Listes Algorithmes Importants en Machine Learning

Régression linéaire

- **Utilisation : Modélisation de relations linéaires entre une variable dépendante et une ou plusieurs variables indépendantes.**

- **Fonctionnement : Minimisation de l'erreur quadratique moyenne pour ajuster une ligne droite aux données.**

Régression logistique

- **Utilisation : Classification binaire ou multiclasse.**

- **Fonctionnement : Modélisation de la probabilité d'appartenance à une classe en utilisant la fonction logistique.**

Arbres de décision

- **Utilisation : Classification et régression.**

- **Fonctionnement : Construction d'un arbre basé sur des règles de décision pour partitionner les données.**

Forêts aléatoires

- **Utilisation : Classification, régression, détection d'anomalies.**

- **Fonctionnement : Agrégation de plusieurs arbres de décision pour réduire le surajustement.**

Machines à vecteurs de support (SVM)

- **Utilisation : Classification et régression.**

- **Fonctionnement : Trouver l'hyperplan qui maximise la marge entre les classes.**

K-moyennes

- **Utilisation : Clustering.**

- **Fonctionnement : Partitionnement des données en K clusters en minimisant la distance entre les points et les centres de cluster.**

K-plus proches voisins (K-NN)

- **Utilisation : Classification et régression.**

- **Fonctionnement : Prédiction basée sur la majorité des K voisins les plus proches dans l'espace des caractéristiques.**

Analyse en composantes principales (PCA)

- **Utilisation : Réduction de dimension.**

- **Fonctionnement : Transformation linéaire pour réduire la dimension tout en conservant la variance maximale.**

Naïve Bayes

- **Utilisation : Classification.**

- **Fonctionnement : Utilisation du théorème de Bayes pour estimer les probabilités conditionnelles.**

Régression polynomiale

- **Utilisation : Modélisation de relations non linéaires.**

- **Fonctionnement : Utilisation de polynômes pour ajuster les données.**

Réseau bayésien

- **Utilisation : Modélisation de dépendances probabilistes entre variables.**

- **Fonctionnement : Représentation graphique des relations entre les variables.**

K-means++

- **Utilisation : Clustering.**

- **Fonctionnement : Une amélioration de l'algorithme K-moyennes qui initialise les centroïdes de manière plus efficace.**

Isolation Forest

- **Utilisation : Détection d'anomalies.**

- **Fonctionnement : Crée un arbre de décision pour isoler des données anormales plus rapidement que les données normales.**

Algorithmes génétiques

- **Utilisation : Optimisation, sélection de caractéristiques.**

- **Fonctionnement : Utilise des opérateurs génétiques (croisement, mutation) pour trouver des solutions optimales.**

Perceptron Multicouche (MLP)

- **Utilisation : Classification, régression, approximation de fonctions.**

- **Fonctionnement : Réseau de neurones avec une ou plusieurs couches cachées.**

Machines à vecteurs de support à noyau (SVM à noyau)

- **Utilisation : Classification et régression non linéaires.**

- **Fonctionnement : Extension des SVM pour gérer des données non linéaires en utilisant des noyaux.**

Méthodes ensemblistes (AdaBoost, Gradient Boosting, XGBoost)

- **Utilisation : Amélioration de la performance des modèles de base.**

- **Fonctionnement : Combinaison de modèles faibles pour créer un modèle plus puissant.**

Méthodes de réduction de dimension (t-SNE, LLE, UMAP)

- **Utilisation : Visualisation de données à haute dimension.**

- **Fonctionnement : Projection des données dans un espace de dimension réduite.**

Régression de Poisson

- **Utilisation : Modélisation de données de comptage.**

- **Fonctionnement : Modèle de régression adapté pour les données de comptage.**

Conseil pour le choix d'un modèle :

1-La Quantité de données que vous avez.

Par exemple, si vous avez de gros Dataset, les modèles adaptés seront plus la Régression Logistique / les Réseaux de Neurones.

A l'inverse si vous avez peu de données, prenez plutôt des algo comme le Support Vector Machines / K-Nearest Neighbours.

=> Si vous avez moins de -100 000 points -> n'importe quels modèles de ML fera l'affaire.

=> Si vous avez plus de +100 000 points -> modèles qui fonctionnent avec la Descente de Gradient.

2-Travailler uniquement avec les algo que vous comprenez.

Sinon, vous risquez de mal déployer l'algo et ou de mal optimisé ses hyper-paramètres.

3-La Structure de vos données (Structurées / Non-Structurées).

On considère en ML que les données qui sont des images, du texte, du son etc

=> ce sont des données non structurées.

A l'inverse, données tabulaires (Excel, Csv..)
=> ce sont des données structurées.

Quand on est en présence de données non-structurés :
Utilisation de réseaux de neurones (Deep-Learning).

Quand on est en présence de données structurés :
Utilisation de modèles (machine-Learning).

4-La Normalité des données.
Il existe 2 types de modèles, Paramétriques / Non-Paramétriques.

=> modèle paramétrique = c'est un modèle qui correspond à une fonction avec un nombre de paramètre limité et défini à l'avance.
Exemples :

- **modèles de régression linéaire**
- **modèles de régression logistique**
- **modèles de Naives Bayes**
- **modèles de petit réseau de neurones**

Explication rapide :

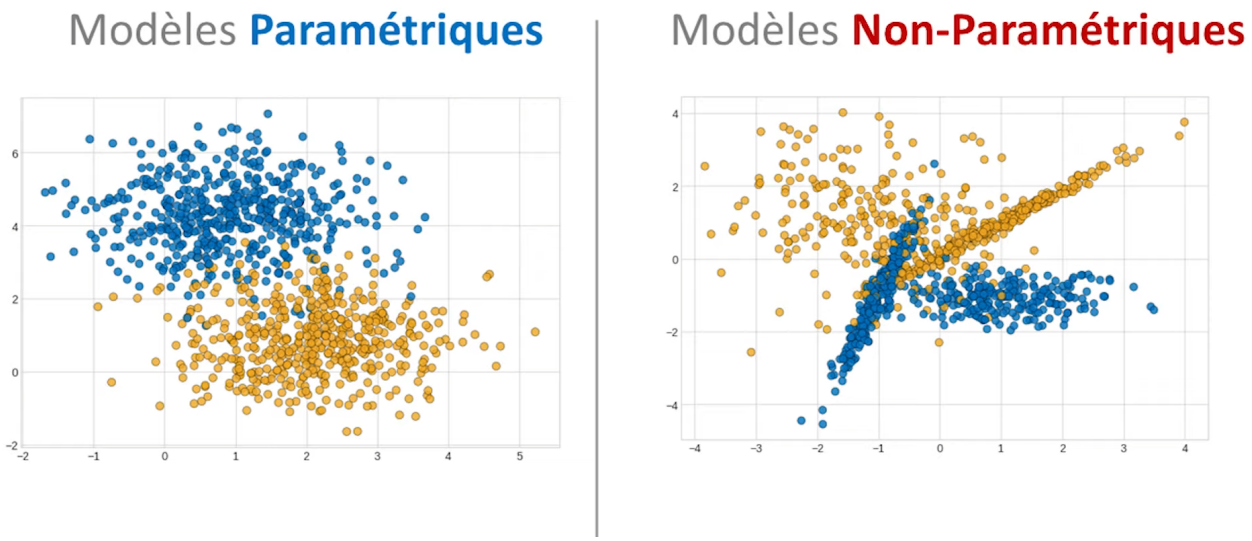
Si on prend par exemple la régression linéaire, alors f(x)=ax+b, on a 2 paramètres ici a et b, ni plus ni moins.

=> modèles non-paramétriques = c'est un modèle qui correspond à une fonction avec un nombre de paramètres infini et inconnu à l'avance.

Exemples :

- **Support Vector Machines**
- **Arbres de décision**
- **Random Forest**
- **K-Nearest Neighbors**

Les modèles paramétriques fonctionnent très bien sur les données qui suivent des lois de probabilités normales.

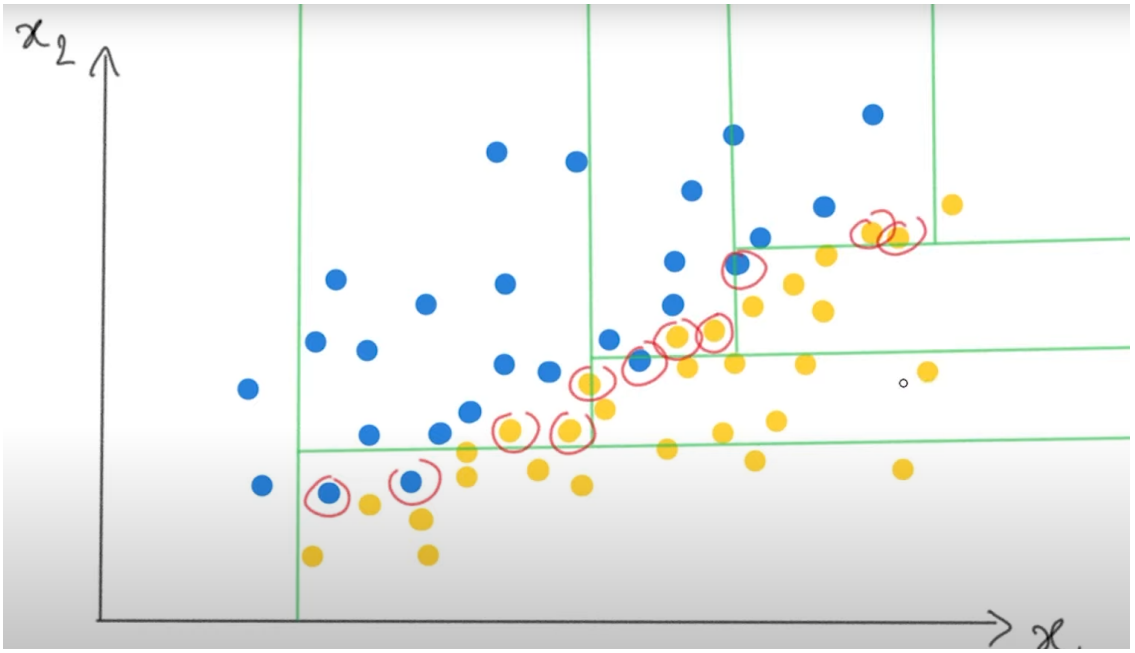


On peut parfois (ça dépend des données) en faisant du pré-processing et ou du features engineering

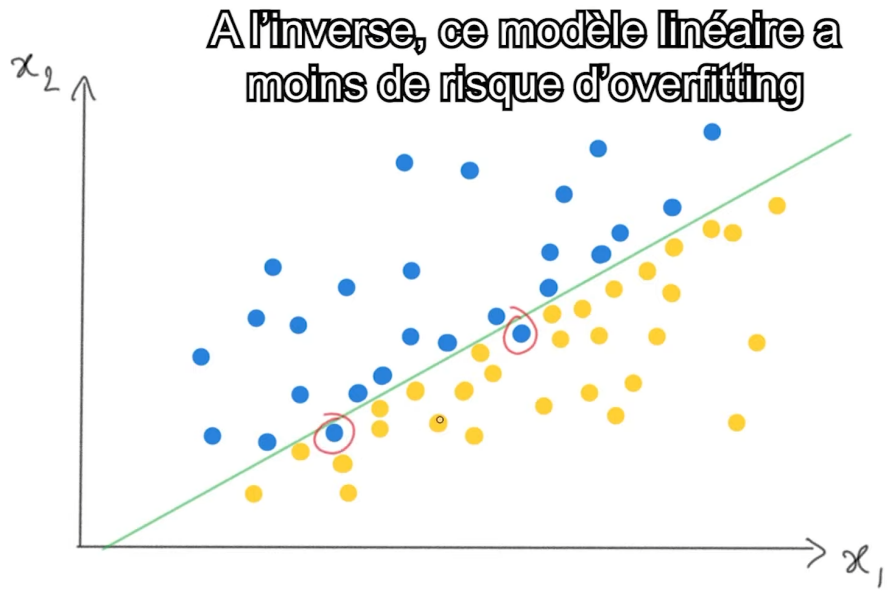
transformer nos données de sortes à ce qu'elles suivent des lois de probabilités normales.

5-Quantitées de Variables Quantitatives / Qualitatives dans notre Dataset.

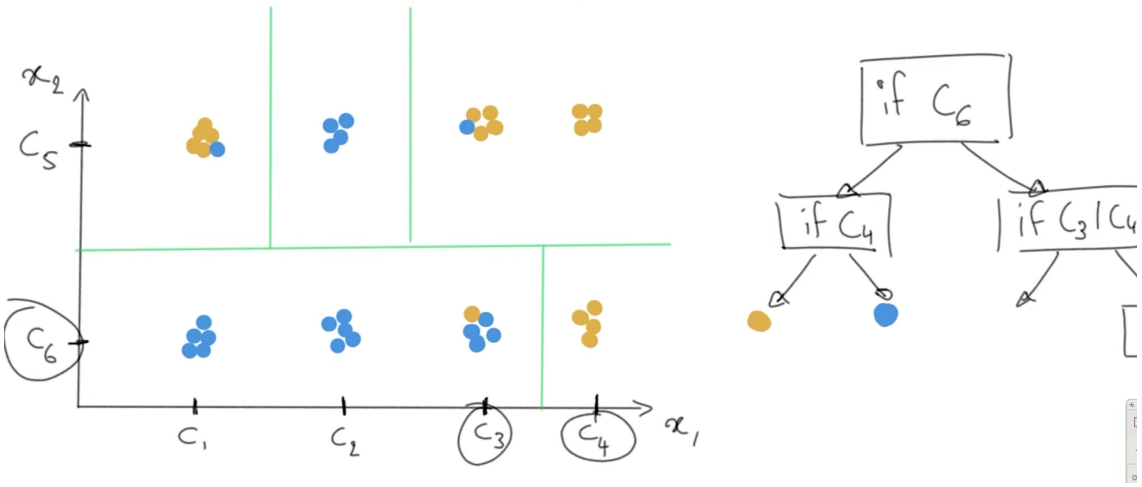
=>Il existe certains types de modèles comme les arbres de décisions qui ne sont pas efficaces lorsqu'on a beaucoup de variables quantitatives et surtout quand on observe des relations linéaires dans ces variables quantitatives.



On voit ici qu'il y a une classification mal faite à cause du système des modèles comme les arbres de décisions qui vont classer en traçant des droites orthogonales (forme d'escalier).



On voit à l'inverse qu'ici un modèle linéaire fait moins d'erreurs.



Mais si on a des données avec beaucoup de catégories, il ne faut pas hésiter à utiliser des arbres de décisions.

Conseil Bonus : Commencez TOUJOURS par utiliser le modèle le plus simple et après si besoin allez vers du plus complexe.

Dans la Réalité, le mieux étant de tester tous les modèles (faire une Pipeline par exemple), et regarder celui qui va fournir la meilleure performance.

```
preprocessor = make_pipeline(PolynomialFeatures(2, include_bias=False), SelectKBest(f_classif, k=10))
```

```
RandomForest = make_pipeline(preprocessor, RandomForestClassifier(random_state=0))
AdaBoost = make_pipeline(preprocessor, AdaBoostClassifier(random_state=0))
SVM = make_pipeline(preprocessor, StandardScaler(), SVC(random_state=0))
KNN = make_pipeline(preprocessor, StandardScaler(), KNeighborsClassifier())
```

Les données sont toujours différentes, on ne peut pas faire de règle bien précise pour la distribution de ces données réelles, donc la meilleure méthode et de tester les algorithmes et dans les faits voir celui qui a la meilleure performance.
Le plus important restera la manière dont sont traitées nos données (pre-processing) avant même le choix du modèle (qui jouera une influence moindre).