

Introduction



GÉNÉRALITÉS : INTELLIGENCE
ARTIFICIELLE & MACHINE LEARNING

Dr. Chaabani Yasmine

yasmin.chaabani@isgb.ucar.tn

Le but du Module ?

Connaitre la place du Machine Learning dans l’Intelligence Artificielle

Avoir une présentation du Machine Learning

Identifier un problème en Machine Learning

Sommaire

Qu'est-ce que le Machine Learning ?

Quels sont les enjeux du Machine Learning ?

Comment faire du Machine Learning ?

Quels sont les langages et outils pour faire du Machine Learning ?

Qui fait du Machine Learning ?

Ce qu'il faut retenir du Machine Learning

Qu'est-ce que le Machine Learning ?

Qu'est-ce que le Machine Learning ?



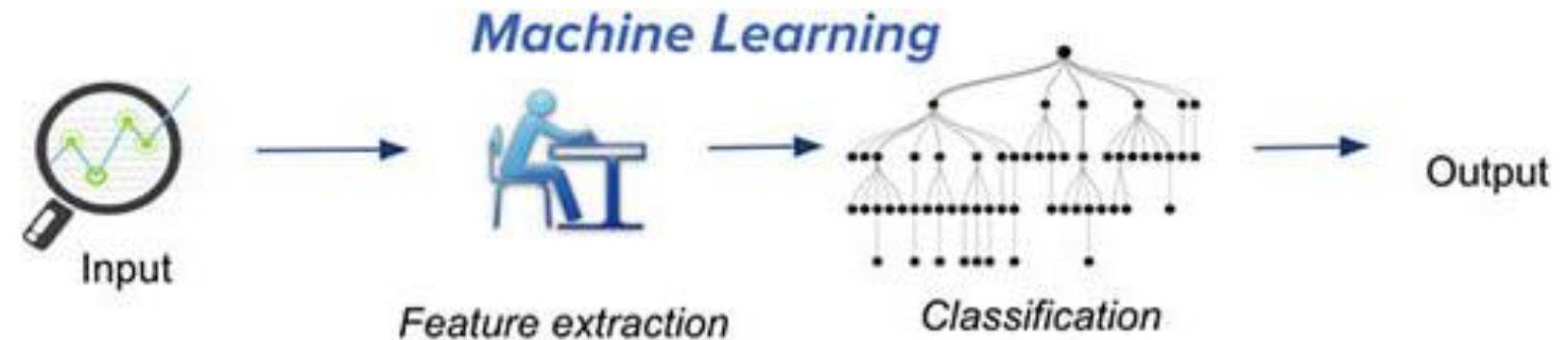
« L'apprentissage automatique est la discipline donnant aux ordinateurs la capacité d'apprendre, sans qu'ils soient explicitement programmés. » - Arthur Samuel (1901-1990)

Arthur Samuel, Américain, Formé au MIT, serait le premier en 1952 créateur d'un programme d'auto-apprentissage (Jeu de Dames)

Qu'est-ce que le Machine Learning ?

Première formulation :

- Les systèmes de Machine Learning doivent apprendre comment combiner des entrées pour formuler des prédictions sur des données qui n'ont pas encore été observées
- Les principes de Machine Learning sont basés sur des probabilités et des statiques et non sur de la logique



Qu'est-ce que le Machine Learning ?

Le Machine Learning est une technologie (ensemble de techniques permettant à la machine de résoudre des problèmes pour les humains) :

- Des données
- Des entraînements
- Des tests
- Des résultats

Qu'est-ce que le Machine Learning ?

Faire des prédictions à partir d'observations pour une ou plusieurs variables numériques (régression) ou catégorielles (classification)

Ajuster les résultats avec une fonction de coût pour savoir si les prédictions sont exactes

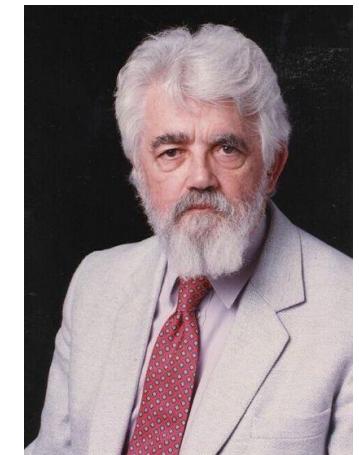
Faire des minimisations pour rendre vrai les prédictions

Idée principale : Apprendre à une machine comment obtenir/construire des résultats

Qu'est-ce que le Machine Learning ?

Qu'est-ce que l'intelligence Artificielle, un peu d'histoire :

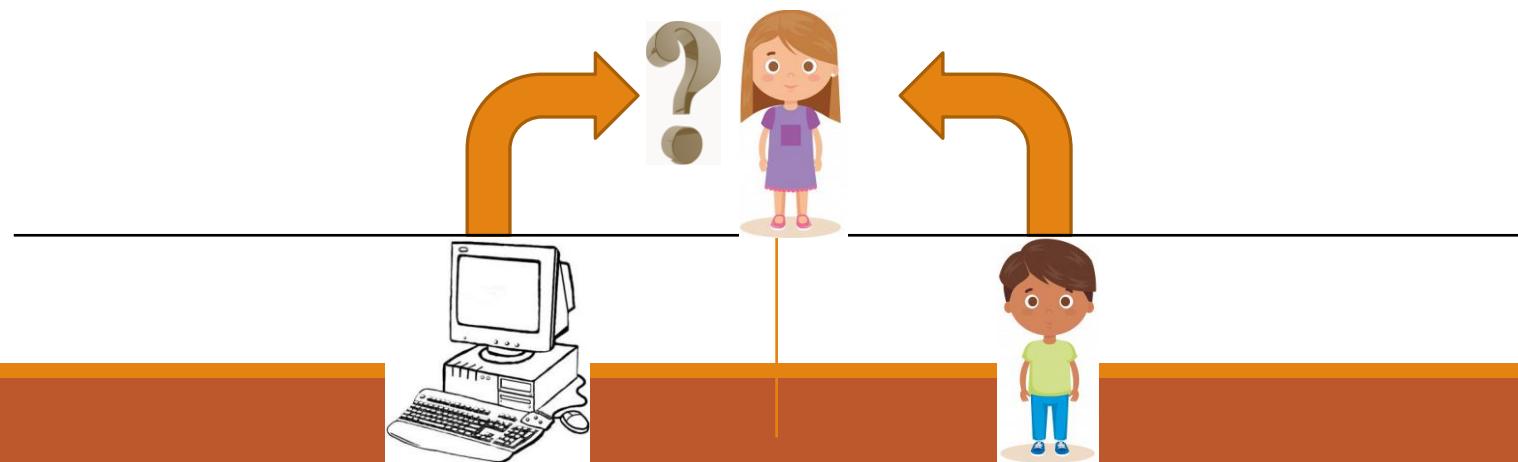
- « Toute activité intellectuelle peut être décrite avec suffisamment de précision pour être simulée par une machine » - John McCarthy (1927 - 2011), principal pionnier de l'intelligence artificielle
- Concevoir des systèmes capables de reproduire l'humain dans ses activités de raisonnement et de comportement



Qu'est-ce que le Machine Learning ?

Qu'est-ce que l'intelligence Artificielle, un peu d'histoire :

- L'intelligence artificielle se développe dans les années 1950
- Empirique = Hypothèses + confirmations expérimentales
- Approche rationaliste = Combinaisons d'outils + ingénierie
- Alan Turing (1912 - 1954) = Définition opérationnelle de l'intelligence artificielle
- Test sur l'indiscernabilité entre l'humain et la machine



Qu'est-ce que le Machine Learning ?

Pour une machine, le test de Turing c'est :

- Traiter le langage naturel pour pouvoir réellement communiquer
- Représenter des connaissances pour stocker le savoir et avoir un raisonnement pour répondre correctement aux questions
- Tirer de nouvelles conclusions
- Être capable d'apprendre
- S'adapter aux nouvelles circonstances

1955 conception de la logique théorique par Newell et Simon

- 1^{er} raisonnement d'une machine



Qu'est-ce que le Machine Learning ?

Après le creux des années 1980, 1990 par manque d'utilisation de grand volume de données, l'IA se focalise sur des tâches spécifiques :

- La traduction automatique
- Aide aux diagnostiques
- La conduite de voiture
- Les détections d'actions (fraudes, manipulations, ...)
- Les assistants personnels
- Les reconnaissances

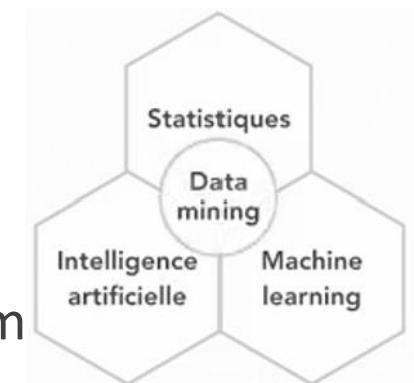
L'apprentissage permet de faire ses tâches et induit le Machine Learning

Qu'est-ce que le Machine Learning ?

La tendance actuelle est :

- Permettre aux ordinateurs d'apprendre automatiquement sans intervention ou assistance humaine
- Se concentrer sur le développement de programmes qui peuvent accéder aux données et les utiliser pour apprendre par eux-mêmes
- Ajuster des actions en conséquence des apprentissages
- S'améliorer avec de l'entraînement

Les processus de Machine Learning sont similaires aux Data Mining ou à la m
prédictive



Qu'est-ce que le Machine Learning ?

Pourquoi l'IA parle à tous ?

- Puissance de calcul augmente (Architecture des processeurs)
- Puissance de stockage (Réseaux, Architecture)
- Disponibilité des données (Réseaux, Cloud)
- Data mining (Big Data)

L'IA approche multidisciplinaires ?

- Réseau bayésiens (modèle probabiliste)
- Modèle de Markov (modèle statistique)
- Deep Learning (réseaux de neurones)
- ...

Qu'est-ce que le Machine Learning ?

Les intelligences artificielles s'appuient sur des modèles :

- Un modèle est une proposition mathématique pour résoudre un problème observé
- Un modèle ne donne pas toujours une solution exacte, il y a des erreurs plus ou moins importantes qu'il faut minimiser
- Un modèle peut ne pas toujours refléter la réalité, il faut prendre beaucoup de soin pour trouver le modèle qui fonctionne et qui correspond à aux demandes

Qu'est-ce que le Machine Learning ?

L'apprentissage commence :

- A partir de données
- Par des observations
- Par des prédictions
- Il suit un modèle mathématique

Les données sont

- Des exemples
- Des expériences
- Des instructions
-

Qu'est-ce que le Machine Learning ?

Un but est :

- Rechercher les tendances dans les données
- Vérifier que les prédictions sont exactes en fonction des données et même avec de nouvelles données
- Prendre de bonnes décisions dans l'avenir en fonction des données

Qu'est-ce que le Machine Learning ?

Avant les années 2000 :

- Le Machine Learning = un problème d'optimisation
- Définition d'une fonction d'erreur pour déterminer le modèle qui minimise cette erreur

Depuis les années 2010 :

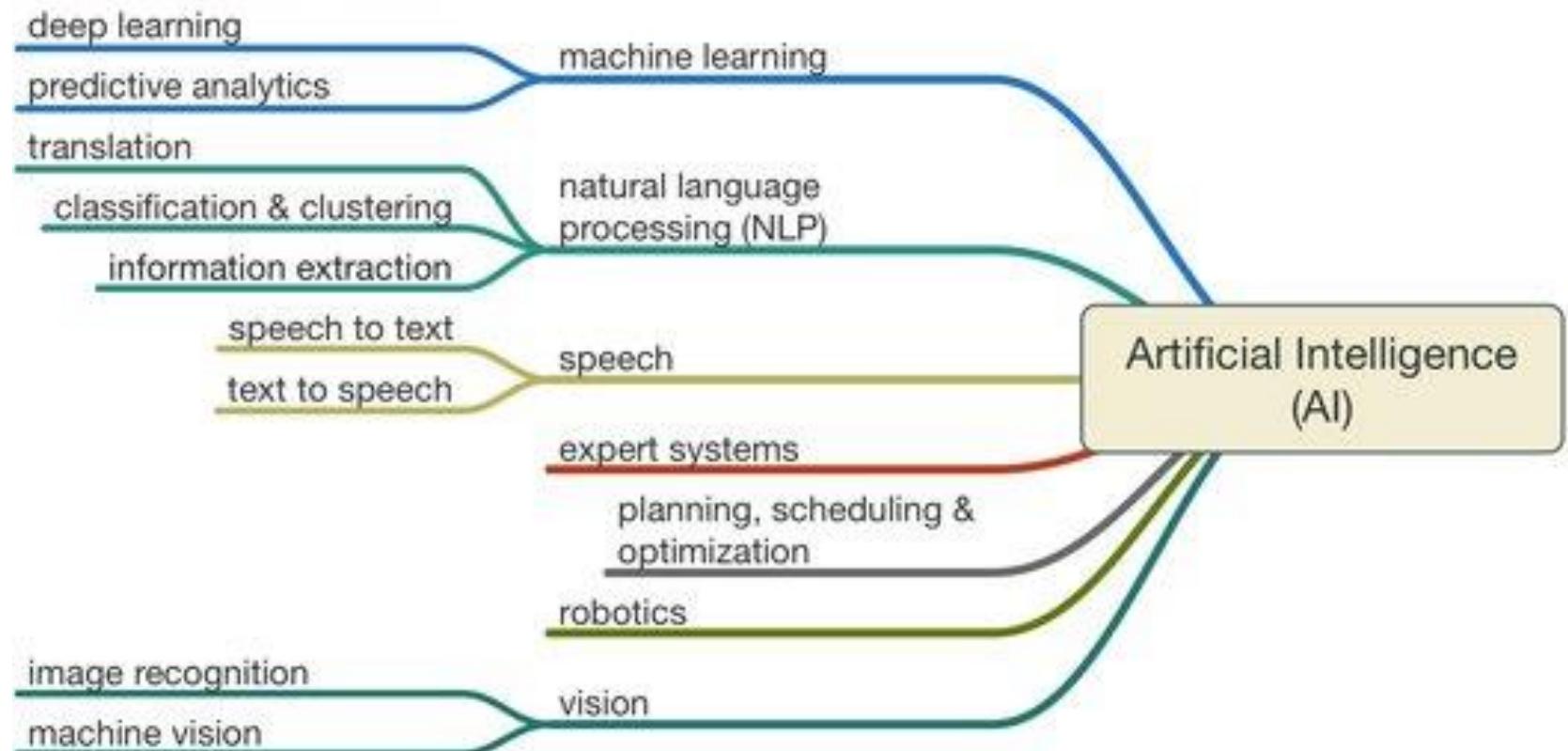
- Série de problèmes identifiés correspondant à des cas concrets
- Démocratisation des Classifications, régressions, ...
- En pratique, trouver le bon assemblage pour correspondre aux besoins

La complexité humaine reste encore très difficile à faire pour une machine

Qu'est-ce que le Machine Learning ?

Un peu de structure :

- Le Machine Learning est une application de l'intelligence artificielle



Qu'est-ce que le Machine Learning ?

Où trouver du Machine Learning ?

- Faire des tâches cognitives
- Apprentissage automatique
- Apprentissage statistique
- Traduction de texte
- Résoudre des questions
- Chercher dans les données
- Classer des observations
- Algorithmes qui évoluent et affinent le résultat

Qu'est-ce que le Machine Learning ?

Les étapes du Machine Learning au cours du temps :

1. Développement des connaissances scientifiques
2. Champ d'étude de la science entre réception et perception
3. Reproduction de comportements humains avec des machines

Qu'est-ce que le Machine Learning ?

Les étapes du Machine Learning au cours du temps :

4. Phase d'apprentissage pour affiner le résultat

- Supervisé
- Non supervisé
- Par renforcement

5. Environnement de plus en plus complexe et changeant (data science)

Qu'est-ce que le Machine Learning ?

La première réalité du Machine Learning :

- Manipulation de données de petites ou moyennes quantités
- Collection d'algorithmes et modélisations pour :
 - Pouvoir apprendre de leur utilisation
 - Améliorer, enrichir une donnée
 - Affiner des recherches
 - Comprendre des phénomènes
 - Créer

Qu'est-ce que le Machine Learning ?

La première réalité du Machine Learning :

- Connaissance en statistique pour guider les modèles d'apprentissage
- Estimation et minimisation des erreurs par le calcul de coûts
- Prouver la véracité des résultats

Qu'est-ce que le Machine Learning ?

Algorithme du Machine Learning :

1. Apprendre les paramètres d'un modèle à partir d'exemple
2. Exemple = description numérique d'un objet associé à une catégorie ou une étiquette
3. L'algorithme choisit en fonction du modèle ajuste les paramètres du modèle :
 - En fonction d'un échantillon d'apprentissage
4. Mesure les performances du modèle :
 - En fonction d'un échantillon de test

Quels sont les enjeux du
Machine Learning ?

Quels sont les enjeux du Machine Learning ?

Pourquoi le Machine Learning devient important ?

Le monde du numérique change et évolue :

- Plus de concepteur en IT
- Plus d'utilisateur d'IT
- Plus de commercialisation
- Plus d'hétérogénéité des domaines et des services
- Augmentation des types de serveur

Quels sont les enjeux du Machine Learning ?

Pourquoi le Machine Learning devient important ?

Le monde du numérique change et évolue :

- Modification des utilisations
- Les objets de plus en plus connectés
- Stagnation des puissances de calculs
- Demande de rapidité
- Demande d'intégrité
- Demande de sécurité

Quels sont les enjeux du Machine Learning ?

Pourquoi le Machine Learning devient important ?

Les informations : le monde du traitement de la donnée change :

1. Prédiction
2. Analyse automatique
3. Le temps réel (avertissements)
4. En flux

Quels sont les enjeux du Machine Learning ?

Pourquoi le Machine Learning devient important ?

Quelles sont les données utilisées en Machine Learning :

1. Stockage ?
2. Volume ?
3. Type

Quels sont les enjeux du Machine Learning ?

Pourquoi le Machine Learning devient important ?

Le Big Data au service du Machine Learning : le monde des données change :

1. En volume
2. En origine
3. En structure
4. En terme de stockage

Quels sont les enjeux du Machine Learning ?

Le Big Data et ses données pour :

- Reconnaissance faciale
- Reconnaissance dans les images
- Reconnaissance audio
- Logs
- Graphes des réseaux sociaux
- Récupérations des flux des capteurs

Quels sont les enjeux du Machine Learning ?

Le Big Data pour la collecte des données :

- Batch :
 - Beaucoup de volume analyser par algorithme pour fournir un fichier de résultat
 - Traitement par lancement de processus et non interactif
- Périodique en batch :
 - Recalcul des données en incluant les nouvelles données arrivées pendant le traitement initial

Quels sont les enjeux du Machine Learning ?

Le Big Data et le traitement des données :

- Interactif :
 - Besoin de faire de l'analyse de données
- Temps Réel :
 - Lancer des algorithmes sur les données fournies pour travailler rapidement dessus

Quels sont les enjeux du Machine Learning ?

Le Machine Learning dans le monde la Big Data :

- Comment analyser correctement les données ?
 - Faible volume
 - Comparable
 - Identifiable
 - Vélocité
- Que permettent les algorithmes :
 - Observation
 - Prédition
 - Tendance
 - Corrélation de données -> Extraction quantitative

Quels sont les enjeux du Machine Learning ?

Le Machine Learning dans le monde la Big Data :

- Les difficultés :
 - Faire le choix du bon modèle à appliquer
 - Dimensionner les données d'apprentissage
 - Faire le bon choix des hyperparamètres
 - Comprendre les notions mathématiques des algorithmes pour estimer les demandes et les limites

Quels sont les enjeux du Machine Learning ?

Le Machine Learning dans le monde la Big Data :

- La pratique :
 - Choisir les algorithmes
 - Instancier les classes et les hyperparamètres
 - Fournir les données d'apprentissage
 - Déterminer les données avec des méthodes de prédictions

Quels sont les enjeux du Machine Learning ?

Un modèle de Machine Learning est :

- Complexe
- Difficile d'en expliquer le fonctionnement

Dans les secteurs spécialisés (exigences de conformité) nécessité :

- Utiliser des modèles de Machine Learning simple
- Permettre aux entreprises d'expliquer comment chaque décision est décidée
- Savoir définir la teneur du résultat

Quels sont les enjeux du Machine Learning ?

Les limites aux Machines Learning :

- La connaissance du cerveau humain est loin d'être connu, et des questions de posent encore :
 - La mémoire ?
 - La connaissance ?
 - Les réflexes ?
 - La réflexion ?
 - Quel point un cerveau utilise-t-il pour reconnaître ?
 - Quel sens sont mis en place pour avoir des émotions ?
 - Comment faisons-nous réellement pour comprendre ?

Quels sont les enjeux du Machine Learning ?

Où le Machine Learning suscite un intérêt actuel ?

- Réussite au niveau des jeux
- Dans les outils du quotidien
- Conscience des grandes collectes des données actuelles

Quels sont les enjeux du Machine Learning ?

Les problèmes du Machine Learning est l'apprentissage hasardeux lié aux données initiales (sur et sous-ajustement):

- Le modèle est trop adapté aux premières données lors de la modélisation
- Le modèle ne peut plus être généralisé aux nouvelles données, et/ou à leur variance

Pour limiter les succès trop actifs :

- Pedro Domingo (Professeur à l'Université de Washington) indique l'importance de garder certaines données séparées lors du test des modèles, et d'utiliser seulement ces données réservées pour tester un modèle choisi, suivi par un apprentissage sur l'ensemble de données



Quels sont les enjeux du Machine Learning ?

Si votre Machine Learning ne fonctionne pas, il faut :

- Plus de données
- Plus de classifications
- Plus d'observations
- Changer les paramètres
- Peut-être changer de modèle

En termes d'objectif, l'apprentissage machine n'est pas une fin ou une solution en soi, il faut avoir une stratégie pour valider les résultats

Quels sont les enjeux du Machine Learning ?

Les Machines qui apprennent sont utiles aux humains :

- Puissance de traitement capable de mettre en évidence ou de trouver plus rapidement des modèles dans de grandes (ou d'autres) données
- Améliorer la capacité des humains à résoudre des problèmes
- Déduire un large éventail de problèmes comme l'aide au diagnostic des maladies, la recherche de solutions au changement climatique mondial.

Comment faire du Machine Learning ?

Comment faire du Machine Learning ?

1. Obtenir les données

- Base de données
- Flux
- Data Set
- Génération des data Frame (tableaux)

2. Analyser les données

- Structurer
- Organiser
- Nettoyer
- Comprendre

Comment faire du Machine Learning ?

3. Comprendre la demande

- Analyse
- Interprétation
- Reconnaissance
- Reconstruction
- Création

4. Comment fonctionne le système

- Description du domaine
- But de l'apprentissage
- Est-ce bien du Machine Learning ?

Comment faire du Machine Learning ?

5. Automatiser les traitements

- Minimisation
- Gestion
- Vélocité
- Véracité

6. Automatiser les méthodes de visualisation

- Représentation
- Lecture
- Visibilité

Comment faire du Machine Learning ?

Approche de l'apprentissage automatique :

- Collection de beaucoup de données pour qu'un algorithme détermine automatique le bon comportement
- Avec les données, le modèle classe les données en plusieurs catégories pour permettre l'identification :
 1. Modèle avec un raisonnement analogue à celui de l'humain
 2. Modèle sans être assimilé à un comportement couramment utilisé par l'humain

Le but est de donner aucun critère sur les données pour que le système s'adapte aux mutations

Comment faire du Machine Learning ?

Possibilité de faire de l'apprentissage par l'exemple :

- Relation de liens et d'associations pour prendre une décision

Les ordinateurs ont l'avantage d'avoir une grande capacité :

- De mémorisation
- De rapidité du traitement de données



La définition de Tom Mitchell :

- « Une machine apprend si ses performances à réaliser une tâche donnée mesurée par un score calculable s'améliore avec l'expérience en fonction du nombre d'exemple considéré »

Comment faire du Machine Learning ?

La vraie problématique est le choix du modèle de Machine Learning approprié

La résolution d'un problème peut prendre du temps si l'approche stratégique n'est pas adoptée

Étape 1 : L'expertise :

- Connaitre le domaine d'application et les résultats souhaités
- Aligner le problème avec les entrées de données potentielles qui doivent être envisagées pour la solution

Comment faire du Machine Learning ?

Étape 2 : Les données :

1. Collecter
2. Nettoyer (suppression des valeurs manquantes, aberrantes et des doublons),
3. Formater (réduire l'ensemble des données aux classes utiles pour l'apprentissage, ou ajouter des classes)

Étape 3 : Les paramètres :

- Observer (avec un outil) les attributs (features) communs aux données et les liens entre les étiquettes (labels) du format (colonnes d'une table)
- Etablir les dépendances entre les attributs et les étiquettes
- Ces facteurs influencent les résultats de l'algorithme

Comment faire du Machine Learning ?

Étape 4 : L'algorithme

- Les données d'entraînement servent à ajuster les modèles
- Utile pour tester pour comparer et tester les performances
- Les données de validation pour estimer les erreurs de prédiction
- Ensemble de tests pour évaluer les performances du modèle sur des données inconnues du modèle



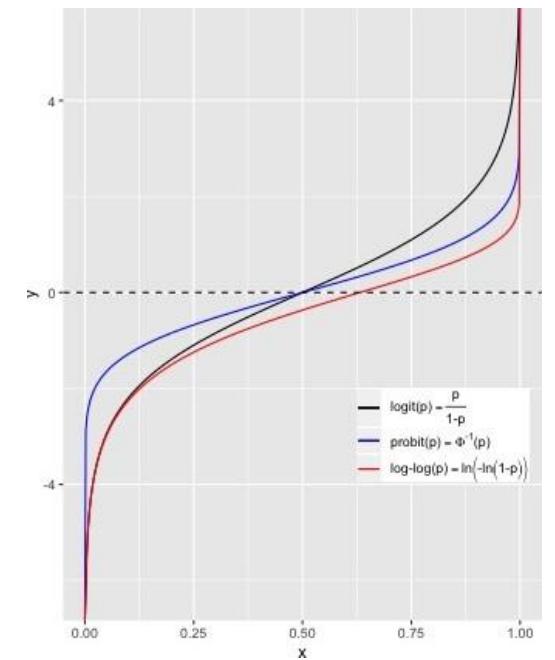
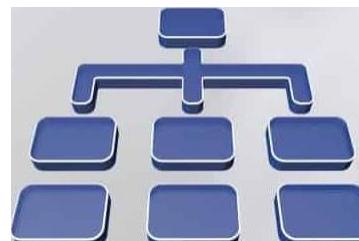
Étape 5 : Le résultat :

- Bien connaître le problème
- Affiner les résultats pour obtenir un niveau de précision acceptable

Comment faire du Machine Learning ?

Plusieurs approches pour amener les Machines à apprendre :

- Régression (linaire, logistique, ...)
- Arbres de décision
- Grappes d'arbres (forêt)
- Couches de réseaux neuronaux (ou maintenant Deep Learning)
- Les diagnostics



Quels sont les langages et
outils pour faire du
Machine Learning ?

Quels sont les langages et outils pour faire du Machine Learning ?

Plateformes Cloud (MLaaS) : Azure ML, Watson, Google, Amazon

- Dédiées à une typologie de données, les algorithmes de Machine Learning utilisent des applications web déployées sur le Cloud

Environnement : Dataiku, OpenML

- Utilisées comme des services (MLaaS) ou déployées dans un environnement

Quels sont les langages et outils pour faire du Machine Learning ?

Python :

- Les modules Scientifiques (numpy, matplotlib, Scipy)
- Les module d'analyses de données et de visualisation (pandas, seaborn)
- Les modules de Machine Learning et de DataSet (scikit-Learn, pickles, hdfs)

R

C/C++

Java

JavaScript

Qui fait du Machine Learning ?

Le Data Scientist englobe le Machine Learning

La data science :

- Un ensemble de technologies (hétérogène)
- Analyser des données
- Découvrir les bonnes approches
- Obtenir des informations, des résultats, des agrégats
- Appliquer des fonctions sur des masses de données
- Répondre à des questions
- Bâtir des modèles
- Définir du traitement dans un langage de programmation
- Mise en place d'algorithmes de Machine Learning

Qui fait du Machine Learning ?

Domaines du Machine Learning :

- Information
- Finance
- Sécurité
- Médecine (Classification des séquences d'ADN, Intégration de Watson au centre de cancérologie Mémorial Sloan Kettering de New York = IA qui s'appuie sur une grande base de données pour établir des diagnostics médicaux poussés,)
- Industrie automobile (les voitures à conduite autonome)
- Technologie dans tout son ensemble (reconnaisances vocale, faciale, objets, caractères, ...)

Qui fait du Machine Learning ?

Le machine Learning est utilisé dans de nombreuses applications, par exemple :

- Le moteur de recommandations d'un fil d'actualité comme Facebook (le Machine Learning personnalise le flux des abonnés, en utilisant la fréquence des publications) affiche un plus grand nombre d'activités dès le début du fil d'actualité
- La gestion de la relation client comme [GRC](#) (Gestion de la Gouvernance, des risques et de la Conformité) qui utilise des modèles pour analyser les e-mails et inviter les commerciaux à répondre en priorité aux messages les plus importants

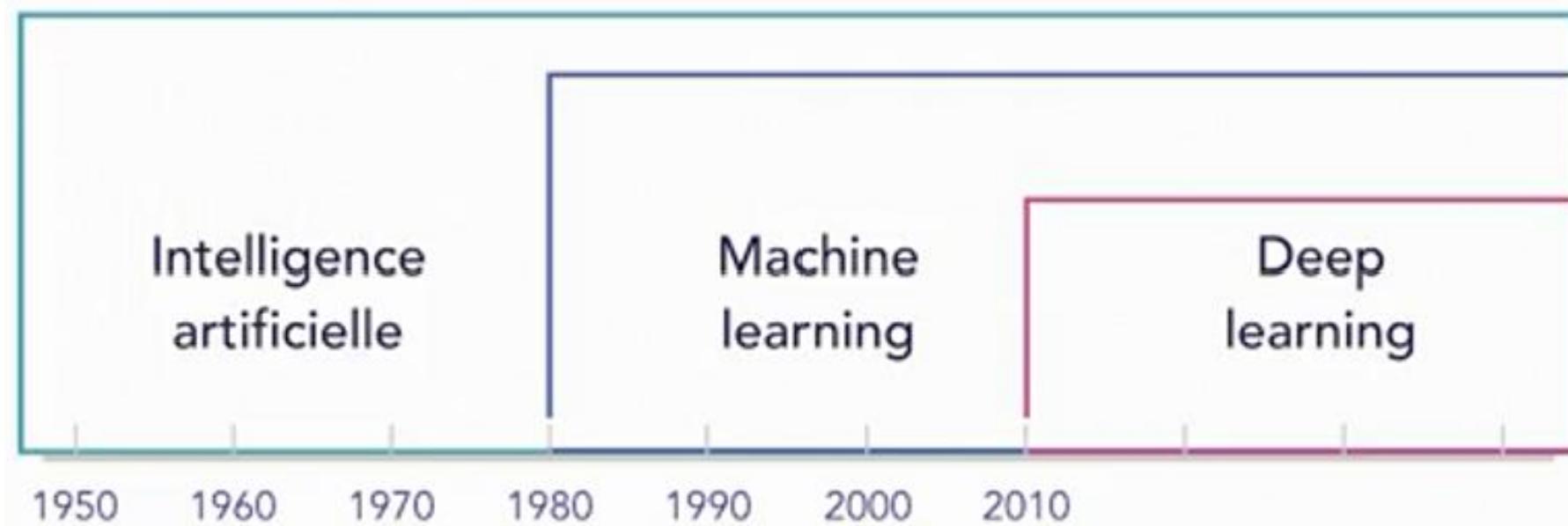
Qui fait du Machine Learning ?

Le machine Learning est utilisé dans de nombreuses applications, par exemple :

- L'informatique décisionnelle comme les solutions en [BI](#) (Business Intelligence) pour identifier les points de données et anomalies
- Les systèmes des ressources humaines comme [GRH](#) (Gestion Ressource Humaine) pour étudier des candidatures et identifier les meilleurs candidats
- Des assistants virtuels qui utilisent des modèles supervisés et/ou non supervisés pour interpréter des langages naturels, des contextes,

Ce qu'il faut retenir du Machine Learning

Ce qu'il faut retenir du Machine Learning



Ce qu'il faut retenir du Machine Learning



Un ordinateur fait des calculs



L'humain résout des problèmes, sait conduire une voiture, reconnaît des objets par le touché et/ou par la vue, fait la cuisine,



L'intelligence artificielle tente par des techniques de créer des systèmes pour simuler l'humain

Ce qu'il faut retenir du Machine Learning

Le machine Learning consiste à écrire un programme pour :

- Apprendre à faire une Tâche avec une stratégie de Performance en s'améliorant avec de l'Expérience

Le Machine Learning utilise des données et des algorithmes :

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Ce qu'il faut retenir du Machine Learning

Le principe du Machine Learning est :

- Apprendre à la machine comment résoudre des problèmes

Le but du reste du cours est de présenter les techniques permettant à la machine d'apprendre à résoudre des problèmes avec :

- Les statistiques et probabilités
- Les modèles prédictifs
- Les fonctions de minimisation des coûts et des erreurs

Ce qu'il faut retenir du Machine Learning

Les modèles de l'IA sont normés par des ISO (International Standard Organization)

ISO/IEC 2382-28:1995 : Intelligence Artificielle, notions fondamentales et systèmes experts, révisée par ISO/IEC 2382:2015

ISO/IEC 2382-31:1997 : Intelligence Artificielle et Machine Learning, révisée par ISO/IEC 2382:2015

ISO/IEC 2382-34:1999 : Intelligence Artificielle et Réseaux neuronaux, révisée par ISO/IEC 2382:2015

Les modèles mathématiques

Rappels : Statistiques & Probabilités

Dr. Chaabani Yasmine
yasmin.chaabani@isgb.ucar.tn

Le but du module ?

Apporter des rappels en mathématiques et statistiques applicables aux problèmes de Machine Learning

Sommaire

Statistiques

Probabilités

Rappels sur les matrices

L'interprétation des données

Modèles statistiques

Statistiques et structures

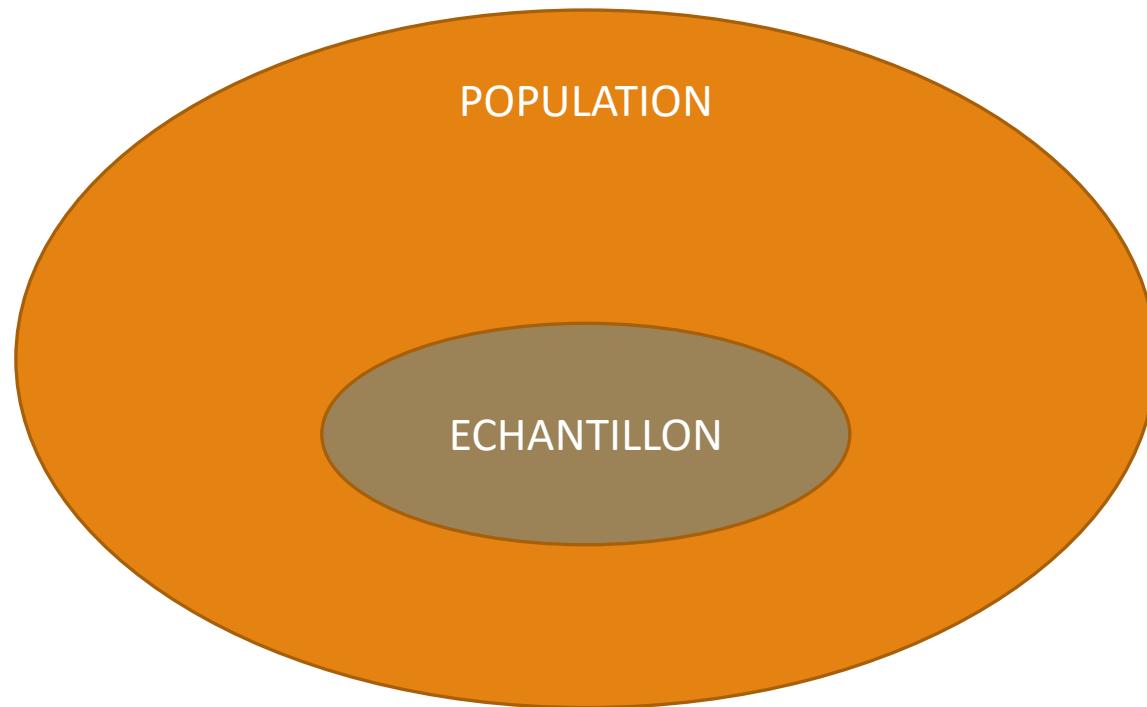
Statistiques

Dans nos programmes de Machine Learning, les mathématiques offrent une abstraction utile et nécessaire pour obtenir les meilleurs modèles et les meilleurs résultats

Nos statistiques vont produire, à partir des données, les tendances à suivre pour prédire les futurs valeurs des variables

En fonction des tendances, les prédictions seront évaluées pour rendre acceptable ou non les apprentissages

Statistiques



Statistiques

La variable indépendante (variables explicatives) :

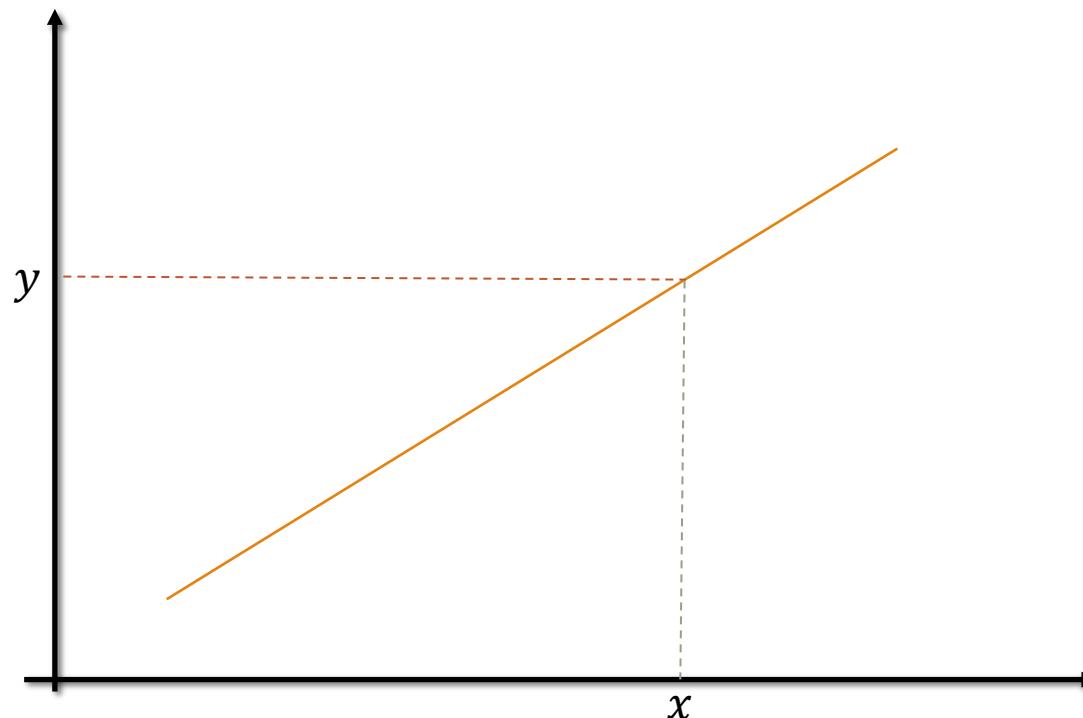
- Manipulée par l'expérimentateur
- Utiliser dans le but d'expliquer, de décrire ou de prédire la ou les variable(s) dépendante(s)
- Souvent représentée sur l'axe des abscisses dans les graphiques de modélisation

La variable dépendante (variable à expliquer) :

- Variable qui subit l'effet de la variable indépendante
- Que nous cherchons à décrire, expliquer, prédire
- Souvent représentée sur l'axe des ordonnées dans les graphiques de modélisation

Statistiques

Exemple : $y = ax + b$



Statistiques

Une analyse statistique qui ne fait appel qu'à une variable dépendante est dite univariée

Une analyse statistique qui porte sur plusieurs variables dépendantes est dite multivariée

Statistiques

L'inférence statistique est :

- Ensemble de techniques pour induire les caractéristiques d'un groupe général (population) à partir de celles d'un groupe particulier (échantillon), en fournissant une mesure de la certitude de la prédiction : la probabilité d'erreur

L'inférence statistique consiste :

- A induire les caractéristiques inconnues d'une population à partir d'un échantillon issu de cette population. Les caractéristiques de l'échantillon, une fois connues, reflètent avec une certaine marge d'erreur possible celles de la population

Statistiques

Les méthodes statistiques ne servent pas uniquement à valider ou non des modèles préétablis, c'est aussi des méthodologies pour extraire des connaissances à partir de données

Les statiques sont une approche essentielle pour la prise de décision

Les statistiques sont un mode de réflexion pour maîtriser la complexité, l'aléatoire et les risques, en apportant la prudence scientifique nécessaire pour limiter les erreurs

Statistiques

Variable qualitative :

- Exprime une qualité, un état, une condition, un statut unique et exclusif comme : La couleur, la catégorie socioprofessionnelle, le sexe, ...

Variable quantitative :

- Exprime une référence à une unité de mesure reconnue, qualifiée aussi de variable métrique comme : La taille, le poids, la surface, la distance, le revenu, l'âge, les prix ...

Statistiques

La variable qualitative peut avoir une forme « pseudo quantitative » à la place de sa forme « nominale » :

- Utilisation des variables qualitatives dans un traitement multivarié
- Rendre manipulable et compatible avec un logiciel statistique

Exemple ordinaire

- 3 = Bon
- 2 = Moyen
- 1 = Mauvais

Statistiques

Exemple nominal

- 1 = Mort
- 2 = Vivant

A noter : Les valeurs numériques forment une référence, des numéros. Elles ne peuvent pas intervenir dans des calculs (somme, moyenne, ...)

Statistiques

Nous avons 2 familles de distribution :

- Continues (monde réel, analogique, échantillon nécessaire car infini)
- Discrètes (monde numérique, ensemble fini ou infini par fonction)

• Caractère statistique quantitatif continu :

- Caractère statistique qui peut prendre toutes les valeurs contenues dans un intervalle réel donné

• Caractère statistique quantitatif discret :

- Caractère statistique qui peut prendre un nombre fini de valeurs dans un intervalle donné

Statistiques

- Caractère statistique qualitatif :

- Caractère non associer à un ensemble numérique discret ou continu (langue, préférence, secteur d'activité, couleur, ...)

Un processus stochastique ou processus aléatoire ou fonction aléatoire représente une évolution, discrète ou à temps continu, d'une variable aléatoire

Le calcul stochastique est l'étude des phénomènes aléatoires dépendant du temps

2 distributions des valeurs : expérimentale et théorique

Statistiques

Encodage ordinal des étiquettes (Label Encoding) avec récupération qualitatif de la donnée

- Transformation des informations littéraux en numérique
- Exemple : Résultat à un examen :

Mention à un examen
Très Bien
Bien
Passable
Insuffisant

Mention à un examen
3
2
1
0

Statistiques

Encodage nominal des étiquettes avec récupération qualitatif de la donnée

- Transformation des informations littéraux en numérique
- Exemple : Couleur en RGB:

Couleur
Bleu
Rouge
Vert

Rouge	Vert	Bleu
0	0	1
1	0	0
0	1	0

Statistiques

En règle général un algorithme fonctionne dont les données ont des valeurs

- Les valeurs manquantes peuvent devenir problématiques et donc il faut :
 - Soit abandonner les observations correspondantes
 - Soit réaliser des imputations :
 - Moyenne
 - Médiane
 - Ecart
 - Estimation (régression, k-Nearest Neighbour (KNN), etc.)

Statistiques

La génération des données se fait avec des distributions statistiques inférentielles normales. Ce sont des modèles avec des paramètres comme la moyenne et la variance.

Si X est une distribution : (x_1, x_2, \dots, x_N) , la Moyenne (notée \bar{x}) est :

- $$\bar{x} = \frac{x_1 + x_2 + \dots + x_n}{N}$$

Dans le cas d'une population entière, la Variance (notée σ^2) est:

- $$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2$$

Dans le cas d'un échantillon, la Variance est :

- $$s^2 = \frac{\sum (x_i - \bar{x})^2}{N-1}$$

Statistiques

L'analyse de la variance (ANOVA : ANalysis Of VAriance) :

- Modèle statistique pour comparer les moyennes d'échantillons selon les modalités d'une variable qualitative
- Les moyennes sont calculées avec une variable continue

Statistiques

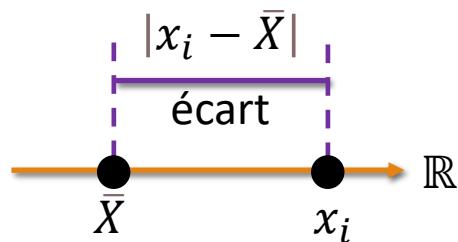
Dans le cas d'une population entière, l'écart type (noté σ) est :

- $\sigma = \sqrt{\nu(x)} = \sqrt{\bar{x}^2 - (\bar{x})^2}$

Dans le cas d'un échantillon de la distribution, l'écart type est :

- $s = \sqrt{\frac{\sum(x_i - \bar{x})^2}{N-1}}$

Ecart à la moyenne :



Statistiques

Covariance ($cov(x:y)$) :

- Fonction à 2 variables pour quantifier les écarts conjoints par rapport aux moyennes respectives
- $cov(x:y) = \overline{xy} - \bar{y}\bar{x}$

Coefficient de corrélation (coef) pour évaluer le coût de justesse linaire :

- $\text{coef} = \frac{cov(x:y)}{\sigma(x) \cdot \sigma(y)}$

Statistiques

Pour ne pas biaiser les mesures, il faut parfois normaliser les variables quantitatives :

- Standard ou min-max :

- $\tilde{y} = \frac{y - \min(y)}{\max(y) - \min(y)}$

- Centrée-réduite :

- $\tilde{y} = \frac{y - \bar{y}}{\sigma(x)}$

Statistiques

Calcul pour obtenir la fréquence d'apparition de x dans un échantillon :

- $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-1}{2\sigma^2}(x - \bar{x})^2\right)$

OU

- $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(\frac{-1}{2\sigma^2}(x - \bar{x})^2\right)}$

Statistiques

Le biais est le terme pour qualifier des erreurs lors de l'évaluation pour la qualité d'un algorithme d'apprentissage

Les algorithmes d'apprentissage doivent prédire au plus juste la relation exacte entre des données d'entrée et des données de sortie

- Le modèle utilisé par l'algorithme est plus simple que le problème qu'il doit apprendre
- L'erreur faite dans les hypothèses du modèle est nommée le « biais »

Statistiques

Le biais d'ancrage :

- Influence sur la première vérité lue ou entendue, même en cas de contradiction

Le biais de données heuristique ou de disponibilité :

- Le contre exemple à la règle (fumer tue, mais il y a un doute par l'exemple)

Le biais de suivi :

- Manque de contradiction

Le biais d'aveuglement :

- Le refus qu'il existe des biais

Le biais de support :

Statistiques

Le biais de sélection :

- L'échantillon statistique n'est pas représentatif de la population

Le biais de mesure :

- Caractérise l'incertitude de la mesure

Le biais de confusion :

- Désigne un ensemble d'erreurs qui peuvent survenir dans l'interprétation des liens entre la variable dépendante et la variable indépendante lors de l'analyse de résultats expérimentaux du fait de l'interférence d'autres variables qui ont été insuffisamment contrôlées par le protocole de recherche

Statistiques

Le biais de confirmation:

- Biais cognitif qui consiste à privilégier les informations confirmant ses idées préconçues ou ses hypothèses et/ou à accorder moins de poids aux hypothèses et informations jouant en défaveur de ses conceptions

Le biais d'évaluation :

- Lorsque la mesure du critère de jugement n'est pas réalisée de la même manière dans les deux groupes de patients

Le biais inductif (ou biais d'apprentissage) :

- Ensemble des hypothèses que l'apprenant utilise pour prédire les sorties d'entrées données qu'il n'a pas rencontrées

Statistiques

Le biais de confort ou d'expert :

- Aucune influence des autres

Le biais d'innovation :

- Oubli des anciens modèles

Le biais de nouveauté :

- Temporel, relatif à l'information, les dernières valeurs « court terme » priment devant les « long terme »

Le biais de projection :

- Considérer des événements faibles avec une trop grande probabilité

Statistiques

Le biais de perception :

- Juger les informations, plutôt que la réalité (mirage)

Le biais de stéréotype :

- Caricature sur les données (déformations involontaires ou volontaires)

Le biais de survivant :

- Facile de le faire car vous l'avez réussi

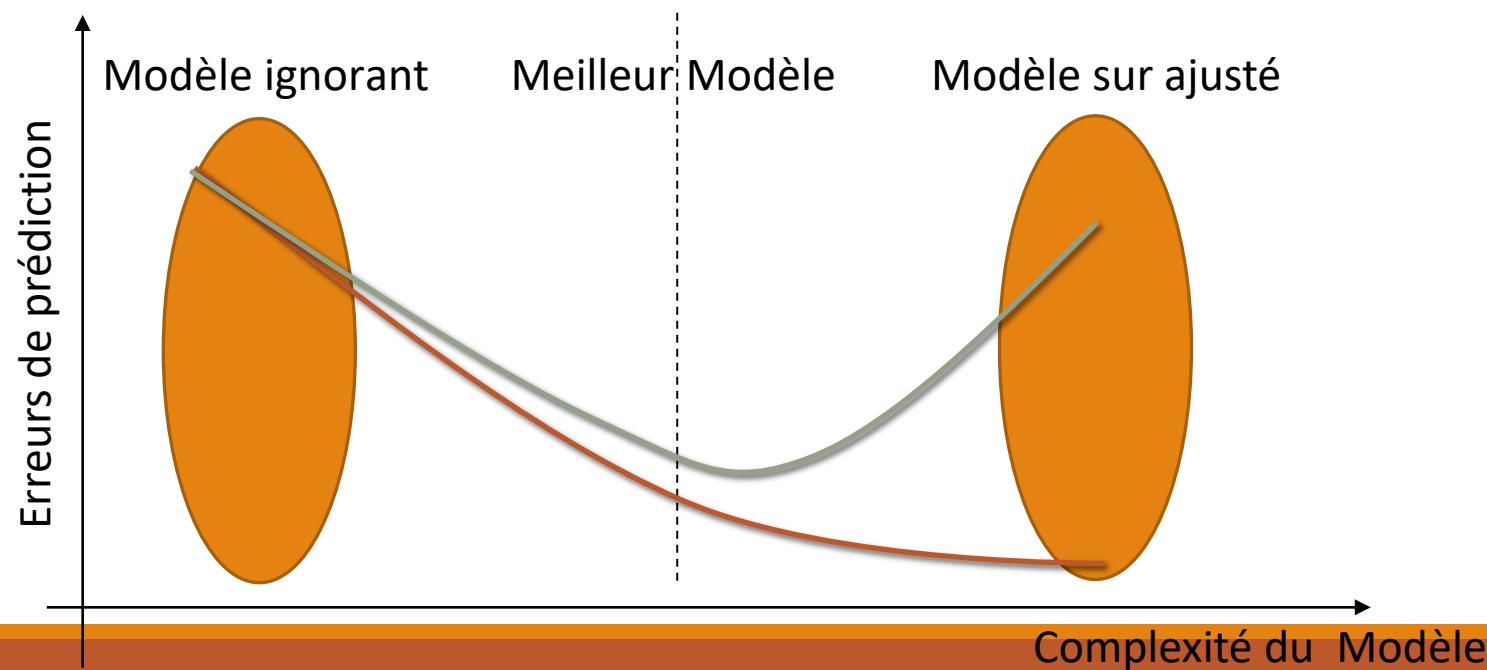
Le biais du zéro risque :

- Elimination des données qui présentent des risques

Statistiques

Erreur empirique = erreur sur l'échantillon d'apprentissage x

Erreur de généralisation : erreur sur toutes les données (inconnue)



Statistiques

Rappel des dérivées :

- $\partial \text{Constante} = 0$

- $\partial x = 1$

- $\partial x^n = nx^{n-1}$

- $\frac{1}{x} = -\frac{1}{x^2}$

- $\sqrt{x} = \frac{1}{2\sqrt{x}}$

Statistiques

Divergence d'un vecteur \vec{u} en coordonnées cartésiennes :

- $div(\vec{u}) = \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}$
- $div(\vec{u}) = \vec{\nabla} \cdot \vec{u}$

Opérateur nabla $\vec{\nabla}$



The diagram shows a vertical vector symbol with four horizontal components. The top component is labeled $\frac{\partial}{\partial x}$, the second $\frac{\partial}{\partial y}$, the third $\frac{\partial}{\partial z}$, and the bottom component is a vertical line.

Statistiques

Le gradient est un vecteur qui prend en argument un scalaire f (contrairement à la divergence) : c'est l'inverse de la divergence :

- $\overrightarrow{\text{grad}}(f) = \vec{\nabla} \cdot f$

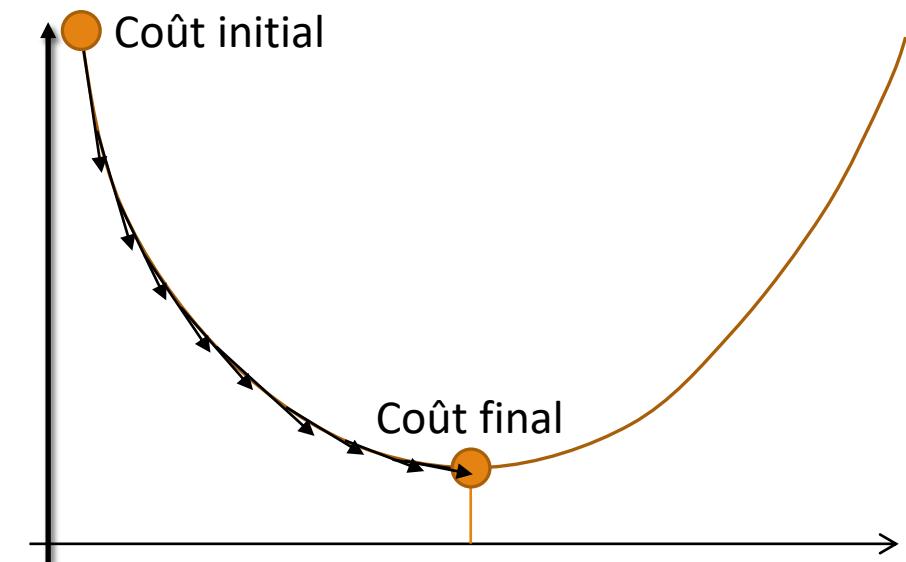
Le gradient d'une fonction de plusieurs variables en un certain point est un vecteur qui caractérise la variabilité de cette fonction au voisinage de ce point

Le gradient est la généralisation à plusieurs variables de la dérivée d'une fonction d'une seule variable

Statistiques

Descente de gradient :

- Algorithme d'optimisation pour trouver le minimum de n'importe quelle fonction convexe en convergeant progressivement vers ce point minimum
- Appliquer une succession de gradients pour obtenir le point minimum (maximum)



Statistiques

Descente de gradient :

- Dans les problèmes d'apprentissage supervisé pour minimiser la fonction coût (par exemple l'erreur quadratique moyenne)

Pour trouver le meilleur modèle :

- Minimiser la fonction coût revient à trouver les paramètres de modèles qui donnent les plus petites erreurs entre notre modèle et les points y du Data set
- Une fois la fonction coût minimisée les prédictions deviennent acceptables

Statistiques

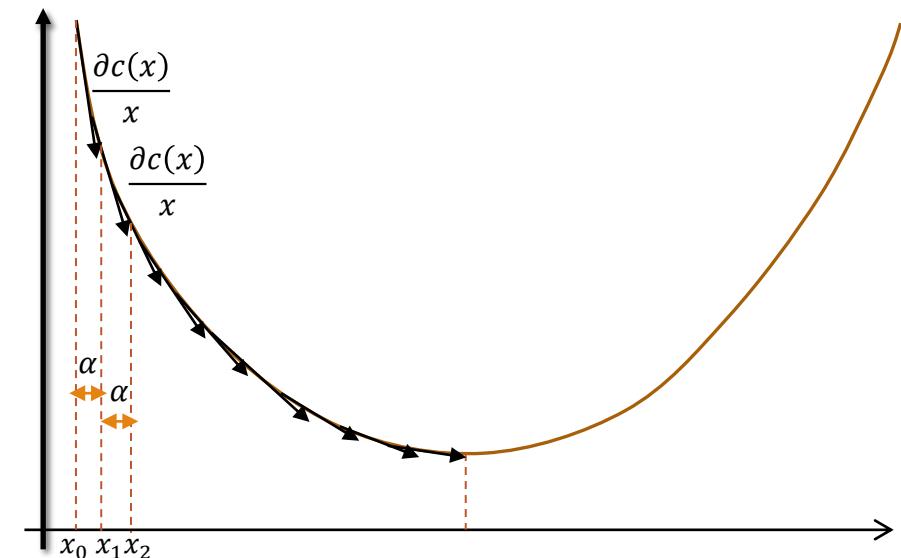
Principe :

- Choix d'un point initial aléatoire
- Mesure de la valeur de la pente en ce point = dérivée
- Progression d'une distance α dans la direction de la pente = nouveau point

Statistiques

Exemple :

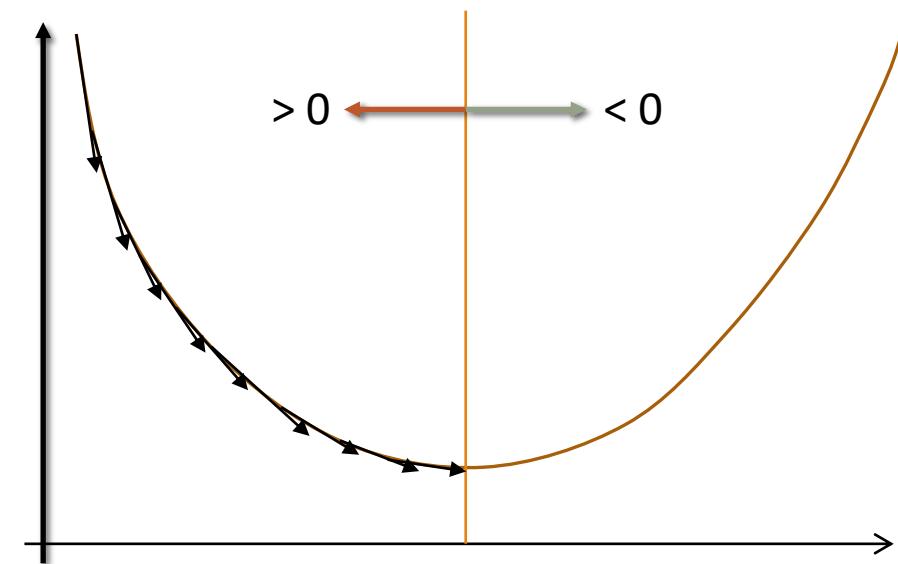
- Soit x, y les paramètres à définir
- Soit C la fonction de coût : $C(x, y)$
- x converge vers le minimum de $C(x, y)$
- $\frac{\partial c(x)}{x}$ donne la direction de la pente qui descend
- α est un hyperparamètre



Statistiques

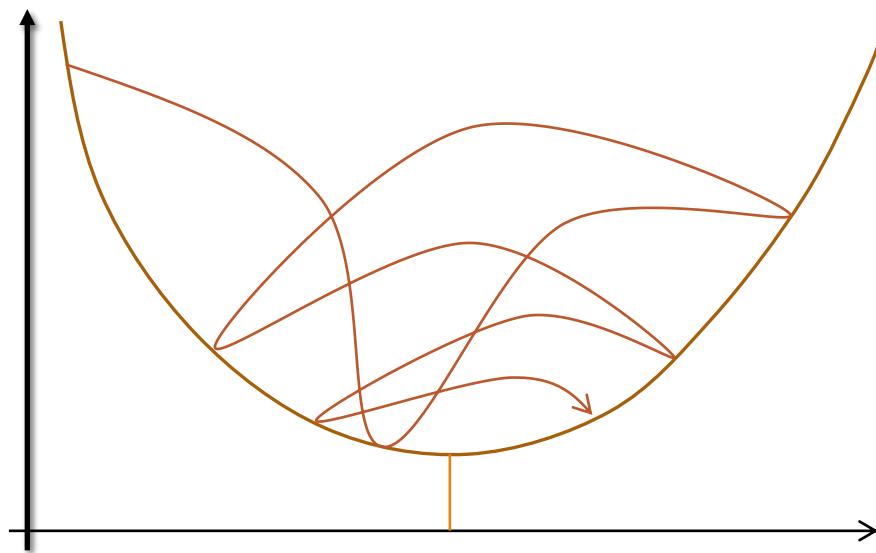
Répéter

- $x_{i+1} = x_i - \alpha \frac{\partial C(x_i)}{\partial x}$

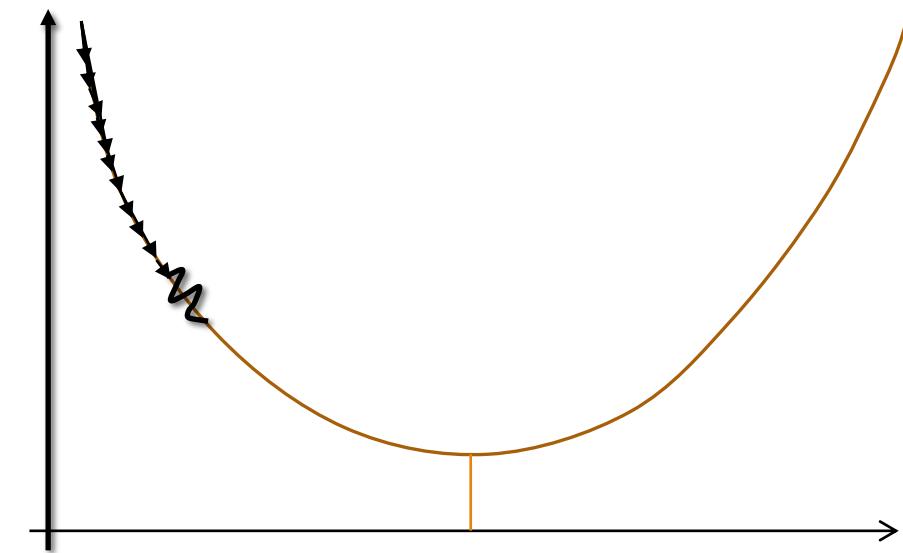


Statistiques

Choix de α trop grand ?



Choix de α trop petit ?



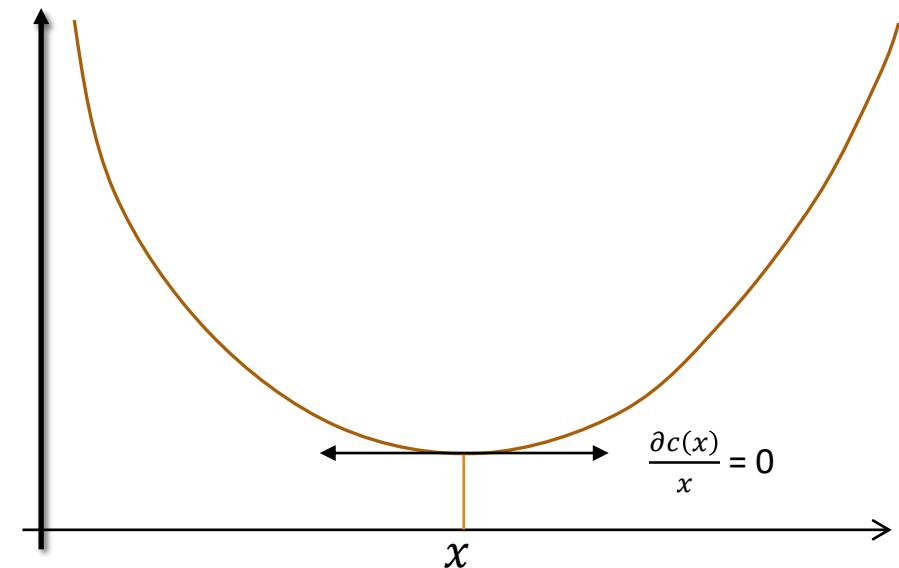
α peut-être choisi avec les valeurs empiriques

DR. YASMINE CHABANI

Statistiques

Les moindres carrés :

- Tangente horizontale au point minimum
- Cette fonction impose des inversions de matrices



Probabilités

Une probabilité est la vraisemblance qu'un évènement se réalise parmi un échantillon

Les probabilités forment le cadre des systèmes complexes pour connaître exactement les aboutissants, en acceptant une part de hasard

Probabilité exactes

Probabilité empirique

Probabilité intuitive

Probabilités

2 tendances sont mises en avant :

- L'interprétation fréquentielle/distribution :
 - Fréquence des occurrences après un nombre infini d'évènements
- L'interprétation bayésienne :
 - Quantifier l'incertitude des phénomènes
 - L'augmentation du nombre d'informations diminue l'incertitude du résultat

Probabilités

Expression d'une probabilité conditionnelle d'un événement A sachant que l'événement B s'est produit est :

- $P_B(A) = \frac{P(A \cap B)}{P(B)}$

Un événement dont la réalisation ne dépend pas du résultat d'un autre événement est peut s'appeler : évènement simple

Un espace probabilisé est un ensemble d'événement d'une expérience aléatoire dans laquelle nous avons attribué une probabilité à chaque événement

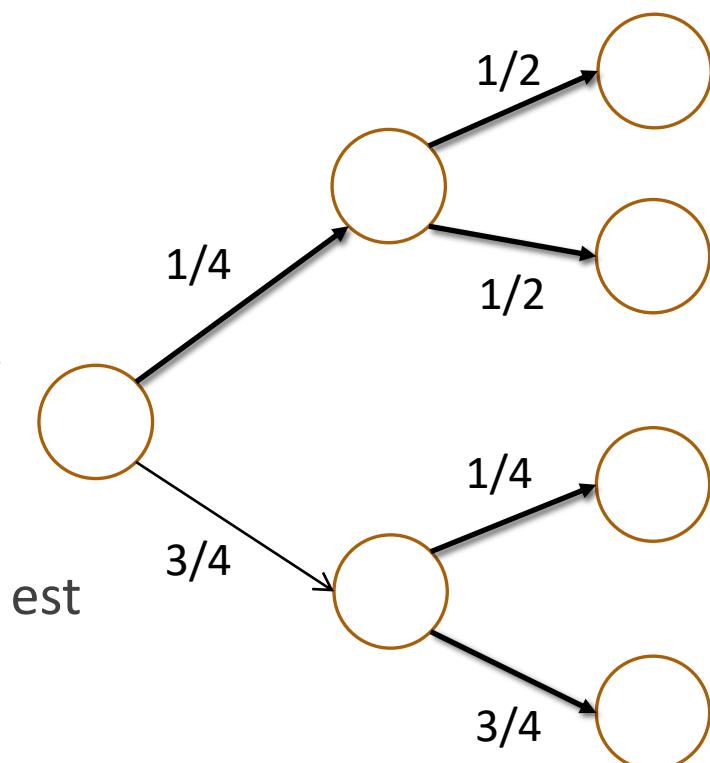
Probabilités

Arbre de probabilité (Orienté - Valué):

1 = La somme des probabilités supportées par les arrêtes sortantes d'un même sommet

La probabilité d'un chemin = le produit des probabilités des arrêtes

La probabilité supportée par l'arrêt entre un sommet 1 et 2 est la probabilité conditionnelle de l'événement du sommet 2 sachant que l'événement su sommet 1 est déjà réalisé



Probabilités

Les lois normales sont des lois de probabilité pour modéliser des phénomènes naturels issus de plusieurs événements aléatoires

Elles sont aussi appelées lois gaussiennes, lois de Gauss ou lois de Laplace-Gauss

Une loi normale est une loi de probabilité absolument continue qui dépend de deux paramètres :

- Son espérance = un nombre réel noté μ
- Son écart type = un nombre réel positif noté σ

Probabilités

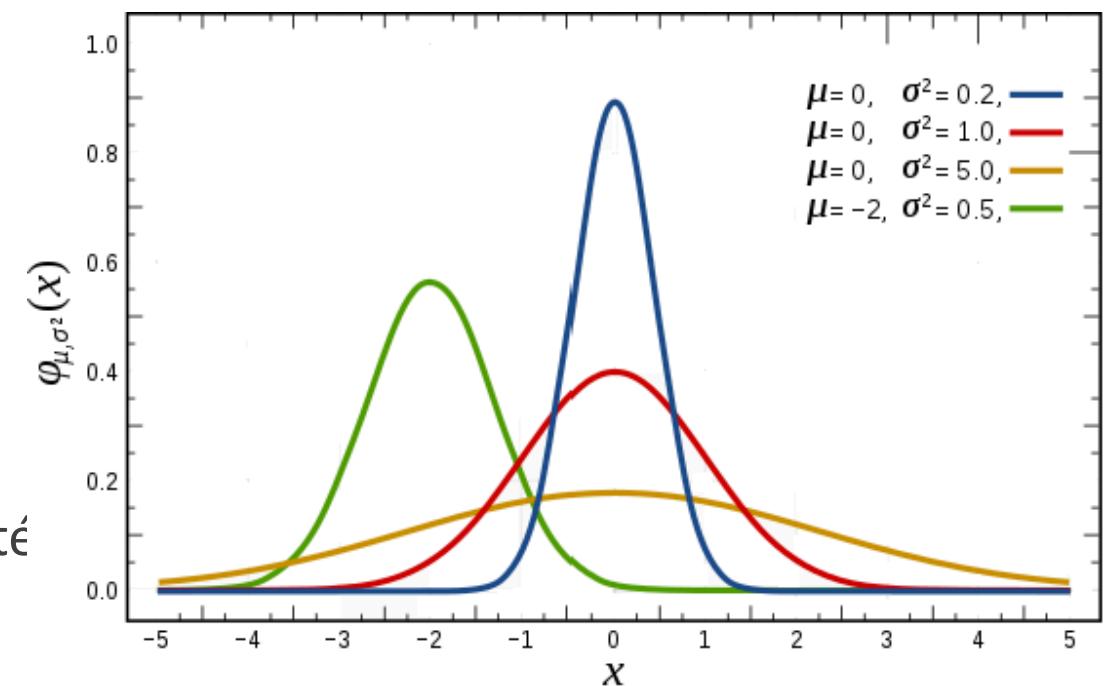
La densité de probabilité de la loi normale d'espérance μ , et d'écart type σ :

- $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

Les représentations graphiques des densités sont les courbes de Gauss ou en cloche

La loi normale centrée réduite est la loi de probabilité absolument continue avec la densité de probabilité : $\varphi : \mathbb{R} \rightarrow \mathbb{R}^+$:

- $\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$



Probabilités

La distribution de Bernoulli ou loi de Bernoulli est une distribution discrète de probabilité :

- $\mathbb{P}(X = x) = \begin{cases} p & \text{si } x = 1 \\ 1 - p & \text{si } x = 0 \\ 0 & \text{sinon} \end{cases}$

Ou

- $\mathbb{P}(X = x) = p^x(1 - p)^{1-x}, x \in \{0,1\}$



JAC. BERNOULLI, MATH.PP.

L'espérance mathématique d'une variable aléatoire de Bernoulli vaut p et la variance vaut $p(1 - p)$

Probabilités

La loi binomiale modélise la fréquence du nombre de succès obtenus lors de la répétition de plusieurs expériences aléatoires identiques et indépendantes

La représentation est un arbre de probabilité à chaque génération de l'arbre, deux branches partent de chaque nœud, une pour le succès et une pour l'échec

Pour chaque expérience appelée épreuve de Bernoulli, il y a :

- Une variable aléatoire qui prend la valeur 1 lors d'un succès et 0 sinon
- La variable aléatoire, somme de toutes ces variables aléatoires, compte le nombre de succès et suit une loi binomiale

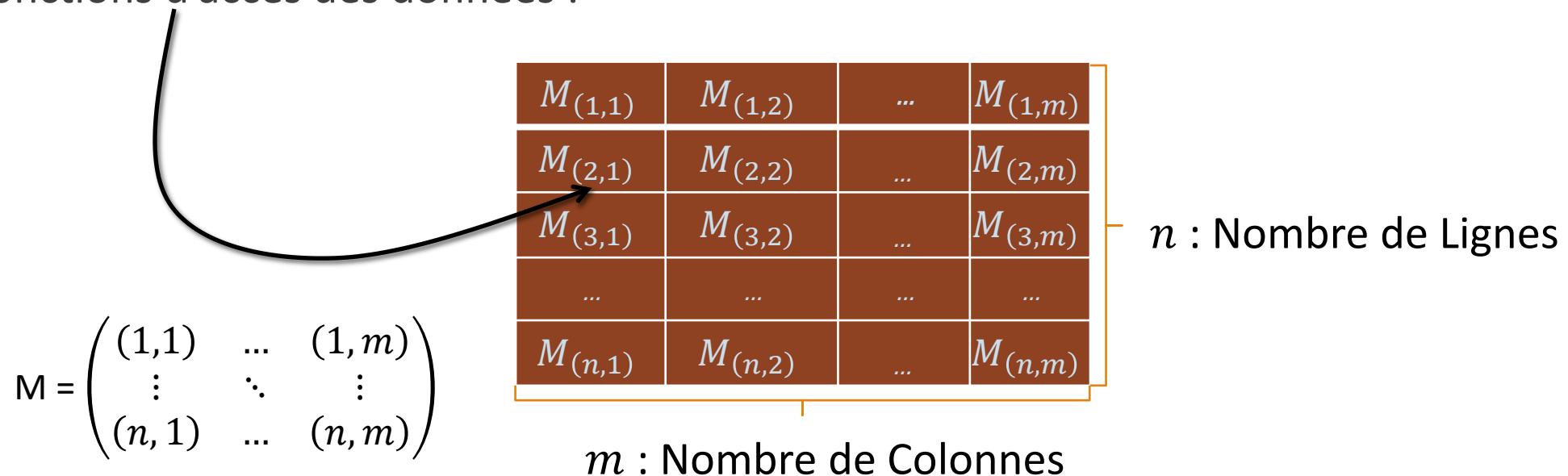
Probabilités

$\forall n$ et $\forall k \leq n$, donnent le nombre de parties de k éléments dans un ensemble de n éléments, il est possible d'obtenir la probabilité de k succès dans une répétition de n expériences :

- $\mathbb{P}(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$
- Avec le coefficient binomial = $\binom{n}{k} = C_n^k = \frac{n!}{k!(n-k)!}$

Rappels sur les matrices

Une matrice est une abstraction mathématique pour représenter des données dotée des fonctions d'accès des données :



Rappels sur les matrices

Addition de $M_{(n,m)} + M_{(n,m)} = M_{(n,m)}$:

$$\begin{pmatrix} 1 & 21 \\ 1 & 3 \\ 9 & 16 \end{pmatrix} + \begin{pmatrix} -6 & 11 \\ 7 & 24 \\ 0 & 77 \end{pmatrix} = \begin{pmatrix} 1 + (-6) & 21 + 11 \\ 1 + 7 & 3 + 24 \\ 9 + 0 & 16 + 77 \end{pmatrix} = \begin{pmatrix} -5 & 32 \\ 8 & 27 \\ 9 & 93 \end{pmatrix}$$

Multiplication de $M_{(n,p)} * M_{(k,m)} = M_{(n,m)}$

$$\begin{pmatrix} 8 & 8 \\ 0 & 0 \\ 2 & 2 \end{pmatrix} \begin{pmatrix} 1 & 16 \\ 0 & 0 \\ 4 & 6 \end{pmatrix} = \begin{pmatrix} 48 & 80 \\ 0 & 0 \\ 6 & 20 \end{pmatrix}$$

Rappels sur les matrices

Soit x un vecteur (matrice une colonne), $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$

La transposée de x est $x^T = (x_1, \dots, x_n)$

Rappels sur les matrices

Soit M une matrice, la transposée M^T de la matrice M est obtenue par symétrie axiale par rapport à la diagonale principale

1	8
51	3
2	27

1	51	2
8	3	27

$$\begin{pmatrix} 1 & 8 \\ 51 & 3 \\ 2 & 27 \end{pmatrix}^t = \begin{pmatrix} 1 & 51 & 2 \\ 8 & 3 & 27 \end{pmatrix}$$

Rappels sur les matrices

Déterminant d'une matrice M

$$|M| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

$$|M| = aei - afh - bdi + bfg + cdh - ceg$$

Rappels sur les matrices

Inverse d'une matrice M :

- Si le déterminant de la matrice M est non nul alors l'inverse de M^{-1} est :

$$M^{-1} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}^{-1} = \frac{1}{\det M} \begin{pmatrix} ei - fh & fg - di & dh - eg \\ ch - bi & ai - cg & bg - ah \\ bf - ce & cd - af & ae - bd \end{pmatrix}^t = \frac{1}{\det M} \begin{pmatrix} ei - fh & ch - bi & bf - ce \\ fg - di & ai - cg & cd - af \\ dh - eg & bg - ah & ae - bd \end{pmatrix}$$

Rappels sur les matrices

Une matrice stochastique (ou matrice de Markov) est une matrice carrée (finie ou infinie) dont chaque élément est un réel positif et dont la somme des éléments de chaque ligne vaut 1.

$$M = \begin{pmatrix} 0.1 & 0.0 & \dots & 0.6 \\ 0.0 & 0.3 & \dots & 0.0 \\ 0.2 & 0.0 & \dots & 0.1 \\ \dots & \dots & \dots & \dots \\ 0.5 & 0.0 & \dots & 0.2 \end{pmatrix}$$

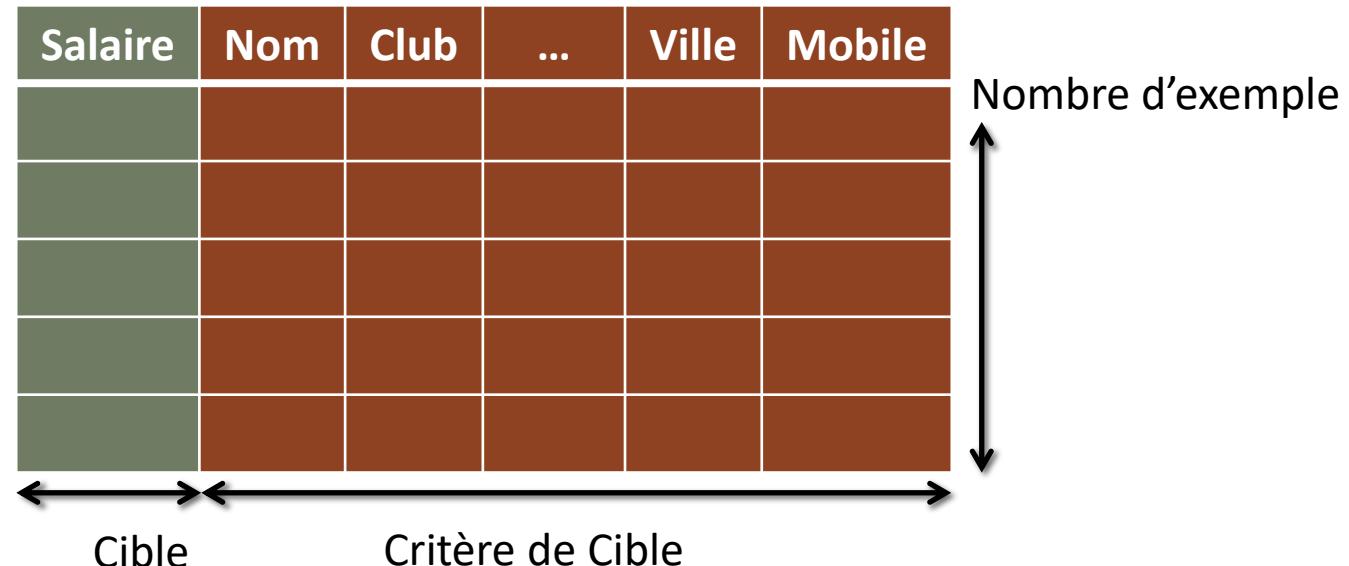


0.1	0	...	0.6
0	0.3	...	0
0.2	0	...	0.1
...
0.5	0	...	0.2

En théorie des probabilités, il s'agit de la matrice de transition d'une chaîne de Markov

Rappels sur les matrices

En machine Learning les données (Data Set) sont rangées dans des matrices :



Rappels sur les matrices

En machine Learning les données ($\text{DataSet}(x, y)$) sont rangées dans des matrices :

y	x_1	x_2	x_3	...	x_n
$y^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$...	$x_n^{(1)}$
$y^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$...	$x_n^{(2)}$
$y^{(3)}$	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$...	$x_n^{(3)}$
...
$y^{(m)}$	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$...	$x_n^{(m)}$

Cible **Critère de Cible : n**

Nombre d'exemple : m

The diagram illustrates a matrix structure for machine learning data. The columns are labeled $x_1, x_2, x_3, \dots, x_n$ and the rows are labeled $y^{(1)}, y^{(2)}, y^{(3)}, \dots, y^{(m)}$. A double-headed vertical arrow on the right is labeled "Nombre d'exemple : m", indicating the number of examples. A double-headed horizontal arrow at the bottom is labeled "Critère de Cible : n", indicating the number of features. The first column is labeled "Cible", which is a French term for target or goal.

Rappels sur les matrices

En machine Learning les données (Data Set(x, y)) sont rangées dans des matrices :

$y \in \mathbb{R}^{m \times 1}$: Vecteur Cible

$$y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{pmatrix}$$

y	x_1	x_2	x_3	\dots	x_n
$y^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	\dots	$x_n^{(1)}$
$y^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	\dots	$x_n^{(2)}$
$y^{(3)}$	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	\dots	$x_n^{(3)}$
\dots	\dots	\dots	\dots	\dots	\dots
$y^{(m)}$	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$	\dots	$x_n^{(m)}$

$X \in \mathbb{R}^{m \times n}$: Matrice des Critères Cible

$$x = \begin{pmatrix} x_1^{(1)} & \dots & \dots & \dots & x_n^{(1)} \\ \dots & \dots & \dots & \dots & \dots \\ x_1^{(m)} & \dots & \dots & \dots & x_n^{(m)} \end{pmatrix}$$

L'interprétation des données

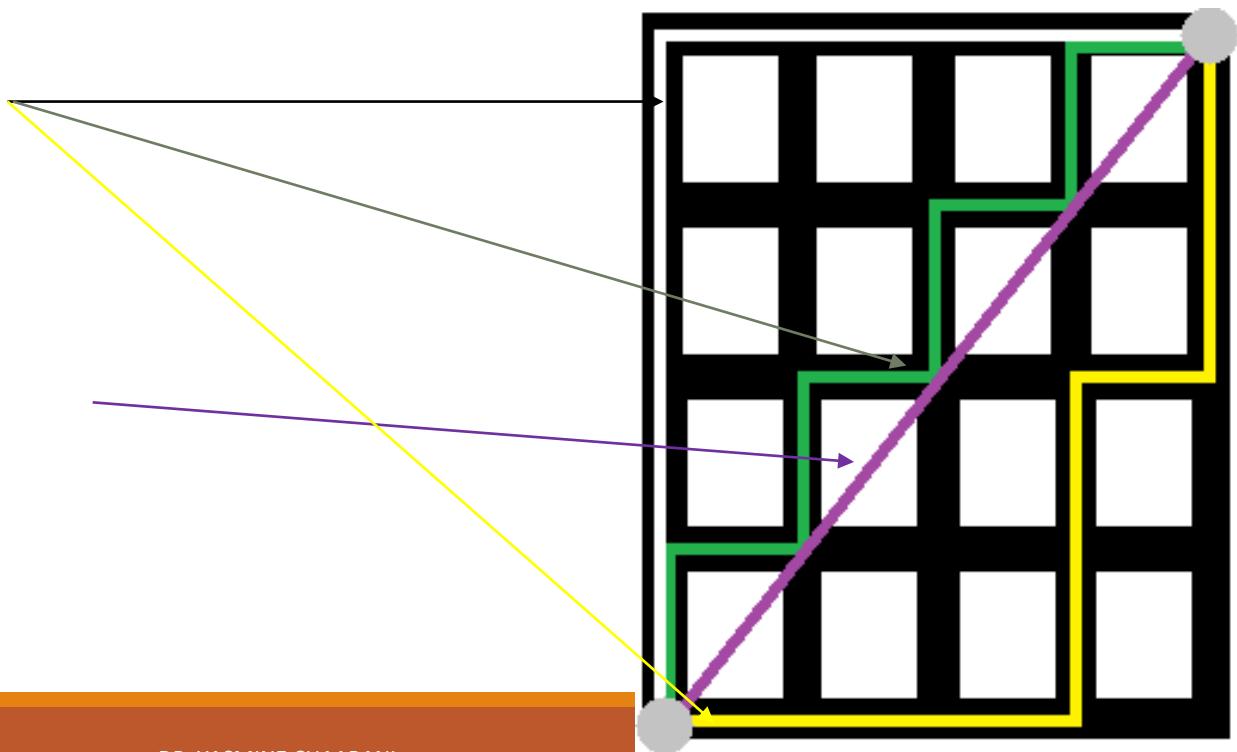
Soit $X = (x_1, x_2, \dots, x_n)$ et $Y = (y_1, y_2, \dots, y_n)$ des ensembles d'observation

Distance de Manhattan :

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

Distance Euclidienne :

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



L'interprétation des données

Soit $X = (x_1, x_2, \dots, x_n)$ et $Y = (y_1, y_2, \dots, y_n)$ des ensembles d'observation, $q - distances$

Distance de Minkowski :

$$\circ d_q(x, y) = (\sum_{i=1}^n |x_i - y_i|^q)^{\frac{1}{q}} = \sqrt[q]{\sum_{i=1}^n |x_i - y_i|^q}$$

Distance de Tchebychev :

$$\circ d_\infty(x, y) = \max_{1 \leq i \leq n} |x_i - y_i| = \lim_{q \rightarrow \infty} \sqrt[q]{\sum_{i=1}^n |x_i - y_i|^q}$$

L'interprétation des données

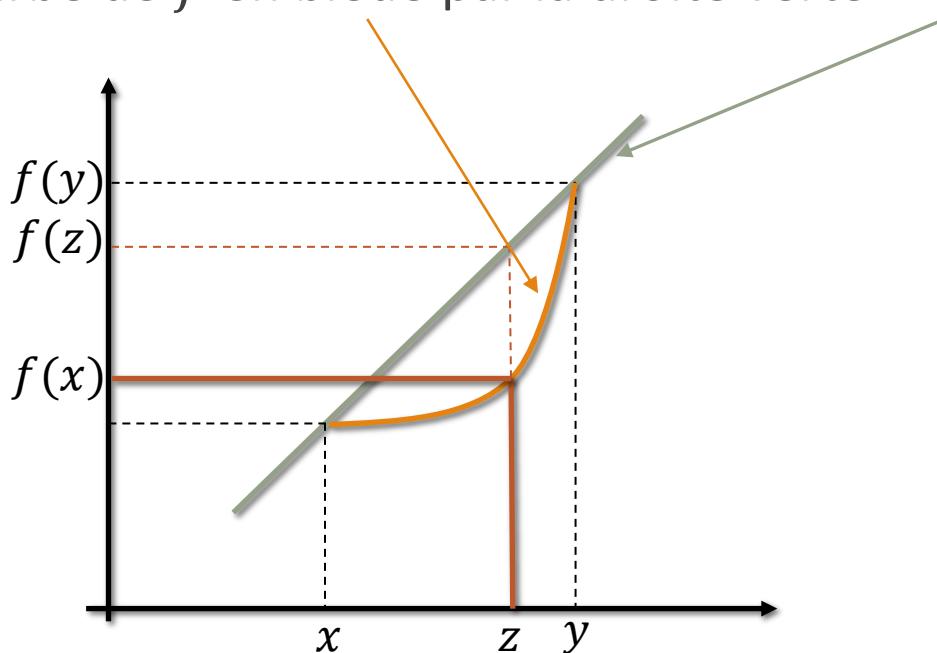
Modèles d'interpolation :

- Linéaire (la fonction d'interpolation des points est une droite)
- Quadratique (la fonction d'interpolation des points est un polynôme de degré 2)
- Cubique (la fonction d'interpolation des points est un polynôme de degré 3)
- Du plus proche voisin (la fonction d'interpolation des points est en escalier)

L'interprétation des données

L'interpolation linéaire :

- Cela consiste à remplacer la courbe de f en bleue par la droite verte



L'interprétation des données

L'interpolation linéaire :

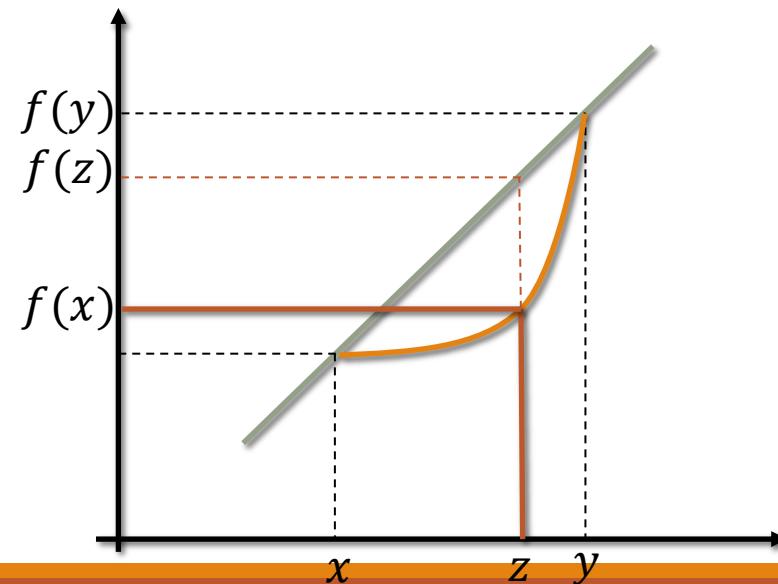
- Soit f une fonction définie sur \mathbb{R}
- Soit $[x, y]$ un intervalle de valeurs sur \mathbb{R}
- z un nombre réel

Si nous n'avons pas de valeur pour $f(z)$, alors il est nécessaire de faire une interpolation linéaire en considérant les données initiales

L'interprétation des données

L'interpolation linéaire :

- Remplacer $f(z)$ par une fonction affine g telle que : $g(x) = f(x)$ et $g(y) = f(y)$
- $$f(z) \approx f(x) + (z - x) \frac{f(y) - f(x)}{y - x}$$



L'interprétation des données

Exemple interpolation linéaire :

- Nous voulons étudier la répartition des âges à partir d'un tableau

Intervalle des âges	Nombre de participant	Nombre de participant cumulés
[0 .. 10[8	8
[10 .. 20[21	29
[20 .. 30[11	40
[30 .. 40[1	41
[40 .. 50[32	73
[50 .. 60[2	75
[60 .. 70[1	76

L'interprétation des données

Exemple interpolation linéaire suite :

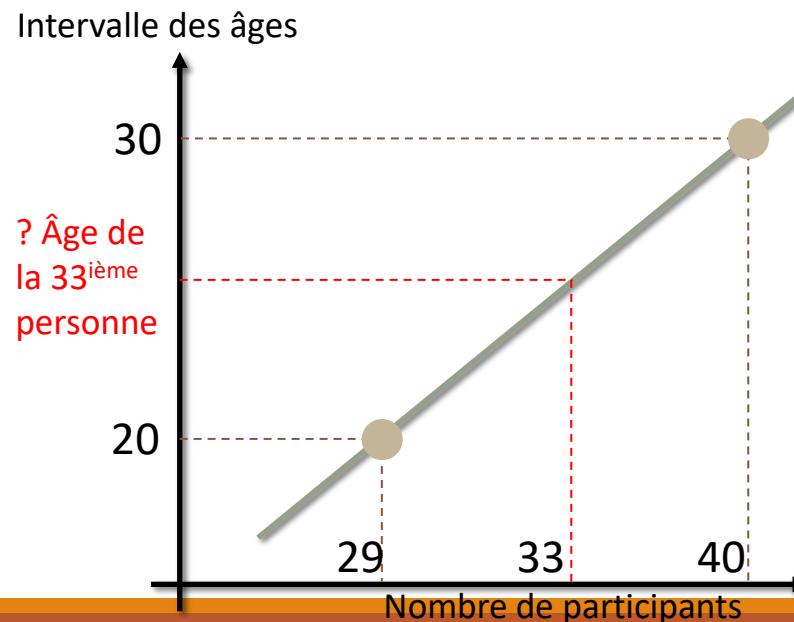
- Nous souhaitons par interpolation linéaire connaitre l'âge de la 33^{ième} personne du tableau

Intervalle des âges	Nombre de participant	Nombre de participant cumulés
[0 .. 10[8	8
[10 .. 20[21	29
[20 .. 30[11	40
[30 .. 40[1	41
[40 .. 50[32	73
[50 .. 60[2	75
[60 .. 70[1	76

L'interprétation des données

Exemple interpolation linéaire suite :

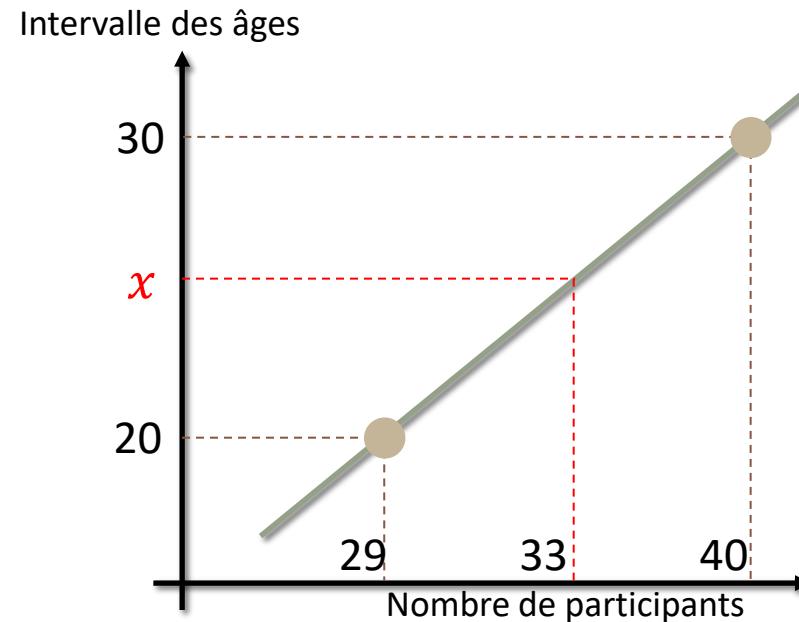
- Nous considérons que les 11 personnes de l'intervalle $[20 .. 30[$ sont réparties proportionnellement à l'intérieur :



L'interprétation des données

Nous pouvons utiliser le théorème de Thales :

- $\frac{x-20}{30-20} = \frac{33-29}{40-29} = \frac{4}{11}$
- $\frac{x-20}{10} = \frac{4}{11}$
- $x - 20 = \frac{4*10}{11}$
- $x = \frac{4*10}{11} + 20 \approx 23,6364$



L'interprétation des données

Fonction polynomiale :

- $f(x) = ax^2 + bx + c$ dont a (différent de 0) représente l'ouverture et le sens de la concavité du graphe. La courbe est de forme parabolique inscrit dans un plan, et à une interception avec l'axe des x , Si $b^2 - 4ac \geq 0$:
 - $f(x) = ax^2 + bx + c, f(x) = 0$ pour $x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$ et $x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$
 - $f(x) = ax^2 + bx, f(x) = 0$ pour $x_1 = 0$ et $x_2 = -\frac{b}{a}$

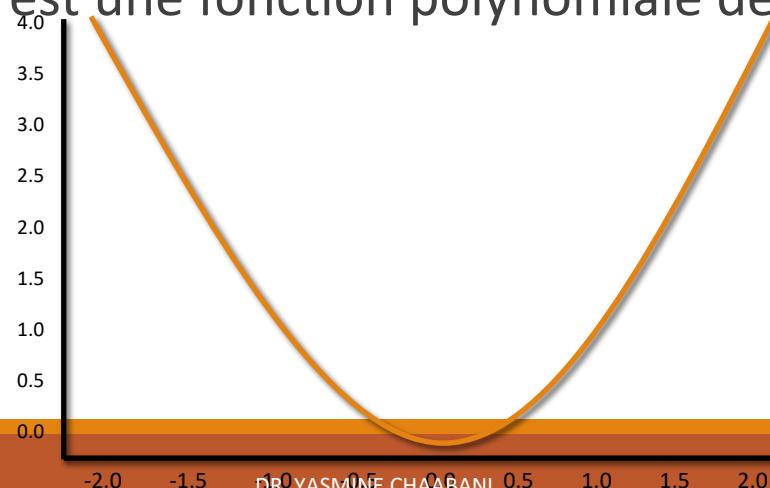
L'interprétation des données

Fonction polynomiale :

- $f(x) = ax^2 + c$, $f(x) = 0$ pour $x_1 = \sqrt{-\frac{c}{a}}$ et $x_2 = -\sqrt{-\frac{c}{a}}$, où $ac < 0$

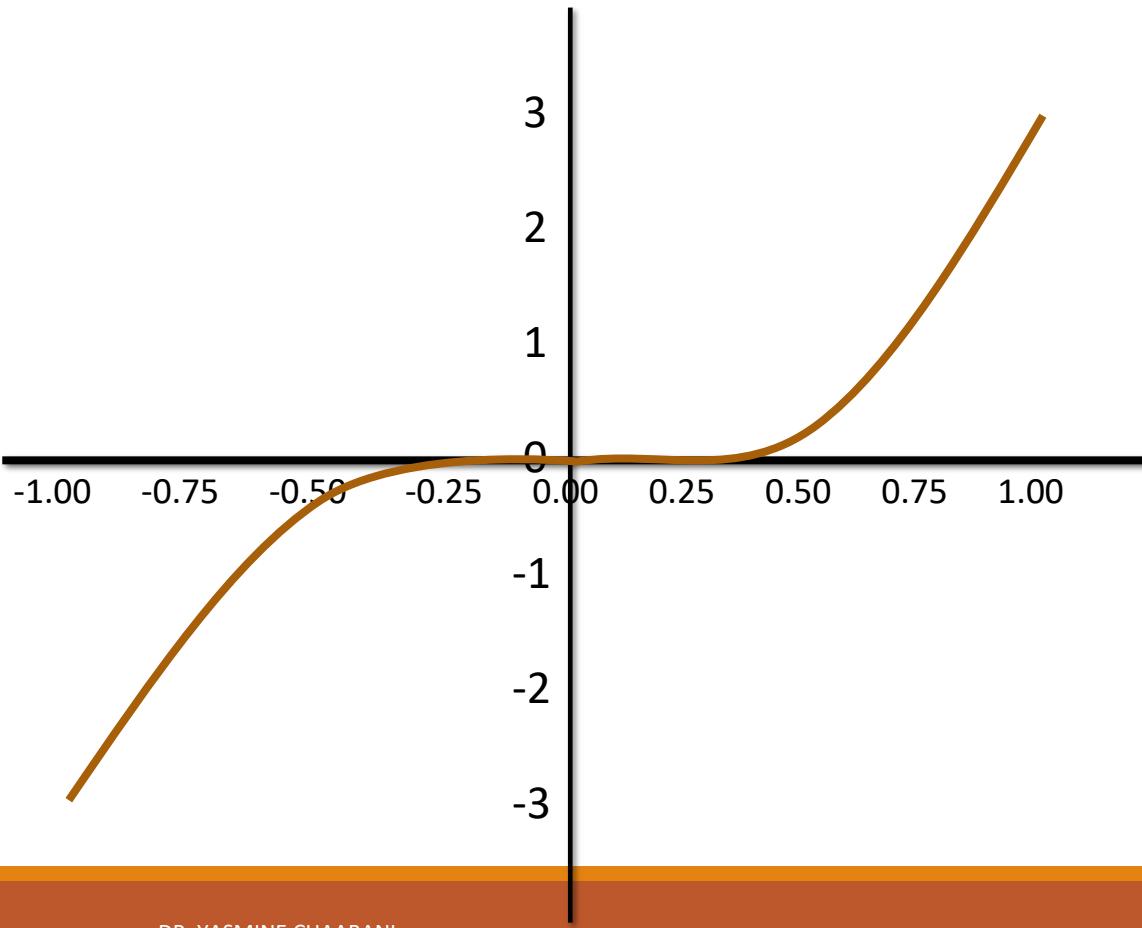
L'interpolation quadratique est une fonction polynomiale de degré 2 :

- $f(x) = ax^2$



Fonction polynomiale :

- $f(x) = ax^3$



Modèles statistiques

La modélisation statistique est :

- Manière simplifiée et formalisée mathématiquement d'approximer la réalité
- Processus qui génèrent vos données
- Permet de faire des prédictions à partir d'approximation
- Equation mathématique utilisée

Modèles statistiques

Exemple : Nous souhaitons modéliser le poids d'une variété de poire

- Manière compliquée :
 - Mesurer le poids de toutes les poires de la variété à travers le monde
 - Afficher les données dans une base de données (fichier)
- Manière simple :
 - Sélectionner un échantillon représentatif de 30 poires de cette variété
 - Calculer la moyenne et l'écart type de cet échantillon
 - Utiliser uniquement ces calculs pour "estimer" le poids de cette variété

Modèles statistiques

Représenter une quantité par une moyenne et un écart type est un cas très simple de modélisation

Les relations entre 2 valeurs, exemple pour, des plantes soumises à une humidité croissante :

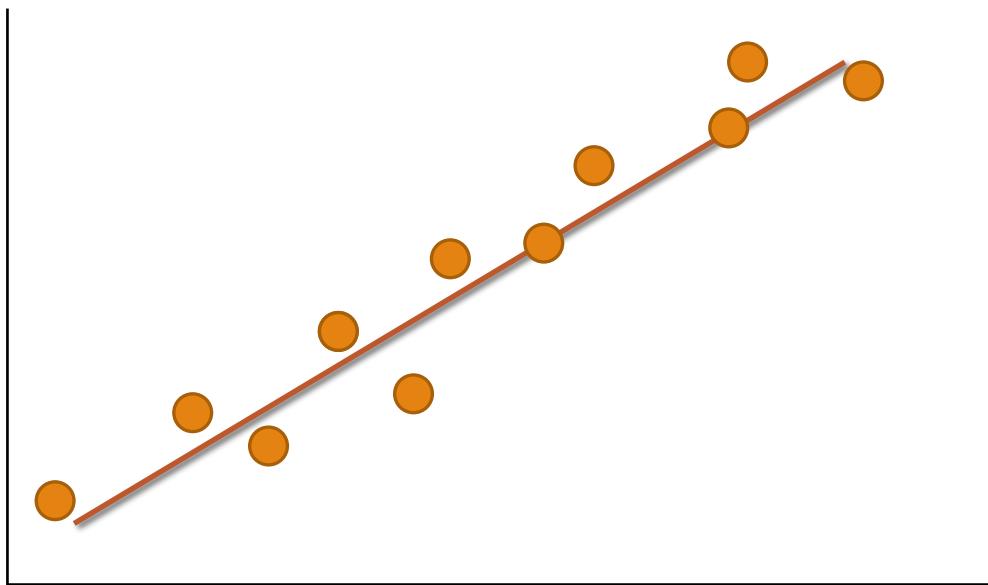
- La hauteur des plantes
- L'humidité du sol

Se représente par une ligne droite caractérisée :

- Une pente
- Une ordonnée à l'origine suite à l'expérience fait sur un échantillon

Modèles statistiques

Le travail de prédiction se fait par lecture du graphique :



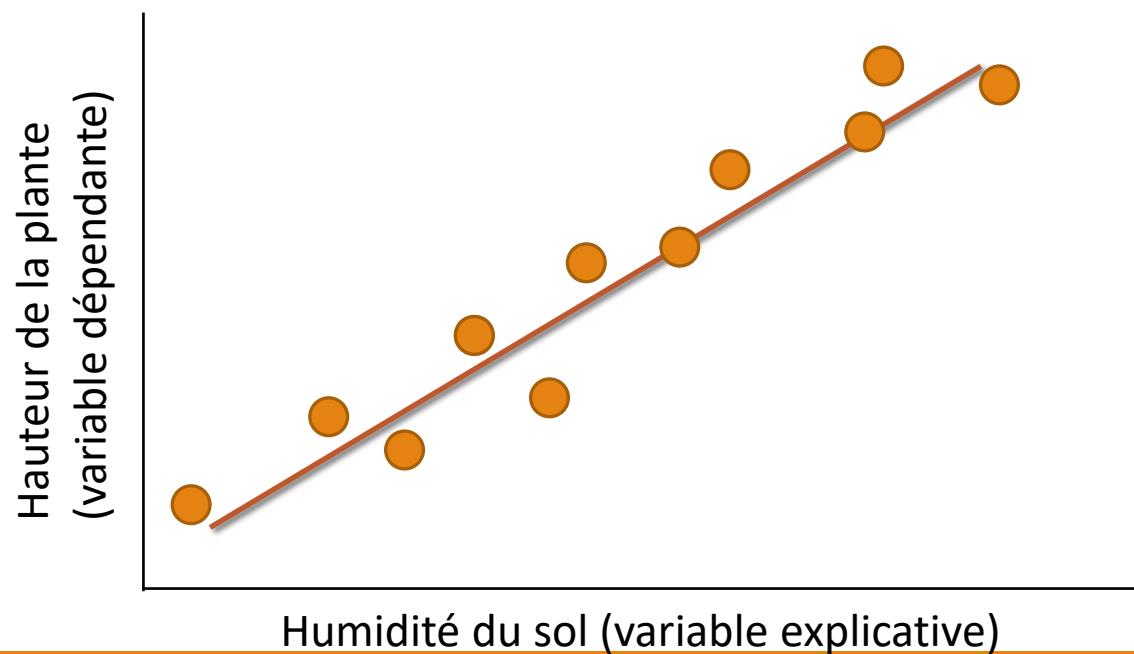
Modèles statistiques

Exercice : Sur le modèle des relations entre les 2 valeurs des plantes soumises à une humidité croissante :

- Quelle est la variable explicative ?
- Quelle est la variable Quantitative ?
- Quelle est la variable dépendante ?

Modèles statistiques

Réponse à l'exercice :



Modèles statistiques

Dans une modélisation paramétrique :

- la (ou les) variable(s) dépendante(s) est liée aux variables explicatives à travers une équation (modèle) impliquant des quantités appelées paramètres du modèle

Dans la régression linéaire sur la hauteur de plantes :

- Les paramètres sont l'ordonnée à l'origine
- La pente 1
- L'équation s'écrit :
 - Hauteur = ordonnée à l'origine + pente * humidité

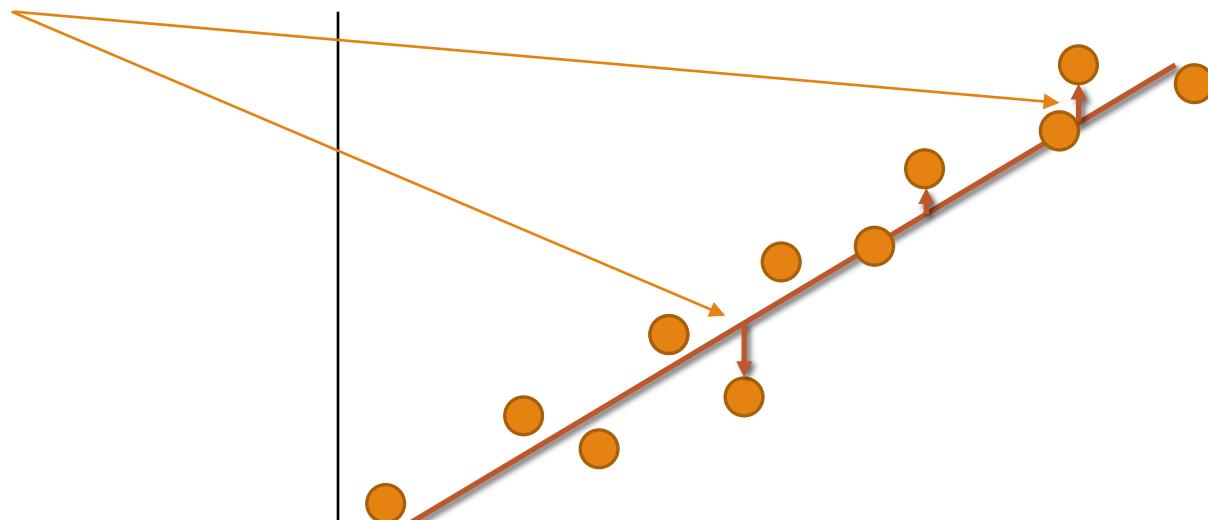
Modèles statistiques

Des calculs permettent d'estimer les paramètres du modèle

Les estimations sont utilisées pour effectuer des prédictions

La régression linéaire simple implique également un troisième paramètre :

- Variance des résidus
- Erreurs d'estimation



Modèles statistiques

Ajustement d'un modèle statistique :

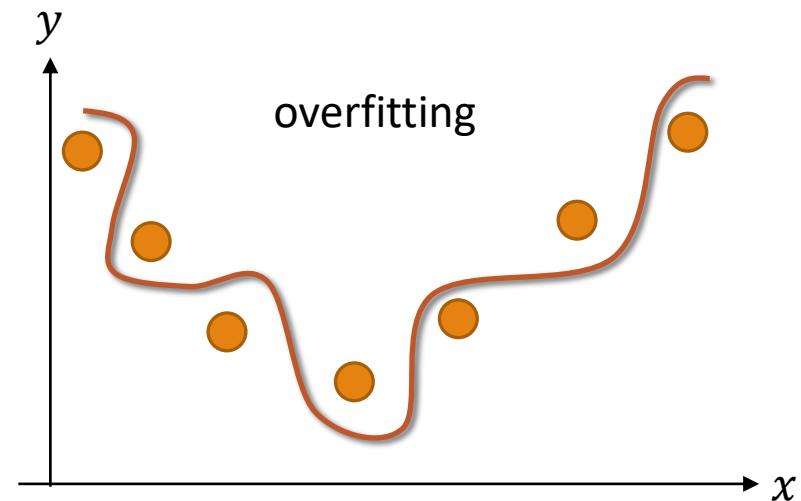
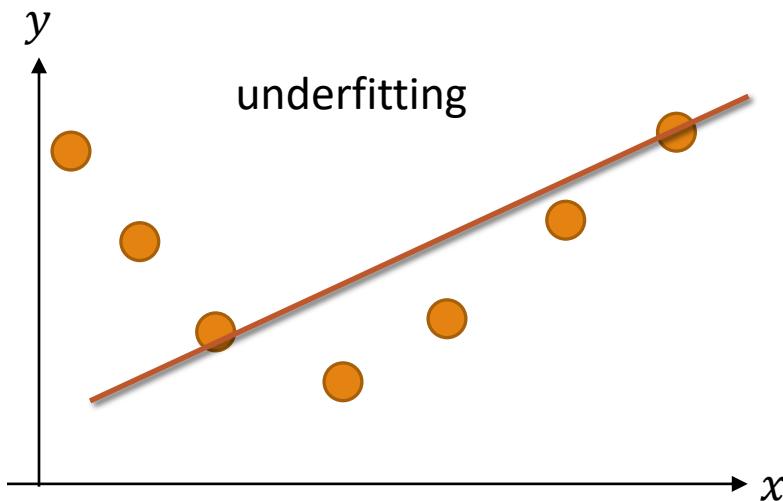
- Pour comprendre la cause profonde de la mauvaise précision du modèle
- Cette analyse guide pour prendre des mesures correctives

Un modèle statistique peut constituer :

- Sous-ajustement
- Surajustement

Modèles statistiques

Exemple d'ajustement :



Modèles statistiques

Le surajustement (surapprentissage), ou surinterprétation (overfitting), est une analyse statistique qui correspond trop précisément à une collection particulière d'un ensemble de données

Cette analyse peut ne pas correspondre à des données supplémentaires ou ne pas prévoir de manière fiable les observations futures

Un modèle sur ajusté est un modèle statistique qui contient plus de paramètres que ne peuvent le justifier les données

Le problème est provoqué par un mauvais dimensionnement de la structure utilisée pour classifier ou faire une régression

Modèles statistiques

Surajustement : Le problème est provoqué par un mauvais dimensionnement de la structure utilisée pour classifier ou faire une régression

De par sa trop grande capacité à capturer des informations, une structure dans une situation de surapprentissage aura de la peine à généraliser les caractéristiques des données

Elle se comporte comme une table contenant tous les échantillons utilisés lors de l'apprentissage et perd ses pouvoirs de prédiction sur de nouveaux échantillons.

Modèles statistiques

Le sous-ajustement (Underfitting) est une analyse statistique qui correspond à une collection mauvaise d'un ensemble de données

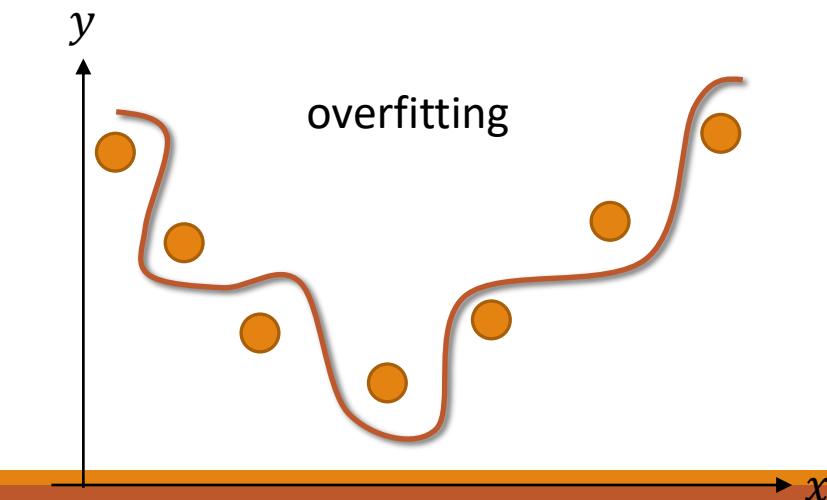
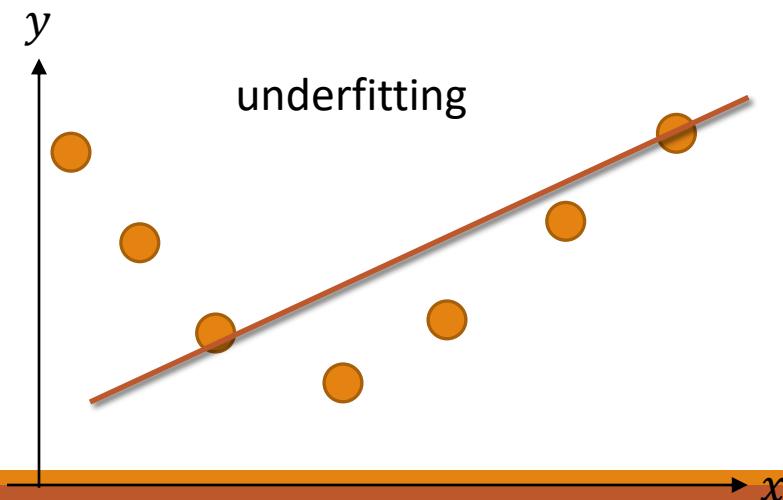
Le modèle n'est pas en mesure de saisir la relation entre les X exemples en entrée et Y les valeurs cibles

De mauvaises performances sur les données de formation sont dues à un modèle trop simple (les entrées ne sont pas expressives) pour décrire correctement la cible

Modèles statistiques

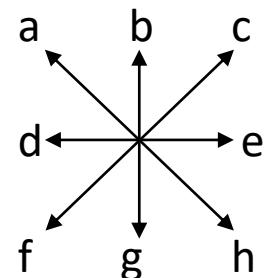
Pour améliorer les performances, il faut augmenter la flexibilité du modèle :

- Ajoutez de nouvelles entités spécifiques et d'autres produits cartésiens d'entités
- Changez le type de traitement d'entités utilisé
- Diminuez le degré de régularisation utilisé



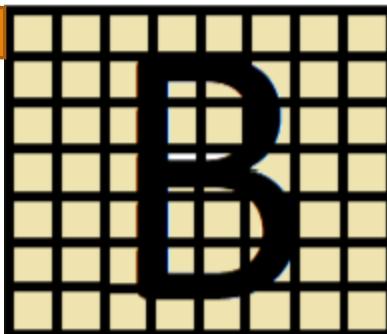
Statistiques et structures

Reconnaissance de forme structurelle



$$X = [x_1, \dots, x_n] \\ = abbca$$

Méthode syntaxique :
Grammaire, automate, Arbre, graphe ...



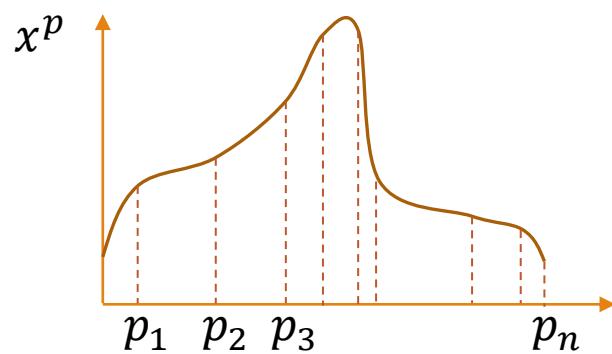
Reconnaissance de forme statistique

$$X = [x_1, \dots, x_n] \\ = 0000 \dots 1111 \dots 0000 \dots$$

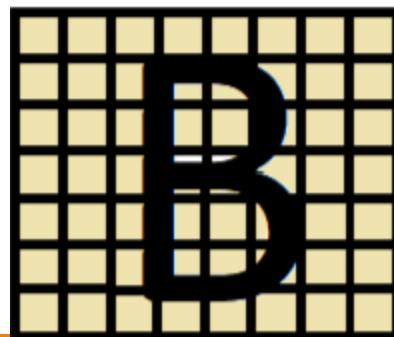
Méthodes statistiques:
Classification

Statistiques et structures

Codage des formes :



$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x^{p_1} \\ x^{p_2} \\ \vdots \\ x^{p_n} \end{pmatrix}$$

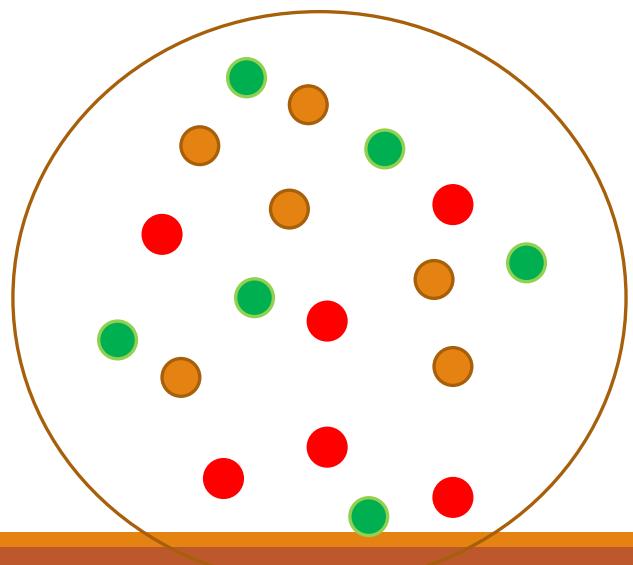


$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^n \end{pmatrix}$$

Statistiques et structures

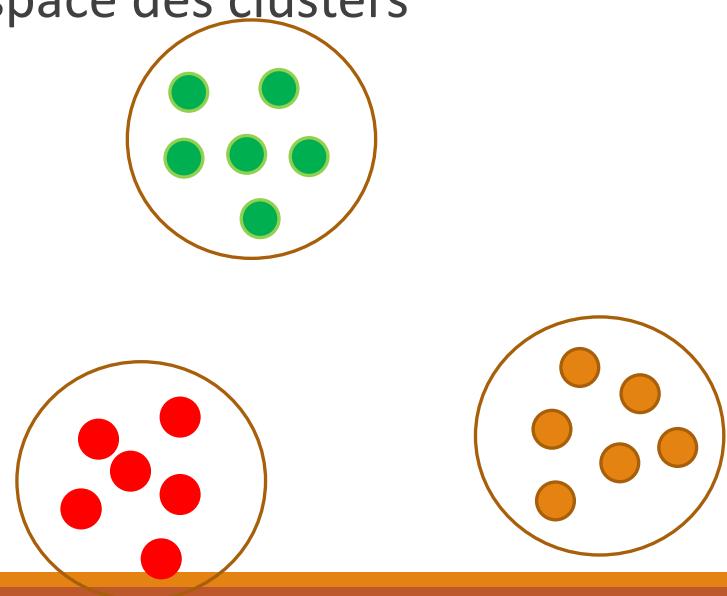
Classification : Clustering

Espace des formes



Regroupement

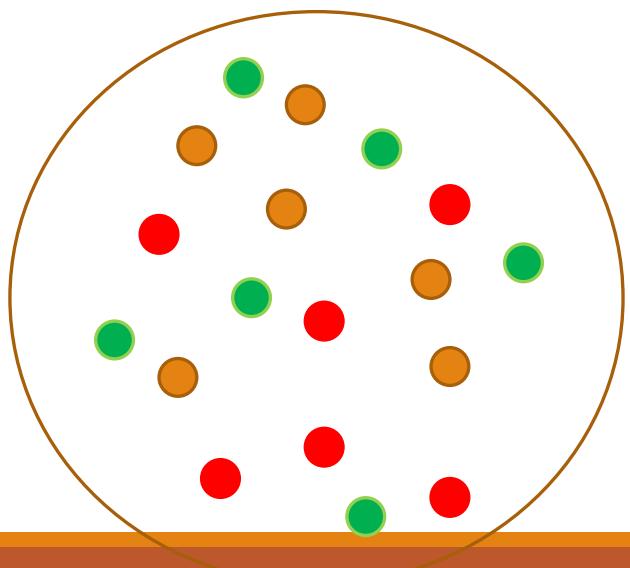
Espace des clusters



Statistiques et structures

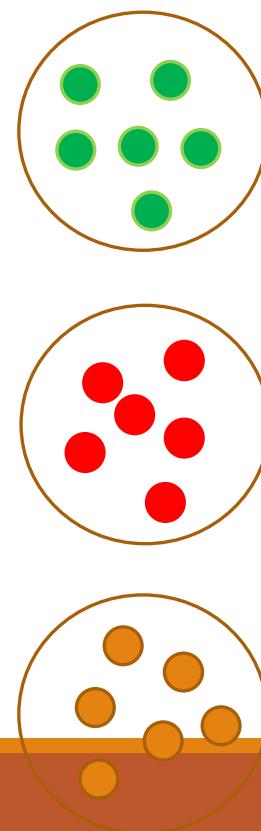
Classification : Classement

Espace des formes

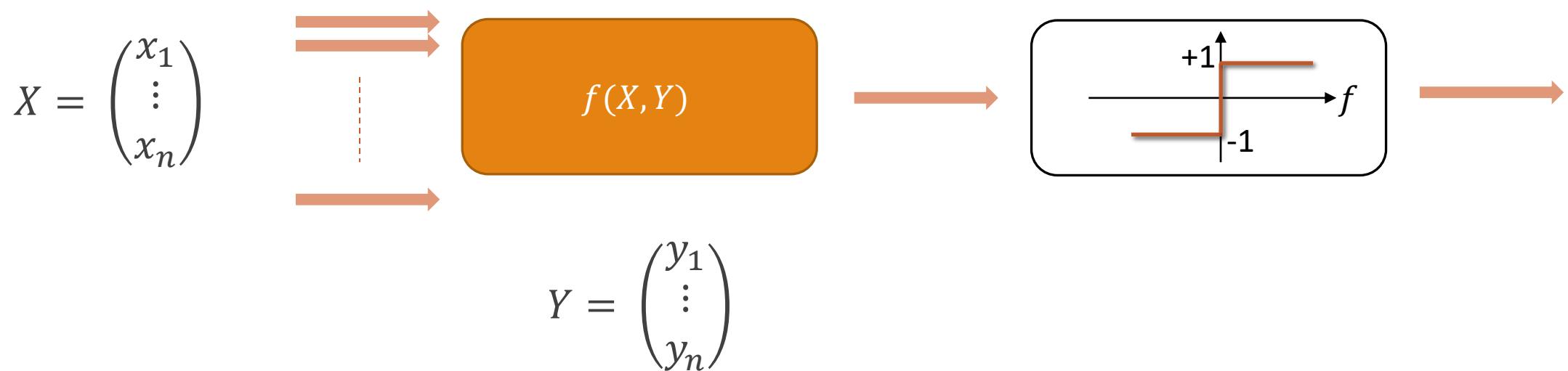


Regroupement

Espace de décision



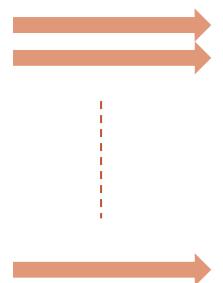
Statistiques et structures



Statistiques et structures

Espace de formes

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$



Espaces de décisions

$$\begin{cases} \text{Probabilité} = P(A|B) \\ \text{ou} \\ \text{Distances} = d(A, B) \end{cases}$$

Langages et outils

GÉNÉRALITÉS : UTILISATION DES
LIBRAIRIES DE PYTHON DANS L'OUTIL
ANACONDA

Dr. Chaabani Yasmine
yasmin.chaabani@isgb.ucar.tn

Le but du module ?

Utiliser des outils pour faire du Machine Learning

Avoir un environnement de développement en python

Comprendre les attributs et méthodes des analyses théoriques précédentes

Aborder le prochain module consacré à l'évaluation de programmes en Machine Learning

Sommaire

Quels sont les langages et outils pour faire du Machine Learning ?

Installation et prise en main de Anaconda

Les modèles dans python

Les mathématiques dans python

Les graphiques dans python

Les calculs scientifiques dans python

Les modules d'analyses de données et de visualisation dans python

Les modules de Machine Learning et de Data Set dans python

Le traitement d'images comme base de données numériques

Quels sont les langages et outils pour faire du Machine Learning ?

Pour créer une intelligence artificielle et pour mieux exploiter ses données, un framework Machine Learning est utile : Anaconda.com (Jupyter, spider)

python :

- Les modules Scientifiques (numpy, matplotlib, Scipy)
- Les modules d'analyses de données et de visualisation (pandas, seaborn, statsmodels)
- Les modules de Machine Learning et de data Set (scikit-Learn, pickles, hdfs)

<https://insights.stackoverflow.com/survey/2019#technology>

Les autres langages comme C/C++, Java, JavaScript, R ne seront pas abordés dans ce cours

Quels sont les langages et outils pour faire du Machine Learning ?



Quels sont les langages et outils pour faire du Machine Learning ?

Les présentations suivantes dans les slides sont centrées sur les packages, librairies et modules en python

Toutes les manipulations sont guidées Machine Learning avec les modules python

Les modules présentés sont appliqués pour faire de l'exploration de données, de l'interprétation de données, de la présentation données et du calcul sur les données

Installation et prise en main de Anaconda

Les bibliothèques python scientifiques pour créer nos programmes en Machine Learning sont disponibles dans Anaconda



En utilisant Anaconda, il n'y a pas d'installation et de configuration à faire

Vous disposez d'un processus d'installation trivial, disponible en python 3

Installation et prise en main de Anaconda

Mise en place Anaconda avec Continuum Analytics : continuum.io

Prise en main de Jupyter et possible de spyder = éditeur de textes



Spyder

Les modules dans python

Dans Jupyter et en python rappel et/ou études des modules :

- math
- random
- statistics
- os
- glob

Les modules dans python

Exercices :

- Faire les instructions des prochains slides avec les modules de python
- Dans Jupyter et pour la suite, nous aurons 2 listes :

```
Liste_1 = [i*2 for i in range(100)]  
Liste_2 = ['Anne', 'Alain', 'Amid', 'Karim', 'Lina', 'Malia', 'Nora', 'Patrick']
```

Les modules dans python

Exercices, avoir de l'aleatoire, faire les instructions suivantes :

```
random.choice(Liste_1 )  
random.choice(Liste_2 )  
random.uniform(0, 11)  
random.randint(1,20)  
random.randrange(50)  
random.sample(range(200), 20)  
random.sample(range(50), random.randrange(20))  
random.sample(Liste_2,5)  
random.shuffle(Liste_2)
```

Les modules dans python

Exercice, avoir de l'aleatoire bloqué qui peut être reproduit :

```
random.seed(nombre)
random.choice(Liste_1)
random.choice(Liste_2)
```

Récupérer le répertoire de travail :

```
os.getcwd()
```

Les modules dans python

Récupérer la liste de fichiers de travail :

- `glob.glob("*")`
- `glob.glob("*.extension")`

Exercice :

- Faire un fichier nommé `nombre.txt` dans le répertoire de travail comportant 10 nombres les uns en dessous des autres
- Faire un fichier nommé `nom.txt` dans le répertoire de travail comportant 5 noms les uns en dessous des autres
- Utiliser la fonction `glob` pour récupérer les fichiers `*.txt`
- Copier le contenu des fichiers `.txt` dans une variable nommée `f`
- Afficher tout les contenus de `f`

Les modules dans python

Réponse possible à l'exercice :

```
with open('nom.txt', 'w') as f:  
    for i in range(5):  
        a = input()  
        f.write("{} \n".format(a))
```

```
with open('nombre.txt', 'w') as f:  
    for i in range(10):  
        f.write("{} \n".format(i))
```

Les modules dans python

Réponse possible à l'exercice :

```
with open('nombre.txt', 'r') as f:  
    l = f.read().splitlines()
```

```
l = [l.strip() for l in open('nom.txt', 'r')] 
```

```
for f in l:  
    with open(f, 'r') as n:  
        print(n.read())
```

Les modules dans python

Exercice

- A partir de tous les fichiers faire une structure pour lister chaque contenu des fichiers avec une référence
- Afficher, un à un, les contenus de chaque fichier comme une liste

Les modules dans python

Réponse possible à l'exercice :

```
d = {}
for f in l:
    with open(f, 'r') as n:
        d[f] = n.read().splitlines()
liste = []
for f in d.values():
    liste.append(f)
```

Les modules dans python

Rappel :

- Le package numpy (NUMerical PYthon) donne accès aux fonctions mathématiques basées sur les tableaux et les matrices en entrées

```
import numpy as np
np.random.RandomState(0).uniform(0, 8, 10)
np.random.RandomState(0).uniform(0, 5, 100)[:, np.newaxis]
```

Les modules dans python

Exercice

- Avec numpy faire un tableau à 1 dimension comportant une liste de 10 entiers pris au hasard
- Quelle méthode dans numpy utiliser pour trouver les dimensions d'un tableau ?
- Quelle méthode dans numpy utiliser pour trouver la taille d'un tableau ?
- Quelle méthode dans numpy utiliser pour trouver la forme d'un tableau ?
- Quel est le type du résultat retourné pour donner la forme d'un tableau ?
- Avec numpy faire une matrice de 5 lignes, 4 colonnes contenant que des zéros
- Avec numpy faire une matrice de 5 lignes, 4 colonnes contenant que des uns
- Avec numpy faire une matrice de 5 lignes, 4 colonnes contenant des valeurs pris au hasard suivant une distribution normale

Les modules dans python

Réponse possible à l'exercice :

```
import numpy as np
import random as r
l = [r.randint(i, 100) for i in range(1, 10)]
t = np.array(l)

t.ndim
t.size
t.shape

# Le résultat de shape est un tuple
a = np.zeros((5,4))
tab = np.ones((5,4))
b = np.random.randn(5,4)
```

Les modules dans python

Exercice :

- Avec numpy faire 2 matrices de 3 lignes, 2 colonnes, avec l'une contenant que des zéros, l'autre que des 1
- Avec numpy fusionner les 2 matrices de façon verticale
- Avec numpy fusionner les 2 matrices de façon horizontale
- Avec numpy faire 2 tableaux contenant l'un 1, 2, 3, 4 et l'autre 5, 6, 7
- Faire, à partir des 2 tableaux, 2 matrices, une de 4 lignes, 3 colonnes et l'autre de 3 lignes, 4 colonnes contenant la duplication par colonnes ou par lignes des valeurs des tableaux. En résultat, nous aurons une matrice avec 3 lignes, 4 colonnes de 1, 2, 3, 4 et l'autre avec 3 lignes, 4 colonnes de 5, 6, 7

Les modules dans python

Réponse possible à l'exercice :

```
a = np.zeros((3,2))
b = np.ones((3,2))

np.concatenate((a,b), axis = 0)
np.concatenate((a,b), axis = 1)

(x, y) = np.meshgrid([1,2,3,4],[5,6,7])
```

Les modules dans python

Exercice :

- Que fait le code suivant et pourquoi l'utiliser ?

```
t = np.array([1, 5, 7])
t.shape
t = t.reshape(t.shape[0],1)
t.shape
t = t.squeeze()

t = np.ones((5,6))
t.ravel()
```

Les modules dans python

En machine Learning, le travail est souvent en 2 dimensions :

- Exercice pour : `t = np.random.randint(0,25,[8,8])`
- Faire un masque simple de valeur dans le tableau : `t < 20`
- Possible de faire de l'indexing : `t[t==20] = 0`
- Possible de redimensionner : `t[t<10]`

Les modules dans python

Réponse possible à l'exercice :

```
t = np.random.randint(0,25,[8,8])
t < 20
t[t==20] = 0
t
```

Les modules dans python

Exemple :

```
a = np.zeros((5,5))
a[::2,::2] = 1
t = np.random.randint(0,10,[5,5])
t = t[t<2]
t
```

Les mathématiques dans python

Travailler sur les axes des tableaux avec numpy

Exercice :

- Faire 2 matrices(4,6) d'entiers pris au hasard inférieur à 10 avec numpy nommés A et B
- Faire la somme des tous les éléments de la matrice A
- Faire la somme de toutes les lignes de la matrice A dans un tableau nommé C
- Faire la somme de toutes les colonnes de la matrice A dans un tableau nommé D
- Faire la somme cumulée de toutes les valeurs de la matrice B dans une tableau nommé E
- Faire la moyenne de tous les entiers de la matrice B dans une variable nommée F
- Faire l'enregistrement des positions des valeurs minimales dans toutes les colonnes de la matrice A dans un tableau nommé G

Les mathématiques dans python

Réponses possible à l'exercice :

```
import numpy as np
A = np.random.randint(0,10,[4,6])
B = np.random.randint(0,10,[4,6])
C = A.sum(axis = 0)
D = A.sum(axis = 1)
E = B.cumsum()
F = B.mean()
G = A.argmin(axis=1)
```

Les mathématiques dans python

Exercice :

- Faire l'écart type des valeurs du tableau B
- Faire la variance sur les valeurs du tableau A
- Tracer une matrice de corrélation pour la matrice B
- Faire 2 tableaux nommés H et I qui donne les valeurs et les fréquences des valeurs de la matrice A

```
for i,j in zip(H[I.argsort()], I[I.argsort()]):  
    print(f'{i} : {j}')
```

- Que fait le code ? :

- Modifier le code pour mieux comprendre le code
- Quelle méthode utiliser pour trouver des Nan dans un tableau ?

Les mathématiques dans python

Réponses possible à l'exercice :

```
B.std()  
  
A.var()  
  
np.corrcoef(B)  
  
H, I = np.unique(A,return_counts=True)  
for i,j in zip(H[I.argsort()], I[I.argsort()]):  
    print(f'Entier {i} : fréquence {j}')  
  
np.isnan(B)
```

Les mathématiques dans python

Exemple :

```
import numpy as np
import statistics as sts
x = np.random.normal(size=1000,loc=180)
# Moyenne :
y = np.mean(x)
print(y)
# Moyenne géométrique :
y = sts.geometric_mean(x)
print(y)
# Moyenne harmonique :
y = sts.harmonic_mean(x)
print(y)
```

Les mathématiques dans python

Exemple :

```
import numpy as np
import statistics as sts
x = [120,16,11,23,81,12,110,9]

# écart-type
et = np.sqrt(np.sum((np.array(x)-np.mean(x))**2)/(len(x)-1))
print(et)

# écart-type d'une population
etp = np.std(x)
print(etp)

# écart-type d'un échantillon (avec division par n-1)
ete = np.std(x, ddof = 1)
print(ete)
# Avec la librairie statistique
etest = sts.stdev(x)
print(etest)
```

Les mathématiques dans python

Exercice :

- Faire une matrice nommée A de 10 lignes, 5 colonnes, avec des valeurs entières comprises entre 0 et 50
- Fixer les valeurs pour retrouver toujours les mêmes valeurs dans le data set, par exemple fixer la gestion aléatoire à 1
- Standardiser la matrice A sur chaque colonne, c'est-à-dire sur chaque colonne de A faire :
 - $\frac{A - \bar{A}}{\sigma(A)}$

Les mathématiques dans python

Réponse possible à l'exercice :

```
import numpy as np

np.random.seed(1)

A = np.random.randint(0,100,[10,5])
B = (A - A.mean(axis=0)) / A.std(axis = 0)
```

Les mathématiques dans python

Bilan :

- Numpy est un package python qui manipule les tableaux de 1 à plusieurs dimensions
- En Machine Learning, nous travaillons principalement sur des tableaux à 2 dimensions, avec 2 axes, 0 (Vertical) et 1 (Horizontal)
- Les attributs à bien conserver sont :
 - Shape (analyser les dimensions du tableau = Combien il y a de lignes, colonnes)
 - Size (taille = comprendre le nombre d'élément du tableau)

Les mathématiques dans python

Bilan :

- Les méthodes :
 - concatenate (pour assembler 2 tableaux en fonction des axes)
 - reshape (pour redimensionner les tableaux)
 - ravel (pour faire la mise à plat des tableaux à 1 dimension)
 - argsort (permet de faire les classements et tris dans les tableaux)
 - Le Boolean, masque directement sur les données du tableau (filtrer dans un tableau)
- Les fonctions mathématiques avancées sont dans scipy

Les graphiques dans python

L'outil graphique montre des informations, les données, les modèles

Le graphique doit aider dans nos déroulements de Machine Learning

Matplotlib dans python au service de nos Machines Learning

Les graphiques dans python

Quel est le but des graphiques ?

Que doit comporter un graphique ?

Combien de méthodes avons-nous pour faire des graphiques dans matplotlib ?

Les graphiques dans python

Pour rendre visible les courbes par plot dans pyplot :

- `plot(x, y, label = ..., lw = ..., ls = ..., c = ...)`
 - `label` → étiquette pour la légende
 - `lw` → Line width
 - `ls` → Line Style
 - `c` → Color

Les graphiques dans python

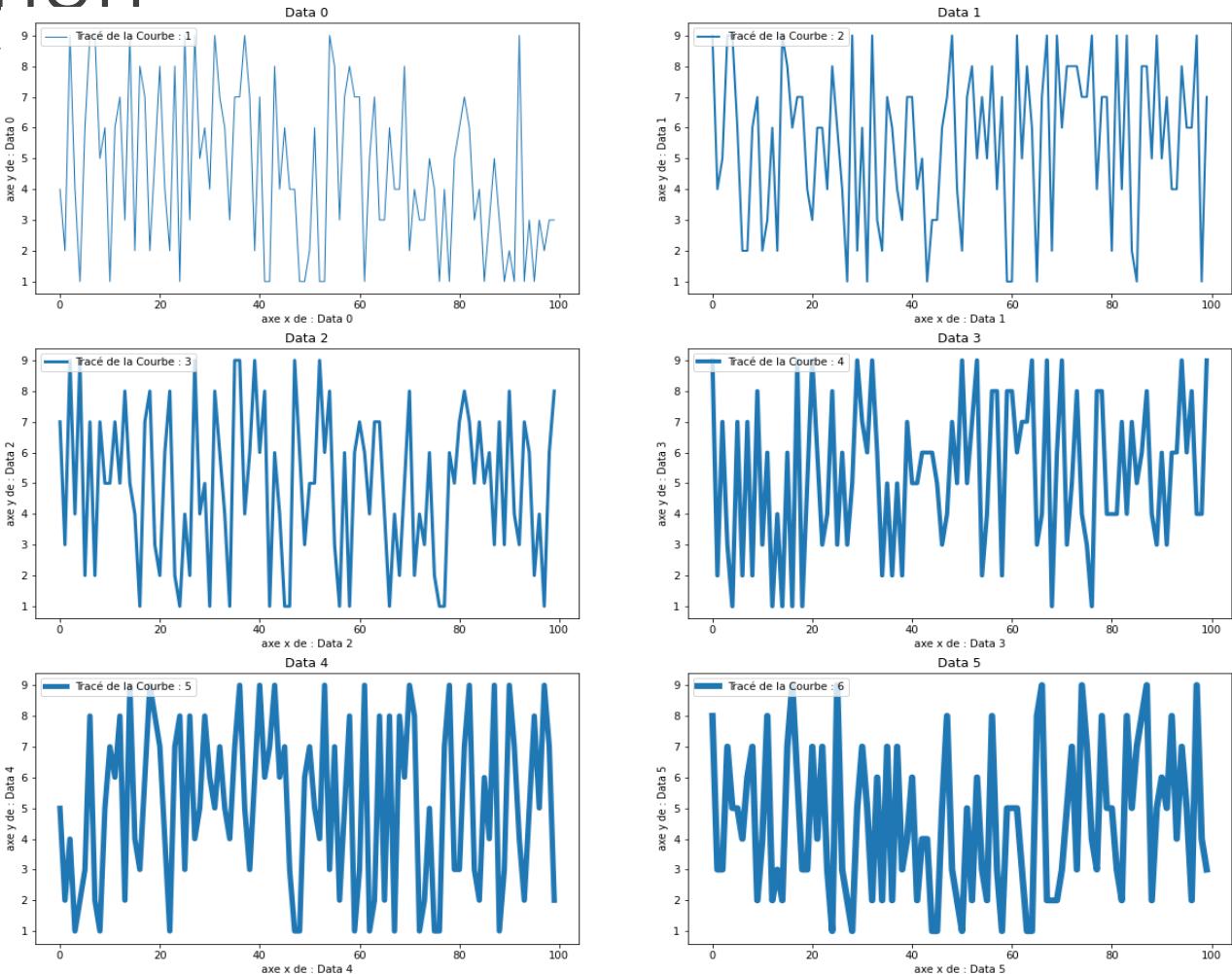
Environnement d'une figure dans pyplot :

- `figure(figsize =)` → cycle de vie du graphique = fenêtre de travail
 - `figsize` → paramètre de taille pour les axes x et y
- `title()`
- `xlabel('...')` → nom de l'axe des x
- `ylabel('...')` → nom de l'axe des y
- `legend()` → en fonction du label de plot
- `show()` → pour visualiser le graphique
- `savefig('...')` → pour sauver le graphique dans le répertoire de travail
- `subplot(NbrLignes, Nbcolonnes, Valposition)` → pour faire des matrices de graphiques

Les graphiques dans python

Exercice :

- Vous disposez du data Set suivant :
 - `dataset = {f"Data {i}" : np.random.randint(1,10,100) for i in range(6)}`
- Faire des méthodes python en utilisant la classe, `matplotlib.pyplot`, pour réaliser les graphiques :



Les graphiques dans python

Réponse possible à l'exercice :

```
import numpy as np
import matplotlib.pyplot as plt

dataset = {f"Data {i}": np.random.randint(1,10,100) for i in range(6)}

def courbe(keys, dataset, i):
    plt.plot(dataset[keys], label = "Tracé de la Courbe : " + str(i), lw = i)
    plt.xlabel('axe x de : ' + keys)
    plt.ylabel('axe y de : ' + keys)
    plt.title(keys)
    plt.legend(loc = 'upper left')
```

Les graphiques dans python

Réponse possible à l'exercice suite :

```
def dessiner (dataset):
    n = len(dataset)
    plt.figure(figsize = (20,35))
    i = 1
    for keys in dataset.keys():
        if n < n/2 + 1:
            plt.subplot(n, 1, i)
            courbe(keys, dataset, i)
        else:
            plt.subplot(n, 2, i)
            courbe(keys, dataset, i)
        i = i+1
    # Sauvegarde de l'image des courbes
    plt.savefig('exo.png')
    plt.show()
```

Les graphiques dans python

En Machine Learning, nous souhaitons résoudre des problèmes d'optimisation, comme trouver le minimum d'une fonction coût, maximiser des revenus, ...

Les graphiques aident à résoudre la classification en Machine Learning :

- scatter()
- projection 3D avec scatter()
- hist(), hist2d()
- contour() (visualiser un modèle avec 3 dimensions en 2d)
- imshow() (permet d'afficher tous les tableaux, matrices, ... de numpy)

Les graphiques dans python

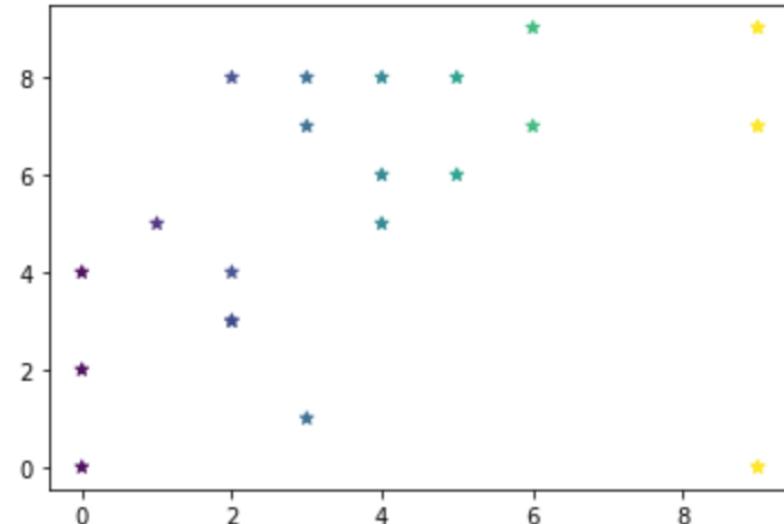
scatter() :

```
import numpy as np
import matplotlib.pyplot as plt

A = np.random.randint(0,10,[4,6])
B = np.random.randint(0,10,[4,6])

plt.scatter(A,B, marker= '*', c = A, alpha = 0.9)

<matplotlib.collections.PathCollection at 0x1c02fcb4df0>
```



Les graphiques dans python

Différents Symboles :

```
point = np.random.RandomState(0)

for symbole in ['o', 's', 'd', 'x', 'v', '.', ',', '+', '*', '^', '<', '>']:
    plt.plot(point.rand(4), point.rand(4), symbole, label="Symbole de :'{0}'".format(symbole))

plt.title('Exemple des symboles sur un graphe')
plt.xlabel('les ordonnées x')
plt.ylabel('les abscisses y')
plt.legend()
plt.xlim(-0.2, 2)
plt.ylim(-0.2, 1.2)
```

Les graphiques dans python

Projection 3D avec scatter() :

```
import numpy as np
import matplotlib.pyplot as plt

x = np.random.random(20)
y = np.random.random(20)
z = np.sin(3*x**2+y**2)

fig = plt.figure(figsize=(5,5))
ax = fig.add_subplot(projection= '3d')
ax.scatter(x,y,z)

ax.plot(x, z, 'r+', zdir='y', zs=2.0)
ax.plot(y, z, 'g+', zdir='x', zs=-1.0)
ax.plot(x, y, 'k+', zdir='z', zs=-2.0)

ax.set_xlim([-1.5, 2.0])
ax.set_ylim([-2.5, 2.0])
ax.set_zlim([-1.0, 2.0])
```

Les graphiques dans python

hist() :

```
x = np.random.randint(1,5,5)
y = np.random.randint(0,4,5)

bins = [i + 0.5 for i in range(0, 6)]

fig = plt.figure(figsize=(8,8))
plt.hist([x, y], bins = bins, color = ['skyblue', 'green'],
         edgecolor = 'red', hatch = '.', label = ['x', 'y'],
         histtype = 'bar')
plt.ylabel('Valeurs')
plt.xlabel('Nombres')
plt.title('Construction Histogramme')
plt.legend()
```

Les graphiques dans python

hist2d() :

```
import numpy as np
import matplotlib.pyplot as plt

plt.hist2d(np.random.randint(1,10,100),np.random.randint(1,10,100), cmap='Blues')
plt.xlabel('axe X')
plt.ylabel('axe y')
plt.colorbar()
```

Les graphiques dans python

contour() :

```
plt.contour(np.random.randint(1,10,100),np.random.randint(1,10,100),np.random.randint(1,10,[100,100]), 20)
```

Les graphiques dans python

imshow() :

```
x = np.random.randint(1,100,[35,35])
plt.imshow(x.T)
plt.colorbar()
```

Les graphiques dans python

Exemple de courbe avec des axes positionnés :

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-1,1,100)
y = 3*x**3

# mettre les axes au centre
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.spines['left'].set_position('center')
ax.spines['bottom'].set_position('center')
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
plt.title('Courbe d\'une fonction cubique')

# Ajouter le tracé de la courbe
plt.plot(x,y)
plt.show()
```

Les graphiques dans python

Exemple de courbe quadratique :

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 28, 70)
y = -4*x ** 2 + 28*x
fig = plt.figure(figsize = (8, 5))
plt.plot(x, y, linewidth = 3)
plt.title('Courbe d\'une fonction quadratique')
plt.xlabel('x ')
plt.ylabel('f(x)')
plt.show()
```

Les graphiques dans python

Plusieurs courbes sur un même graphique :

```
import numpy as np
import matplotlib.pyplot as plt

A = np.random.randint(1,10,[5,5])
B = A * 2
C = np.random.randint(1,10,5)

plt.figure(figsize=(10,10))

plt.subplot(1,2,1)
plt.plot(A[0],label='Courbe de A[0]',lw = 1, ls = '--', marker = 'o', c="red")
plt.plot(A[1],label='Courbe de A[1]',lw = 1, ls = '--', marker = '+', c="yellow")
plt.plot(B[2],label='Courbe de B[2]',lw = 1, ls = '--', marker = 'o', c="blue")
plt.plot(B[3],label='Courbe de B[3]',lw = 1, ls = '--', marker = '+')
plt.title('Cours Machine Learning pour A et B')
plt.xlabel('axe des X')
plt.ylabel('axe des Y')
plt.legend()

plt.subplot(1,2,2)
plt.plot(C,label='Courbe de C',lw = 1, ls = '--', marker = '+')
plt.title('Cours Machine Learning pour C')
plt.xlabel('axe des X')
plt.ylabel('axe des Y')
plt.legend()

# Sauvegarde de l'image des courbes
plt.savefig('fichier.png')
plt.show()
```

Les calculs scientifiques dans python

Le package SciPy (SCientific PYthon) va nous offrir des modules pour faire nos calculs scientifiques

Le package ajoute à notre environnement d'analyse et de développement des fonctions en utilisant les données stockées dans les tableaux et les matrices comme :

- Interpolation entre coordonnées
- Calcul et résolution de systèmes d'équations différentielles
- Traitement et analyse de signal

Les calculs scientifiques dans python

Interpolation linéaire en utilisant `scipy.interpolate` :

- Permet d'ajouter des valeurs aux valeurs initiales pour avoir plus de données
- Permet de suivre une équation linéaire entre les points

Exercice :

- Création d'un data Set de 2 tableaux, avec [8,29,40] et (20,30,3)
- Création de la fonction interpolation avec `interp1d` de `scipy.interpolate` sur 100 valeurs
- Affichages de données produite
- Visualisation avec `matplotlib` du résultat

Les calculs scientifiques dans python

Réponse possible de l'exercice :

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
```

- # Création d'un data Set

```
y = np.array([8,29,40])
x = np.linspace(20,30,3)
```

Les calculs scientifiques dans python

Réponse de l'exercice suite :

- # Affichage de la création

```
plt.plot(x,y, 'o')  
plt.show()
```

- # Création de la fonction interpolation f en mode linéaire

```
f = interp1d(x, y, kind='linear')
```

Les calculs scientifiques dans python

Réponse possible de l'exercice suite :

- # résultats de la fonction interpolation f sur de nouvelles données

```
X_new = np.linspace(20,30,100)
```

- # Affichage de l'ensemble des valeurs

```
print(X_new)
```

Les calculs scientifiques dans python

Réponse possible de l'exercice fin :

- # visualisation avec matplotlib

```
plt.plot(x,y,'o', label = 'Données')
plt.plot(X_new, f(X_new), label = 'Linaire')
plt.legend(loc='best')
plt.show()
```

Les calculs scientifiques dans python

Interpolation quadratique et cubique en utilisant `scipy.interpolate` :

- Permet d'ajouter des valeurs aux valeurs initiales pour avoir plus de données
- Permet de suivre une équation polynomiale de degré 2 et/ou 3 entre les points

Exercice :

- Création d'un data Set de 2 tableaux, avec $(0, 10, 10)$ et `np.sin(-x**2/4.0)`
- Création des fonctions d'interpolation linéaire, quadratique et cubique avec `interp1d` de `scipy.interpolate` sur 100 valeurs
- Visualisation avec `matplotlib` du résultat

Les calculs scientifiques dans python

Réponse possible de l'exercice :

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
```

- # Création d'un data Set

```
x = np.linspace(0, 10, 10)
y = np.sin(-x**2/4.0)
```

Les calculs scientifiques dans python

Réponse possible de l'exercice suite :

- # création des fonctions interpolation f sous différents modes

```
f1 = interp1d(x, y)
f2 = interp1d(x, y, kind='quadratic')
f3 = interp1d(x, y, kind='cubic')
X_new = np.linspace(0, 10, 100)
```

Les calculs scientifiques dans python

Réponse possible de l'exercice fin :

- # Affichage de la création

```
plt.plot(x,y,'o',X_new,f1(X_new),'-', X_new, f2(X_new),'--', X_new, f3(X_new),'+')
plt.legend(['Données', 'Linaire', 'quadratique', 'cubique'], loc='best')
plt.show()
```

Les calculs scientifiques dans python

Calcul de l'intervalle de confiance à 95% :

- En statistiques, l'intervalle de confiance établit la marge d'erreur entre les données de l'échantillon et les données de la population
- Si votre population est grande, le travail se fait sur l'échantillon pour l'étendre aux résultats de l'ensemble
- Il y a risque d'erreur, d'où l'intervalle de confiance avec sa marge d'erreur
- L'intervalle de confiance se calcule par rapport à une moyenne et un écart type

Les calculs scientifiques dans python

Soit α le degré de confiance, σ l'écart type, n la taille de l'échantillon et $Z_{\frac{\alpha}{2}}$ le coefficient de confiance (ou valeur critique)

Loi normale

La marge d'erreur = valeur critique * erreur type : $Z_{\frac{\alpha}{2}} * \frac{\sigma}{\sqrt{n}}$

Les calculs scientifiques dans python

Fonction possible :

```
import numpy as np
from scipy.stats import t as student

def intervalleConfiance(data, alpha = .05 ):
    long = len(data)
    confiance = student.ppf( 1 - .5 * alpha, long-1 ) * np.sqrt( np.var(data) / long )
    return np.mean(data), np.mean(data)-confiance, np.mean(data) + confiance
```

Les calculs scientifiques dans python

Fonction possible :

```
import numpy as np
import scipy.stats as sst

def intervalle_Confiance(data, tolerance=0.95):
    long = len(data)
    moyenne, se = np.mean(data), sst.sem(data)
    confiance = se * sst.t.ppf((1 + tolerance) / 2., long-1)
    return moyenne, moyenne-confiance, moyenne+confiance
```

Les calculs scientifiques dans python

Fonction possible :

```
import numpy as np
import scipy.stats as st

data = range(10,20)

st.t.interval(0.95, len(data)-1, loc=np.mean(data), scale=st.sem(data))
```

Les modules d'analyses de données et de visualisation dans python

La librairie pandas (python Data Analysis Library), spécialisée dans l'analyse des données, offre une préparation pour la modélisation statistique :

```
import pandas as pd
```

Les données peuvent provenir des fichiers csv conditionnés :

```
dataframe = pd.read_csv("Ontime.csv")
```

L'objet produit par pandas est une matrice "data frame" :

```
dataframe.head()
```

Les modules d'analyses de données et de visualisation dans python

Pour la dimension : .shape

Pour les columns : .columns

Pour le type de données : .dtypes

Les "data frame" sont mutables :

```
dataframe['Date'] = pd.to_datetime('2021-03-' + dataframe['DAY_OF_MONTH'].apply(str))  
dataframe['day_name']=dataframe['Date'].dt.weekday  
  
dataframe.columns
```

Les modules d'analyses de données et de visualisation dans python

Les données peuvent se ranger avec un pivot du tableau :

```
data = dataframe.head().pivot(index='DISTANCE', columns='Date')  
data
```

Les modules d'analyses de données et de visualisation dans python

Les données peuvent provenir de fichiers textes conditionnés

```
import pandas as pd  
maladie = pd.read_csv('maladie.txt', sep=";", decimal='.')  
maladie.head()
```

Demandez à l'ordinateur de faire

```
data = d.values  
x = data[:, :-1]
```

```
y = data[:, -1]
```

Les modules d'analyses de données et de visualisation dans python

La librairie seaborn de python permet de créer des graphiques statistiques

Conçue avec matplotlib, elle intègre les structures de données de pandas

```
import seaborn as sns  
fig = sns.scatterplot(x="age", y="chd", data=maladie)
```

Les modules d'analyses de données et de visualisation dans python

Exercice manipulation et recherche d'informations :

- Etape 1 : Charger les librairies pandas et numpy
- Etape 2 : Charger le fichier CreditData.csv avec comme indice premier 'ID'
- Etape 3 : Examiner le contenu du fichier
- Etape 4 : Faire une instruction pour afficher une liste de toutes les femmes qui sont diplômées et dont son statut à prêt est 0

Les modules d'analyses de données et de visualisation dans python

Réponse possible aux étapes 1, 2, 3 et 4 :

```
import pandas as pd  
import numpy as np  
data=pd.read_csv('CreditData.csv',index_col='ID')
```

```
data.info()  
data.head()
```

```
data.loc[(data["Genre"]=="Femme") & (data["Education"]=="Diplome") &  
         (data["Statut_du_pret"]=="O"), ["Genre","Education","Statut_du_pret"]]
```

Les modules d'analyses de données et de visualisation dans python

Quelques fonctions pour compter et visualiser le nombres d'informations :

- .values_counts()

```
data[ 'Education' ].value_counts()
```

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x='Education', data = data)
plt.show()
```

Les modules d'analyses de données et de visualisation dans python

La fonction apply :

- Est une fonction pour manipuler les données et créer de nouvelles variables
- Retourne une valeur après l'inspection des lignes / colonnes d'un Data Frame en relation une fonction
- La fonction est une fonction par défaut ou définie par le concepteur
- Etape 5 : Avec la fonction apply et une fonction que vous devez créer, trouvez dans le fichier initial les valeurs manquantes de chaque ligne et colonne

Les modules d'analyses de données et de visualisation dans python

Réponse possible pour l'étape 5 :

```
# Fonction pour définir la non présence d'une valeur
def valeurManquante(x):
    return sum(x.isnull())

# Application de la fonction valeur Manquante pour chaque colonne:
print("Valeurs manquantes par colonne:")
print(data.apply(valeurManquante, axis=0))
# axis=0 pour appliquer la fonction sur chaque colonne

# Application de la fonction valeur Manquante pour chaque Ligne:
print("\nValeurs manquantes par ligne:")
print(data.apply(valeurManquante, axis=1).head())
# axis=1 pour appliquer la fonction sur chaque ligne
```

Les modules d'analyses de données et de visualisation dans python

Pour limiter les problèmes, il faut harmoniser les données et les compléter

- La fonction fillna() pour les valeurs Nan
- Les fonctions mean, mode, median pour mettre à jour les valeurs manquante

```
from scipy.stats import mode  
data[ 'Genre' ].mode()
```

- Si le résultat est un tableau de plusieurs possibilité, prendre le premier :

```
data[ 'Genre' ].mode()[0]
```

Les modules d'analyses de données et de visualisation dans python

```
data['Genre'].fillna(data['Genre'].mode().iloc[0], inplace=True)
data['Marie'].fillna(data['Marie'].mode().iloc[0], inplace=True)
data['Travailleur_independant'].fillna(data['Travailleur_independant'].mode().iloc[0], inplace=True)

# Teste des valeurs manquantes pour valider:
print(data.apply(valeurManquante, axis=0))
```

Les modules d'analyses de données et de visualisation dans python

pandas est utilisé pour créer des tableaux croisés dynamiques de style MS Excel

Dans le fichier, par exemple, il y a des valeurs manquantes dans le montant du prêt

Nous pouvons imputer le montant du prêt avec les valeurs moyennes des groupes Genre, Marie et Travailleur_indépendants :

```
valeur_groupe = data.pivot_table(values=["Montant_du_pret"],  
                                 index=["Genre", "Marie", "Travailleur_independant"],  
                                 aggfunc=np.mean)
```

Les modules d'analyses de données et de visualisation dans python

La librairie Statsmodels de python utilise pandas pour le stockage des données et la librairie patsy pour décrire les modèles par des formules, par convention :

- endog sont les variables à prédire (variables réponse)
- exog sont les variables références (variables explicatives)

Si maladie est une Data frame pandas avec les colonnes *tobacco*, *aae* et *chd* :

```
import patsy
```

```
y, X = patsy.dmatrices('tobacco ~ age + chd ', data = maladie, return_type = 'dataframe')
```

y est la Data frame de la réponse et *X* la Data frame des variables indépendantes, avec une colonne supplémentaire Intercept qui est à 1

Les modules d'analyses de données et de visualisation dans python

y est la Data frame de la réponse : *tobacco*

X la Data frame des variables indépendantes,
avec une colonne supplémentaire Intercept qui
est à 1 : *age* + *chd*

tobacco		Intercept	age	chd	
0	12.00	0	1.0	52.0	1.0
1	0.01	1	1.0	63.0	1.0
2	0.08	2	1.0	46.0	0.0
3	7.50	3	1.0	58.0	1.0
4	13.60	4	1.0	49.0	1.0
...
457	0.40	457	1.0	58.0	0.0
458	4.20	458	1.0	52.0	1.0
459	3.00	459	1.0	55.0	0.0
460	5.40	460	1.0	40.0	0.0
461	0.00	461	1.0	46.0	1.0

Les modules d'analyses de données et de visualisation dans python

Il est possible d'ajuster le modèle :

```
import statsmodels.api
```

```
model = statsmodels.api.OLS(y, X)
```

```
result = model.fit()
```

```
result.summary()
```

OLS Regression Results

Dep. Variable:	tobacco	R-squared:	0.223			
Model:	OLS	Adj. R-squared:	0.220			
Method:	Least Squares	F-statistic:	65.85			
Date:	Mon, 22 Feb 2021	Prob (F-statistic):	7.19e-26			
Time:	16:01:09	Log-Likelihood:	-1301.1			
No. Observations:	462	AIC:	2608.			
Df Residuals:	459	BIC:	2621.			
Df Model:	2					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	-2.1691	0.590	-3.677	0.000	-3.328	-1.010
age	0.1236	0.014	8.868	0.000	0.096	0.151
chd	1.4758	0.428	3.451	0.001	0.635	2.316
Omnibus:	189.568	Durbin-Watson:	2.042			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	902.518			
Skew:	1.764	Prob(JB):	1.05e-196			
Kurtosis:	8.868	Cond. No.	143.			

Les modules d'analyses de données et de visualisation dans python

```
import statsmodels.formula.api
```

```
model = statsmodels.formula.api.ols('tobacco ~ age + chd', data = maladie)
```

```
result = model.fit()
```

Les modules d'analyses de données et de visualisation dans python

Les résultats comportent le modèle et le modèle comporte les données :

```
result.model
```

```
'tobacco'
```

```
result.model.data
```

```
<statsmodels.base.data.PandasData at 0x117a6781d90>
```

```
result.model.data.ynames
```

```
'tobacco'
```

```
result.model.data.exog
```

```
array([[ 1., 52., 1.],  
       [ 1., 63., 1.],  
       [ 1., 46., 0.],  
       ...,  
       [ 1., 55., 0.],  
       [ 1., 40., 0.],  
       [ 1., 46., 1.]])
```

```
result.model.data.xnames
```

```
['Intercept', 'age', 'chd']
```

Les modules d'analyses de données et de visualisation dans python

Si la donnée n'a pas d'étiquette explicite, il est possible de modifier les noms :

```
import pandas

noms = ['seismic','seismoacoustic','shift','genergy','gpuls','gdenergy','gdgpuls','ghazard',
        'nbumps','nbumps2','nbumps3','nbumps4','nbumps5','nbumps6','nbumps7','nbumps89','energy','maxenergy','class']

dataframe1 = pandas.read_csv('seismic.csv')

dataframe2 = pandas.read_csv('seismic.csv', names = noms)
```

Les modules de Machine Learning et de Data Set dans python

Le package Scikit-learn dans python est une boite d'outils efficace pour créer des Machines Learning

Scikit-learn peut être utilisé pour faire de l'apprentissage statistique, des classifications non supervisées (classifier des données qui n'ont pas de catégorie prédéterminées), comme l'analyse de partitionnement (clustering)

Le package détient des Data Set prêt à être utilisé pour former des Machines Learning

Les modules de Machine Learning et de Data Set dans python

Exemple de données disponibles dans sklearn :

```
from sklearn import datasets
iris = datasets.load_iris()
boston = datasets.load_boston()
digits = datasets.load_digits()

iris.feature_names, boston.keys(), digits.target
```

Les modules de Machine Learning et de Data Set dans python

Exemple pour récupérer des données :

```
import numpy as np
from sklearn import datasets
iris_X, iris_y = datasets.load_iris(return_X_y=True)
np.unique(iris_y), np.unique(iris_X)
```

Les modules de Machine Learning et de Data Set dans python

Exemple pour récupérer et afficher des données :

```
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
X, y = make_blobs(n_samples=300, centers=2, cluster_std=3.6)
plt.scatter(X[:,0],X[:,1])
plt.scatter(X[:,0],X[:,1], c=y)
```

Les modules de Machine Learning et de Data Set dans python

Pour affecter des données avec un pourcentage :

```
import pandas as pd
import sklearn.model_selection as sms
d = pd.read_csv("maladie.txt",sep=';')
data = d.values
X = data[:, :-1]
y = data[:, -1]

Xtrain, Xtest, ytrain, ytest = sms.train_test_split(X,y, train_size=0.7, random_state = 0)
print(Xtrain.shape,Xtest.shape,ytrain.shape,ytest.shape,)

Xtrain, Xtest, ytrain, ytest = sms.train_test_split(X,y, test_size=0.3, random_state = 0)
print(Xtrain.shape,Xtest.shape,ytrain.shape,ytest.shape,)
```

Les modules de Machine Learning et de Data Set dans python

Création d'un data Set :

```
import numpy as np
import matplotlib.pyplot as plt

# Graine aléatoire de numpy pour obtenir toujours les mêmes résultats
np.random.seed(1)

# Génération de 2 matrices
a = np.array([[1.05, -0.35], [0.35, 1.05]])
b = np.array([[3.5, 0], [0, 2]])

# Génération des données de la classe 1
classe1 = (np.random.randn(100,2)).dot(a).dot(b)

# Génération des données de la classe 2
classe2_1 = np.random.randn(25,2)+[-10, 2]
classe2_2 = np.random.randn(25,2)+[-7, -2]
classe2_3 = np.random.randn(25,2)+[-2, -6]
classe2_4 = np.random.randn(25,2)+[5, -7]

# Création du Data Set
dataSet = np.concatenate((classe1, classe2_1, classe2_2, classe2_3, classe2_4))

# Génération des étiquettes de classe
etiq_1 = np.ones(100, dtype=int)
etiq_2 = np.zeros(100, dtype=int)
étiquettes = np.concatenate((etiq_1, etiq_2))
```

Les modules de Machine Learning et de Data Set dans python

Visualisation du data Set :

```
# Les échantillons du premier groupe sont en jaune (*y*), ceux du deuxième groupe en gris (*grey*)
cmp = np.array(['y','grey'])
plt.figure(figsize=(10,6))

# La couleur se fait en fonction des classe 1 et 2
plt.scatter(dataSet[:,0],dataSet[:,1], c=cmp[etiquettes], s=50)
```

Les modules de Machine Learning et de Data Set dans python

Visualisation du data Set de test et d'entraînement :

```
from sklearn.model_selection import train_test_split

plt.figure(figsize=(10,6))
X_train1, X_test1, y_train1, y_test1 = train_test_split(dataSet, etiquettes, test_size=0.33)

plt.scatter(X_train1[:,0],X_train1[:,1],c=cmp[y_train1],s=50)
plt.scatter(X_test1[:,0],X_test1[:,1],c='none',s=50,edgecolors=cmp[y_test1])
```

Le traitement d'images comme base de données numériques

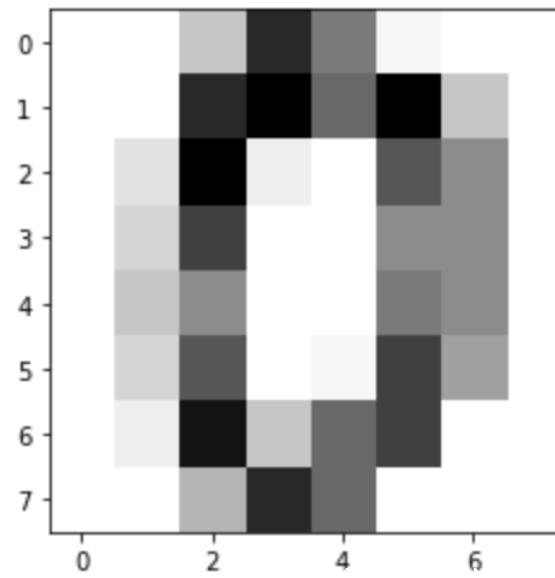
Exemple :

```
from sklearn import datasets
import matplotlib.pyplot as plt

digits = datasets.load_digits()

plt.imshow(digits.images[0],cmap='Greys')
```

<matplotlib.image.AxesImage at 0x15b07870df0>



Le traitement d'images comme base de données numériques

```
from scipy import misc
import numpy as np
import matplotlib.pyplot as plt
img = misc.face()
plt.imshow(img)
plt.show()
type(img)
```



Le traitement d'images comme base de données numériques

```
from scipy import misc
import numpy as np
import matplotlib.pyplot as plt
img = misc.face(gray=True)
plt.imshow(img, cmap = "gray")
plt.show()
img.shape
```



Le traitement d'images comme base de données numériques

Que fait ce code ?

```
h = img.shape[0]
v = img.shape[1]
zoom_img = img[h//4 : -h//4, h//4 : -h//4]
plt.imshow(zoom_img, cmap=plt.cm.OrRd)
plt.show()
```

Le traitement d'images comme base de données numériques

Que fait ce code ?

```
from scipy import misc
import numpy as np
import matplotlib.pyplot as plt
img = misc.face(gray=True)
h = img.shape[0]
v = img.shape[1]
zoom_img = img[h//4 : -h//4, h//4 : -h//4]
zoom_img[zoom_img > 200] = 255
plt.imshow(zoom_img, cmap=plt.cm.OrRd)
plt.show()
```

Le traitement d'images comme base de données numériques

D'autres possibilités de charger des images :

```
from PIL import Image
import numpy as np
image = Image.open('fichier.png')

# Transformation de l'image en tableau numpy
img = np.array(image)
plt.axis('off') # Suppression des cadres gradués
plt.imshow(image)
```

Le traitement d'images comme base de données numériques

Le traitement de l'image pour déterminer les fréquences des données :

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture

# Charger l'image
img = plt.imread("spam.jpg",0)

# Gestion en tableau d'un échantillon à partir de l'image
image = img.ravel() # Mise à plat de l'image

# Suppression des pixels de fond (intensités 0 et 1)
image = image[image != 0]
image = image[image != 1]
```

Le traitement d'images comme base de données numériques

Le traitement de l'image pour déterminer les fréquences des données, suite :

```
# Formation de La gaussienne : Gauss
Gauss = GaussianMixture(n_components = 2)
Gauss = Gauss.fit(X=np.expand_dims(image,1))

# Evaluation de La gaussienne Gaus_M
Gauss_x = np.linspace(0,254,5000)

# Calculer Les probabilités Logarithmiques pondérées pour chaque échantillon
Gauss_y = np.exp(Gauss.score_samples(Gauss_x.reshape(-1,1)))
```

Le traitement d'images comme base de données numériques

Le traitement de l'image pour déterminer les fréquences des données, fin :

```
# Histogrammes de tracé et courbes Gaussiennes

fig, ax = plt.subplots(figsize=[10, 5])
ax.hist(image, bins=20, density=True)
ax.plot(Gauss_x, Gauss_y, color="red", lw=5, label="Gauss")

ax.set_xlabel("Intensité des pixels")
ax.set_ylabel("Fréquence")
plt.legend(loc = 'upper left')
plt.title('Courbe gaussienne en fonction de la photo')

plt.show()
```

Modèles D'évaluation

INTRODUCTION : CALCULS DES TAUX
D'ERREUR SUR LES MODÈLE
D'APPRENTISSAGE

Dr. Chaabani Yasmine
yasmin.chaabani@isgb.ucar.tn

Sommaire

Présentation

Le principe d'évaluation

Matrice de corrélation

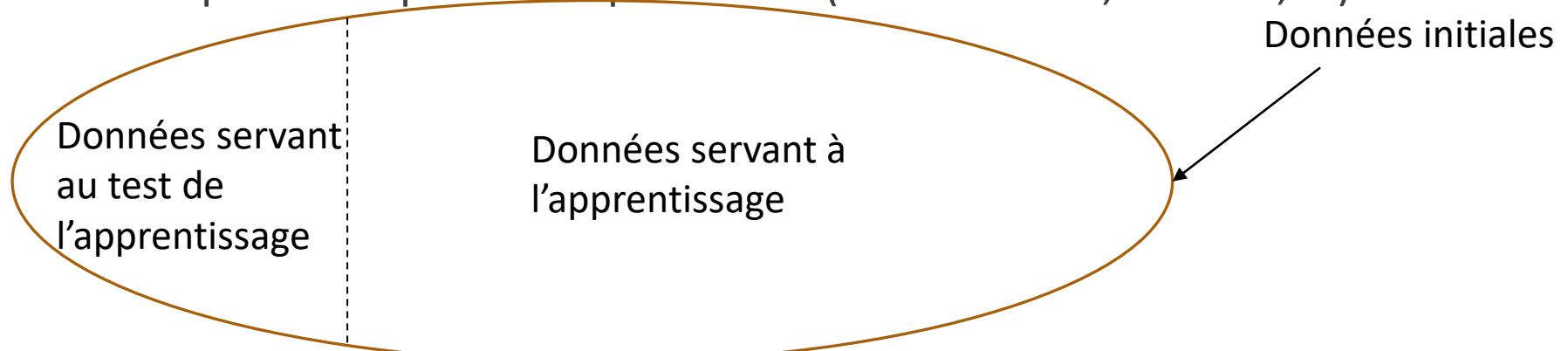
Matrice de confusion

Présentation

Pour valider (ou infirmer) un résultat, il faut garantir la qualité du processus d'apprentissage

Approches :

- Utiliser une partie des données pour apprendre et à se servir des autres données pour tester le résultat
- Différentes mesures pour comparer des processus (taux d'erreur, F-score, ...)

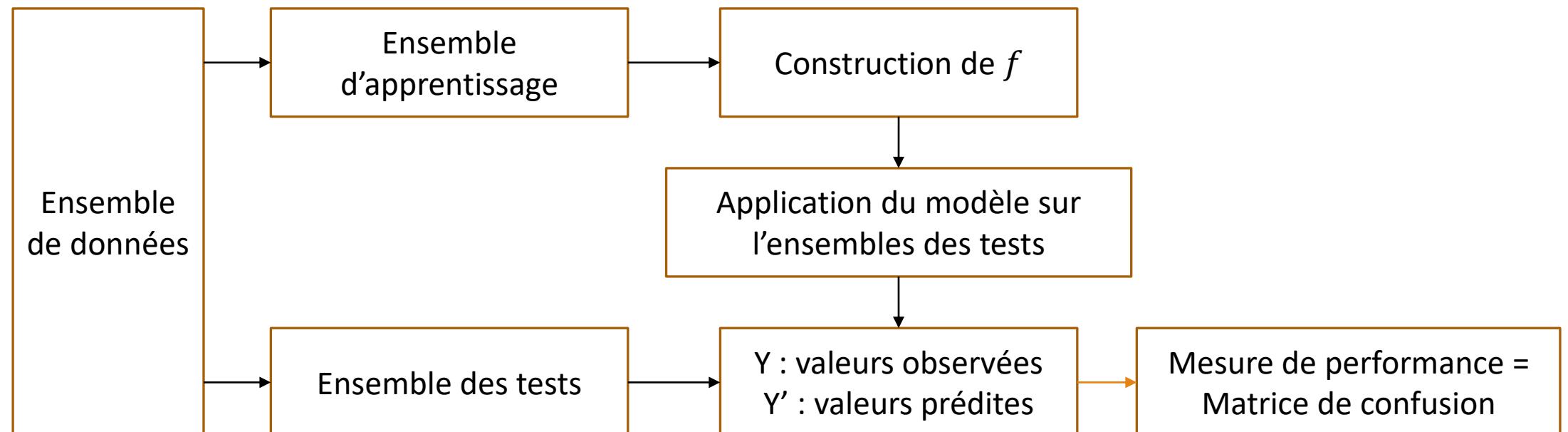


Présentation

Soit y : variable cible

x_1, \dots, x_n : variables explicatives

Soit f une fonction pour établir la relation $y = f(x_1, \dots, x_n) + \varepsilon$



Présentation

Evaluation de la performance d'un modèle = mesure des erreurs entre prédictions et valeurs réelles :

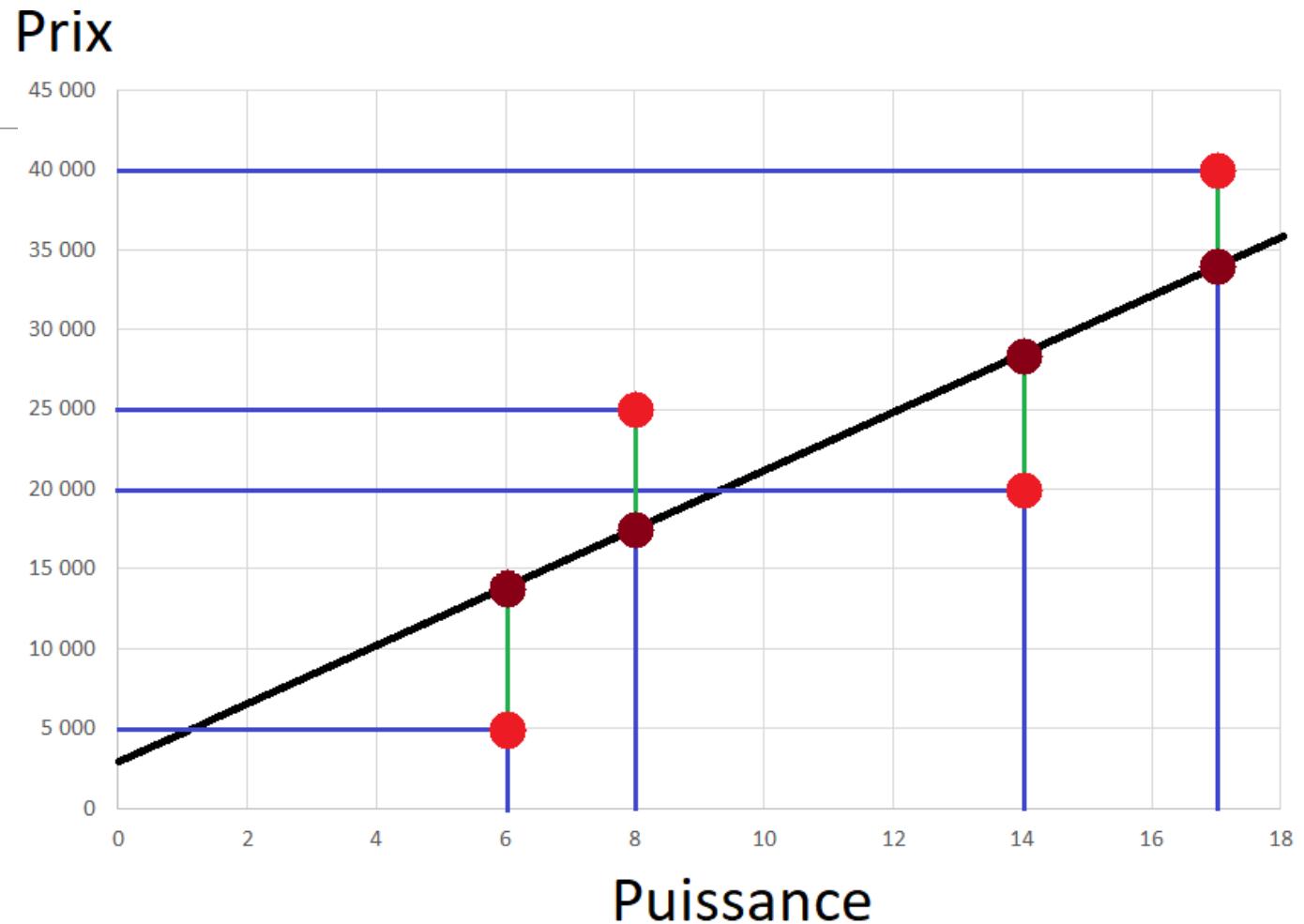
- MSE (Mean Squared Error)
- RMSE (Root Mean Squared Error)
- Men AE (Mean Absolue Error)
- Median AE (Median Absolue Error)
- R2 : coefficient de détermination
- Validation croisée (Cross Validation)

Présentation

Prix réel

Prix prédit

Erreur = Prix réel - Prix prédit



Présentation

2 options de calculs :

- Erreur absolue = $|Prix réel - Prix prédit|$
- Erreur quadratique = $(Prix réel - Prix prédit)^2$

Pour un ensemble de n données :

- $MSE = \frac{1}{n} \sum (Prix réel - Prix prédit)^2$
- $RMSE = \sqrt{\frac{1}{n} \sum (Prix réel - Prix prédit)^2}$
- Mean AE = $\frac{1}{n} \sum |Prix réel - Prix prédit|$
- Median AE = median(|Prix réel - Prix prédit|)

Présentation

Exercice avec le tableau :

Prix Réel	180	190	1050	1150
Prix Prédit	200	250	1050	1150

- Rechercher en `sklearn.metrics` et `numpy` les différents exemples d'erreur :
 - MSE (Mean Squared Error)
 - RMSE (Root Mean Squared Error)
 - Men AE (Mean Absolute Error)
 - Median AE (Median Absolute Error)

Présentation

Réponse possible à l'exercice :

```
import numpy as np
from sklearn.metrics import *
prix_reel = np.array([180, 190, 1050, 1150])
prix_predit = np.array([200, 250, 1050, 1150])
print("Mean AE : ", mean_absolute_error(prix_reel,prix_predit))
print("Median AE : ", median_absolute_error(prix_reel,prix_predit))
print("MSE : ", mean_squared_error(prix_reel,prix_predit))
print("RMSE : ", np.sqrt(mean_squared_error(prix_reel,prix_predit)))
```

Présentation

Quand utiliser la AE (Absolute error) et la SE (Squared Error) ? :

- AE → Erreurs avec des modèles linéaires
- SE → Erreurs avec des modèles exponentiels

Quand utiliser la MSE plutôt que la MAE ?

MSE → vous accordez une **grande importance aux grandes erreurs.**

MAE → l'importance d'une erreur est **linéaire** avec son amplitude. Si le Dataset contient des valeurs **aberrantes (outliers)**.

Présentation

Le coefficient de détermination évalue la performance du modèle au niveau des variations présents dans les données :

$$\circ R^2 = 1 - \frac{\sum (prix_{real} - prix_{predict})^2}{\sum (prix_{real} - \overline{prix_{real}})^2} = 1 - \frac{\sum \text{erreur quadratique}}{\sum \text{variance}}$$

- Utile pour tous les estimateurs de régression : score()
 - Sklearn.linear_model.LinearRegression
 - Sklearn.neighbors.KNeighborsRegression
 - Sklearn.svm.SVR

$$R^2 = 1 - \frac{\text{erreurs}}{\text{variance}}$$

erreurs << variance

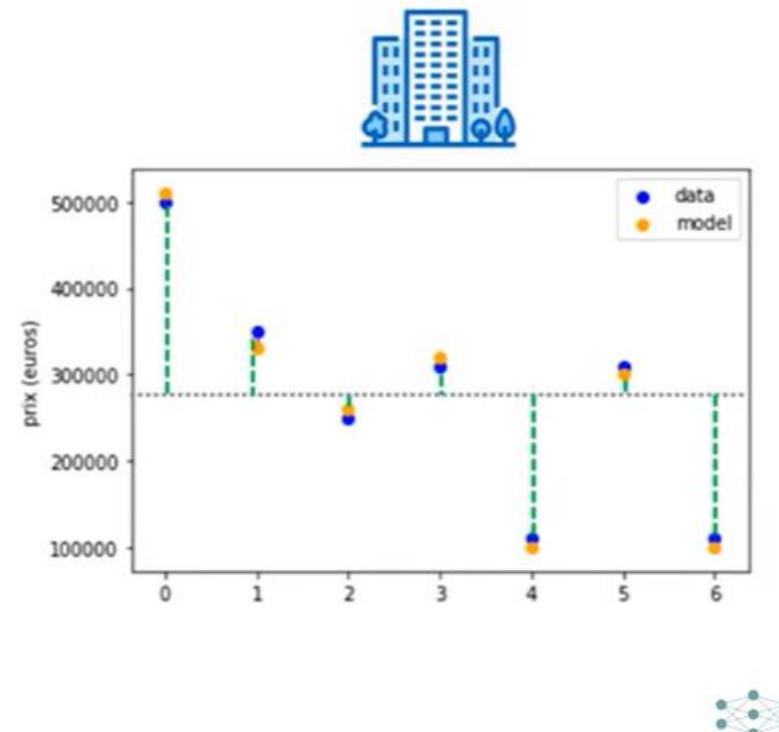
Coefficient de détermination R2

$$R^2 = 1 - \frac{\sum (y_{vrai} - y_{pred})^2}{\sum (y_{vrai} - \bar{y}_{vrai})^2}$$

1000
100000

$$R^2 = 1 - 0.01$$

$$\boxed{R^2 = 0.99}$$



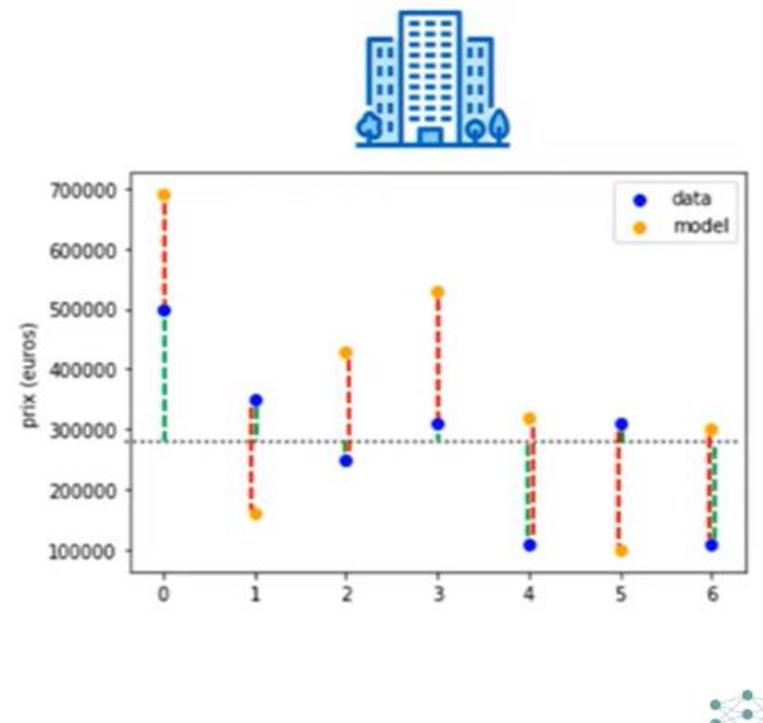
Coefficient de détermination R2

$$R^2 = 1 - \frac{\sum (y_{vrai} - y_{pred})^2}{\sum (y_{vrai} - \bar{y}_{vrai})^2}$$

200000
100000

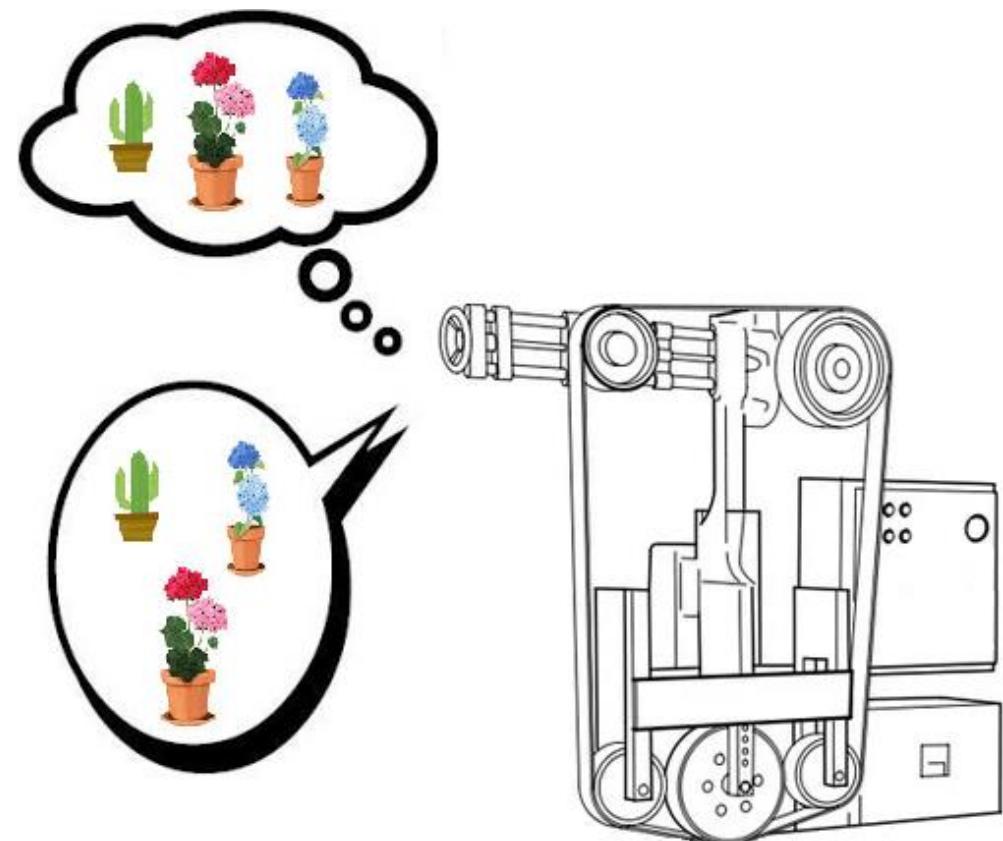
$$R^2 = 1 - 2$$

$$\boxed{R^2 = -1}$$

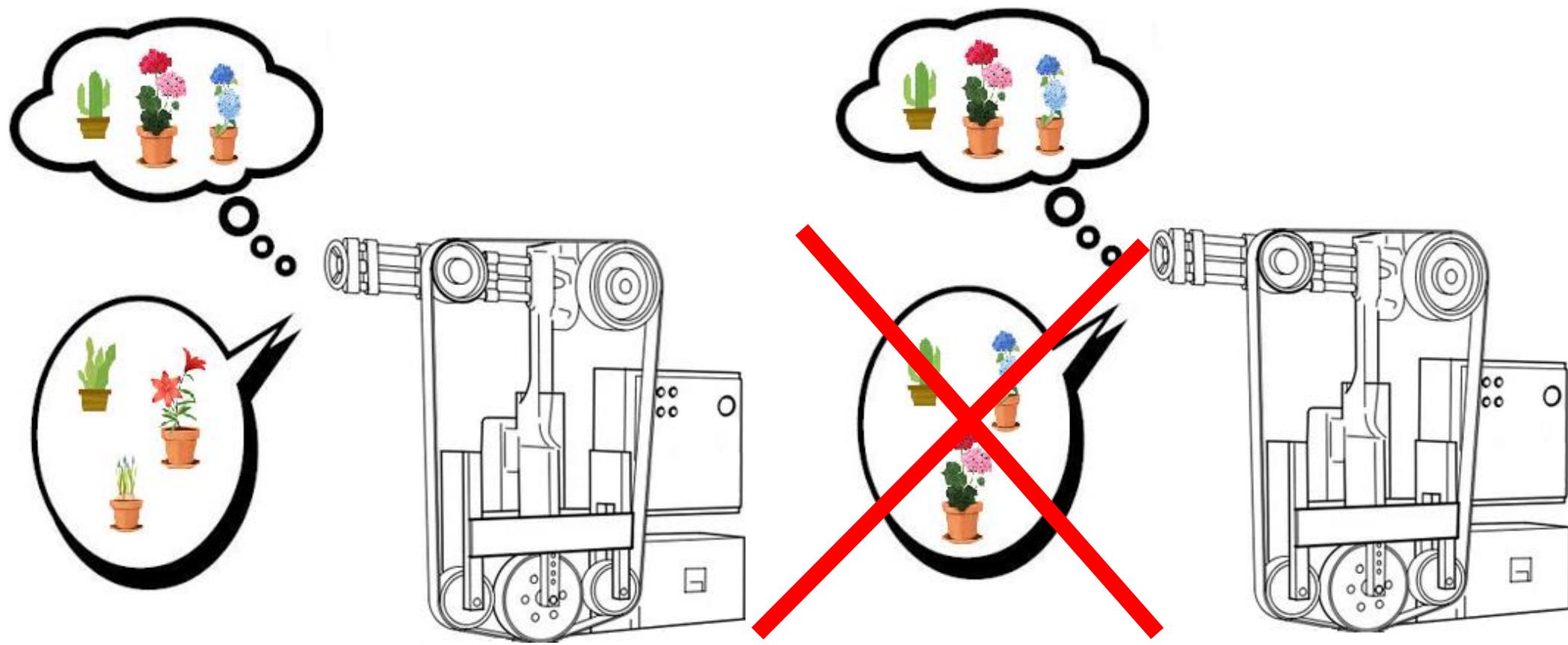


Le principe d'évaluation

- Jamais évaluer la performance d'un modèle avec les données qui ont servies pour faire l'apprentissage
- Il faut tester des données que la machine n'a jamais vues



Le principe d'évaluation



Le principe d'évaluation

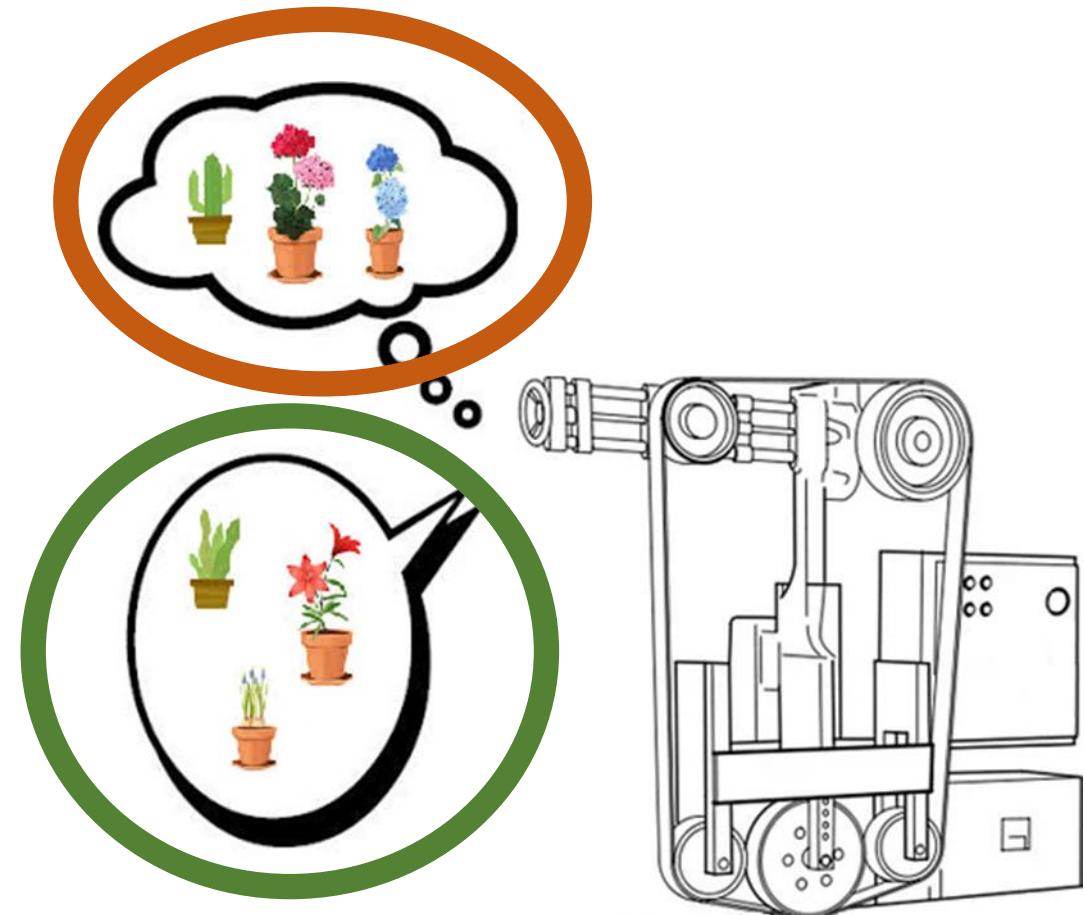
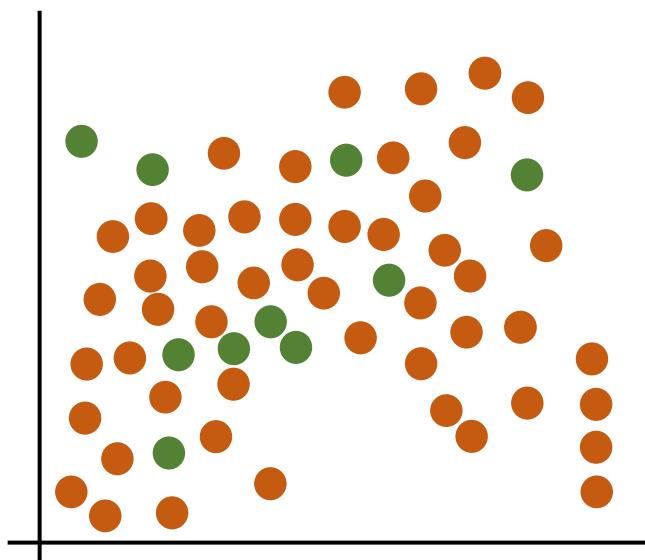
Données

- Nous prenons les données disponibles dans sklearn comme iris :

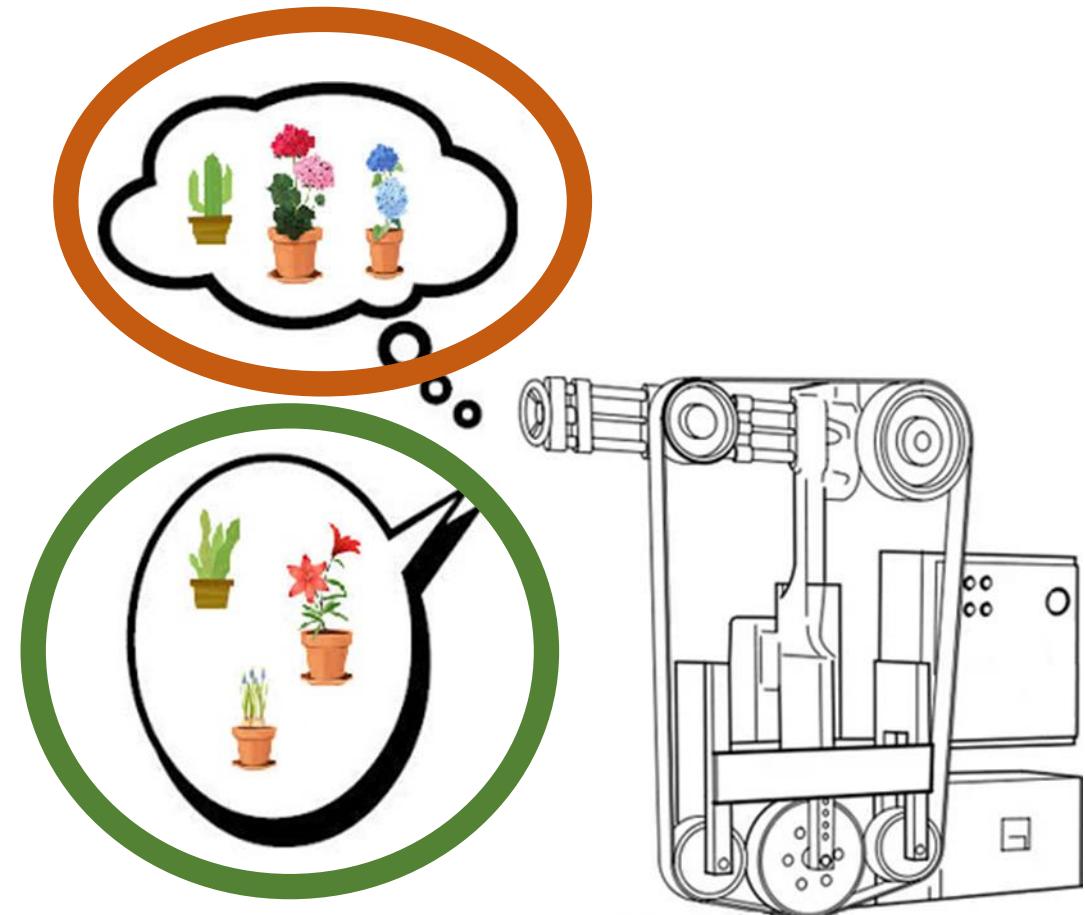
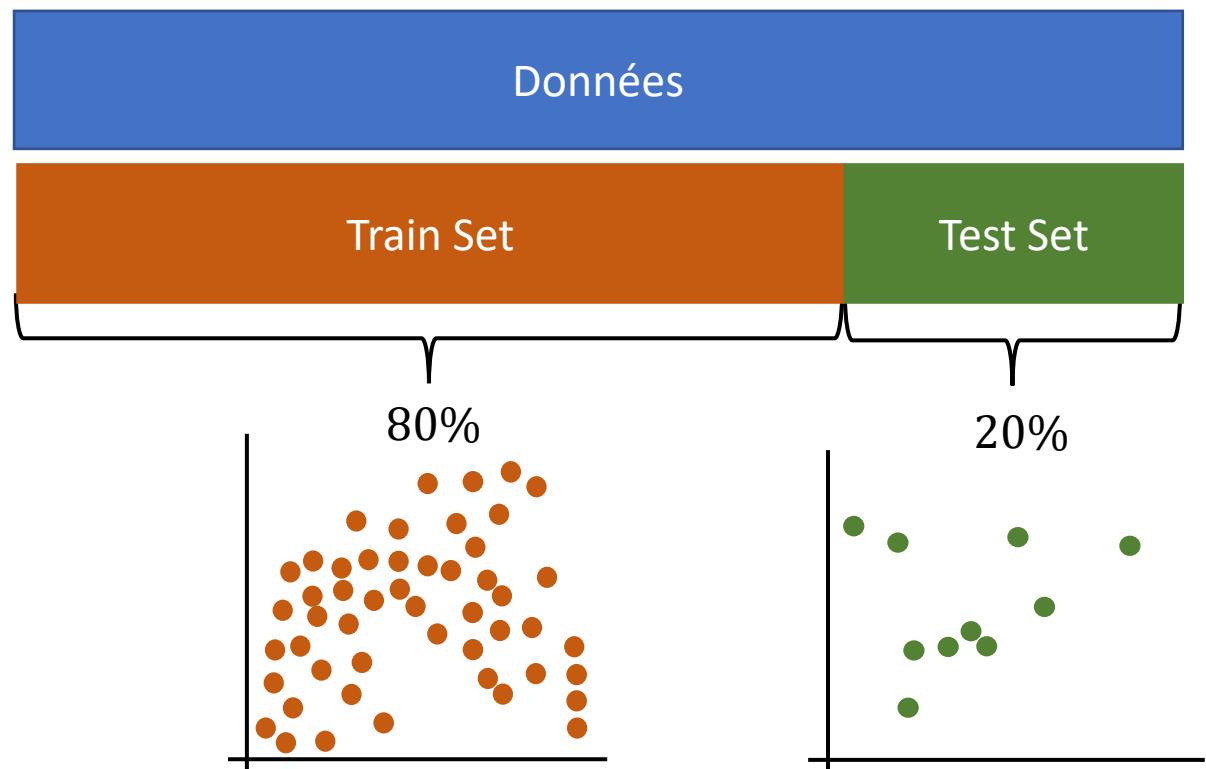
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

iris = load_iris()
```

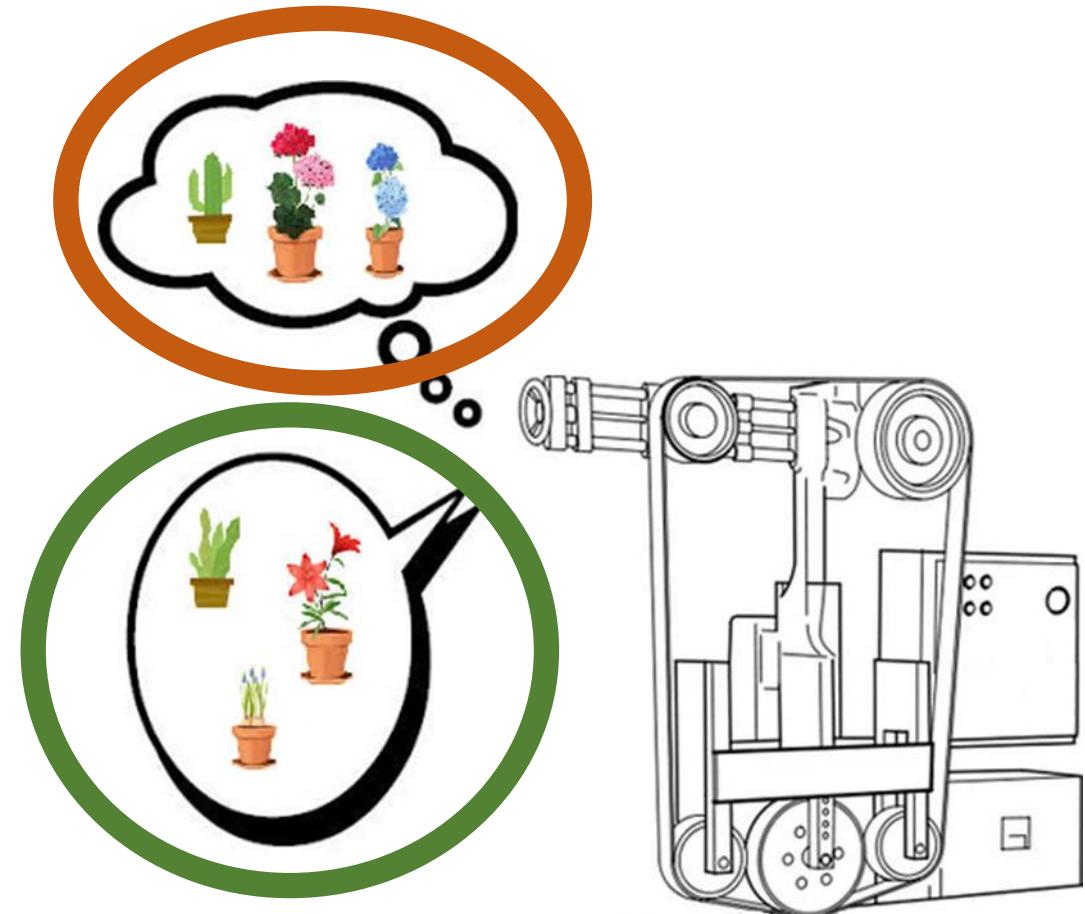
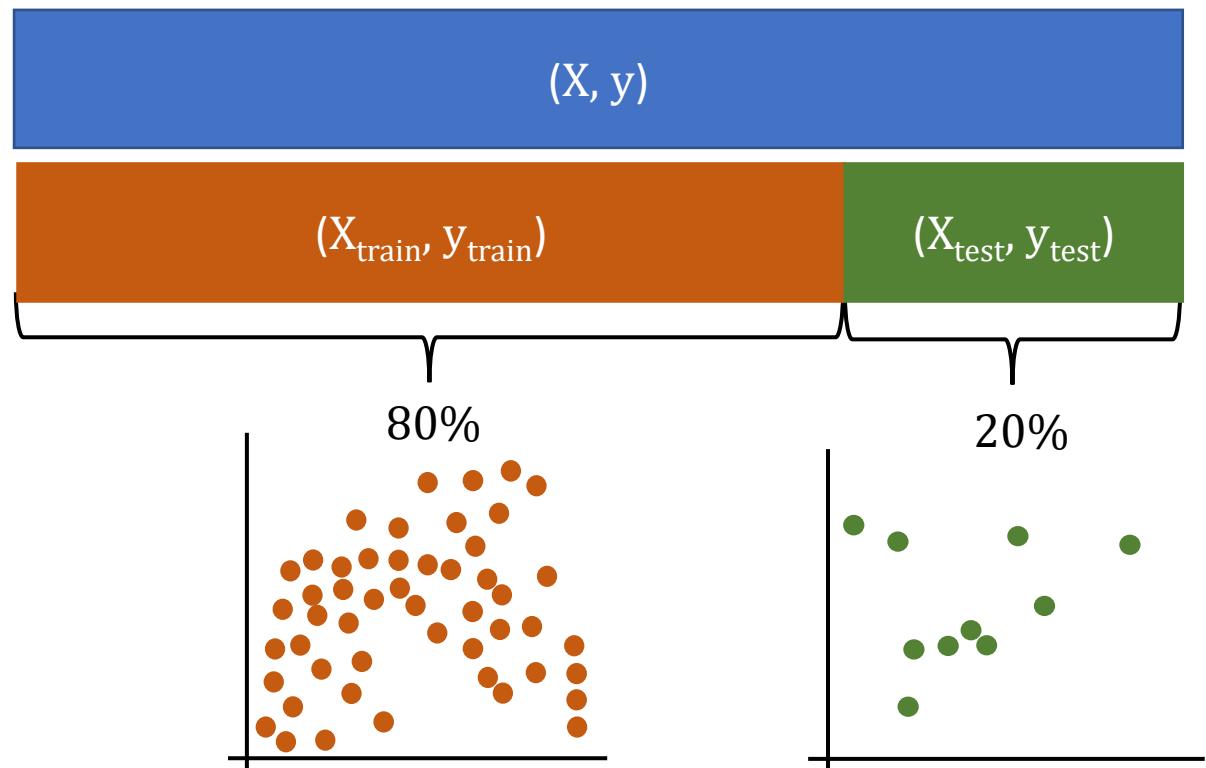
Le principe d'évaluation



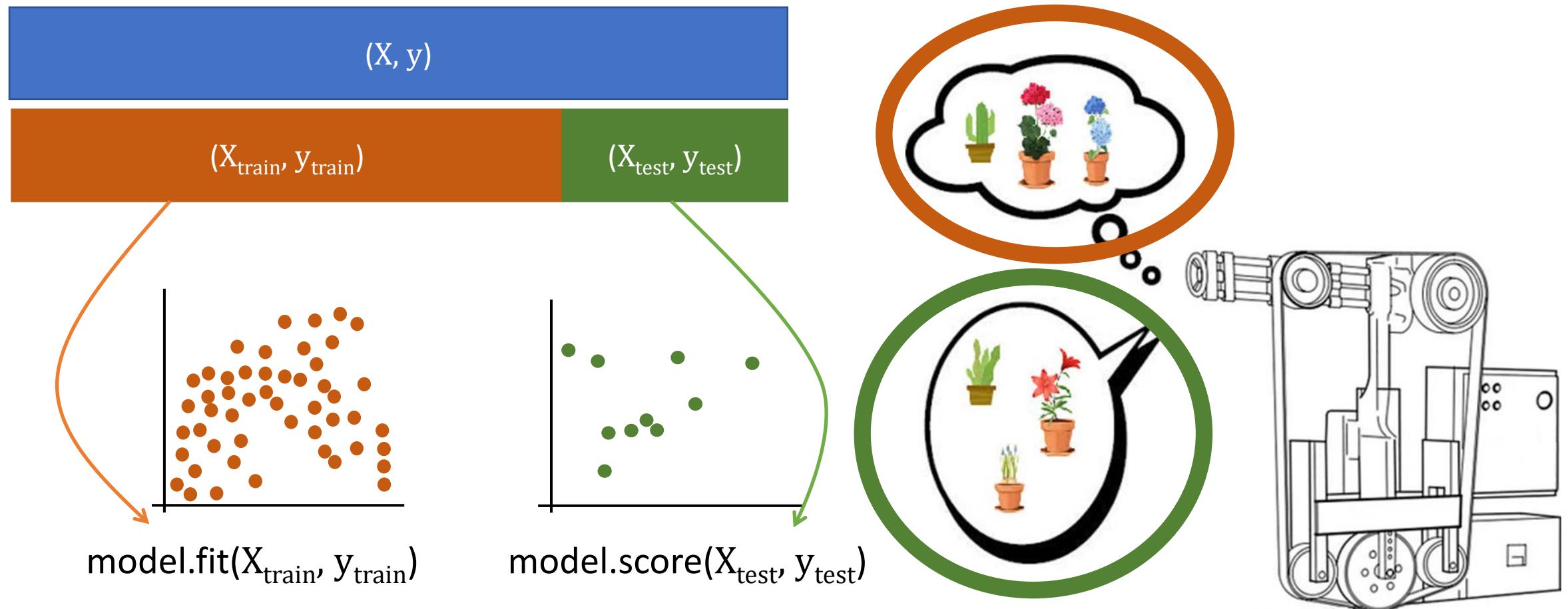
Le principe d'évaluation



Le principe d'évaluation

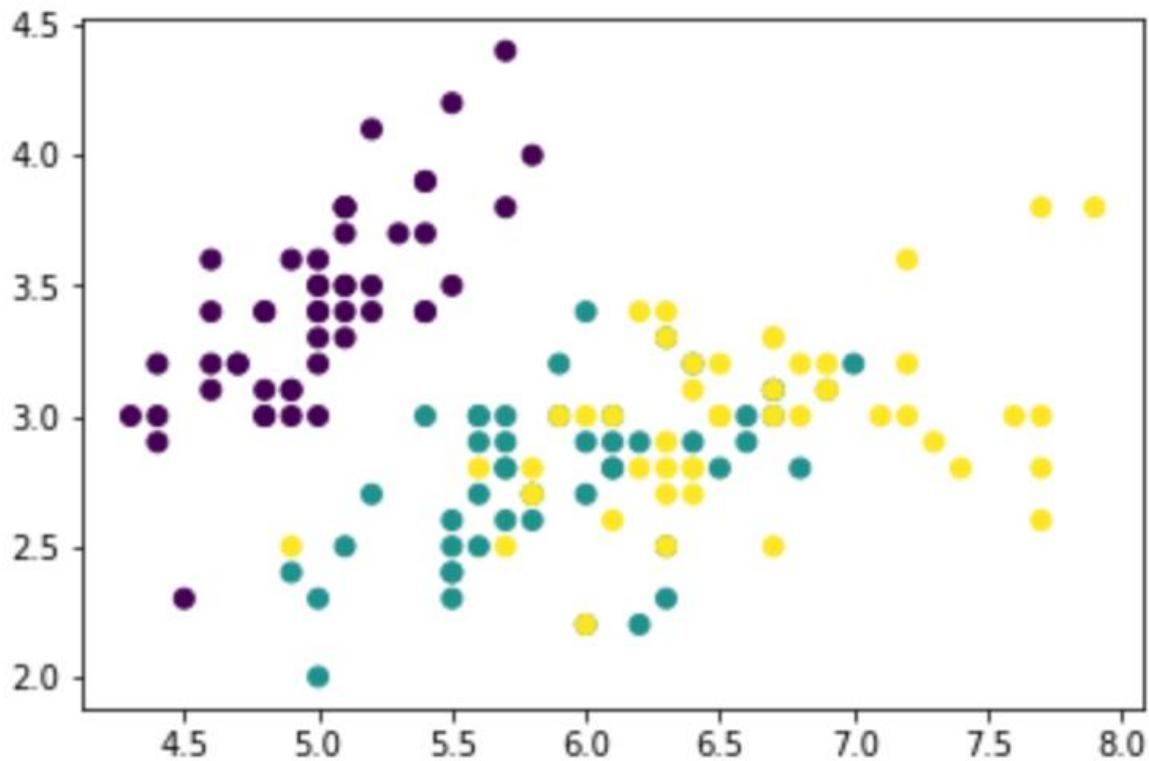


Le principe d'évaluation



Le principe d'évaluation

```
X = iris.data  
y = iris.target  
  
plt.scatter(X[:,0], X[:,1], c = y)
```



Le principe d'évaluation

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state = 5)

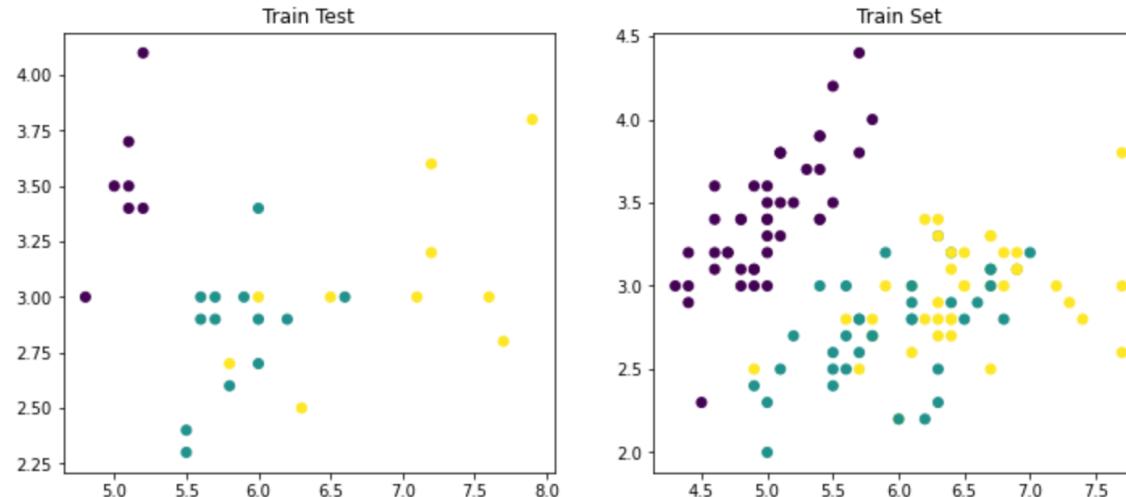
print('Data set', X.shape)
print('Train set', X_train.shape)
print('Test set', X_test.shape)
```

Data set (150, 4)
Train set (120, 4)
Test set (30, 4)

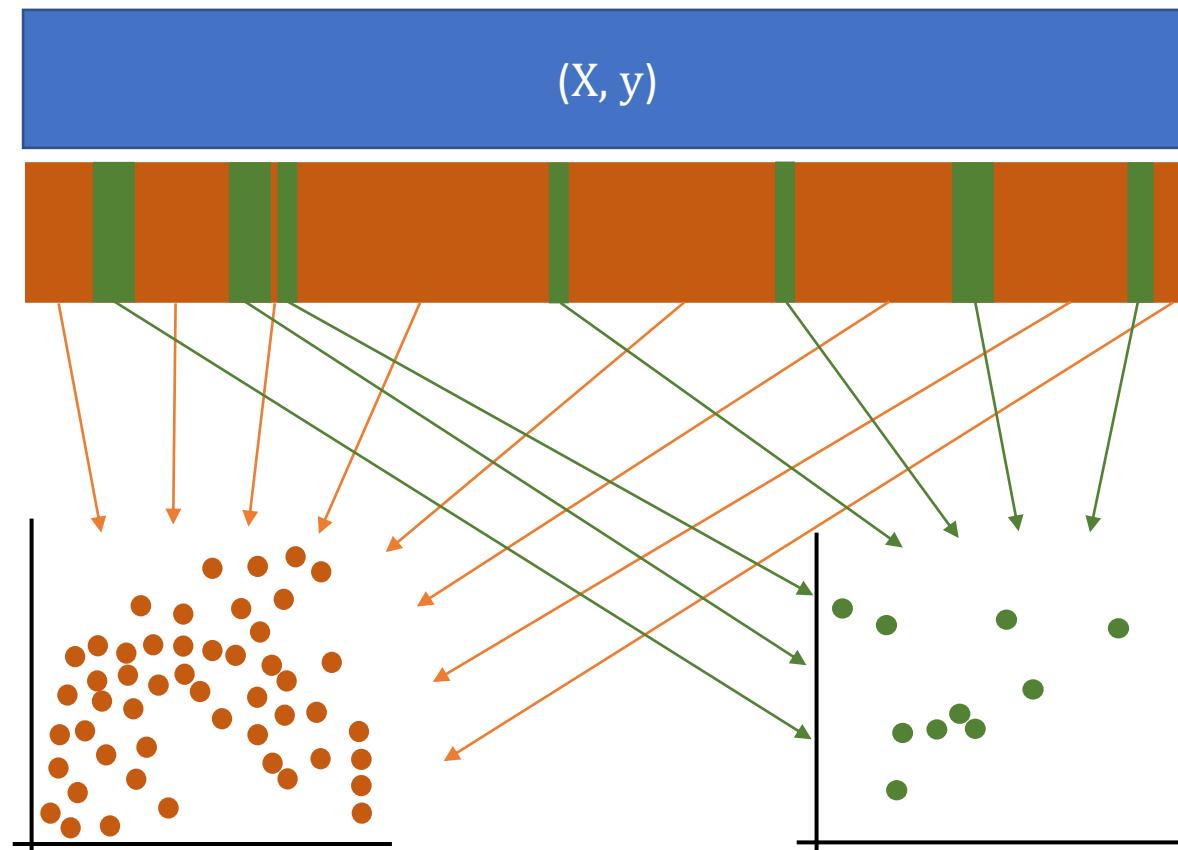
Le principe d'évaluation

```
plt.figure(figsize=(12,5))
plt.subplot(122)
plt.scatter(X_train[:,0],X_train[:,1], c=y_train)
plt.title('Train Set')
plt.subplot(121)
plt.scatter(X_test[:,0],X_test[:,1], c=y_test)
plt.title('Train Test')

Text(0.5, 1.0, 'Train Test')
```



Le principe d'évaluation



Le principe d'évaluation

- Phase de l'apprentissage :

```
from sklearn.neighbors import KNeighborsClassifier  
  
model = KNeighborsClassifier(n_neighbors=1)  
model.fit(X_train, y_train)  
print('Train score : ', model.score(X_train, y_train))
```

- Train score : 1.0 : veut-il dire que le modèle va réussir à 100 % de prédiction ?
- Il faut faire le test sur des données pas encore vues par le modèle :

```
print('Train score : ', model.score(X_test, y_test))
```

Le principe d'évaluation

- Entrainer 
- Evaluer 
- Améliorer le modèle : 
 - Comment améliorer les hyperparamètres ?
 - Comment avoir de meilleurs résultats ?
 - Quelles sont les limites à l'amélioration ?

Matrice de confusion

Le principe d'évaluation

- Matrice de confusion pour l'évaluation de la qualité d'un modèle de classification
- Montre les erreurs de classement

```
from sklearn.metrics import confusion_matrix  
confusion_matrix(y_test, m.predict(X_test))
```

- La matrice de confusion graphiquement pour une interprétation :

```
from sklearn.metrics import plot_confusion_matrix  
  
model = grid.best_estimator_  
model.fit(X_train, y_train)  
  
plot_confusion_matrix(m, X_test, y_test, cmap='magma_r')
```

Le principe d'évaluation

- Les limites des data Set
 - Evolution du modèle en fonction du nombre de données ?
 - Combien de données avoir un modèle :

```
from sklearn.model_selection import learning_curve

n , train_score, validation_score = learning_curve(m,X_train, y_train,
                                                    train_sizes= np.linspace(0.1, 1.0, 50), cv = 5)

plt.plot(n, train_score.mean(axis=1), label = 'train')
plt.plot(n, validation_score.mean(axis=1), label= 'validation')

plt.legend(loc = 'best')
```

Le principe d'évaluation

- La démarche à suivre :
 - Diviser le data Set avec `train_test_split`
 - Faire de l'optimisation avec GridSearch pour trouver les meilleurs hyperparamètres en utilisant cross validation avec une bonne stratégie de découpage
 - Bien évaluer le modèle avec les métriques
 - Utiliser les courbes d'apprentissage et d'évaluation pour éviter l'overfitting et savoir si le modèle peut encore s'améliorer

Matrice de confusion

- La matrice de confusion (matrice carrée) résume la réussite des prédictions d'un modèle de classification
- Corrélation entre les étiquettes et les classifications d'un modèle
- Les matrices de confusion sont calculées en utilisant les prédictions d'un modèle sur un ensemble de données
- Une matrice de confusion permet de comprendre les forces et les faiblesses d'un modèle

Matrice de confusion

- Permet de comparer deux modèles alternatifs pour comprendre lequel est le meilleur pour votre application
- Une matrice de confusion est calculée en utilisant les prédictions d'un modèle sur un ensemble de données

Matrice de confusion

- Exemple :

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.metrics import plot_confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

X, y = make_classification(random_state=0)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

clf = SVC(random_state=0)
clf.fit(X_train, y_train)
SVC(random_state=0)

plot_confusion_matrix(clf, X_test, y_test)
plt.show()

plot_confusion_matrix(clf, X_test, y_test, cmap='Greys_r')
```

Matrice de confusion

- Dans la terminologie de l'apprentissage supervisé, la matrice de confusion est un outil pour mesurer la qualité d'un système de classification
- Par exemple :
 - Nous avons un modèle de classification binaire pour classer des prototypes en Classe_1 et Classe_2
 - Nous souhaitons savoir combien de prototypes seront faussement considérés en Classe_1 et combien en Classe_2

Matrice de confusion

- Pour toutes les matrices de confusion :
 - Un axe pour les prédictions du modèle
 - Un axe pour les étiquettes réelles de la base de données

		Valeurs de prédiction	
		Predicted Negative	Predicted Positive
Valeurs Réelles	Real Negative	VRAI NEGATIF	FAUX POSITIF
	Real Positive	FAUX NEGATIF	VRAI POSITIF

Matrice de confusion

- Permet de comparer deux modèles alternatifs pour comprendre lequel est le meilleur pour votre application
- Une matrice de confusion est calculée en utilisant les prédictions d'un modèle sur un ensemble de données

Matrice de confusion

- Exemple pour une matrice de confusion :

```
from sklearn.metrics import confusion_matrix
y_reel = [1, 1, 0, 2, 0, 1, 0, 2, 2]
y_predict = [0, 1, 0, 1, 0, 1, 1, 1, 2]
confusion_matrix(y_reel, y_predict)

array([[2, 1, 0],
       [1, 2, 0],
       [0, 2, 1]], dtype=int64)
```

Matrice de confusion

- En partition binaire, "Real positive" et "Real négative" font référence aux étiquettes de vérité de base de votre ensemble de données :
 - Une donnée est vrai (1) ou fausse (0)
 - Un patient a une maladie (1) ou non (0)
 - Une radiographie montre une zone sombre (1) ou non (0)
- "Predicted Positive" et "Predicted Negative" font référence aux prédictions de votre modèle, ce que votre modèle pense de l'étiquette

```
from sklearn import metrics
y_reel = [1, 0, 0, 0, 1, 1, 1]
y_predit = [1, 0, 1, 0, 1, 1, 1]

confusion_matrix(y_reel, y_predit)
```

Matrice de confusion

- Interprétation d'une matrice de confusion :
 - Vrai Positif (True Positive : TP) = Nombre d'exemples positifs classés correctement positifs
 - Vrai Negatif (True Negative : TN) = Nombre d'exemples négatifs classés correctement comme négatifs
 - Faux Positif (False Positive : FP) = Nombre d'exemples négatifs classés à tort comme positifs
 - Faux Négatif (False Negative : FN) = Nombre d'exemples positifs classés à tort comme négatifs

```
from sklearn.metrics import multilabel_confusion_matrix
y_true = ["Chat", "Fourmis", "Chat", "Chien", "Fourmis", "Oiseau"]
y_pred = ["Chat", "Fourmis", "Chat", "Chat", "Fourmis", "Oiseau"]
multilabel_confusion_matrix(y_true, y_pred, labels=["Fourmis", "Chien", "Oiseau"])
```

Matrice de confusion

- Calcul de la justesse pour de la classification multiple :

- Accuracy =
$$\frac{\text{Prédiction correctes}}{\text{Faux positif} + \text{Faux négatif} + \text{Vrai positif} + \text{Vrai négatif}}$$

- Calcul de la justesse pour de la classification binaire :

- Accuracy =
$$\frac{\text{Vrai positif} + \text{Vrai négatif}}{\text{Faux positif} + \text{Faux négatif} + \text{Vrai positif} + \text{Vrai négatif}}$$

- Taux d'erreur = 1 - taux de réussite (accuracy) :

- $$\frac{\text{Faux positif} + \text{Faux négatif}}{\text{Faux positif} + \text{Faux négatif} + \text{Vrai positif} + \text{Vrai négatif}}$$

Matrice de confusion

- Les matrices de confusion contiennent des informations pour calculer diverses statistiques de performance, comme :
 - Précision = $\frac{Vrai\ positif}{Vrais\ positif + Faux\ positif}$ = *Taux d'exactitude des prédictions positives*
 - Rappel = $\frac{Vrai\ positif}{Vrais\ positif + Faux\ négatif}$ = *Taux positif réel*
- La Précision et le Rappel, sont nécessaires ensuite pour calculer l'aire sous la courbe précision-recours (AUPRC), une autre mesure de performance

```
from sklearn import metrics
y_reel = [1, 0, 0, 0, 1, 1, 1]
y_predit = [1, 0, 1, 0, 1, 1, 1]
print(metrics.precision_score(y_reel , y_predit))
print(metrics.recall_score(y_reel , y_predit))
```

Matrice de confusion

- Une matrice de confusion est utilisée pour calculer le taux de vrai positif (True Positive Rate : TPR) et le taux de faux positif (False Positive Rate : FPR) :
- $$TPR = \frac{Vrai\ Positif}{(Vrai\ Positif + faux\ Négatif)}$$
- $$FPR = \frac{Faux\ Positif}{(Faux\ Positif + Vrai\ Négatif)}$$
- Le TPR et le FPR seront nécessaires ensuite pour calculer la surface sous la caractéristique de fonctionnement du récepteur (AUC), une mesure de performance populaire

Matrice de confusion

- Un "seuil de décision" : nombre pour décider si une probabilité doit indiquer la classe positive ou la classe négative
- Un seuil de décision commun de 0,5 signifie que toute probabilité inférieure à 0,5 appartient à la classe négative, inversement appartient à la classe positive
- En pratique, le seuil est libre, par exemple 0,99 indique que l'événement doit avoir une probabilité supérieure à 0,99 pour être "positif". Indiquant que la majorité de vos événements seront considérés comme négatifs.

Matrice de confusion

- Un seuil de décision de 0 = Seuils de décision extrême, chaque événement est positif :
 - Donc soit un vrai positif, soit un faux positif
 - Taux positif réel = 1, ($TPR = TP / (TP + FN)$ avec $FN = 0$, donc $TPR = TP/TP = 1$)
 - Taux de faux positifs =1, ($FPR = FP / (FP + TN)$ avec $TN = 0$, donc $FPR = FP/FP = 1$)
- La valeur de la précision dépend de l'inclinaison de vos données

Matrice de confusion

- Inversement pour un seuil de décision de 1, chaque événement est négatif :
 - Donc, soit un vrai négatif, soit un faux négatif
 - Le taux positif réel = 0 ($TPR = TP / (TP + FN)$ avec $TP = 0$, donc $TPR = 0$)
 - Le taux de faux positifs = 0 ($FPR = FP / (FP + TN)$ avec $FP = 0$, donc $FPR = 0$)
 - La précision sera indéfinie ($\text{précision} = TP / (TP + FP) = 0/0$)

Matrice de confusion

- Votre choix du seuil de décision dépend de l'application en aval de votre modèle
- Exemple : Dans certaines applications médicales, les faux négatifs peuvent être pires que les faux positifs. Par exemple, dans la classification automatisée des photos dermatologiques, il ne faut pas manquer un cas de mélanome. Dans ce cas, il faut un seuil de décision plus bas afin de classer davantage d'exemples comme positifs, ce qui fera baisser votre taux de faux négatifs mais augmenter votre taux de faux positifs

Matrice de confusion

- Exemple : Imaginez une société avec un modèle prédictif utilisé pour décider si les criminels doivent être enfermés ou libérés. Dans ce cas, un faux positif peut être pire qu'un faux négatif. Le modèle ne doit pas déterminer à tort qu'un innocent est coupable et l'envie en prison et inversement
- Des courbes AUC et AUPRC peuvent aider à visualiser les compromis induits par les différents seuils de décision, et peuvent aider à choisir un bon seuil de décision pour l'application de votre modèle en aval

Matrice de confusion

- Les seuils de décision permettent de traduire les probabilités prévues en étiquettes prévues
- Si votre modèle produit des probabilités, il faut utiliser un seuil de décision pour transformer ces probabilités en étiquettes prédites. Une fois les étiquettes prédites, il est possible de construire une matrice de confusion.

Matrice de confusion

- Exemple :

```
from sklearn import neighbors, datasets
from sklearn.metrics import accuracy_score, plot_confusion_matrix
# Charger la donnée
iris = datasets.load_iris()

# Sélection dans la donnée la variable cible et l'ensemble des données test
X = iris.data[:, :2]
y = iris.target

# Utilisation de l'évaluation knn des plus proches voisins pour faire de la classification
knn = neighbors.KNeighborsClassifier(n_neighbors=7) # <-- Hyperparamètre
clf = knn.fit(X, y)

# Faire une variable de prédiction
y_pred = knn.predict(X)

matrice = plot_confusion_matrix(clf, X, y_pred, cmap='Greys_r')

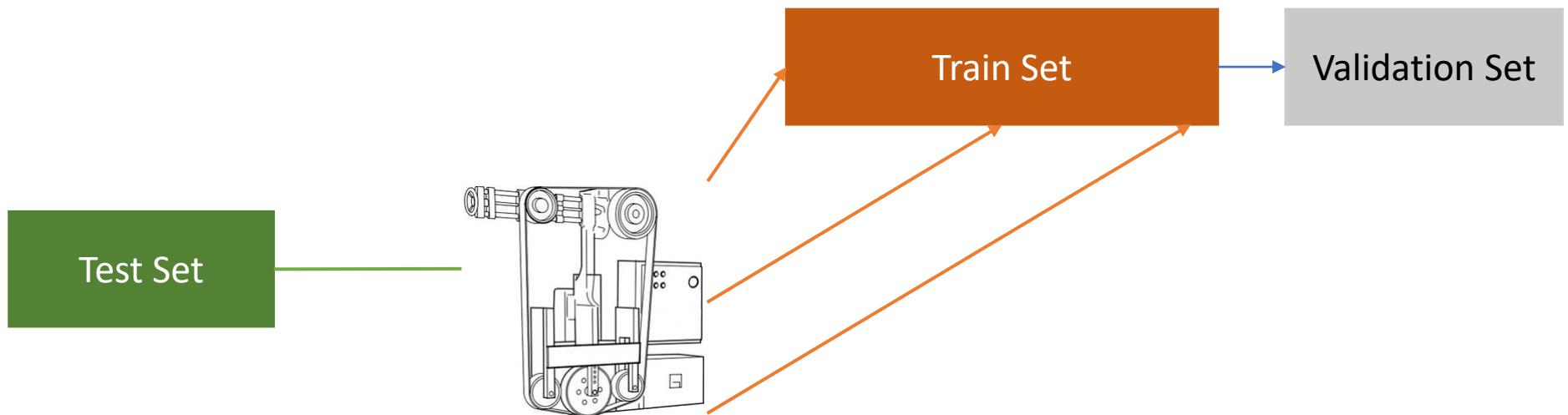
matrice = plot_confusion_matrix(clf, X, y)

print(y)
print(y_pred)

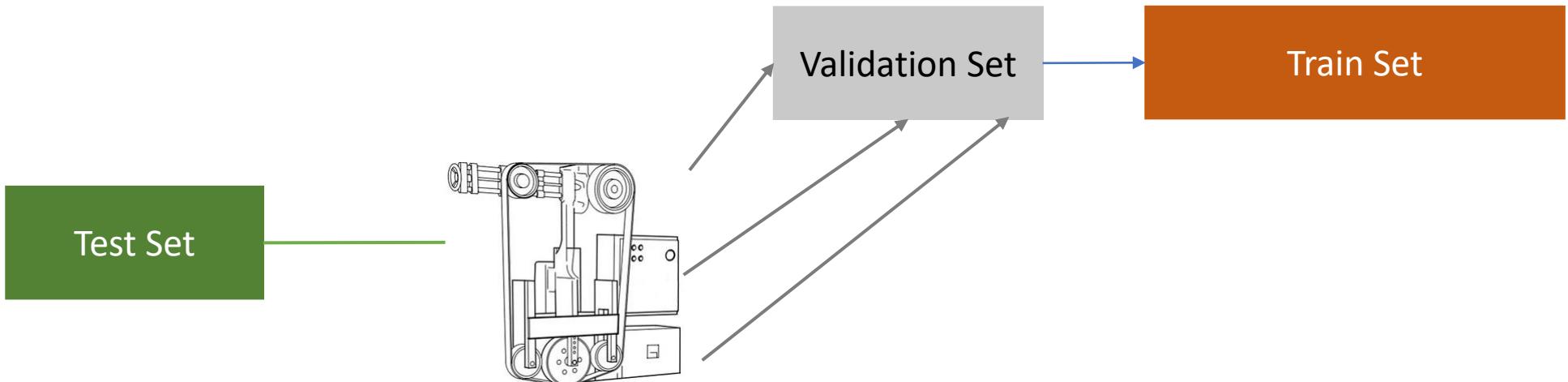
# Afficher le taux de l'évaluation cible / prédiction
accuracy_score(y, y_pred)
```

Cross validation

Le principe d'évaluation



Le principe d'évaluation



Le principe d'évaluation

- Cross Validation : Entrainer et valider le modèle sur plusieurs découpes du Train Set

Train Set					Modèle 1	Modèle 2
Validation	Test	Test	Test	Test	0.90	0.91
Test	Validation	Test	Test	Test	0.87	0.89
Test	Test	Validation	Test	Test	0.86	0.87
Test	Test	Test	Validation	Test	0.93	0.92
Test	Test	Test	Test	Validation	0.88	0.91
Moyenne					0.888	0.9

Le principe d'évaluation

- Cross Validation : Entrainer et valider le modèle sur plusieurs découpes du Train Set

Train Set					Modèle 1	Modèle 2
Validation	Test	Test	Test	Test	0.90	0.91
Test	Validation	Test	Test	Test	0.87	0.89
Test	Test	Validation	Test	Test	0.86	0.87
Test	Test	Test	Validation	Test	0.93	0.92
Test	Test	Test	Test	Validation	0.88	0.91
Moyenne					0.888	0.9

Le principe d'évaluation

- Plusieurs techniques de cross validation : KFold, StratifiedKFold, ShuffleSplit :

```
from sklearn.model_selection import cross_val_score  
  
print(cross_val_score(KNeighborsClassifier(5), X_train, y_train, cv=5, scoring='accuracy'))  
print(cross_val_score(KNeighborsClassifier(5), X_train, y_train, cv=5, scoring='accuracy').mean())
```

- Par exemple comme résultat :

```
[1.          1.          1.          0.95833333  0.95833333]  
0.9833333333333334
```

Le principe d'évaluation

- Pour comparer plusieurs résultats avec des hyperparamètres différents :

```
from sklearn.model_selection import validation_curve

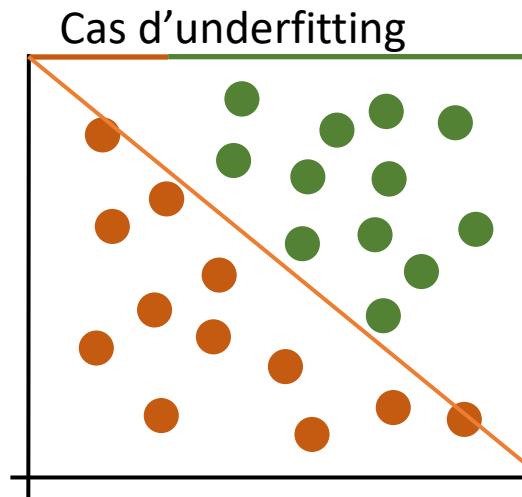
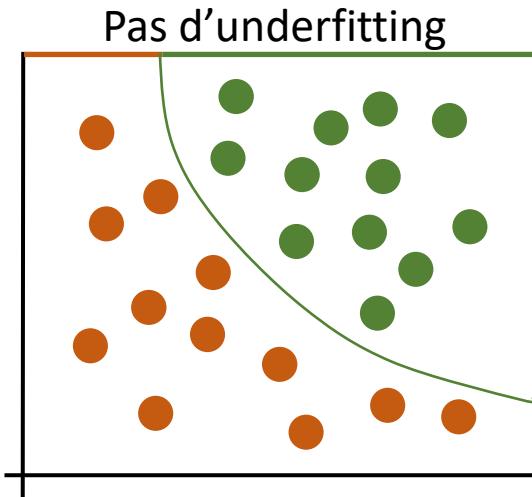
i = np.arange(1, 50)
train_score, validation_score = validation_curve(KNeighborsClassifier(), X, y,
                                                param_name="n_neighbors", param_range=i, cv=5)
print(validation_score.mean(axis=1))
```

- Afficher les résultats pour avoir une meilleure lecture :

```
plt.plot(i,train_score.mean(axis=1), label="train score")
plt.plot(i,validation_score.mean(axis=1), label="validation score")
plt.legend(loc="best")
```

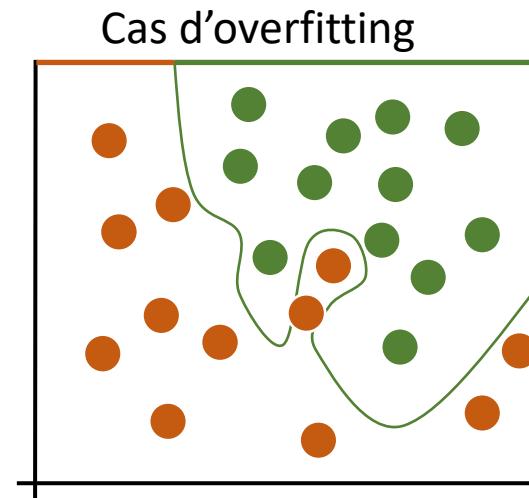
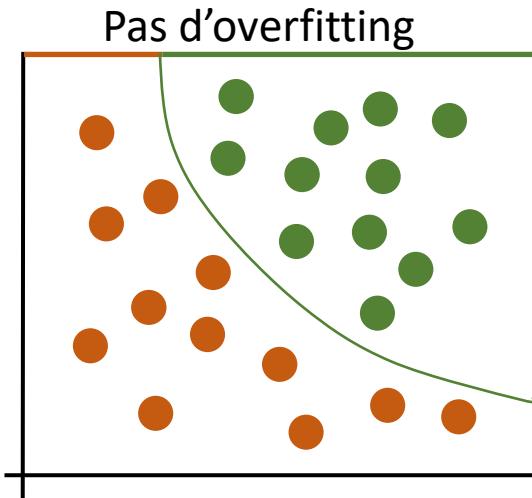
Le principe d'évaluation

- Cas Underfitting (sous-ajustement) :
 - Le modèle manque d'ajustement par rapport au train Set
 - Le modèle ne généralise pas exactement et fait des erreurs de prédictions



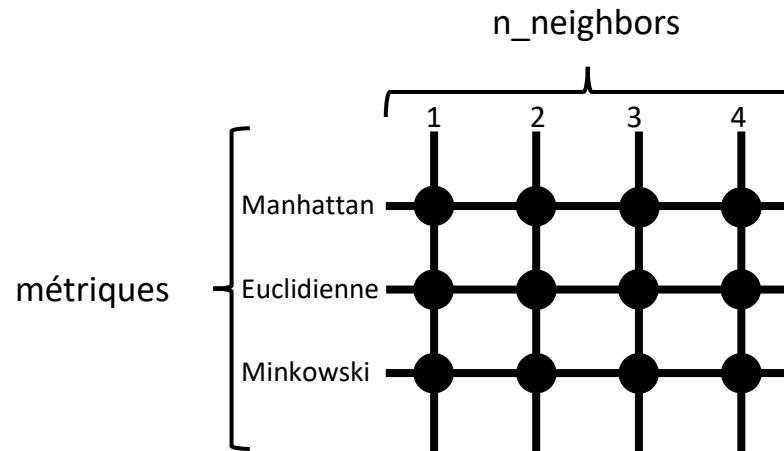
Le principe d'évaluation

- Cas Overfitting (sur-ajustement) :
 - Le modèle est trop bon sur le train Set (par exemple le nombre de voisin = 1)
 - Le modèle a perdu tout sens de généralisation et fait des erreurs de prédictions



Le principe d'évaluation

- Pour tout tester, il faut avoir le modèle avec les meilleurs paramètres possibles :
 - Comparer les différentes performances
 - Faire des combinaisons avec cross_validation
 - Faire une grille de modèles avec toutes les combinaisons et les métriques :



Le principe d'évaluation

- Utilisation de GridSearchCV() :

```
from sklearn.model_selection import GridSearchCV

p_grid = {'n_neighbors' : np.arange(1,50), 'metric': ['manhattan','euclidean','minkowski']}

grid = GridSearchCV(KNeighborsClassifier(), p_grid, cv=4)
grid.fit(X_train, y_train)

grid.best_score_

grid.best_params_

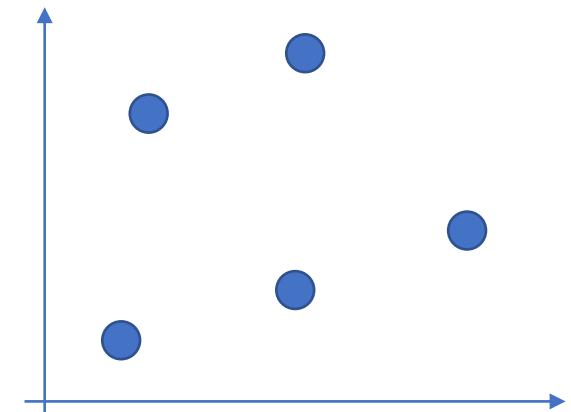
m = grid.best_estimator_

m.score(X_test,y_test)
```

Matrice de corrélation

Matrice de corrélation

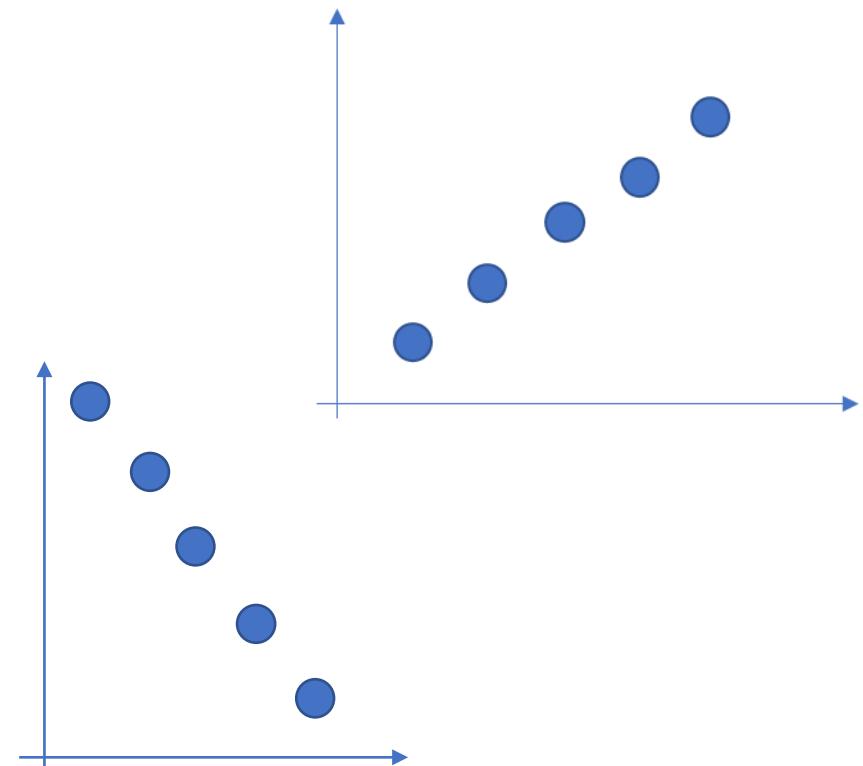
- Une corrélation est un test statistique d'association entre les variables mesurées sur une échelle de -1 à 1
- Mesure comment le changement dans une variable est associée au changement dans une autre variable
- Pas de corrélation :
 - Pas d'association entre les changements des deux variables



Matrice de corrélation

- Corrélation positive :
 - Si une variable augmente, l'autre augmente aussi
- Corrélation négative (inverse) :
 - Si une variable augmente, l'autre diminue

Valeur r	Force
0,0 - 0,2	Faible corrélation
0,3 - 0,6	Corrélation modérée
0,7 - 1,0	Forte corrélation



Matrice de corrélation

- Tests de corrélation :
 - Modèle pour évaluer en même temps la dépendance entre plusieurs variables :
 - Corrélation de Pearson
 - Corrélation de Kendall
 - Corrélation de Spearman
- Matrice de corrélation :
 - Structure contenant les coefficients de corrélation entre les variables
 - La coloration met en évidence les variables les plus corrélées entre elles

Matrice de corrélation

- Corrélation de Pearson (Coefficient de corrélation produit-moment de Pearson) :
 - Mesure de la relation linéaire entre deux variables A et B
 - Soit $Cov(A, B)$ est la covariance entre A et B
 - Soit $\sigma(A)$ est l'écart type de A
 - Le coefficient de corrélation de Pearson ρ est :
$$\bullet \quad \rho(A, B) = \frac{Cov(A, B)}{\sigma(A)\sigma(B)} = \frac{\overline{AB} - \bar{A}\bar{B}}{\sqrt{\overline{A^2} - \bar{A}^2} \sqrt{\overline{B^2} - \bar{B}^2}}$$
 - La covariance est toujours inférieure au produit des écarts types, donc la valeur $\rho(A, B)$ varie entre -1 et 1

Matrice de corrélation

- Le coefficient de corrélation de Pearson mesure l'association linéaire entre les variables:
 - **+1** : Corrélation positive complète
 - **+0,8** : Forte corrélation positive
 - **+0,6** : Corrélation positive modérée
 - **0** : Aucune corrélation quelle qu'elle soit
 - **-0,6** : Corrélation négative modérée
 - **-0,8** : Forte corrélation négative
 - **-1** : Corrélation négative complète

Matrice de corrélation

- Les variables avec des associations linéaires élevées, ont leur coefficient de corrélation proche de +1 ou -1
- Les variables statistiquement indépendantes ont des coefficients de corrélation proches de zéro
- Les associations non linéaires entre les variables ont un coefficient de corrélation nul ou proche de zéro, impliquant que les variables avec des associations élevées peuvent ne pas avoir une valeur élevée du coefficient de corrélation de Pearson

Matrice de corrélation

- La corrélation de Spearman est un test non paramétrique sans hypothèse sur la distribution des données
- 2 hypothèses pour le test de corrélation de Spearman :
 - Chaque variable est une mesure ordinaire, de rapport ou d'intervalle
 - Une relation monotone entre les variables testées

Matrice de corrélation

- La corrélation de Kendall est un test non-paramétrique sans hypothèse sur la distribution des données
- 2 hypothèses :
 - Chaque variable doit être une mesure ordinaire, de rapport ou d'intervalle
 - Homoscédasticité des données : $\forall i \nu(x_i) = \sigma^2$

Applications Machine Learning

PRÉSENTATION : DIFFÉRENTS MODÈLES
D'APPRENTISSAGE EN PYTHON AVEC
ÉVALUATION DES PRÉDICTIONS

Dr. Chaabani Yasmine
yasmin.chaabani@isgb.ucar.tn

Le but du module ?

Faire des Machines Learning de la donnée aux résultats

Sommaire

Présentation

Comment choisir son modèle

Machine Learning par les modèles prédictifs

Machine Learning par les modèles statistiques

Présentation

Les biais

Conditionnement des données et réduction de dimension

La méthode

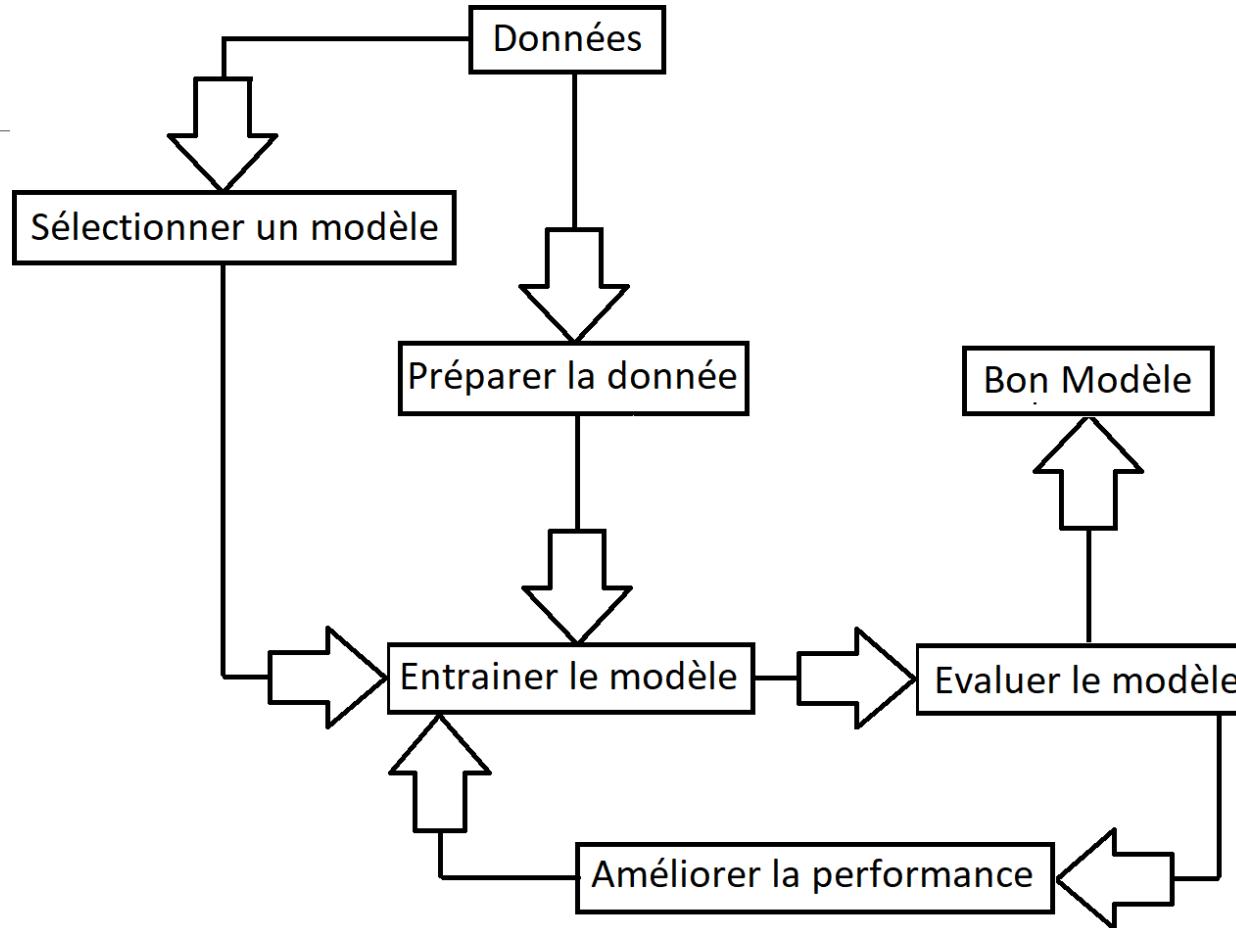
Le coût

Le résultat

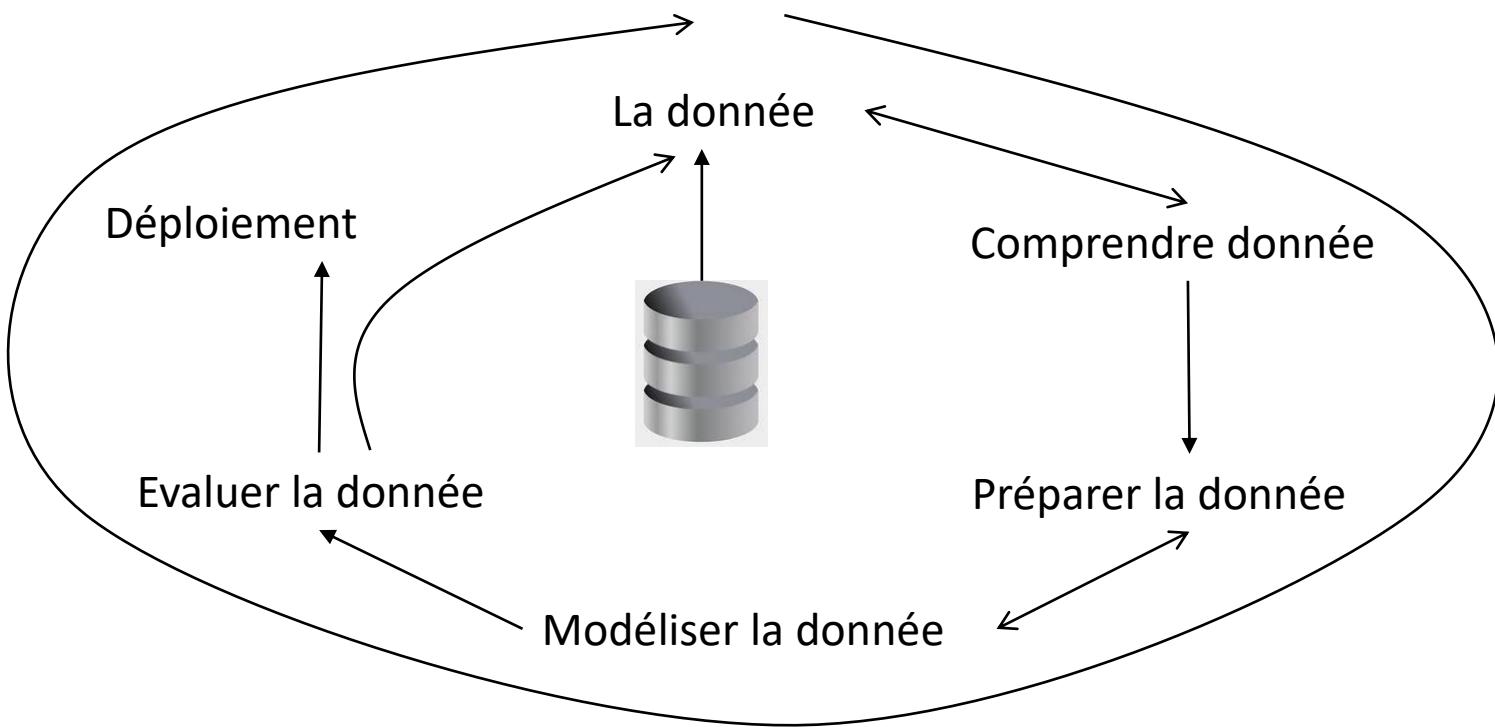
Présentation

1. Préparer la donnée
2. Inspecter la donnée
3. Modéliser la donnée
4. Sélectionner l'algorithme d'apprentissage (modèle)
5. Faire la division de la donnée en fonction de l'algorithme d'apprentissage (biais)
6. Faire l'entrainement avec les données (paramétrage du modèle, train, test)
7. Evaluer le modèle (cross validation, matrice de confusion, Naïve Bayes)
8. Améliorer la performance (paramétrage du modèle)
9. Acceptation du modèle

Présentation

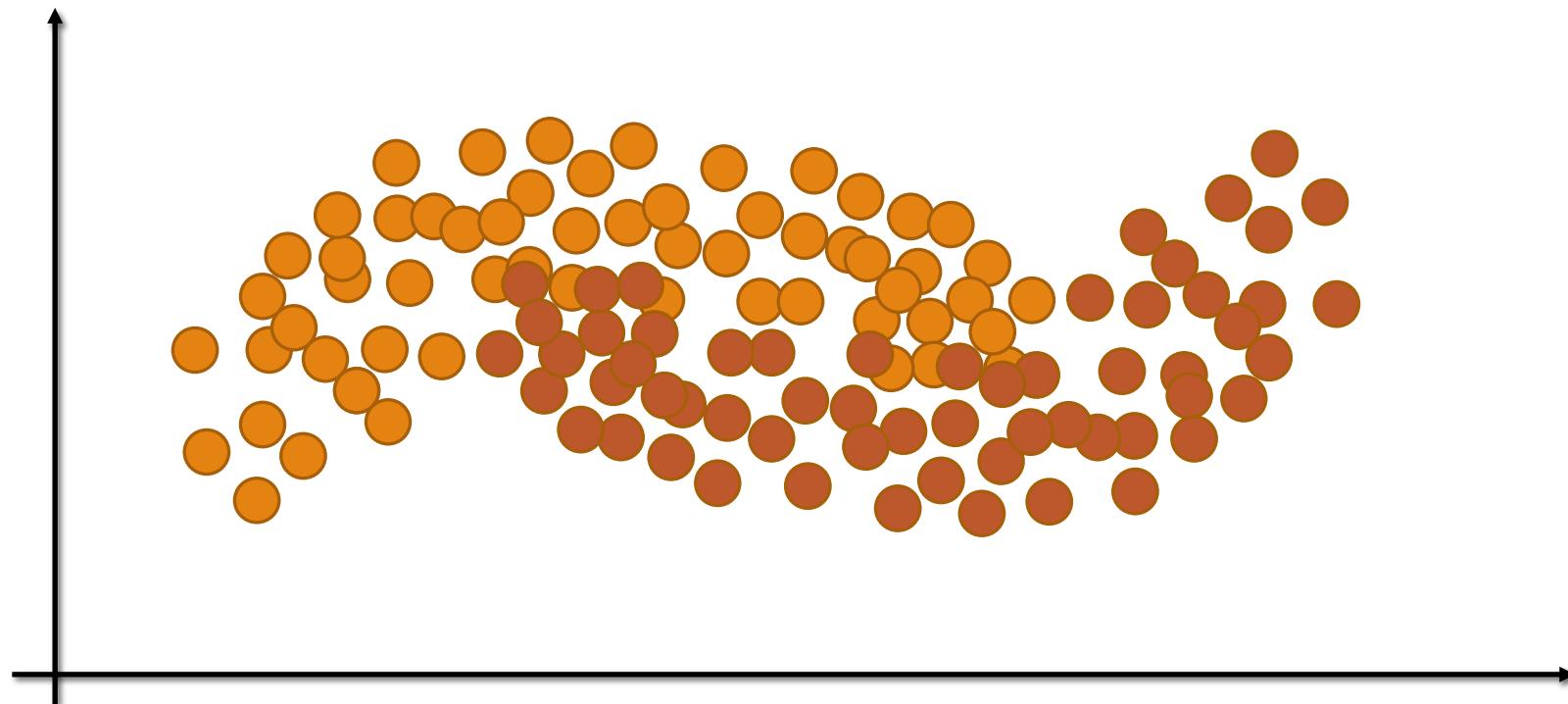


Présentation



Comment choisir son modèle

Avec un ensemble de données : Comment choisir le bon modèle?



Comment choisir son modèle

Il est biaisé de penser qu'à chaque jeu de données correspond un modèle adapté

Chaque type de modèle répond à des questions spécifiques :

- Une glycémie liée à un diabète particulier peut être expliquée par une variable qualitative (Sexe des individus) :
 - Le modèle ANOVA est approprié
- Cette glycémie peut être expliquée par une variable quantitative (âge des individus) :
 - La régression linéaire simple est appropriée (augmentation ou diminution linéaire de cette glycémie en fonction de l'âge)

Comment choisir son modèle

Utiliser toujours des modèles simples et connus :

- Pour une Régression :
 - Régression Linéaire
- Pour une classification :
 - Régression logistique
 - Les proches voisins : Neighbors
 - Vecteur machine avec un Kernel linéaire : SVM, linear
 - Arbre de décision

Comment choisir son modèle

Les critères à appliquer pour choisir un modèle :

1. La quantité de la donnée
2. La structures de la donnée (structurée/non structuré)
3. La norme de la donnée
4. La teneur quantitatives et qualitatives de la donnée

Comment choisir son modèle

Les modèles se choisissent en fonction de la quantité de données disponibles

Pour le DataSet peu volumineux :

- Kneighbors
- SVM

Pour le DataSet Volumineux :

- Régression logistique/Descente de gradient
- Réseaux de Neurone (TensorFlow, Torch,...)

Comment choisir son modèle

Les modèles se choisissent en fonction de la structuration de la donnée :

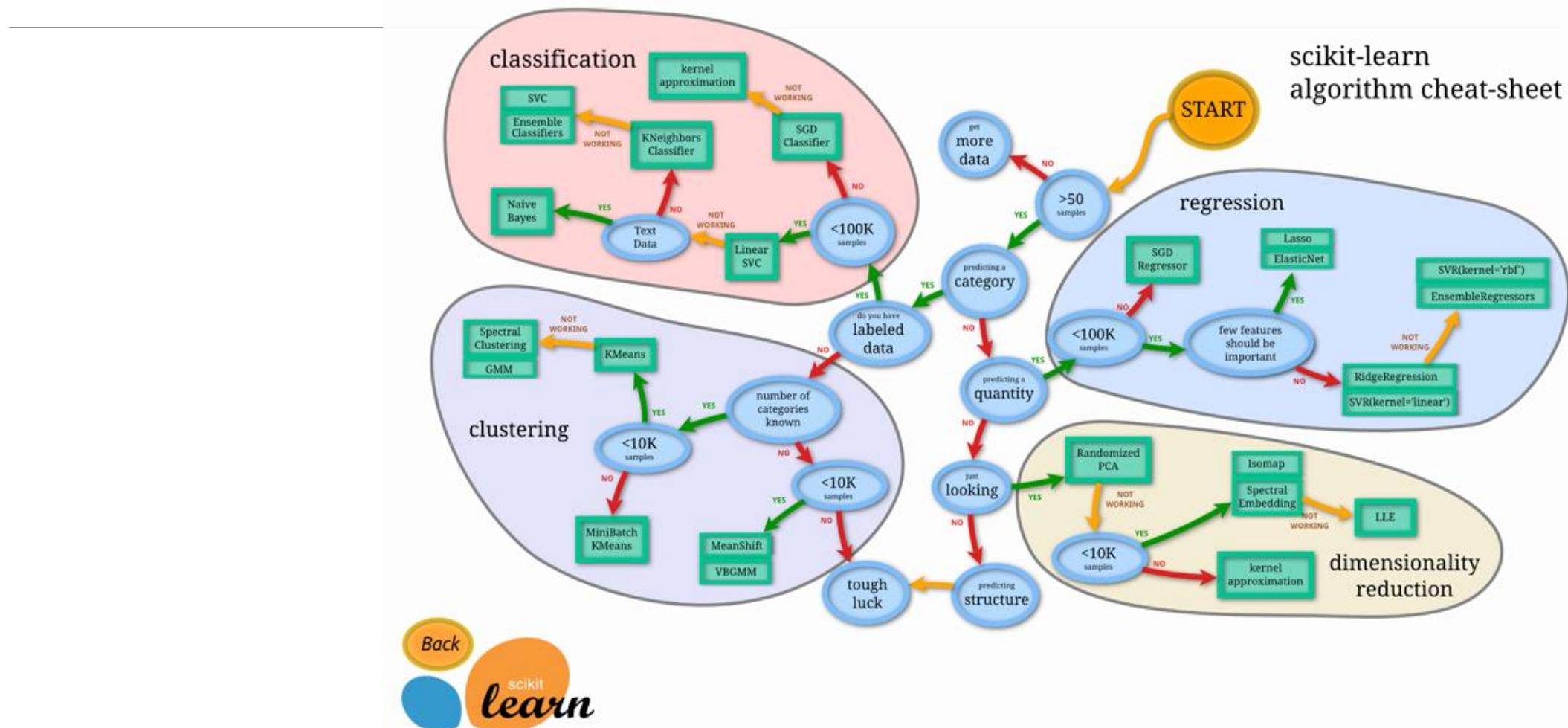
Structurées

- Déjà étiquetées

Non Structurés

- Photos
- Son

Comment choisir son modèle



Machine Learning par les modèles statistiques

Nous avons les données sous la forme d'un tableau :

Y	8	5	10	11	27	8	11	0	3
X	34	20	76	78	100	34	78	0	10

```
import numpy as np  
  
X = np.array([34,20,76,78,100,34,78,0,10])  
y = np.array([8,5,10,11,27,8,11,0,3])
```

Machine Learning par les modèles statistiques

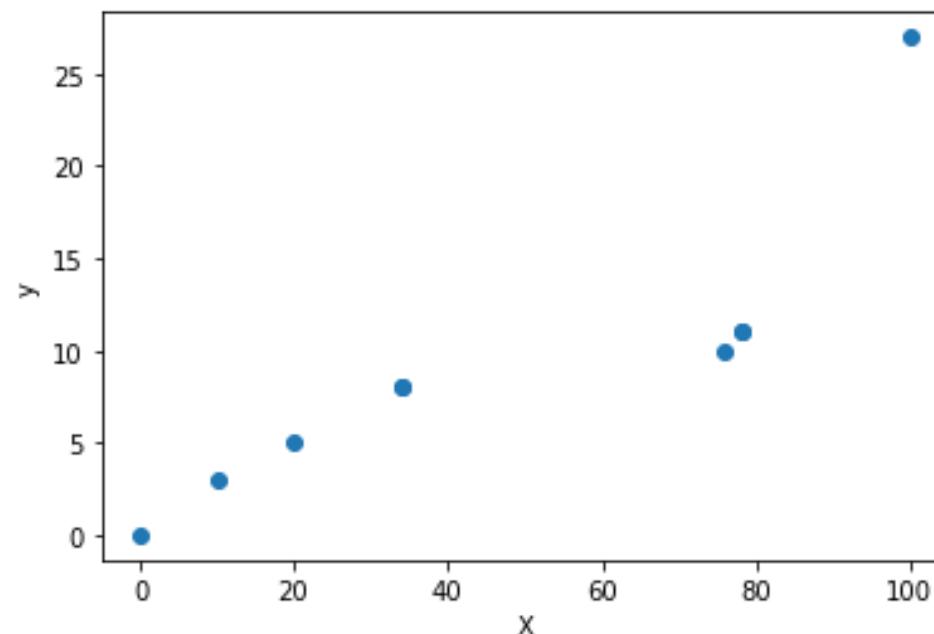
Nous faisons la corrélation entre les données avec le module statistique :

- Coefficient de Pearson = 0.8624238840508004

```
from scipy.stats import pearsonr  
  
coeff_pearson,_ = pearsonr(X,y)  
print("coefficient de Pearson = {}".format(coeff_pearson))
```

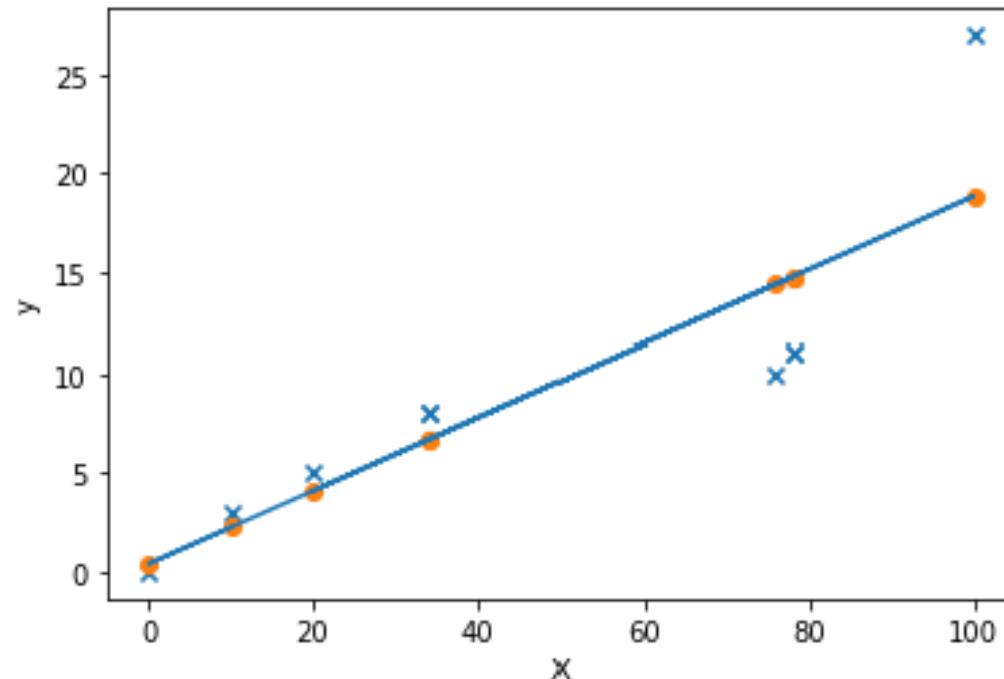
Machine Learning par les modèles statistiques

Nous regardons graphiquement le positionnement des données :



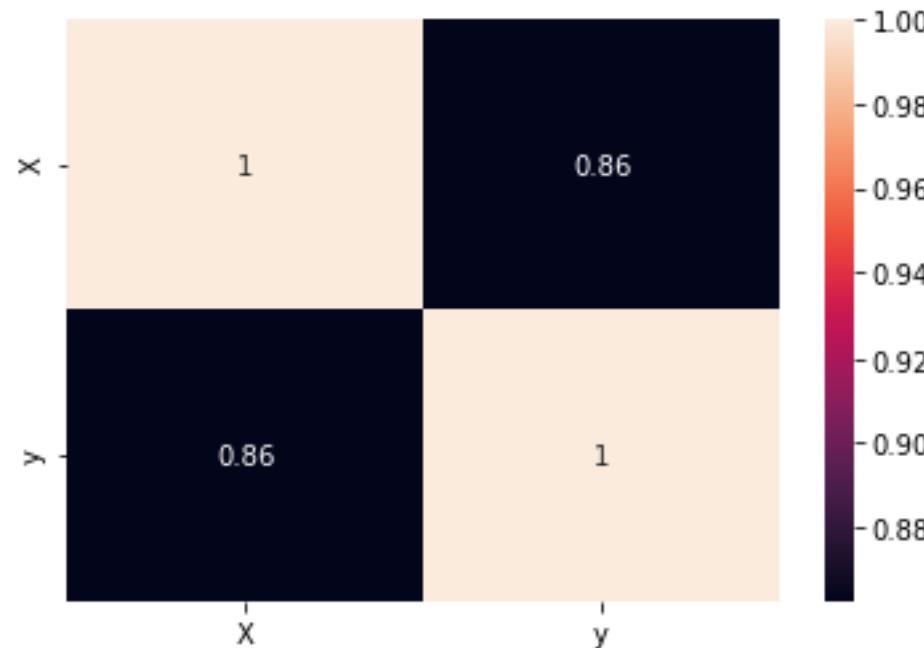
Machine Learning par les modèles statistiques

Avec le visuel des données, un apprentissage linéaire est envisageable :



Machine Learning par les modèles statistiques

Matrice de corrélation des données est la suivante :



Machine Learning par les modèles statistiques

Nous un tableau entre données réelles et prédictes :

Y_reel	1	0	0	1	0	1	0	0	1	0	1	0
Y_predit	1	1	0	1	0	1	1	0	1	0	0	0

```
import pandas as pd

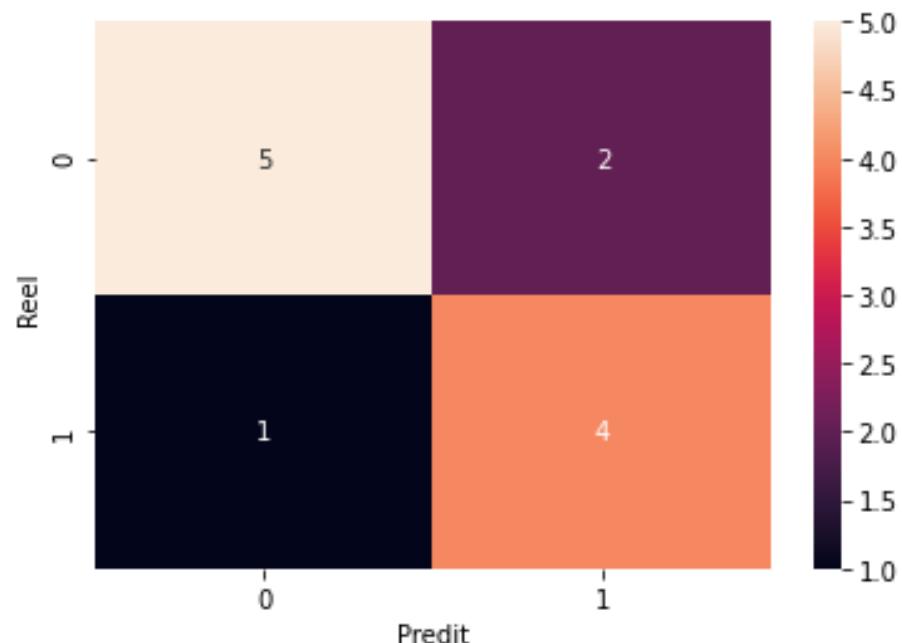
data = {'y_Reel': [1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0],
        'y_Predit': [1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0]
       }

data_Frame = pd.DataFrame(data, columns=['y_Reel', 'y_Predit'])
```

Machine Learning par les modèles statistiques

Nous pouvons estimer l'erreur avec une matrice de confusion :

```
confusion_matrix = pd.crosstab(data_Frame['y_Reel'], data_Frame['y_Predit'], rownames=['Reel'], colnames=['Predit'])
sn.heatmap(confusion_matrix, annot=True)
```

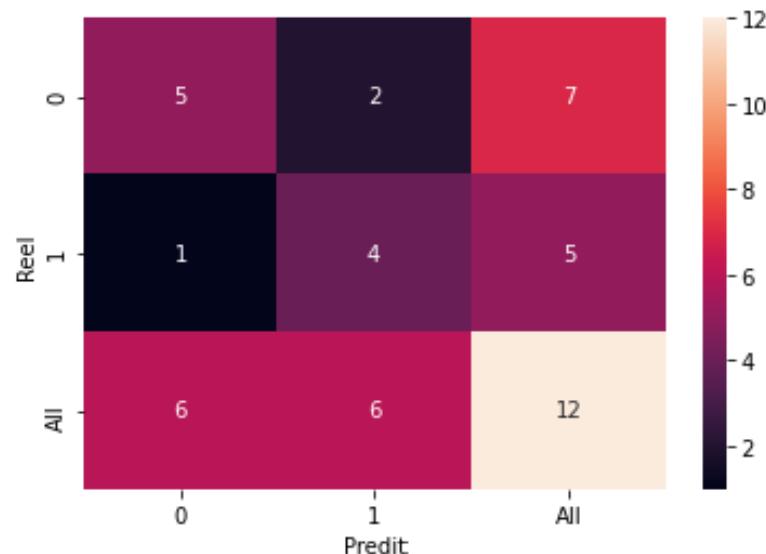


Machine Learning par les modèles statistiques

Si nous voulons le décompte des colonnes :

```
confusion_matrix = pd.crosstab(data_Frame['y_Reel'], data_Frame['y_Predit'], rownames=['Reel'],
                                 colnames=['Predit'], margins = True)

sn.heatmap(confusion_matrix, annot=True)
```



Non supervisé

Clustering

Détection d'anomalie

Réduction de dimension

Non supervisé

Non supervisé :

- Au lieu de fournir des données X et y :



Pince



Tournevis

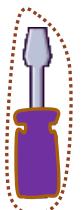


Marteau

- Que la donnée x est disponible :

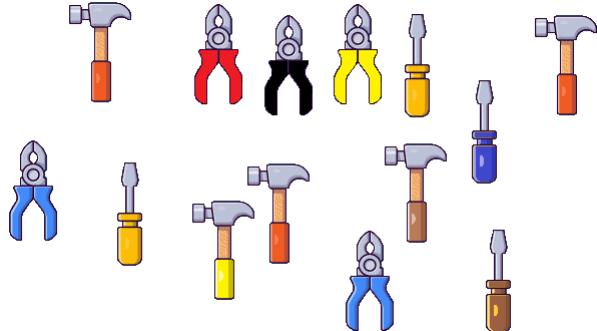


- Analyser la structure des données X pour apprendre à réaliser certaines tâches :

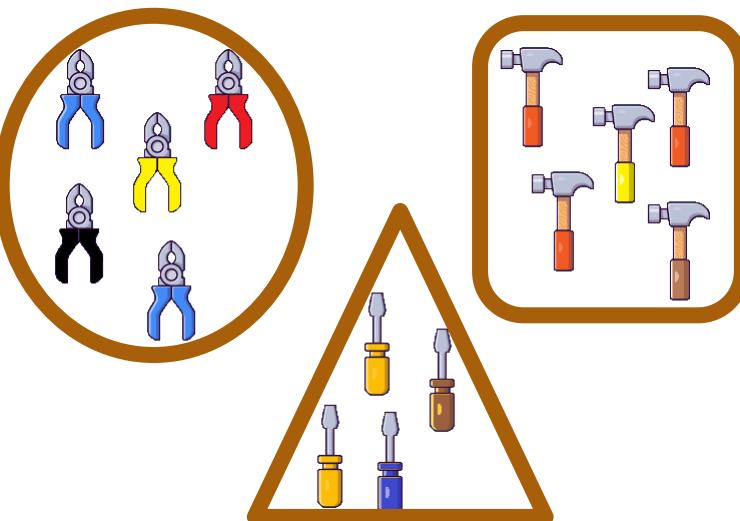


Non supervisé

Données X initiales :



Apprentissage par ressemblance :



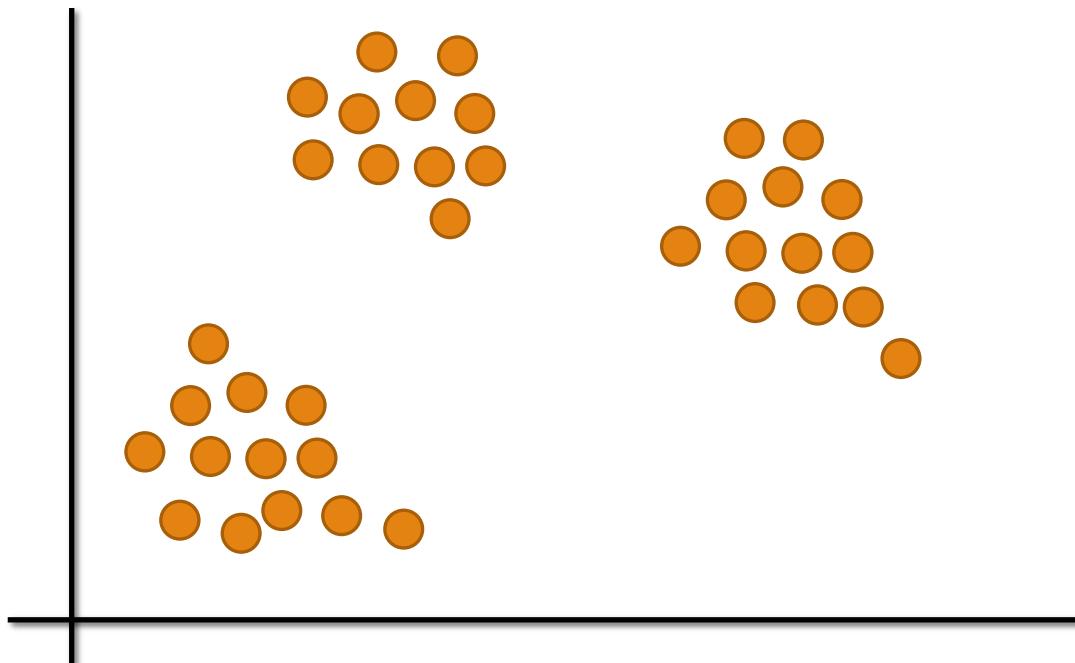
Clustering :

- Algorithme : K-means clustering

Non supervisé

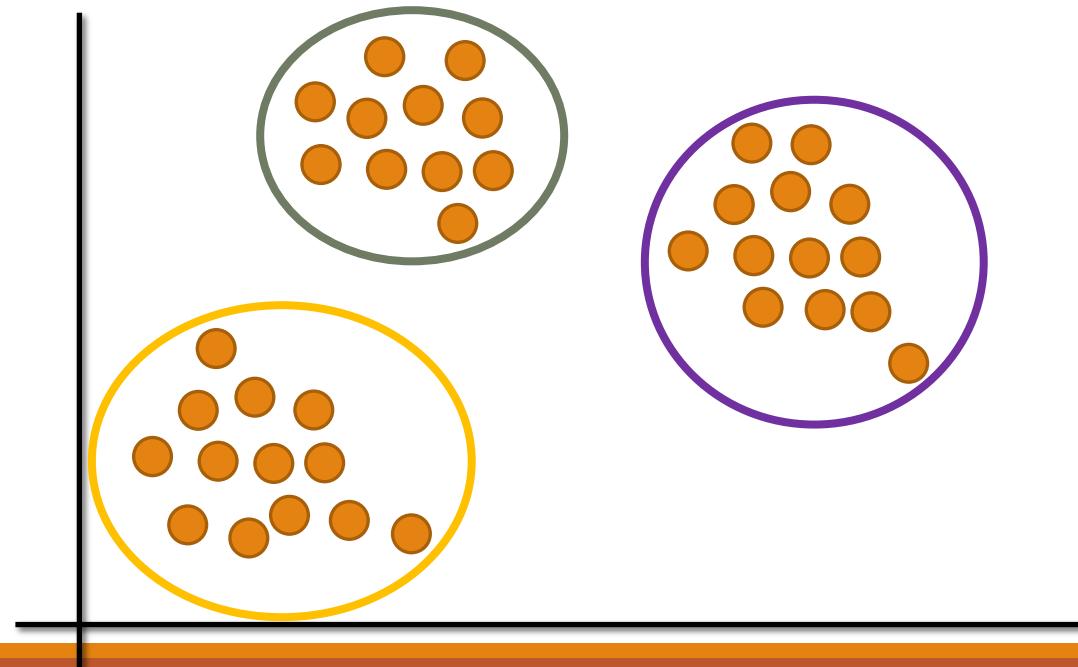
Laisser la machine à classer nos données selon leur ressemblance :

- Fonctionnement du KMeans Clustering :



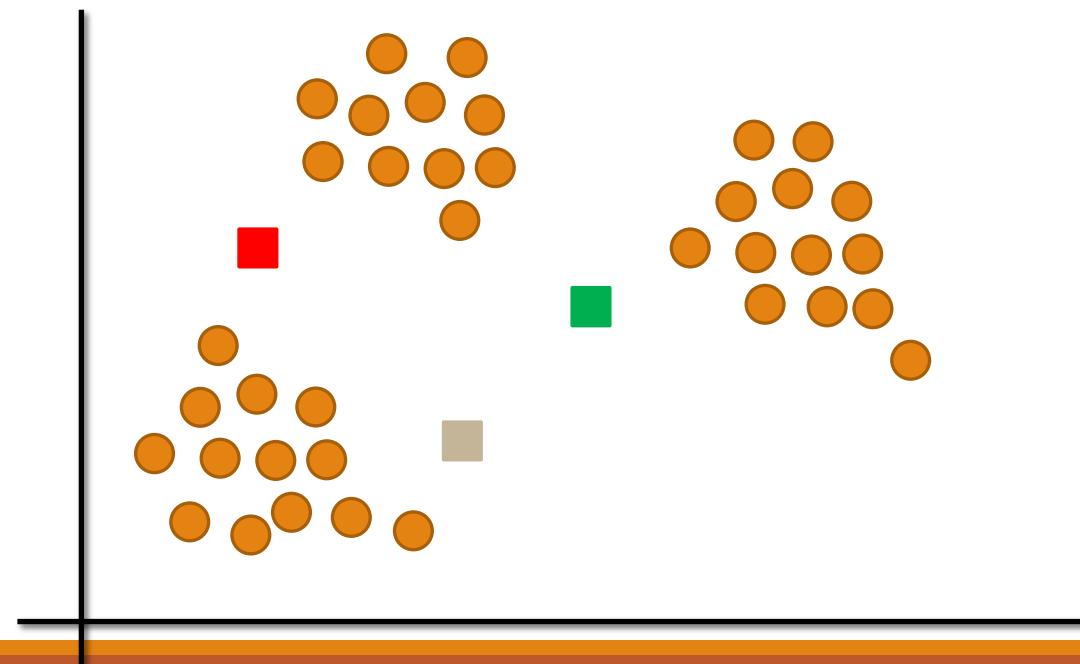
Non supervisé

Comment regrouper les données en Cluster ?



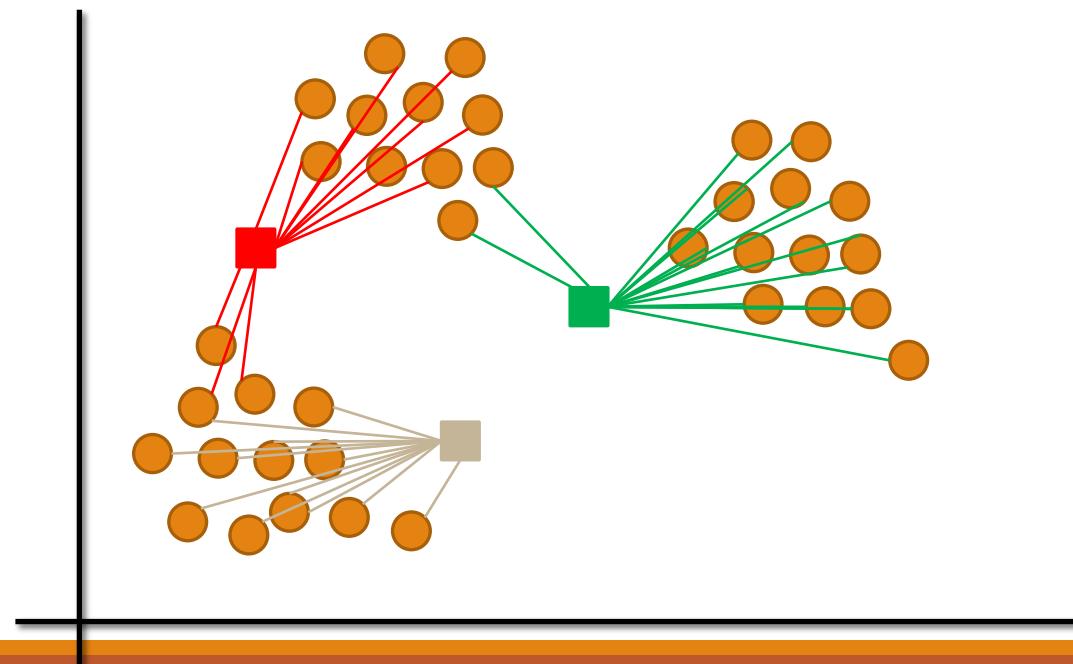
Non supervisé

Phase 1 : Il faut placer des centroïdes (ou le centre géométrique d'une région bidimensionnelle) :



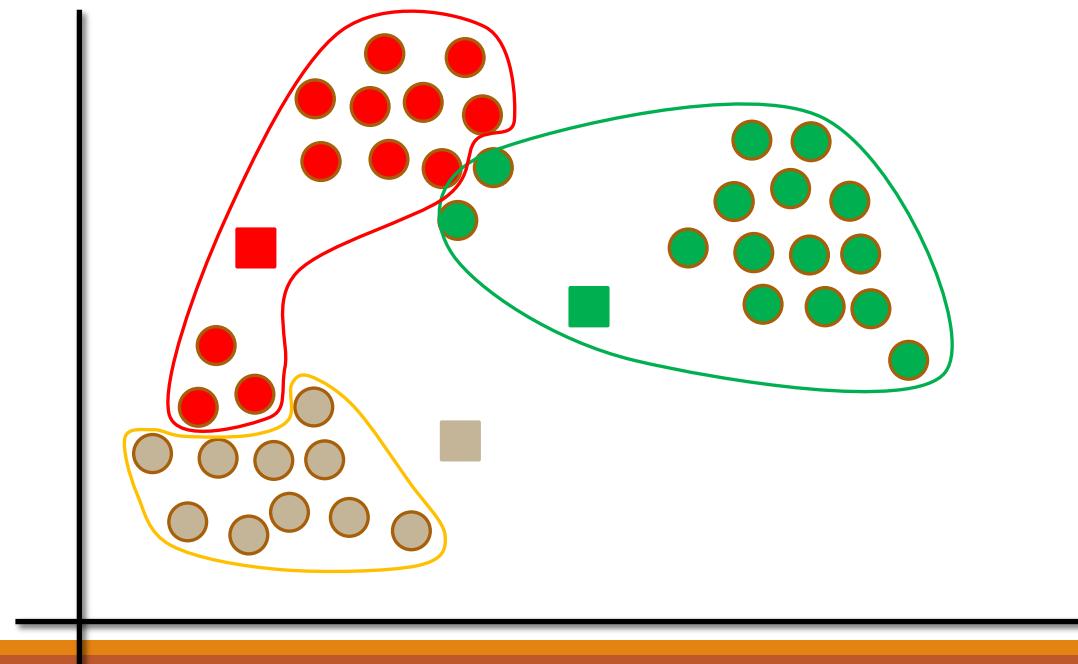
Non supervisé

Phase 2 : Lier les données proches aux centroïdes :



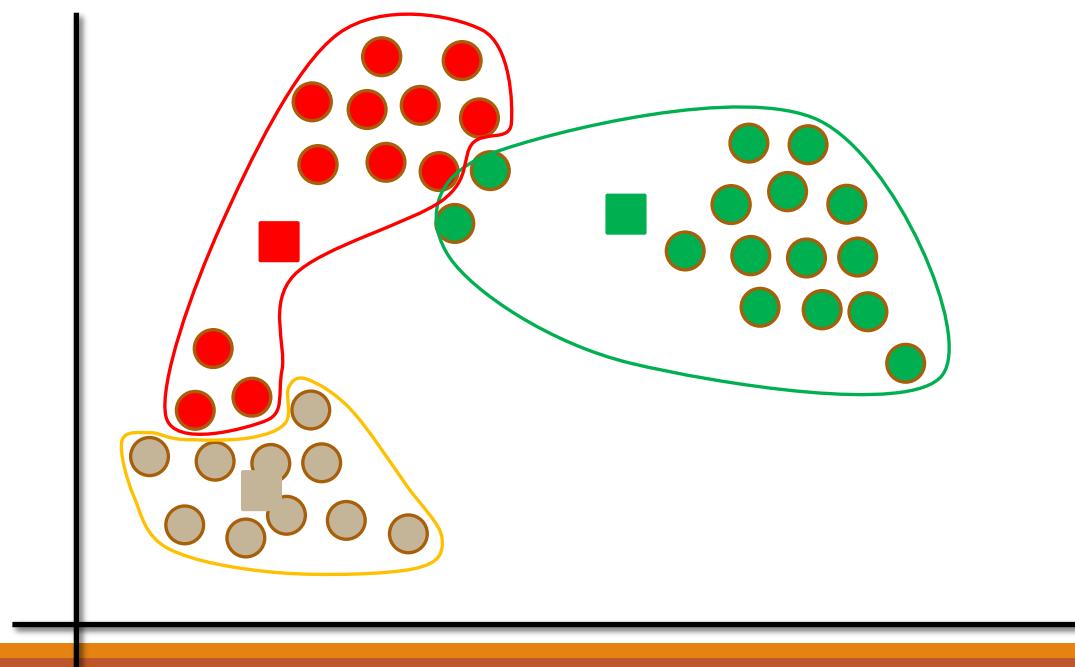
Non supervisé

Regroupes les données de même centroïde :



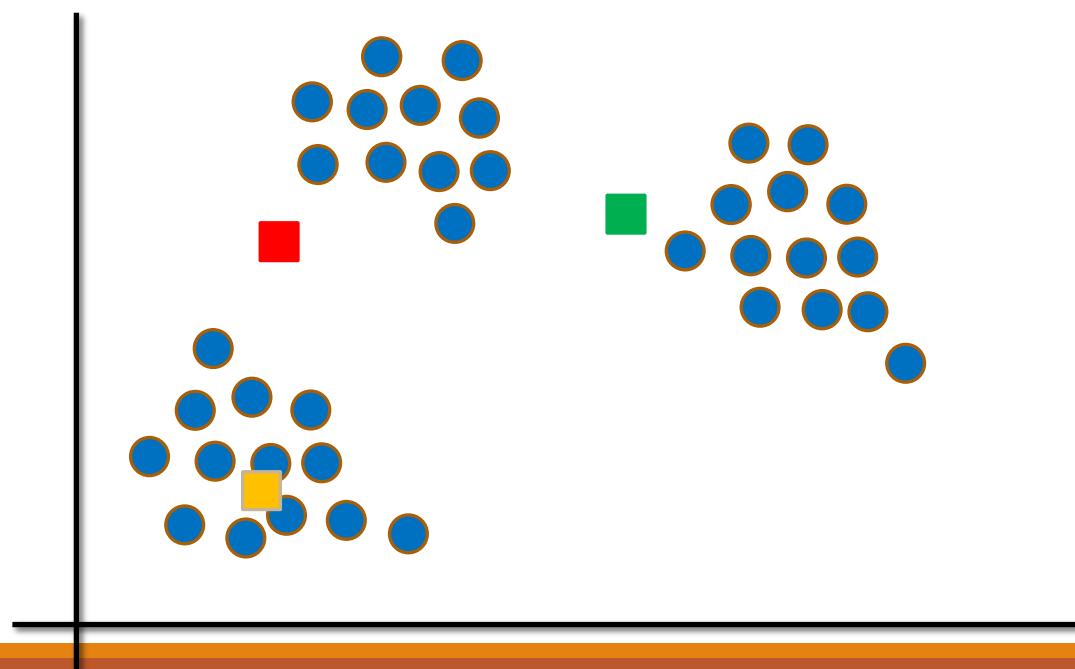
Non supervisé

Déplacer les centroïdes au centre de chaque groupe précédemment formé :



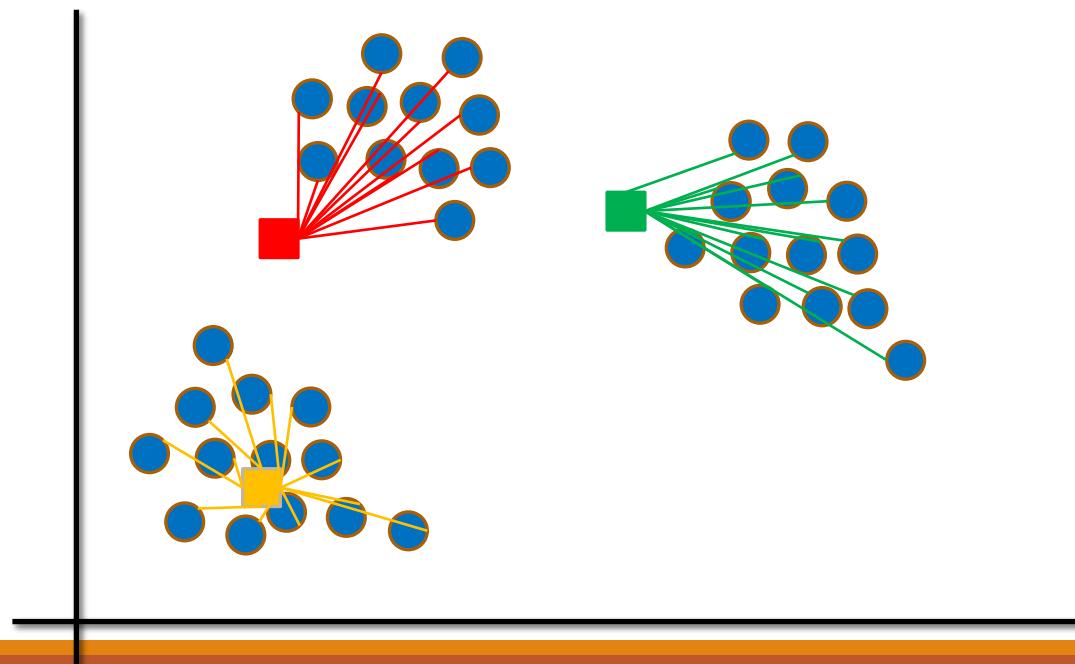
Non supervisé

Recommencer le début de la phase 2 :



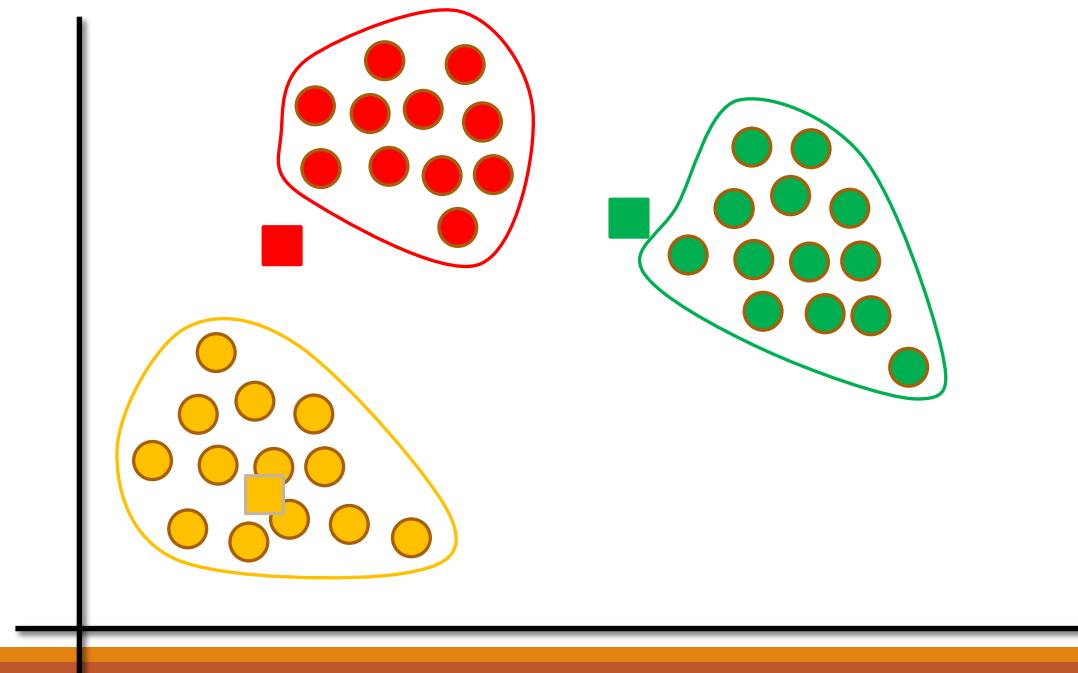
Non supervisé

Lier les données proches aux centroïdes :



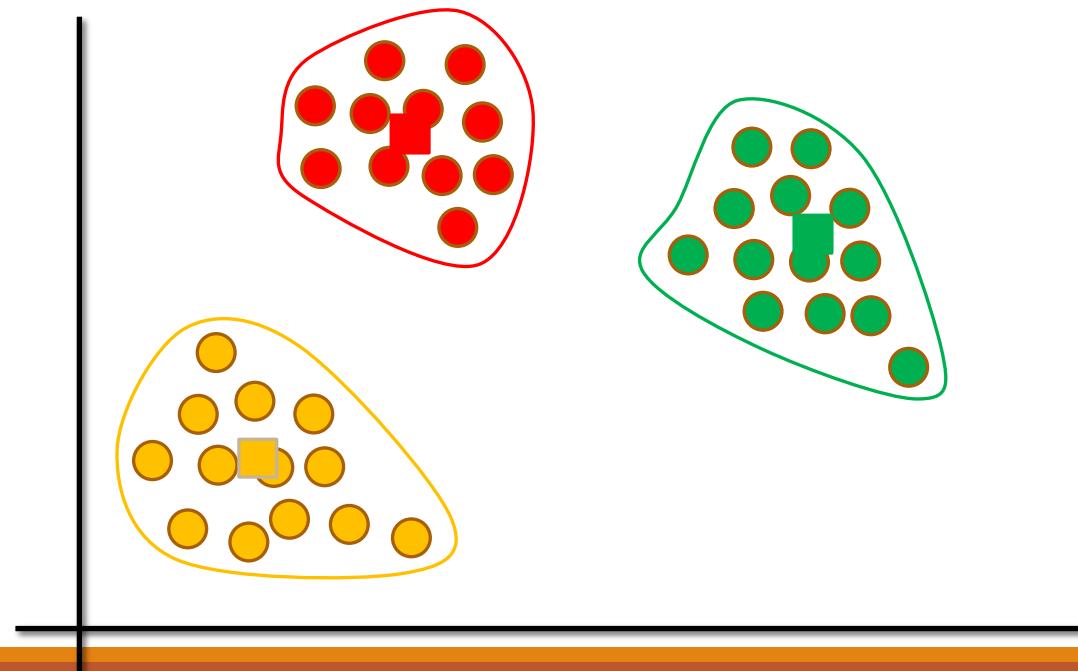
Non supervisé

Regroupes les données de même centroïde :



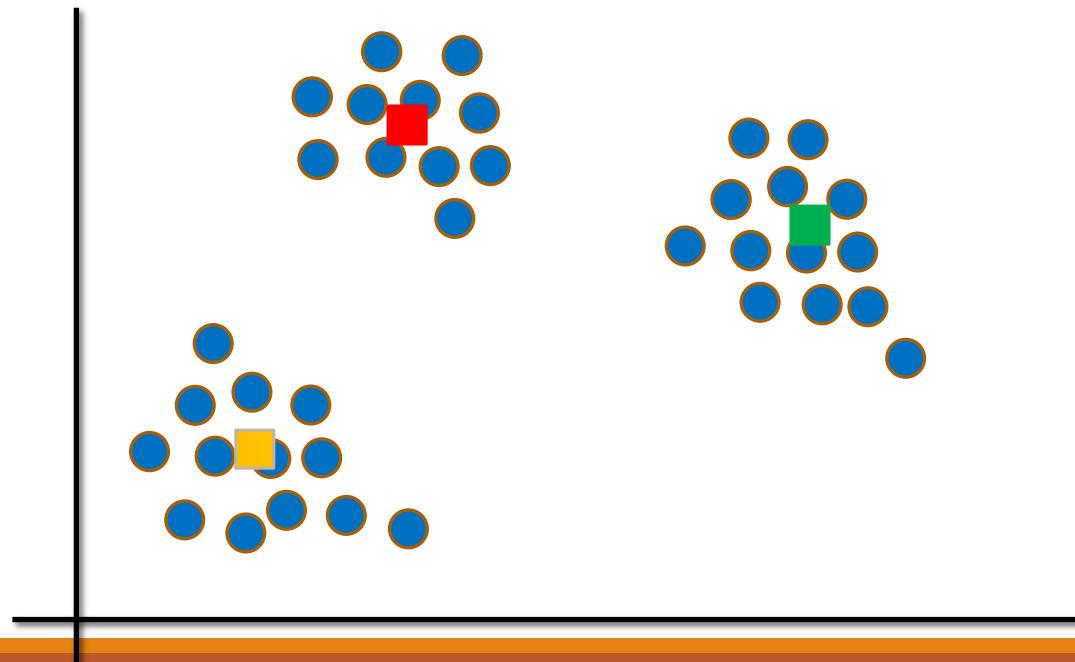
Non supervisé

Déplacer les centroïdes au centre de chaque groupe précédemment formé :



Non supervisé

Fin de l'exécution quand les centroïdes ne peuvent plus être déplacés :



Non supervisé

L'algorithme KMeans clustering c'est :

Phase 1 : L'initialisation :

- Récupération du DataSet
- Place des centroïdes au hasard parmi le DataSet

Phase 2 : Des itérations pour :

- Affectation des données du DataSet au centroïde le plus proche
- Déplacer le centroïde au centre du groupe de données affectées
- Vérifier la place des centroïdes

Non supervisé

Exemple :

Récupération des données :

```
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
X, y = make_blobs(n_samples=400, centers=3, cluster_std=1)
plt.scatter(X[:,0],X[:,1])
```

Importation de KMeans :

```
from sklearn.cluster import KMeans
```

Non supervisé

Prédiction en fonction de la donnée :

```
model = KMeans(n_clusters = 3, n_init = 10, max_iter = 300)
model.fit(X)
plt.scatter(X[:,0],X[:,1], c=model.predict(X))
plt.scatter(model.cluster_centers_[:,0], model.cluster_centers_[:,1], c='r')

model.cluster_centers_
```

Non supervisé

Exercice :

- Faire une cross validation pour l'exemple KMeans précédent en faisant varier le nombre de centre de 1 à 20
- Utiliser `inertia_` qui est la sommes des carrés des distances des échantillons à leur centre de cluster le plus proche :
`print(model.inertia_)`
- Faire une représentation graphique de l'évolution avec tous les centres

Non supervisé

Réponse possible à l'exercice :

```
inertia = []
K_range = range(1,20)
for k in K_range:
    model = KMeans(n_clusters=k).fit(X)
    inertia.append(model.inertia_) # stockage des coûts
```

```
plt.plot(K_range, inertia)
plt.xlabel('nombre de cluster')
plt.ylabel('Cout du modèle (Inertia)')
```

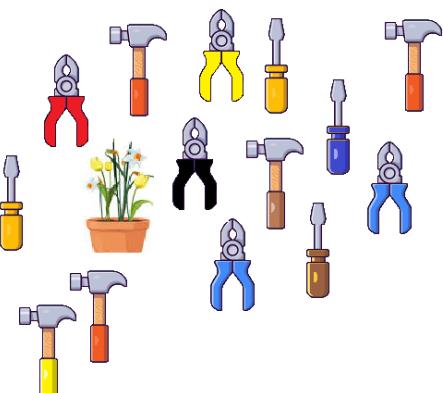
Non supervisé

Non supervisé :

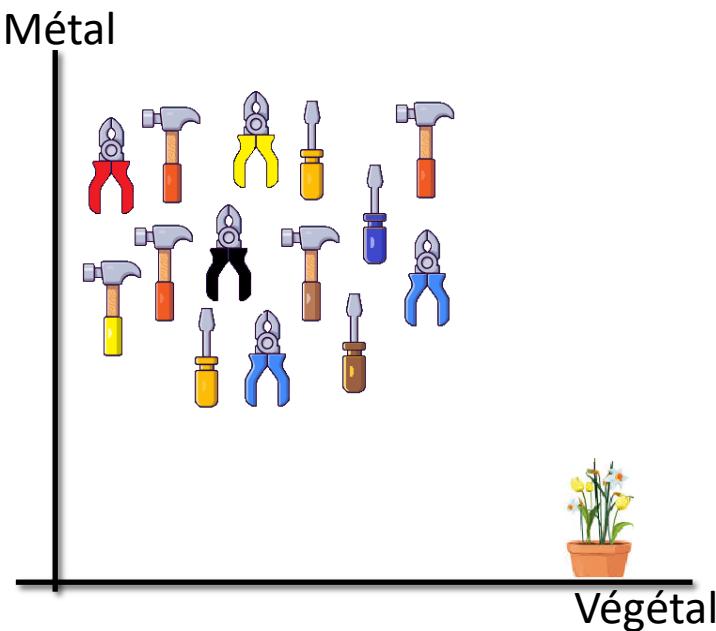
- Détection d'anomalie, la machine trouve des échantillons dans les caractéristiques x sont loin de celles des autres échantillons.
- Isolation Forest
- Reduction de la dimensionnalité = Comment

Non supervisé

Donnée X initiale :



Avec critère de détection :

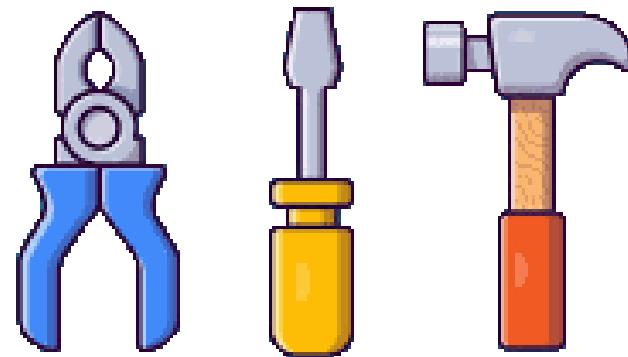


Détection d'anomalies :

- Algorithme : Isolation Forest

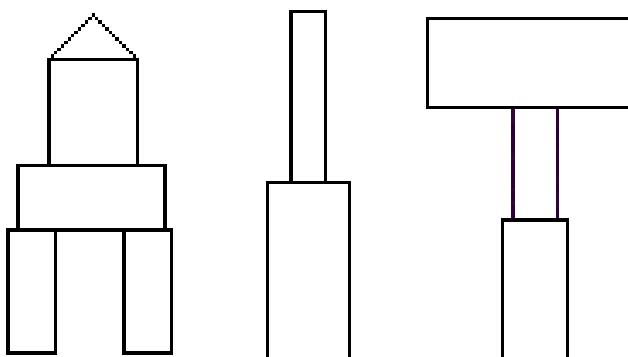
Non supervisé

Projection des données X dans des espaces et composantes de plus petites dimensions



Réduction de la dimensionnalité :

Algorithme : PCA (Principal Component Analysis)



Machine Learning par les modèles prédictifs

La régression Linéaire avec numpy :

- Le but est de faire une génération de données aléatoires avec une tendance linéaire en utilisant `make_regression`
- Le data Set (X,y) contient 200 exemples, et une seule variable x .
- Chaque fois que la cellule est exécutée, des données différentes sont générées

Machine Learning par les modèles prédictifs

```
import numpy as np
from sklearn.datasets import make_regression
import matplotlib.pyplot as plt

np.random.seed(0)
X, y = make_regression(n_samples=200, n_features=1, noise=16.0)
plt.scatter(X, y)
```

Machine Learning par les modèles prédictifs

Il faut toujours vérifier les dimensions de X et y, et il manque les dimensions (200, 1)

Il faut corriger le problème avec un redimensionnement : np.reshape

```
print(X.shape)
print(y.shape)

# redimensionner y
y = y.reshape(y.shape[0], 1)

print(y.shape)
```

Machine Learning par les modèles prédictifs

Création d'une matrice X_new qui aura la colonne de Biais.

Principe :

- Concaténer le vecteur X et un vecteur 1 (avec np.ones) de dimension égale a celle de x

```
X_new = np.hstack((X, np.ones(X.shape)))
print(X_new.shape)
```

Machine Learning par les modèles prédictifs

La première étape et la création d'un vecteur avec un paramètre θ (initialisé avec des coefficients aléatoires)

Ce vecteur est de dimension (2, 1)

Pour reproduire le même vecteur θ , nous utilisons le np.random.seed(0).

```
np.random.seed(0)  
theta = np.random.randn(2, 1)
```

Machine Learning par les modèles prédictifs

Pour le modèle linéaire nous $F=X.\theta$

Test du modèle, pour le modèle initial défini par la valeur de θ

```
def modele_Lineaire(X, theta):
    return X.dot(theta)

plt.scatter(X, y)
plt.plot(X, modele_Lineaire(X_new, theta), c='r')
```

Machine Learning par les modèles prédictifs

Nous construisons la fonction coût avec l'erreur quadratique moyenne :

- Mesure des erreurs du modèle sur le data set (X, y) et implémenter l'erreur quadratique moyenne avec Mean Squared Error (MSE)
- $$C(\theta) = \frac{1}{2n} \sum (X \cdot \theta - y)^2$$

```
def fonction_cout(X, y, theta):
    n = len(y)
    return 1/(2*n) * np.sum((modele_Lineaire(X, theta) - y)**2)
```

Machine Learning par les modèles prédictifs

La formule du gradient pour MSE est :

$$\circ \frac{\partial C(\theta)}{\partial \theta} = \frac{1}{n} X^T * (X * \theta - y)$$

```
def gradient(X, y, theta):
    n = len(y)
    return 1/n * X.T.dot(modele_Lineaire(X, theta) - y)
```

La fonction permet la descente de gradient:

$$\circ \theta = \theta - \alpha * \frac{\partial C(\theta)}{\partial \theta}$$

```
def descente_gradient(X, y, theta, ratio_apris, n_iterations):
    # création d'un tableau de stockage pour enregistrer l'évolution du Cout du modèle
    historique_cout = np.zeros(n_iterations)

    for i in range(0, n_iterations):
        # mise à jour du paramètre theta (formule du gradient descent)
        theta = theta - ratio_apris * gradient(X, y, theta)

        # Enregistrement de la valeur du Cout au tour i dans cost_history[i]
        historique_cout[i] = fonction_cout(X, y, theta)

return theta, historique_cout
```

Machine Learning par les modèles prédictifs

Pour la phase d'entraînement :

- Il faut un nombre d'itérations
- Un pas d'apprentissage α

```
n_iterations = 5000  
ratio_appris = 0.01
```

Entrainement du modèle :

```
theta_final, historique_cout = descente_gradient(X_new, y, theta, ratio_appris, n_iterations)  
predictions = modele_Lineaire(X_new, theta_final)
```

Machine Learning par les modèles prédictifs

Le résultat est observable :

```
plt.scatter(X, y)
plt.plot(X, predictions, c='r')
```

Pour vérifier l'algorithme de descente de gradient, il faut l'évolution de la fonction coût à travers les itérations

```
plt.plot(range(n_iterations), historique_cout)
```

La courbe diminue à chaque itération jusqu'à stagner à un niveau minimal (proche de zéro)

Si la courbe ne suivait pas ce motif, alors le pas historique_cout est peut-être trop élevé, il faut prendre un pas plus faible

Machine Learning par les modèles prédictifs

Pour évaluer la réelle performance du modèle, nous pouvons utiliser le coefficient de détermination de la méthode des moindres carrés: \mathbb{R}^2

```
def moindre_carrés(y, predictions):
    a = ((y - predictions)**2).sum()
    b = ((y - y.mean())**2).sum()
    return 1 - a/b
```

```
moindre_carrés(y, predictions)
```

Machine Learning par les modèles prédictifs

La régression logistique analyse la relation entre une variable dépendante (DV) et une variable indépendante (IV) lorsque le DV est dichotomique

DV est la variable de résultat = la variable prédictée

IV sont les variables considérées avoir une influence sur le résultat = variables prédictives

Régression logistique simple = le modèle contient 1 IV

Régression logistique multiple = le modèle contient 2 et + IV

Machine Learning par les modèles prédictifs

Hypothèses pour les modèles de régression logistique :

- La DV est catégorique (binaire)
- S'il existe plus de 2 catégories en termes de types de résultats, il convient d'utiliser une régression logistique multinomiale

- Indépendance des observations
 - Il ne peut s'agir d'une conception de mesures répétées, c'est-à-dire de la collecte de résultats à deux moments différents

Machine Learning par les modèles prédictifs

Exemple de Régression Logistique :

- Prédire et expliquer le diabète (variable à prédire) à partir des caractéristiques des personnes (âge, IMC, etc.) (variables explicatives)
- Fichier utilisé texte «EtudeDiabete.txt»
- Utilisation de la librairie Pandas pour récupérer les données du fichier .txt :

```
import pandas as pd  
file = pd.read_table("EtudeDiabete.txt", sep="\t", header=0)
```

Machine Learning par les modèles prédictifs

Liste des 9 caractéristiques dans le fichier :

- 'pregnant', 'diastolic', 'triceps', 'bodymass', 'pedigree', 'age', 'plasma', 'serum', 'diabète'

A partir des données d'un fichier .txt, nous les transformons en une donnée tableau numpy, pour récupérer les 8 premières et prédire la 9ième :

```
data = file.values
# X matrice des variables explicatives
X = data[:,0:8]
# y vecteur de la variable à prédire
y = data[:,8]
```

Machine Learning par les modèles prédictifs

Division des données pour avoir des variables d'apprentissage et de test :

```
from sklearn.model_selection import train_test_split  
  
Xtrain, Xtest, ytrain, ytest = train_test_split(X,y,test_size = 0.4,random_state=0)
```

Donnez la répartition des données en fonction de la division ?

Comment faire varier la division ?

Machine Learning par les modèles prédictifs

Solveur pour une regression logistique dans sklearn :

- 'liblinear', 'newton-cg', 'lbfgs', 'sag', 'saga'
- Choix du lineaire : liblinear

```
from sklearn.linear_model import LogisticRegression  
  
# Création d'une instance de la classe  
clf = LogisticRegression(solver='liblinear')
```

Machine Learning par les modèles prédictifs

A partir des données précédentes construction du modèle :

```
# Exécution de l'instance sur les données d'apprentissage  
# La construction du modèle prédictif  
m = clf.fit(Xtrain,ytrain)
```

Quelques données de statistiques sont possibles avec :

- Coef_
- Intercept_

Machine Learning par les modèles prédictifs

Prédiction sur l'échantillon de test :

```
ypred = m.predict(Xtest)
```

Importation des métriques pour les mesures de performances :

```
import sklearn.metrics as smt
```

Matrice de confusion :

```
matrice_c = smt.confusion_matrix(ytest, ypred)
```

Calcul du taux de succès :

```
succes = smt.accuracy_score(ytest, ypred)
```

Comment le taux de succès est-il calculé ?

Machine Learning par les modèles prédictifs

Donnez le taux d'erreur ?

Calcul du Rappel :

```
rappel = smt.recall_score(ytest, ypred, pos_label='positive')
```

Pour ajuster les modèles et demandes, il faut aussi programmer :

```
import numpy as np

# Ecriture de notre fonction d'évaluation - exemple celle de Rappel
def rappel(a,b):
    # Matrice de confusion de sklearn.metrics
    matrice_conf = smt.confusion_matrix(a,b)
    # Les négatifs sont à l'indice 0 dans la matrice
    resultat = matrice_conf[0,0] / np.sum(matrice_conf[0,:])
    return resultat

## Pour rendre utilisable la fonction rappel construite - transformation en objet scorer
rappel = smt.make_scorer(rappel,greater_is_better=True)

# Utilisation de l'objet scorer
# Remarque : m est le modèle élaboré sur l'échantillon d'apprentissage

new_rappel = rappel(m,Xtest,ytest)
```

Machine Learning par les modèles prédictifs

Lors du traitement de petit fichier (en nombre d'observation), le schéma d'apprentissage de test peut être pénalisé :

- Réduction des informations disponibles pour créer le modèle

Exercice :

- Construire le modèle sur la totalité des données
- Constaté qu'il existe une différence

Machine Learning par les modèles prédictifs

Réponse à l'exercice :

```
from sklearn.linear_model import LogisticRegression

# Création d'une instance de la classe
clf = LogisticRegression(solver="liblinear")

# Exécution de l'instance sur la totalité des données (X,y)
m_all= clf.fit(X,y)

# Affichage des résultats
print(m.coef_,m.intercept_)
print(m_all.coef_,m_all.intercept_)
```

Machine Learning par les modèles prédictifs

Il est donc possible d'évaluer les performances à l'aide des techniques de rééchantillonnage par la validation croisée :

```
import sklearn.model_selection as sms

# Evaluation en validation croisée : 10 cross-validation
succes = sms.cross_val_score(clf,X,y,cv=10,scoring='accuracy')

# Détail des itérations
print(succes)

# Moyenne des taux de succès = estimation du taux de succès en Cross Validation CV
print(succes.mean())
```

Machine Learning par les modèles prédictifs

Score et ciblage = construction d'une courbe de gain avec :

- Attribuer un score aux individus
- Trier de manière décroissante (score élevé = forte appétence au produit)
- Estimer à l'aide de la courbe de gain le nombre de positifs en fonction d'une taille de cible choisie

Machine Learning par les modèles prédictifs

```
# Calcul des probabilités d'affectation sur l'échantillon de test
# Probabilités d'affectation aux classes Negative, Positive
probas = clf.predict_proba(Xtest)

# Score des présences
score = probas[:,1]

# Transformation en 0/1 des positifs/négatifs de Ytest
# Classe d'appartenance Negative, Positive
positif = pandas.get_dummies(ytest).values

# Récupérations de la 2ième colonne (indice 1)
positif = positif[:,1]
# [ 1  0  0  1  0  0  1  1 ...]

# Nombre total de positif
import numpy as np
nb_positif = np.sum(positif)
# 103 individus positifs dans l'échantillon test
```

Machine Learning par les modèles prédictifs

```
# Index pour trier avec un score croissant
index = np.argsort(score)
# [ 55  45 265 271 276 162 ... 11  255  159]
# L'individu n°55 a le score le plus faible, suivi du n°45, ... , l'individu n°159 a le score le plus élevé

# Inverser le score pour avoir une décroissance pour avoir les plus forte probabilité en premier
index = index[::-1]

# Tri des individus (des valeurs 0/1)
positif_tri = positif[index]
# [ 1  1  1  1  1  0  1 1 ...] Les positifs sont en tête des données après le tri décroissant

# Somme cumulée
positif_cumul = np.cumsum(positif_tri)
# [ 1  2  3  4  5  5  67 ... 103]
```

Machine Learning par les modèles prédictifs

```
# Calcul du Rappel
rappel = positif_cumul / nb_positif
# [ 1/103  2/103  3/103  4/103  5/103  5/103  6/103  7/103 ... 103/103]
# Nombre des observations suivant l'échantillon de test
nombre = ytest.shape[0]

# 308 individus dans l'échantillon de test
# Séquence de valeurs de 1 à 308 avec un pas de 1
taille = np.arange(start=1,stop=rappel.shape[0]+1,step=1)
# [1  2  3  4  5  ...  308]

# En proportion
taille = taille / nombre
# [ 1/308  2/308  3/308 ... 308/308]
```

Machine Learning par les modèles prédictifs

```
# Graphique avec matplotlib
import matplotlib.pyplot as plt

# Titre et Légendes
plt.title('Courbe de gain')
plt.xlabel('Taille de cible')
plt.ylabel('Rappel')

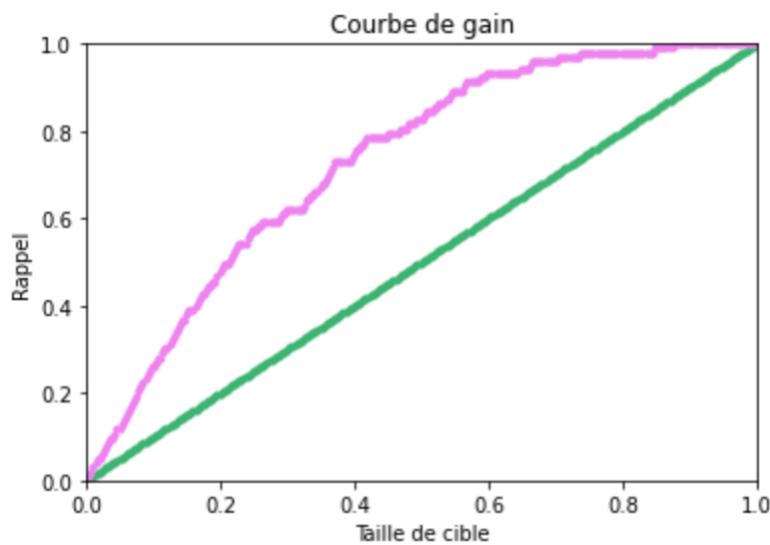
# Limites en abscisse et ordonnée
plt.xlim(0,1)
plt.ylim(0,1)

# La diagonale pour préparer ROC - AUC
plt.scatter(taille,taille,marker='.',c='mediumseagreen')

# Courbe couple (taille, rappel)
plt.scatter(taille,rappel,marker='.',c='violet')
```

Machine Learning par les modèles prédictifs

- La courbe de gain se calcule entre Rappel et taille :



Exercice avec SVM (reprise de l'exercice sur la régression Logistique) :

- Les algorithmes de Machine Learning ont des paramètres qui ne sont pas toujours facile à déterminer pour obtenir les meilleures performances avec les données à traiter
- Il est possible de faire plusieurs modèles

```
#svm
from sklearn import svm

# Le noyau RBF et C = 1.0
ker = svm.SVC()

# Modélisation
m = ker.fit(Xtrain,ytrain)

# Prédiction sur l'échantillon de test
ypred = m.predict(Xtest)

# Matrice de confusion
# Pas mieux que le classifieur par défaut (prédir systématiquement 'negative', classe majoritaire)
print(smt.confusion_matrix(ytest,ypred))

# Succès en test
print(smt.accuracy_score(ytest,ypred))
```

Après constat qu'il n'y a pas d'amélioration dans l'apprentissage et les questions sont :

- La méthode SVM est-elle inadaptée ?

- Le paramétrage est-il inadapté ?

```
# Import de la classe
import sklearn.model_selection as sk

# Combinaisons de paramètres à évaluer
parametres= [{"C":[0.1,1,10],'kernel':['rbf','linear']}]

# Evaluation en validation croisée de 3 x 2 = 6 configurations
# accuracy sera le critère à utiliser pour sélectionner la meilleure config
# ker est l'instance de la classe de svm.SVC
grid = sk.GridSearchCV(estimator=ker,param_grid=parametres,scoring='accuracy')

# Recherche longue car gourmande en calculs
grille = grid.fit(Xtrain,ytrain)

# Résultat pour chaque combinaison
print(pd.DataFrame.from_dict(grille.cv_results_).loc[:,["params","mean_test_score"]])
```

```
# Meilleur paramétrage
print(grille.best_params_)

# {'C' : 10, 'kernel' : 'linear'}
# Meilleur performance estimée pour la validation croisée
print(grille.best_score_)
# 0.7564

# Prédiction avec le modèle «optimal» : {'C' : 10, 'kernel' : 'linear'}
ytest = grille.predict(Xtest)

# Taux de succès en test
print(smt.accuracy_score(ytest,ypred))
```

La sélection de variables c'est la recherche de modèles pour plusieurs avantages :

- Interprétation
- Déploiement (moins de var. à renseigner)
- Performances en généralisation (ou du moins maintien des performances)

La méthode implémente celle de la RFE de scikit-learn :

- Elimine au fur et à mesure les coefficients les plus faibles en valeur absolue
- Une standardisation des variables est nécessaire
- S'arrête à la moitié ou pour un nombre spécifié de variables

```
# Importer la classe LogisticRegression
from sklearn.linear_model import LogisticRegression

# Création d'une instance de la classe
clf = LogisticRegression(solver="liblinear")

# Algorithme de sélection de var.
from sklearn.feature_selection import RFE
selecteur= RFE(estimator=clf)

# La recherche
solution = selecteur.fit(Xtrain,ytrain)

# Nombre de variables sélectionnées
print(solution.n_features_)
```

```
# Liste des variables sélectionnées 4 = 8 / 2 variables sélectionnées
print(solution.support_)
# Variables sélectionnées : pregnant, bodymass, pedigree, plasma.
# [True False False True True False True False]

# Ordre de suppression
print(solution.ranking_)
# [1 2 4 1 1 3 1 5]
# Serum a été retirée en premier, puis triceps, puis âge, puis diastolic.
# Les variables restantes sont indexées 1.
```

```
# Réduction de la base d'apprentissage aux variables sélectionnées
# En utilisant le filtre booléen sol.support_
X_new_train = Xtrain[:,solution.support_]

print(X_new_train.shape)
# (460, 4) → 4 variables restantes
# Construction du modèle sur les explicatives sélectionnées

m_solution = clf.fit(X_new_train,ytrain)
#réduction de la base test aux mêmes variables

X_new_test= Xtest[:,solution.support_]

print(X_new_test.shape)
# (308, 4)

# Prédiction du modèle réduit sur l'échantillon de test
ypred = m_solution.predict(X_new_test)
```

Question :

- Comparez les résultats avec ceux faits en régression logistique ?

Exercices

Exercice : Avec les données d'iris, appliquer à :

- Naïve Bayes
- K plus proches voisins
- Arbre de décision

Etape 1 : Récupérer les données de Iris dans le dataSets de sklearn

Etape 2 : Visualisez graphiquement les données

Etape 3 : Faire 2 modèles de prédictions Gaussiennes en Naïve Bayes

Etape 4 : Visualiser les prédictions Gaussiennes en Naïve Bayes

Exercices

Etape 6 : Faire 2 visualisations graphiques des Iris mal classés en Naïve Bayes

Etape 7 : Faire une prédiction en $K = 6$ plus proches voisins

Etape 8 : Faire une prédiction avec plusieurs valeurs de K

Etape 9 : Visualiser les prédictions en K plus proches voisins

Etape 10 : Faire une prédiction avec les arbres de décisions

Etape 11 : Visualiser la prédiction avec les arbres de décisions

Exercices

Réponse possible étape 1 :

Les données Iris

```
from sklearn.datasets import load_iris  
irisData = load_iris()
```

```
print(irisData.DESCR)
```

```
print(irisData.data)
```

```
print(irisData.target)
```

Exercices

Visualiser les données avec un graphique

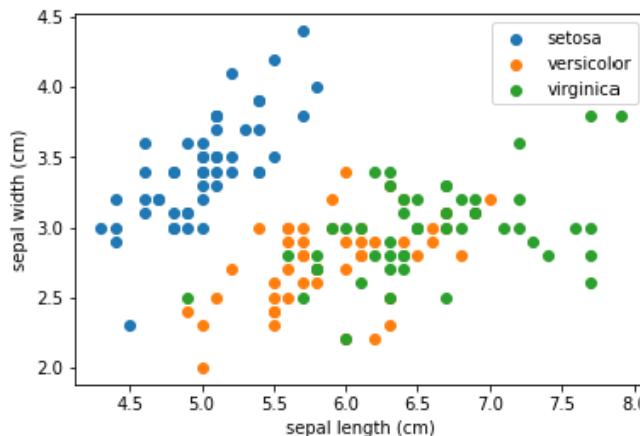
Réponse possible étape 2 :

```
import matplotlib.pyplot as plt

X = irisData.data
Y = irisData.target

abscisse = 0
ordonnee = 1

plt.xlabel(irisData.feature_names[abscisse])
plt.ylabel(irisData.feature_names[ordonnee])
for i in range(3):
    plt.scatter(X[Y==i][:, abscisse], X[Y==i][:, ordonnee], label=irisData.target_names[i])
plt.legend()
plt.show()
```



Exercices

Réponse possible étape 3 :

Naïve Bayes - Modèle 1

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split

classifieur = GaussianNB()

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.20)
# Apprentissage sur les données d'entraînement :
classifieur.fit(X_train, y_train)

# Utilisation du classifieur appris sur les données de test :
y_predits = classifieur.predict(X_test)
```

Exercices

Réponse possible étape 3 :

Naïve Bayes - Modèle 2

```
from sklearn.naive_bayes import GaussianNB  
naive_bayes = GaussianNB()  
naive_bayes.fit(X,Y)  
Y_predit = naive_bayes.predict(X)
```

Exercices

Réponse possible étape 4 – Modèle 1 – Naïve Bayes :

```
print(len(X_test[y_test != y_pred]), "est le nombre d'observations mal classées sur un total d'observations de ", len(X))

from sklearn.metrics import accuracy_score
print("Taux de réussite : ", accuracy_score(y_test,y_pred))
```

Réponse possible étape 4 – Modèle 2 – Naïve Bayes :

```
print(len(X[Y != Y_predit]), "est le nombre d'observations mal classées sur un total d'observations de ", len(X))

from sklearn.metrics import accuracy_score
print("Taux de réussite : ", accuracy_score(Y,Y_predit))
```

Exercices

Réponse possible étape 5 :

```
abscisse = 2
ordonnee = 3

plt.xlabel(irisData.feature_names[abscisse])
plt.ylabel(irisData.feature_names[ordonnee])
for i in range(3):
    plt.scatter(X_train[y_train==i][:, abscisse],X_train[y_train==i][:,ordonnee],label=irisData.target_names[i])

print(y_test)
print(y_predits)
plt.scatter(X_test[y_test != y_predits][:, abscisse],X_test[y_test != y_predits][:,ordonnee],color='black')
plt.legend()
plt.show()
```

```
abscisse = 2
ordonnee = 3

plt.xlabel(irisData.feature_names[abscisse])
plt.ylabel(irisData.feature_names[ordonnee])
for i in range(3):
    plt.scatter(X[Y==i][:, abscisse],X[Y==i][:,ordonnee],label=irisData.target_names[i])

plt.scatter(X[Y != Y_predit][:, abscisse],X[Y != Y_predit][:,ordonnee],color='black')
plt.legend()
plt.show()
```

Exercices

Réponse étape 6,7 :

K plus proches voisins

```
from sklearn.neighbors import KNeighborsClassifier  
kppv = KNeighborsClassifier(n_neighbors=6)  
kppv.fit(X,Y)  
Y_predit = kppv.predict(X)
```

Réponse étape 8 :

```
print(len(X[Y != Y_predit]), "est le nombre d'observations mal classées sur un total d'observations de ", len(X))
```

Exercices

Réponse étape 9 :

Arbre de décision

```
from sklearn.tree import DecisionTreeClassifier  
arbre_decision = DecisionTreeClassifier()  
arbre_decision.fit(X,Y)  
Y_predit = arbre_decision.predict(X)
```

Réponse étape 10 :

Visualisation de l'arbre de décision

```
from sklearn import tree  
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)  
tree.plot_tree(arbre_decision)
```

Exercices

Réponse étape 11 :

```
fn=['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
cn=['setosa', 'versicolor', 'virginica']
fig, axes = plt.subplots(nrows = 1, ncols = 1, figsize = (4,4), dpi=300)
tree.plot_tree(arbre_decision, feature_names = fn, class_names=cn, filled = True);
fig.savefig('arbre_decision.png')
```

• Chapitre 4 : Arbre de décision

- [IV-1 Description](#)
- [IV-2 ID3](#)
- [IV-3 C4.5](#)
- [IV-4 CART](#)
- [IV-5 Avantages](#)
- [IV-6 Limites](#)
- [IV-7 Un peu de programmation](#)

IV-1 Description

Un arbre de décision est un modèle très simple. Etant donnée plusieurs caractéristiques, la décision se commence par un de ces caractéristiques ; si ce n'a pas suffisant, on utilise une autre, ainsi de suite. Il est largement connu et utilisé dans de nombreuses entreprises pour faciliter le processus de prise de décision et l'analyse des risques. Il a été largement utilisé dans les années 1960-1980 pour la construction de systèmes experts. Les règles sont introduites manuellement, pour cette raison ce modèle a perdu sa popularité après les années 80. L'apparition des méthodes mathématiques pour construire les arbres de décision fait revenir ce modèle à la bataille des algorithmes de l'apprentissage automatique.

Il existe plusieurs algorithmes automatiques pour construire les arbres de décision :

- **ID3** (Itérative Dichotomiser 3): développé en 1986 par Ross Quinlan. Il peut être appliqué seulement sur les caractéristiques nominales. Il est utilisé pour le classement.
- **C4.5**: une extension de ID3 par Ross Quinlan. Il peut être appliqué sur tous les types de caractéristiques. Il est utilisé pour le classement.
- **C5.0**: une extension commerciale de C4.5, toujours par Ross Quinlan.
- **CART** (Classification and Regression Trees): comme C4.5 mais utilise d'autres métriques. Aussi, l'algorithme supporte la régression.

L'algorithme général de création d'un arbre de décision :

1. Déterminer la meilleure caractéristique dans l'ensemble de données d'entraînement.
2. Diviser les données d'entraînement en sous-ensembles contenant les valeurs possibles de la meilleure caractéristique.
3. Générer de manière récursive de nouveaux arbres de décision en utilisant les sous-ensembles de données créés.
4. Lorsqu'on ne peut plus classifier les données, on s'arrête.

IV-2 ID3

L'algorithme ne fonctionne que sur des caractéristiques nominales. Donc, si on a des caractéristiques continues, il faut appliquer la discréétisation. Aussi, il est utilisé pour le classement seulement.

IV-2-1 Sélectionner la meilleure caractéristique

Cet algorithme utilise la fonction entropie et le gain d'information pour décider quelle est la meilleure caractéristique. Etant donnée un ensemble de classes **C**, l'entropie d'ensemble de donnée **S** est exprimée par :

$$H(S) = \sum_{c_i \in C} -P(c_i) \log_2 P(c_i)$$

Où (en divisant le nombre des échantillons d'une certaine classe sur le nombre de tous les échantillons dans les données d'entraînement) :

$$P(c_i) = \frac{|c_i|}{|S|}$$

Etant donnée un vecteur de caractéristiques \vec{f} , en utilisant les valeurs d'une caractéristique f_j , on peut diviser l'ensemble de donnée S en plusieurs sous-ensembles groupés dans un ensemble S_j . Le gain d'information est mesuré en se basant sur la différence entre l'entropie originale de S et celle après sa division en se basant sur une caractéristique f_j .

$$IG(S, f_j) = H(S) - \sum_{S_{jk} \in S_j} P(S_{jk}) H(S_{jk})$$

Où :

$$P(S_{jk}) = \frac{|S_{jk}|}{|S|}$$

La caractéristique ayant plus de gain d'information est celle sélectionnée comme meilleure. Aussi, la valeur avec entropie nulle est considérée comme feuille de l'arbre.

IV-2-2 Exemple

On veut estimer une décision (jouer ou non) en se basant sur 4 caractéristiques : temps, température, humidité et vent. On va construire un arbre de décision en se basant sur les données suivantes :

Temps	Température	Humidité	Vent	Jouer
Ensoleillé	chaude	haute	non	Non
Ensoleillé	chaude	haute	oui	Non
Nuageux	chaude	haute	non	Oui
Pluvieux	douce	haute	non	Oui
Pluvieux	fraîche	normale	non	Oui
Pluvieux	fraîche	normale	oui	Non
Nuageux	fraîche	normale	oui	Oui
Ensoleillé	douce	haute	non	Non
Ensoleillé	fraîche	normale	non	Oui
Pluvieux	douce	normale	non	oui
Ensoleillé	douce	normale	oui	oui
Nuageux	douce	haute	oui	oui
Nuageux	chaude	normale	non	oui
Pluvieux	douce	haute	oui	non

On calcule la probabilité de chaque classe :

- $P(\text{jouer}=\text{oui}) = 9/14$
- $P(\text{jouer}=\text{non}) = 5/14$

On calcule, ensuite, l'entropie de l'ensemble des données :

$$H(S) = - P(\text{jouer}=\text{oui}) * \log_2(P(\text{jouer}=\text{oui})) - P(\text{jouer}=\text{non}) * \log_2(P(\text{jouer}=\text{non}))$$

$$H(S) = - 9/14 * \log_2(9/14) - 5/14 * \log_2(5/14)$$

$$H(S) = 0.41 + 0.53 = 0.94$$

Pour chaque caractéristique, on calcule le gain d'information.

temps :

Le caractéristique “temps” divise les données sur 3 sous-ensembles. Voici le nombre des occurrences de chaque classe dans chaque sous-ensemble :

Temps	Jouer (oui)	Jouer (non)
Ensoleillé	2	3
Nuageux	4	0
Pluvieux	3	2

On calcule la probabilité de chaque ensemble :

- $P(S_{\text{ensoleillé}}) = 5/14$

- $P(S_{\text{nuageux}}) = 4/14$
- $P(S_{\text{pluvieux}}) = 5/14$

On calcule l'entropie de chaque ensemble :

- $H(S_{\text{ensoleillé}}) = - 2/5 * \log_2(2/5) - 3/5 * \log_2(3/5) = 0.971$
- $H(S_{\text{nuageux}}) = - 4/4 * \log_2(4/4) - 0/4 * \log_2(0/4) = 0$
- $H(S_{\text{pluvieux}}) = - 3/5 * \log_2(3/5) - 2/5 * \log_2(2/5) = 0.971$

Le gain d'information de la caractéristique "temps":

$$IG(S, \text{temps}) = H(S) - P(S_{\text{ensoleillé}}) * H(S_{\text{ensoleillé}}) - P(S_{\text{nuageux}}) * H(S_{\text{nuageux}}) - P(S_{\text{pluvieux}}) * H(S_{\text{pluvieux}})$$

$$IG(S, \text{temps}) = 0.94 - 5/14 * 0.971 - 4/14 * 0 - 5/14 * 0.971$$

$$IG(S, \text{temps}) = 0.247$$

En calculant le gain d'information des autres caractéristiques

	temps	température	humidité	vent
IG	0.247	0.029	0.152	0.048

Donc, la première caractéristique à vérifier dans l'arbre sera "temps". Comme l'entropie du temps étant "nuageux" est 0, cet ensemble contient des échantillons de la même classe. Donc, cet ensemble forme une feuille.

Division des données selon la caractéristique "temps"

On fait la même chose sur les sous-ensembles.

L'arbre de décision en utilisant ID3

IV-3 C4.5

Cet algorithme est une amélioration sur l'algorithme ID3. Parmi les améliorations :

- Transformer les caractéristiques continues (numériques) en caractéristiques nominales dynamiquement.
- Les caractéristiques sans valeurs sont ignorées lors du calcul de l'entropie et le gain d'information.
- Élagage des arbres après la création.

IV-3-1 Sélectionner la meilleure caractéristique

Pour décider quelle est la meilleure caractéristique afin de diviser l'ensemble de données, C4.5 utilise une extension du gain d'information connue par **rapport de gain**. Lorsqu'on a une caractéristique ait un grand nombre de valeurs, elle sera favorisée par le gain d'information. Le rapport de gain fait face au problème de biais en normalisant le gain d'informations à l'aide de l'information de division.

Etant donné :

- C : un ensemble de classes
- S : un ensemble de données d'entraînement
- \vec{f} : un vecteur de caractéristiques
- S_j : un hyper-ensemble contenant des ensembles avec les mêmes valeurs de la caractéristique f_j .
- $IG(S, f_j)$: le gain d'information en divisant l'ensemble de données S avec la caractéristique f_j .

L'information de dévision SI (Split Information) peut être calculée comme suit :

$$SI(S, f_j) = - \sum_{S_{jk} \in S_j} \frac{|S_{jk}|}{|S|} * \log_2\left(\frac{|S_{jk}|}{|S|}\right)$$

Et le rapport de gain GR (Gain Ratio) est calculé comme suit :

$$GR(S, f_j) = \frac{IG(S, f_j)}{SI(S, f_j)}$$

Donc, la caractéristique avec le plus grand rapport de gain sera prise comme caractéristique de division.

IV-3-2 Traitement des caractéristiques continues

ID3 ne supporte pas les caractéristiques avec des valeurs continues ; comme l'âge, le prix, etc. C4.5 introduit le support de ce type de caractéristiques en cherchant le meilleur seuil qui peut diviser l'ensemble des valeurs d'une caractéristique en deux.

Afin de sélectionner la bonne division, on suit l'algorithme suivant à chaque fois qu'on veuille comparer une caractéristique avec d'autres :

- Pour chaque valeur V_{jk} d'une caractéristique f_j
 - Diviser l'ensemble de données S en deux sous-ensembles : les données avec $f_j > V_{jk}$ et celles avec $f_j \leq V_{jk}$

- Calculer le rapport de gain GR de cet ensemble en le divisant avec un seuil V_{jk} sur la caractéristique f_j
- La valeur qui maximise le rapport de gain est prise comme seuil de dévision

IV-3-3 Élagage des arbres (pruning)

Pour éviter le sur-apprentissage (créer un arbre avec une grande profondeur), on peut utiliser la technique d'élagage. Il existe deux types d'élagage :

- Pré-élagage : utiliser des critères d'arrêt de la division. Par exemple : nombre minimum des échantillons dans un nœud, un taux d'homogénéité d'un sous-ensemble.
- Post-élagage : construire l'arbre, ensuite éliminer les branches qui n'améliorent pas la performance de l'arbre.

Voici l'algorithme de post-élagage utilisé par C4.5:

- Construire l'arbre de décision
- Transformer l'arbre à un ensemble de règles de la forme (Si [préconditions] Alors [résultat]) en traversant l'arbre depuis la racine jusqu'à une feuille.
- Supprimer les préconditions qui n'améliorent pas la performance d'une règle.
- Arranger les règles élaguées en se basant sur leurs performances. On commence par la règle qui donne plus de performance et on se termine par celle qui donne moins.

IV-4 CART

L'algorithme CART est similaire à celui de C4.5 avec quelques différences :

- Il supporte la régression.
- Il utilise d'autres critères pour sélectionner la meilleure caractéristique. Il essaye de minimiser une fonction de coût.
- Il utilise le pré-élagage en utilisant un critère d'arrêt.
- Il crée des arbres binaires (je ne sais pas comment il le fait avec des caractéristiques nominales ayant plusieurs valeurs).

IV-4-1 Sélectionner la meilleure caractéristique

Dans le cas de CART, la meilleure caractéristique est celle qui minimise une fonction de coût $E(S)$. Chaque caractéristique f_j doit diviser un ensemble de données sur deux sous-ensembles : gauche (S_g) et droit (S_d). Donc, celle avec la plus petite valeur de l'Indice de diversité sera choisie.

$$ID(S, f_j) = \frac{|S_G|}{|S|} * E(S_G) + \frac{|S_D|}{|S|} * E(S_D)$$

Dans le cas de classement, CART utilise l'index de diversité Gini pour mesurer l'erreur de classification. Etant donnée un ensemble de classes C , la fonction Gini Index de l'ensemble de donnée S est exprimée par :

$$E(S) = \sum_{c_i \in C} P(c_i)(1 - P(c_i)) = 1 - \sum_{c_i \in C} P^2(c_i)$$

Où (en divisant le nombre des échantillons d'une certaine classe sur le nombre de tous les échantillons dans les données d'entraînement):

$$P(c_i) = \frac{|c_i|}{|S|}$$

Dans le cas de régression, on utilise la somme des carrés résiduelle. Etant donné un ensemble de données S , la somme des carrés résiduelle est calculée par :

$$E(S) = \frac{1}{|S|} \sum_{i=1}^{|S|} (y_i - \bar{y})^2$$

Où y_i sont les valeurs attendues, et

$$\bar{y} = \frac{1}{|S|} \sum_{i=1}^{|S|} y_i$$

est la valeur de sortie estimée.

IV-4-2 Élagage des arbres (pruning)

CART utilise le pré-élagage qu'on le réfère par la condition d'arrêt. Le critère le plus utilisé pour arrêter la division est le nombre minimal des échantillons dans un nœud. Si on atteint ce nombre, on ne divise plus et on considère le nœud comme feuille avec la classe dominante comme classe de sortie en cas de classement, ou la moyenne des sorties en cas de régression.

Aussi, on peut appliquer un post-élagage. La méthode la plus simple est de supprimer une feuille et de tester la performance sur des données de validation. Tant qu'il n'y a pas une chute de performance, on continue l'opération.

IV-5 Avantages

Parmi les avantages des arbres de décision :

- Ils sont simples à comprendre et à interpréter. On peut visualiser les arbres. Aussi, on peut expliquer les résultats obtenus facilement.
- Ils peuvent travailler sur des données avec peu de préparation. Par exemple, ils n'ont pas besoin de la normalisation des données.
- Ils acceptent les données numériques et nominales. Les autres algorithmes d'apprentissage sont spécialisés dans un seul type de données.
- Ils donnent de bonne performance même si leurs hypothèses sont un peu violées par le modèle réel à partir duquel les données ont été générées.

IV-6 Limites

Parmi les avantages des arbres de décision :

- Ils peuvent être aussi complexes, ils ne généralisent pas bien (overfitting: surapprentissage). On peut régler ça en fixant le nombre minimum des échantillons dans les feuilles ou en fixant la profondeur maximale de l'arbre.
- Ils peuvent être instable à cause des variations des données.
- Il existent des concepts qui sont un peu difficile à apprendre par les arbres de décision. Ils ne sont pas faciles à exprimer, par exemple : XOR.
- Ils peuvent être biaisés à la classe dominante. Donc, il faut balancer les données avant d'entrainer le système.
- Ce n'ai pas garanti de tomber sur l'arbre de décision optimal.

IV-7 Un peu de programmation

Il existe des outils pour construire des arbres de décision. Dans cette démo, on utilise toujours [Scikit-learn](#):

- Langage : python.
- Algorithmes : CART.
- Type de classification : classement et régression.
- Limites : ne supporte pas les caractéristiques nominales.
- Visualisation : oui (en utilisant [Graphviz](#)).

Concernant ID3, on peut construire un module (comme l'algorithme est simple), ou on peut chercher des modules sur internet ; par exemple [decision-tree-id3](#).

IV-7-1 Le classement (ID3)

On va reprendre l'exemple précédent : décider de jouer en se basant sur des caractéristiques nominales. Consulter le fichier [data/jouer0.csv](#)

temps	temperature	humidite	vent	jouer
ensoleile	chaude	haute	non	non
ensoleile	chaude	haute	oui	non
nuageux	chaude	haute	non	oui
pluvieux	douce	haute	non	oui
pluvieux	fraiche	normale	non	oui
pluvieux	fraiche	normale	oui	non
nuageux	fraiche	normale	oui	oui
ensoleile	douce	haute	non	non
ensoleile	fraiche	normale	non	oui
pluvieux	douce	normale	non	oui
ensoleile	douce	normale	oui	oui
nuageux	douce	haute	oui	oui
nuageux	chaude	normale	non	oui
pluvieux	douce	haute	oui	non

Les données sont sauvegardées sous format CSV ([data/jouer0.csv](#)). On va utiliser **pandas** pour lire le fichier.

```
import pandas
#lire le fichier csv
data = pandas.read_csv("../data/jouer0.csv")
```

On sépare les données en : entrées (les caractéristiques) et sorties (les classes: comestible ou toxique). Dans ce fichier, les classes (qui sont le résultat attendu) sont dans la dernière colonne, et les autres caractéristiques (les entrées) sont dans les colonnes restantes.

```
# séparer les données en: entrées et sorties
X = data.iloc[:, :-1] #les caractéristiques
y = data.iloc[:, -1] #les résultats (classes)
```

On va utiliser un outil sur Github: <https://github.com/svaante/decision-tree-id3>. On a modifié ce programme pour qu'il accepte les caractéristiques comme chaînes de caractères (string). Le code modifié est fourni avec ce tutorial: [codes/arbre/](#).

```
# importer l'estimateur
from id3a import Id3Estimator
# créer un estimateur
estimator = Id3Estimator()
# entraîner l'estimateur
estimator.fit(X, y)
```

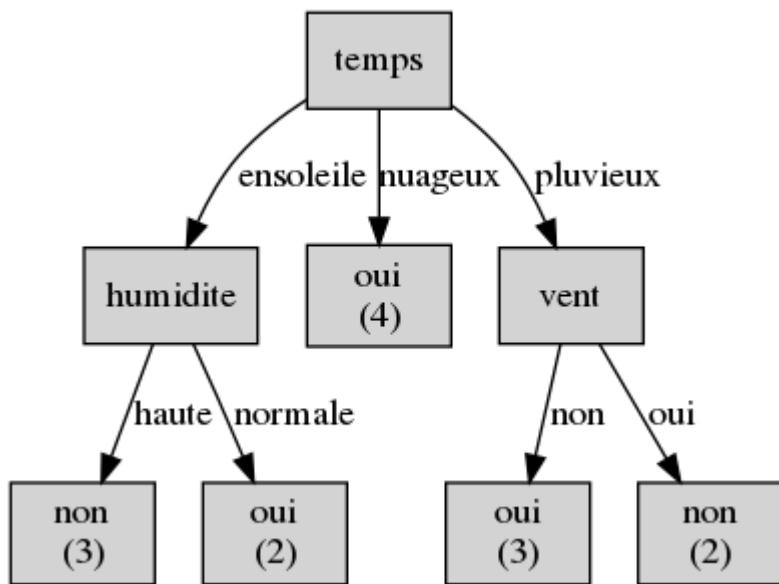
On peut exporter l'arbre sous forme Graphviz.

```
from id3a import export_graphviz
# expoter l'arbre sous format graphviz
export_graphviz(estimator.tree_, "resultat.dot", data.columns.tolist())
```

On peut visualiser ce fichier en utilisant [Graphviz](#). Il faut l'installer sur machine. Aussi, on peut exporter ce fichier vers une image PNG.

```
dot -Tpng resultat.dot -o resultat.png
```

Le résultat sera :



Si vous voulez implémenter cet algorithme de zéro, voici quelques outils que vous pouviez utiliser :

- [scipy.stats.entropy](#)

temps	température	humidité	vent	jouer
ensoleillé	30	85	non	non
ensoleillé	27	90	oui	non
nuageux	28	78	non	oui
pluvieux	21	96	non	oui
pluvieux	20	80	non	oui
pluvieux	18	70	oui	non
nuageux	18	65	oui	oui
ensoleillé	22	95	non	non
ensoleillé	21	70	non	oui

temps	température	humidité	vent	jouer
pluvieux	24	80	non	oui
ensoleillé	24	70	oui	oui
nuageux	22	90	oui	oui
nuageux	27	75	non	oui
pluvieux	22	80	oui	non

IV-7-2 Le classement (CART)

Retenant l'exemple précédent avec une petite modification : on utilise des caractéristiques continues.

Consulter le fichier [data/jouer.csv](#).

On commence toujours par la lecture du fichier en utilisant **pandas**. Et, bien sûr, séparer les données d'entrée et la sortie.

```
import pandas

#lire le fichier csv
data = pandas.read_csv("../data/jouer.csv")

# séparer Les données en: entrées et sorties
X = data.iloc[:, :-1] #les caractéristiques
y = data.iloc[:, -1] #les résultats (classes)
```

Pour construire un arbre de classement, on va utiliser [`sklearn.tree.DecisionTreeClassifier`](#). L'outil ne supporte pas les caractéristiques nominales : "temps" et "vent" dans notre cas. Dans ce cas, on va les encoder comme des caractéristiques numériques. Il existe une bibliothèque pour encoder les caractéristiques nominales comme numériques. Vous pouvez consulter <https://github.com/scikit-learn-contrib/categorical-encoding>.

IV-7-2-1 Encodage One Hot (Pandas)

On va utiliser l'encodage **One Hot** fourni par [`pandas.get_dummies`](#).

```
X_dum = pandas.get_dummies(X)

# imprimer Les premières lignes des données
print X_dum.head()
```

Lorsqu'on imprime les données avec les nouvelles colonnes, ça va nous donner:

```
temperature  humidite  temps_ensoleile  temps_nuageux  temps_pluvieux
vent_non    vent_oui
```

0	30	85	1	0	0	1
0						
1	27	90	1	0	0	0
1						
2	28	78	0	1	0	1
0						
3	21	96	0	0	1	1
0						
4	20	80	0	0	1	1
0						

On utilise le classifieur fourni par **Scikit-learn**.

```
from sklearn.tree import DecisionTreeClassifier

# créer un estimateur
estimator = DecisionTreeClassifier()
# entraîner l'estimateur
estimator.fit(X_dum, y)
```

On peut exporter l'arbre sous forme Graphviz en utilisant la fonction [sklearn.tree.export_graphviz](#). Ici, on va donner comme paramètres: le classifieur, le fichier de sortie, les noms des colonnes et les noms des classes.

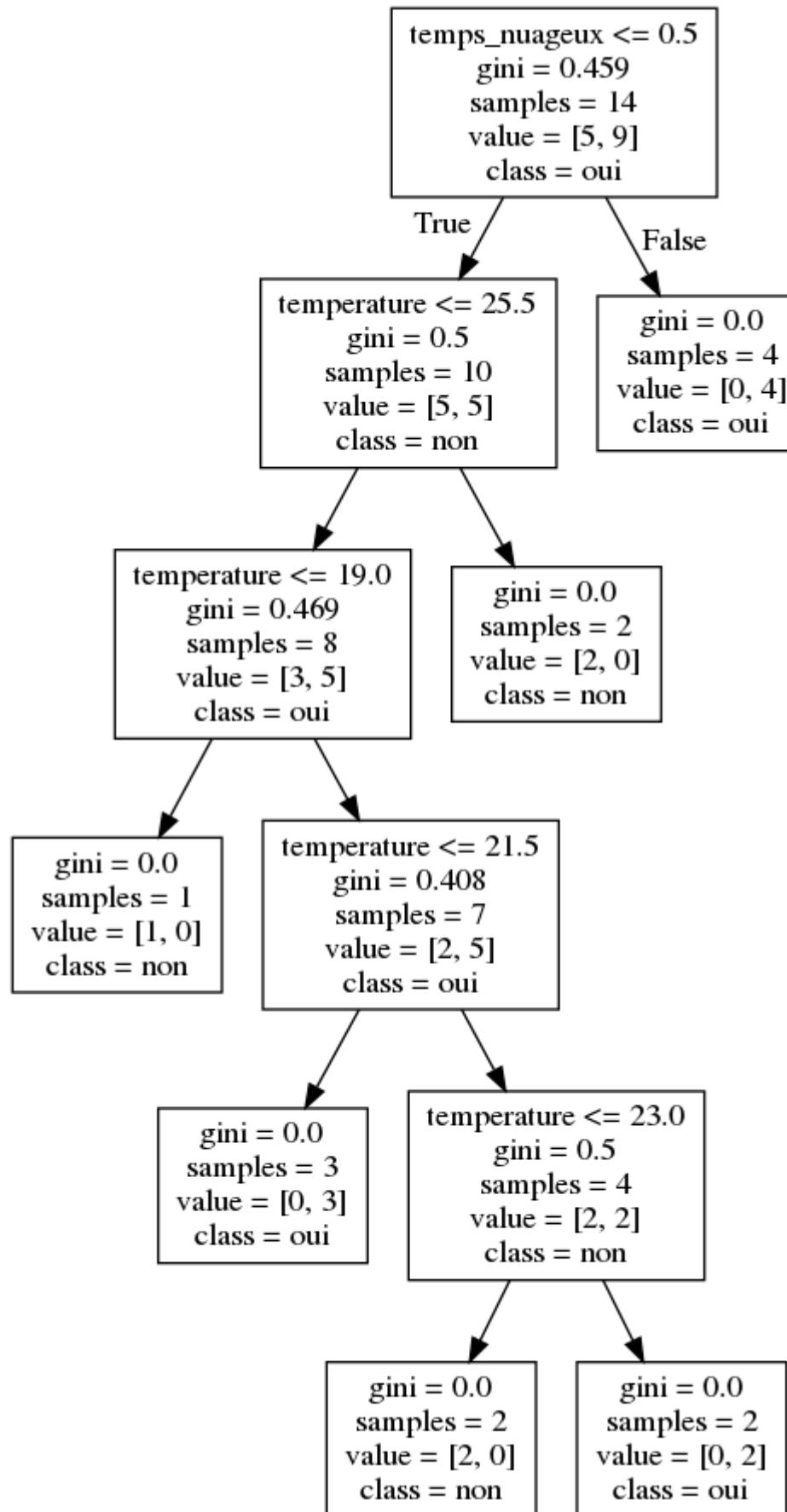
```
from sklearn.tree import export_graphviz

# exporter l'arbre sous format graphviz
export_graphviz(estimator,
    out_file="arbre_cart0.dot",
    feature_names = X_dum.columns,
    class_names=estimator.classes_)
```

On peut visualiser ce fichier en utilisant [Graphviz](#). Il faut l'installer sur machine. Aussi, on peut exporter ce fichier vers une image PNG.

```
dot -Tpng arbre_cart0.dot -o arbre_cart0.png
```

Le résultat sera :



IV-7-2-2 Encodage DictVectorizer (Scikit-learn)

On va utiliser [sklearn.feature_extraction.DictVectorizer](#). On transforme les données stockées sous forme d'un **DataFrame** vers une liste des dictionnaires (tableaux associatifs). Cette structure peut, ensuite, être transformée en utilisant **DictVectorizer**. Enfin, on transforme la structure résultante vers un **DataFrame**.

```
from sklearn.feature_extraction import DictVectorizer

# Transformer X à une Liste de dicts
list_dicts = X.T.to_dict().values()
# créer une instance du transformateur
vec = DictVectorizer()
# transformer
new_list_dicts = vec.fit_transform(list_dicts).toarray()
# créer un nouveau DataFrame
X_vec = pandas.DataFrame(new_list_dicts, columns=vec.get_feature_names())

# imprimer Les premières lignes des données
print X_vec.head()
```

L'opération donne le même résultat que **pandas.get_dummies**.

	humidite	temperature	temps=ensoleile	temps=nuageux	temps=pluvieux	
vent=non	vent=oui					
0	85.0	30.0	1.0	0.0	0.0	1.0
0.0						
1	90.0	27.0	1.0	0.0	0.0	0.0
1.0						
2	78.0	28.0	0.0	1.0	0.0	1.0
0.0						
3	96.0	21.0	0.0	0.0	1.0	1.0
0.0						
4	80.0	20.0	0.0	0.0	1.0	1.0
0.0						

Le reste est le même qu'avant.

IV-7-3 La régression (CART)

Pour la régression, on utilise [sklearn.tree.DecisionTreeRegressor](#) de la même façon. La seule chose différente est que les résultats y sont numériques.