

Sommaire

Outils et techniques de l'IA

- **Machine Learning**
- **Les principales bibliothèques**
- **Apprentissage supervisé**
- **Apprentissage non supervisé**
- **Démarche d'apprentissage**

Machine Learning

Apprentissage artificiel

- *La but consiste à doter la machine de capacités lui permettant de tirer partie de sa propre expérience, et d'améliorer son comportement.*
- *Une branche de l'informatique en interaction avec plusieurs disciplines : Logique, Recherche opérationnelle, Neurosciences, Linguistique, Vision, Probabilités, Statistiques, Théorie des jeux, Heuristiques ...*
- *Un aspect fondamental : conception de méthodes d'apprentissage ou d'adaptation qui permettent à un système d'améliorer ses performances.*
- *On peut également citer d'autres types d'activités :*
 - *reconnaissance et interprétation des données (écriture, traitement des images, le diagnostic; la surveillance...)*
 - *aide à la décision (banque, finance, médical, militaire...)*
 - *planification d'actions et la robotique (définir des suites d'actions)*
 - *traitement du langage naturel (moteur recherche, commande vocale ...)*

Machine Learning

Apprentissage artificiel

- *Un grand nombre de méthodes de l'intelligence artificielle s'appuient sur des modules d'apprentissage, basés sur des techniques d'optimisation.*
- *L'apprentissage peut consister à extraire des règles à partir d'exemples.*

Les différentes approches

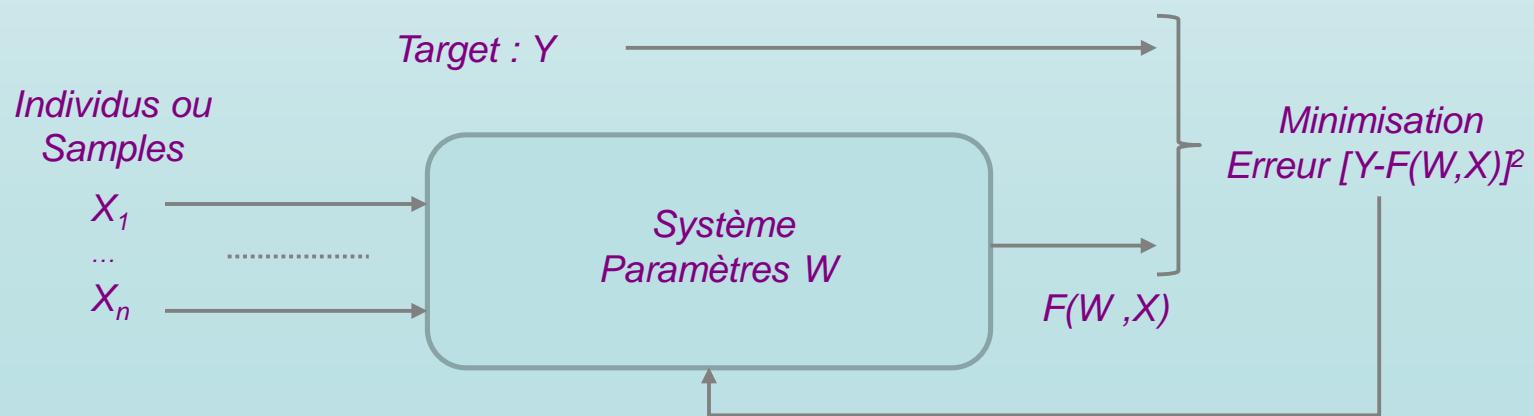
Apprentissage	Supervisé	Non-Supervisé	Evolutionnaire	Renforcement
Qu'est-ce qu'on apprend	Relations	Structures	L'optimum d'une fonction	Lois d'action
Quelles informations	Sorties désirées		Une fonction	Récompenses
Les formes d'apprentissage	Par instruction	Par observation	Par évolution et sélection	Par évaluation
Les lois d'apprentissage	Gradient ou distance	Auto-organisation	Stochastique	Différences temporelles

- *La tendance actuelle est de faire coexister dans un système plusieurs modes de représentation.*

Machine Learning

Apprentissage supervisé

- Les algorithmes connaissent les réponses qu'ils doivent obtenir et sont "guidés" par les données préalablement étiquetées (target).
- Ils peuvent être de 2 types : modèles de régression (target continue) ou de classification (target discrète).
- Les paramètres des modèles (poids pour les neurones) sont ajustés afin de diminuer l'écart entre les résultats attendus et ceux obtenus.
- Exemples : Réseaux de neurones, Support Vector Machine, Régression Linéaire, Arbres de décision, Plus proches voisins, Approches statistiques et probabilistes : Méthodes bayésiennes ...



Machine Learning

Apprentissage supervisé

- *L'objectif principal est d'apprendre une association entre les Samples et la Target à partir d'un processus d'apprentissage.*
- *Le processus d'apprentissage évalue les coefficients W de l'expression $F(W,X)$ qui permettent d'approximer au mieux la Target Y.*
- *A partir de cette expression il sera possible de calculer les sorties pour de nouvelles valeurs de X, afin d'estimer les sorties Y correspondantes.*
- *Les algorithmes d'apprentissage supervisés se divisent en deux grandes catégories : Classification ou Régression.*
- *Classifications : les targets sont des valeurs discrètes et non ordonnées (données catégorielles), elles appartiennent à une catégorie spécifique.*
- *Ces algorithmes mesurent le nombre de samples bien classés.*
- *Régression : les targets ont des valeurs numériques continues.*
- *Pour ces algorithmes les performances des modèles on cherchera à minimiser la racine carré de l'erreur quadratique moyenne (RMSE).*

Machine Learning

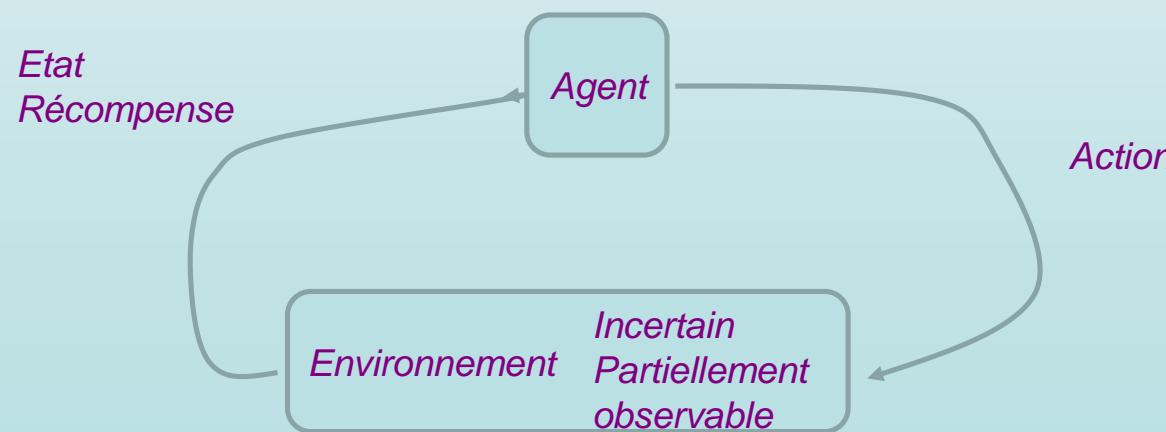
Apprentissage non supervisé

- *Dans un apprentissage non supervisé, la machine reçoit uniquement un jeu de données (samples) mais n'a pas de Target associé.*
- *Les algorithmes devront apprendre sur la base des exemples, à analyser la structures de ces données afin de réaliser certaines tâches.*
- *Les algorithmes peuvent par exemple apprendre à classer les données sur la base de ressemblances : Clustering, data-mining*
- *Les algorithmes peuvent détecter des anomalies, en identifiant les données dont la structure est très éloignées de celles des autres*
- *La réduction de la dimensionnalité qui consiste à simplifier la structure des données tout en conservant les principales caractéristiques.*
- *Exemples d'apprentissage non supervisé : Algorithme K-means clustering, Algorithme isolation forest, Analyse en composante principale...*

Machine Learning

Apprentissage par renforcement

- Le principe consiste pour un agent plongé dans un environnement d'apprendre comment se comporter.
- Les données d'apprentissage proviennent de l'environnement.
- Un agent dans un état s à l'instant t peut réaliser des actions qui le conduiront, avec une certaine probabilité, vers un état s' au temps $t+1$.
- À chaque actions ($s + a \rightarrow s'$) l'agent obtiendra une récompense, le but des algorithmes consiste à maximiser la somme des récompense.



Machine Learning

Algorithmes génétiques

- *S'inspirent de la théorie de l'évolution et des règles de la génétique qui permettent aux êtres vivants de s'adapter à leur environnement.*
- *Trois mécanismes gèrent l'évolution d'une population d'individus :*
 - *La sélection permet aux individus les mieux adaptés à l'environnement de survivre plus longtemps que les autres.*
 - *Le croisement génère des enfants qui vont hériter d'une combinaison des gènes de leurs parents.*
 - *La mutation modifie de façon aléatoire certains gènes pour les individus.*
- *Une fonction objectif complète ces mécanismes et permet de mesurer l'adéquation d'un individu à son environnement.*
- *Le croisement et la mutation font évoluer la population.*
- *Les individus sont ensuite évalués par la fonction objectif et leur survie dépendra de leur niveau d'adaptation.*

Machine Learning

Les réseaux de neurones

- Ces algorithmes peuvent être utilisés aussi bien dans des cas d'apprentissage supervisé que non supervisé.
- Ils sont performants mais nécessitent de très nombreuses informations (données textuelles, sons, images...).
- De plus leurs résultats ne sont pas facilement explicables.

Classification naïve bayésienne

- Sont des algorithmes supervisés qui supposent l'indépendance des variables, et reposent sur le théorème de Bayes.
- Les algorithmes définissent eux-mêmes des règles qui vont leurs permettent de classifier un ensemble d'individus.
- La taille des données d'apprentissage n'a pas besoin d'être importante pour que les algorithmes convergent.

Machine Learning

Arbres de décision

- *Algorithme supervisé basé sur une structure d'arbre. Les feuilles sont les valeurs à estimer, les branches les combinaisons entre les variables.*
- *L'arbre doit être vu comme l'ordre des décisions à réaliser sur les variables pour prédire une valeur spécifique de la variable à estimer.*

Forêts aléatoires (random forest)

- *random forest est composé de plusieurs arbres de décision, travaillant de manière indépendante sur des parties différentes d'un problème.*
- *Sur chaque partie un modèle d'entraînement est exécuté. Les résultats sont ensuite combinés pour obtenir la prévision la plus solide.*

Régression linéaire

- *Ces algorithmes d'apprentissage supervisé mettent en relation plusieurs variables explicatives avec des variables mesurées.*
- *L'approche consiste à supposer que les variables explicatives sont indépendantes les unes des autres.*

Machine Learning

Régression logistique

- Ces modèles supervisés permettent de prédire la probabilité qu'un événement arrive à partir des coefficients de régression.
- Ces modèles sont facilement interprétables et sont très utilisés par des applications dans le domaine de la santé ou de la finance.

Machines à vecteurs de support (SVM)

- Sont des algorithmes supervisés qui appliquent une transformation non linéaire sur les données pour identifier une séparation linéaire.
- Les transformations consistent à rechercher un espace, généralement de dimension supérieure, dans lequel on peut projeter les valeurs, où l'on pourra trouver un séparateur linéaire.
- Les SVM peuvent utiliser des classificateurs linéaires pour résoudre un problème non linéaire. Support Vector Regression (SVR) est une version de régression des SVM.

Machine Learning

K-means

- Sont des algorithmes non supervisés qui regroupent des données selon une similarité calculée à partir de leurs caractéristiques.
- Au sein d'un jeu de données, il permet de regrouper les données similaires dans K groupes (clusters).
- Une même donnée ne peut se retrouver dans deux clusters différents. Ces clusters sont constitués de façon à minimiser une certaine fonction.

Analyse en composante principale - ACP

- Sont des algorithmes non-supervisés, qui réduisent le nombre de variables d'un système en créant de nouvelles variables indépendantes.
- L'objectif est de rendre les données à la fois plus simples et plus adaptées à la modélisation.

Les bibliothèques

Les principales bibliothèques

- **Numpy**
- **Pandas**
- **Matplotlib**
- **Seaborn**

Les bibliothèques

Machine Learning & python

- *Le machine learning manipule des données (DataSet) au travers de tableaux composés d'individus (ou samples) enregistrés sur les lignes et de colonnes correspondant aux variables (ou features).*
- *Plusieurs bibliothèques permettent de traiter les DataSet.*
- ***numpy*** *permet d'effectuer des calculs numériques sur des données composées exclusivement d'entiers ou de réels.*
- ***pandas*** *est une librairie spécialisée dans l'analyse des données, elle manipule des DataSet de type DataFrame (principe d'Excel).*
- ***matplotlib*** *permet de créer et de visualiser des données sous la forme de différents types de graphiques (courbes, histogramme, box, ...).*
- ***seaborn*** *est un outil de Data Visualisation basé sur matplotlib, permettant de transformer des données en graphiques et diagrammes.*
- ***sklearn*** *est une bibliothèque pour le machine learning, elle propose également une palette de méthodes pour piloter le pre-processing des DataSets en amont des phases d'apprentissage.*

Les bibliothèques

Numpy

- Numpy manipule des tableaux (array) multi-dimensionnels. Cette bibliothèque est généralement renommée np.

```
import numpy as np
```

Importation de la bibliothèque

- L'initialisation d'un matrice numpy peut se faire par la lecture des données sur des fichiers de textes (loadtxt) ou en utilisant une des fonctions telles que array(), zeros(), random.randint()

```
Tab = np.zeros((5,5))
```

Création d'une matrice 5*5 de zeros

```
# Création d'une matrice 5*5 composé de valeurs aléatoire comprises entre 1 et 10
```

```
Tab = np.random.randint(1, 10,(5,5))
```

- Plusieurs fonctions mathématiques sont accessibles (sin, sqrt, exp, ...)

```
Tab = np.linspace(0,10,11)  
np.exp(Tab)
```

Création d'un tableau de 11 valeurs

Calcul de l'exponentielle de toutes les valeurs

- L'attribut shape donne un accès à la taille du tableau

Les bibliothèques

Pandas

- *Pandas est une bibliothèque qui permet de traiter des données brutes et volumineuses en tables (Series et DataFrame) adaptées pour les modèles d'apprentissage.*
- *Bien que très lié à numpy les données dans Pandas ne sont pas forcément homogènes (string, réels, entiers, date ...).*
- *La bibliothèque Pandas est un outil essentiel pour la Data Science, elle permet : le nettoyage, la transformation, l'analyse, la modélisation, la visualisation et le reporting des données.*
- *Pandas est moins bien adapté aux traitements de données telles que les images, les fichiers audio ou certaines données textuelles.*
- *Pandas permet de manipuler deux types de données les DataFrames et les Series*
- *Les Series sont des tables à une seule dimension généralement utilisées pour traiter des séries temporelles.*
- *Les DataFrames sont des tableaux composés de Series.*

Les bibliothèques

DataFrame

- Les *DataFrames* sont vues comme des collections de colonnes (*features*) étiquetés via un *label*.
- Une ligne des *DataFrames* représente un individu (*sample*) qui est repéré par un *index*.

Diagram illustrating the structure of a DataFrame:

	features ou variables							
	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
3	Adelie	Torgersen	Nan	Nan	Nan	Nan	Nan	2007
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	female	2007
...

Annotations:

- Index:** Points to the row index (0, 1, 2, 3, 4) on the left.
- features ou variables:** Points to the column headers at the top.
- Label:** Points to the column header "species".
- samples ou individus:** Points to the data rows (rows 0-4).

Les bibliothèques

Accès aux informations

- Tout type de liste, de matrice ou de dictionnaire peut être transformé en un DataFrame. Les données peuvent également être récupérées à partir de fichiers à différents formats : (csv, json, xml, html, sql ...)

```
import pandas as pd          # Importation de la bibliothèque
# Création d'un DataFrame à partir de listes
Tab = np.linspace(0,10,11)    # Création d'une série de valeurs
data = pd.DataFrame(data={'valeur' : tab, 'exp' : np.exp(tab)})
# Lecture de données à partir de fichiers en spécifiant le caractère séparateur
data_penguins=pd.read_csv('Penguins.csv', delimiter=',')
```

- Principaux attributs sur le DataFrame

<code>data_penguins.shape</code>	# taille du dataframe : (344, 8)
<code>data_penguins.columns</code>	# noms des colonnes : (['species', 'island', ...])
<code>data_penguins.index</code>	# index : RangeIndex(start=0, stop=344, step=1)
<code>data_penguins.dtypes</code>	# affiche le type de chaque features

- L'accès aux features se fait via l'instruction `dataframe[label]`
- L'accès aux samples par l'instruction `dataFrame.loc[index]`

Les bibliothèques

Les features numériques

- La méthode `describe()` produit une description statistique des données numériques (valeurs non nulles, moyenne, min, max, écart type ...)
- Elle permet de savoir s'il y a des valeurs manquantes, des outliers ...
- La méthode `corr()` renvoie une matrice de corrélation pour chaque variables numériques entre elles.

Les features object

- La méthode `value_counts()` compte le nombre d'occurrences de chaque valeur unique. Cette méthode peu également s'appliquer à des features de type `int`, mais n'a pas d'intérêt sur les features de type `float`.

```
data_penguins['species'].value_counts()      # Affichage des différentes espèces
Adelie 152
Gentoo 124
Chinstrap 68
```

- Les méthodes `unique()` et `nunique()` retournent la liste des valeurs uniques ainsi que le nombre total de valeurs différentes.

Les bibliothèques

Sélection des données

- *L'indexation booléenne est une fonctionnalité puissante qui permet de filtrer des données d'un DataFrame à l'aide d'un vecteur de booléens.*
- *Le vecteur de booléens est passé dans l'opérateur d'indexation ([]) du DataFrame.*
- *Afin de filtrer les données on utilise en général un critère de comparaison sur une ou plusieurs features du DataFrame afin d'obtenir en retour un vecteur de booléens.*
- *Ce vecteur est ensuite utilisé dans l'opérateur d'indexation du DataFrame pour sélectionner uniquement les données souhaitées.*

```
# recherche des valeurs manquantes sur la variable sex
df = data_penguins['sex'].isna() # on obtient un tableau de booléens
# sélection des samples ayant une valeur manquante dans la variable sex
data_penguins[df]
```

```
df = data_penguins['body_mass_g'] >= 6000] # individus les plus gros
data_penguins[df]
```

Les bibliothèques

Modification des données

- La méthode `apply()` permet d'appliquer une fonction à chaque données sur les lignes ou les colonnes d'un `DataFrame`.
- Les transformations à appliquer sont passées en argument à la méthode sous la forme de fonctions ou de `lambda` fonction.
- Comme la plupart des méthodes d'apprentissage utilisent uniquement des données numériques, il est nécessaire de modifier les variables de type `object`, `boolean`, ... en données numériques
- La méthode `apply()` permet d'effectuer ces modifications, bien que `sklearn` dispose de méthodes adaptées pour le faire.

Modification de la variable `sex` via une fonction anonyme

```
data_penguins['sex']=data_penguins['sex'].apply(lambda x : 0 if x=='male' else 1)
```

Modification de la variable `island` via une fonction

```
data_penguins['island']=data_penguins ['island'].apply(code_island)
```

```
def code_island(x):  
    if x=='Biscoe' : return 0  
    if x=='Dream' : return 1  
    return 2
```

Les bibliothèques

Principales méthodes et fonctions

- La méthodes `drop()`, `dropna()` ... permettent de supprimer des variables (option `axis=1`) ou des individus (`axis=0`).
- Les méthodes `sort_values()` et `sort_index()` permettent de trier les DataFrames par rapport aux valeurs des variables ou sur les index.
- Les méthodes `sum()`, `mean()`, `max()`, `min()` ... permettent d'accéder aux informations statistiques sur les variables ou les individus.
- La méthode `groupby()` permet de regrouper des individus en fonction des valeurs d'une variable ou d'un ensemble de variables.

Calcul des valeurs moyennes des pingouins en fonction du sex
`data_penguins.groupby(['sex']).mean()`

- La fonction `pandas.cut()` est utilisée pour séparer les éléments en différents bacs. Cette fonction est principalement utilisée pour effectuer une analyse statistique sur des données scalaires.

Découpage des individus en 20 bacs sur la base de leur masse corporelle
`pd.cut(data_penguins['body_mass_g'], bins=20).value_counts().sort_index()`

Les bibliothèques

Matplotlib

- La bibliothèque `matplotlib` est indispensable à `python` pour effectuer des tracés de courbes.
- Quel que soit le style de programmation, il est nécessaire d'importer le module `pyplot` de `matplotlib`. On lui donne en général l'alias `plt`.
- L'instruction `plot()` permet de tracer des courbes qui relient des points dont les abscisses et ordonnées sont fournies en arguments.
- La méthode `subplots()` est utilisée pour créer une ou plusieurs grilles de sous-graphiques (axes) dans une seule figure.

```
Import matplotlib.pyplot as plt
fig, ax = plt.subplots(2,2)
X = np.arange(0.0, 5.0, 0.02)
ax[0,0].plot(X, (X-2)*(X-1)*(X-3))
ax[0,1].plot(X, 1/(X+1))
X = np.arange(0.0, 5.0, 0.2)
ax[1,0].scatter(X, 2*np.cos(X), marker='.')
ax[1,1].scatter(X, 2*np.sin(X), marker='.'
```

Les bibliothèques

Seaborn

- Seaborn propose plusieurs fonctions permettant de visualiser les relations entre les variables ainsi que les liens statistiques qui existent.
- Seaborn est une surcouche de Matplotlib, elle permet de générer des graphiques de qualité et est bien adaptée à l'utilisation des dataFrames.
- En tant que surcouche de Matplotlib, il y a de grandes ressemblances entre les deux librairies.
- Les fonctions reçoivent les dataframes utilisés pour tracer les graphiques, ainsi que les variables à afficher.
- La plupart des graphiques dispose d'un paramètre hue, permettant d'afficher les graphiques en fonction d'une variable catégorielle.

```
import seaborn as sns # Importation de la bibliothèque
# Lien entre la masse corporelle et la longueur de nageoire en fonction du sexe
sns.scatterplot(data=Data_csv, x='body_mass_g', y='flipper_length_mm', hue='sex')
# Lien entre la masse corporelle et la longueur de nageoire en fonction de l'espèce
sns.scatterplot(data=Data_csv, x='body_mass_g', y='flipper_length_mm',
hue=species')
```

Les bibliothèques

Apprentissage : scikit-learn

- Scikit-learn : est une librairie d'apprentissage automatique qui contient tous les algorithmes du machine learning.

#Apprentissage supervisé

Perceptron multicouche (réseau de neurones)
Machines à vecteur de support (SVM)
K plus proches voisins (KNN)
Arbres de décision (Decision Tree)
Régression linéaire (Linear Regression)
Régression logistique (Logistic Regression)
Classification naïve bayésienne (Naive Bayes)

#Apprentissage non-supervisé

Partition en k moyennes (K-means clustering)
Regroupement hiérarchique (CAH)
Processus Gaussien de classification (GPC)
Processus Gaussien de régression (GPR)
Réduction de dimension (ACP)

- Le processus d'apprentissage supervisé est toujours le même :

#ETAPE 1 : Création d'un estimateur + Entrainement
`model = 'estimateur.scikit-learn' ('hyper-paramètres')`
`model.fit (X, y)`

#ETAPE 2 : Prédiction
`ypred = model.predict(X)`

#ETAPE 3 : Evaluation du modèle
`model.score(X, y)`

Apprentissage supervisé

Apprentissage supervisé

- Régression linéaire
- Support Vector Machine
- KNeighbors
- Arbres de décision

Régression linéaire

Présentation

- La régression linéaire est avec l'algorithme du perceptron un des algorithmes le plus utilisé en machine learning.
- Un modèle de régression linéaire est un outil statistique très utilisé pour l'étude de données multidimensionnelles.
- La target (Y) est quantitative tandis que les samples peuvent être quantitatives ou catégorielles (qualitatives).
- Pour une régression linéaire la target dépend uniquement des features ($1, X^1, \dots, X^p$). Si l'on a n samples les sorties y_i sont obtenues :

$$y_i = \beta_0 + \beta_1 x_i^1 + \dots + \beta_p x_i^p + \varepsilon_i \text{ avec } i = 1 \dots n$$

- les β_j ($j=0$ à p) sont les poids du modèle que l'on doit estimer.
- Les ε_i sont des termes d'erreurs (résidus) commis par le modèle.
- Pour qu'un modèle de régression linéaire soit acceptable les erreurs doivent être indépendants et normalement distribuées
- Enfin les β doivent être constants.

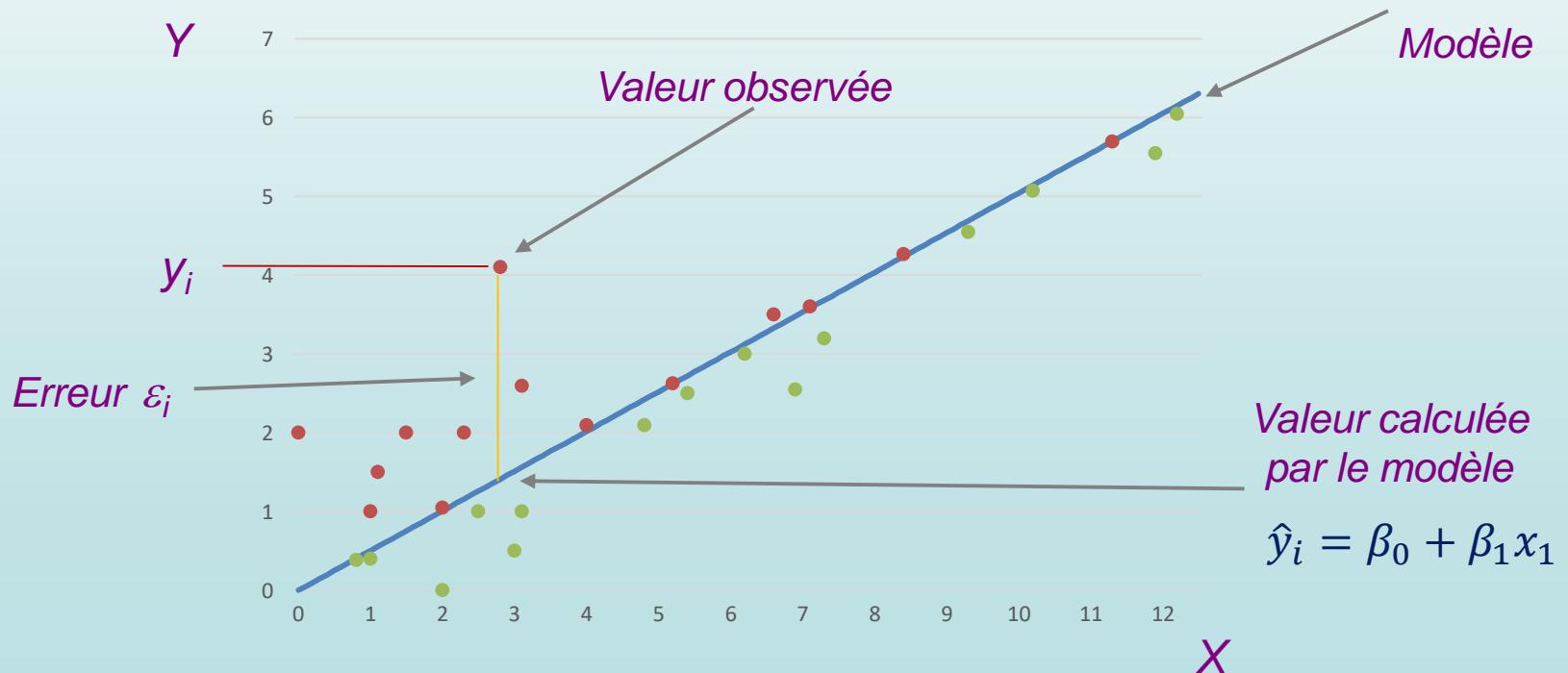
Régression linéaire

Interprétation

- De la représentation linéaire on en déduit que l'impact de X^i sur Y ne dépend que des paramètres β_j .

Exemple avec un seule feature X : $p=1$

$$\frac{\partial Y}{\partial X^j} = \beta_j$$



Régression linéaire

Estimation des coefficients

- Le modèle d'apprentissage consiste à chercher les valeurs de β_j qui minimisent la somme des erreurs commises sur les samples.
- La régression linéaire consiste à minimiser l'erreur quadratique :

$$\min \left(\sum_{i=1}^n (\varepsilon_i)^2 \right) = \min \left(\sum_{i=1}^n (y_i - \beta_0 + \beta_1 x_i^1 + \dots + \beta_p x_i^p)^2 \right) = \min \|Y - X\beta\|^2$$

- Le minimum est obtenu lorsque la dérivée de :

$$\|Y - X\beta\|^2 = (Y - X\beta)^t (Y - X\beta) = Y^t Y - 2\beta^t X^t Y + \beta^t X^t X \beta$$

- S'annule soit pour : $X^t y - X^t X \beta = 0$
- Si la matrice $X^t X$ est inversible on obtient le minimum $\hat{\beta} = (X^t X)^{-1} X^t y$.
- On en déduit ensuite les valeurs prédictes : $\hat{Y} = X(X^t X)^{-1} X^t Y$

Régression linéaire

Régression linéaire sous Excel

Modèle	Prix	Cylindrée	Puissance	Poids	Conso
Dalhatsu Cuore	11600	846	32	650	5,7
Suzuki Swift 1.0 GLs	12490	993	39	750	5,8
Fiat Panda Mambo L	10450	899	29	730	6,1
VW Polo 1.4 60	17140	1390	44	955	6,5
Opel Corse 1.2i Eco	14825	1195	33	895	6,8
Subaru Vivio 4WD	13730	658	32	740	6,8
Toyota Corolla	19490	1331	55	1010	7,1
Ferrai 456 GT	285000	5474	325	1690	21,2
Mercedes S 600	183900	5987	300	2250	18,7
Maserati Ghibli GT	92500	2789	209	1485	14,5
Oprel Astra 1.6i 16v	25000	1597	74	1080	7,4
Peugeot 306 XS 108	22350	1761	74	1100	9
Renault Safrane 2.2 V	36600	2165	101	1500	11,7
Seat Ibiza 2.0 GTI	22500	1983	85	1075	9,5
VW Golf 2.0 GTI	31580	1984	85	1155	9,5
Citroen ZX Volcano	28750	1998	89	1140	8,8
Fiat Escort 1.4i PT	22600	1580	65	1080	9,3
Ford Escort 1.4i PT	20300	1390	54	1110	8,6
Honda Civic joker 1.4	19900	1396	66	1140	7,7
Volvo 850 2.5	39800	2435	106	1370	10,8
Ford Fiesta 1.2 Zetec	19740	1242	55	940	6,6
Hyundai Sonata 3000	38990	2972	107	1400	11,7
Lancia K 3.0 LS	50800	2958	150	1550	11,9
Mazda Hatchback V	36200	2497	122	1330	10,8
Mitsubishi Galant	31950	1998	66	1300	7,6
Opel Omega 2.5i V6	47700	2496	125	1670	11,3
Peugot 806 2.0	36950	1998	89	1560	10,8
Nissan Primera 2.0	26950	1997	92	1240	9,2
Seat Alhamra 2.0	36400	1984	85	1635	11,6
Toyota Previa salon	50900	2438	97	1800	12,8
Volvo 960 Kombi aut	49300	2473	125	1570	12,7

Variables observées X^1, X^2, X^3, X^4

Variable Y a estimer

Statistiques de la régression

Coef multiple σ	0.9773
Coef σ^2	0.9551
Coef ajusté σ^2	0.9482
Ecart type	0.8096

SCE/SCT
Y est bien expliqué

	D° liberté	Som carrés
Régression	4	362.8341
Résidus	26	17.0432
Total	-20	379.8774

SCE, SCR, SCT

	Coefficients	Probabilité
Constante	2.4636	0,00047
Prix	2.0011E-05	0,02893
Cylindrée	-0.00050	0,38331
Puissance	0.0251	0,01754
Poids	0.0042	5.44E-05

Valeurs $\hat{\beta}$

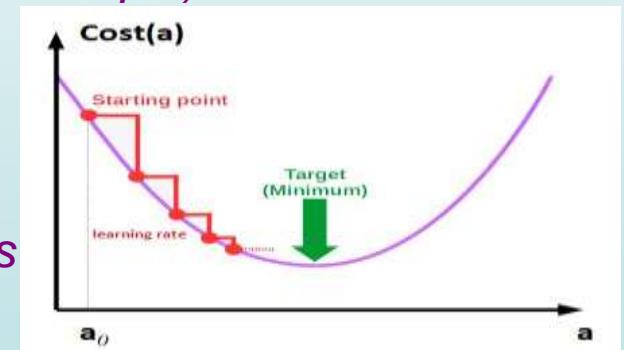
Régression linéaire

Régression linéaire : Descente de gradient

- Inverser une matrice est impossible lorsque la taille de X est importante, les algorithmes utilisent plutôt des descentes de gradient.
- Pour le Machine et le Deep Learning, la descente de gradient est un des algorithmes les plus importants (Régression, réseaux de Neurones ...).
- C'est un algorithme d'optimisation qui permet de trouver le minimum de n'importe qu'elle fonction convexe (erreur quadratique)

La descente de gradient

- La suite des valeurs des a^j convergera vers un minimum si l'on se déplace d'une petite distance η (learning rate) dans la direction de la dérivée de la fonction.

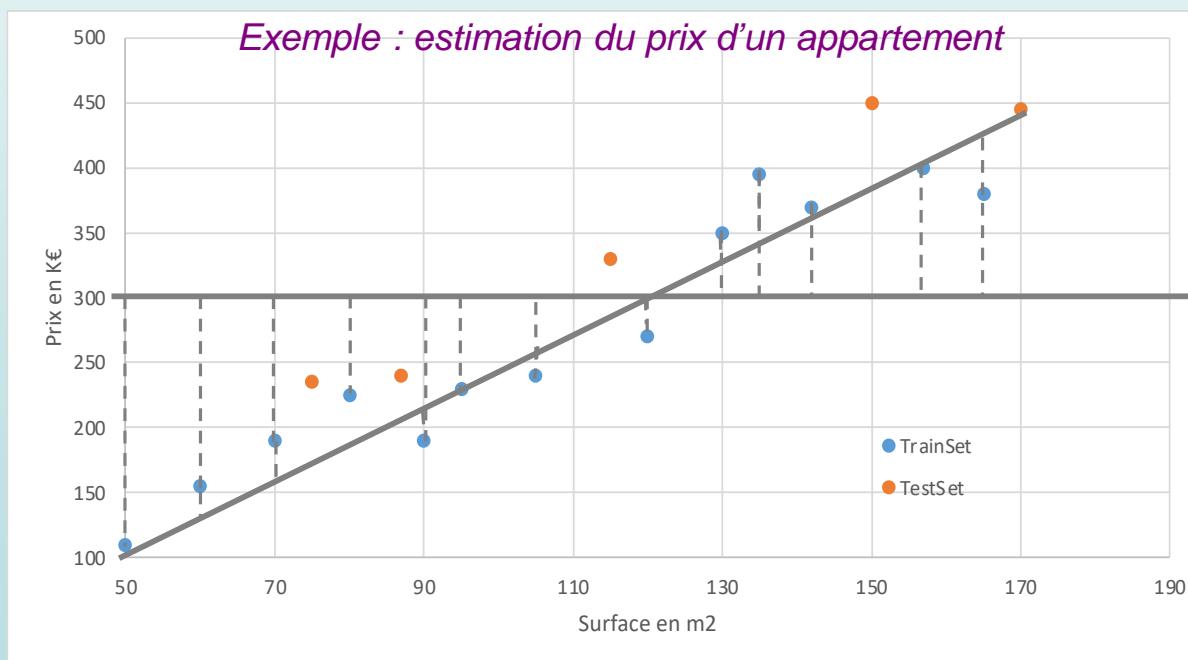


$$a^k = a^{k-1} - \eta \frac{\partial F(a)}{\partial a}$$

Régression linéaire

Mesure de la qualité d'un modèle

- Après l'apprentissage les modèles sont utilisés pour des prédictions.
- On utilise une partie des données (*TrainSet*) pour « caler » les paramètres du modèle et une partie (*TestSet*) pour le valider.
- La qualité d'un modèle doit être mesurée sur la partie *TestSet*.



A partir d'un modèle de type
 $\text{Prix} = ax + b$
 (x surface – a et b paramètres)

Le but est de trouver a et b qui minimisent les erreurs sur le TrainSet

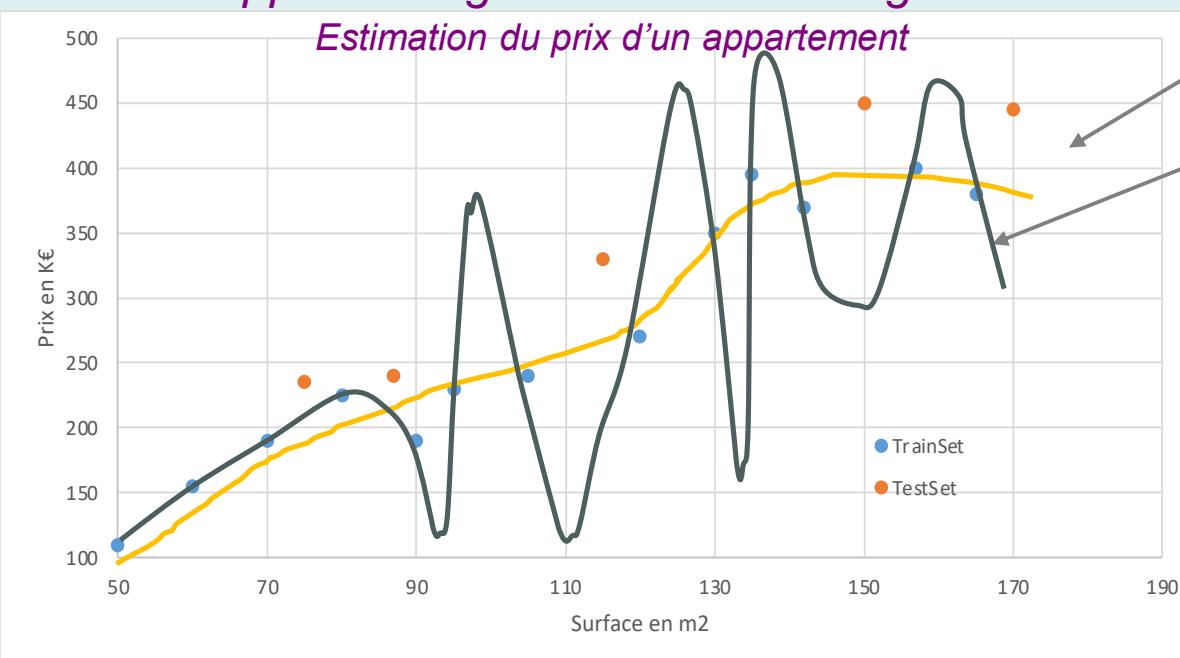
Pour a=0 et b=300 on a des erreurs (190, 145, 110 ..., 100, 80) = 1185

Pour a=2,9 et b=-45,8 on a (10, 25,8, 110 ..., 12, 55,4) = 320,4

Régression linéaire

Mesure de la qualité d'un modèle

- Il est possible d'augmenter la dimension d'un problème pour que l'algorithme d'apprentissage « colle » mieux au TrainSet.
- Pour cela il suffit d'ajouter des features correspondant à $X^2, X^3, \dots X^n$
- Les erreurs sur le TrainSet tendent alors vers 0, mais un trop grand apprentissage conduit à une dégradation des résultats sur le TestSet.



Equation de degré 4

Equation de degré 13

Le modèle de degré 13 à bien appris sur le TrainSet, mais les erreurs commises par rapport à un modèle de degré 4 sur le TestSet sont plus grandes

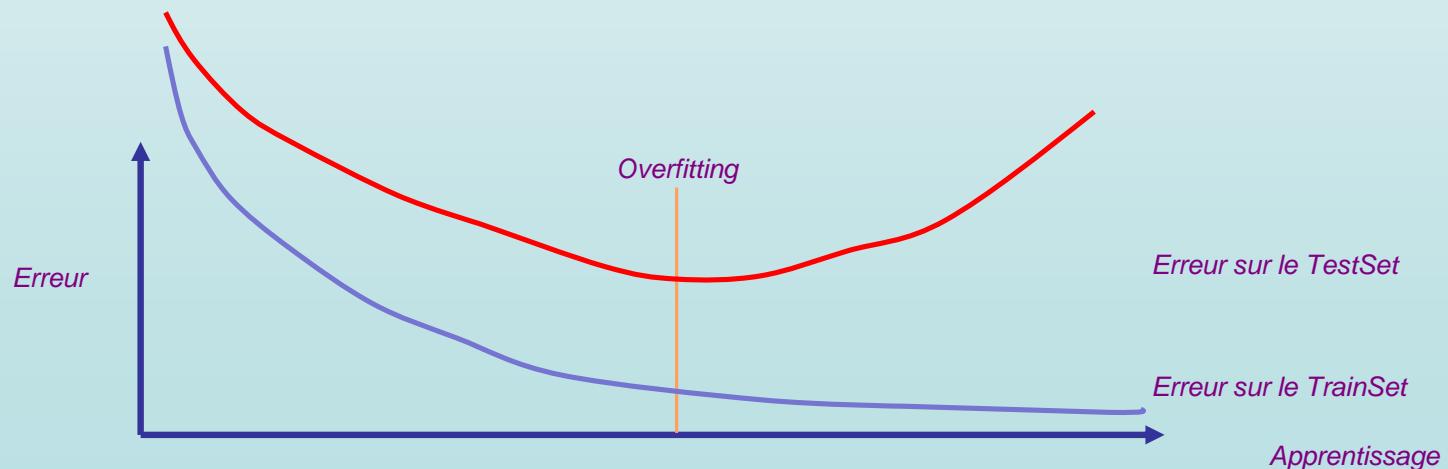


Overfitting

Régression linéaire

Mesure de la qualité d'un modèle : Overfitting ou Underfitting

- Des causes de mauvaises performances en Machine Learning sont dues à du sur ou du sous apprentissage (Overfitting et Underfitting).
- L'overfitting se produit lorsqu'un modèle s'adapte très bien au TrainSet.
- Dans ce cas on apprend les corrélations générales du modèle mais également le "bruit" dû aux imprécisions sur les données.
- Le modèle est alors parfaitement adapté aux données du TrainSet mais prédira très mal les sorties des données non encore vues : TestSet.



Régression linéaire

Régression linéaire : scikit-learn

- Exemple prix des locations

```
# Lecture des données
```

```
dt=pd.read_csv('locations.csv', index_col=0)
```

```
# Ajout de la colonne unité
```

```
dt['one']=np.ones(dt.shape[0])
```

```
X = dt.drop(['loyer'], axis=1)
```

```
Y = dt['loyer']
```

- Exemple : modèle non linéaire

```
# Création des données
```

```
x=np.linspace(-2,2,100)
```

```
noise = np.random.normal(0,0.3,100)
```

```
dt=pd.DataFrame(x, columns=['x'])
```

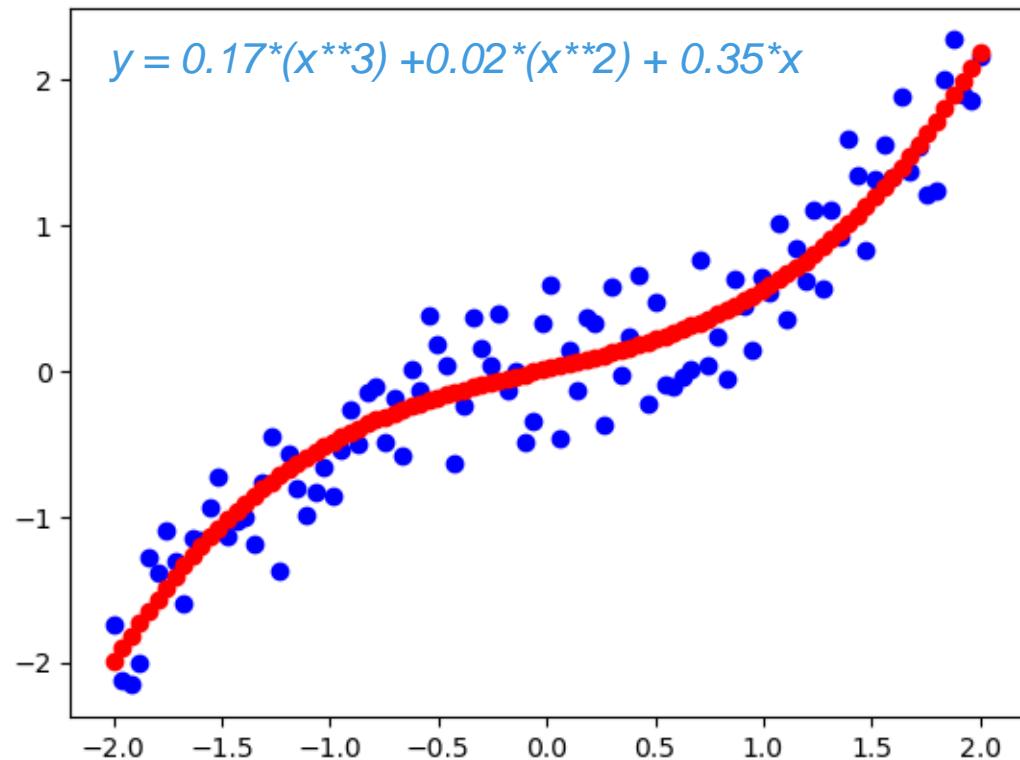
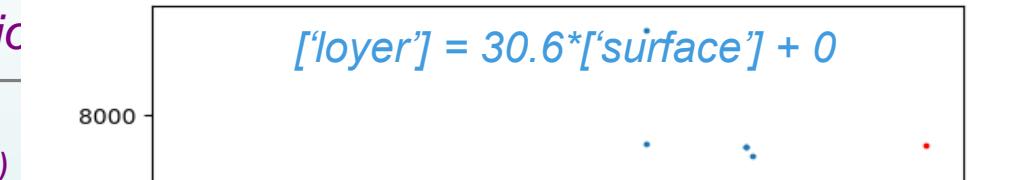
```
dt['y']=(dt['x']*abs(dt['x'])/2)+noise
```

```
dt['one']=np.ones(dt.shape[0])
```

```
dt['x**2']=dt['x']**2
```

```
dt['x**3']=dt['x']**3
```

Score = 0.82



Support vector machine

Machines à vecteurs de support (SVM)

- Les SVM (séparateurs à vastes de marges) sont des algorithmes initialement définis pour la prévision d'une variable qualitative, dont le principe est de séparer les données à l'aide d'une frontière.
- Ces algorithmes tentent de trouver un hyperplan de telle sorte que la distance entre les différents groupes et la frontière soit maximale.
- La fonction de décision permet d'associer à chaque sample sa classe.
- Les SVM ont été généralisés à la prévision des variables quantitatives.
- Contrairement à la plupart des algorithmes d'apprentissage, les SVM peuvent être plus pénalisés par le nombre de samples que de features.
- Aussi, si le nombre de samples est "faible" on utilise des SVM pour l'apprentissage et des RdN lorsque le DataSet est conséquent.
- Cependant, il existe des versions performantes (en temps de calcul) qui permettent de prendre en compte des bases de données volumineuses.
- En fonction du Dataset, ces algorithmes ont des performances de même ordre, voir supérieures, à celle d'un RdN.

Support vector machine

La notion de marge maximale

- Les SVM reposent sur deux principes :
 - La notion de marge maximale : distance entre la frontière et les échantillons les plus proches (vecteurs supports).
 - Si les données ne sont pas linéairement séparables, on transforme l'espace de représentation en un espace de plus grande dimension, dans lequel il existe une séparation linéaire.
 - Pour rendre les données linéairement séparables on utilise des fonctions appelées fonctions noyaux.
- Les SVM sont des séparateurs linéaires, la frontière est une droite qui maximise sa distance avec les points les plus proches de cette frontière.
- Ces points sont d'ailleurs appelés vecteurs support.
- Comme la quasi totalité des cas ne sont pas linéairement séparables, on augmente la dimension et on reconsidère le problème dans cet espace là.

Support vector machine

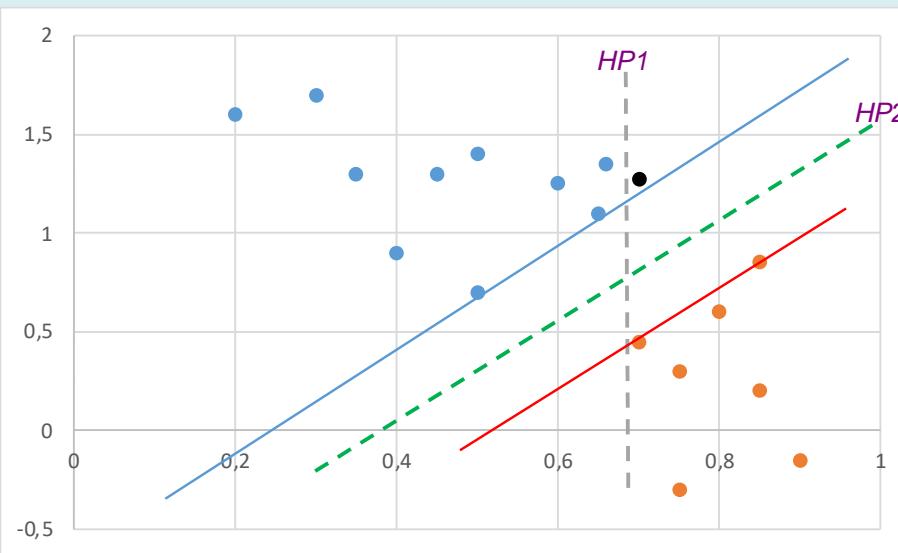
La notion de marge maximale

- Pour des classes linéairement séparables, une marge ferme est utilisée.
- Dans le cas contraire une marge souple peut être calculée à partir d'un changement d'espace de dimension.

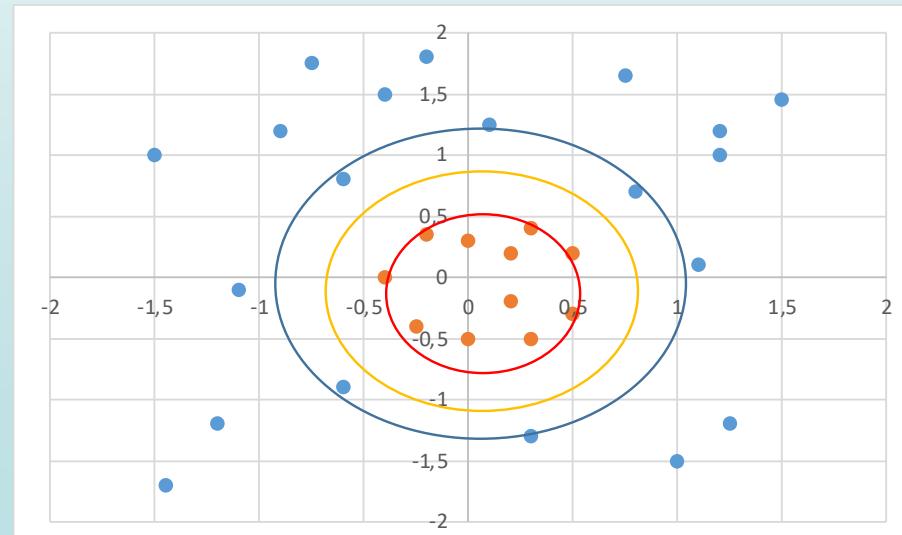
A partir de l'hyperplan HP1, le point ($x=0.69$, $y=1.2$) serait classé dans la classe rouge, avec un forte chance d'être classé de manière incorrecte.

L'hyperplan HP2 pour lequel la distance entre les groupes est maximale, à plus de chance de classer correctement le point ($x=0.69$, $y=1.2$).

En reportant les points dans un espace à 3 dimension tel que $z = x^2 + y^2$, il existe alors un hyperplan (par exemple : $-0.1x-0.1y+z-0.56=0$) qui sépare les deux classes.



Exemple 1 : Marge ferme



Exemple 2 : Marge souple

Support vector machine

Outils python pour la classification

- Afin d'évaluer un modèle il est nécessaire de prédire les résultats sur des données différentes de celles qui ont servie pour l'apprentissage.
- scikit learn propose une fonction permettant de découper un DataSet en un jeu de TrainSet et de un jeu de TestSet.
- La fonction reçoit les données X, la target Y et un taux de découpage.

```
# Fonction de découpage d'un DataSet du module model_selection
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split( X , Y, test_size=taux)
```

- La qualité d'un modèle de classification peut se mesurer en utilisant une matrice de confusion qui affiche comment sont classés les individus.
- La matrice de confusion compare les résultats attendus Y avec les résultats estimés par la fonction prédiction.

```
# Matrice de confusion du module metrics
from sklearn.metrics import confusion_matrix
```

Support vector machine

Classement des iris (SVC)

- scikit-learn propose plusieurs DataSets, en particulier un jeu de données d'Iris classés par un botaniste Ronald Fisher en 1936.
- Cette base contient 150 samples et 5 features. 4 features représentent la longueur et la largeur des pétales et de sépales et la dernière l'espèce.

Chargement du DataSet

```
from sklearn
iris = load_iris()
```

Découpage des données (2/3, 1/3)

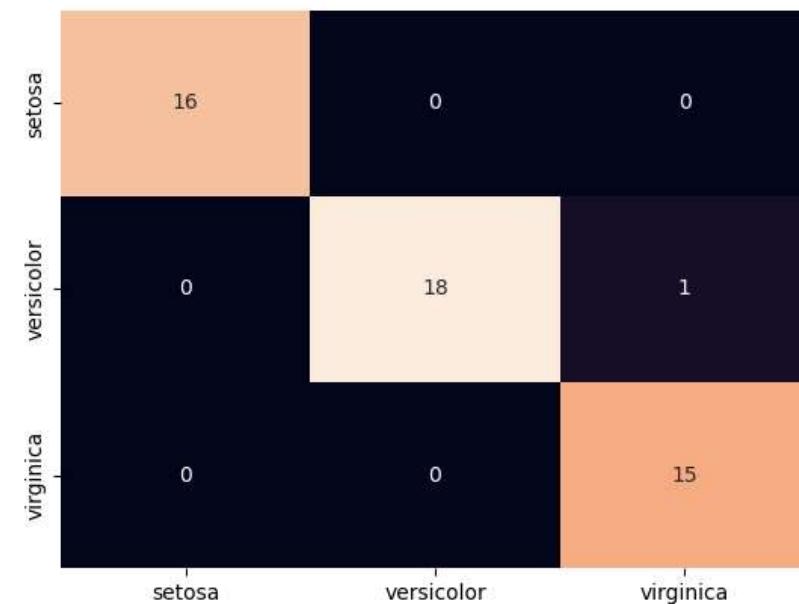
```
from sklearn
X_train, X_test,
```

Utilisation d'un modèle de SVM

```
from sklearn
model = SVC()
Y_pred = model
```

Evaluation des classements

```
from sklearn
sns.heatmap(
```



Support vector machine

Utilisation des SVM

- *Un SVM nécessite d'augmenter la taille du domaine si les données ne sont pas linéairement séparable.*
- *Lorsque les informations sont importantes (> 1Million) il peut être nécessaire d'augmenter considérablement le domaine pour obtenir des résultats acceptables.*
- *Le nombre de paramètres va alors augmenter considérablement.*
- *Les temps de calcul deviennent donc trop important.*
- *Il est préférable d'utiliser des algorithmes moins gourmands en temps de calcul et donc bien plus efficaces.*
- *Les réseaux de neurones ou les régression logistiques sont alors bien plus adaptés.*
- *De même si les données sont non structurées (images, textes, son...) les réseaux de neurones sont bien adaptés alors qu'un SVM ne l'est pas.*

Support vector machine

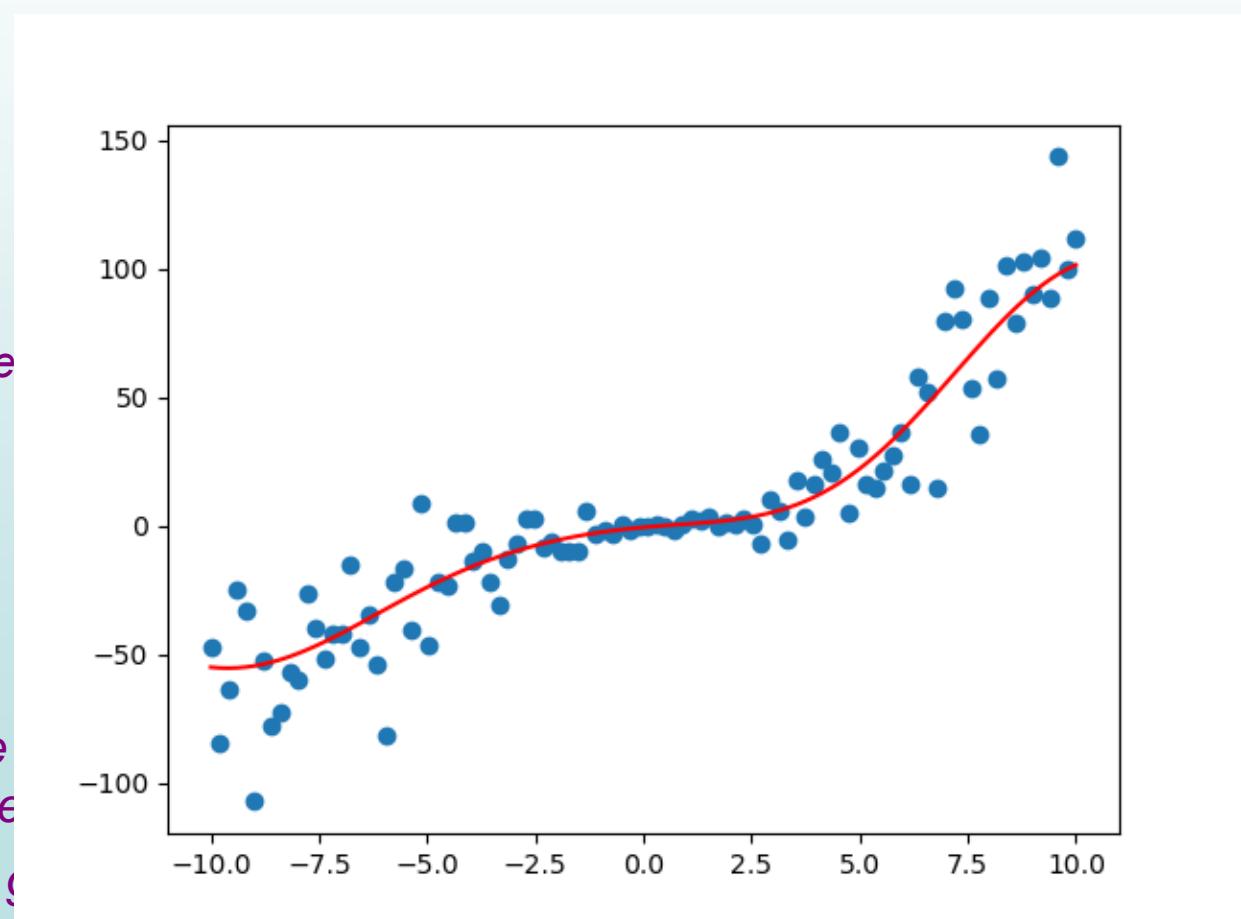
Régression polynomiale (SVR)

*Création d'un
dataSet*

*Utilisation d'un modèle
de régression*

Affichage

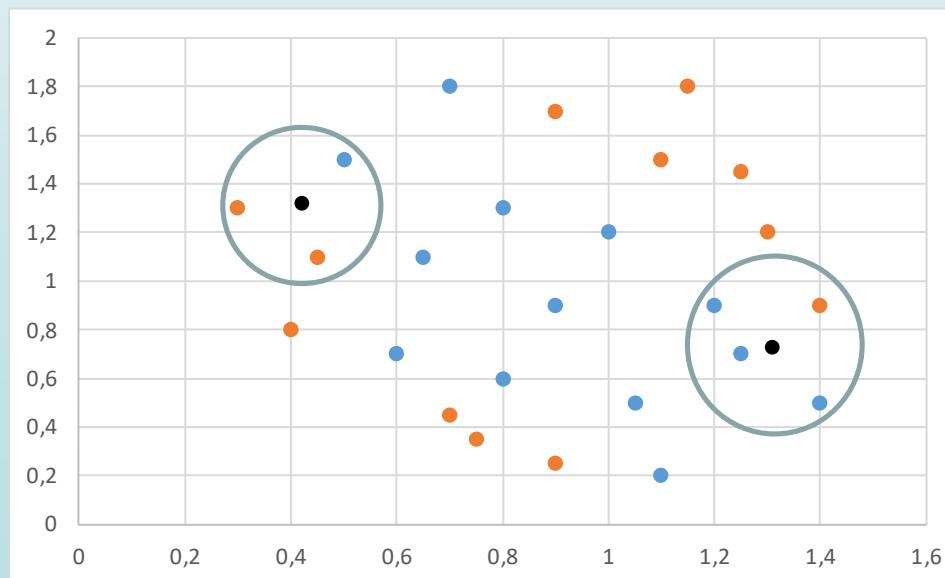
- *Le paramètre compromis entre la précision et l'importance des estimations sur la fonction noyau.*
- *Le paramètre qui détermine l'importance des estimations sur la fonction noyau.*



K Neighbors

K plus proches voisins : K-Neighbors

- *K voisins est une méthode supervisée, pour la classification ou la régression lorsque les données ont des étiquettes discrètes ou continues.*
- *Cet algorithme ne nécessite pas de phase d'apprentissage, il classe une donnée en fonction de sa distance par rapport à d'autres données.*
- *L'algorithme calcule la distance de la donnée à ses k voisins les plus proches, puis prédit la classe de l'individu par un vote.*



Pour $k=3$, la donnée $(0.43, 1.32)$ a deux voisins rouges et un bleu. Cette donnée est alors classée dans le groupe rouge

Pour $k=4$, la donnée $(1.3, 0.73)$ a trois voisins bleus et un rouge. Cette donnée est alors classée dans le groupe bleu

K Neighbors

K-Neighbors – mesure de distance

- Le choix de la mesure de distance peut impacter les résultats.
- La distance Euclidienne est la mesure la plus couramment utilisée, elle réalise une mesure en "ligne droite" entre deux individus X et Y :

$$d(X, Y) = \sqrt{\sum_{i=1}^x (x_i - y_i)^2}$$

$X = (x_1, x_2, \dots, x_n)$
 $Y = (y_1, y_2, \dots, y_n)$

- La distance de Manhattan mesure la valeur absolue entre deux points.

$$d(X, Y) = \sum_{i=1}^x |x_i - y_i|$$

- Il existe d'autres distances : Distance de Minkowski qui est une généralisation des distances Euclidienne et de Manhattan, Distance de Hamming plutôt utilisé pour des vecteurs de booléens...

K Neighbors

Classement des iris (K-Neighbors)

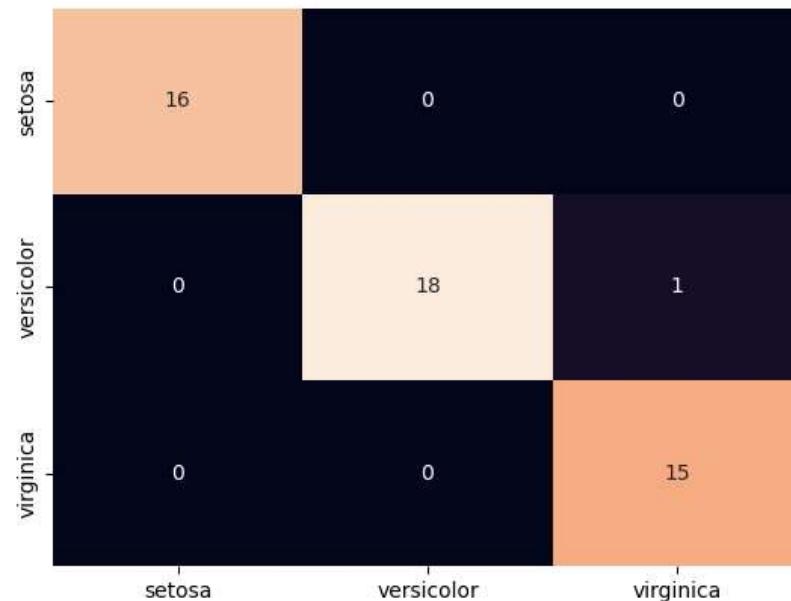
- L'utilisation d'un KNeighborsClassifier permet de classifier les données.
- Par défaut ce modèle utilise 5 voisins et une distance Euclidienne comme hyper paramètres.

Utilisation d'un modèle de K-NC

```
from .  
mode  
mode  
Y_pre  
  
impot  
sns.h  
plt.sh
```

Utilisation de seaborn pour l'affichage

- Le modèle est sensible au
- Avec un nombre de n_{neigh} étant de type "virginica".



e

K Neighbors

Utilisation des K-Neighbors

- *Un K-Neighbors est facile à implémenter et très simple à comprendre.*
- *Il s'adapte facilement à mesure que de nouveaux individus sont ajoutés et permet de les prendre en compte sans avoir à réentraîner le modèle.*
- *Seulement deux hyper paramètres sont nécessaires pour définir le modèle : la valeur k et une mesure de distance.*
- *Mais les K-Neighbors utilisent plus de mémoire de stockage de données par rapport aux autres approches.*
- *Ces algorithmes fonctionnent mal avec des entrées de données de grande dimension. Le nombre d'erreurs augmente avec la taille des vecteurs.*
- *L'algorithme est très dépendant de la valeur k . Des différences notables peuvent apparaître en fonction de ce paramètre.*
- *Avec des k trop petits le modèle peut sur-ajuster les données alors que des valeurs trop grandes ont tendance à "lisser" les prédictions.*

Arbres de décision

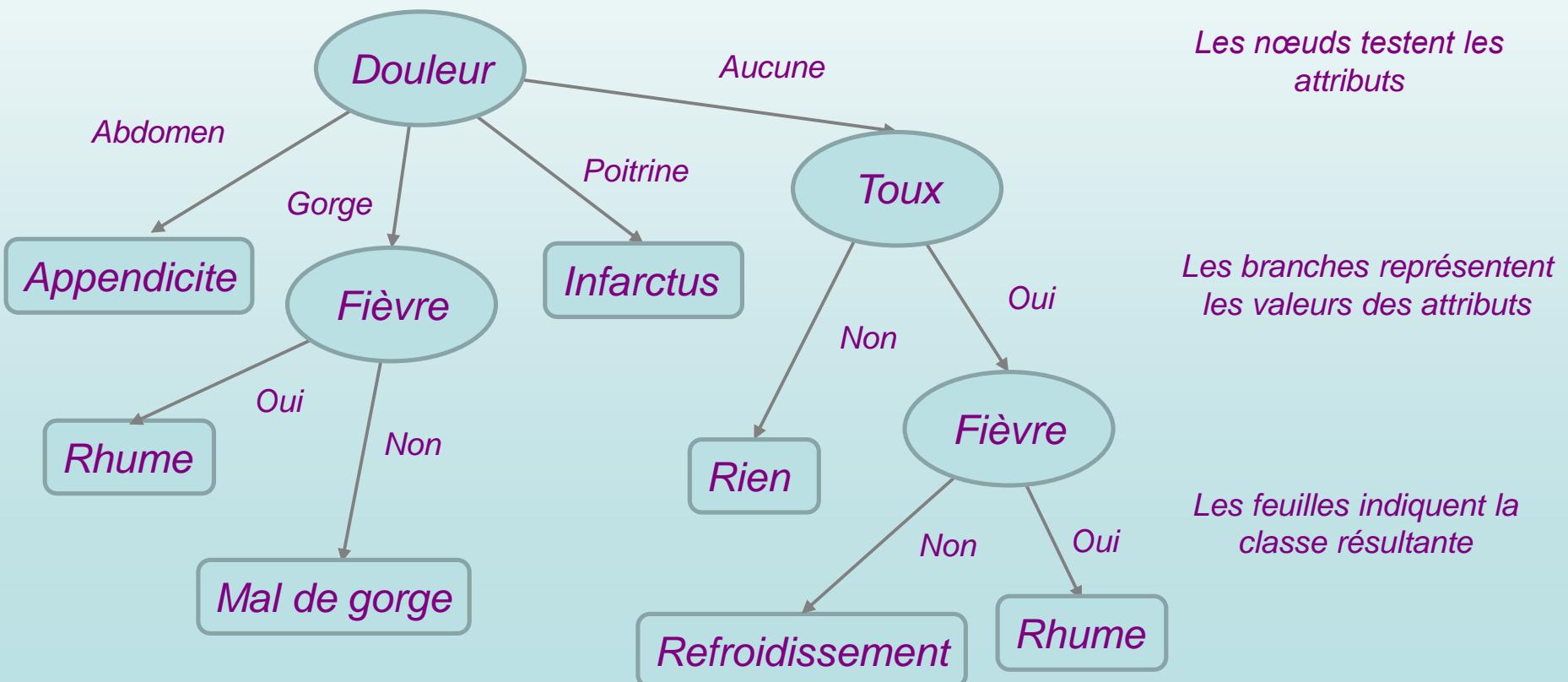
Présentation

- *Les arbres de décision (AD) sont une catégorie d'arbres utilisée dans l'exploration de données et en informatique décisionnelle.*
- *Ils permettent d'obtenir un résultat en se basant sur une série de tests en vue de la prédiction d'un résultat ou d'une classe.*
- *La construction d'un AD se réalise en deux étapes.*
- *Etape 1 : la construction est un processus itératif de division de l'espace des données en régions de plus en plus pures en terme de classes.*
- *Chaque nœud intermédiaire réalise un test portant sur une variable dont le résultat indique la branche à suivre dans l'arbre.*
- *Pour classer un nouveau cas il faut suivre le chemin partant de la racine à une feuille de l'arbre en effectuant les différents tests à chaque nœud.*
- *Etape 2: l'élagage ("pruning") consiste à supprimer les feuilles peu représentatives pour garder de bonnes performances prédictives.*
- *Il existe un grand nombre de logiciels permettant de modéliser un processus sous la forme d'un arbre de décision : CART, CHAID, C4.5 ...*

Arbres de décision

Représentation

- Ces logiciels diffèrent les uns des autres par les critères qui vont permettre de sélectionner l'ordre dans lesquels seront réalisés les tests.



Arbres de décision

Arbre de décision : construction

- La construction qui correspond à la phase de prédiction est totalement automatique, elle n'est en rien dirigée par les utilisateurs.
- Les algorithmes de construction consistent à trouver les variables les plus discriminantes, c'est-à-dire celles qui séparent le mieux les données.
- Il existe plusieurs mesures qui permettent de sélectionner ces variables.

Mesure : Entropie de la target

- L'algorithme ID3 (Itérative Dichotomiser 3) permet de sélectionner la variable qui a le gain d'information basé sur l'entropie le plus élevé.
- L'entropie mesure le degré de désorganisation des données. Plus la valeur de l'entropie est grande et plus les données sont désorganisées.
- L'entropie de Shannon pour une **variable cible** (target) est donnée par :

$$H(S) = \sum_{c \in S} -P(c) * \log_2(P(c))$$

Par convention

$$\log_2(0) = 0$$

S est le jeu de données (DataSet)
 c les classes de S
 P(c) proportion d'individus de classe c
 par rapport au nombre d'individus de S

Arbres de décision

Mesure : Entropie d'une variable non cible

- Pour un variable V à estimer, pouvant prendre les valeurs v_1 à v_n , son entropie dépend de l'entropie de V sur chacune de ses valeurs.

$$H(V, S) = \sum_{i=1}^n \frac{car(v_i)}{car(S)} * \sum_{c \in S} -\frac{car(c)}{car(v_i)} * \log_2 \left(\frac{car(c)}{car(v_i)} \right)$$

- $P(car(c)|car(v_i))$ est la proportion d'individus de V avec la valeur v_i et qui appartiennent à la classe c de S .
- Le gain d'information, mesure la différence entre l'entropie de base et celle séparant les individus sur la variable a estimer.

$$G(V, S) = H(S) - H(V, S)$$

- Le premier test se fera sur la variable qui a le gain le plus élevé.
- Le processus recommence ensuite sur chacune de branche, avec un DataSet réduit aux données appartenant à la classe sélectionnée.
- Le processus s'arrête quand les éléments d'un nœud ont la même valeur pour la variable cible (homogénéité).

Arbres de décision

Exemple : Entropie

- L'entropie de la variable cible (Achat) et obtenues à partir des deux probabilités d'acheter ou pas un ordinateur.*
- Pour les autres l'entropie est la somme des entropies de chaque cas.*

Age	Statut	Revenu	Solvabilité	Achat PC
<= 30	Chomeur	Elevé	Bon	Non
<= 30	Salarié	Elevé	Bon	Non
[31..40]	Chomeur	Moyen	Mauvais	Non
> 40	Etudiant	Bas	Bon	Oui
> 40	Salarié	Moyen	Mauvais	Oui
> 40	Etudiant	Bas	Bon	Oui
[31..40]	Etudiant	Bas	Bon	Oui
<= 30	Salarié	Moyen	Mauvais	Non
<= 30	Etudiant	Bas	Mauvais	Oui
> 40	Salarié	Elevé	Bon	Oui
<= 30	Etudiant	Moyen	Bon	Oui
[31..40]	Salarié	Moyen	Bon	Oui
[31..40]	Etudiant	Elevé	Mauvais	Oui
> 40	Chomeur	Moyen	Bon	Non

$$E(Achat) = -\frac{9}{14} \log_2 \left(\frac{9}{14} \right) - \frac{5}{14} \log_2 \left(\frac{5}{14} \right) = 0,940$$

$$E(Age \leq 30) = \frac{5}{14} \left(-\frac{2}{5} \log_2 \left(\frac{2}{5} \right) - \frac{3}{5} \log_2 \left(\frac{3}{5} \right) \right) = 0,347$$

$$E(30 < Age \leq 40) = \frac{4}{14} \left(-\frac{3}{4} \log_2 \left(\frac{3}{4} \right) - \frac{1}{4} \log_2 \left(\frac{1}{4} \right) \right) = 0.232,$$

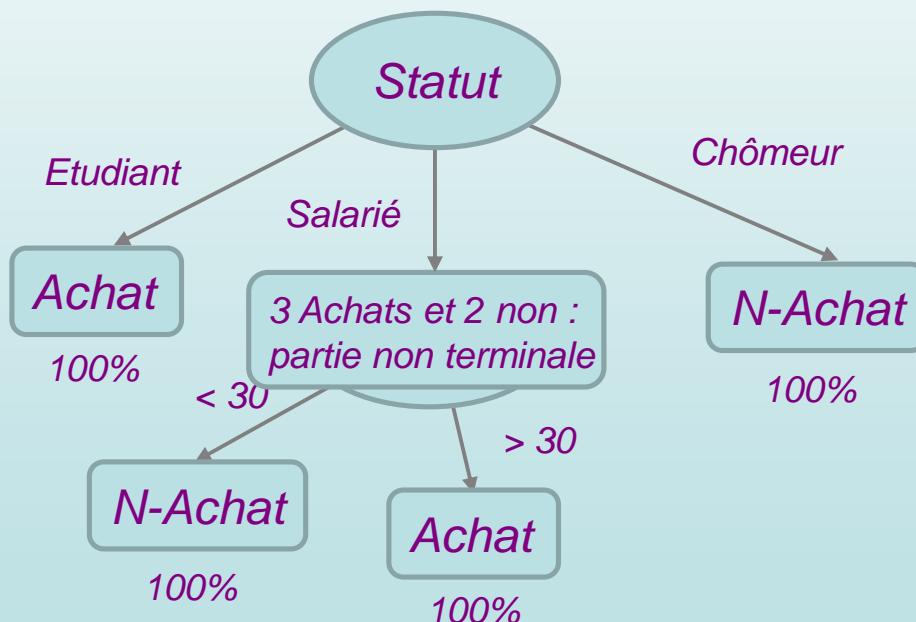
$$E(Age > 40) = \frac{5}{14} \left(-\frac{1}{5} \log_2 \left(\frac{1}{5} \right) - \frac{4}{5} \log_2 \left(\frac{4}{5} \right) \right) = 0,258$$

$$E(Age) = 0,347 + 0.232 + 0,258 = 0.836$$

Arbres de décision

Mesure : Entropie

- On obtient les gains suivants : $G(\text{Age}) = 0.104$, $G(\text{Statut}) = 0.594$, $G(\text{Revenu}) = 0.226$ et $G(\text{Solvabilité}) = 0.003$.
- La variable *Statut* qui a le gain le plus élevé, est choisie.



La classe Salarié est de 5 individus
 $E(\text{Achat}) = 0.97$
 $G(\text{Age}) = 0.97$ et $G(\text{Statut})=G(\text{Sol})=0.02$

Arbres de décision

Prix des maisons à Boston

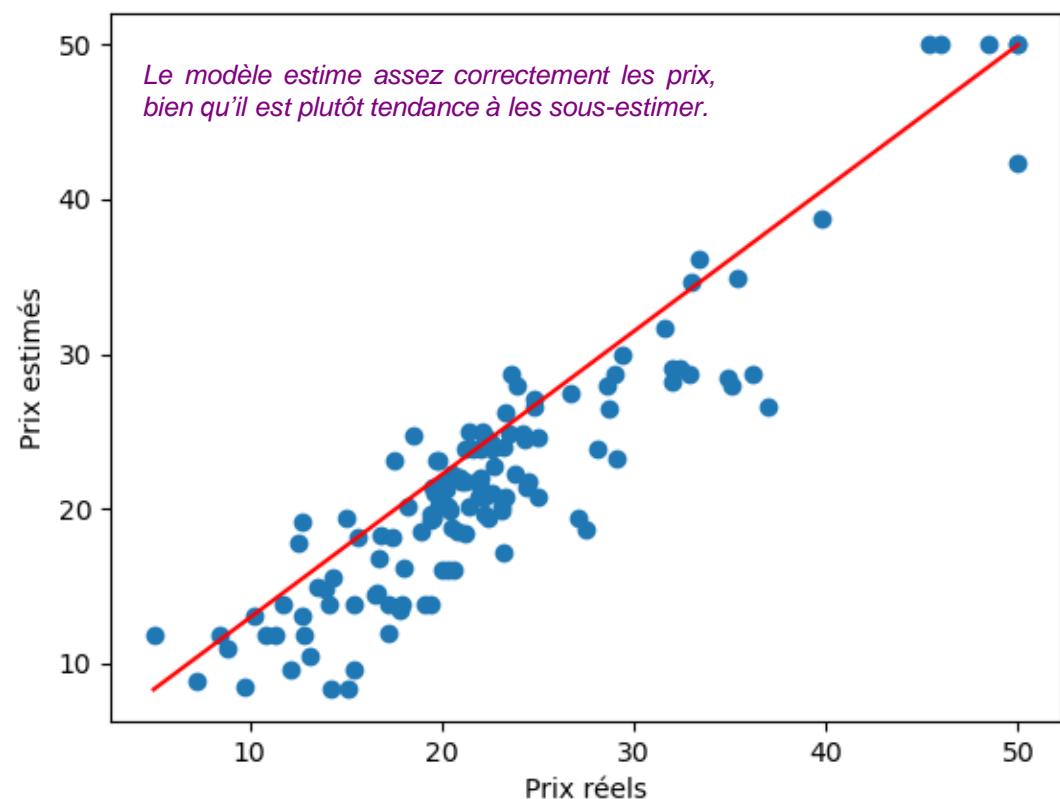
- Le dataset Boston contient 394 samples, 13 features et une target correspondant au

Lecture du DataSet
avec pandas

Création des données de
test et de train

Estimation

Affichage des résultats



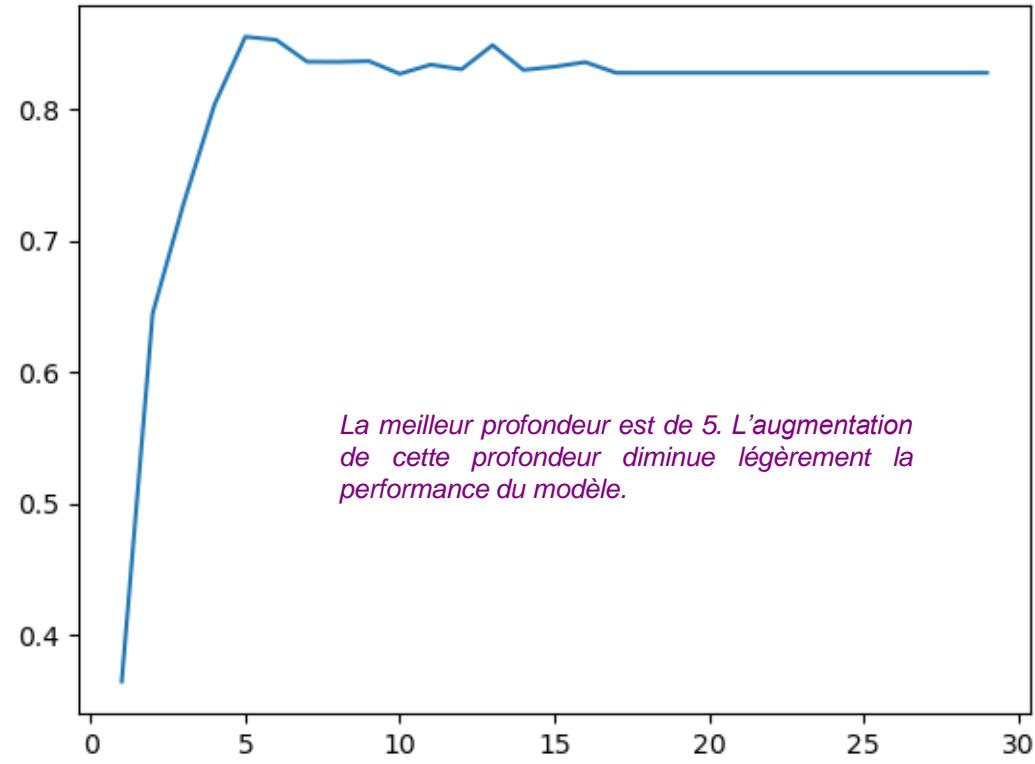
Arbres de décision

Profondeur des arbres de décision

- Pour des problèmes non-linéaires les arbres peuvent être de grande taille avec beaucoup de feuilles.
- Les profondeurs peuvent être élevées ('bruit') ce qui contribue au surapprentissage.
- Pour éviter ce phénomène il faut trouver un compromis correct de généralisation.
- L'hyper paramètre α permet de régler la profondeur.

Evaluation des résultats avec différentes profondeurs

- Si le bruit est important la profondeur n'a pas d'influence.



Arbres de décision

Utilisation des arbres de décision

- Les AD sont des algorithmes faciles à comprendre et à interpréter. Ils sont aussi à la base d'approches ensemblistes très puissantes.
- Généralement on utilisera un AD pour évaluer un modèle et mieux le comprendre avant de passer à d'autres approches plus performantes
- un AD évalue l'importance des features, ce qui simplifie les traitements ultérieurs à base d'algorithmes d'apprentissage "boites noires".

```
importance = pd.DataFrame(data=model.feature_importances_,  
                           index=X_test.columns, columns=['%'])  
importance.sort_values(by='%', ascending=False)
```

RM : Nb de pièces ; LSTAT : % pauvres ; CRIM : % criminalité
 ZN : % zones résidentielles ; CHAS : rivière ; B : % de noirs
 On constate que 8 variables ont un impact < 1% sur le prix

	%		%
RM	0.642410	INDUS	0.006011
LSTAT	0.226771	NOX	0.004333
CRIM	0.068226	AGE	0.00183
TAX	0.025530	ZN	0.000000
RAD	0.011249	CHAS	0.000000
DIS	0.007301	B	0.000000
PTRATIO	0.006336		

- Les arbres de décision aboutissent souvent à des nœuds finaux trop précis, ce qui conduit très souvent à du sur-apprentissage.
- Les performances peuvent se dégrader avec la taille du dataset.

Méthodes ensemblistes

Méthodes ensemblistes

- Les méthodes ensemblistes sont des outils d'apprentissage très puissants, ils reposent sur le principe qu'une foule d'amateurs à plus souvent raison qu'un expert seul.
- Cette idée est basée sur le concept de la sagesse des foules (*wisdom of crowd*).
- La théorie ensembliste montre que la combinaison de "modèles faibles" permet d'améliorer la performance globale, bien mieux que la combinaison de modèles performants et robuste au bruit.
- On utilise le plus souvent des arbres de décision pour entraîner les modèles faibles à partir d'un jeu de données.
- Le bagging est une méthode parallèle dans laquelle les modèles faibles sont tous indépendants les uns des autres,
- Le boosting est une méthode dans laquelle les modèles fonctionnent en séquentiel. Chaque modèle apprend à construire son raisonnement à partir des performances du précédent.

Méthodes ensemblistes

Le bagging : Random Forest

- La puissance des méthodes ensemblistes est obtenue lorsque les modèles faibles sont aussi différents les uns des autres que possible, afin que les erreurs des uns soient compensés par les forces des autres.
- Un random forest est constitué d'un ensemble d'arbres de décision indépendants, chaque arbre dispose d'une vision parcellaire du problème du fait d'un double tirage aléatoire :
 - le tree bagging : correspond à un tirage aléatoire d'individus (de taille racine(N)) sur l'ensemble des N individus d'origine.
 - le feature sampling : un tirage aléatoire sur les features.
- Une prédiction pour un individu inconnu est effectuée par tous les arbres, le résultat final est alors la prédiction majoritaire pour un problème de classification, ou la moyenne des prédictions pour une variable continue.
- Si les modèles ont au moins 50% de performance et un minimum de diversité, les résultats finaux seront meilleurs que pour un modèle "fort".
- Outre la qualité des estimations, le découpage des DataSets permet de réduire considérablement les temps de calculs.

Méthodes ensemblistes

Le boosting : AdaBoost et Gradient Boosting

- *Dans le bagging les modèle apprennent sur des régions différentes de l'espace des données, les erreurs commises sont alors décolérées.*
- *Le boosting consiste à entraîner un modèle puis à utiliser un second modèle qui se concentre sur les erreurs commises par le précédent.*
- *Adaboost (Adaptative Boosting) permet de construire un classifieur de manière itérative, en forçant un modèle faible à se concentré sur les erreurs du modèle précédent en pondérant les exemples d'entraînement.*
- *Adaboost est un algorithme particulier de la famille des algorithmes de Gradiant Boosting pour lesquels le différences se font sur les fonctions de coût (erreur exponentielle dans Adaboost, quadratique, entropie...)*
- *Enfin il existe une troisième technique d'ensemble le staking/*
- *Dans ce cas un modèle est entraîné sur les prédictions d'une foule. Le modèle va estimer qui à tort ou qui a raison parmi les réponses et retourne sa propre réponse.*

Apprentissage non supervisé

Apprentissage non supervisé

- Analyse en composante principale
- Classification ascendante hiérarchique
- K-means
- DBSCAN
- Isolation Forest

Apprentissage non supervisé

Présentation

- *Les algorithmes vont apprendre à classer les individus selon leur ressemblance au sein de groupes différentes.*
- *La similarité est basée sur des distances (Euclidienne), l'entropie (dispersion), la réduction d'échelle (centrer-réduire) ou les probabilités.*
- *On crée des groupes d'individus dans lesquels les individus se ressemblent et sont différents d'un groupe à l'autre.*
- *La plupart du temps l'apprentissage non supervisé est associé au Clustering (segmentation, regroupement) qui cherche à décomposer les d'individus en plusieurs sous ensembles les plus homogènes possibles.*
- *Dans certains cas on cherche à analyser les relations entre les variables ou détecter des associations.*
- *Enfin, la réduction de dimensionnalité consiste à réduire l'espace de représentation des données afin d'accélérer les traitements et/ou d'éliminer les informations inutiles (bruits).*

Analyse en composante principale

La réduction de dimensionnalité

- *Le principe consiste à chercher les liens entre les variables. Les liaisons linéaires les plus fréquentes, se mesurent via un coefficient de corrélation.*
- *Si les coefficients sont élevés (proche de 1) les variables apportent la même information : il est alors possible de les réduire.*
- *Par exemple : Le poids et la taille des individus sont corrélés.*
- *L'Analyse en Composante Principale est une méthode de statistique exploratoire qui synthétise et hiérarchise l'information contenue dans un tableau de données.*
- *Si le nombre de variables est important les modèles apprennent bien mais sont souvent incapables de généraliser.*
- *La réduction du nombre de variables conduit à une plus grande robustesse ou une meilleure stabilité de l'algorithme.*
- *Les variables inutiles écartées, les modèles sont alors plus simples à traiter, l'apprentissage est plus rapide et les erreurs dues à des variables peu représentatives sont supprimées.*

Analyse en composante principale

Normalisation des données

- En analysant les corrélations entre les variables, il est alors possible de réduire la dimensionnalité.
 - Mais si les variables sont mesurées dans différentes unités (kg, km, cm, ...) les résultats de l'ACP sont fortement affectés.
 - Afin de s'affranchir des unités il est souvent nécessaire de centrer (moyenne nulle) et réduire (écart type égal à 1) les données.

- scikit-learn propose plusieurs méthodes permettant de centrer, réduire, normaliser : StandardScaler, MinMaxScalar

```
from sklearn.preprocessing import StandardScaler  
# Transformation des données d'un DataSet : moyenne nulle et un écart type égal à 1  
sc = StandardScaler()  
X = sc.fit_transform(DataSet) # Matrice centré réduite
```

Analyse en composante principale

Matrice des corrélations : ACP

- L'ACP cherche à transformer des variables liées entre elles ou corrélées en de nouvelles variables dé-correlées les unes des autres.
- Elle résume l'information en réduisant le nombre de variables.
- Le principe consiste à décomposer la matrice des corrélations entre les variables en un produit de 3 matrices.
- 1 – Construire la matrice des corrélations.

$$Cor(X) \rightarrow \frac{1}{N} [X]^t [X] \quad \text{soit}$$

$$Cor_x = X.T.dot(X)/len(X)$$

- 2 – Décomposer la matrice des corrélations dans une base diagonale.

$$Cor(X) \rightarrow [vec][val][vec]^t \quad \text{soit}$$

$$Vp = np.linalg.eig(Cor_X)$$

- val est une matrice diagonale (valeurs propres λ_i) correspondant à l'importance des dimensions les unes par rapport aux autres.
- vec contient les coordonnées des individus dans cette nouvelle base.

Analyse en composante principale

Les composantes principales : ACP

- Les rapport $\lambda_i / \sum(\lambda_i)$ permettent de connaitre le pourcentage d'information contenu dans chaque dimension.
- scikit-learn propose une bibliothèque PCA permet l'analyse en composantes principales.
- Dans un modèle PCA il est possible d'indiquer le nombre de composants à conserver (`n_components`). Tous sont conservés par défaut.

```
from sklearn.decomposition import PCA          # Bibliothèque spécifique pour l'ACP
acp = PCA(svd_solver='full')
Coord = acp.fit_transform(X)
acp.explained_variance_                         # Matrice des coordonnées dans le nouvel espace
                                                # affichage des valeurs propres
```

- La fonction `fit_transform` retourne les coordonnées des individus dans la nouvelle base (matrice `vec`), l'attribut `explained_variance_` le différentes valeurs propres (λ_i).
- La fonction `inverse_transform` permet de retourner aux valeurs initiales à partir du jeu de données .

Analyse en composante principale

Exemple scikit-learn : <https://husson.github.io/data.html>

```
capitales = pd.read_csv('Capitales.csv', delimiter=";", index_col= "villes")      # lecture des données
capitales = capitales.iloc[:,range(0,12)]                                         # Récupération des données à traiter (Jan-Déc)
```

Capitales

Villes	Janvier	Février	Mars	Avril	Mai	Juin	Juillet	Août	Septembre	Octobre	Novembre	Décembre	Moyenne	Amplitude	Latitude	Longitude	Région	
Amsterdam	2.9	2.5	5.7	8.2	12.5	14.8	17.1	17.1	14.5	11.4	7	4.4	9.9	14.6	52.2	4.5	Ouest	
Athènes	9.1	9.7	11.7	15.4	20.1	24.5	27.4	27.2	23.8	19.2	14.6	11	17.8	18.3	37.6	23.5	Sud	
Berlin	-0.2	0.1	4.4	8.2	13.8	16	18.3	18	14.4		10	4.2	1.2	9.1	18.5	52.3	13.2	Ouest
Bruxelles	3.3	3.3	6.7	8.9	12.8	15.6	17.8	17.8		15	11.1	6.7	4.4	10.3	14.4	50.5	4.2	Ouest
Budapest	-1.1	0.8	5.5	11.6	17	20.2	22	21.3	16.9		11.3	5.1	0.7	10.9	23.1	47.3	19	Est

```
from sklearn.decomposition import PCA
acp = PCA(svd_solver='full')                                              # Objet l'ACP
# fit.transform = Apprentissage + Transformation. Présentation des données en DataFrame
capitales_acp = pd.DataFrame(acp.fit_transform(capitales), index= capitales.index) # instanciation d'un modèle d'ACP
```

acp.explained_variance_ratio_

Retourne le ratio des 12 valeurs propres

Ratio = [0.869 ; 0.114 ; 0.010 ; 0.004 ...]

La projection des données sur le premier axe conserve 87% de l'information contenue dans les données, la seconde 11%. Une représentation dans la base liée aux deux premières valeurs propres fournit 98% de l'information. La représentation des individus peut se limiter à ces deux axes.

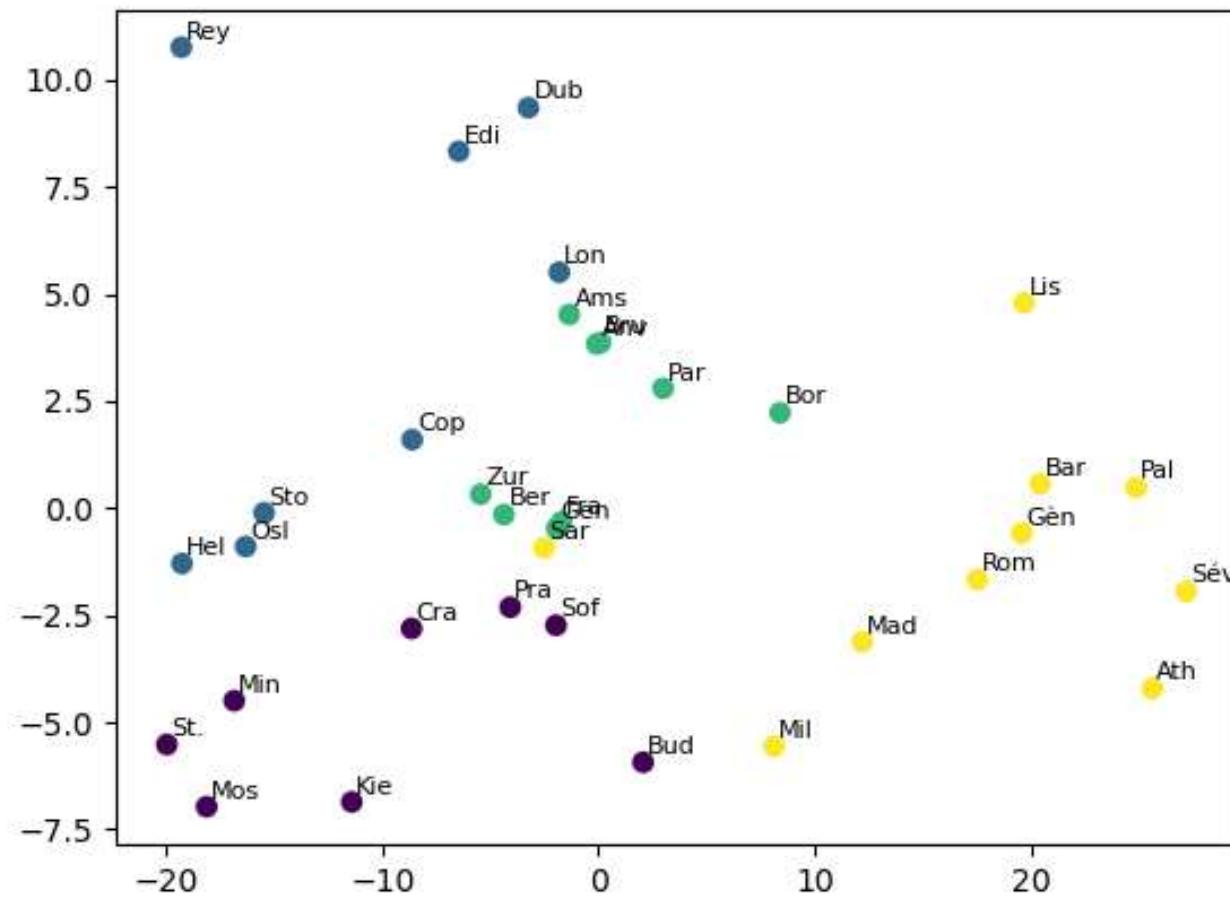
Analyse en composante principale

Les changements

- En comparaison avec la première, cette carte est en effet plus étendue.

Un LabelE
numérique

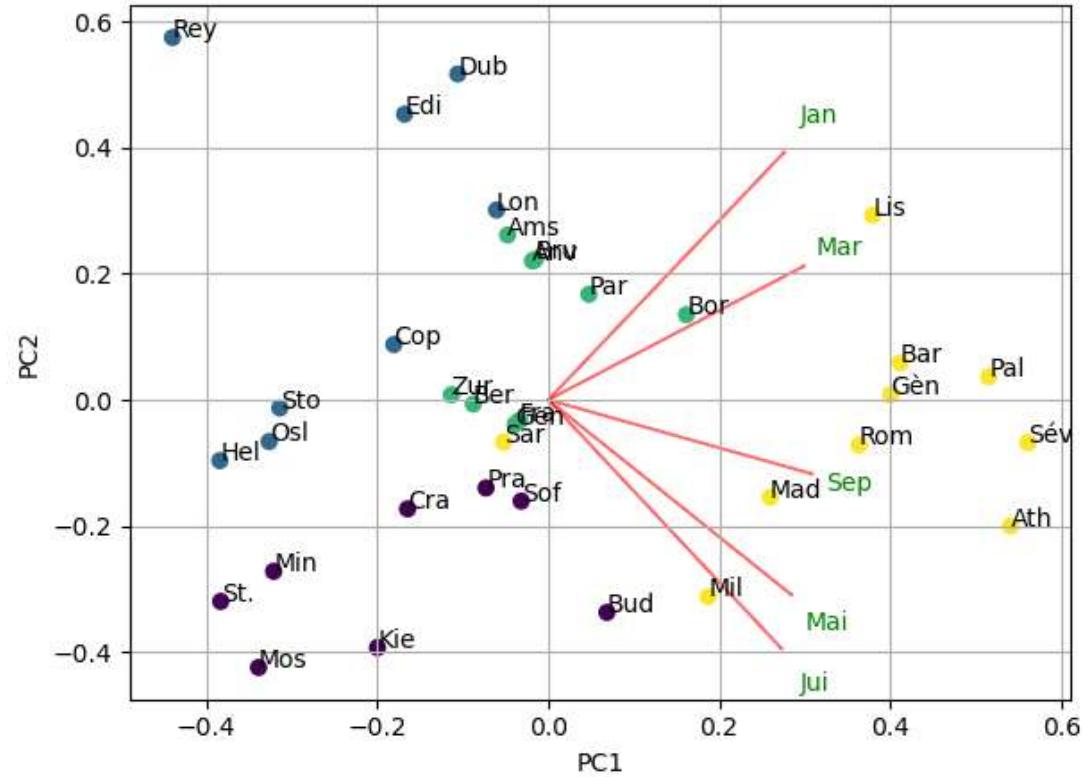
Affichage



Analyse des vari

```
def myplot(x, y, corX, corY, la
    scalex = 1.0/(x.max() - x.m
    scaley = 1.0/(y.max() - y.m
    plt.scatter(x * scalex, y * sc
    for i in range(len(x)):
        plt.text(x[i]*scalex, y[i]*sc
    for i in labels:
        plt.arrow(0, 0, corX[i], co
        plt.text(corX[i] * 1.15, co
    plt.xlabel("PC{}".format(1))
    plt.grid() ; plt.show()
```

```
labels=[0, 2, 4, 6, 8]
myplot(capitales_acp[0], capi
```



Axe 1 : Tous les mois (Jan-Dec) sont corrélés positivement sur l'axe des x. Les capitales qui ont des valeurs positives sur l'axe x ont des températures élevées toute l'année.

Valeurs moyennes des températures sur l'année.

Axe 2 : Les mois (Avr-Oct) sont corrélés négativement à l'axe 2. Les villes avec des valeurs positives ont peu de différences entre les hivers et les étés. C'est l'inverse pour les autres villes.

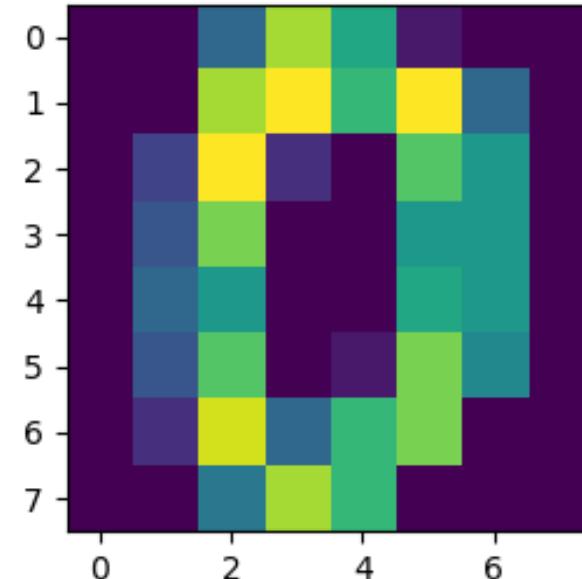
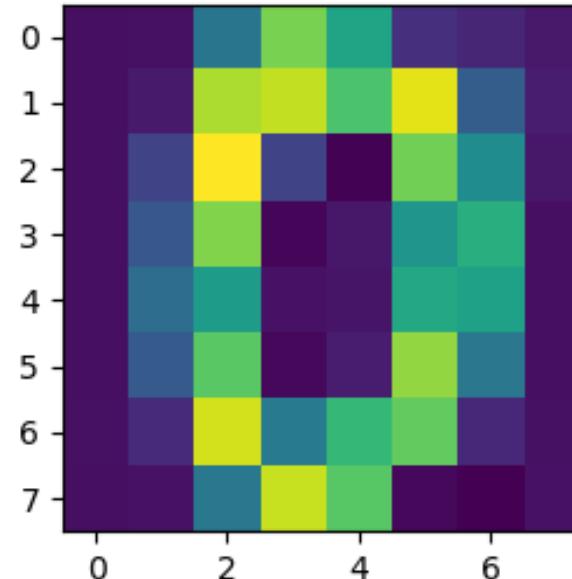
Variabilité des températures en les hivers et les étés.

Analyse des données

- *digit*
- *Pourquoi*
- *nécessaire*

```
from sklearn.datasets import load_digits
digits = load_digits()
acp = PCA()
digits_acp = acp.fit_transform(digits.data)
# Calcul cumulé (expliqué)
explained = np.cumsum(acp.explained_variance_ratio_)
print(np.argmax(explained))
plt.plot(explained)
```

```
acp = PCA(n_components=2)
digits_acp = acp.fit_transform(digits.data)
digits_inverse = acp.inverse_transform(digits_acp)
fig, ax = plt.subplots(1, 2)
ax[0].imshow(digits[0])
ax[1].imshow(digits_inverse[0])
```



Classification Ascendante Hiérarchique

Principes de la CAH

- La CAH est une méthode de classification qui consiste à regrouper les individus qui se ressemblent au sein d'un même ou groupe.
- On cherche au sein d'un groupe les individus soient semblables et que les groupes soient le plus dissemblables possible.
- Le critère de ressemblance s'exprime sous la forme la distance qui sépare chaque individu pris deux à deux.
- Un groupe est identifié par son barycentre (CG) qui correspond à la somme moyenne des paramètres de ses n individus qui le compose.

$$CG = \frac{1}{n} \left(\sum_{i=1}^n x_i^1, \sum_{i=1}^n x_i^2, \dots, \sum_{i=1}^n x_i^m \right) = (x_g^1, x_g^2, \dots, x_g^m)$$

- La distance entre deux groupes CG_1 et CG_2 correspond à la distance entre leur centre de gravité.

$$\begin{aligned} CG_1 &= (x_{g1}^1, x_{g1}^2, \dots, x_{g1}^m) \\ CG_2 &= (x_{g2}^1, x_{g2}^2, \dots, x_{g2}^m) \end{aligned}$$

$$d^2 = \sum_i^m (x_{g1}^i - x_{g2}^i)^2$$

Classification Ascendante Hiérarchique

Principes de la CAH

- CAH estime deux inerties (*intra-groupe* I_{intra} et *inter-groupe* I_{inter}).
- L'inertie d'un groupe I_g mesure la dispersion des individus autour de son barycentre CG_g . Elle correspond à la distance des individus à CG_g .

$$I_g = \frac{1}{n_g} \sum_i^{n_g} d^2(CG_g, ind_i)$$

ind_i ième individu du groupe
 n_g nombre d'individus du groupe

- L'inertie intra-groupe I_{intra} est la somme pondérée des inerties.

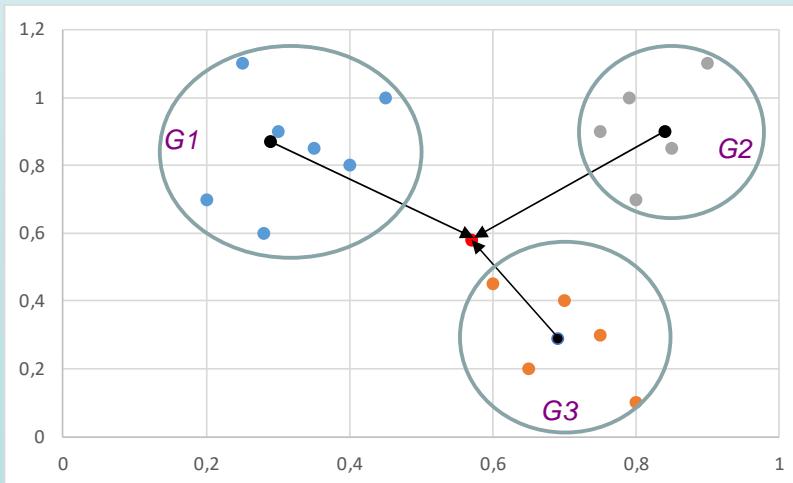
$$I_{intra} = \sum_i^{nb_g} \frac{n_i}{n} I_i$$

nb_g nombre de groupes
 n nombre d'individus
 i numéro de la groupe

- L'inertie inter-groupe :

$$I_{inter} = \sum_i^{nb_g} \frac{n_i}{n} d^2(CG, CG_{gi})$$

CG est le barycentre des données



Classification Ascendante Hiérarchique

Principes de la CAH

- CAH est une méthode de classification itérative.
- L'algorithme consiste à partir de la situation initiale où chaque individu constitue un groupe à lui tout seul, soit un total de n groupes.
- Les conditions initiales en termes d'inertie sont : $I_{intra}=0$ et $I_{inter}=I_{total}$
- L'algorithme cherche les 2 groupes les plus proches au sens de la distance euclidienne pour former un nouveau groupe.
- Etape 1 : La création de ce groupe augmente l'inertie intra-groupe. On remplace ensuite les deux groupes par le couple placé au barycentre des 2 groupes, le nombre de groupes est alors de $n-1$.
- Etape 2 : L'algorithme recalcule les distances entre les groupes et l'on regroupe les 2 plus proches, ce qui provoque une nouvelle augmentation de l'inertie intra-groupe couplée d'une diminution du nombre de groupes.
- L'algorithme s'arrête lorsque le nombre de groupes est égal à 1.

Classification : Exemple

Etape 1 : 8 Groupes, $I_{intra} = 0 - I_{inter} = 24.25$

La distance $[A,C] = 2$ est minimum, on crée alors le groupe AC. On donc 7 groupes et $I_{intra} = 0.25$

Etape 2 : 7 Groupes, $I_{intra} = 0.25 - I_{inter} = 24$

La distance $[AC,B] = 2$ est minimum, on crée le groupe ACB. On a alors 6 groupes et $I_{intra} = 0.25 + 0.33$

Etape 3 : 6 Groupes, $I_{intra} = 0.58 - I_{inter} = 23.67$

La distance $[E,G] = 2.83$ est minimum, on crée le groupe EG. On a maintenant 5 groupes et $I_{intra} = 0.53 + 0.5$

Etape 4 : 5 Groupes, $I_{intra} = 1.08 - I_{inter} = 23.17$

La distance $[D,F] = 3.16$ est minimum, on crée le groupe DF. On a maintenant 4 groupes et $I_{intra} = 1.08 + 0.63$

Etape 5 : 4 Groupes, $I_{intra} = 1.71 - I_{inter} = 22.54$

La distance $[EG,H] = 3.61$ est minimum, on crée le groupe EGH. On a maintenant 3 groupes et $I_{intra} = 1.71 + 1.08$

Etape 6 : 3 Groupes, $I_{intra} = 2.79 - I_{inter} = 21.46$

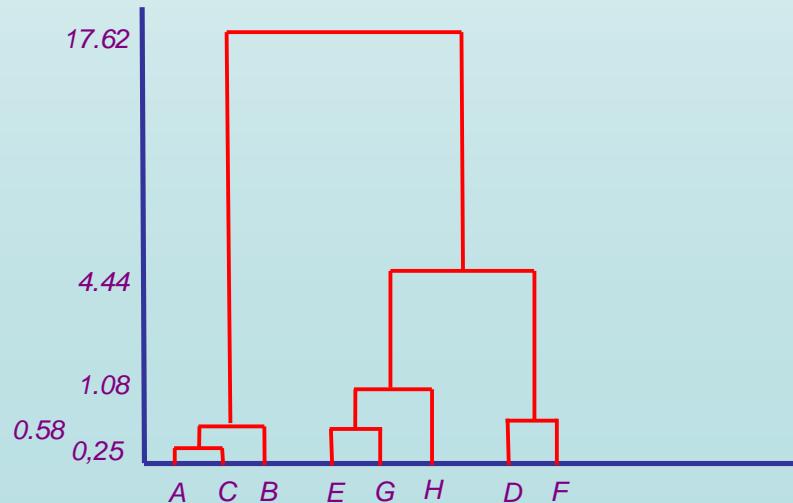
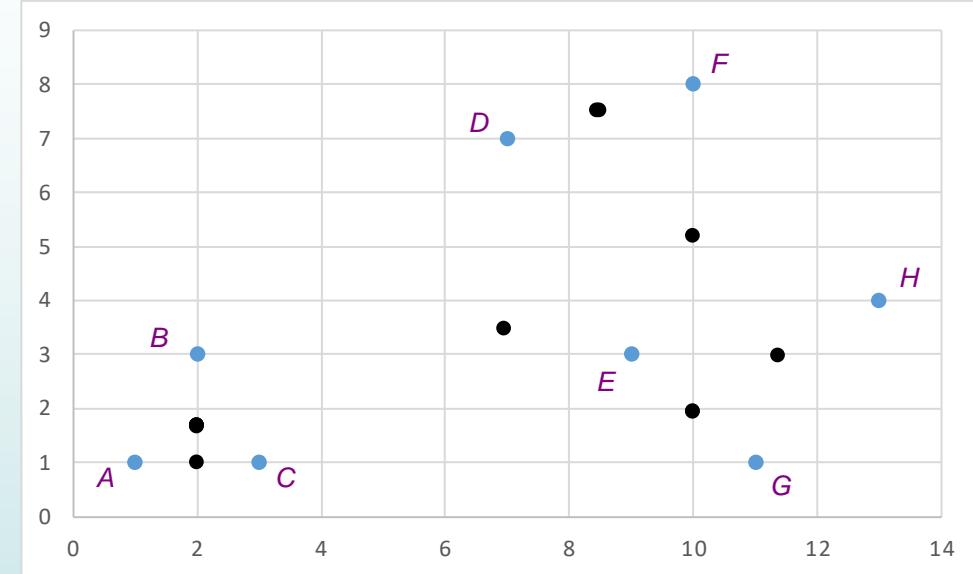
La distance $[EGH,DF] = 5.4$ est minimum, on crée le groupe EGHDF. Il reste 2 groupes et $I_{intra} = 2.79 + 4.44$

Etape 7 : 2 Groupes, $I_{intra} = 7.23 - I_{inter} = 17.02$

Les groupes ACB et EGHDF sont regroupés, il ne reste alors plus qu'un seul groupe et $I_{intra} = 7.23 + 17.02 = 24.25$

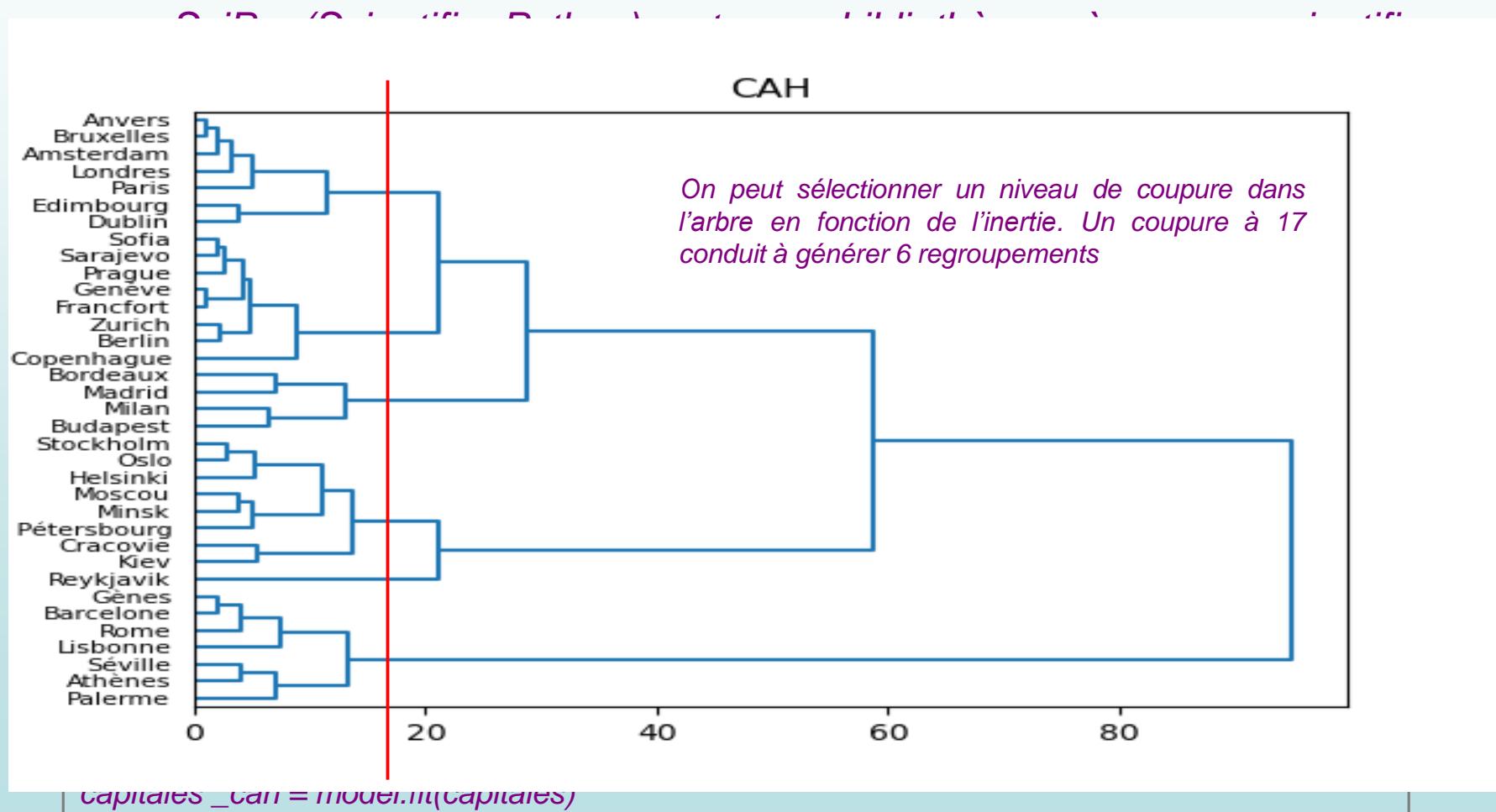
Final 1 groupe , $I_{intra} = 24.25$

Classification Ascendante Hiérarchique



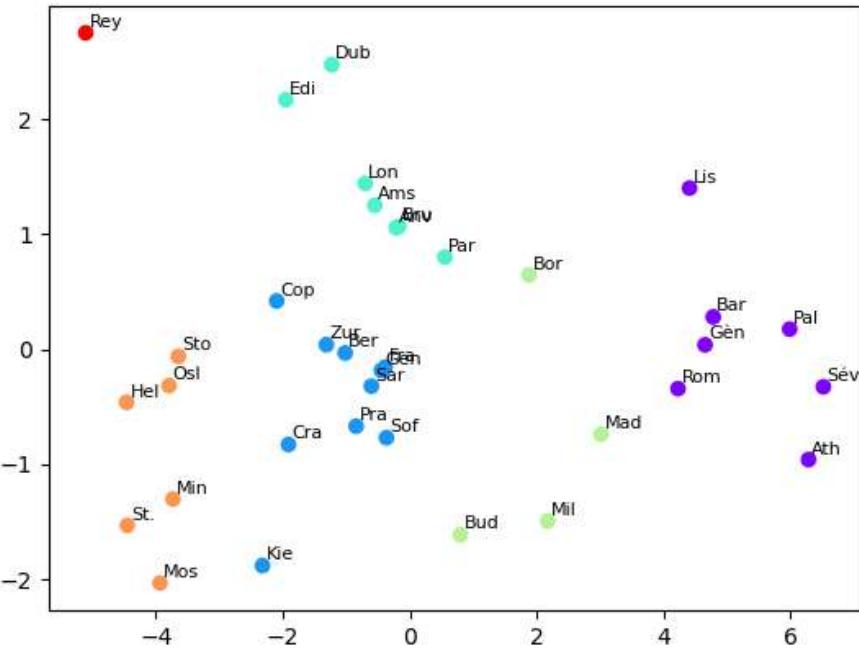
Classification Ascendante Hiérarchique

Exemple sur les capitales



Classification Ascendante Hiérarchique

Affichage des clusters



```
# Affichage des groupes
capitales['cluster']=cluster.labels_
groups = capitales.groupby('cluster')
for name, group in groups:
    print(f'cluster {name} : {list(group.index)}')
    bels_, cmap='rainbow'
```

```
cluster 0 : ['Helsinki', 'Kiev', 'Cracovie', 'Minsk',  
            'Moscou', 'Oslo', 'Stockholm', 'St. Pétersbourg']  
cluster 1 : ['Athènes', 'Lisbonne', 'Rome',  
            'Barcelone', 'Gènes', 'Palerme', 'Séville']  
cluster 2 : ['Amsterdam', 'Bruxelles', 'Dublin',  
            'Londres', 'Paris', 'Anvers', 'Edimbourg']  
cluster 3 : ['Budapest', 'Madrid', 'Bordeaux', 'Milan']  
cluster 4 : ['Berlin', 'Copenhague', 'Prague', 'Sofia',  
            'Sarajevo', 'Francfort', 'Genève', 'Zurich']  
cluster 5 : ['Reykjavik']
```

Inconvénient : Le principal défaut de CAH est le temps de calcul qui devient rédhibitoire si le nombre d'individus est élevé. On procède alors à la **CAH Mixte**, qui consiste à la faire précéder par une phase de pré-regroupement, en utilisant l'algorithme K-means par exemple,

K-means

K-moyennes ou K-means

- *K-moyennes est un algorithme qui regroupe les individus en K clusters.*
- *Principe de la méthode :*
 - *On choisit au départ K centroïdes, par exemple K individus.*
 - *Chaque individu est relié au centroïde le plus proche.*
 - *On calcule ensuite le barycentre des différents clusters et on déplace les centroïdes sur ces positions.*
 - *On itère tant que les centroïdes changent.*
- *Le nombre de cluster se fait à partir de la variance des clusters.*
- *Sur K-means utilisation de l'algorithme avec différentes valeurs de K, on retient le nombre de clusters à partir duquel la variance ne se réduit plus significativement.*
- *La variance est la somme des distances centroides/individus.*
- *L'attribut inertia_ permet d'accéder à la variance des modèles.*

K-means

Kmeans : Exemple

1 - Choix des 3 centroïdes (C_1, C_2, C_3) = (E, F, I)

Les individus (E, G) sont affectés au groupe de C_1

Les individus (A, B, C, D, F, H, J) au groupe C_2

Les individus (I, K) au groupe C_3

2 - Les trois centroïdes sont maintenant

$(C_1, C_2, C_3) = ((3.3, 5.2), (2.6, 1.9), (5, 4.2))$

Les individus (E, G) sont affectés au groupe de C_1

Les individus (A, B, C, D, F) au groupe C_2

Les individus (H, I, J, K) au groupe C_3

3 - Les trois centroïdes sont maintenant

$(C_1, C_2, C_3) = ((3.3, 5.2), (1.76, 1.7), (4.8, 3.3))$

Les individus (E, G) sont affectés au groupe de C_1

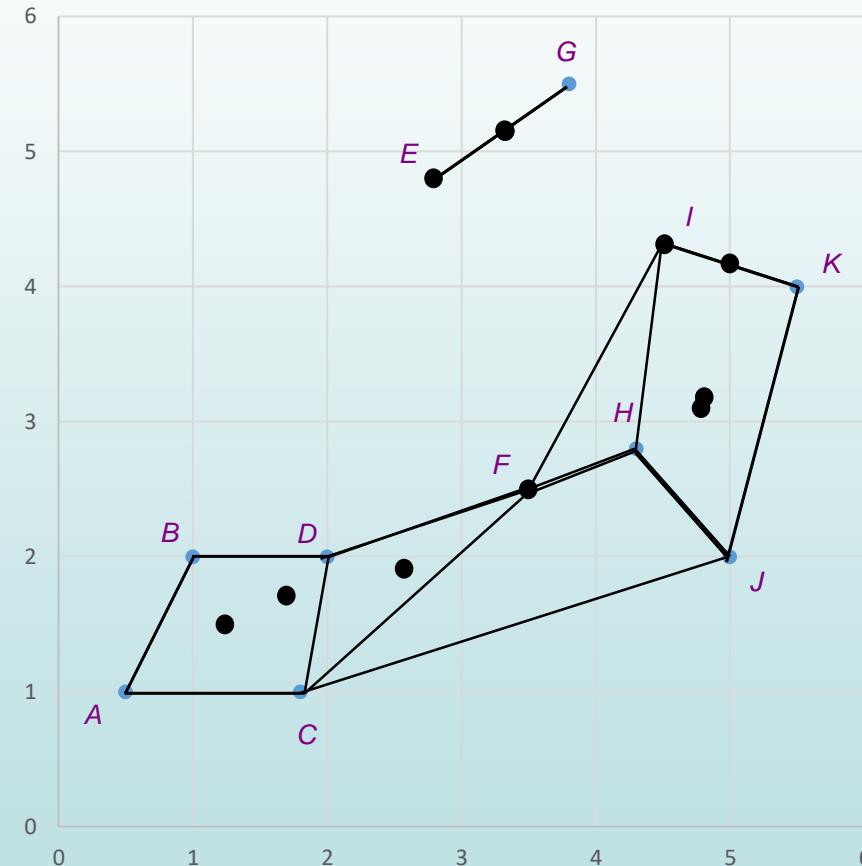
Les individus (A, B, C, D) au groupe C_2

Les individus (F, H, I, J, K) au groupe C_3

4 - Les trois centroïdes sont maintenant

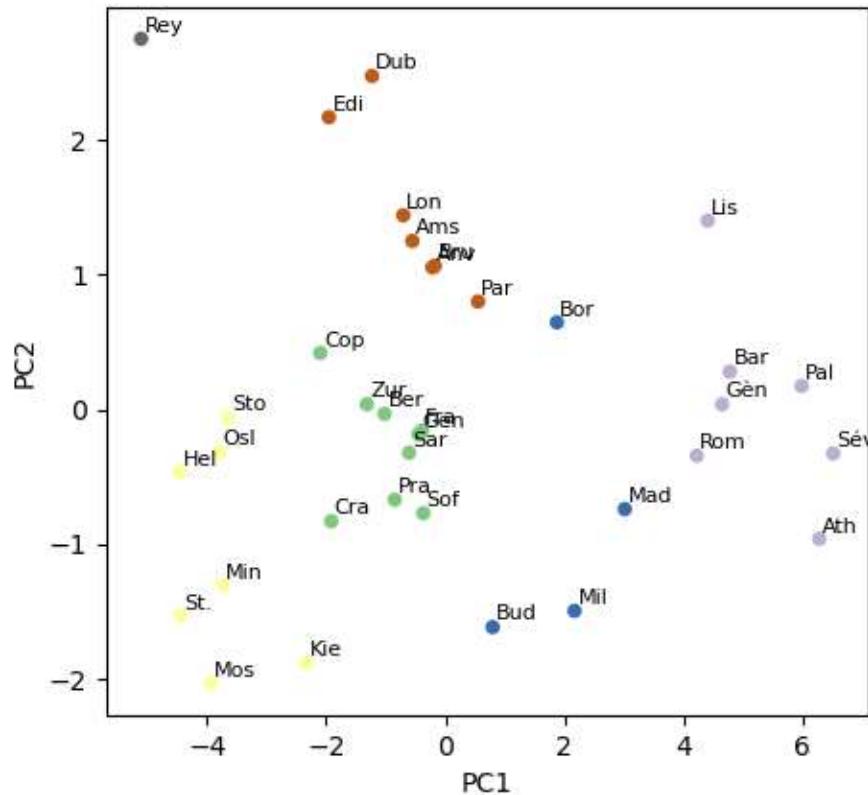
$(C_1, C_2, C_3) = ((3.3, 5.2), (1.3, 1.5), (4.7, 3.1))$

A partir de cette configuration les individus ne changent plus de groupe l'algorithme s'arrête.



K-means

Exemple sur les capitales



En croisant les groupes de cah et de kmeans
print(pd.crosstab(cluster.labels_, kms.labels_))

	0	1	2	3	4	5
1	0	7	0	0	0	0
2	7	0	1	0	0	0
3	0	0	0	0	0	1
4	0	0	0	0	4	0
5	0	0	8	0	0	0
6	0	0	0	7	0	0

Les groupes ne sont pas numérotés de la même façon mais les résultats montrent qu'une seule ville est classée différemment entre cah et k-means.

K-means

Avantages

- *L'algorithme k-means possède bien des avantages, mais aussi des inconvénients, en particulier il est très facile à implémenter.*
- *L'algorithme est flexible, il s'adapte aux changements sur les données.*
- *L'algorithme permet de partitionner les gros datasets.*
- *Les temps de calcul sont acceptables, car la complexité de l'algorithme est de l'ordre $O(kmn^2)$ où k est le nombre de cluster n le nombre d'individus et m le nombre de variables.*

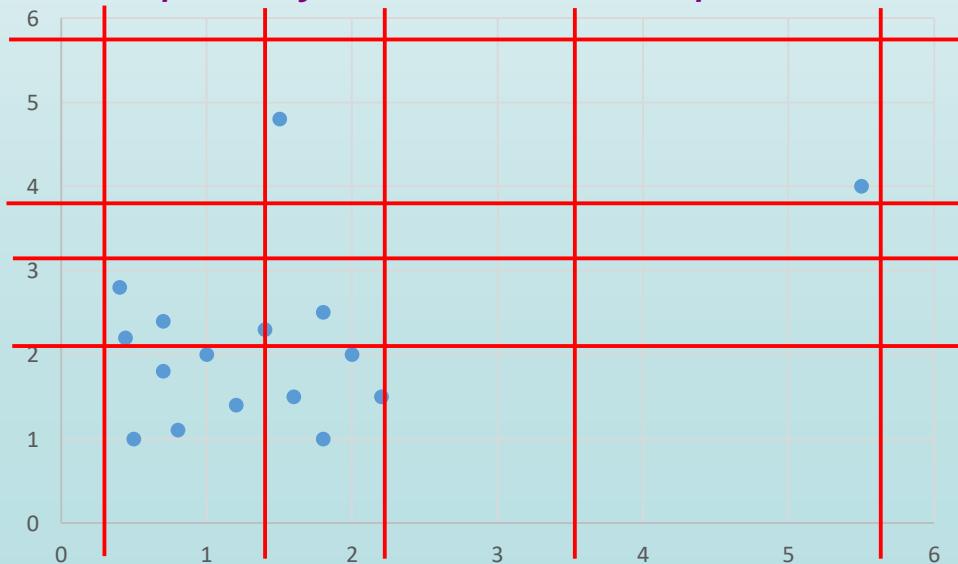
Inconvénients

- *Le nombre de clusters doit être défini à l'avance.*
- *L'algorithme converge vers des optimums locaux, en fonction du choix des centroïdes initiaux. Pour éviter cela il est préférable de choisir des centroïdes très éloignés des nuages de points et les uns des autres.*
- *Le modèle est fortement influencé par les valeurs aberrantes (outliers).*
- *Des résultats médiocres sur des données non linéairement séparables.*

Isolation Forest

Isolation Forest

- *Isolation Forest est une technique de détection d'anomalies (outliers). On cherche à identifier les individus dont les caractéristiques sont très éloignées de celles des autres individus.*
- *La méthode consiste à effectuer une série de splits de manière aléatoire, afin d'isoler les individus éloignés de la majorité.*
- *Plus le nombre de splits nécessaire pour isoler un individu est petit et plus il y aura de chance que cet individu soit une anomalie.*



Un premier tirage aléatoire nous donne les splits :
 $(Y, 5.8)$; $(Y, 3.1)$; $(X, 3.5)$...

Au troisième split les individus $(1.5, 4.8)$ et $(5.5, 4)$ sont isolés.

Le second tirage donne les résultats:
 $(X, 1.4)$; $(X, 5.8)$; $(Y, 2.1)$; $(Y, 3.8)$; $(X, 0.2)$; $(X, 2,3)$...

Au quatrième split l'individu $(1.8, 2.5)$ est isolé

Au sixième split les individus $(1.5, 4.8)$ et $(5.5, 4)$ sont isolés.

Afin d'éviter de traiter le point $(1.8, 2.5)$ comme une anomalie, l'algorithme effectue une série de splits et conserve la moyenne du nombre de splits nécessaire pour isoler les individus.

Isolation Forest

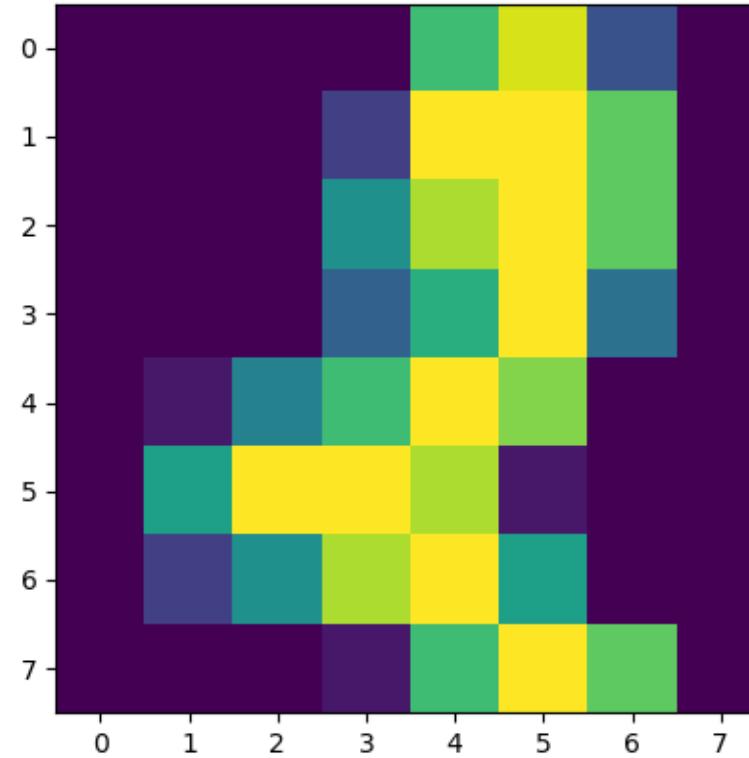
Exemples

- Détection d'anomalies

```
from sklearn.ensemble import IsolationForest
X = np.loadtxt("Data1F.txt")
model = IsolationForest(contamination=0.1)
model.fit(X)
# Predict est un tableau de 0 ou 1
plt.scatter(X[:,0], X[:,1], c=model.predict(X))
plt.show()
```

- Identification d'anomalies

```
digits = load_digits()
images = digits.images
X = digits.data ; y=digits.target
model = IsolationForest(contamination=0.01)
model.fit(X)
anomalies = np.where(model.predict(X) == -1)[0]
plt.imshow(images[anomalies][0])
plt.title(y[anomalies][0])
```



DBSCAN

Les anomalies avec DBSCAN

- *L'algorithme DBSCAN utilise deux paramètres, le rayon ε et le nombre minimum de points (min_samples) devant se situer dans ce rayon.*
- *Le principe de la méthode consiste :*
 - *Compter le nombre de points dans le rayon ε d'un individu.*
 - *Un individu qui compte au moins (min_samples-1) voisins, est considérée comme un individu cœur.*
 - *Les individus au voisinage d'un cœur appartiennent au même cluster. Si des individus cœurs sont proches on obtient une séquence d'individus cœur qui constitue un unique cluster.*
 - *Un individu qui n'a pas d'individu cœur dans son voisinage est considérée comme une anomalie.*
- *DBSCAN utilise la distance euclidienne comme métrique.*
- *Il n'est pas nécessaire de définir le nombre de clusters et de plus l'algorithme permet de gérer les valeurs aberrantes ou les anomalies.*
- *Le principale problème consiste à choisir la bonne valeur pour epsilon.*

Apprentissage non supervisé

Exemple

- Un modèle de type *NearestNeighbors* peut être entraîné pour obtenir une estimation de la valeur d'*epsilon*.
- La valeur d'*epsilon* peut être choisie de telle sorte que +90% des individus aient un voisin.

Création d'un jeu de données

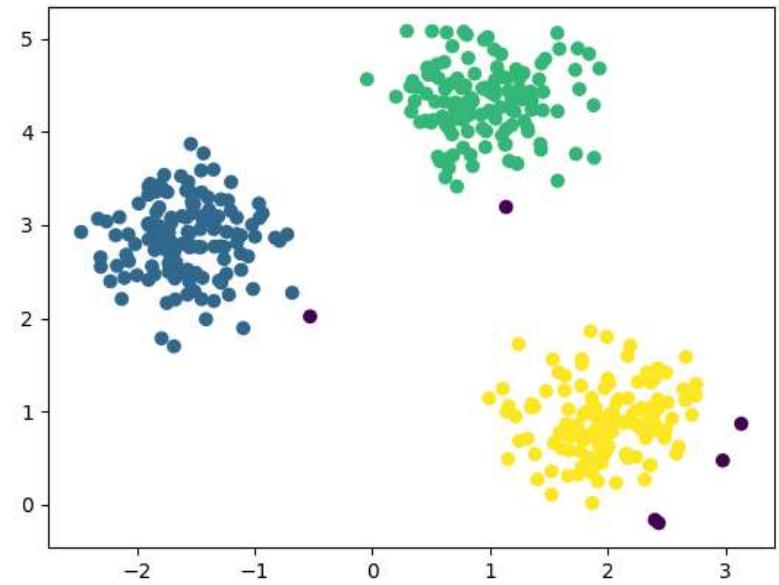
```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=400, cluster_std = 0.4,
plt.scatter(x=X[:,0], y=X[:,1], s=3)
```

Calcul des distances

```
from sklearn.neighbors import NearestNeighbors
model = NearestNeighbors(n_neighbors=2).fit(X)
distances, indexes = model.kneighbors(X)
plt.plot(np.sort(distances, axis=0)[:, 1])
```

Calcul des distances

```
from sklearn.cluster import DBSCAN
model = DBSCAN(eps = 0.4, min_samples=5)
y_pred = model.fit_predict(X)
plt.scatter(X[:,0], X[:,1], c = y_pred);
```



Démarche d'apprentissage

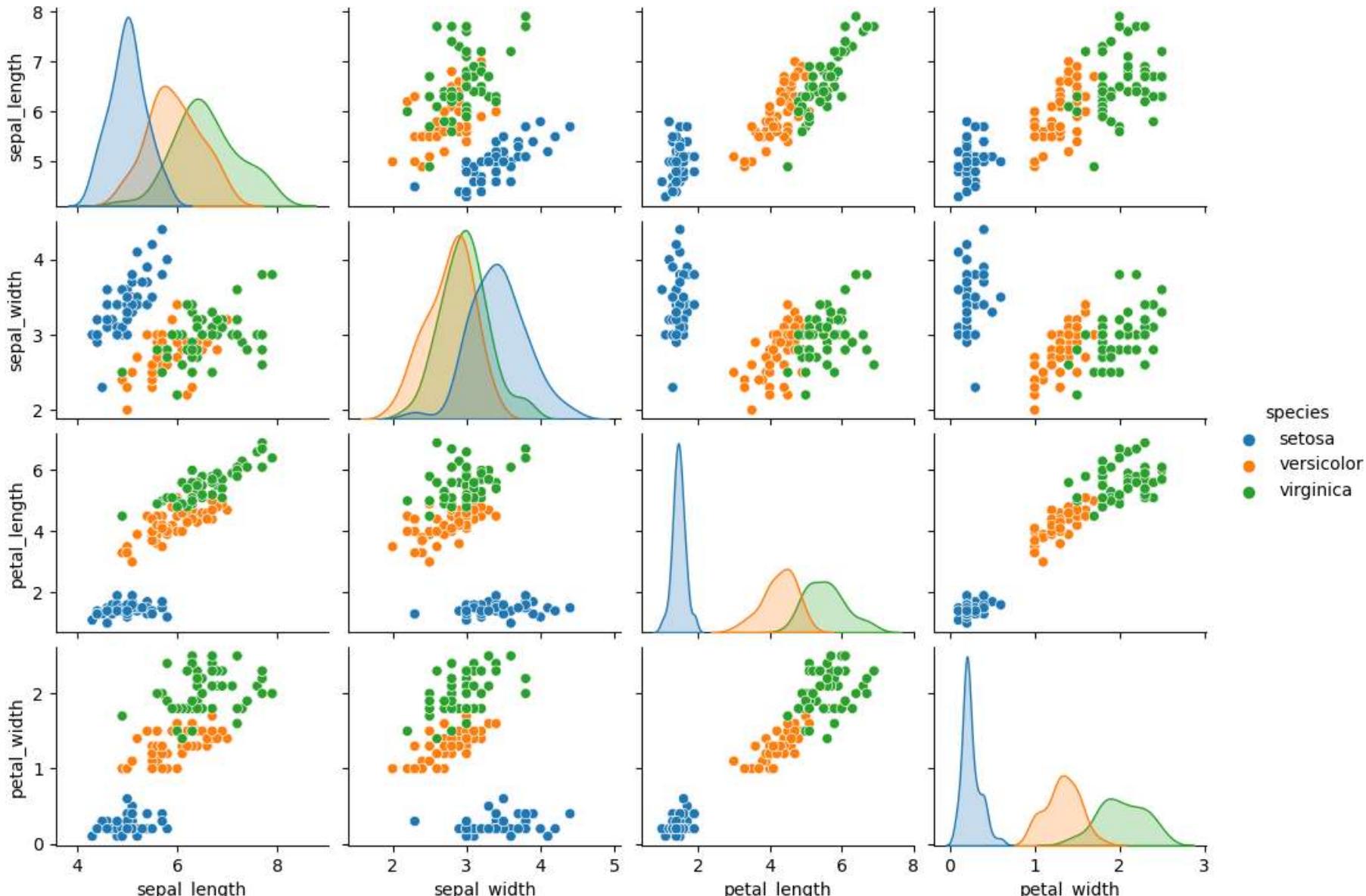
Les Etapes du machine learning

- Pré-traitement des données
- Traitement des données
- Exemple
- Validation d'un modèle

Pré-traitement des données

Présentation

- *Le prétraitement est une étape importante, car il est impossible pour les algorithmes d'utiliser des données brutes, issues de la collecte.*
- *Après avoir été récoltées, la qualité des données doit être vérifiée.*
- *Ce travail début par une analyse fine du DataSet et par l'identification des différentes variables impliquées dans le processus d'apprentissage.*
- *Le nettoyage consiste à "réparer" les erreurs, à remplacer ou à supprimer les données manquantes et à supprimer les doublons.*
- *L'analyse des corrélations et des dépendances entre les variables permet de bien cerner le problème, de supprimer des variables inutiles, d'en regrouper certaines ou d'en créer de nouvelles.*
- *Transformer les données catégorielles en données numériques.*
- *Mettre toutes les données à une échelle commune (centrées-réduites).*
- *Eventuellement réduire le volume de données pour faciliter le traitement sans perdre en qualité.*



Pré-traitement des données

Analyse globale du Dataset

- *Etape 1 : Récupération des données*
 - Télécharger le DataSet. Bien qu'il soit possible d'utiliser des tableaux numpy, il est plus efficace d'utiliser des DataFrame
 - **Principales fonctions de pandas** : pd.read_csv(), pd.read_excel(), pd.read_json(), pd.read_html(),
- *Etape 2 : Analyse globale du DataSet*
 - Quels est la taille du DataSet, quelles sont les Features
 - **Principales propriétés** : Data.shape ; Data.columns ; Data.dtypes
 - **Méthodes** : Data.info(), Data.describe(), Data.value_counts()
 - **Visualisation** : sns.histplot(), sns.boxplot()
- *Etape 3 : Type de problème*
 - *Type d'apprentissage* : Supervisé ou non supervisé.
 - *Supervisé* : target le type d'algorithme régression classification.

Pré-traitement des données

Nettoyage des données

- *Etape 4 : Nettoyage des données*
 - Identifier les features qui n'ont pas d'impacte sur le résultat, les éléments dupliqués et les supprimer si nécessaire.
 - **Méthodes** : Suppression : `Data.drop()` , `Data.drop_duplicates()`. Identification : `Data.duplicated()`.
 - Identifier et supprimer les outliers. Après identification des valeurs aberrantes la suppression utilise le boolean indexing.
 - Identifier puis remplacer ou supprimer les valeurs manquantes.
 - **Principales méthodes** : Identification - `Data.isnull()`, `Data.isna()` ; Suppression – `Data.dropna()`; Modification – `Data.fillna()`.
 - Identifier les données mal orthographiées, mal renseignées.
 - **Principale méthode** : `Data.replace({ 'dictionnary' })`.

Pré-traitement des données

Relations entre les variables

- *Etape 5 : Analyser les relations entre les variables.*
 - *Visualisation des relations entre les features et la target.*
 - **Principales méthodes** : `sns.pairplot()`, `sns.lmplot()`, `sns.kdeplot()`,
`sns.scatterplot()`, `sns.barplot()`, `sns.histplot()`, `sns.countplot()`,
`sns.lineplot()`.
 - *Relation linéaire entre une features et la target.*
 - **Principales méthode** : `mutual_info_regression()`
 - *Analyse des corrélations* : `Data.corr()`.
 - *Sélection d'une catégorie de données* : `Data.groupby()`.
- *Etape 6 : Modification du dataSet.*
 - *Il peut être nécessaire d'ajouter des informations sur la base de features existantes.*

Pré-traitement des données

Modification des données

- *Etape 7 : Encoder les données catégorielles.*
 - Les données catégorielles doivent être transformées en données numériques. Deux types d'encodage (`sklearn.preprocessing`)
 - **Encodage ordinal via des transformers** : `LabelEncoder` et `OrdinalEncoder`. Utilisation des méthodes `fit_transform()`
 - **Encodage One Hot** : Chaque catégorie d'une variable est représentée par une colonne binaire spécifique : `pd.get_dummies()` ou `OneHotEncoder()`.
 - **Principale méthode** : `Data.replace()`.
- *Etape 8 : Transformation des données.*
 - Il peut être nécessaire de les centrer et réduire les données.
 - **Bibliothèque** : `sklearn.preprocessing – StandardScaler`
 - Utilisation de l'ACP pour conserver les informations importantes et en supprimant le « bruit ».

Traitement des données

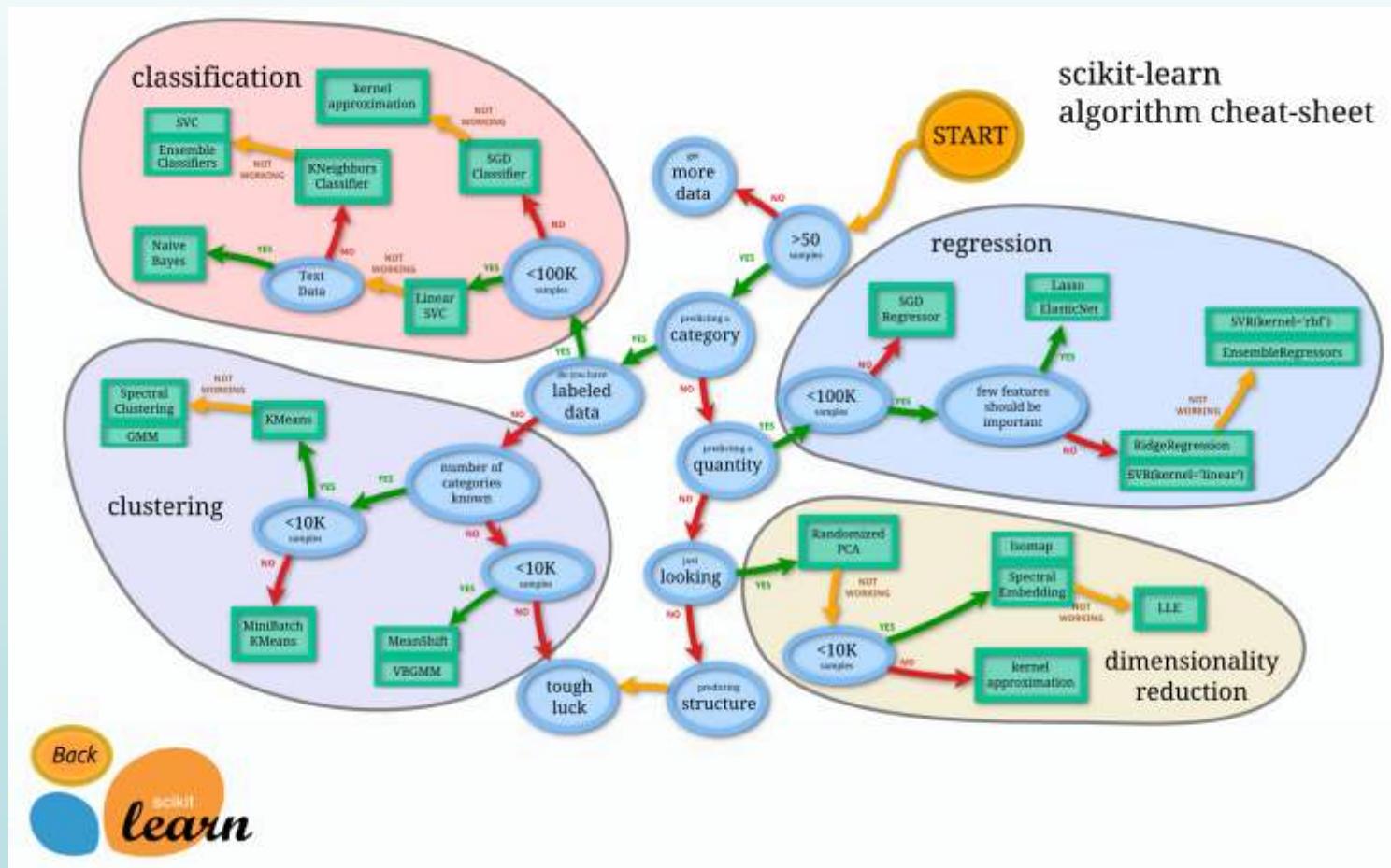
Apprentissage

- *Etape 10 : Séparation les données.*
 - Séparer les données en un *trainSet* et un *testSet*
 - **Bibliothèque** : *sklearn.model_selection - train_test_split*.
- *Etape 11 : Apprentissage.*
 - Choisir un algorithme d'apprentissage adapté au problème.
 - **Classification** : *linear_model, KNeighborsClassifier, SVC, DecisionTreeClassifier, RandomForestClassifier, neural_network ...*
 - **Regression** : *SVR, neighbors, RandomForestRegression, tree, neural_network ...*
 - **Clustering** : *Kmeans, DBSCAN, IsolationForest*
- *Etape 12 : Calage et amélioration des modèles.*
 - **Principaux outils** : *cross_val_score, confusion_matrix, validation_curve*

Traitement des données

Choix de la méthode d'apprentissage

- sklearn propose une démarche permettant de choisir une méthode.



Exemple

Traitement d'un jeu données : Pinguins

- *Etape 1 : Récupération des données – lecture d'un fichier csv*

```
Data = pd.read_csv(' .../Penguins.csv', delimiter=';')
```

- *Etape 2 : Analyse globale du DataSet*

Informations générales sur le DataSet

```
Data.head()          # retourne les 5 premières samples
Data.shape           # 344 samples et 8 features
Data.dtypes          # float64 (bill_length_mm, bill_depth_mm, flipper_length_mm,
                    # body_mass_g) ; object (species, island, sex) ; int64 (year)
```

Contenu du DataSet

```
Data.info() ou Data.isna()      # 2 infos manquent dans les features de type float64
et onze dans la feature sex
Data.describe()                 # Il ne semble pas y avoir de outliers, les moyennes
                                # sont très différentes, il sera nécessaire de les normaliser.
Data.[Data['bill_length_mm'].isna()] # On constate que deux individus n'ont aucune
                                    # information sur les données de type float. Elles ne peuvent pas être utilisées
```

Exemple

Pré-traitement

- *Etape 2 : Analyse globale du DataSet*

Détaille sur les variables catégorielles

```
for col in Data.select_dtypes(include='object') # parcours des variables catégorielles
    print(Data[col].value_counts())
# 3 species : [Adelie : 152 ; Gentoo : 124 ; Chinstrap : 68 ]
# 3 island : [ Biscoe : 168 ; Dream : 124 ; Torgersen : 52 ]
# 2 sexe : [ male : 168 ; female : 165 ]
Il n'y a pas de données mal orthographiées. Il nous faudra vérifier lors du split que les proportions (species, island et sexe) seront respectées entre les trainSet et les testSet.
```

- *Etape 3 : Type de problème*

Apprentissage supervisé – Target = ‘species’ – problème de régression

- *Etape 4 : Nettoyage des données*

La variable year

```
sns.heatmap(Data.corr(), annot=True)      # Comme on pouvait s'y attendre la
                                            # variable year n'est pas corrélée aux autres variables numérique.
print(Data.groupby(['sexe', 'year']).size())   # Pas de corrélation entre sexe et year
print(Data.groupby(['criteries', 'island', 'year']).size()) # Pas plus qu'avec les autres
```

Exemple

Nettoyage des données

- Etape 4 : Nettoyage des données

```
Suppression de la variable 'year'
Data.drop('year', axis=1, inplace=True)
Suppression des lignes non renseignées
Data.dropna(subset=['bill_length_mm'], axis=0, inplace=True) # 342 samples 7 features
Aucune duplication de variables
Data.duplicated().value_counts() # False = 342 => True = 0
Aucune donnée n'est mal orthographié : etape 2 - Il n'y a pas d'outlier : etape 2
```

Cas des données manquantes sur le sexe des pingouins

```
Data['sex'].value_counts() # False = 333 True = 9
Data[Data['sex'].isna()==True].index # [8, 9, 10, 11, 47, 178, 218, 256, 268]
4 sur 51 valeurs pour (Adelie, Torgersen) ; 1 sur 56 pour (Adelie, Dream) et 4 sur 123
pour (Gentoo, Biscoe). La répartition male/femelle est équivalente, on peut supposer
que cette variable n'a pas d'impact. Il est donc possible de conserver les individus.
Data['sex'] = Data['sex'].replace(np.nan, np.random.choice(['male', 'female']))
```

*Les pingouins de Torgersen sont de l'espèce Adelie. Il n'y a donc aucune variabilité pour
cette île, on peut donc supprimer tous ces individus.*

```
Data = Data[ Data['island'] != 'Torgersen']
```

Exemple

Visualisation des relations

- *Etape 5 : Ajout de nouvelles features : Non nécessaire.*
- *Etape 6 : Analyser les relations entre les variables.*

```
sns.pairplot(Data, hue='species')           # Analyse Features / target
Data.groupby(['species'])['bill_length_mm'].mean()    # 39mm contre 48mm
# Pingouins Adelie : ont un bec nettement plus petits que les autres
# Pingouins Gentoo : se différencient par de plus grandes nageoires, une masse
corporelle plus importante, et une profondeur de bec moins importante.
Ces différences devraient permettre de classer facilement les individus
```

Analyse Features (numérique) / 'island'

```
sns.pairplot(Data, hue='island')
# Les pingouins sur l'ile de Biscoe ont un bec un peu plus profond, des nageoires plus
courtes et une masse corporelle un peu moins importante. Ils ont alors plus de chance
d'appartenir à l'espèce Gentoo, qu'aux autres espèces.
```

Analyse Features (numérique) / 'sex'

```
sns.pairplot(Data, hue='sex')
# Les femelles sont moins grandes et moins grosses que les males. Mais ce paramètre
ne devrait pas avoir un impact significatif sur le classement.
```

Exemple

Encodage des données

- *Etape 7 : Encoder les données catégorielles.*

```
# Il n'est pas nécessaire d'encoder les variables 'island' et 'sex' en One hot car elles
n'ont que deux valeurs différentes. 'species' doit aussi être codée de manière ordinal.
from sklearn.preprocessing import OrdinalEncoder
encoder = OrdinalEncoder()
columns_object = Data.select_dtypes(include='object').columns # noms des colonnes
Data[columns_object]=encoder.fit_transform(Data[columns_object])
```

- *Etape 8 : Rendre les variables comparables*

```
#Seules les variables de type float peuvent être centrées et réduites
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
columns_float = Data.select_dtypes(include='float64').columns
Data[columns_float]=sc.fit_transform(Data[columns_float])
```

```
# Ces deux transformations peuvent se faire en utilisant un make_column_transformer
from sklearn.compose import make_column_transformer
transformer = make_column_transformer( (OrdinalEncoder(), columns_object),
                                         (StandardScaler(), columns_float))
columns = columns_object.append(columns_float )
Data[columns] = transformer.fit_transform(Data)
```

Exemple

Traitement et apprentissage

- *Etape 9 : Réduire la dimension : Non nécessaire.*
 - *Etape 10 : Séparer les données.*

- #### ▪ *Etape 11 : Apprentissage*

Validation d'un modèle

Evaluation

- Les données pour évaluer un modèle ne doivent pas servir lors de la phase d'apprentissage.
- Le Dataset doit alors être divisé en deux parties : des données d'entraînement ou TrainSet et des données de test ou TestSet.
- La fonction `train_test_split` du module `modele_selection` permet de découper un Dataset en un jeu pour l'entraînement et un pour le test.

```
# Récupération de la fonction de découpage des données en Train et Test
from sklearn.model_selection import train_test_split
iris = datasets.load_iris()
# Découpage du DataSet en un jeu de données pour l'apprentissage et un pour l'évaluation
Xtrain, Xtest, Ytrain, Ytest = train_test_split(iris.data, iris.target, test_size=0.2)
model = KNeighborsClassifier(n_neighbors=3)          # Modèle trois plus proches voisins
model.fit(Xtrain, Ytrain)                          # Apprentissage : données entraînement
print(model.score(Xtest, Ytest))                  # Evaluation : données de test
```

```
from sklearn.metrics import confusion_matrix
# Retourne les erreurs de classement du modèle
confusion_matrix(Ytest, model.predict(Xtest))
```

Evaluation de la pertinence d'un modèle

Validation d'un modèle

Les hyper-paramètres

- Les *hyper-paramètres* sont des paramètres qui ne sont pas directement appris par les modèles, ils doivent être ajustés.
- Les algorithmes d'apprentissage, de clustering ... utilisent des *hyper paramètres* spécifiques.
- Dans scikit-learn, ils sont passés en tant qu'arguments au constructeur des classes du modèle. La fonction `modèle.get_params()` retourne la liste des *hyper-paramètres* du modèle ainsi que leurs valeurs.
- Pour régler les *hyper-paramètres* et augmenter le score de la fonction d'apprentissage, on doit disposer d'un jeu de données, différent du *TestSet* et du *TrainSet*.
- On découpe alors le *Dataset* en un *TrainSet* un *TestSet* et *ValidationSet*.
- On entraîne le modèle sur le *TrainSet* avec différentes valeurs des *hyper-paramètres*, le *ValidationSet* permet de retenir le meilleur réglage.
- Le modèle avec le meilleur réglage peut alors être évalué sur le *TestSet*.

Validation d'un modèle

Cross validation

- Un modèle doit être évaluer sur des données différentes de celles utilisées pour l'apprentissage, issues du découpage d'un Dataset. Mais les résultats peuvent différer d'un découpage à l'autre.
- Pour s'assurer de la robustesse du modèle, on utilise la cross validation.
- Pour cela on découpe le TrainSet en n segments de manière aléatoire et on en utilise 1 pour la validation et les autres pour l'apprentissage.
- On effectuera alors $n-1$ séries d'apprentissages avec à chaque fois un segment de validation différent.
- La fonction `cross_val_score` permet d'évaluer un modèle en utilisant la cross validation.

```
# Récupération de la fonction pour la cross validation
from sklearn.model_selection import cross_val_score
iris = datasets.load_iris()
Xtrain, Xtest, Ytrain, Ytest = train_test_split(iris.data, iris.target, test_size=0.2)
model = KNeighborsClassifier(n_neighbors=3)          # Modèle trois plus proches voisins
score = cross_val_score(model, Xtrain, Ytrain, cv=5) # Découpage du TrainSet en 5 segments.
```

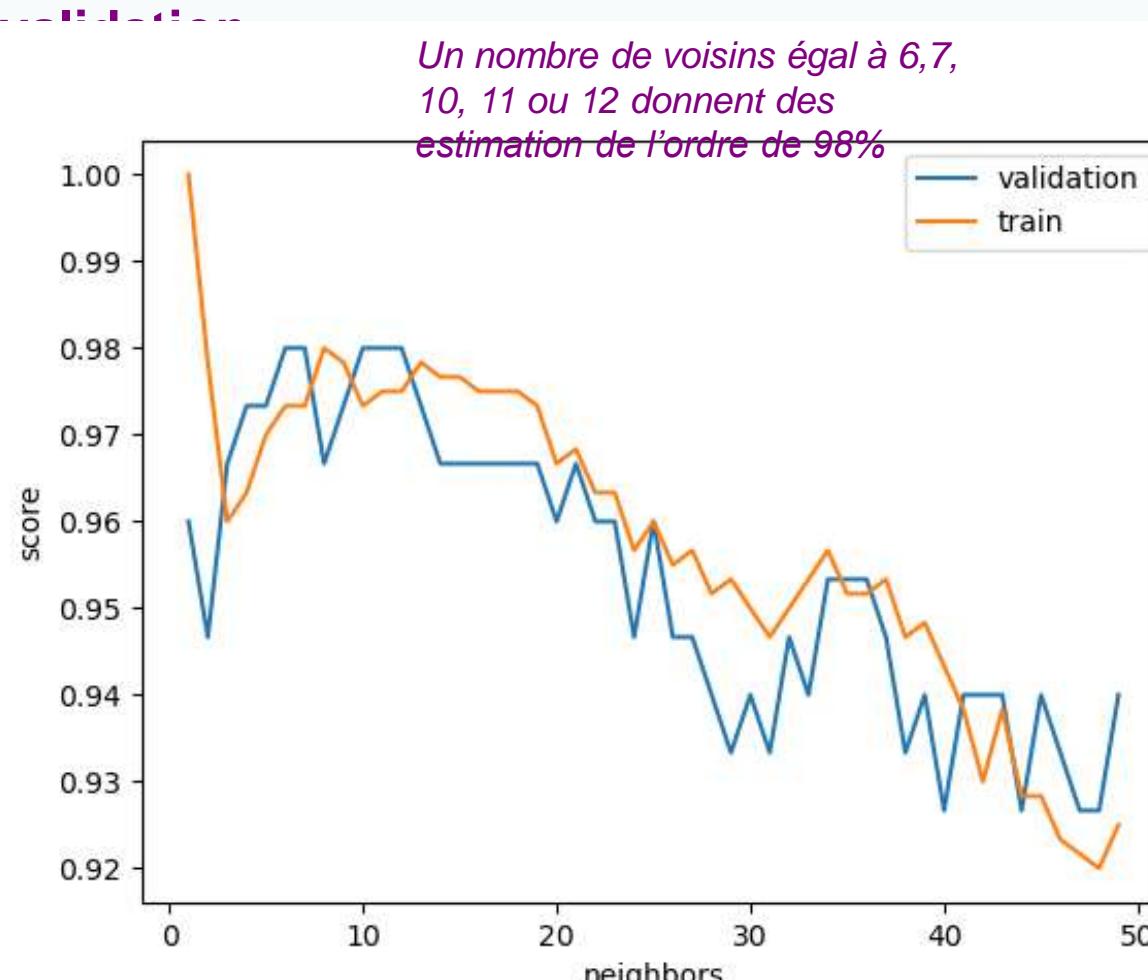
Validation d'un modèle

Courbes de validation

- Afin de paramétriser les hyperparamètres
- La fonction `cross_val_score` effectue des tests automatiques
- La fonction `KNeighborsClassifier` gère l'hyper-paramètre `n_neighbors`

```
# Récupération de l'ensemble de données
from sklearn.datasets import load_iris
iris = datasets.load_iris()
model = KNeighborsClassifier(n_neighbors=3)
```

```
# valScore est ici une liste de scores
plt.figure()
plt.plot(k, valScore)
plt.plot(k, trainScore)
plt.xlabel('neighbors') ; plt.ylabel('score') ; plt.legend() ; plt.show()
```



Validation d'un modèle

Les réglages

- L'algorithme `KNeighborsClassifier` est configuré avec 8 hyper-paramètres, comme l'algorithme, la métrique ou le nombre de voisins.
- Grid Search permet le réglage des hyper-paramètres, pour chaque combinaisons des valeurs des hyper-paramètres spécifiés dans un dictionnaire, le programme crée un modèle associé et évalue le meilleur.
- Si `n_neighbors`, `algorithm` et `metric` sont configurer pour tester (20, 4, 3) valeurs différentes la fonction construira $20 * 4 * 3 = 240$ modèles.

```
from sklearn.model_selection import GridSearchCV
# Création d'un dictionnaire des hyper-paramètres à tester avec les valeurs associées
params_grid = {'n_neighbors': np.arange(1,20), 'metric': ['euclidean', 'manhattan', 'minkowski'],
               'algorithm': ['auto', 'ball_tree', 'kd_tree','brute']}
iris = datasets.load_iris()
Xtrain, Xtest, Ytrain, Ytest = train_test_split(iris.data, iris.target, test_size=0.2)
grid = GridSearchCV(KNeighborsClassifier(), params_grid, cv=5)
grid.fit(Xtrain, Ytrain)
grid.best_score_ ; grid.best_params_
model = grid.best_estimator_
# Retourne le meilleur score , les paramètres
# Retourne le meilleur modèle
```

Validation d'un modèle

Quantité de données

- L'objectif est de maximiser la performance
- La fonction de validation fonctionne

```
from sklearn.model_selection import cross_val_score
# Cette fonction renvoie les scores pour un nombre de décompositions donné
model = KNeighborsClassifier(n_neighbors=3)
percent = np.linspace(10, 120, 11)
N, trainScore, valScore = percent, np.zeros(len(percent)), np.zeros(len(percent))
plt.figure()
plt.plot(N, valScore)
plt.plot(N, trainScore)
plt.xlabel('percent')
```

