|  |
| --- |
| EPSI EISI 1A |
| MACHINE LEARNING |
| MSPR1 |

|  |
| --- |
| Dimitri  31/01/2025 |

**Table des matières**

[1 INTRODUCTION 4](#_Toc193639945)

[1.1 IA vs Machine learning 4](#_Toc193639946)

[1.1.1 Intelligence artificielle 4](#_Toc193639947)

[1.1.2 Machine learning 4](#_Toc193639948)

[1.1.3 Comparaison IA et machine learning 4](#_Toc193639949)

[1.2 Apprentissage supervisé vs Apprentissage non supervisé 5](#_Toc193639950)

[1.3 But apprentissage supervisé 6](#_Toc193639951)

[1.4 But apprentissage non supervisé 6](#_Toc193639952)

[1.5 Apprentissage partiellement supervisé 7](#_Toc193639953)

[1.6 Choisir entre apprentissage supervisé et non supervisé 7](#_Toc193639954)

[1.7 Régression VS Classification 7](#_Toc193639955)

[2 PROBABILISTE VS DETERMINISTE 8](#_Toc193639956)

[3 INITIATION MACHINE LEARNING OPENCLASSROOMS 9](#_Toc193639957)

[3.1 Introduction 9](#_Toc193639958)

[3.1.1 Modélisation statistique vs modélisation prédictive 9](#_Toc193639959)

[3.1.2 Variables 9](#_Toc193639960)

[3.1.3 Bruit 10](#_Toc193639961)

[3.1.4 Nature itérative 10](#_Toc193639962)

[3.2 Modèles et algorithmes 11](#_Toc193639963)

[3.2.1 Définition modèle 11](#_Toc193639964)

[3.2.2 Choix 12](#_Toc193639965)

[3.2.3 Modèles de régression 12](#_Toc193639966)

[3.2.4 Modèles à base d’arbre 13](#_Toc193639967)

[3.2.5 Réseaux de neurones et Deep Learning 13](#_Toc193639968)

[3.2.6 Algorithme 13](#_Toc193639969)

[3.3 D’une problématique business à la mise en production 14](#_Toc193639970)

[3.3.1 Déroulement 14](#_Toc193639971)

[3.3.2 Phase 1 : Définir les spécifications à partir de la problématique 15](#_Toc193639972)

[3.3.3 Phase 2 : Concevoir le prototype et valider la faisabilité du projet 16](#_Toc193639973)

[3.3.4 Phase 3 : Mise en production 17](#_Toc193639974)

[3.4 Quizz 18](#_Toc193639975)

[3.5 Evaluer son premier modèle prédictif 21](#_Toc193639976)

[3.5.1 Description dataset 21](#_Toc193639977)

[3.5.2 Objectif du modèle 22](#_Toc193639978)

[3.5.3 Entrainement 22](#_Toc193639979)

[3.5.4 Evaluer la performance d’un modèle 23](#_Toc193639980)

[3.6 Régression linéaire 23](#_Toc193639981)

[3.6.1 Un ou plusieurs variables 23](#_Toc193639982)

[3.6.2 Limites de la régression 24](#_Toc193639983)

[3.6.3 Améliorations d’un modèle 24](#_Toc193639984)

[3.6.4 Validation croisée 25](#_Toc193639985)

[3.7 Régression logistique 25](#_Toc193639986)

[3.7.1 Différents types 25](#_Toc193639987)

[3.7.2 Rapport entre classification et régression 26](#_Toc193639988)

[3.7.3 Concepts de la classification binaire 27](#_Toc193639989)

[3.7.4 Evaluer les performances du modèle 27](#_Toc193639990)

[3.8 K-means 31](#_Toc193639991)

[3.8.1 Généralités 31](#_Toc193639992)

[3.8.2 Comment ? 31](#_Toc193639993)

[3.8.3 Algorithme du k-means 31](#_Toc193639994)

[3.8.4 Evaluer la performance du modèle k-means 32](#_Toc193639995)

[3.8.5 Coefficient de silhouette 32](#_Toc193639996)

[3.9 Arbre de décision 33](#_Toc193639997)

[3.9.1 Géénralités 33](#_Toc193639998)

[3.10 Transformer les jeux de données 35](#_Toc193639999)

[3.10.1 Rôle central du jeu de données 35](#_Toc193640000)

[3.10.2 Où trouver ses données 36](#_Toc193640001)

[3.10.3 Créer son propre dataset 36](#_Toc193640002)

[3.10.4 Réduire influence des préjugés dans prédictions du modèle 37](#_Toc193640003)

[3.11 Améliorer un jeu de données 37](#_Toc193640004)

[3.11.1 Repérer les données manquantes 37](#_Toc193640005)

[3.11.2 Détecter les outliers 37](#_Toc193640006)

[3.11.3 Que faire des outliers 39](#_Toc193640007)

[3.11.4 Normaliser / standardiser les valeurs numériques 39](#_Toc193640008)

[3.11.5 Récapitulatif des transformations 40](#_Toc193640009)

[3.12 Transformer les variables 40](#_Toc193640010)

[3.12.1 Traitement des catégories textuelles 40](#_Toc193640011)

[3.13 Améliorer le modèle 42](#_Toc193640012)

[3.13.1 Reconnaitre un modèle qui sous-performe 42](#_Toc193640013)

[3.13.2 Exemple 43](#_Toc193640014)

[3.13.3 Minimiser le biais du modèle 43](#_Toc193640015)

[3.13.4 Identifier et compenser l’overfit 44](#_Toc193640016)

[3.14 Augmenter la robustesse du modèle 45](#_Toc193640017)

[3.14.1 Problème de la multicolinéarité 45](#_Toc193640018)

[3.14.2 Régularisation 46](#_Toc193640019)

[3.14.3 Validation croisée 48](#_Toc193640020)

[3.14.4 Résumé 49](#_Toc193640021)

[3.15 Ensemble learning (Random forests) 50](#_Toc193640022)

[3.15.1 Bagging et forêts aléatoires 50](#_Toc193640023)

[3.15.2 Boosting (appliqué aux arbres) 53](#_Toc193640024)

[4 Chatou 55](#_Toc193640025)

[5 Références 57](#_Toc193640026)

# INTRODUCTION

## IA vs Machine learning

### Intelligence artificielle

* Utilisation de technologies pour créer des machines et des ordinateurs capables d’imiter des fonctions cognitives associées à l’intelligence humaine
  + Visualisation
  + Compréhension langage humain
  + Analyse de données
  + Proposer des recommandations
* Pas un système en soi, il s’agit d’un ensemble de technologies implémentées dans un système pour lui permettre de raisonner, d’apprendre et de résoudre un problème complexe (1)
* IA est un terme général, regroupant différents types d’algos :
  + Machine learning
  + Deep learning
  + Robotique
  + Systèmes experts
  + Traitement du langage naturel

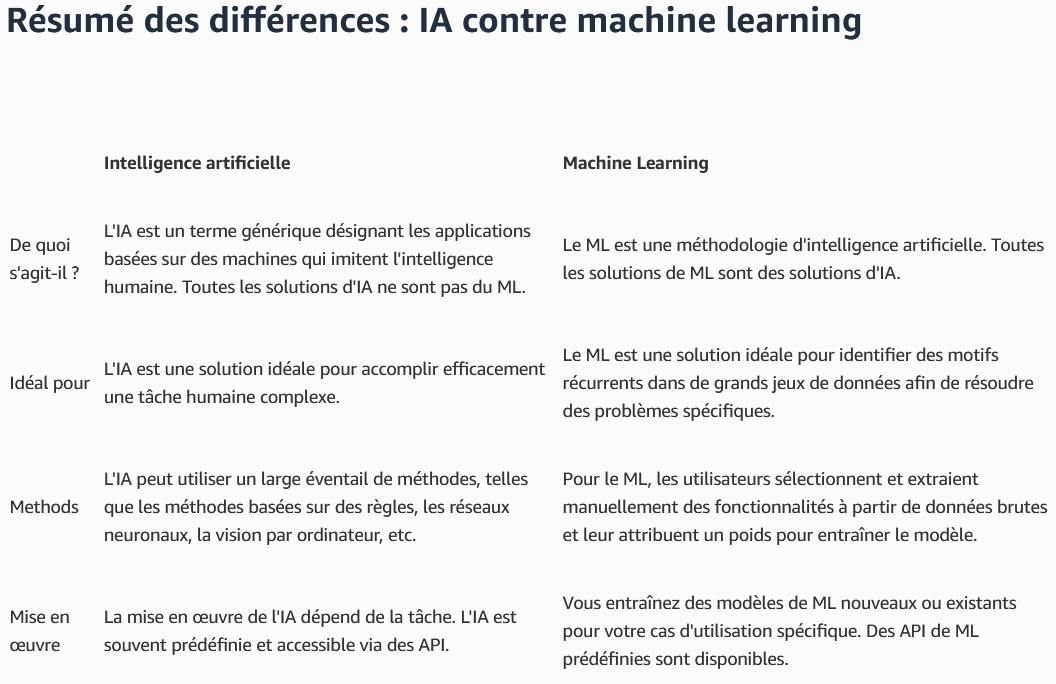
### Machine learning

* Apprentissage automatique est un sous-ensemble de l’intelligence artificielle
* Permet aux machines ou systèmes d’apprendre et de s’améliorer automatiquement grâce à l’expérience.
* Au lieu de programmation explicite, le machine learning utilise des algorithmes pour analyser de grandes quantités de donner, tirer des enseignements des insights et prendre des décisions.
* Le **modèle** de machine learning sont à considérer comme le **résultat** de l’apprentissage
  + Ce que le programme apprend de l’exécution d’un algorithme sur des données d’entraînement

### Comparaison IA et machine learning

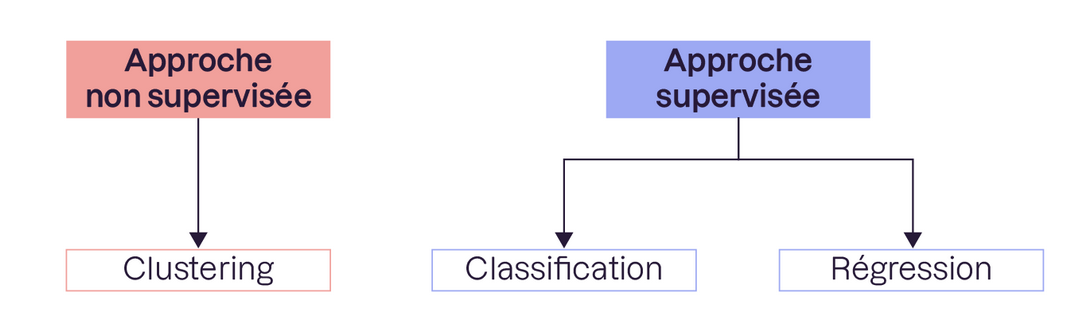
IA englobe l’idée d’une machine de simuler l’intelligence humain.

Machine learning vise à apprendre à une machine à effectuer une tâche spécifique et à fournir des résultats précis en identifiant des modèles.



(2)

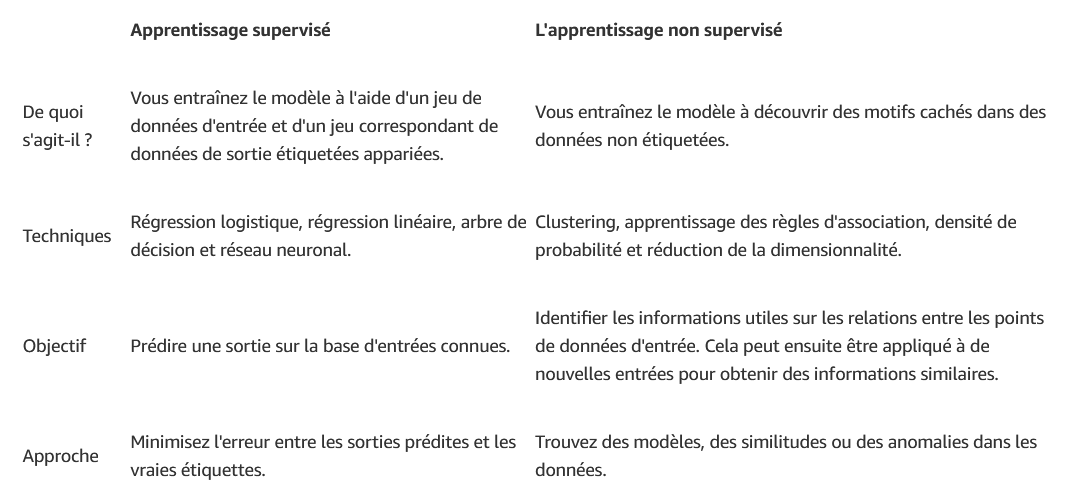
## Apprentissage supervisé vs Apprentissage non supervisé



La principale différence entre le machine learning supervisé et non supervisé réside dans le type de données utilisé (3).

* Apprentissage supervisé :
  + Données étiquetées
  + Ont une compréhension de base de ce que des valeurs de sortie correctes devraient être
  + Algo utilise un échantillon d’ensemble de données pour s’entraîner à faire des prédictions, et s’ajuste de manière itérative afin de minimiser les erreurs.
  + Ces données sont étiquetées à des fins de contexte et fournissent les valeurs de sortie souhaitées pour permettre au modèle de donner une réponse « correcte ».
* Apprentissage non supervisé :
  + Fonctionnent indépendamment pour apprendre la structure inhérente des données, sans conseil ni instructions spécifiques.
  + On fournit des données d’entrée non étiquetées et on laisse l’algo identifier tout modèle naturellement présent dans l’ensemble de données

Ces deux types de machine learning sont déployés pour résoudre différents types de problèmes.



(4)

## But apprentissage supervisé

* Adapté aux taches de **classification** et de **régression**
  + Prévisions météorologiques
  + Changement de prix
  + Analyse des sentiments
  + Détection de spam
* Axé sur l’apprentissage des RELATIONS entre les données d’entrée et de sortie
  + Ex : utile pour prédire les heures de vol en fonction de paramètres spécifiques
    - Conditions météorologiques
    - Trafic aéroportuaire
    - Heures de pointe
* Classification
  + On prédit des catégories
    - Binaire (oui / non)
    - Ordinales (petit / moyen / grand)
    - Nominales (chat / chien / brebis)

## But apprentissage non supervisé

* Apprentissage non supervisé plus couramment utilisé pour l’analyse exploratoire des données et les **tâches de clustering**
  + Détection d’anomalie
  + Visualisation BIG DATA
  + Segmentation de la clientèle
* Utile pour DECOUVRIR de nouveaux modèles et relations entre des données brutes et non étiquetées
  + Ex : identifier des groupes d’acheteurs qui achètent ensemble des produits associés afin de proposer d’autres articles à recommander à des clients similaires

## Apprentissage partiellement supervisé

* Combine des aspects de l’apprentissage supervisé et non supervisé
  + Utilise des données étiquetées et non pour entraîner un modèle prédictif
    - Petite quantité de données étiquetées pour entrainer modèle initial, qui peut être utilisé pour prédire des étiquettes sur une plus grande quantité de données non étiquetées.
    - Le modèle est ensuite appliqué de manière itérative aux données étiquetées initialement et aux données avec étiquettes prédites (pseudo-étiquettes).

## Choisir entre apprentissage supervisé et non supervisé

Dépend des objectif et exigences globaux, des cas d’utilisation que l’on souhaite résoudre

* Les données sont-elles étiquetées ?
  + Apprentissage supervisé nécessite des données étiquetées
  + Evaluer si organisation dispose du temps, des ressources et de l’expertise nécessaire pour valider et étiqueter les données
* Quels sont les objectifs ?
  + Déterminer type de problème que l’on essaie de résoudre.
    - On souhaite créer un modèle de prédiction ou
    - Découvrir de nouveaux insights ou schémas cachés dans les données
* Quel type d’algorithme a-t-on besoin ?
  + Est-ce que les algorithmes peuvent prendre en charge le volume de données et correspondre aux dimensions requises (comme le nombre de fonctionnalités et d‘attributs)

## Régression VS Classification

* Attention, Il ne faut pas confondre les modèles dits « de régression » et la tâche de régression
  + Tâche de régression
    - Le modèle prédit une variable continue
  + Modèle de régression
    - Modèle statistique
  + Chaque classe de modèle peut être utilisée pour
    - La régression
      * Prédire variable continue
    - La classification
      * Prédire variable discrète

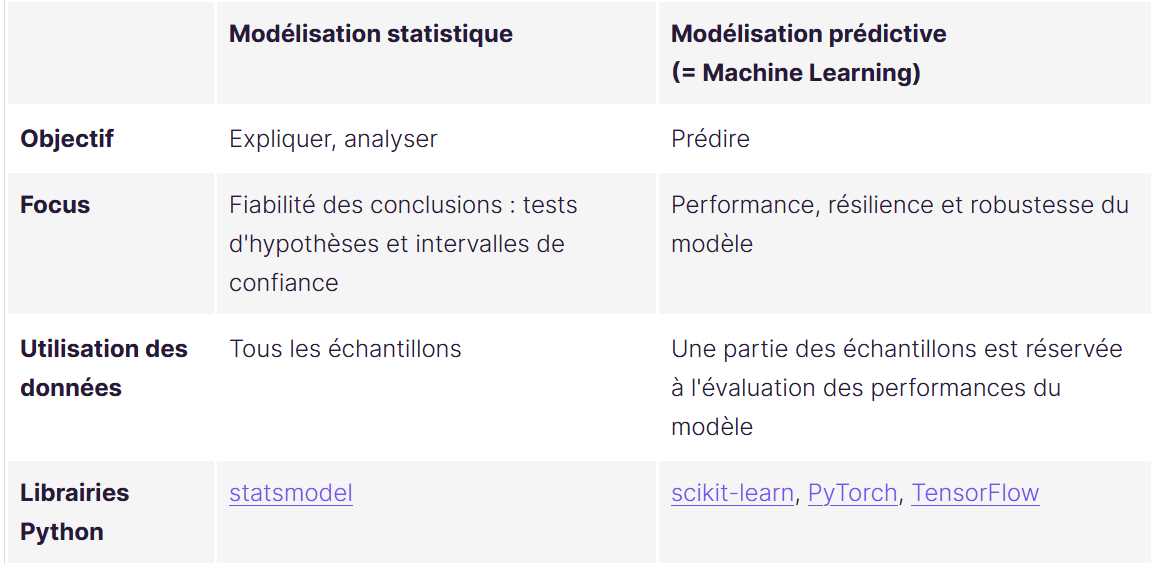
# PROBABILISTE VS DETERMINISTE

# INITIATION MACHINE LEARNING OPENCLASSROOMS

## Introduction

### Modélisation statistique vs modélisation prédictive

* Modélisation statistique
  + Démarche d’
    - Analyse
    - Interprétation
  + Expliquer une variable à partir des dynamiques des variables
    - Tests d’hypothèses + modèles mathématiques
    - Conclusion de l’analyse
* Modélisation prédictive
  + Prédiction de qualité grâce à une métrique choisie au préalable
  + Capacité à extrapoler
    - Généraliser ses prédictions
    - Robustesse du modèle face à de nouveaux échantillons
  + On ne cherche pas à expliquer la logique conduisant à une prédiction donnée
    - Fonctionnement interne trop complexe pour être interprétable par un humain
      * Shap (5) et Lime (6) permettent d’expliquer



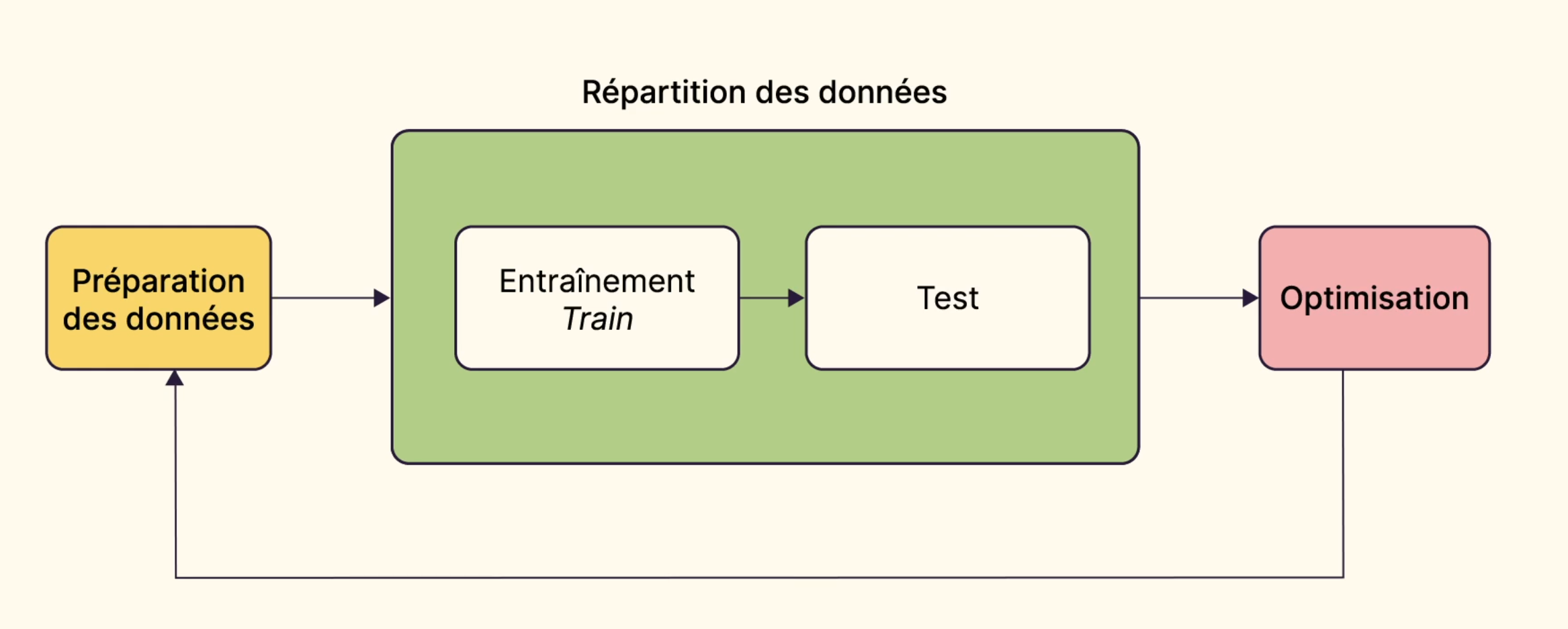
### Variables

* Variable cible
  + Sujet de la prédiction
* Variables prédictives
* Jeu de donnée exploitable
  + Suppose que les variables aient une relation entre elles
  + Le dataset n’est pas un regroupement totalement aléatoire de données qui auraient été collectées on ne sait comment

### Bruit

* Bruit
  + Représente l’information qui n’est pas capturée par le modèle
  + Poids = fct(age, taille) + bruit

### Nature itérative



#### Préparation des données

* Data Cleaning
  + Identifier et corriger toutes les anomalies présentes dans le data set
    - Données manquantes
    - Données aberrantes
  + Normalisation de données
    - Harmonisation des données en les ramenant à une échelle commune
  + Numérisation
    - Conversion de texte, image, catégorie en chiffre

#### Validation croisée

* Être capable de faire des prédictions sur des données pas encore rencontrées
  + Séparation des données en Train (80%) et Test (20%)
  + Existe même données conservées uniquement pour la validation après l’entrainement et l’optimisation
    - Evite d’optimiser le modèle sur le data set
    - *Au fil des expériences, on risque d'optimiser manuellement le modèle vis-à-vis du dataset de test. Donc pour éviter cela, on va scinder le dataset en non pas 2 mais 3 parties pour être bien sûr qu’une partie des données ne soit jamais visible par le modèle. On a donc au préalable à la validation croisée réservé une partie dite de* ***validation*** *sur laquelle on ne regardera les performances du modèle qu'une fois celui-ci totalement entraîné et optimisé. Cependant, dans certains domaines d'application, comme par exemple la prédiction des stocks, on veut être absolument sûr que cette séparation soit absolue, et donc on n’a qu'un shot pour utiliser le dataset de validation. Une fois cette cartouche utilisée, on ne veut plus toucher au modèle de peur de le corrompre. Mais c'est toutefois un cas extrême de séparation de l'évaluation et de l'entraînement d'un modèle.*
* Répéter ce découpage plusieurs fois et de façon aléatoire pour s’assurer que le modèle performe dans tous les cas de répartition / test
  + Validation croisée

#### Optimisation

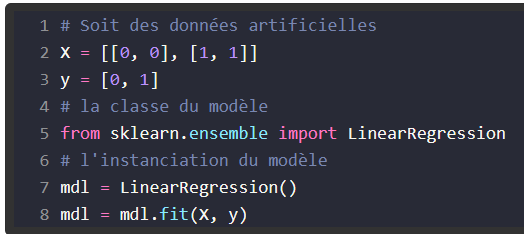
* Améliorer la performance du modèle en modifiant ses paramètres et en observant son score sur chaque version train / test du dataset
* Chaque type de modèle a sa propre famille de paramètres qui dépend des librairies utilisées

## Modèles et algorithmes

### Définition modèle

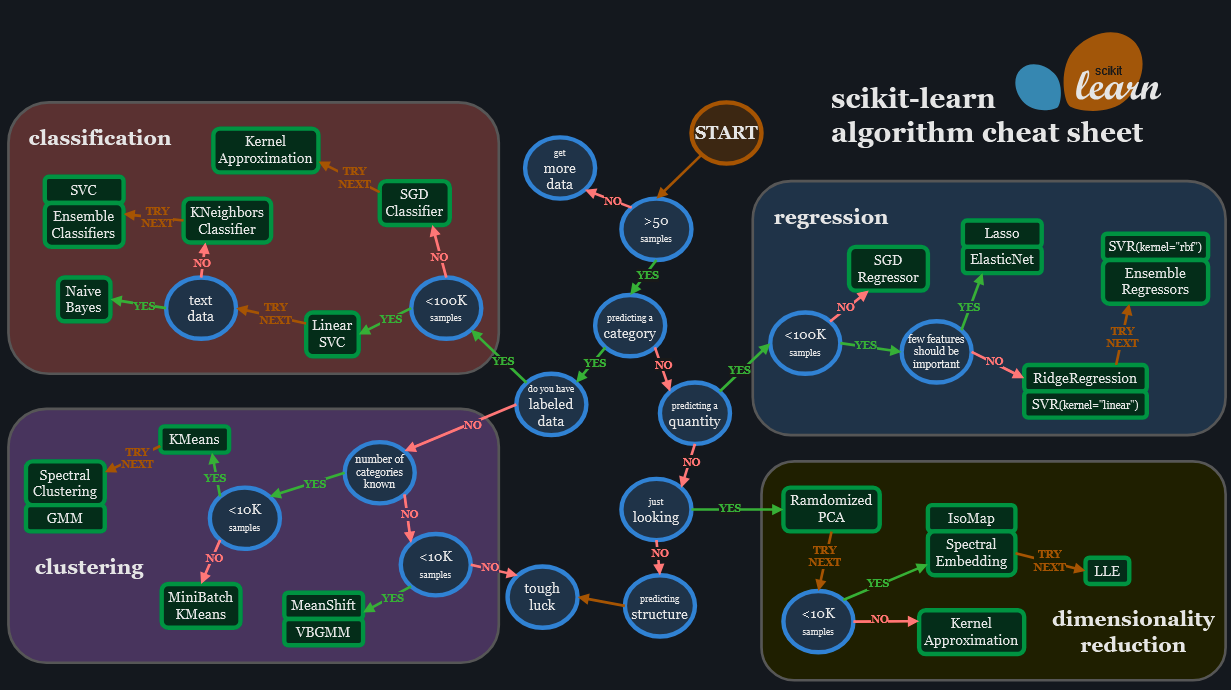
* Type du modèle
  + Méthode issue d’un raisonnement mathématique et traduite en code
    - XGBoost, NN, LR
* Instanciation après entraînement du modèle sur les données

Concrètement :

* Type du modèle
  + En Python, importé via une librairie
    - Exemple modèle de régression linéaire
    - 
* Instanciation du modèle après entraînement
  + En Python, utiliser la méthode fit()
    - 

### Choix

* En fonction de
  + La tâche à effectuer
    - Régression
    - Classification
    - Clustering
  + Nature des données
    - Chiffres
    - Catégories
    - Texte
    - Images
    - Sons
    - Séries temporelles
  + Volume des données
  + Caractéristiques statistiques des données
    - Linéarité
    - Stationnarité
    - Amplitude
    - Complétude
    - Cohérence

 (7)

* Aujourd’hui on distingue 3 classes principales
  + Les modèles de régression
  + Les modèles à base d’arbres
  + Les modèles de neurones et le Deep Learning

### Modèles de régression

* Adaptés à jeux de données
  + De taille limitée
    - 100 à 10 000 (info chatou)
  + Où les relations entre les variables sont linéaires (au moins jusqu’à un certain point)
* On parle aussi de Generalized Linear Model (GLM)
  + Famille de modèles qui comprend
    - La régression linéaire sous toutes ses formes
    - Des variantes pour la classification

### Modèles à base d’arbre

* Différents types
  + Arbre de décision (decision tree)
  + Forêt aléatoire (random forest)
  + XGBoost (et ses nombreuses variantes)
* Modèles adaptés à des jeux de données
  + Tabulaires
    - Chaque ligne correspond à une observation
    - Chaque colonne correspond à une variable
  + De taille plus conséquente
    - 10 000 à 100 000 (info chatou)
      * On commence à tirer avantage des arbres
    - 100 000 +
      * Forêts aléatoires et XGBoost deviennent plus performants que les modèles de régression classique, surtout si les relations entre variables sont non linéaires
    - Milliards
      * XGBoost et d’autres algos parallélisables (dépend des ressources matérielles RAM, GPU, etc.)
  + Robustes face aux données manquantes et observations aberrantes (**outliers**)

### Réseaux de neurones et Deep Learning

* Adaptés à des jeux de données
  + De gros volume
    - Gains de performances significatifs lorsqu’appliqués à de gros volume de données
  + De nature complexe (plus polyvalents dans la nature des données prises en compte)
    - Images ou vidéos

### Algorithme

* Algorithme
  + Série d’instructions définies et ordonnées qui permettent d’effectuer une tâche
* Algorithme d’entrainement
  + Notion clé
    - Erreur d’estimation
      * Différence entre
        + La valeur estimée
        + La valeur de vérité (groundtruth)
    - **Root Mean Square Error** (RMSE) (racine de l’erreur quadratique moyenne)
      * Carré des différences entre valeur réelle et valeur estimée
* Algorithme fréquemment utilisé en ML (DL inclus)
  + **Gradient stochastique** (stochastique = aléatoire en mathématique)
    - On cherche à estimer un vecteur h qui minimise un critère que l’on appelle une **fonction de coût** (= RMSE ?)
    - On se dote d’un paramètre, le **learning rate** (taux d’apprentissage) α
      * Permet de régler l’intensité de la correction de l’estimation apportée par l’erreur d’estimation
        + Vitesse de convergence de l’algorithme
    - De façon simplifiée
      * Calcul de l’erreur à partir de l’estimation et des valeurs réelles
        + Erreur = f(estimation, groundtruth)
      * Mise à jour de l’estimation
        + Estimation = estimation \* α \* f(erreur)
      * Stop en fonction d’un seuil d’erreur minimal
    - En pratique
      * Phrase d’initialisation du vecteur h
        + Par exemple, uniquement des zéros
      * A chaque itération, tant que l’erreur est assez grosse
        + Sélection des échantillons (aléatoire ou sous-échantillonnage)
        + Calcul de l’erreur d’estimation en fonction de h : e = f(h)
        + Maj de h en fonction de l’erreur h = h + α \* e
      * Le paramètre sert à régler la rapidité de convergence de l’algorithme et la précision des résultats
        + Si α petit

Convergence lente mais meilleure précision

* + - * + Si α grand

Convergence rapide mais résultats moins précis

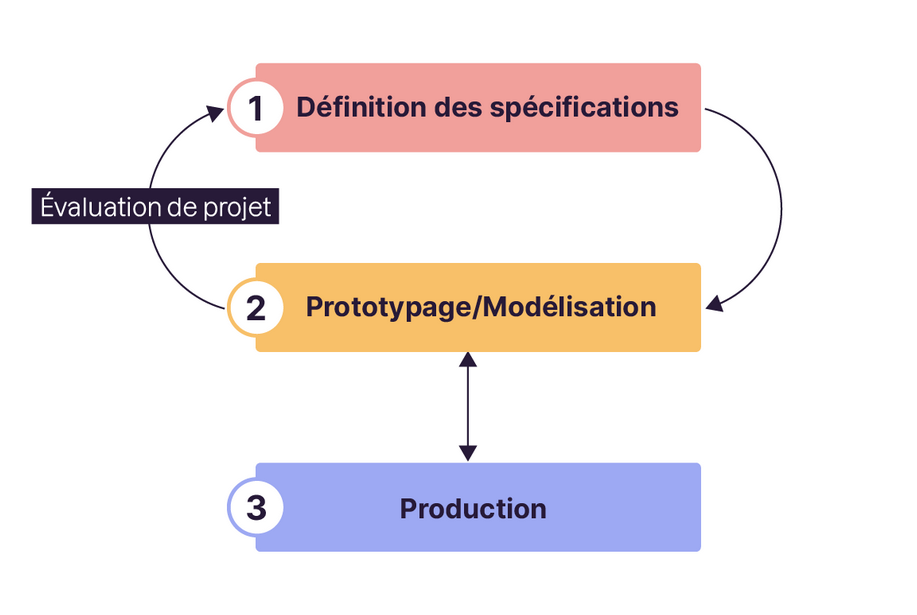
* + - * + Si α trop grand

Explosion et non convergence

## D’une problématique business à la mise en production

### Déroulement

* 3 grandes phases
  + Définition
  + Prototypage
  + Production



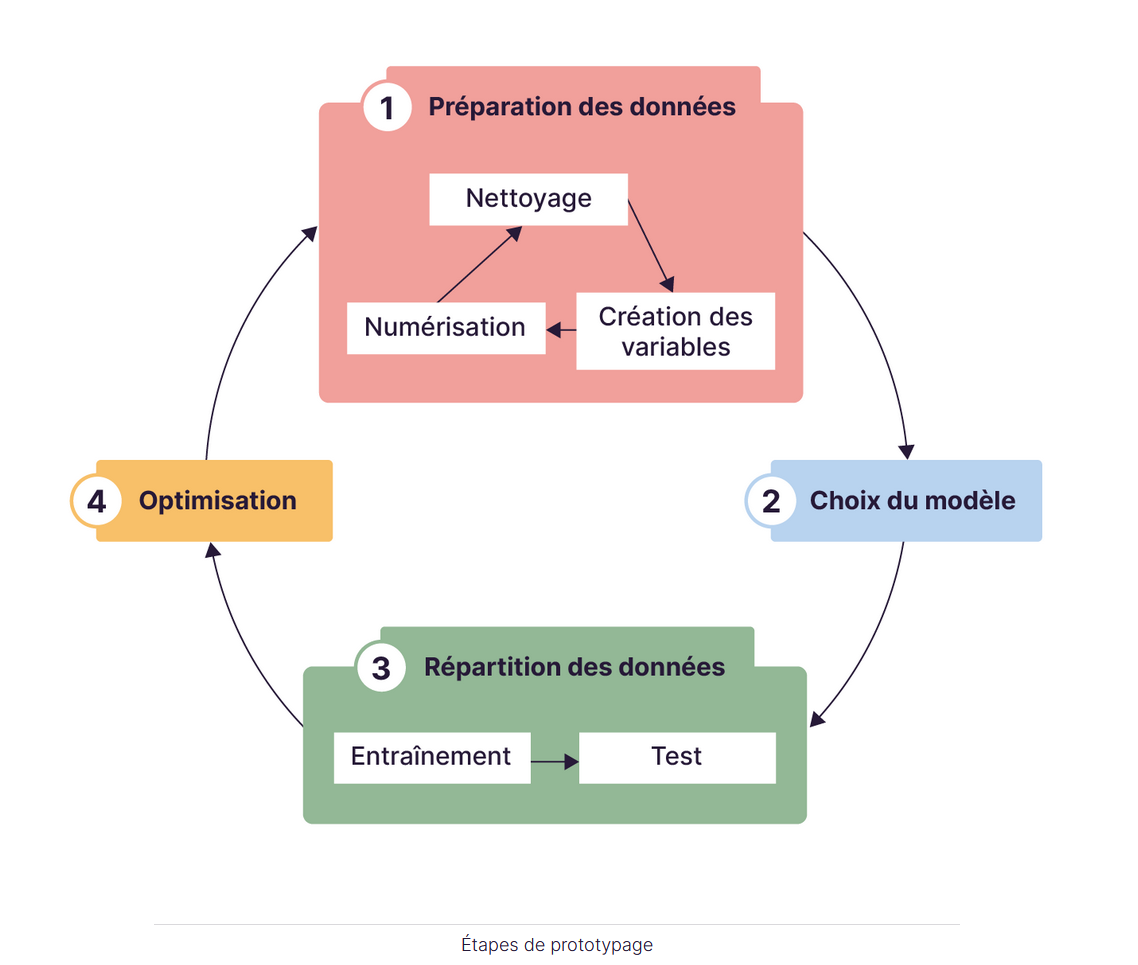
### Phase 1 : Définir les spécifications à partir de la problématique

Il faut

* Des données pertinentes
* Un sujet / produit clairement défini
* Montrer qu’il y a un net avantage à exploiter l’approche prédictive plutôt qu’une solution plus simple
  + Rule base solution (solution basée sur des règles)
  + Projet de ML est complexe : il faut calculer le gain réellement par la démarche ML
    - Réaliser en premier lieu une étude de benchmark
      * Comment définir le succès du projet ?
      * Comment mesurer la performance du système ?
      * Quelle métrique utiliser pour le scoring du modèle ?
    - Exemples de benchmark pour comparer
      * Prédiction météo
        + On prend temps de la veille
      * Prédiction du prix d’une course taxi
        + Distance \* prix/km
      * Estimation des ventes d’un produit
        + Moyenne des 3 derniers mois
      * Cas d’une classification binaire
        + Modèle doit faire mieux qu’un résultat 100% aléatoire
    - Exemple de projet où le ML donne un avantage
      * Prédire défaillance d’une pièce ou d’un serveur informatique
      * Prédire risque de défaut d’un crédit
      * Classer automatiquement un grand volume de fichiers sons ou images
      * Détecter des contenus agressifs ou des fake news sur les réseaux sociaux
* Respecter le RGPD

### Phase 2 : Concevoir le prototype et valider la faisabilité du projet

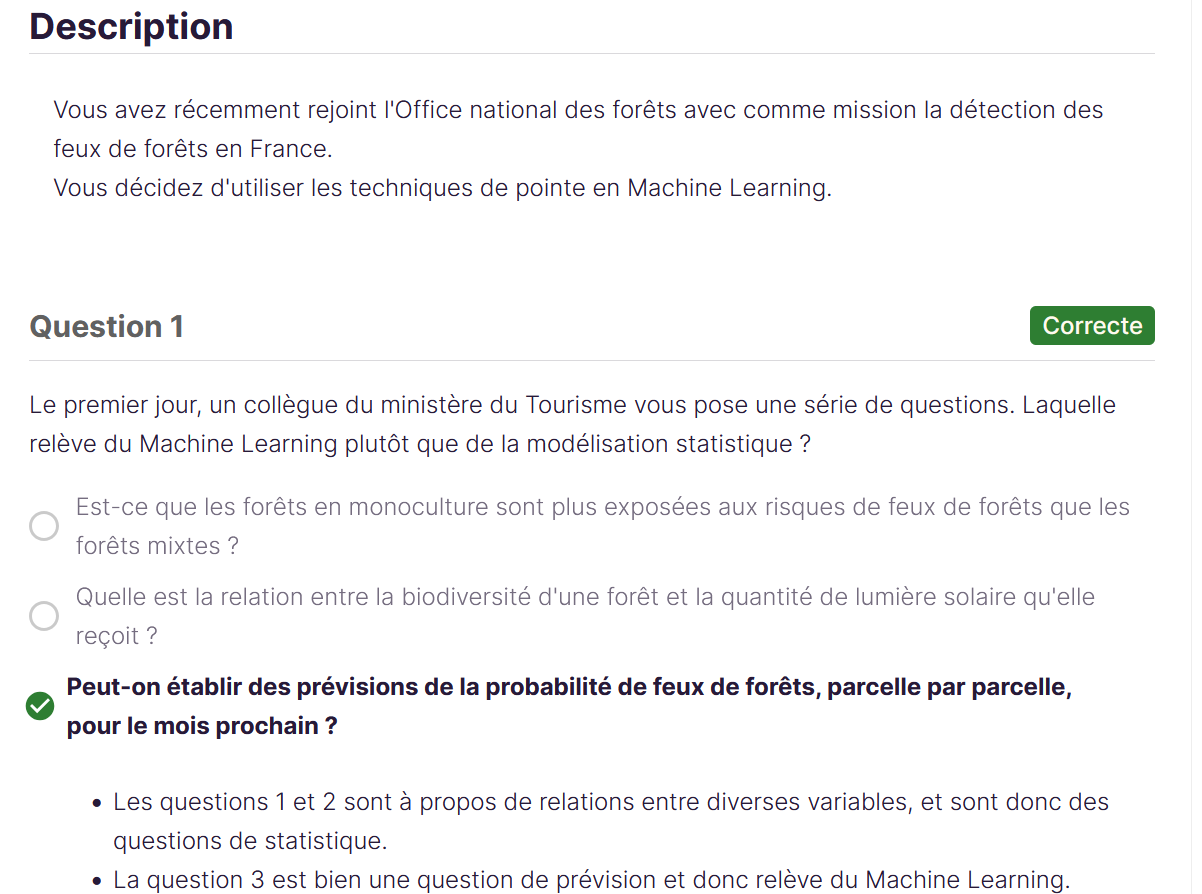
* Une fois fixé une version du jeu de données et une indication du benchmark de performance à dépasser, il faut obtenir un modèle
  + **Performant**
    - Bon score vis-à-vis de la métrique choisie
  + **Robuste**
    - Stable face à de nouvelles données
  + En fonction du contexte, on pourra choisir un modèle moins performant mais plus robuste, ou inversement
* Processus itératif
  + 1/ Mettre en forme les données
    - Nettoyage
      * Résoudre les outliers
    - Feature engineering
      * Création de nouvelles variables à partir des features existantes
        + Prendre le carré d’un prix par exemple
    - Numériser les données
      * Catégoriques, textuelles, images
  + 2/ Choisir type de modèle
    - GLM
    - Tree
    - NN
    - …
  + 3/ Répartir les données Train - Test
  + 4/ Optimisation des paramètres du modèle

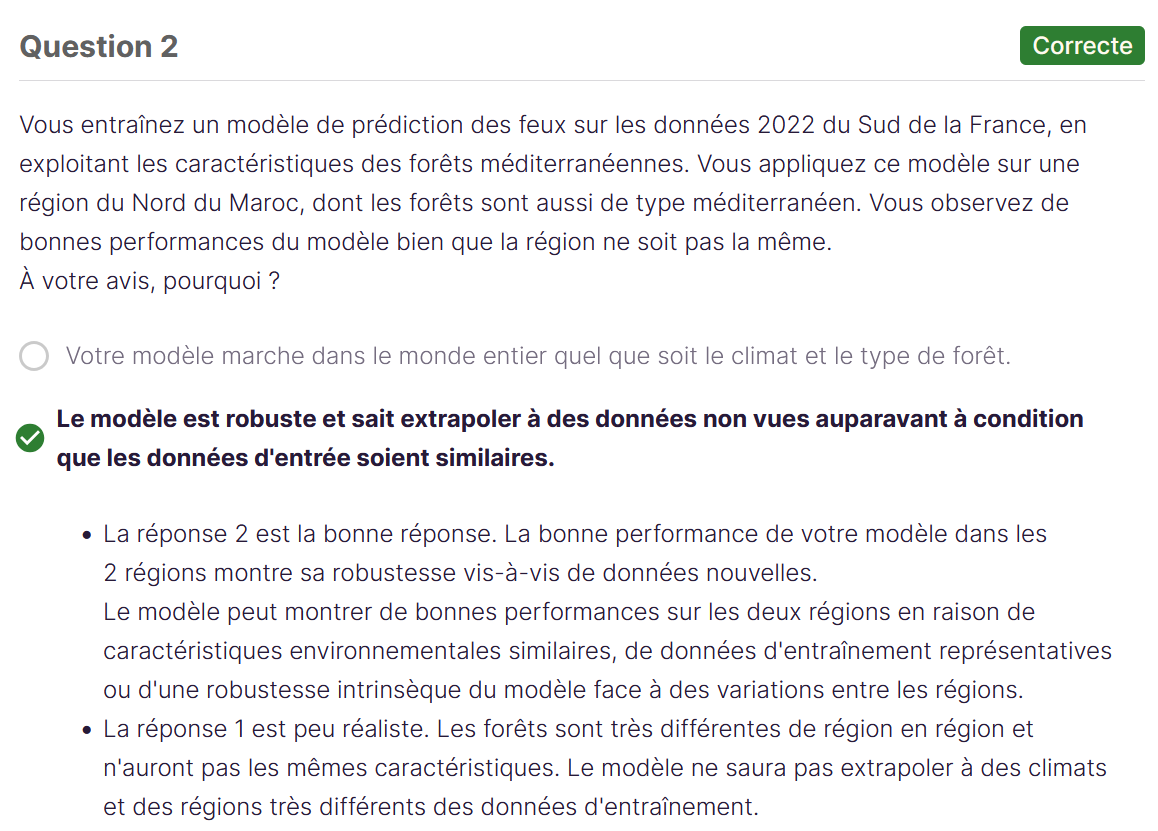


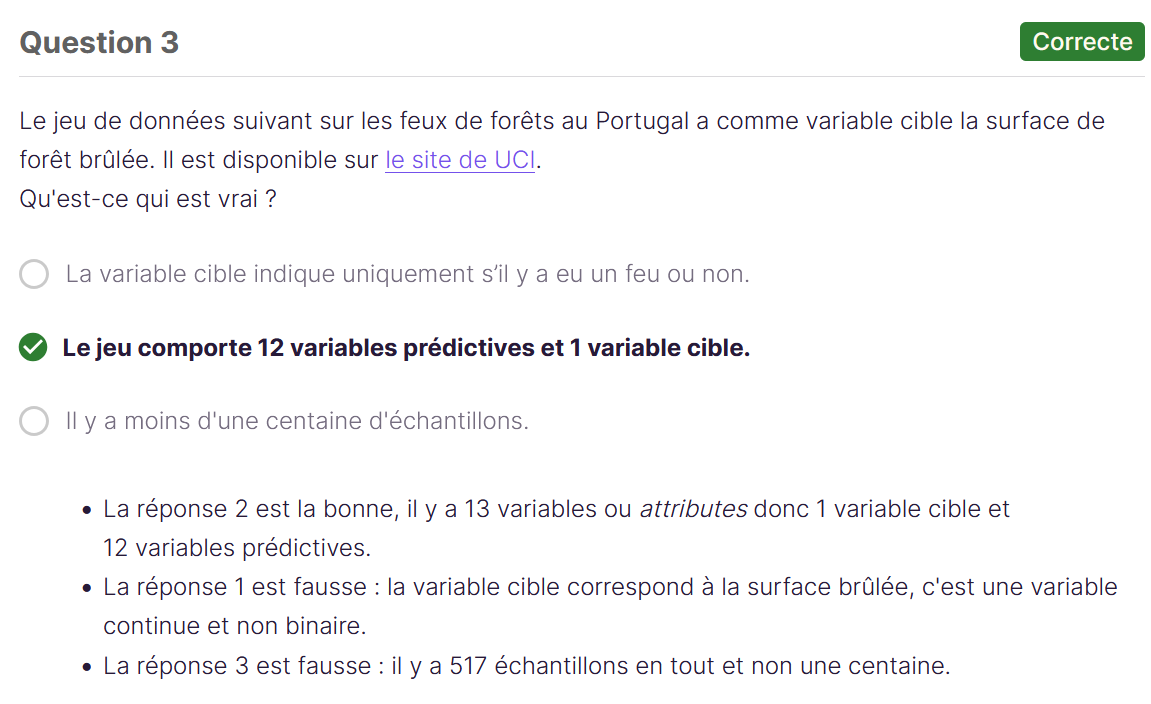
### Phase 3 : Mise en production

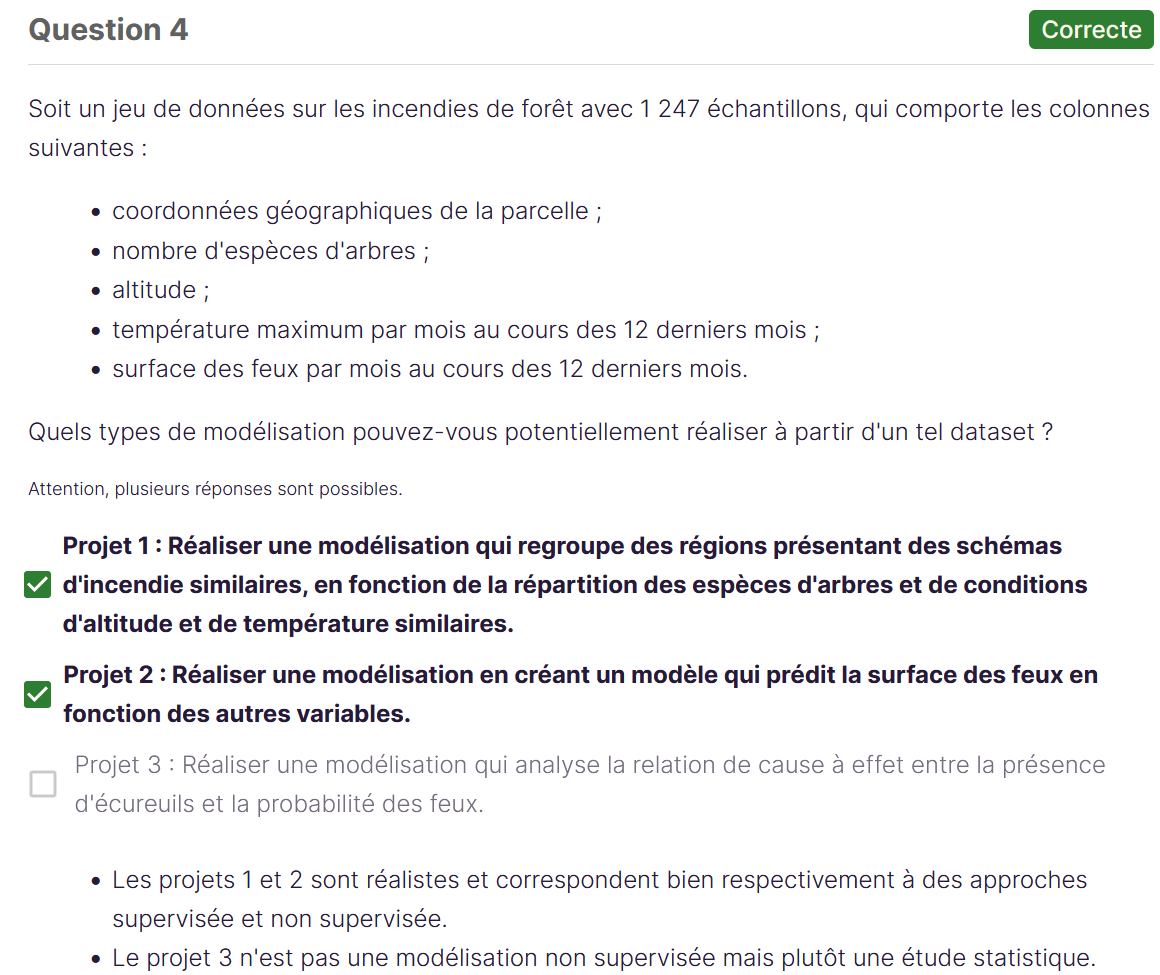
* Une fois optimisé, le modèle va intégrer le produit final
  + Mise en production + surveillance des modèles
    - Travail des MLOps et DevOps
      * Automatisation de
        + Mise à jour
        + Ré entrainement
        + Déploiement
        + Surveillance
* Drift du modèle
  + Il est probable que les conditions sous-jacentes aux données d’entrainement du modèle ne seront que transitoires
    - Les modèles risquent à un moment de perdre de leur efficacité
      * Il faut alors les réentraîner

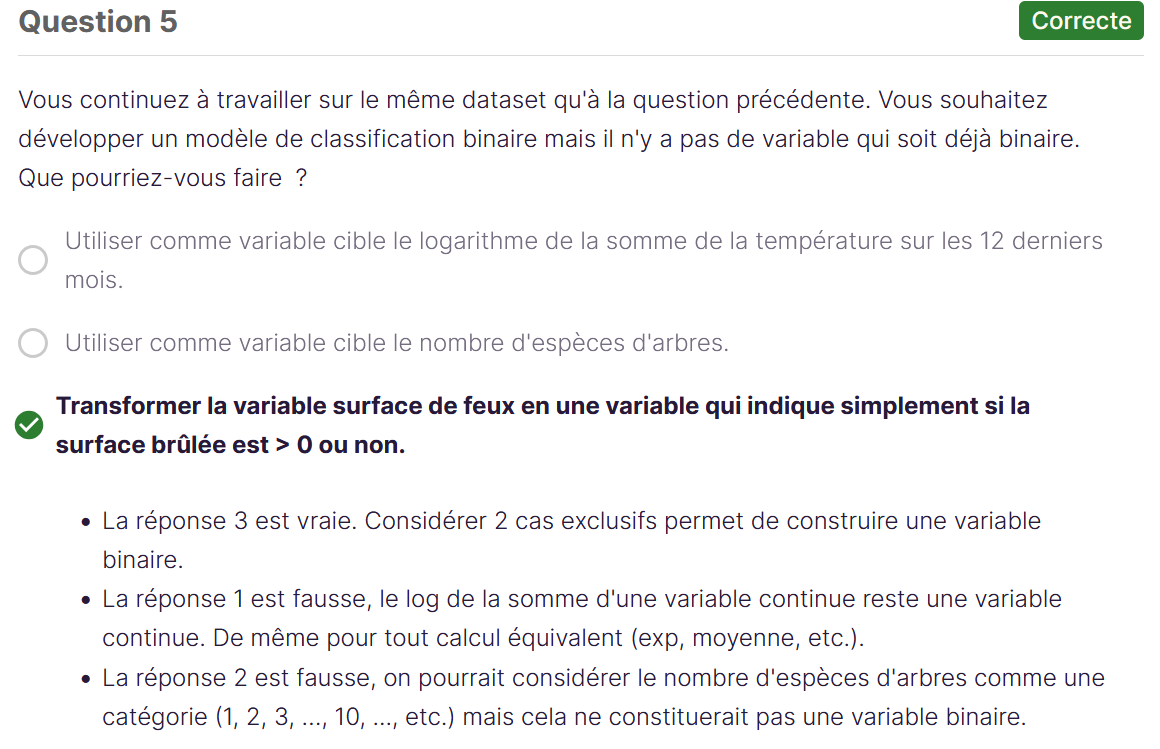
## Quizz

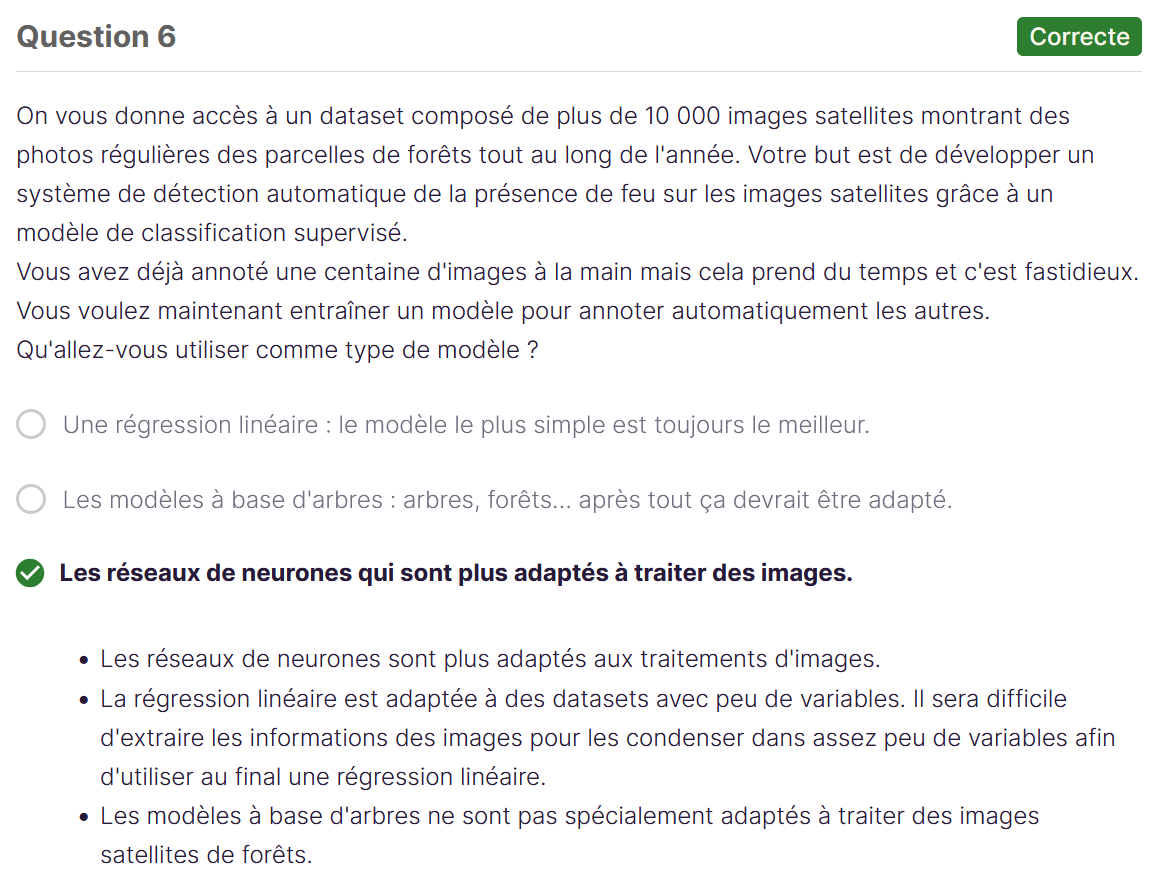


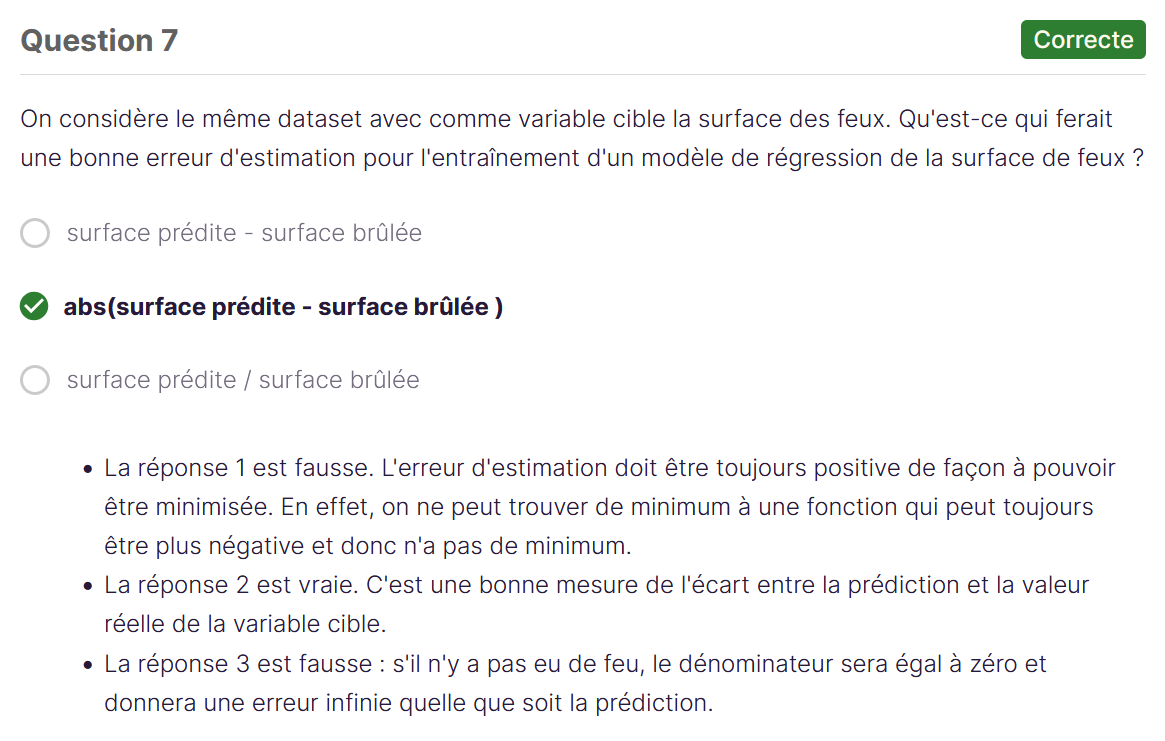


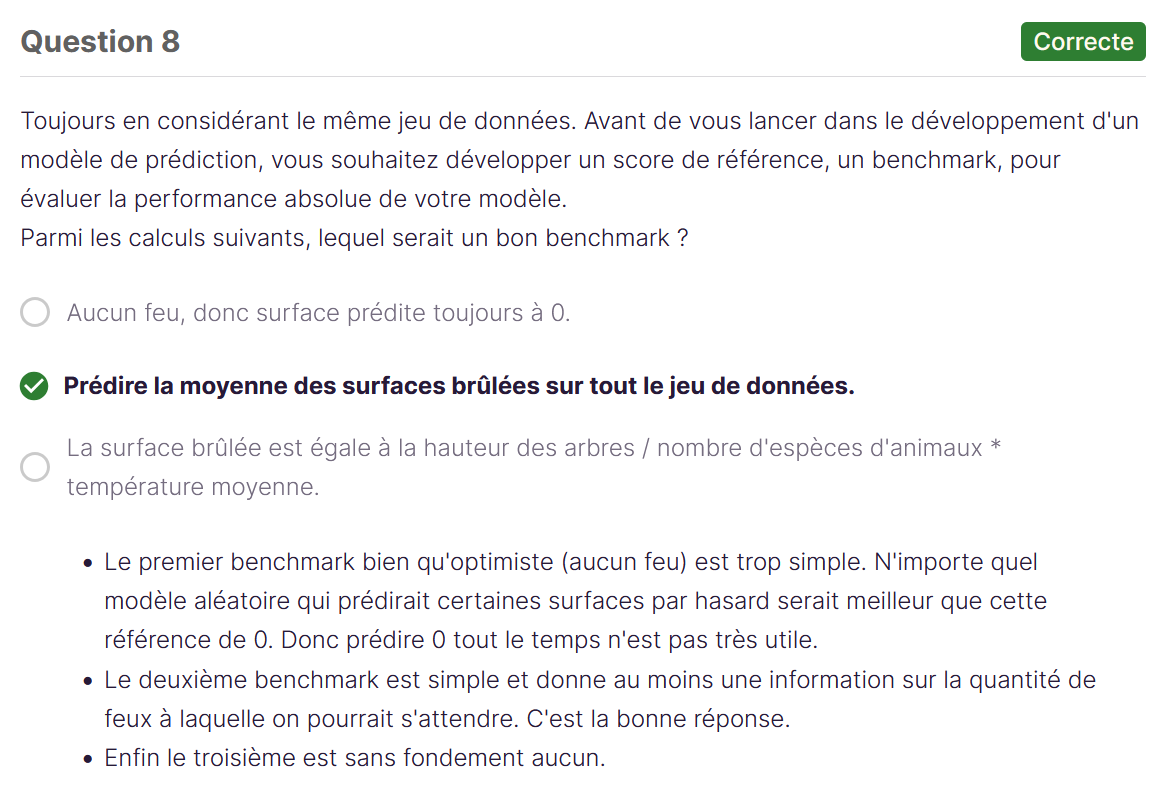












## Evaluer son premier modèle prédictif

### Description dataset

* Dataset
  + 237 échantillons
    - Âge
      * En mois (de 139 à 250 mois = 11.5 à 20 ans)
    - Sexe
      * Binaire (0 pour les garçons et 1 pour les filles)
    - Taille
      * Cm (de 128.27 à 182.88)
    - Poids
      * Variable cible
      * Kg (de 22.9 à 77.78)

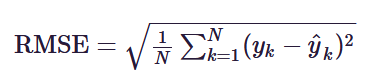
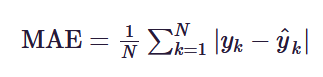
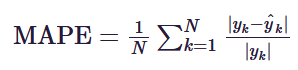
### Objectif du modèle

* Nous cherchons à construire un modèle qui prédit la variable cible (le poids de l’enfant) à partir des autres variables
  + On cherche donc les coefficients a, b, c tels que
    - Poids = a \* sexe + b \* âge + c \* taille + bruit
      * **Le bruit représentant l’information qui n’est pas capturée par le modèle linéaire**

### Entrainement

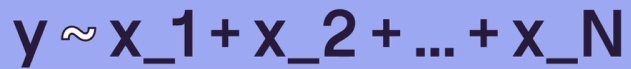
* **X** est la matrice qui contient **les prédicteurs**
  + **Matrice de design**
    - Représente les données d’entrées
      * Chaque ligne = échantillon
      * Chaque colonne = variable d’entrée
* **Y** est le **vecteur** de la variable cible
* Minimiser la fonction de coût
  + Permet d’arrêter l’algorithme quand certain seuil d’erreur atteint
  + Permet de mettre à jour progressivement les coefficients du modèle
  + La fonction de coût est souvent liée au choix du modèle
  + Lorsque l’on fait fit(X, y)
    - On démarre l’algorithme qui va minimiser la fonction de coût jusqu’à atteindre le seuil minimum d’erreur
  + Exemple pour la régression linéaire de scikit-learn
    - **Fonction de coût** est la **MSE** (Mean Squared Error) (**loss function**)
      * 
        + N : Nombre d’échantillons d’entraînement
        + Yk : Valeurs cibles
        + k: Valeurs prédites par le modèle
* Peu importe la classe du modèle, l’entrainer consiste à enchaîner les étapes
  + 1. Charger les données
  + 2. Transformer / améliorer les données
  + 3. Scinder les données en train / test ou train / test / validation
  + 4. Entrainer le modèle avec plusieurs configurations de paramètres (si le modèle s’y prête)
  + 5. Calculer le score de chaque version du modèle

### Evaluer la performance d’un modèle

* Evaluer la performance du modèle une fois entrainé
  + **Score de performance**
* Calcul dépend de la tâche ML considérée
  + Evaluation d’un modèle de classification différentes d’un modèle de régression
* Dans le contexte d’une régression, nous avons les scores suivants
  + Scores relatifs à l’envergure de la variable cible, et donc difficile de comparer entre les modèles
    - **RMSE**
      * Root Mean Square Error
      * Ecart quadratique moyen
      * 
    - **MAE**
      * Mean Absolute Error
      * Erreur absolue moyenne
      * 
  + Scores en pourcentage par rapport à la valeur cible
    - **MAPE**
      * Mean Absolute Percentage Error
      * Erreur absolue moyenne en pourcentage
      * 
        + Contrairement à (si proche de 1 alors meilleur modèle), MAPE, MAE, et RMSE : au plus ils sont petit, meilleur est le modèle
* Il existe de nombreuses variantes de métriques d’évaluation. Dans la documentation scikit-learn
  + Pour la régression
    - 15 métriques
  + Pour le clustering
    - 28 métriques

## Régression linéaire

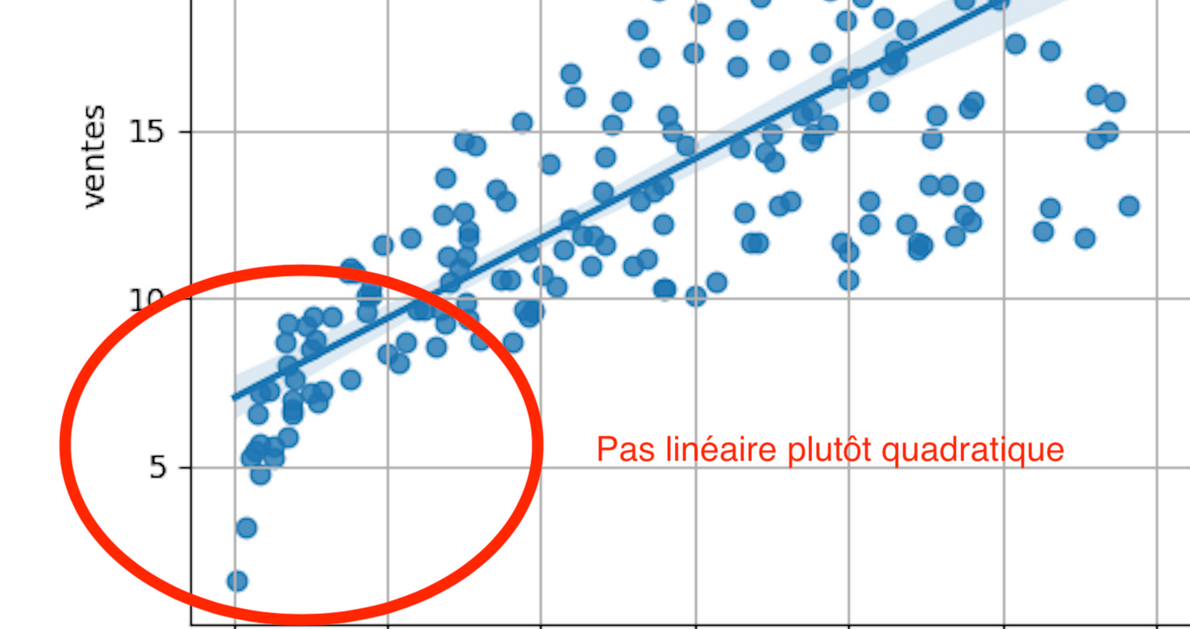
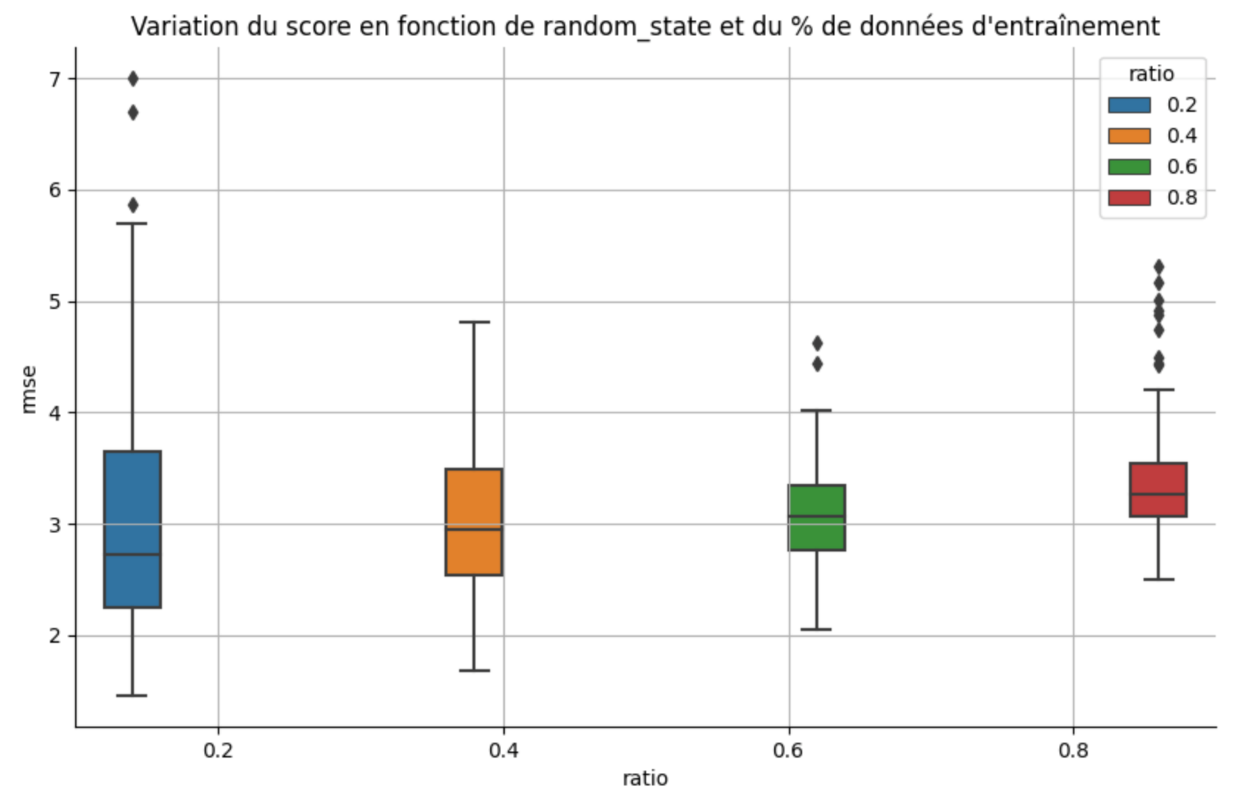
### Un ou plusieurs variables

* **Régression univariée**
  + Lorsque l’on a une seule variable de prédiction
* **Régression multivariée**
  + Plusieurs prédicteurs / variables indépendantes
  + 
    - 
    - 

### Limites de la régression

* 1ère limite
  + Utiliser une régression linéaire à partir d’un cas non linéaire ne marchera pas
    - Il faut s’assurer que la relation entre la variable cible et les prédicteurs soit linéaire
* 2ème limite
  + Variations d’amplitudes significative peuvent fausser l’importance des coefficients
    - Les valeurs des coefficients de la régression linéaire sont en effet inversement proportionnelles à l’amplitude de la variable associée
      * Les plus grandes valeurs vont dont avoir plus d’influence sur la réduction de la fonction de coût lors de la descente de gradient
    - Il faut alors **normaliser les variables** afin que leur amplitude soit comprise dans un intervalle comparable
      * Chaque coefficient de la régression linéaire reflète alors directement l’impact de la variable sur la variable cible (cad son pouvoir de prédiction)

### Améliorations d’un modèle

* Ajouter un terme quadratique = **régression polynomiale**
  + 
  + Donc ajouter *tv*2 pour obtenir
    - *Ventes* = *a*∗*tv* + *b*∗*tv*2 + *c*
      * Polynôme du second degré de variable tv
  + Mais l’amplitude de *tv2* va être bien plus grande que celle des autres variables
    - Important de normaliser les variables
      * MinMaxScaler
* **Random\_state**
  + En faisant varier ce paramètre, les données seront réparties parmi les sous-ensembles entraînement et test de façon différente et le modèle ne sera pas entraîné sur les mêmes échantillons
    - Si des valeurs extrêmes se retrouvent dans le sous-ensemble d’entrainement mais pas dans le sous-ensemble de test, le modèle va les prendre en compte va faire varier les coefficients de régression et va diminuer le score du modèle car le sous-ensemble de test n’en contient pas.
* Comparaison de l’influence de size\_test et random\_state
  + 

### Validation croisée

## Régression logistique

### Différents types

* La variable cible est une catégorie et non une valeur continue comme dans la régression
* Plusieurs types de classification suivant les valeurs prisent par la variable cible
  + **Classification binaire**
    - Variable cible prend 2 valeurs exclusives
      * Oui / Non
      * 0 / 1
      * Vrai / Faux
      * Fonctionne / Ne fonctionne pas
  + **Classification multiclasse**
    - Variable cible peut prendre plus de 2 valeurs
      * **Classification ordinale**
        + Valeurs sont ordonnées

1, 2, 3

Mauvais, moyen, bon

* + - * **Classification nominale**
        + Catégories ne sont pas ordonnées

Couleurs, Nationalités, animaux, etc.

* + **Classification multilabel**
    - Un échantillon peut appartenir à plusieurs catégories à la fois
      * Genres de musique
      * Taxinomie d’un animal
        + Tigre est un animal, mammifère, carnivore, félin

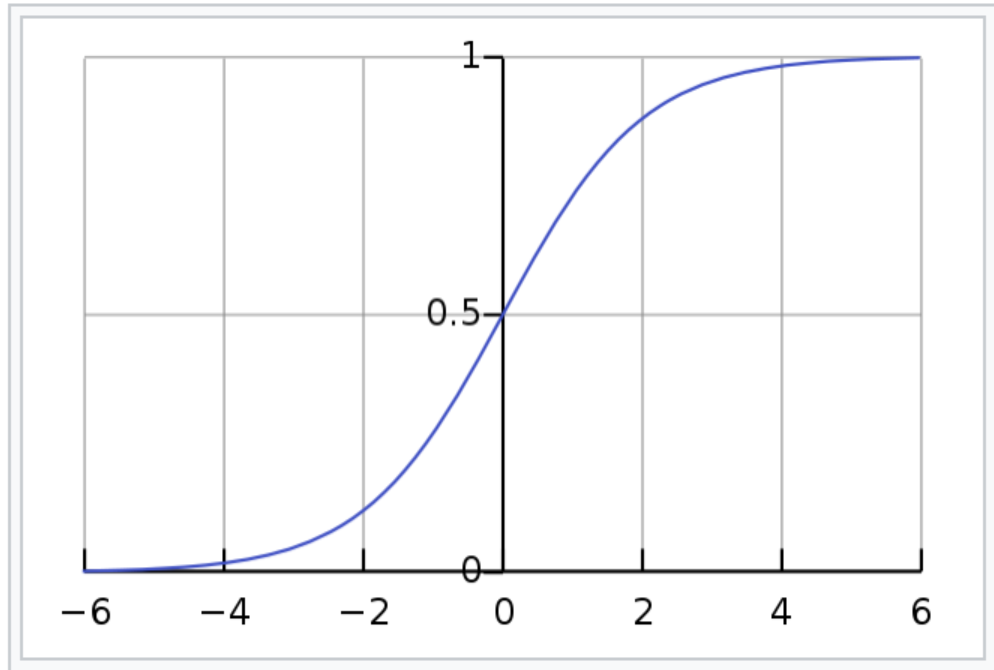
### Rapport entre classification et régression

* Principe général est similaire à celui de la régression
  + Ce qui change surtout
    - Métriques de score des modèle et leurs interprétations
* On adapte la régression linéaire au cas de la classification en interprétant la prédiction de la régression linéaire comme une des catégories de la variable cible
  + Soit y = ax + b
  + Soit f une fonction qui projette y dans l’intervalle [0, 1]
    - Z = f(y) = f(ax + b)
    - Z peut être interprété comme étant la probabilité que la prédiction soit dans une des catégories.
      * Si seul de classification = 0.5
        + Si f(y) < 0.5

Alors catégorie 0

* + - * + Si f(y) > 0.5

Alors catégorie 1

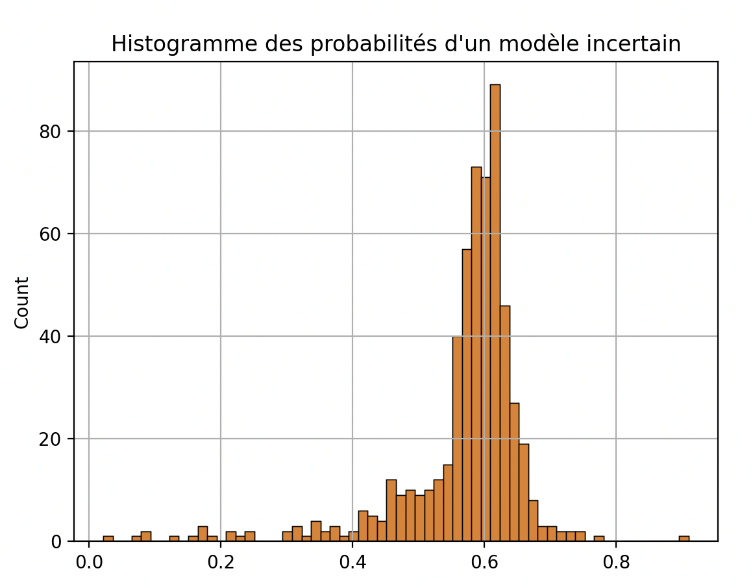
* + - Cette fonction de projection de toute valeur réelle dans l’intervalle [0, 1] s’appelle la fonction logistique
      * 
      * 

### Concepts de la classification binaire

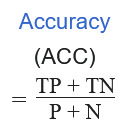
* Plusieurs étapes
  + Modéliser une régression linéaire
  + Projeter la prédiction *y* variables dans l’intervalle [0, 1]
    - Logistique(y)
  + Interpréter logistic(y) comme probabilité que *y* soit dans une catégorie ou dans l’autre en fonction d’un seuil t = x
    - Si logistic(y) > x
      * Catégorie 1
    - Si logistic(y) < x
      * Catégorie 2

### Evaluer les performances du modèle

#### Histogramme des probabilités des prédictions

* Bon exemple
  + 
    - Modèle est assez fiable au niveau de ses prédictions
      * La plupart des prédictions ont une probabilité proche de 0 ou de 1
* Mauvais exemple
  + 
    - Majorité des prédictions entre 0.55 et 0.65

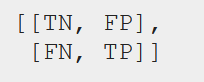
#### Accuracy

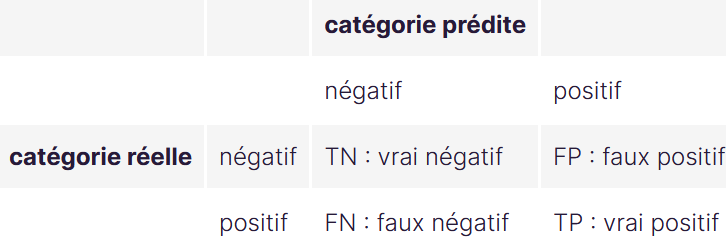
* Accuracy (exactitude) (ACC)
  + = échantillons bien classés / échantillons au total
    - TP + TN / P + N
    - 

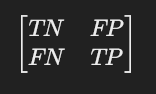
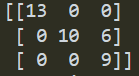
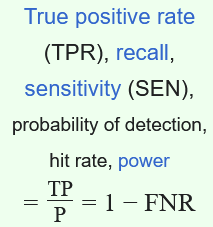
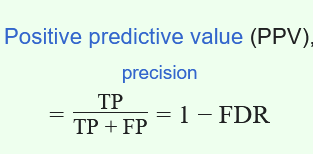
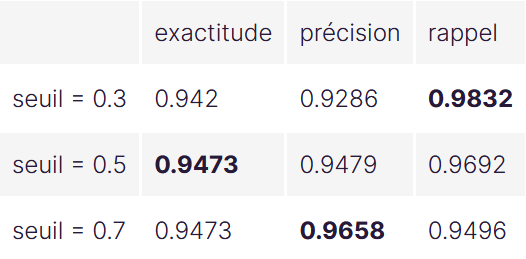
#### Matrice de confusion



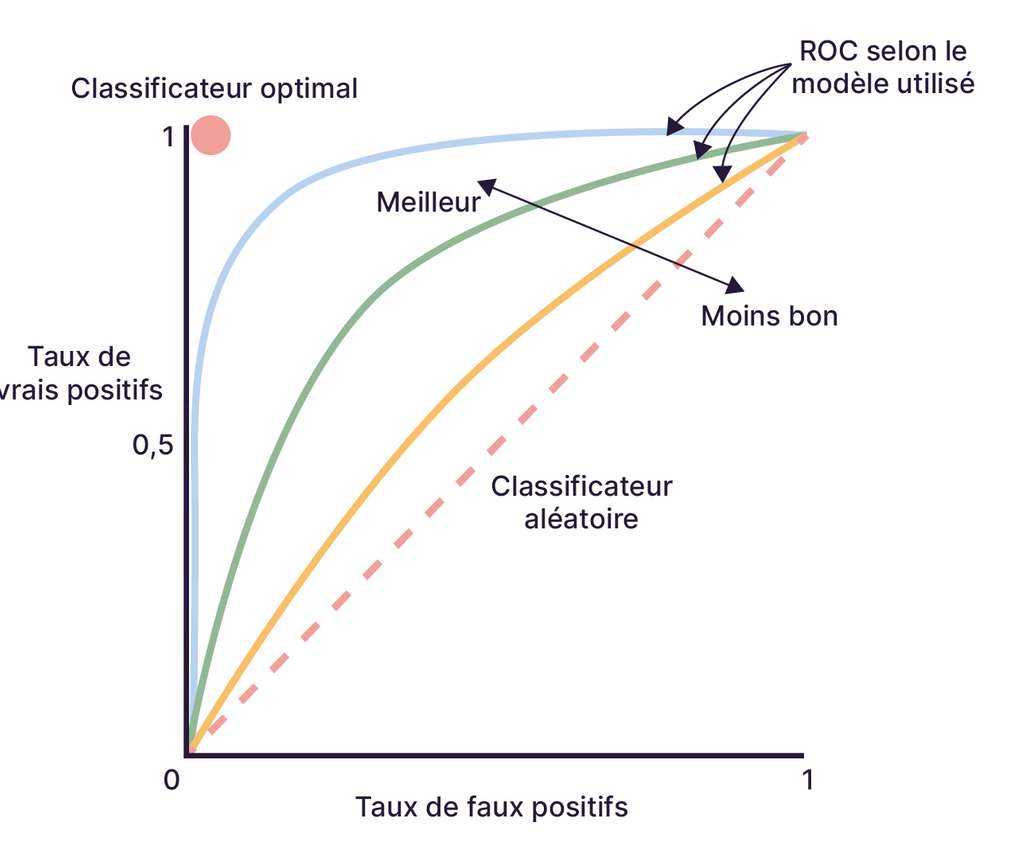
Dans scikit-learn :

* 



* Existe matrice de confusion binaire
  + 
* Et matrice de confusion multiclasse
  + 
  + 
* **Recall** (rappel) (True Positive Rate) (TPR) (8)
  + TP / (TP + FN)
    - Vrais positifs / tous les positifs réels
    - Capacité à détecter les positifs lorsqu’ils sont positifs
      * Sensibilité
    - Adapté pour minimiser les faux négatifs
* **Précision** (Positive predictive value)
  + TP / (TP / FP)
    - Vrais positifs / tous les positifs détectés
      * Probabilité d’avoir raison quand on dit « positif »
    - Adapté pour minimiser les faux positifs
    - 
* Calculer les paramètres pour plusieurs seuils
  + 

#### ROC AUC score

* Tracer
  + Taux de vrais positifs / taux de faux positifs
  + Rappel (TPR) (True Positive Rate) par rapport au FPR (False Positive Rate) en fonction des seuils de classification
  + TPR = TP / (TP + FN)
  + FPR = FP / (FP + TN)
  + 
    - Diagonale représente un modèle purement aléatoire
    - But, se rapprocher en haut à droite
      * Détecte tous les positifs
      * Aucun résultat positif est un FP
    - La métrique qui nous intéresse ici est la **surface entre la courbe et la diagonale**
      * **ROC – AUC**
      * Prend en compte tous les seuils de classement entre 0 et 1 et donc ne dépend pas d’une valeur spécifique de ce seuil

## K-means

### Généralités

* Technique de prédiction non supervisée
  + Partitionnement (Clustering)
    - Regrouper les échantillons du dataset qui se ressemblent de façon automatique
    - On partitionne les données en un **nombre fini de sous-ensembles cohérents** que l’on peut appeler *groupes*, *classes* ou *catégories*
* L’algorithme de clustering a deux objectifs
  + Minimiser la différence entre les échantillons d’un même groupe
    - On veut des groupes denses et homogènes
  + Maximiser la différence entre les groupes
    - On veut des groupes bien séparés les uns des autres
* Le clustering est utilisé lorsque l’on veut comprendre le dataset sans apriori préalable
  + L'analyse des données géospatiales pour déterminer les caractéristiques spécifiques de certaines forêts dans le contexte de la biodiversité ou de la prévention des feux
  + L'analyse de texte pour faciliter la compréhension et l'organisation de grands ensembles de données textuelles. Par exemple l'analyse de tous les articles d'une publication pour identifier les thèmes principaux ou récurrents
  + La détection d'anomalies pour regrouper les données normales dans un cluster et en considérant toute donnée non conforme à ce cluster comme une anomalie.
  + Le marketing pour mieux cibler les clients en fonction de leurs comportements d'achat, préférences ou caractéristiques démographiques.

### Comment ?

* Comment définir la **similarité des échantillons** ?
  + Se traduit par une faible distance entre les échantillons.
    - La mesure la plus usitée est la **distance euclidienne**
* Comment trouver le **nombre de clusters optimal ?**
  + Paramètre d’entrée du modèle
    - Décision dépend du contexte, des connaissances des données et des objectifs
    - Pas toujours possible de le connaître à l’avance, il faut donc l’estimer avant d’entraîner le modèle
      * Existe plusieurs méthodes
        + Elbow method
        + Méthode de la silhouette
* Quels types d’algorithmes pour le clustering ?
  + 14 modèles de clustering dans Scikitleanr
    - Le plus simple : k-means

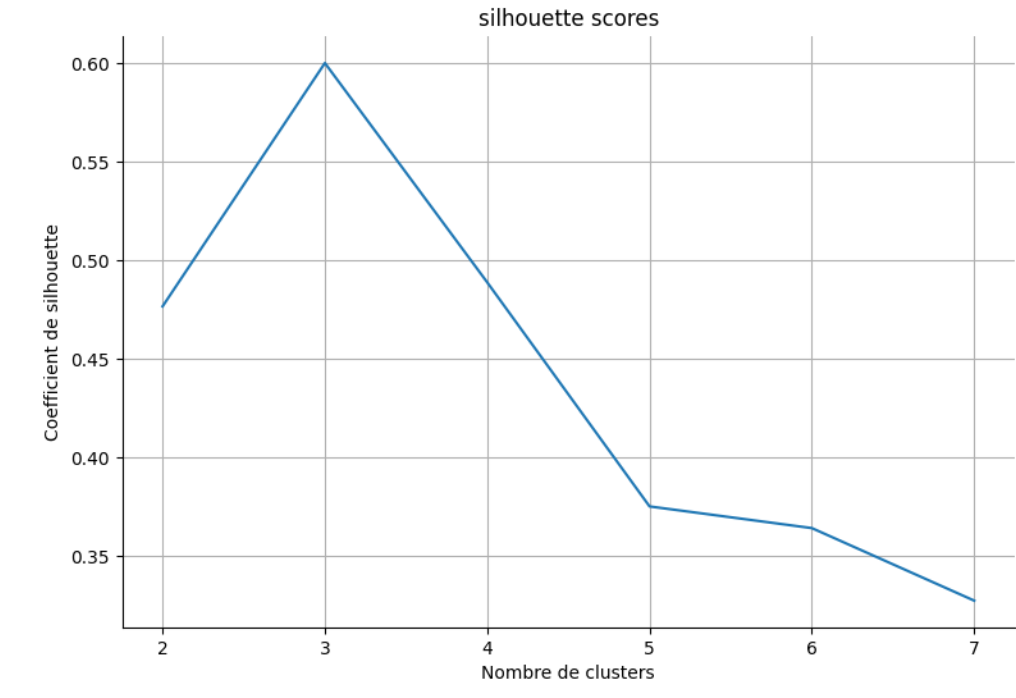
### Algorithme du k-means

* Phase d’initialisation
  + On choisit arbitrairement un nombre de clusters K
  + Le modèle sélectionne aléatoirement K points come centres de clusters initiaux (centroïdes)
* A chaque itération
  + Attribution des données
    - Chaque point de données est associé au cluster dont le centroïde est le plus proche
  + Mise à jour des centroïdes
    - Les coordonnées des centroïdes des clusters sont recalculées en prenant en compte les nouveaux points
* Fin quand
  + Centroïdes ne bougent plus
  + Nombre d’itérations fixés en avance
* En résumé
  + Nécessite de choisir judicieusement le nombre de clusters au départ
  + Sensible à l’initialisation des centroïdes

### Evaluer la performance du modèle k-means

* Méthode 1 : Calculer la distance entre les points et leur centroïde respectif
  + La distance entre les points d’un même cluster doit être minimale
* Méthode 2 : Faire une estimation graphique
* Méthode 3 : Utiliser le coefficient de silhouette

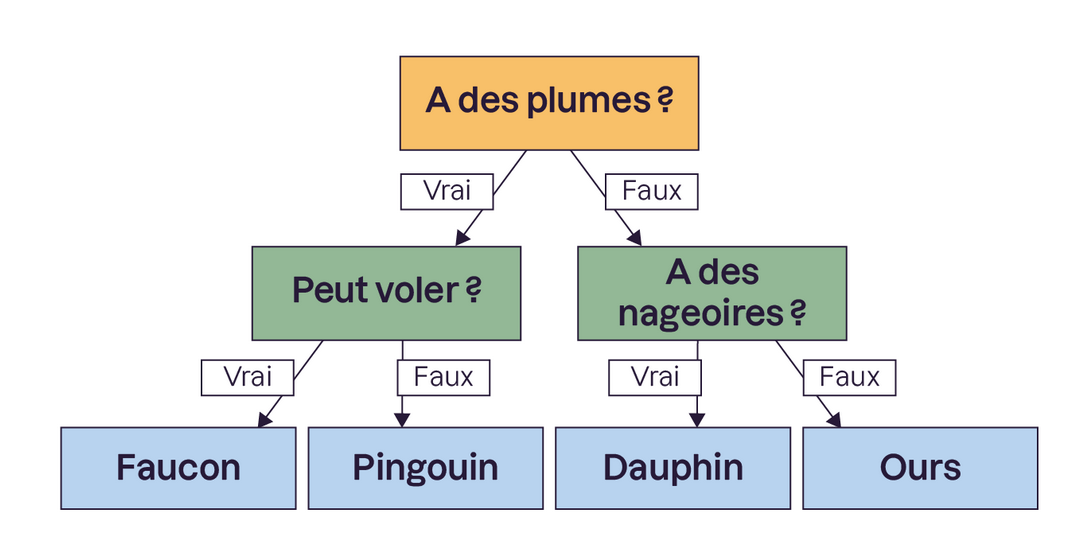
### Coefficient de silhouette

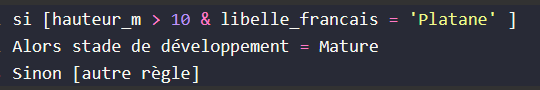
* Prend en compte
  + La densité des clusters
  + L’écart entre les différents clusters
* Paramètres
  + Distance moyenne intra-groupe (a)
    - Moyenne de la distance de chaque échantillon avec son centroïde dans chaque groupe
  + Distance moyenne entre les groupes (b)
    - Distance entre chaque échantillon et le groupe le plus proche dont l’échantillon ne fait pas partie
* Coefficient de silhouette = (b – a) / max(a, b)
  + Compris entre -1 et 1
    - -1 (mauvais) et 1 (excellent)
    - 

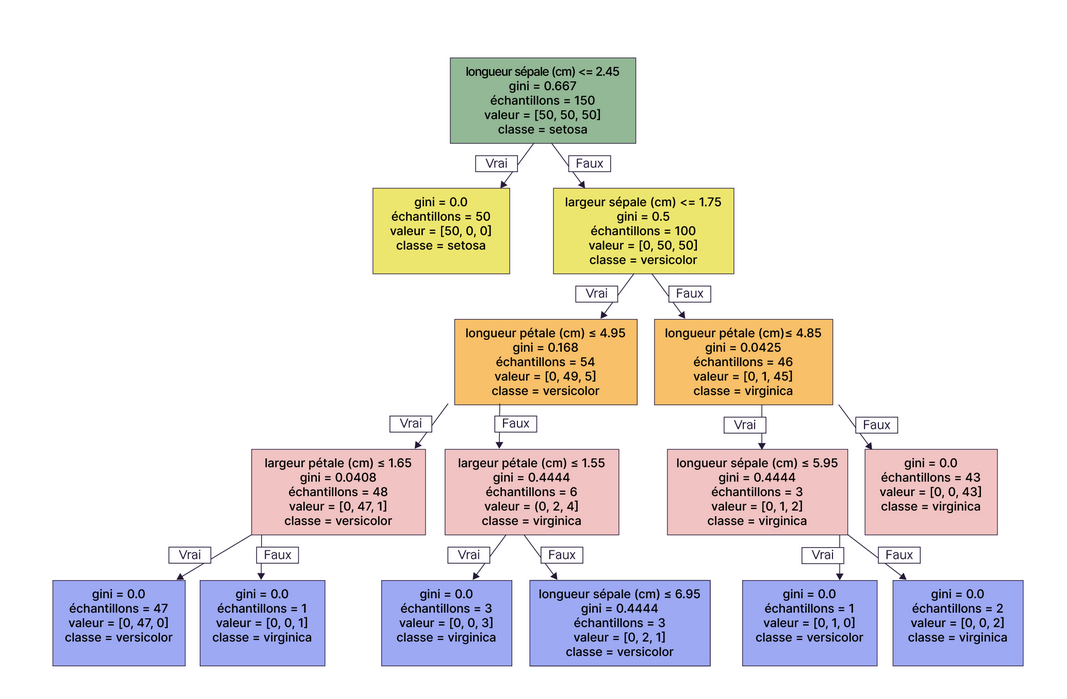
## Arbre de décision

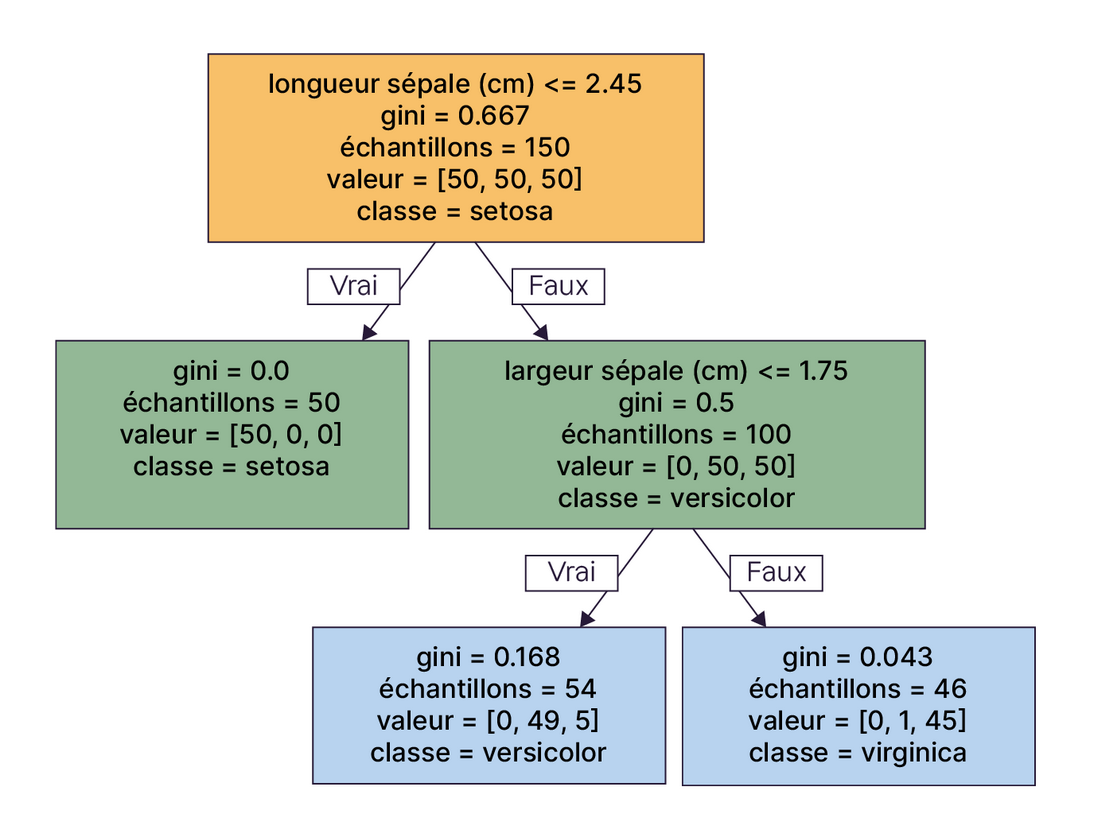
### Géénralités

* Enchainement
  + De règles de classification
    - Établies automatiquement
      * À partir de variables prédictrices



* + Carrés = **nœuds**
  + Liens entre nœuds = **branches**
* **Les nœuds créent des bifurcations de branches**
  + Ces règles de bifurcation sont établies à partir de critères statistiques sur les variables
    - ****
  + L’algorithme d’entrainement détermine les critères des différentes règles
    - Variables concernées
    - Seuils
    - Valeurs catégoriques
  + L’algorithme répond à un critère prédéfinit pour établir la règle de bifurcation
    - Dans scikit-learn, il s’agit du paramètre **criterion**
      * La fonction qui mesure la qualité d’une bifurcation en 2 branches
* Un arbre profond aura beaucoup de règles
  + Un arbre peu profond aura peu de règles
    - Pour limiter la profondeur d’un arbre de décision
      * **Pruning** (elagage)
        + Dans scikit-learn, pruning = max\_depth
* **Hyper paramètres**
  + Les paramètres d’un modèle qui modifient son comportement, comme
    - Max\_depth
    - Criterion





## Transformer les jeux de données

### Rôle central du jeu de données

* Existe deux approches pour la construction des modèles
  + **Model-centric**
    - La plus courante et la plus enseignée
    - On nous fournit un dataset propre, à nous de créer un modèle performant
    - Types de modèles les plus efficaces sont connus et disponibles
      * Open source via des librairies
        + Scikit-learn
        + TensforFlow
        + Etc.
      * Services cloud ML
        + Vertex AI
        + SageMaker
        + Etc.
    - Les gains de performances ne se font pas sur l’aptitude à optimiser le modèle (car ils sont déjà optimisés), mais sur l’élaboration du jeu de données
  + **Data-centric**
    - Approche plus centrée sur les données que sur le modèle
    - Recouvre une série de pratiques dont la finalité est de booster les performances du modèle en manipulant les données
    - Être attentif
      * A la qualité du jeu de données
        + Valeurs manquantes
        + Outliers (valeurs extrèmes)
        + Mauvais étiquetages
        + Biais de représentation
      * Aux erreurs du modèle
        + Comprendre pourquoi certains échantillons posent problème et transformer ces caractéristiques en variables que le modèle puisse apprendre

### Où trouver ses données

* [Dataset Search](https://datasetsearch.research.google.com/)
  + Lancé en 2018, Google lance son moteur de recherche dédié aux jeux de données
* [Kaggle](https://www.kaggle.com/datasets)
  + Plateforme de compétition de Machine Learning
* [UCI](https://archive.ics.uci.edu/)
  + University California, Irvine
* Sites institutionnels
  + [Opendata.paris](https://opendata.paris.fr/pages/home/)
  + [Data.london.gov](https://data.london.gov.uk/dataset)
  + [Dati.comune.roma](https://dati.comune.roma.it/)
  + [Data.europa](https://data.europa.eu/en)
* Biodiversité : agences scientifiques, ONG, agents nationaux
  + [ADEME](https://data.ademe.fr/)
  + [EdF](https://opendata.edf.fr/pages/welcome/)
  + [WWF](https://globil-panda.opendata.arcgis.com/)
  + [GBIF](https://www.gbif.org/)
* [BigQuery](https://cloud.google.com/bigquery/public-data?hl=fr)
  + Service big data de Google Cloud
* Librairies
  + Scikit-learn
    - [Toy dataset](https://scikit-learn.org/stable/datasets/toy_dataset.html)
    - [Real World](https://scikit-learn.org/stable/datasets/real_world.html)
  + [Statsmodels](https://www.statsmodels.org/devel/datasets/index.html#available-datasets)
  + [R Datasets](https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/00Index.html)

### Créer son propre dataset

* Existe méthodes dans Scikit-learn pour créer dataset en fonction de type de tâche à accomplir
  + Make\_blobs()
  + Make\_regression()
  + Make\_classification()
* Existe d’autres méthodes plus complexes

### Réduire influence des préjugés dans prédictions du modèle

* Un modèle biaisé est un modèle dont les prédictions sont systématiquement distordues
  + Risque de décisions systématiquement inéquitables ou inexactes
* Différents risques de biais
  + Biais d’échantillonnage des données d’entraînement
    - S’assurer de **l’exhaustivité**
      * Des valeurs des variables prédictrices
      * Des valeurs des variables cibles
    - Mettre en place **stratégie de remédiation**

## Améliorer un jeu de données

* Travail sur un dataset réel
  + Data set des arbres de paris intra-muros et périphérie proche
    - 200 000 arbres
* Plus proche d’un dataset rencontré en milieu professionnel car
  + Des données manquantes
  + Des données aberrantes / extrèmes (outliers)
  + Des catégories trop nombreuses ou sous-représentées

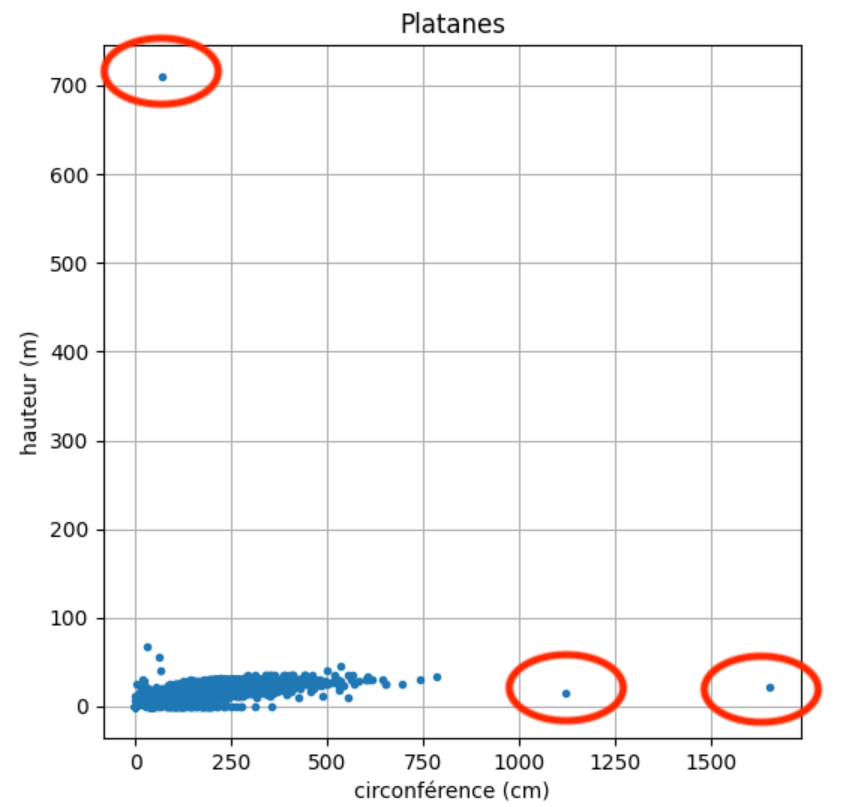
### Repérer les données manquantes

* 3 Stratégies possibles
  + **Supprimer**
    - Tous les échantillons pour lesquels cette valeur est manquante
    - Cela vaut seulement si concerne qu’une petite partie du dataset
  + **Remplacer**
    - Les valeurs manquantes par une valeur spécifique qui indique que la valeur est indisponible
      * -1 ou 0
      * None (pour une catégorie)
    - On espèce que le modèle saura prendre en compte l’information
  + **Inférer**
    - Les valeurs manquantes à partir
      * Des valeurs disponibles
        + Par exemple la moyenne
      * D’autres variables
        + Régression linéaire à partir des autres variables
* Certaines données manquantes peuvent provenir de la manière de récolter les données et de les classer (par exemple arrondir une variable peut transformer une variable aléatoire continue en variable aléatoire discrète)

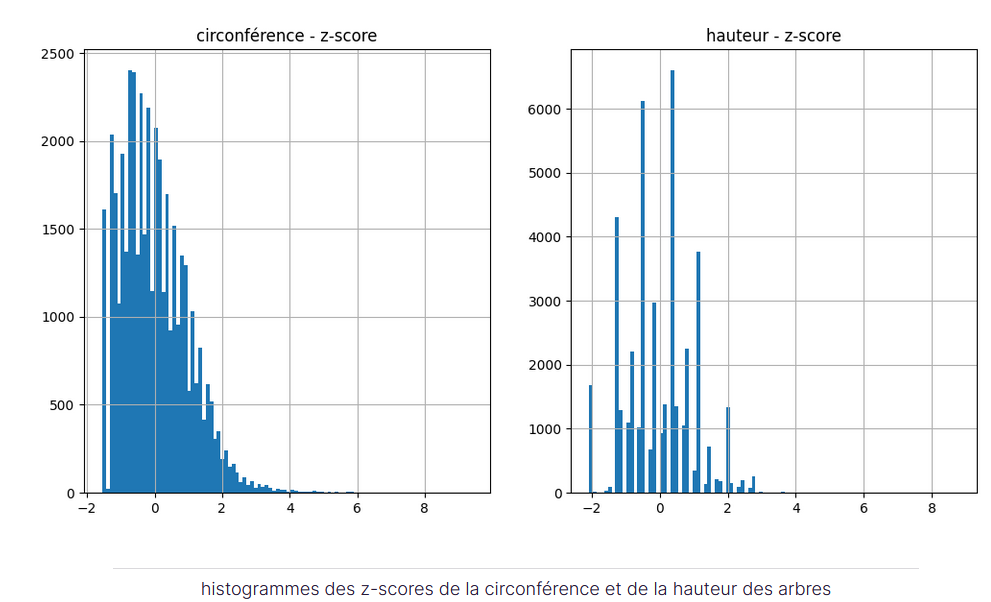
### Détecter les outliers

* Deux types d’outliers
  + Données aberrantes
  + Données extrêmes
    - Contiennent également de l’information
* Influence des outliers
  + Influence la moyenne, et les statistiques de la distribution de la variable

#### Visualisation de la distribution de la variable

* Histogramme
* Boxplot
  + 
    - On peut enlever ces 3 échantillons sans se poser de question

#### Méthode du z-score

* Z-score mesure de combien d’écarts-types une valeur est éloignée de la moyenne de la variable
  + On considère qu’un z-score > 2 ou > 3 est un outlier
  + 
    - On observe sur cette figure que la hauteur prend plutôt des valeurs discrètes que continue, sans doute dû à la méthode de mesure de la hauteur des arbres (effet arrondi manifeste)

#### Méthode de l’IQR

* Différence entre le 25e centile (Q1) et le 75e centile (Q3) (EI => écart interquartile) (IQR => interquartile range)
  + Sont considéré comme aberrantes les données
    - < Q1 – 1.5 \* IQR
    - > Q3 + 1.5 \* IQR

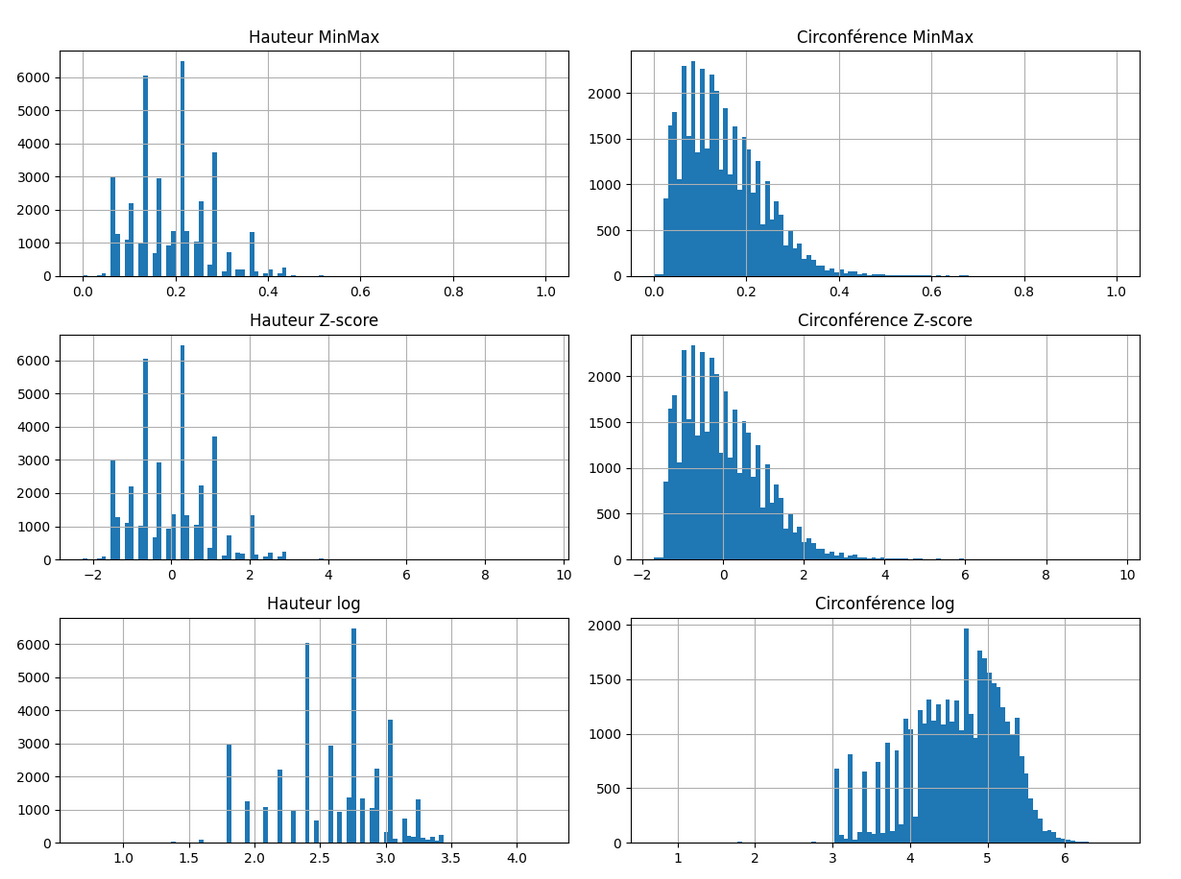
### Que faire des outliers

* Le bon choix ne peut se faire qu’a partir d’une connaissance du sujet
  + Prendre le log de la valeur
    - Cela réduit fortement la dispersion de la distribution et l’influence des outliers
    - Toujours rajouter +1 à une variable positive quand on prend son log pour éviter le log(0)
  + Instaurer une règle arbitraire
    - Choisir de fixer une limite supérieure à la variable
  + Discrétiser la variable par intervalle
    - En laissant le dernier intervalle ouvert pour inclure les valeurs extrêmes
    - Pour discrétiser la variable en un nombre fini d’intervalles, on utilise les méthodes qcut et cut de Pandas
      * Qcut
        + Scinde la variable en intervalles de volumes sensiblement égaux en fonction de leur fréquence
      * Cut
        + Scinde les données en intervalles de même taille

### Normaliser / standardiser les valeurs numériques

* La régression linéaire ou logistique sont sensibles à l’amplitude de valeurs des variables
  + Celles dont les valeurs sont plus grandes dominent le processus d’apprentissage
  + Les variables plus petites sont cachées
  + Performances du modèle vont se dégrader
  + Normaliser les variables pour qu’elles aient des amplitudes similaires
* Normalisation (centrer)
  + Min-Max Scaling
  + 
* Standardisation
  + Z-Score Standardization
  + 
    - Données sont centrées réduites
      * Moyenne = 0 et écart-type = 1

### Récapitulatif des transformations



## Transformer les variables

* Transformer les variables
  + Pour faciliter l’apprentissage du modèle
* Modèle n’accepte que des chiffres
  + Ici, on va apprendre à traiter les catégories textuelles
  + Si texte sous forme libre
    - Traitement du langage (NLP)

### Traitement des catégories textuelles

* On distinguera
  + Le cas binaire où l’on a deux catégories
  + Les cas
    - Un peu plus que 2
    - Carrément beaucoup

#### Le cas binaire

* La variable prend **2 valeurs distinctes et exclusives**
  + Oui / Non
  + Vrai / Faux
  + Mort / Vivant
  + Homme / Femme
  + Positif / Négatif
* Suffit de transformer les valeurs en chiffres en choisissant de manière arbitraire une valeur pour chaque catégorie
  + Oui = 1 et Non = 0

#### Le cas non binaire pour mais peu de catégories

* On distingue
  + Les **catégories ordonnées (ordinales)**
    - Petit, Moyen, Grand
    - Froid, Tiède, Chaud
  + Les **catégories non ordonnées**
    - Bleu, Rouge, Vert
    - Epicé, Salé, Sucré
* **L’encodage ordonné**
  + Remplacer les catégories ordinales par des **nombres croissants** sans modifier la quantité d’informations contenues dans le jeu de données
* **L’encodage one-hot**
  + Associer pour **chaque valeur de la catégorie** une **nouvelle variable binaire** qui indique si la variable originale avait la valeur en question
    - Si N valeurs possibles, N-1 catégories (pour éviter le **dummy variable trap**)
      * Pour éviter que le modèle ne créé une linarité entre les nouvelles variables crées
        + Si valeurs bleu, blanc, rouge

Nouvelles variables blanc(0 ou 1), rouge(0 ou 1)

Bleu devient tacite et il n’existe pas de relation entre blanc et rouge

Sinon bleu + blanc + rouge = 1

Ainsi, le poids des valeurs sera

Y = intercept + a \* blanc + b \* rouge + bruit

Intercept équivaut donc à bleu (inféré tacitement)

Et a et b la déviation de la prédiction lorsque blanc est vrai ou rouge est vrai

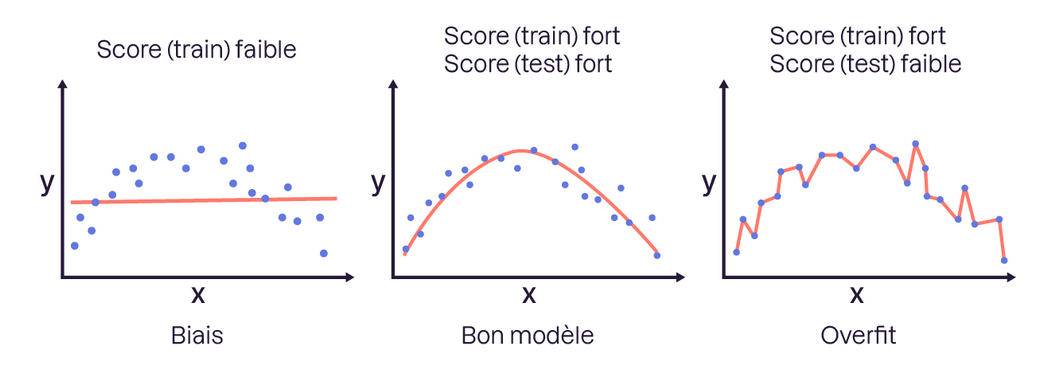
* + Attention à la **curse of dimension**
    - Dans le dataset de l’exemple, nous passons de 8 variables à 18
      * L’information des autres variables est diluée dans ces variables nouvellement crées
* L’encodage binaire
  + ???

## Améliorer le modèle

* Les modèles de régression linéaire et logistique n’ont pas de paramètres
  + Tout dépend des prédicteurs utilisés
  + Impossible de jouer sur des paramètres pour améliorer leur performance
* **Hyper paramètres**
  + Les paramètres d’un modèle qui modifient son comportement
    - Pour l’arbre de décision
      * Max\_depth
      * Criterion

### Reconnaitre un modèle qui sous-performe

* Pour évaluer un modèle, il faut scinder le dataset en 2 sous-ensembles (entraînement et test)
* Entraîner le modèle
  + Lui faire apprendre la dynamique et les relations entre les variables dans les échantillons d’entraînement
    - On suppose que, au sens statistique, le sous-ensemble de test, d’entraînement et de production ont la même distribution statistique
* Le calcul du score sur les données d’entraînement et de test vont nous permettre de distinguer un bon modèle d’un mauvais modèle
  + Score(train) faible + Score(test) faible
    - Modèle n’arrive pas à créer de corrélations fortes
      * On parle de biais du modèle
  + Score(train) fort + Score(test) fort
    - Modèle est bon et sait généraliser
  + Score(train) fort + Score(test) faible
    - Le modèle colle trop aux données d’apprentissage et ne sait pas extrapoler
      * Overfit



* + L’optimisation d’un modèle via ses paramètres consiste à trouver le juste équilibre entre le biais et l’overfit
* Un modèle qui overfit est un modèle trop complexe
  + Arbres
    - Trop de profondeur
  + Régression
    - Trop de variables
  + K-means
    - Trop de clusters
* Stratégies de remédiation
  + Augmenter taille du dataset d’entrainement
  + Régularisation

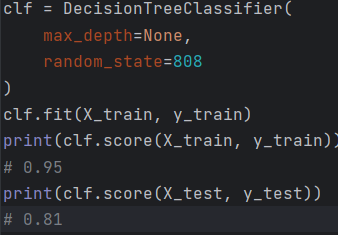
### Exemple

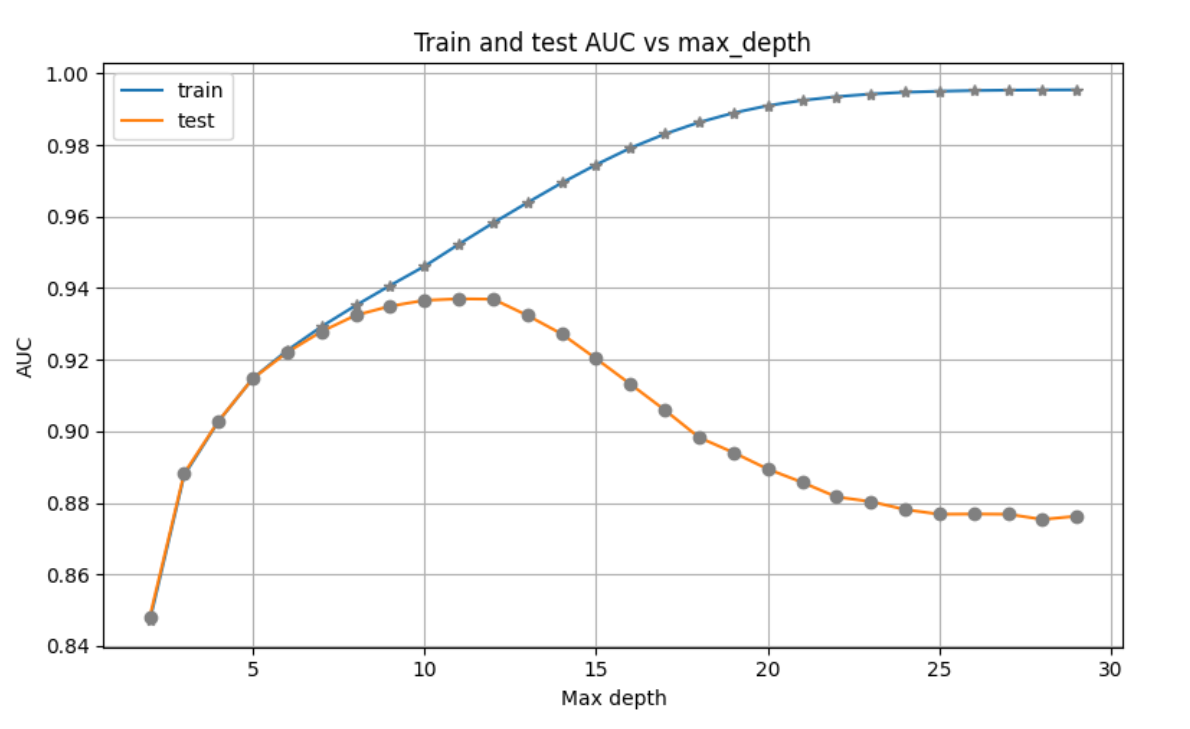
* Dataset des arbres de paris
  + Nous allons construire un modèle qui prédit le stade de développement
    - Jeune (arbre)
    - Jeune (arbre)Adulte
    - Adulte
    - Mature
  + Dans le dataset, ces données sont numérisées
    - 1
    - 2
    - 3
    - 4
* Comme modèle, nous allons utiliser un DecisionTreeClassifier de scikit-learn

### Minimiser le biais du modèle

* Plusieurs stratégies sont envisageables
  + **Ajouter** des **données**
    - Sur un dataset trop petit, le modèle n’aura pas assez d’exemples pour assimiler les dynamiques internes. Pour pallier à ce problème, nous pouvons
      * **Collecter** de nouvelles données et les ajouter au dataset
      * Utiliser des techniques d’augmentation de données qui **créent** des **échantillons artificiels** et gonflent donc artificiellement le dataset d’entraînement
  + **Feature engineering**
    - On **transforme** ou on **ajoute des variables** pour encoder plus d’informations exploitables par le modèle
  + **Modification** des **paramètres** du **modèle**

### Identifier et compenser l’overfit

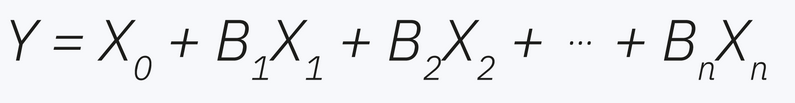
* 
  + Lorsque l’on retire le nombre maximum de profondeur des arbres décisionnels, on augmente la complexité, et on s’expose à de l’overfiting
    - Le modèle essaie de coller au maximum aux données d’entrainements
      * En conséquence, le modèle n’arrive pas à prédire des données nouvelles
* Overfit en régression linéaire quand il y a trop de variables prédictives
  + En règle générale, la complexité expose à l’overfit
  + Solution ici
    - Augmenter la quantité de données
      * Dilution des informations des variables
    - Réduire le nombre de variables
* Exemple de détection d’underfit / overfit
  + Visualisation du score via un AUC\_ROC de train et test avec un seuil de profondeur croissant



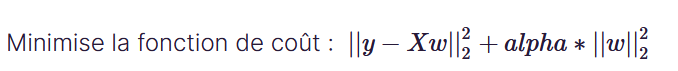
* A gauche du modèle
  + Sous-performance (biais)
* Au centre
  + Meilleure performance
  + Max\_depth 9 ou 10
* A droite
  + Overfit

## Augmenter la robustesse du modèle

### Problème de la multicolinéarité

* Multicolinéarité
  + Plusieurs variables indépendantes d’une équation de régression linéaire sont corrélées
  + Les variables multicolinéaires peuvent nuire aux prévisions des modèles sur des données inédites
    - Augmentent la complexité et overfiting
* Colinéarité
  + Uniquement deux variables indépendantes d’une analyse de régression sont corrélées
* Orthogonalité
  + Les variables indépendantes ne sont pas corrélées
* 
  + Y représente la variable prédite (variable dépendante)
  + X représente un prédicteur (variable indépendante)
  + B représente le coefficient de régression associé
    - Mesure la modification de Y pour cuaque unité de modification au niveau du prédicteur correspondant (Xn)
      * En supposant que tous les autres prédicteurs restent constants (9)

### Régularisation

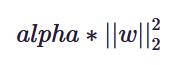
* On rajoute une contrainte au modèle pour qu’il ne puisse plus coller aux données du modèle
  + Régression linéaire avec régulation dans scikit-leanr
    - Ridge
      * 
      * Régression linéaire simple
        + 

y = variable cible

X = matrice des prédicteurs

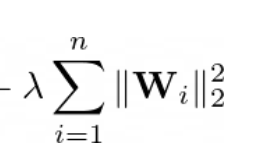
w = vecteur de coefficient de la régression linéaire

Fonction de coût est la norme quadratique de l’erreur d’estimation

* + - * Pour ridge
        + On ajoute  à la fonction de coût

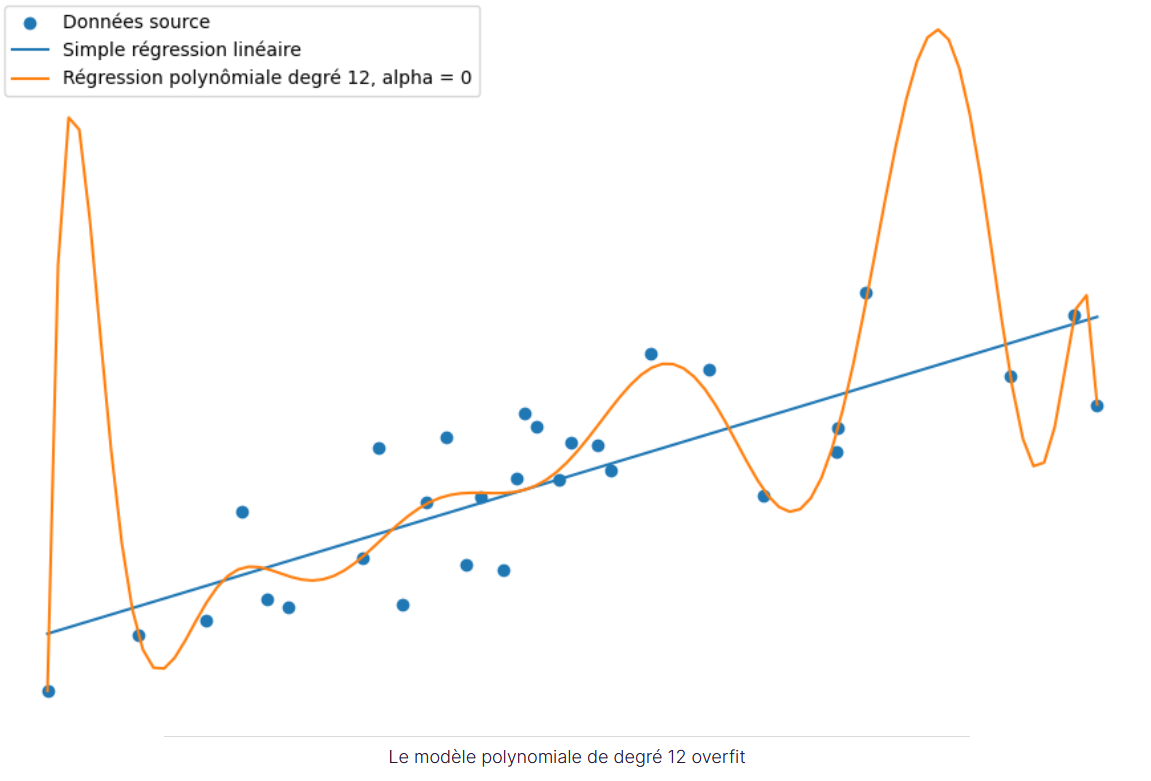
Contrainte (pénalité) sur les coefficients de la régression (10)

Somme des carrés des coefficients (11)

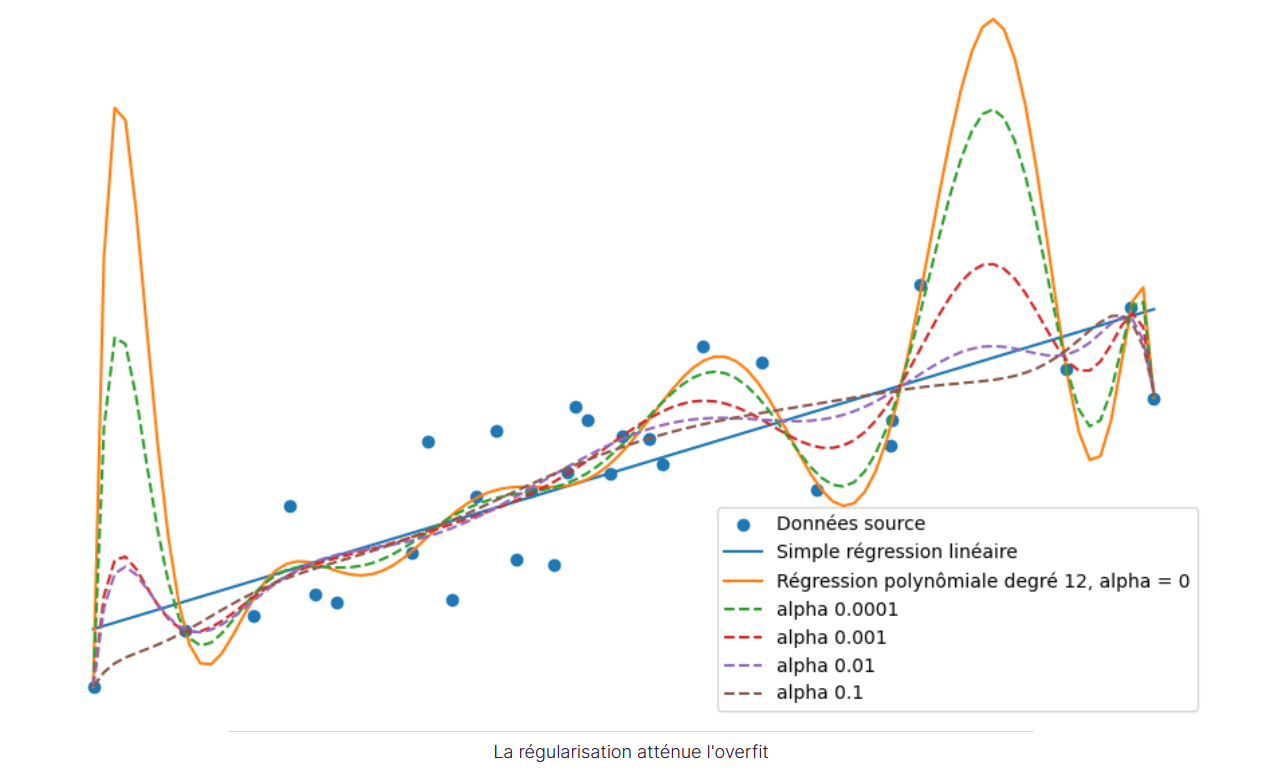


Alpha est le paramètre qui permet de régler la quantité de régularisation que l’on veut ajouter

* La régularisation va décourager le modèle d’assigner une importance excessive à une unique feature, et prévient également le modèle de devenir trop complexe
  + Pénalise les poids trop élevés
    - Amène à des poids plus léger et plus stable qui permettent une meilleure généralisation



* + Overfit du dataset via une régression polynômiale de degré 12



* + En augmentant la quantité de régulation (à travers des valeurs croissantes du paramètre alpha), on observe un effet d’atténuation de la sensibilité du modèle
* En résumé
  + La régularisation d’un modèle peut prendre plusieurs formes
    - L1 au lieu de L2
      * L2 = norme quadratique
      * L1 = norme de premier degré
        + 

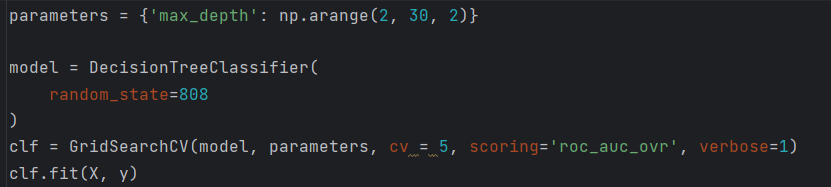
W est la somme de la valeur absolue des coefficients du vecteur w

* + - L’équivalent du modèle Ridge avec la régularisationL1 s’appelle Lasso
  + Pour les arbres de décision, régulariser le modèle consiste à limiter la profondeur de l’arbre ou à jouer sur d’autres paramètres
  + Dans les réseaux de neurones, la régularisation prend une autre forme comme la technique du Dropout, mais le principe reste le même
* **La régularisation permet de contrer la problématique de la multicolinéarité**
  + **Alpha va introduire une variable qui va casser cette multicolinéarité et donner une certaine indépendance à chaque variable**

### Validation croisée

* La répartition arbitraire des échantillons en sous-ensemble de test et d’entraînement peut être problématique si leurs caractéristiques statistiques divergent fortement entre les 2 sous-ensemble
  + Certains échantillons « pathologiques » peuvent se retrouver majoritairement dans un des sous-ensembles
    - Exemple
      * Dans le dataset des arbres à Paris, si la plupart des arbres particulièrement hauts se retrouvent dans la partie test, le modèle ne va pas prendre en compte ces hauteurs anormales dans son entraînement et donc ne pourra pas les prédire. Son score(test) sera alors mauvais alors que son score(train) sera bon
        + Impression d’overfit alors que non
* La validation croisée permet de réduire ce risque
  + Division du dataset en K sous-ensembles de taille égale
  + A tour de rôle, chacun des sous-ensembles jouera le rôle de sous-ensemble de test
    - Les K-1 autres sous-ensembles serviront à entraîner le modèle
  + Pour chaque configuration entraînement / test, on va calculer le score de performance du modèle
  + A la fin, on choisira le modèle qui offre la meilleure moyenne des scores sur toutes les configurations entraînement / test



* Existe plusieurs méthodes pour faire validation croisée dans scikit-learn
  + Méthode utilisée dans l’exo : GridSearchCV
    - Juste milieu entre automatisation totale et implémentation manuelle
  + 
    - Cv
      * Nombre de sous-ensemble, de splits de la validation croisée
        + On choisit le plus souvent une valeur autour de cv = 5
* Attention
  + On choisit
    - 14 valeurs possible du paramètre max\_depth
    - 5 plis pour la validation croisée
  + Donc on entraine
    - 14 \* 5 = 70 modèles
  + En cherchant à optimiser plusieurs paramètres à la fois, on risque de faire exploser le nombre de modèles et donc le temps d’entraînement du classificateur global
    - Il vaut mieux être circonspect et limiter l’espace des paramètres

### Résumé

* La régularisation
  + Permet d’éviter que le modèle ne colle trop aux données d’apprentissage
    - Et perde sa capacité d’extrapolation de données fraîches
  + Prend des formes différentes en fonction des modèles envisagés
    - Profondeur des arbres de décision
    - Régularisation de type
      * L2 (Ridge)
        + Pas de coefficient de corrélation à 0 ? (chatou)
      * L1 (Lasso)
        + Coefficient de corrélation possible à 0
* La validation croisée
  + Permet de s’affranchir de l’influence des échantillons hors normes en moyennant l’entraînement du modèle sur plusieurs sous-ensembles de test et d’entraînement
  + La fonction GridSearchCV permet à la fois de
    - Implémentation la validation croisée
    - Sélectionner les paramètres optimaux du modèle
      * Sachant qu’il n’y a pas de paramètres pour la régression linéaire ?!

## Ensemble learning (Random forests)

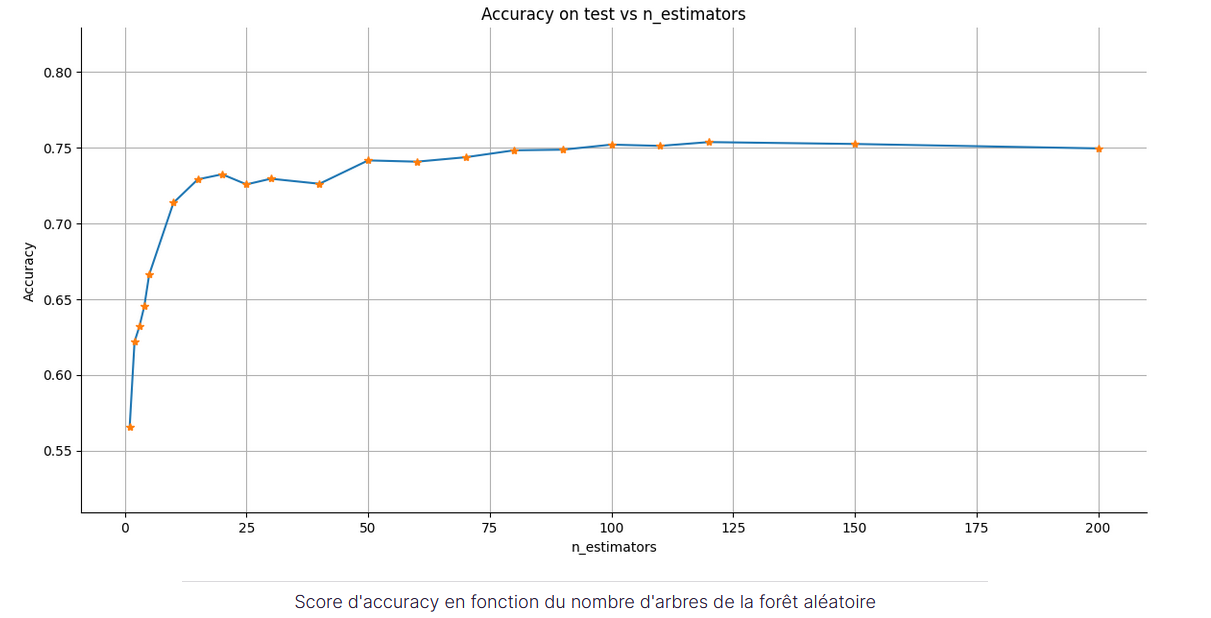
* Apprentissage d’ensemble
  + Combiner plusieurs modèles sous-optimisés pour
    - Construire un modèle global performant, stable, robuste
  + Principe applicable à de nombreux types de modèles mais rencontré surtout avec les arbres de décision
  + Concept
    - Au lieu de prendre l’avis d’un expert (le modèle optimisé), on va se fier au vote de nombreux amateurs
  + Dans la pratique, il permet de
    - Réduire
      * Le biais
      * L’overfit
    - Apporter
      * Robustesse et stabilité face
        + Aux outliers
        + Au bruit
* On considère deux techniques
  + **Bagging**
    - Appliqué aux arbres de décision
      * Donne naissance au modèle de forêt aléatoire
  + **Boosting**
    - Engendre la famille des XGBoost (12)
      * Plus délicat à optimiser que les forêts aléatoires
      * Souvent le modèle gagnant sur des données tabulaires

### Bagging et forêts aléatoires

* Bagging
  + Entrainer de multiples instances d’un modèle dont les performances sont par construction faibles
    - Ce type de modèle = weak learner (prédicteur faible)
      * Exemple : arbre fortement contraint en profondeur
  + Chaque instance est entrainée sur une partie des données d’entraînement et sur une partie des variables prédictives
  + On combine ensuite les prédictions des prédicteurs faibles pour former le modèle d’ensemble
    - Pour la régression
      * Moyenne des prédictions des prédicteurs faibles
    - Pour la classification
      * Vote (catégorie la plus souvent prédite par les prédicateurs faibles)
* Scikit-learn offre des forêts aléatoires avec comme prédicteurs faibles des arbres de décision
  + Pour la classification
    - RandomForestClassifier
  + Pour la régression
    - RandomForestRegressor
* On retrouve dans ces deux modèles les paramètres d’un arbre de décision (max\_depth, etc.), plus des paramètres spécifiques à l’apprentissage d’ensemble
  + N\_estimators
    - Nombre d’arbres de la forêt
  + Max\_features
    - Nombre de variables considérées pour chaque arbre
  + Bootstrap
    - Indique si chaque arbre est entraîné sur l’intégralité du jeu de données ou une partie échantillonnée
      * Le bootstrap est une technique d’échantillonnage très utile. A chaque sélection d’un échantillon, l’élément est remis dans l’ensemble source. L’ensemble échantillonnée contiendra donc des doublons, mais cela offre des propriétés statistiques fortes utiles

#### Réduire le biais

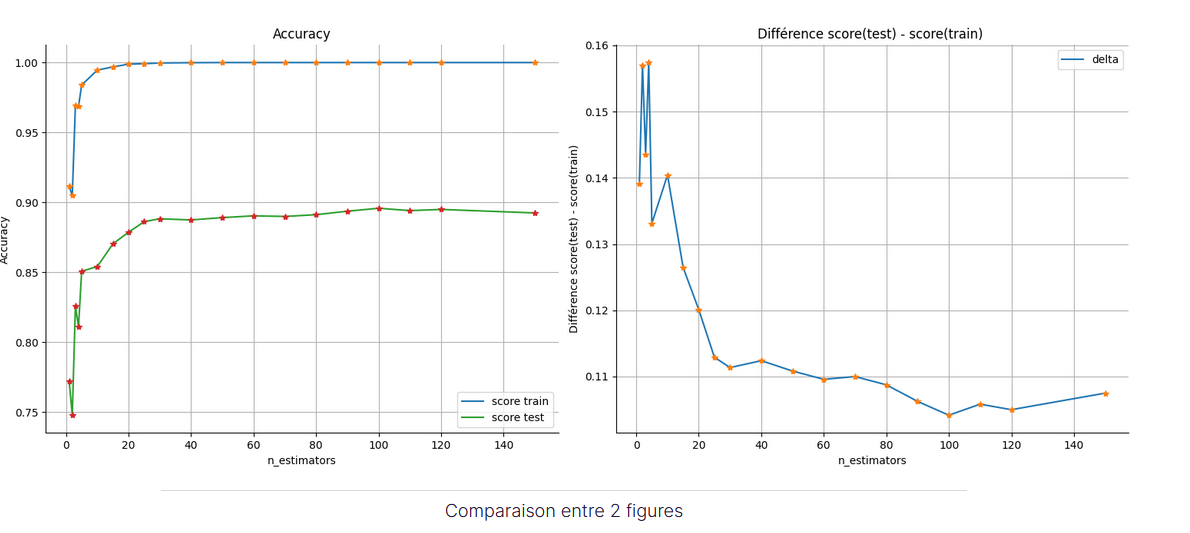
* On fait évoluer un nombre d’arbres identiques (profondeur 2) et on regarde l’évolution du score (accuracy)



* + Le score atteint un plateau à 50 arbres

#### Réduire la variance

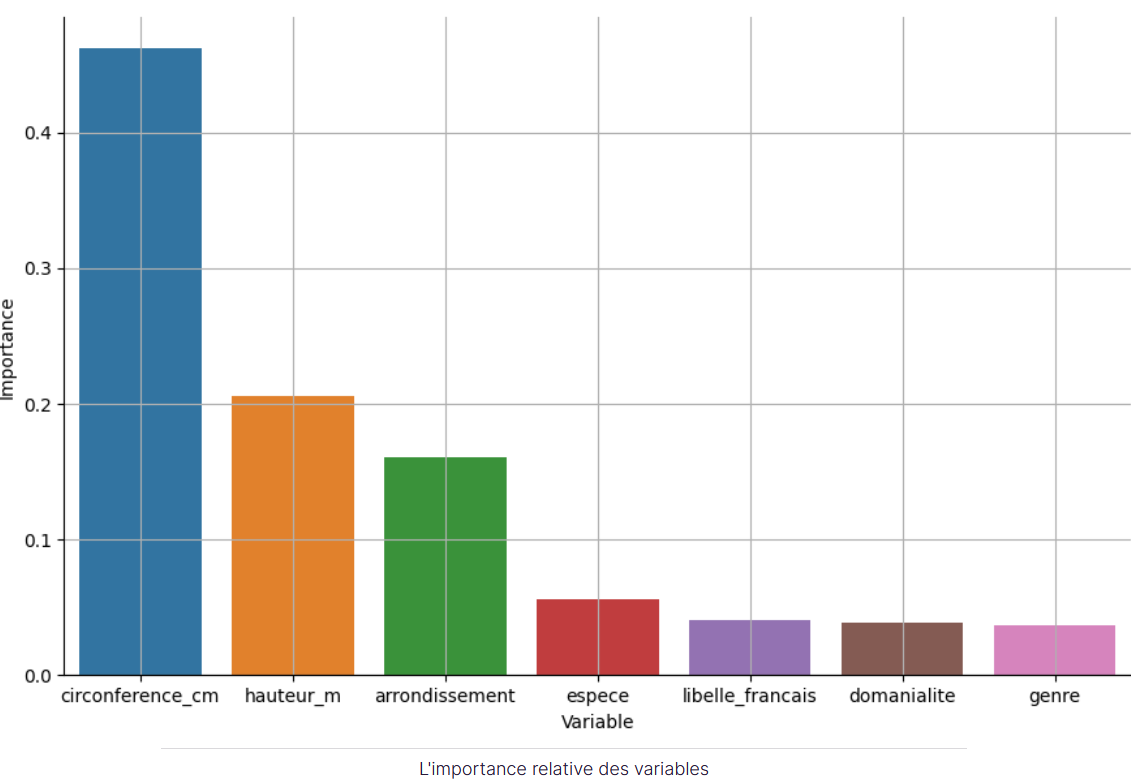
* Equivaut à réduire l’overfit du modèle
  + Si l’ensemble est constitué d’arbres dont la profondeur n’est pas contrainte
    - Chaque modèle peut être considéré comme overfité
  + Nous passons aussi le paramètre max\_features à None
    - Toutes les variables sont prises en compte dans chaque prédicteur faible



* + Le modèle a rapidement un score excellent en train, mais au fur et a mesure que l’on rajoute des arbres, la capacité d’extrapolation est améliorée
* Le foret aléatoire est un modèle fantastique pour réduire l’overfit de multiples façons
  + Augmenter le nombre d’arbres
  + Contraindre l’arbre de base en réduisant max\_depth
  + Réduire le nombre de variables sur lesquelles chaque arbre de base est entraîné

#### Estimer l’importance de chaque variable

* Chaque arbre n’est pas estimé sur toutes les variables
  + Il est donc possible d’estimer l’importance de chaque variable
    - Basé sur la moyenne et l’écart-type de l’accumulation de la baisse de l’impureté dans chaque arbre (13)
      * En effet, chaque variable n’est pas contenue dans tous les arbres. On peut alors estimer l’effet de l’incorporation de cette variable à un nœud et son impact sur l’impureté
      * Puis chaque variable est normalisée sur un intervalle de 1 pour meilleure comparaison



### Boosting (appliqué aux arbres)

#### Généralités

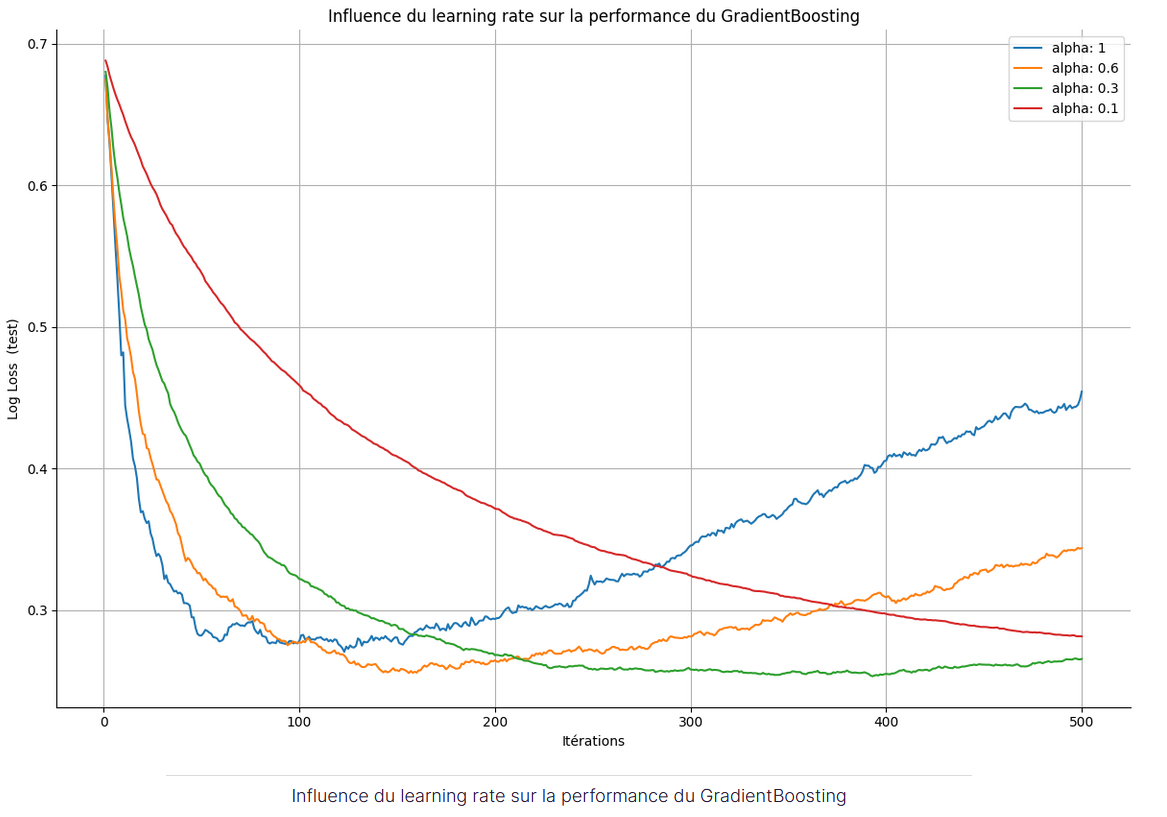
* Le modèle de forêt aléatoire est intrinsèquement parallèle
  + Les arbres sont entraînés en même temps sur des parties du dataset
* Le boosting enchaîne l’entraînement des prédicteurs faibles de façon séquentielle
  + Se concentre à chaque itération sur les échantillons qui ont généré le plus d’erreurs
* Boosting + arbres de décision = **XGBoost** (Extrême Gradient Boostring)
  + Progéniture d’une forêt aléatoire avec un gradient stochastique
  + Existe modèles scikit-learn
    - GradientBoostingClassifier()
    - GradientBoostringRegressor()
  + Existe d’autres variantes appelées
    - LightGBM
    - CatBoost
  + Toutes ces librairies implémentent des versions différentes de XGBoost et se distinguent par
    - Rapidité d’entraînement
    - Optimisation de la mémoire de l’ordinateur
    - Constitution des ensembles d’arbres
    - Traitement des variables catégoriques
    - Nom et signification des paramètres

#### Sélectionner les paramètres d’un Gradient Boosting

* Il faut prendre en compte un nouveau paramètre
  + Le learning\_rate (taux d’apprentissage)
    - Quantité de corrections prises en compte à chaque itération
    - Learning rate élevé
      * Fait converger l’algorithme rapidement au détriment de l’erreur
    - Learning rate faible
      * Ralenti la convergence de l’algorithme mais en même temps réduit l’erreur jusqu’à atteindre un plateau
* Optimiser un modèle devient plus difficile au fur et à mesure que le nombre de paramètres augmente
  + Stratégie globale (à adapter en fonction du contexte)
    - 1. Fixer max\_depth à 3 et le learning\_rate à 0.1
    - 2. Optimiser le nombre d’arbre en faisant un grid search avec validation croisé sur différentes valeurs (ex 10, 50, 100, 200, 500)
    - 3. Une fois nbr d’arbre optima trouvé, optimiser le learning rate
    - 4. Faire varier max\_depth
      * A ce stade, ne devrais plus changer grand-chose

#### Réduire la variance

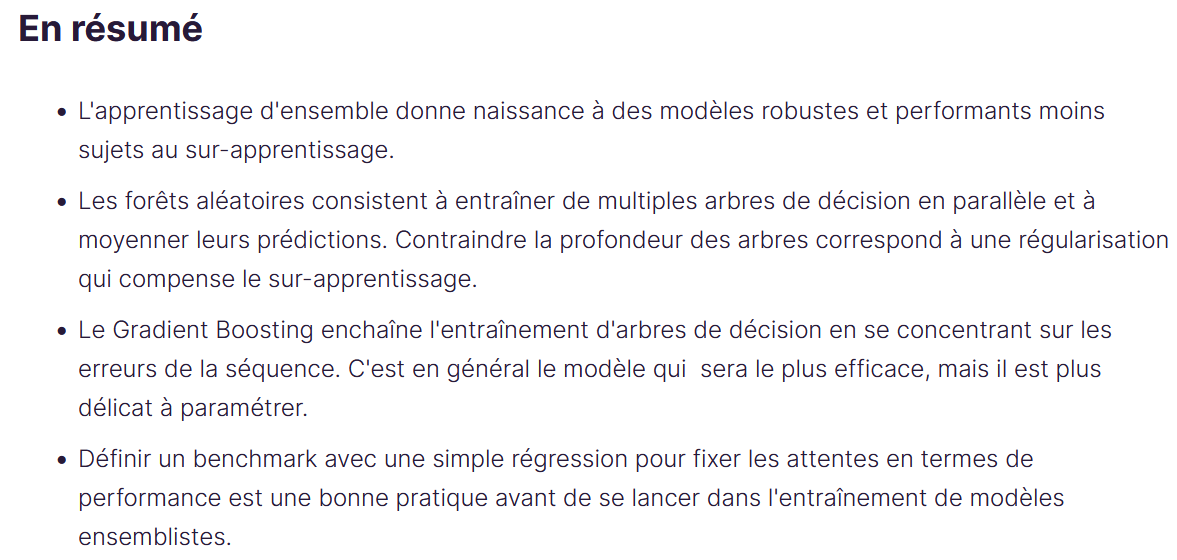
* Les forêts aléatoires overfittent rarement
* Les boosted trees peuvent plus facilement overfitter
  + Solution
    - Jouer sur learning rate
* Exemple de tuning de GradientBoostingClassifier
  + Utiliser la métrique log\_loss pour la classification
    - Prend en compte l’écart entre la probabilité de la prédiction d’appartement à la bonne classe et la classe réelle
      * Echantillon qui a un score de proba de 0.6 et qui appartient à la catégorie 1 sera pénalisé par rapport à un échantillon avec un score de 0.9
  + On obtient la probabilité des prédictions du modèle à chaque itération grâce à la fonction staged\_predict\_proba
    - On calcule le log\_loss à chaque itération et pour chaque valeur du learning rate



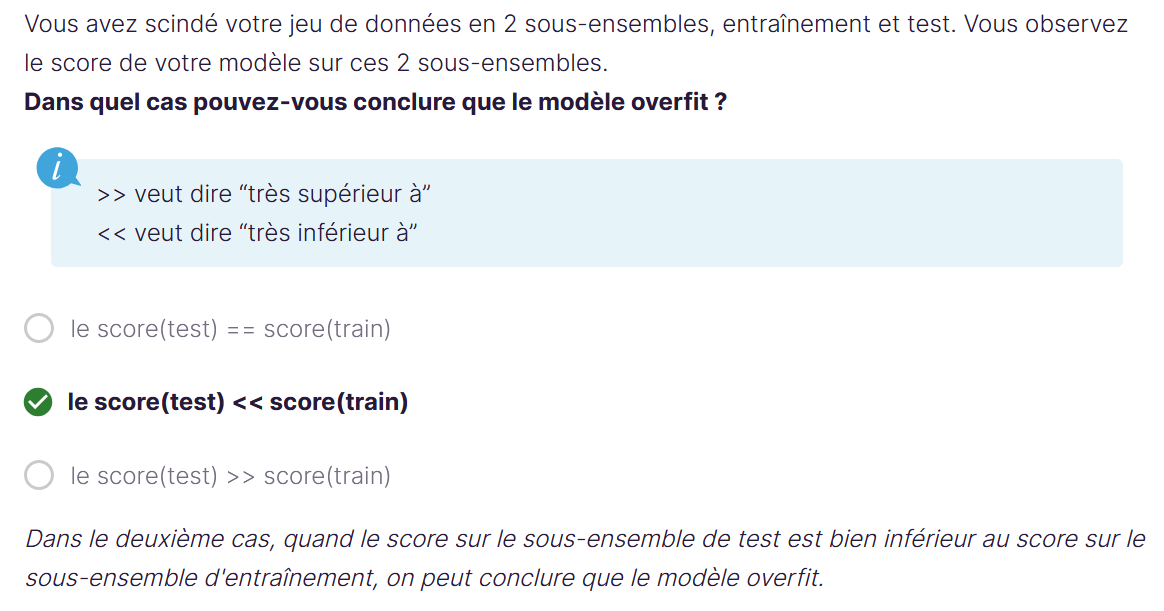
* Learning rate = 1 et = 0.6
  + Le modèle converge puis diverge
    - Le score sur le test repart à la hausse après avoir diminué
      * Le modèle perd sa capacité d’extrapolation
* Learning rate = 0.1
  + Modèle converge mais plus lentement que pour learning\_rate = 0.3

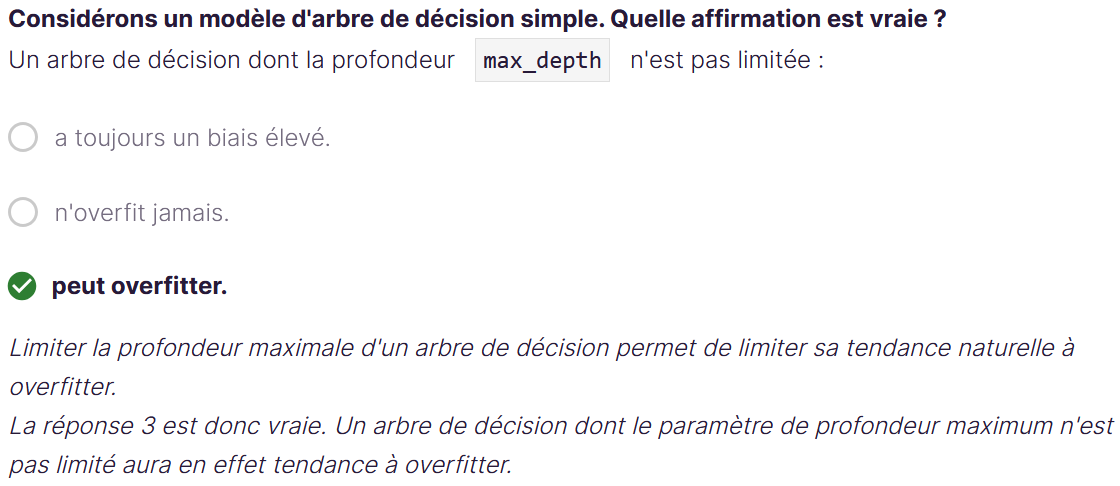
## Choisir le bon modèle

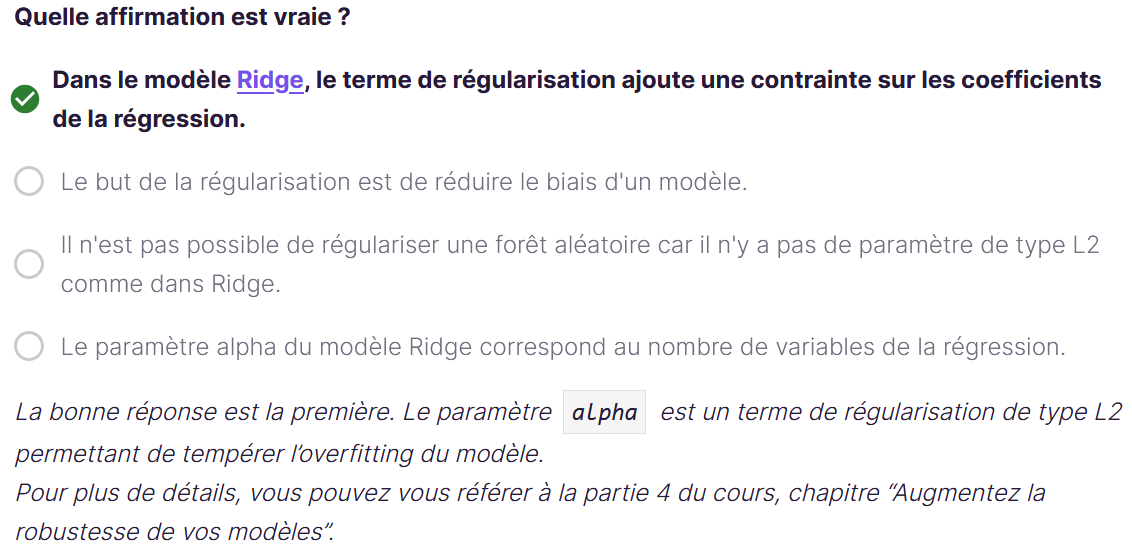
* Avoir une idée au début du projet de machine learning d’avoir une idée des performances que l’on pourrait obtenir
  + Selon le contexte
    - Une précision de 60% (à peine meilleur qu’un pile ou face) peut être fantastique
    - Une précision de 95% peut entrainer de lourdes pertes pour l’entreprise (détection de fraude)
* Stratégie simple consiste à commencer par une régression linéaire / logistique
  + Pour comprendre les dynamiques entre les variables de prédiction et la variable cible
    - Permet de sélectionner les meilleures variables prédictives et d’obtenir un benchmark des performances attendues
      * Permet de fixer rapidement des valeurs de bases pour évaluer les performances des modèles plus complexes

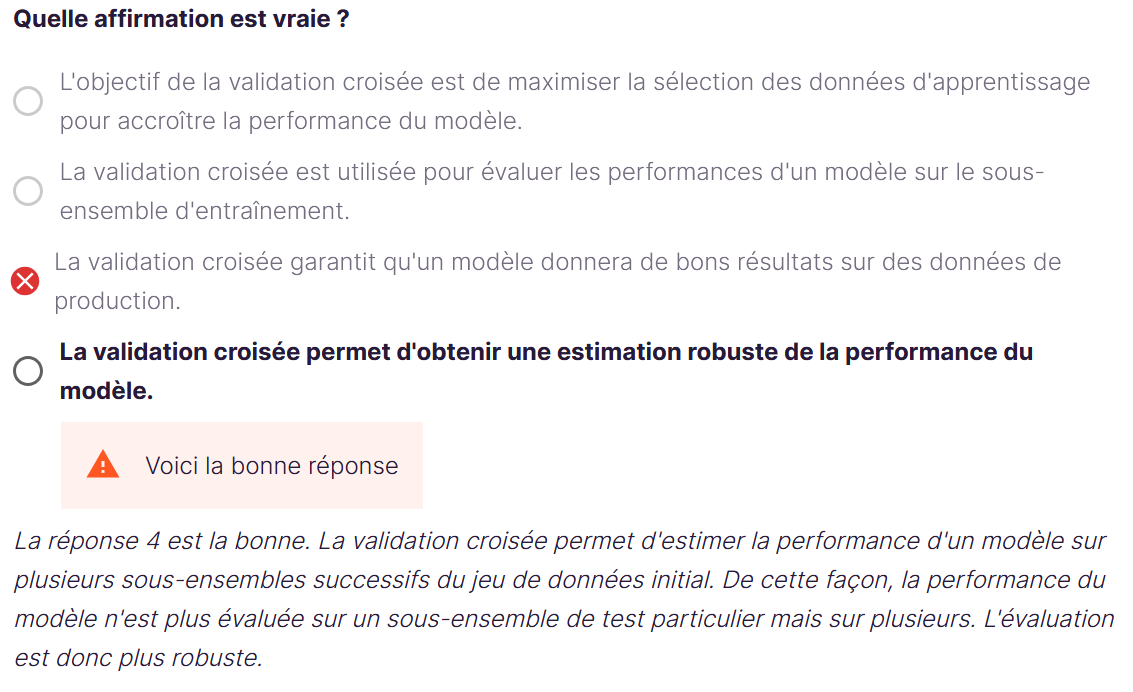


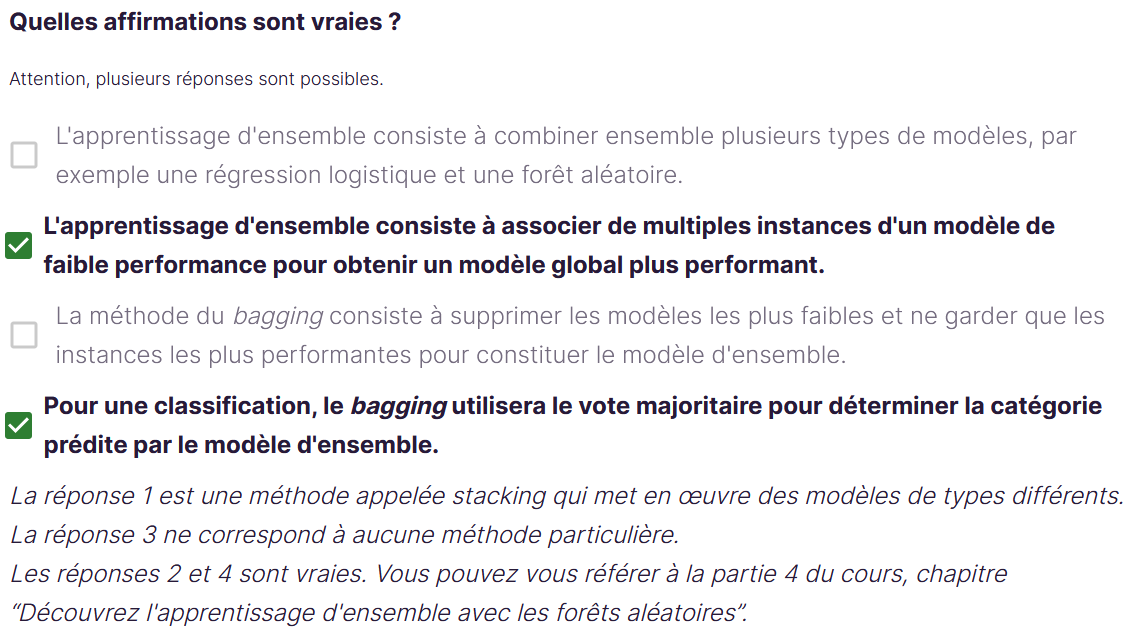
## Quizz

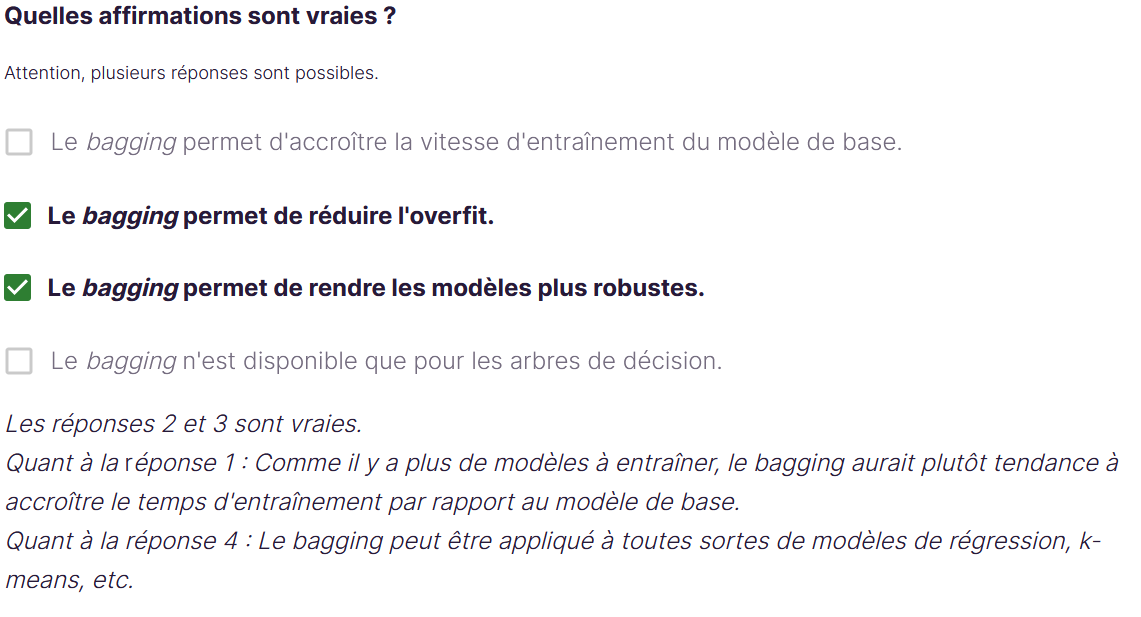


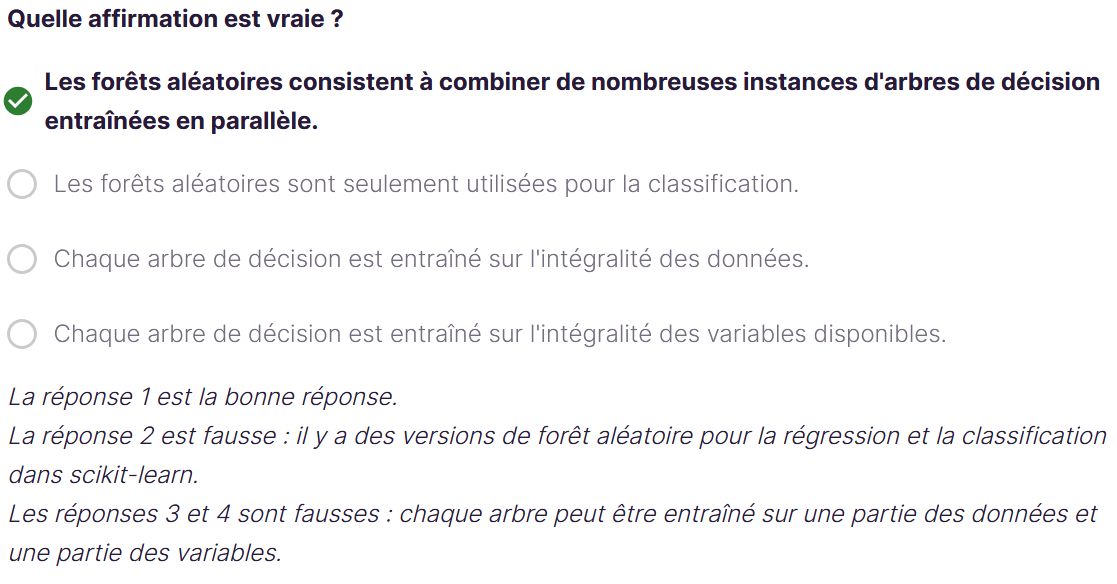


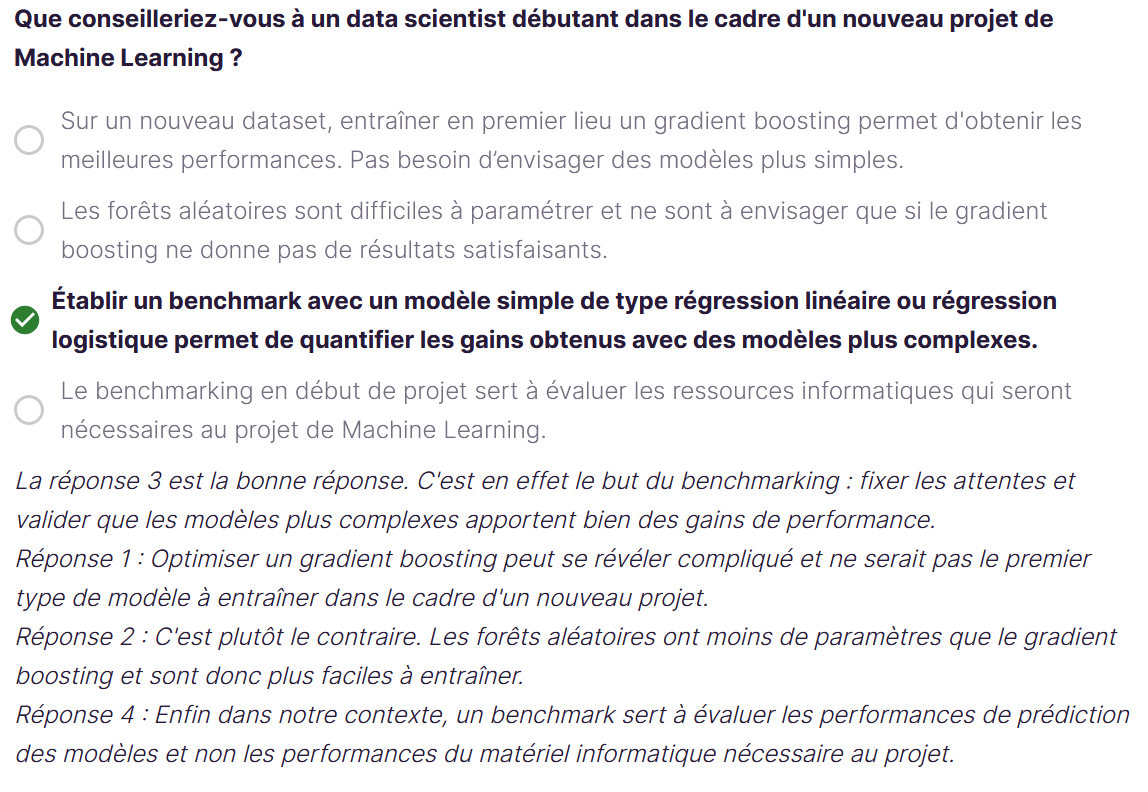












# Chatou

Pour un projet de prédiction électorale avec des données tabulaires, voici quelques pistes de modèles à tester :

1. **Modèles de base (baseline) :**
   * **Régression logistique** ou **régression linéaire** (si vous prédisez des scores ou des pourcentages) :  
     Ce sont des modèles simples qui vous permettront d'établir une première référence.
   * **Arbres de décision simples** :  
     Ils offrent une bonne interprétabilité et servent de base pour comparer des modèles plus complexes.
2. **Modèles d'ensemble :**
   * **Forêts aléatoires (Random Forests)** :  
     Elles gèrent bien les données structurées, réduisent la variance et fournissent des mesures d'importance des variables.
   * **Boosting (XGBoost, LightGBM, CatBoost)** :  
     Ces algorithmes, en particulier XGBoost, ont prouvé leur efficacité sur des données tabulaires complexes et peuvent souvent capter des interactions non linéaires. Ils requièrent cependant une optimisation fine des hyperparamètres.
3. **Autres approches :**
   * **Modèles linéaires régularisés (Lasso, Ridge, Elastic Net)** :  
     Ils permettent de gérer le surapprentissage, surtout si vous avez beaucoup de variables et souhaitez effectuer une sélection de caractéristiques.
   * **Modèles à base de réseaux de neurones** :  
     Si vous disposez d’un jeu de données volumineux, des architectures simples comme des perceptrons multicouches peuvent être explorées, même si pour des données tabulaires, leur performance est souvent comparable à celle des méthodes d'ensemble.
4. **Considérations complémentaires :**
   * **Feature Engineering :**  
     En électorale, les interactions entre variables (par exemple, le lien entre chômage et taux de criminalité) peuvent être importantes. N’hésitez pas à créer des variables dérivées ou à explorer des transformations non linéaires.
   * **Validation croisée et calibration :**  
     Utilisez des techniques robustes pour valider vos modèles et calibrer les probabilités de prédiction, car dans un contexte électoral, la qualité des probabilités prédites est souvent aussi cruciale que la simple classification.
   * **Explicabilité du modèle :**  
     Étant donné le caractère sensible et stratégique des prédictions électorales, des outils comme SHAP ou LIME peuvent aider à interpréter les décisions du modèle et à identifier quelles variables influencent le plus les prédictions.

# Références

1. **Intelligence artificielle (IA) ou machine learning (ML). *cloud.google.com.* [En ligne] [Citation : 03 02 2025.] https://cloud.google.com/learn/artificial-intelligence-vs-machine-learning?hl=fr.**

**2. Quelle est la différence entre l'IA et le machine learning ? *aws.amazon.com.* [En ligne] [Citation : 03 02 2025.] https://aws.amazon.com/fr/compare/the-difference-between-artificial-intelligence-and-machine-learning/.**

**3. Apprentissage supervisé et non supervisé: quelle est la différence ? *cloud.google.com.* [En ligne] [Citation : 31 01 2025.] https://cloud.google.com/discover/supervised-vs-unsupervised-learning?hl=fr.**

**4. Quelle est la différence entre l’apprentissage supervisé et non supervisé ? *aws.amazon.com.* [En ligne] [Citation : 03 02 2025.] https://aws.amazon.com/fr/compare/the-difference-between-machine-learning-supervised-and-unsupervised/.**

**5. shap. *github.com.* [En ligne] 08 03 2025. [Citation : 09 03 2025.] https://github.com/shap/shap.**

**6. lime. *github.com.* [En ligne] 2021. [Citation : 09 03 2025.] https://github.com/marcotcr/lime.**

**7. Choosing the right estimator. *scikit-learn.org.* [En ligne] [Citation : 11 03 2025.] https://scikit-learn.org/stable/machine\_learning\_map.html.**

**8. Confusion matrix. *wikipedia.org.* [En ligne] [Citation : 16 03 2025.] https://en.wikipedia.org/wiki/Confusion\_matrix#Table\_of\_confusion.**

**9. Qu’est-ce que la multicolinéarité ? . *ibm.com.* [En ligne] 21 11 2023. [Citation : 22 03 2025.] https://www.ibm.com/fr-fr/think/topics/multicollinearity.**

**10. 1.1.2. Ridge regression and classification. *scikit-learn.org.* [En ligne] [Citation : 22 03 2025.] https://scikit-learn.org/stable/modules/linear\_model.html#ridge-regression.**

**11. Understanding L2 Regularization: Enhancing Neural Network Training for Better Generalization. *medium.com.* [En ligne] 22 08 2024. [Citation : 22 03 2025.] https://vitalitylearning.medium.com/understanding-l2-regularization-enhancing-neural-network-training-for-better-generalization-95d16df2b46e.**

**12. Qu’est-ce que XGBoost ? . *ibm.com.* [En ligne] 09 05 2024. [Citation : 23 03 2025.] https://www.ibm.com/fr-fr/think/topics/xgboost.**

**13. Feature importance based on mean decrease in impurity. *scikit-learn.org.* [En ligne] 1.6.1. [Citation : 23 03 2025.] https://scikit-learn.org/stable/auto\_examples/ensemble/plot\_forest\_importances.html.**