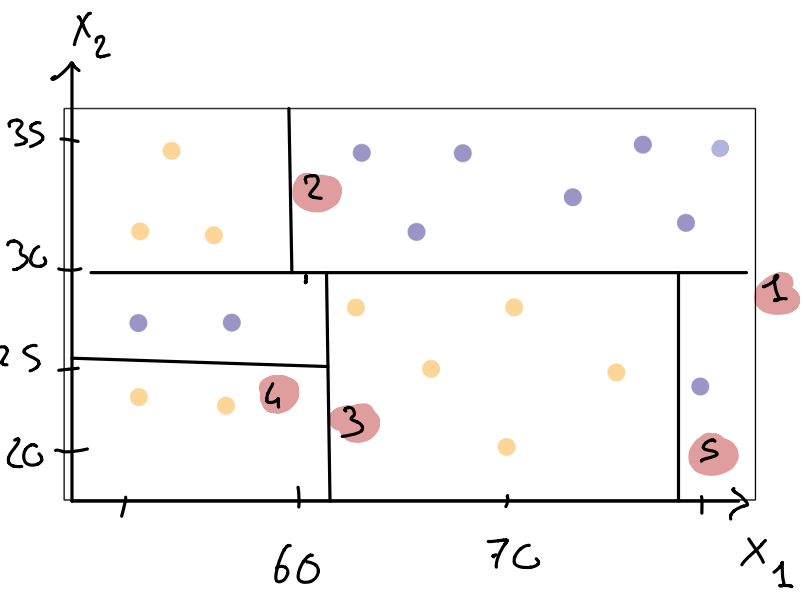
- What is a good tree?
- What do I need to choose?
- How to choose?

What do you think?



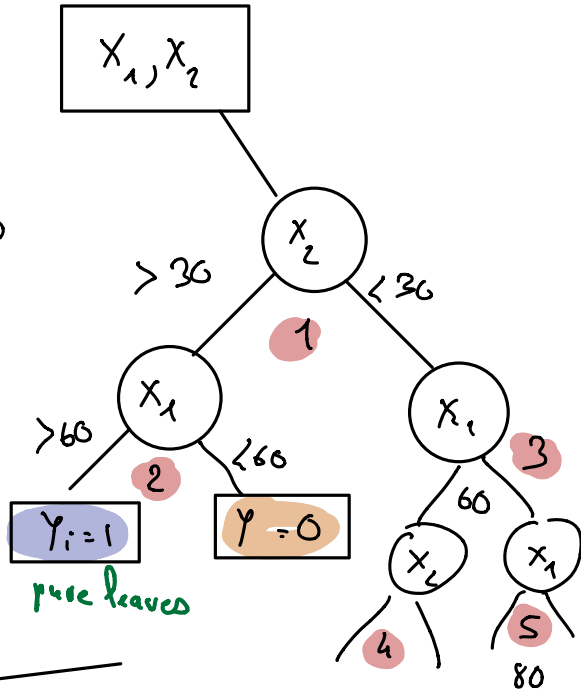
Building trees : CART

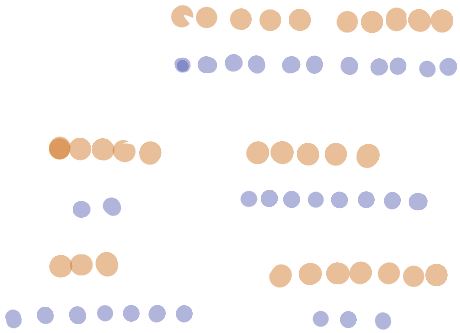
Restart at 11h30



$Y_i = 1$

$Y_i = 0$





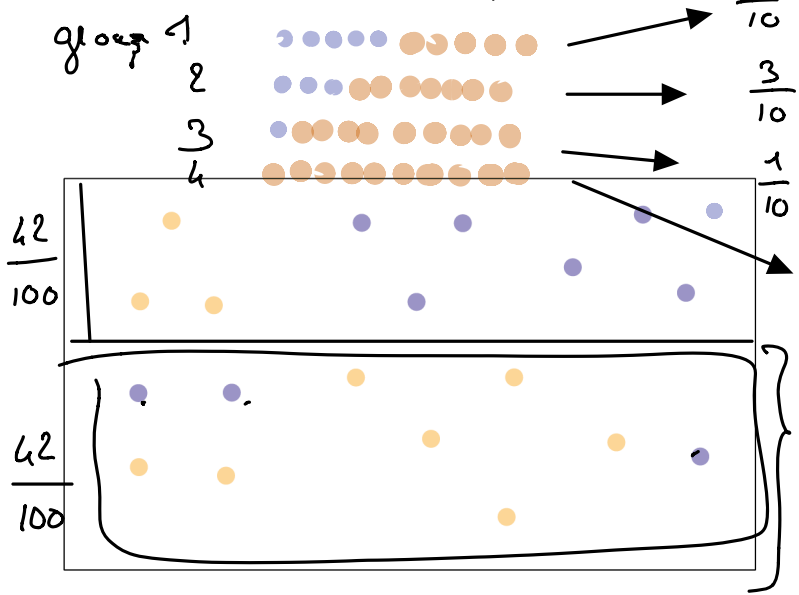
Gav

split 1 X_1 vs 60 → 45°

split 1 bis X_2 vs 30 → 42%

Todo : choose a quality criterion.

Gini's impurity



Classified

$\frac{5}{10} \times \frac{5}{10} + \dots = \frac{50}{100}$

$\frac{3}{10} \times \frac{7}{10} + \frac{7}{10} \times \frac{3}{10} \rightarrow \frac{42}{100}$

$\frac{1}{10} \times \frac{9}{10} + \frac{9 \times 1}{100} \rightarrow \frac{18}{100}$

0

Gini : in python, default choice!

Blue pts

$$\sum_{i=1}^{2} = \frac{3}{10} \times \left(\frac{7}{10} \right) = \frac{21}{100}$$

$$+ \frac{7}{10} \left(\frac{3}{10} \right) = \frac{21}{100}$$

Gini impurity measures **how often a randomly chosen element from the set** would be **incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset**.

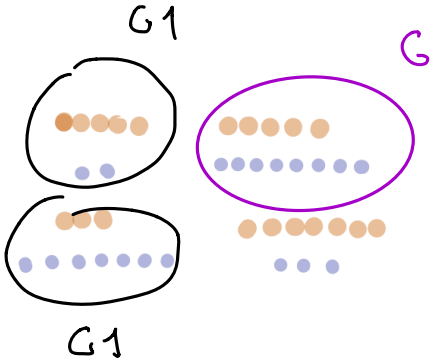
nb of classes.

prob of misclass. this point

$$I_G(p) = \sum_{i=1}^{J} p_i (1 - p_i) = 1 - \sum_{i=1}^{J} p_i^2$$

For two classes, related to the variance if classified as Bernoulli r.v..

prob of picking class i



split 1 X_1 vs 60

split 1 bis X_2 vs 30

45%

$\frac{1}{7} + \frac{4}{13}$ Score split 1

vs $\frac{42}{100}$ split 2

Best criterion at the end

Split 1

	Gini	#pts
G1 (2,5)	$2\left(\frac{2}{7} \times \frac{5}{7}\right) = \frac{20}{49}$	7
G2 (8,5)	$2\left(\frac{5}{13} \times \frac{8}{13}\right) = 80/169$	13

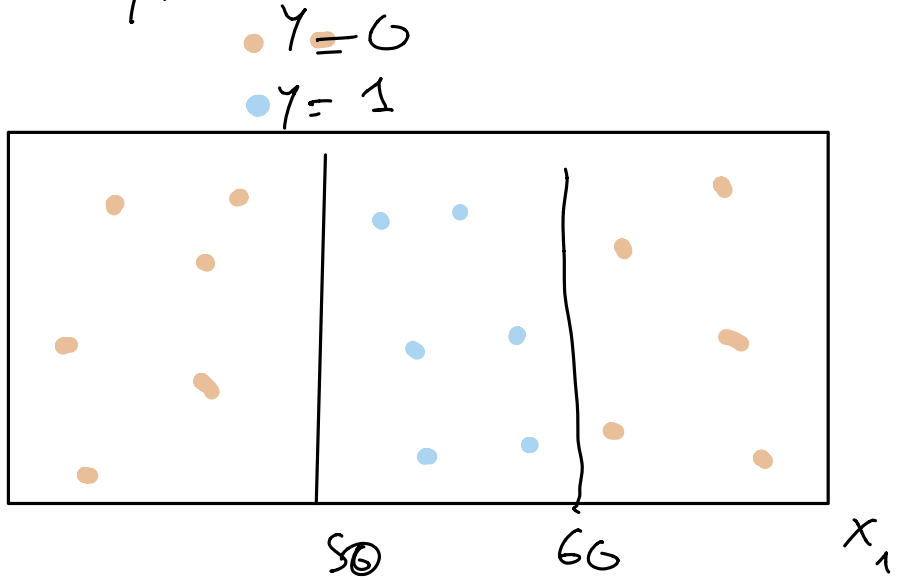
$\frac{7}{20} \times \frac{20}{49} = \frac{1}{7}$

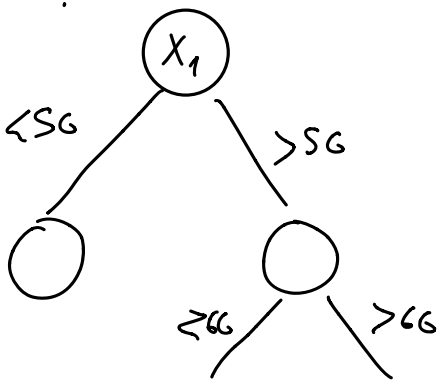
$\frac{13 \times 80}{20 \times 169} = \frac{4}{13}$

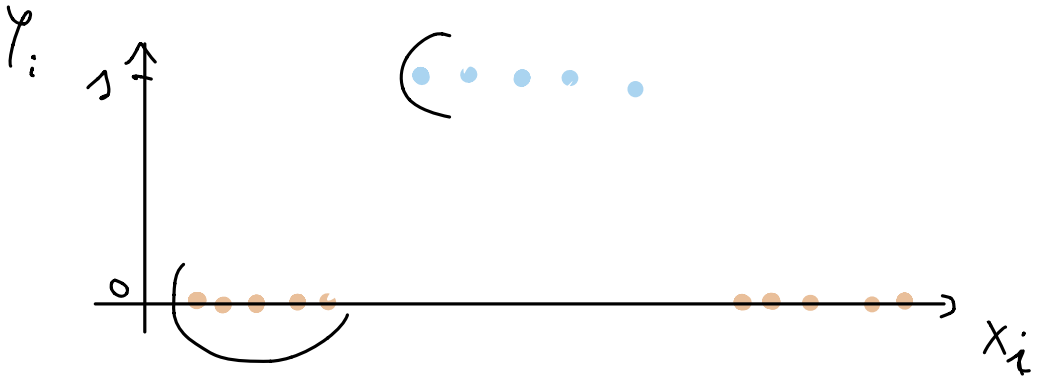
Split 2:

	Gini's impurity	#pts	Average
G1 →	$\frac{42}{106}$	10	$\frac{42}{100}$ Score for split 2.
G2 →	$\frac{42}{100}$	10	

Summary.

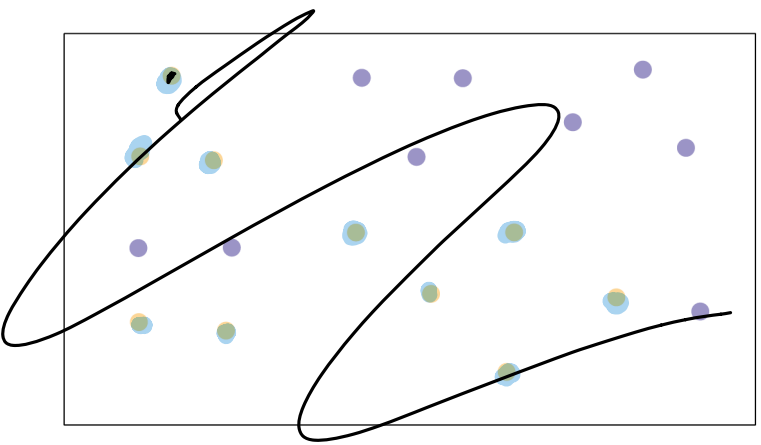


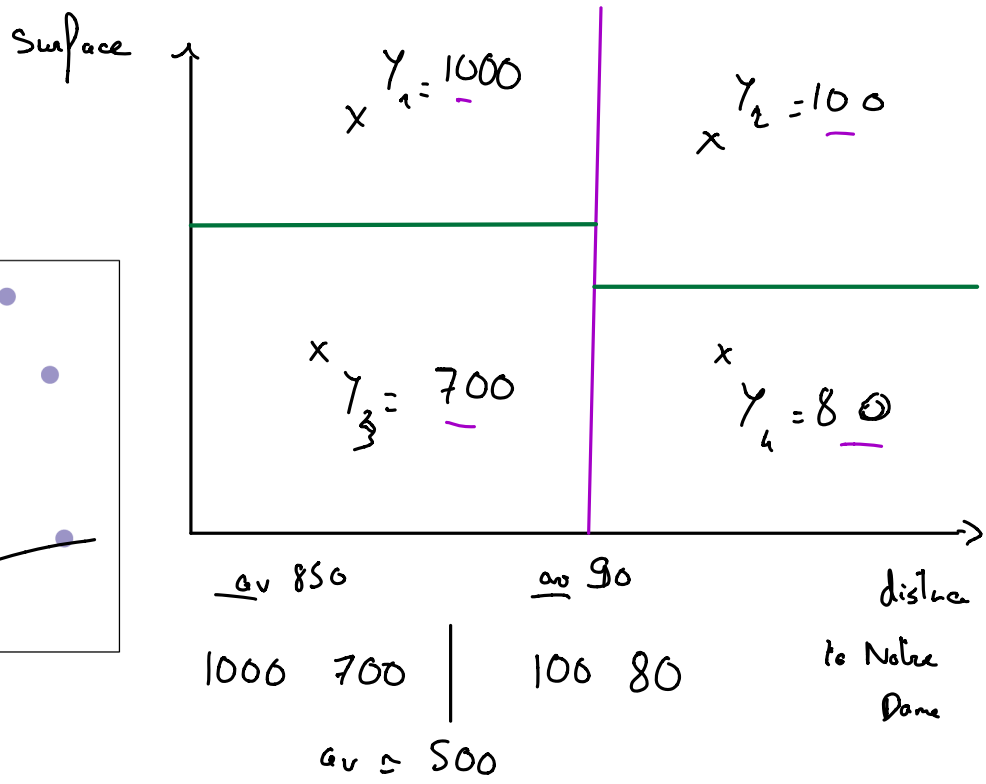




Yesterday: linear models

Building trees : CART





How to chose the best cut ?

- e.g., Given a possible cut, measure the reduction of « impurity »:
  - in regression: mean-squared-error $\sum_{i \in C}(Y_i - \bar{Y}_C)^2$
  - in classification: Gini's impurity.
- Find dimension and threshold with optimal impurity reduction.
- Iterate until stopping criterion is met.

Learn

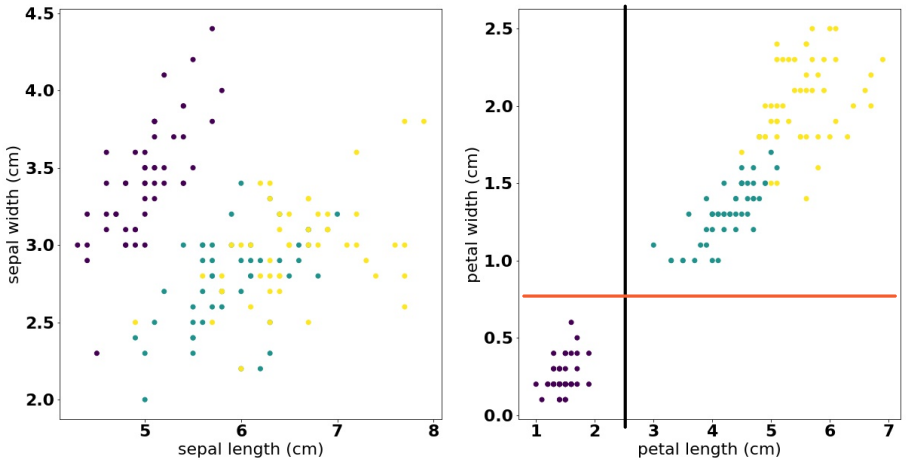
until "pure leaves" (Gini = 0) → until reduction in Gini is small
until a maximal depth.
until a minimal nb of pts per leaf.

Example: Iris Dataset

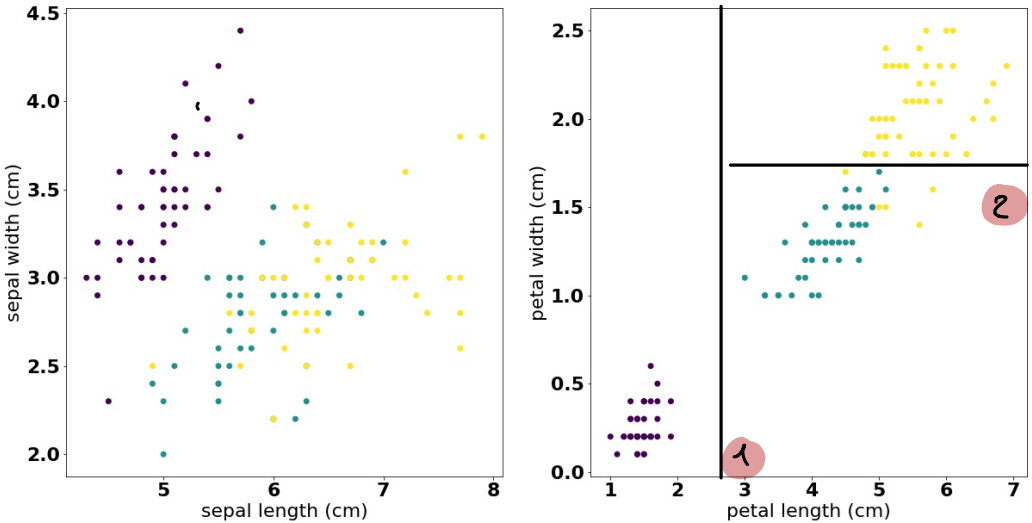
Figure: Iris: setosa, versiocolor, virginica

One of the most widely used toy dataset in classification:

- 3 classes : *['setosa', 'versicolor', 'virginica']*
- 50 points per class.
- 4 features: *['sepal length', 'sepal width', 'petal length', 'petal width']*



Exercise: Guess Classification tree for Iris Dataset



In pratice : Scikit Learn

Universal Python library for Machine Learning:

- Very easy to use
- Very well documented
- Open Source

`https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`

Example: Iris Dataset

What do we need to specify ?

1 : load data

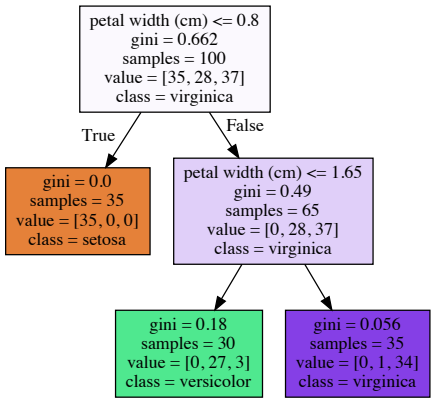
2 : choose parameters of the tree:

* impurity : Gini is default
* stopping criterion e.g. depth

3 : Learn. clf.fit(X_{train}, Y_{train})

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split

# Choose the Learning algorithm
clf = DecisionTreeClassifier(max_depth = 2,
                             random_state=0)

#Load the dataset
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, test_size=0.33, random_state=43)

#Check accuracy
cross_val_score(clf, iris.data, iris.target, cv=10)

# Fit the model
clf.fit(X_train,y_train)

# Plot tree
plot_tree(clf, feature_names = fn,
                class_names=cn,
                filled = True)
plt.show()
```

Output:



Example:

```python
# Check
X_test[0,:], iris.target_names[y_test[0]]
```

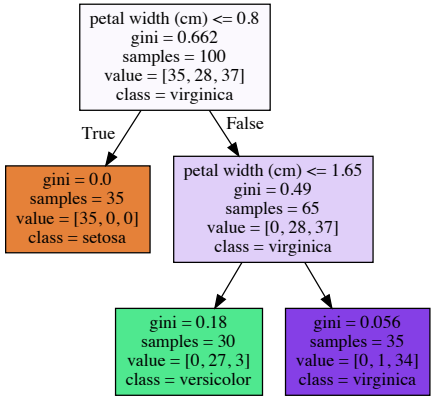
Returns:

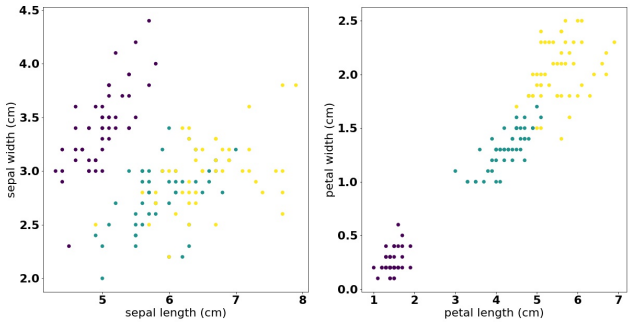
(array([4.8, 3.1, 1.6, 0.2]), 'setosa')

Code: https://colab.research.google.com/drive/1dXYIjacNAeASgl54vIWyWjMQlfgigSVA?usp=sharing#scrollTo=VeJHkJd7RsDw

Example: Iris Dataset

Output:





Example:

```
# Check
X_test[0,:], iris.target_names[y_test[0]]
```

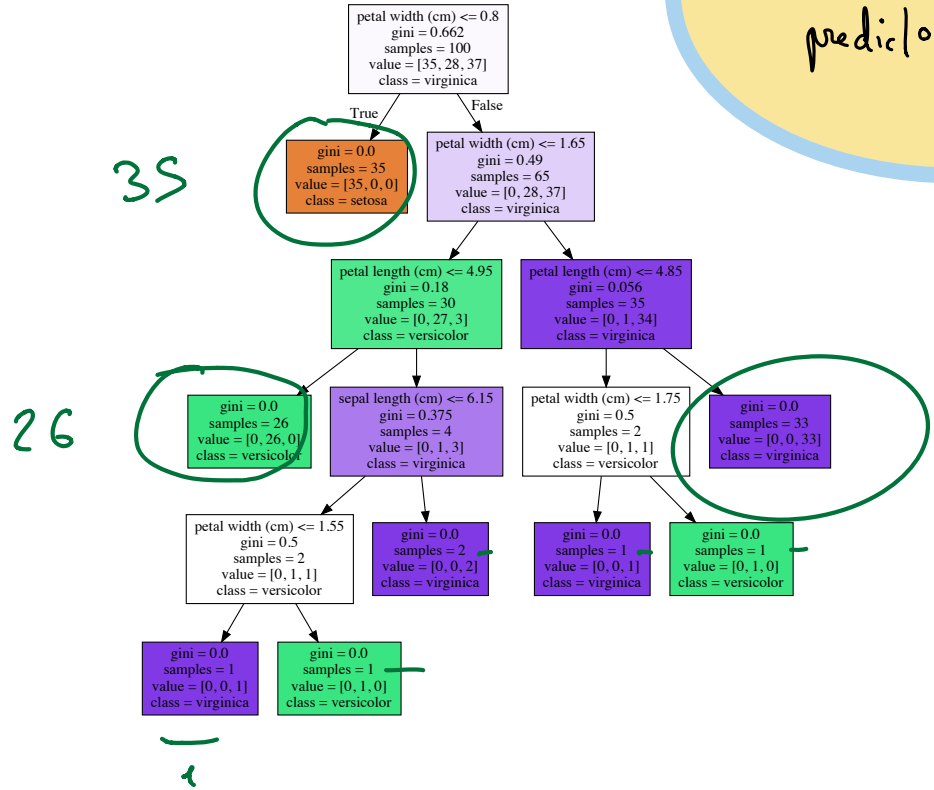
Returns:
(array([4.8, 3.1, 1.6, 0.2]), 'setosa')

Code: `https://colab.research.google.com/drive/1dXYIjacNAeASgl54vIWyWjMQlfgigSVA?usp=sharing#scrollTo=VeJHkJd7RsDw`

What if I use a deeper tree ? Or if I do not specify the depth in the model definition ?

Deeper trees

```
# Choose the Learning algorithm
clf = DecisionTreeClassifier(random_state=0)
```



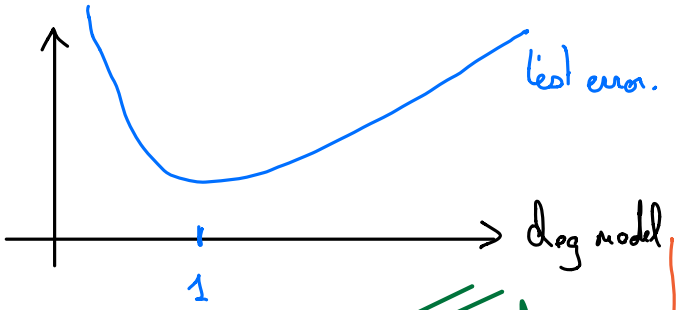
- When does the algorithm stop then ?
- What do you think about it ?

Overfitting : → cross validdo to choose parameters → depth, etc.

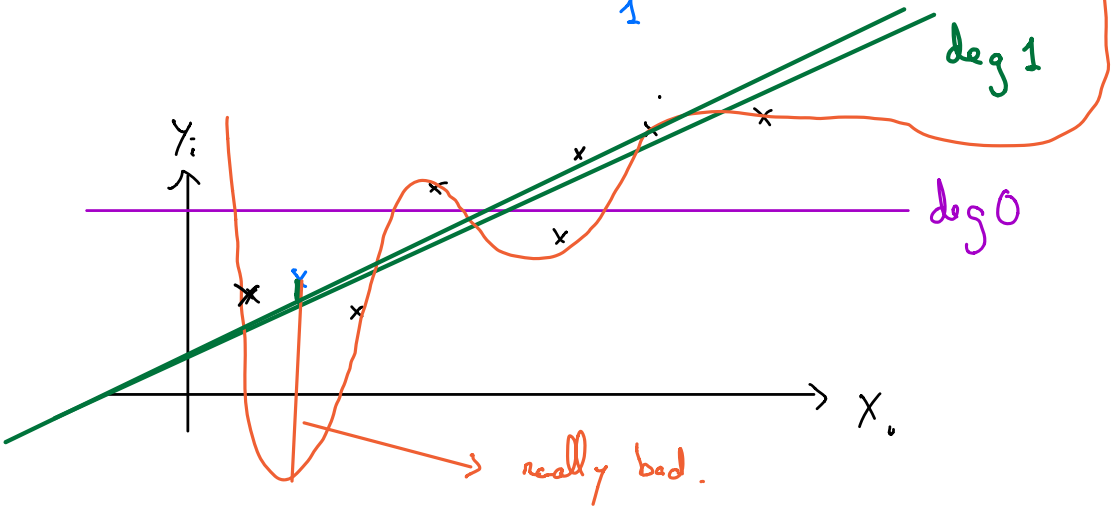
when I increase the cplxty of model. | and I learn

perf train improves

perf test : improves at first degrades

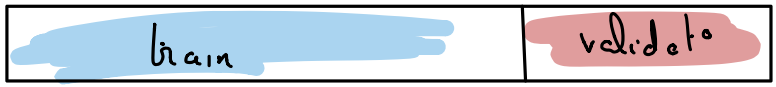


models depend on the degree of the poly



x train set.

x test point



sets

easy.

cross validate

D_1

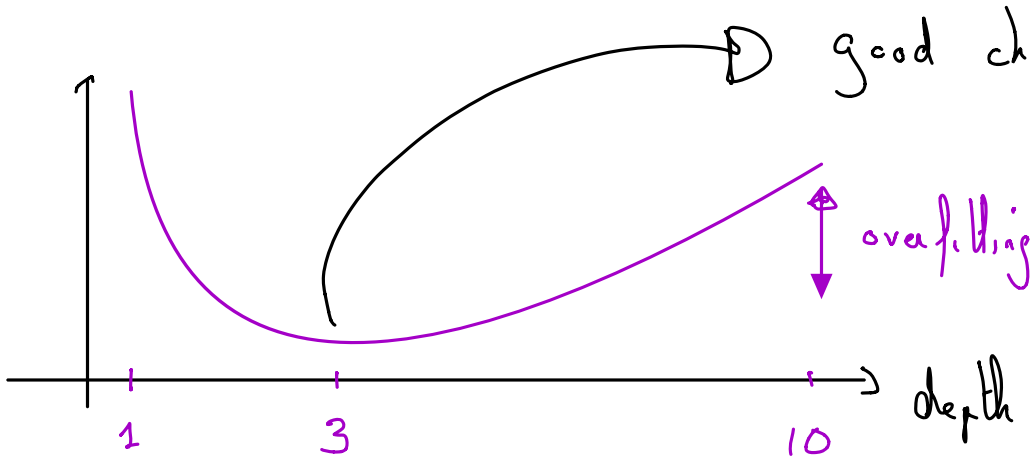
several train/val splits.

depth = 3

⋮

depth = 10

fit on train , evaluate validate error.



good choice is depth=3

Stopping / Pruning

on the train set !!

Trees that achieve perfect classification may suffer from over-fitting. (see example in the lab)

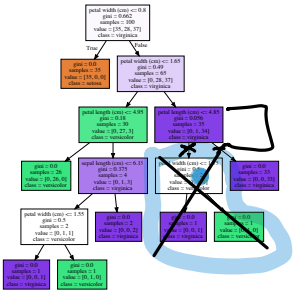
↳ poor perf on the test

To avoid this problem, we can reduce the complexity of the trees:

1. Stopping rules
   - Set a minimum number of samples inside each leaf.
   - Set a maximum depth.
2. Pruning
   - Reduced error pruning.
   - Cost complexity pruning.

Pruning : example of Cost complexity pruning.

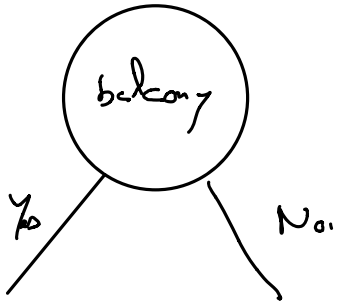
We create a sequence of Trees by changing one subtree at each step into a leaf, until we get only the root: the subtree is chosen to minimize the error made.[1] Then the test error is evaluated along the sequence, to choose the optimal pruning.



[1] *more details here*

Cart: pros and cons

Lpm



Pros:

- Simple to understand, interprete, visualize
- Implicitly perform features/variable selection
- can handle both categorical and numerical data
- little effort for data preparation
- Non linear relationships do not affect performance ✓ See before.
- Can work with large number of observations.

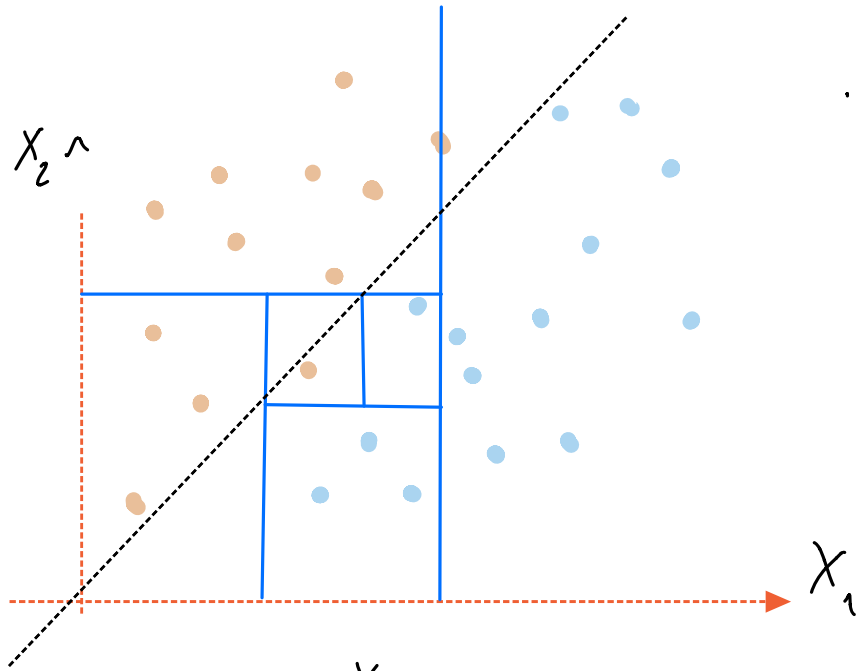
true in theory not implemented in sklearn.

Cons:

1. Can create over-complex trees: overfitting → use one hot encoding!
2. Unstability: small variations of data can generate completely different trees
3. Cannot guarantee global optimal tree →
4. Biased if some classes dominate

first split.
- d possible features
- n possible thresholds.

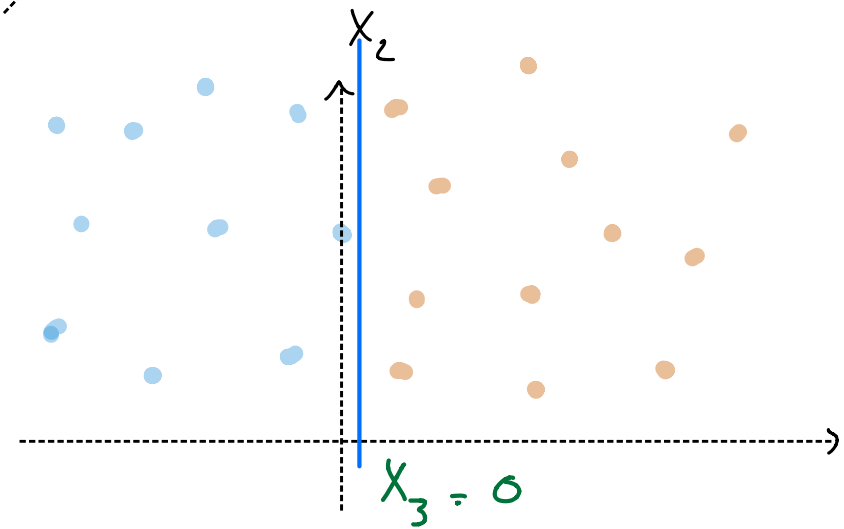
Solution : Using Random Forests !



Grigind tree.

Creating new features can drastically simplify the tree!

With a new feature $X_3 = X_2 - X_1$



$X_3 = 0 \Longleftrightarrow X_1 = X_2$

$X_3 = X_2 - X_1$

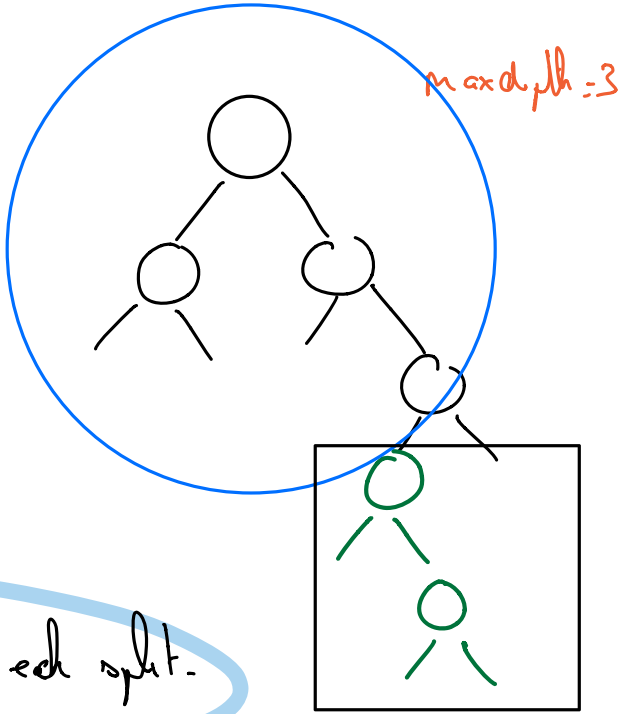
Outline

Random Forests

To put it in simple words: « random forests builds multiple decision trees and merges them together to get a more accurate and stable predictions ».
↬key idea: create variable trees + aggregate them

How to create variability ?

* different data !
* change parameters
  * max_depth
  * change purity criterion
* allow only some features at each split.



Random Forests

To put it in simple words: « random forests builds multiple decision trees and merges them together to get a more accurate and stable predictions ».
↳key idea: create variable trees + aggregate them

How to create variability ?

- Instead of the most important features, search the best feature among a random subset (ntry). ntry amongst d possible.
- Work on subsets of data (bootstrap=True).
- More `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`

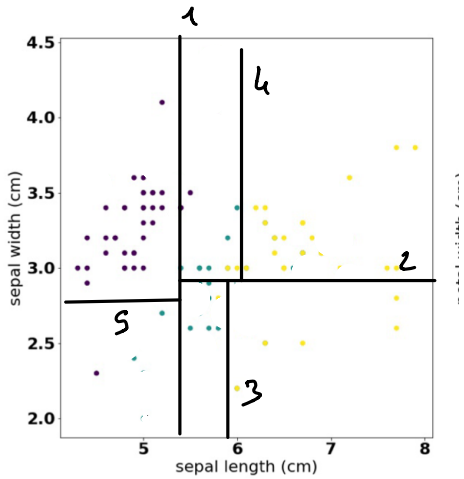
```
# Creating and fitting a Random Forest
from sklearn.ensemble import RandomForestClassifier

plt.figure(figsize=(20,10))
clf = RandomForestClassifier(n_estimators = 100,
                             max_depth=3, bootstrap = True,
                             random_state = 0)

clf.fit(X_train,y_train)
```

nb of trees in the forest

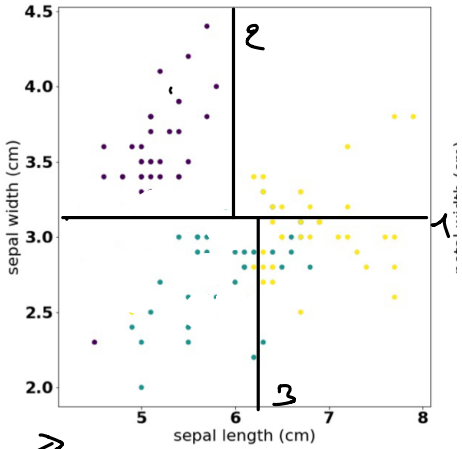
How to aggregate ? How to interpret a Random Forest ?

Bootstrap.

Tree 1



Tree n°2

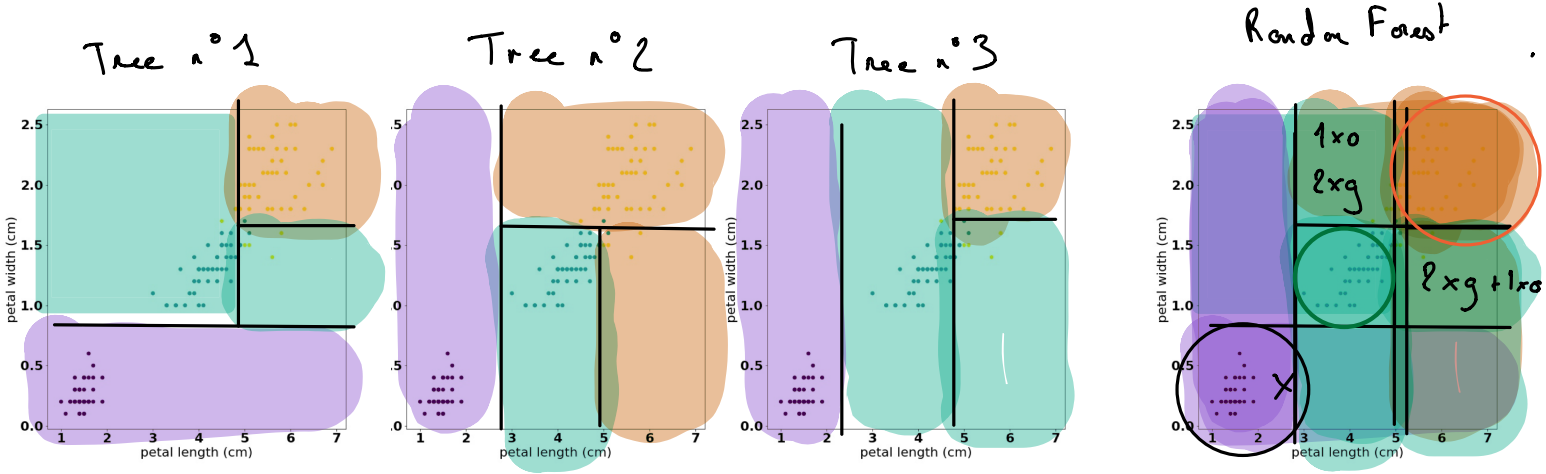


subsampled points

→ I randomly delete points in the dataset for each tree!

Aggregation

We using a voting or averaging process !



Feature importance

- how much tree nodes using a particular feature reduce impurity along the trees.
- suggests feature selection rule: drop out features with low importance to avoid overfitting.
- Attribute **feature_importance_** in RandomForestRegressor of Sklearn.

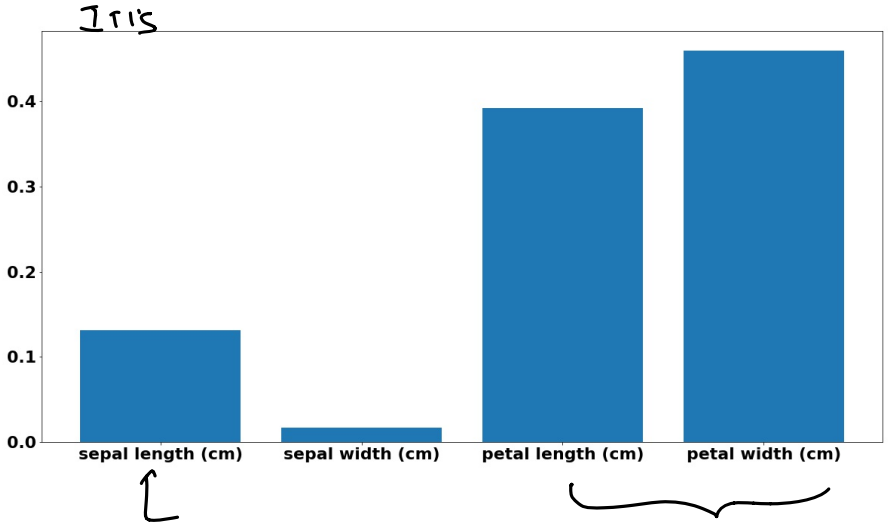


Figure: Feature importane output for a Random Forest Classifier in Python

Comparison with decision trees

Cons:

- Less interpretable
- Slower to run

Pros:

- Better in higher dimension
- More stable outputs
- Feature selection

Improving predictions - more arguments.

1. **n_estimators** : number of trees in the forest
   - improves prediction / stability
   - slows down the algorithm
2. **max_features** : maximum number of features considered to split a node
3. **min_sample_leaf** : minimum number of samples that should remain in a leaf node.

RF pros and cons

n pts. = 100 pts.

depth : 7. ⟶ $2^7 = 128$

how many samples per leaf?

$n_{est} = 100$

depth = 3 ⟶ $\log_2 n$

bootstrap = True

RF Pros:

1. works for classification and regression
2. default hyperparameters often produce reasonable prediction -> easy to use.
3. avoid overfitting if some good trees in the forest and easy feature selection -> high dimensional pb.
4. hard to beat in performance. ⟶ s.o.t.a.
5. Bonus : supports multiple outputs!

10^6 ⟶ depth $\ll 20$

$\simeq 10$

RF Cons:

1. ~~Fast~~ to train but slow to provide new predictions -> ineffective for « real-time predictions ».
2. Good for prediction but bad for description.

↳ inference time

have to compute n_{est} predict°

What about Regression ?

↳ aggregat° is by averaging instead of voting !

Questions?

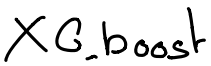
Boosting

Another import technique (very powerful !)

Main idea:

- Give more importance to difficult point iteratively
- Incrementally building an ensemble by training each new instance to emphasize the training instances previously mis-modeled.

Example: AdaBoost



See: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html`

Bonus: Multiple Outputs

- **Goal:** predict several outputs simultaneously
- **Solution:**
  - each leaf contains a value for each output
  - to chose the splits, we use a (weighted) average of the impurity of each output.

Bonus: complexity

Total complexity for one decision tree:

Worst case:

$$O(n^2 d)$$

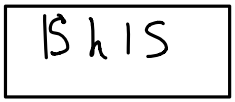
Balanced case:

$$O(nd)$$

Tips

- Decision trees tend to overfit: limit the depth or use `min_samples_leaf` (5 is a good initial pick)
- Visualize the tree with a small depth first
- Balance your dataset: trees tend to be biased towards dominating class.

Conclusion



1. Trees are one of the simplest method (the most intuitive / human like)
2. Random Forests give excellent results in many applications.

Lab :

- Example on a synthetic dataset of time series
- Application to inflation prediction in Brazil.