

# Machine Learning

## Collecte, préparation des données et mise en œuvre

Cours 1

# Un cours et un atelier

- BDOE826 Machine Learning: Collecte, préparation des données et mise en œuvre
  - Traitement préalable des données pour le Machine Learning.
  - Prédire les rescapés du Titanic avec des algorithmes simples.
- BDOE827 Atelier: Mise en œuvre du Deep Learning
  - Prédire les rescapés du Titanic avec des réseaux de Neurones.

# Contenu du Cours

- Ces cours introduisent le machine Learning avec des algorithmes simples puis aborde les réseaux de Neurones, qui sont de loin la technologie dominante en IA.
- Philosophie / Ethique < 1%. Discussion possible en début de cours.
- Fondements Mathématiques : 20 %. Pas d'évaluation dessus.
- Ingénierie # 80 %
  - Méthodologie
  - Utilisation des principales librairies
  - Construction d'un process d'apprentissage (« Pipe »)
- 1/3 cours, 2/3 TP environ. Les TPs sont notés.

## Objectifs

Vous rendre opérationnels sur un projet intégrant du Machine Learning.

# Le Prof

- Ingénieur en Informatique (ENSIIE)
- Développeur puis Chef de Projet puis CTO
- NLP Program Manager pour Case Law Analytics.
- Consultant pour la banque publique d'investissement (BPI).
- Enseignant Machine Learning à L'EPSSI depuis 4 ans.
- CTO-Fondateur Interpets : Analyse IA du comportement animal.

# Signalétique



- Ce cours contient des mathématiques.
- Un niveau Terminale S est nécessaire pour tout comprendre.
- Ce cours a pour objectif de vous donner une compréhension profonde, mais aussi d'être accessible à tous.
- Une signalétique a donc été introduite :



**A connaître absolument.**

Nécessaire à la fois à la suite du cours et aux TP.



Optionnel.

Explications supplémentaires plus avancées, pour ceux que ça intéresse.

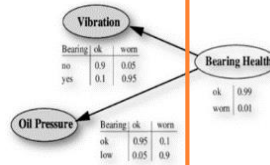
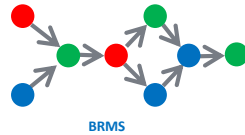


Piège à éviter en TP.

- Les slides sans signalétique sont des explications / Illustrations.
- Faciles à comprendre, mais pas indispensables à connaître.

# Concepts préalables

# Types de modèle IA



Réseaux Bayésiens



Arbres de décision



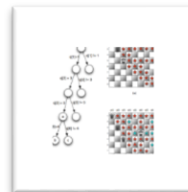
Modèles de forêt

Données

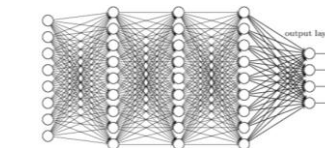
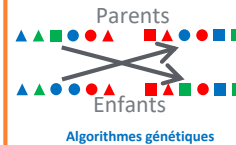
Machine Learning

Règles

Expertises



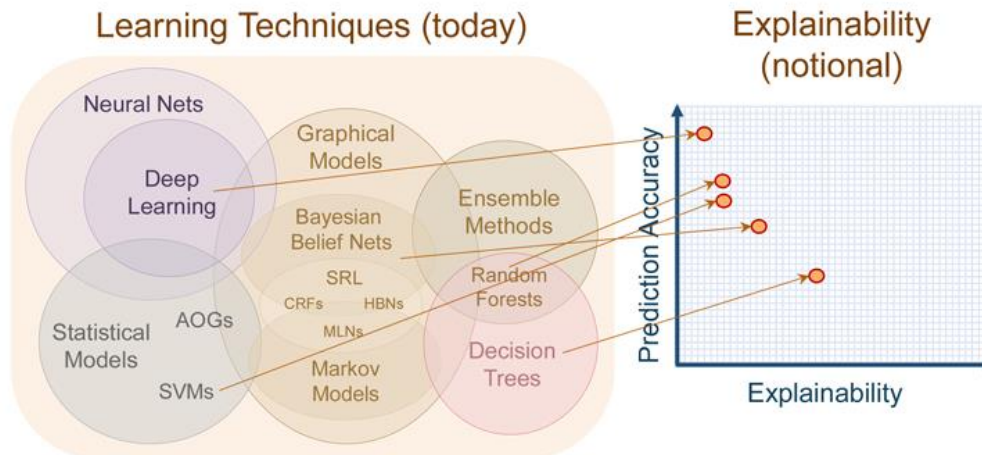
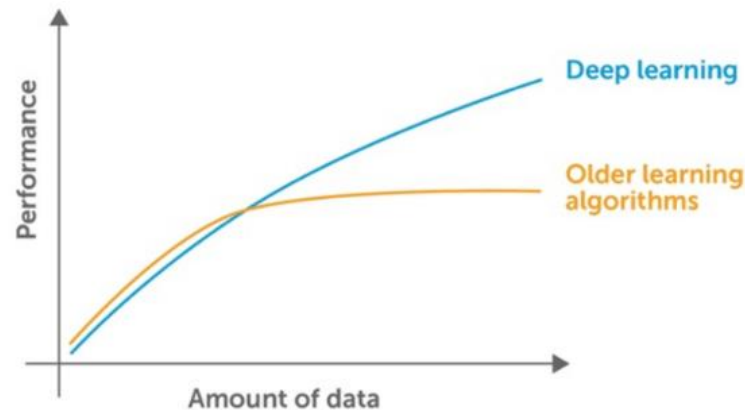
Programmation par contraintes



Réseaux de neurones

Boîte noire

# L'apprentissage profond





# Apprentissage et Inférence



- L'apprentissage consiste à faire apprendre le traitement d'un problème au programme qui ne sait rien faire au départ.
- Après l'apprentissage on peut questionner le programme pour le problème concerné. C'est l'inférence.

## Apprentissage

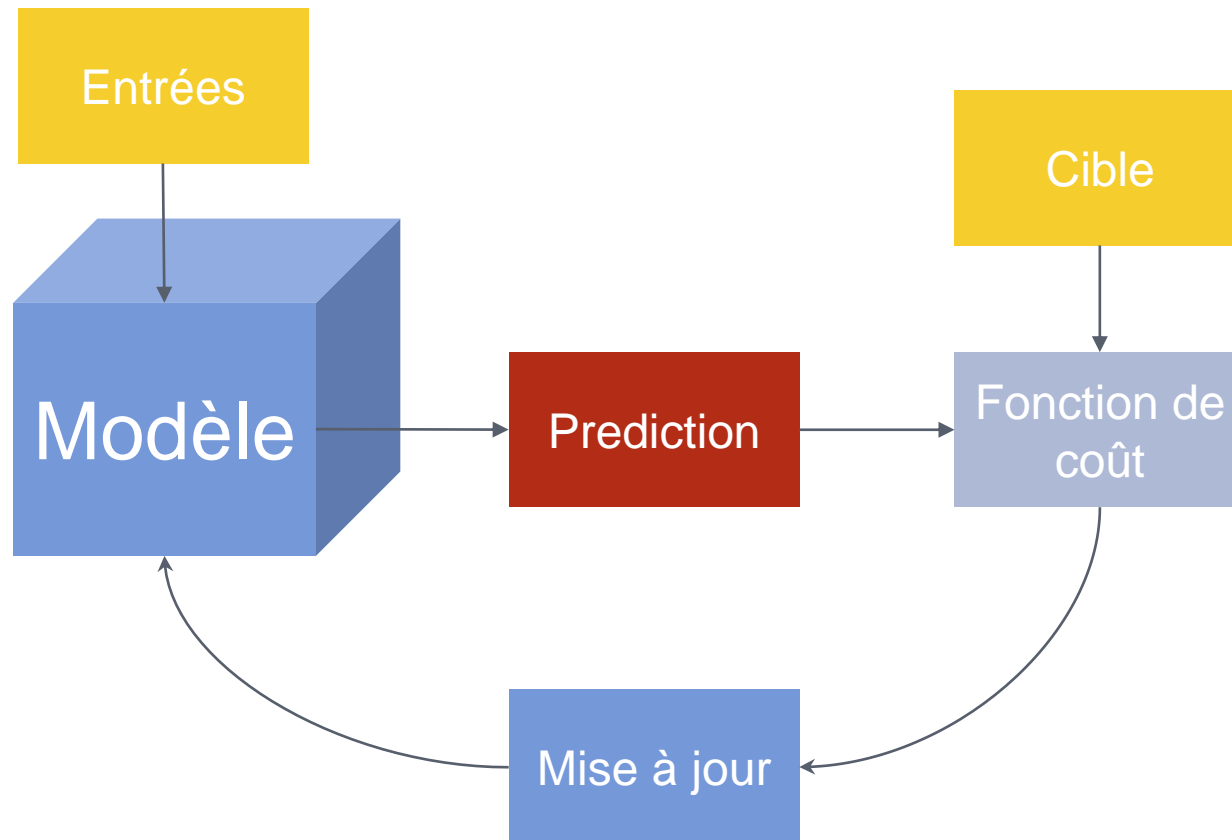


## Inférence




Panda  
Roux

# Apprentissage



# Python

- **Python** est un langage interprété, orienté objet, dynamiquement typé.
- Classé N°1 à l'indice TIOBE des langages depuis 2020. 
- Langage des scientifiques et Data Scientists, mais aussi scripting, web...
- De nombreuses bibliothèques disponibles, notamment pour le traitement automatique. Ces bibliothèques sont souvent en C++, pour gagner en performance.
- Spécificité syntaxique : donne un sens aux indentations

Fonction <b>factorielle</b> en C	Fonction <b>factorielle</b> en Python
<pre>int factorielle(int n) {     if (n &lt; 2) {         return 1;     }     else {         return n * factorielle(n - 1);     } }</pre>	<pre>def factorielle(n):     if n &lt; 2:         return 1     else:         return n * factorielle(n - 1)</pre>

# Notebooks

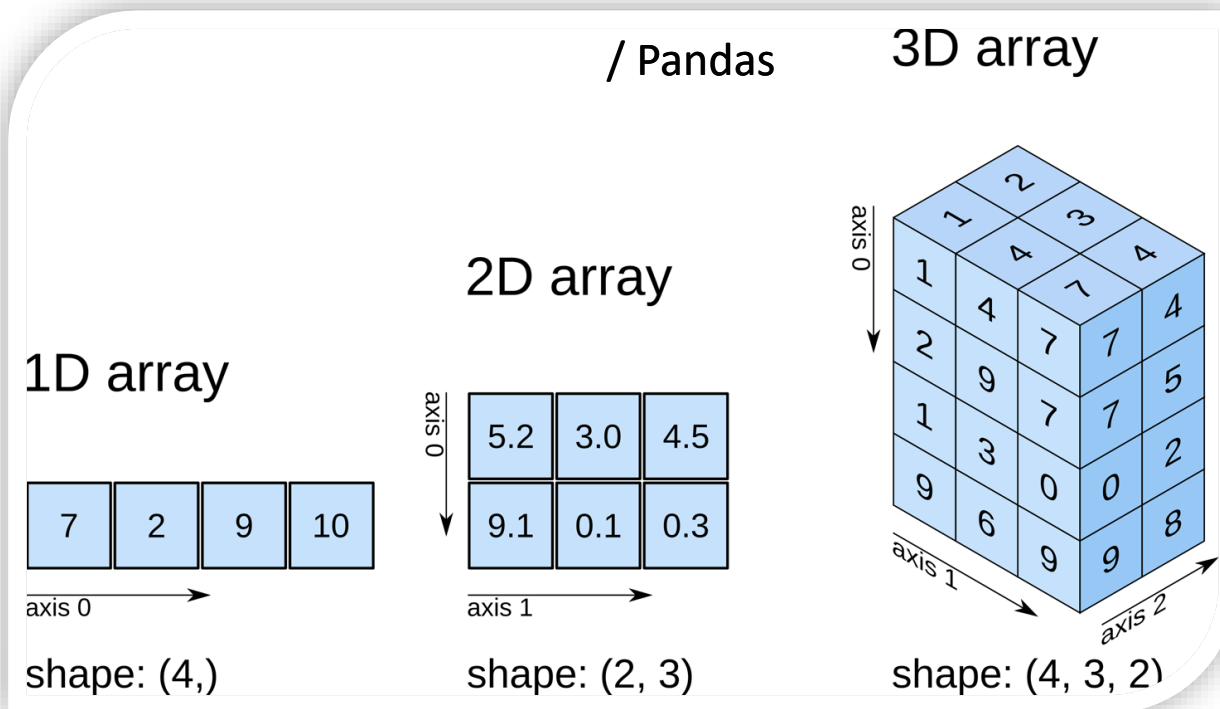
- **Jupyter Notebook** est un environnement de développement basé sur le web.
- Vient du monde Python, mais supporte maintenant plus de 40 langages.
- Extension VS Code, ou disponible en Ligne (Google Collab par ex).



# Numpy / Pandas



- **Numpy** est une librairie Python destinée à manipuler des matrices ou tableaux multidimensionnels (tenseurs), ainsi que des fonctions mathématiques opérant sur ces tableaux. Peut aussi contenir des chaînes de caractères.
- Il sert de format à de nombreuses autres bibliothèques: SciPy (calcul scientifique), matplotlib (visualisation), scikit-learn (ML)...



# Pandas

- **Pandas** est une bibliothèque Python de manipulation de données. Ses objets principaux sont **Dataframe** et **Serie**.
- On peut le voir comme un tableau Numpy en 2D ou 1D, avec un index et des noms de colonne.



Column Label/ Header	0	1	2	3	4	
Index Label	Name	Age	Marks	Grade	Hobby	
0	S1	Joe	20	85.10	A	Swimming
1	S2	Nat	21	77.80	B	Reading
2	S3	Harry	19	91.54	A	Music
3	S4	Sam	20	88.78	A	Painting
4	S5	Monica	22	60.55	B	Dancing

Diagram illustrating a Pandas DataFrame structure with labels and indices.

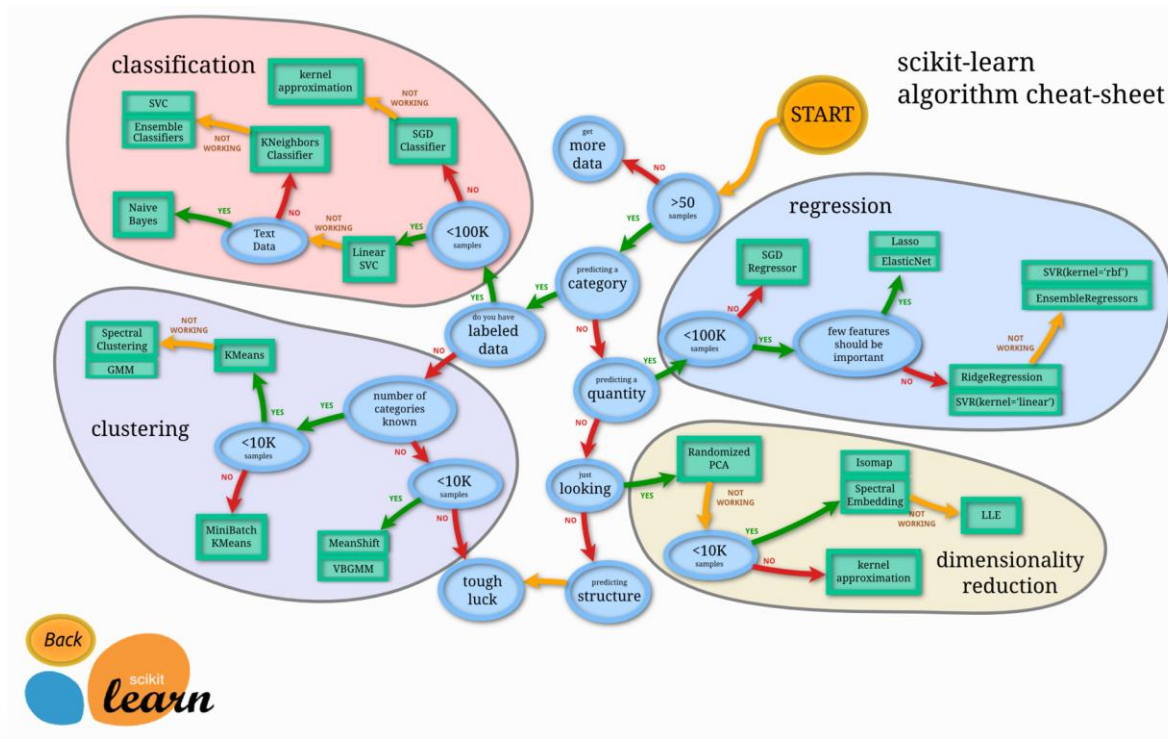
Labels and Indices:

- Column Index:** Points to the header row (Name, Age, Marks, Grade, Hobby).
- Row Index:** Points to the index column (0, 1, 2, 3, 4).
- Row:** Points to a specific row (e.g., Row 3).
- Column:** Points to a specific column (e.g., Column 2).
- Element/ Value/ Entry:** Points to a specific cell (e.g., 88.78).

# Scikit Learn (sklearn)



- **Scikit-learn** est une boîte à outils Python pour le Machine Learning.
- Créé en France (INRIA), utilisé mondialement.
- Elle intègre de nombreux algorithmes IA, mais ne fonctionne pas sur GPU.
- API standardisée pour tous les algorithmes (objets **Classifieur** et **Regressor**).
- Outils de préparations de données, combinaisons et évaluation de modèles.





# Le Dataset Iris



**Iris Versicolor**



**Iris Setosa**



**Iris Virginica**

	sepal length	sepal width	petal length	petal width	target
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa

- 50 exemples pour chaque type d'Iris.
- Disponible dans Scikit\_Learn en utilisant la fonction `load_iris()` du package [sklearn.datasets](#).
- Pour les besoins de l'exemple, des valeurs manquantes ont été ajoutées.



# Utilisation de Collab

- Par défaut, vous pouvez utiliser **Google Colab** pour réaliser les TPs. Mais vous êtes libre d'utiliser un autre environnement qui vous est plus familier (Jupyter Lab, VS Code...).
- Rendez-vous sur <https://colab.research.google.com/> et connectez-vous avec un compte Google.
- Allez dans « File/Open Notebook »
- Choisissez l'onglet GitHub
- Entrez le nom du repository « <https://github.com/jch44/EPIS-ML-I1> »
- Sélectionnez `tp_MachineLearning.ipynb` et ouvrez le dans un nouvel onglet.
- Le fichier n'est pas modifiable. Dans File choisissez Save Copy in Drive.
- Il faut maintenant récupérer les données avec lesquelles nous allons travailler.



# Récupération des données


- Recupérez et décompressez le dossier data.zip dans votre learning Box.
- Ajouter un dossier Data dans Collab.
- Uploader les données passagers.csv et test.csv (ou iris.csv pour l'exemple)



- Les données disparaîtront entre deux connections, il faudra les recharger. Si vous voulez éviter cela, utilisez « mount drive »
- Vous pouvez aussi récupérer l'exemple du cours:
  - Cours1\_IntroPython.ipynb

# Résultat Final

← → ↻ 🏠 colab.research.google.com/github/jch44/EPSI-ANN-B3/blob/main/Cours1\_IntroPython.ipynb#scrollTo=nqJN3GxfHbzi

 Cours1\_IntroPython.ipynb

Fichier Modifier Affichage Insérer Exécution Outils Aide Impossible d'enregistrer les modifications

Fichiers

bin  
boot  
content  
drive  
MyDrive  
sample\_data  
datalab  
dev  
etc  
home  
lib  
lib32  
lib64  
media  
mnt  
opt  
proc  
python-apt  
root  
run  
sbin  
srv  
sys  
tmp  
tools  
usr  
var  
NGC-DL-CONTAINER-LICENSE

Drive

+ Code + Texte Copier sur Drive

[1] from google.colab import drive  
drive.mount('/content/drive')

Mounted at /content/drive

Exemples de fonctions Pandas utiles pour le TP 1

Importations des librairies utiles

```
[ ] import numpy as np
import pandas as pd

print('numpy %s, pandas %s'%(np.__version__,pd.__version__))
```

Creation depuis un CSV et affichage d'un Dataframe Pandas

```
[ ] my_dataframe = pd.read_csv('Data/iris.csv', dtype={'id_Iris': np.int})
my_dataframe
```

Quelques Affichages du dataframe

```
[ ] my_dataframe.sample(5)

[ ] my_dataframe.info()

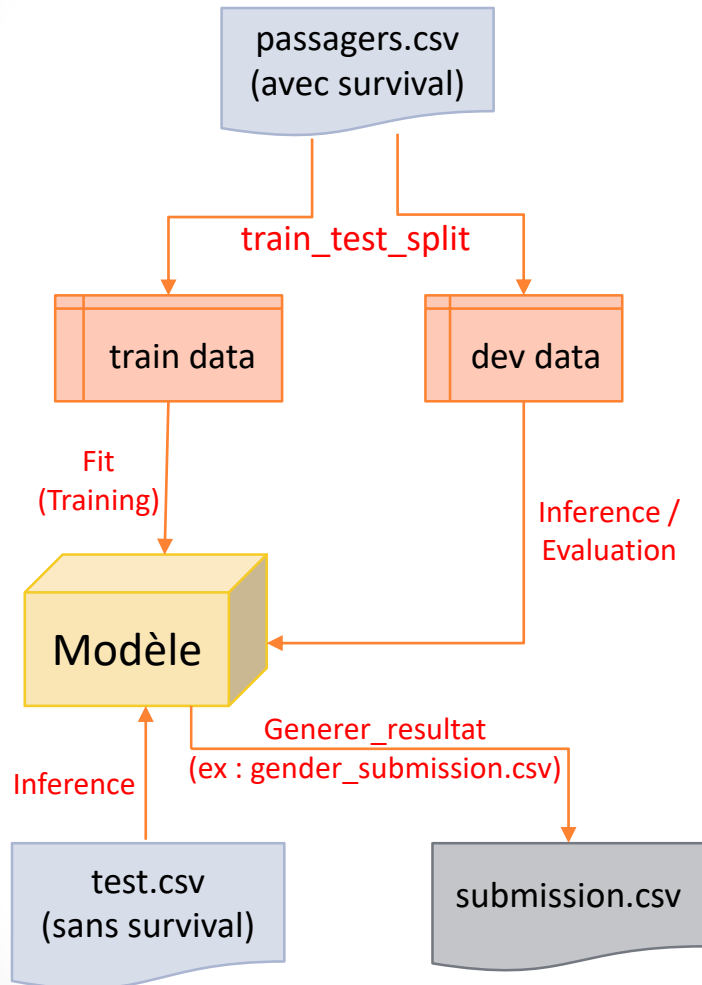
[ ] my_dataframe.mean()

[ ] my_dataframe.dtypes
```

# Projet Titanic sur Kaggle

- **Kaggle** (<https://www.kaggle.com/>) est une plateforme web de compétitions en science des données. Propose aussi des cours, des exemples et des datasets. Racheté par Google en 2017.
- Connexion avec votre compte Google.
- Aller dans Competitions puis chercher « **Titanic - Machine Learning from Disaster** »
- Joindre la compétition.
- Vous pourrez alors envoyer vos prédictions, et connaître votre classement. A partir de **78%** de bonne réponses, vous êtes bien.

# Comprendre les 3 datasets



- On utilise les données `train_clean.csv` qui contient la réponse pour entraîner le modèle et mesurer sa performance.
- Quand le modèle est prêt, on génère le fichier `submission.csv` depuis le fichier `test_clean.csv`.

→ kaggle

# Intro Python

Notebook Cours1\_IntroPython.ipynb

# Collecte de données



# Features Engineering

- Le Features Engineering (« Ingénierie des caractéristiques ») consiste à **travailler sur ces données** avant de créer un modèle.
- On estime le temps passé à 40% du temps total d'un projet
- Il s'agit de :
  - **Comprendre** ses données.
  - Identifier les **Features les plus importantes**, pour en éliminer.
  - Créer des **nouvelles Features** qui vont simplifier l'apprentissage.
  - Enfin, on peut agir sur données :
    - Les **corriger** en supprimant les données aberrantes, ou les normalisant.
    - Les **compléter** en attribuant des valeurs aux données manquantes.
    - Les **augmenter** en inventant de nouveaux exemples à partir des existants.



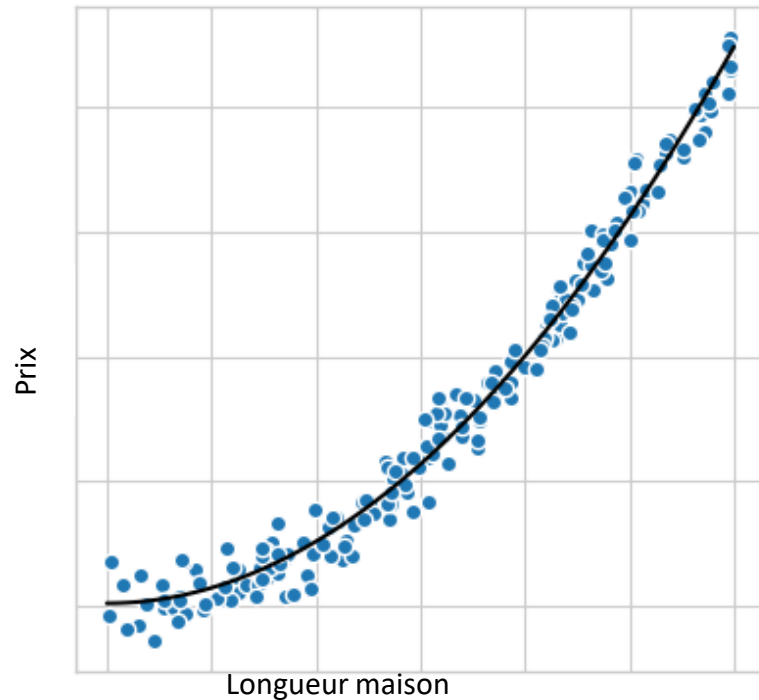
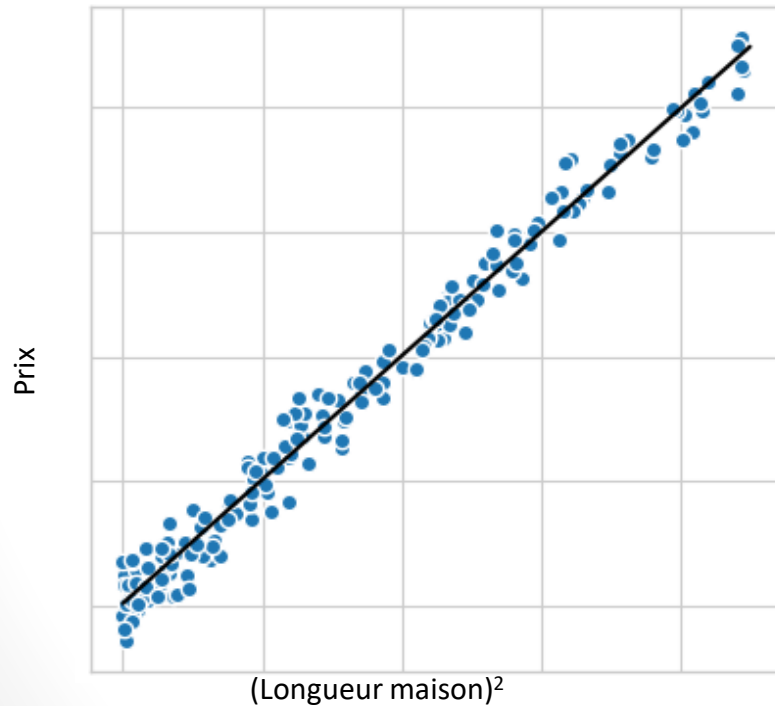
# Features Engineering



Exemple simple :

Le variable « Longueur Maison » mise au carré permet d'avoir une relation linéaire, plus simple à modéliser.

Estimation Prix maison



# Comprendre ses données

# Exploration des données



- Il est important de comprendre et connaître ses données.
- Afficher des échantillons.
- Identifier si des données sont manquantes.
- Identifier les types des données : Les modèles n'acceptent que des données de type numérique.
- Analyser les données statistiques sur les colonnes :
  - Moyenne (mean)
  - Écart type (standard deviation)
  - Min
  - Max
  - Quartiles ou déciles



# Prioriser les Features



# Importance des Features

- On va chercher l'impact de chaque feature sur le label.
- Pour les features valeurs numériques, nous pouvons utiliser un modèle RandomForest et regarder la propriété « Feature Importance »

[Cours2\\_ex3\\_ForestModel.ipynb](#)

```
Longueur Sépale (cm): 11%  
Largeur Sépale (cm): 2%  
Longueur Pétale (cm): 44%  
Largeur Pétale (cm): 42%
```

- Pour un Feature de type catégorie, Nous pouvons regarder la distribution du label selon les valeurs du Features.
- La technique d'information mutuelle suit le même principe (évaluer l'impact d'un feature sur le label) mais est plus élaborée.
- ⚠ ces techniques permettent d'évaluer chaque feature, mais pas des combinaisons de features.

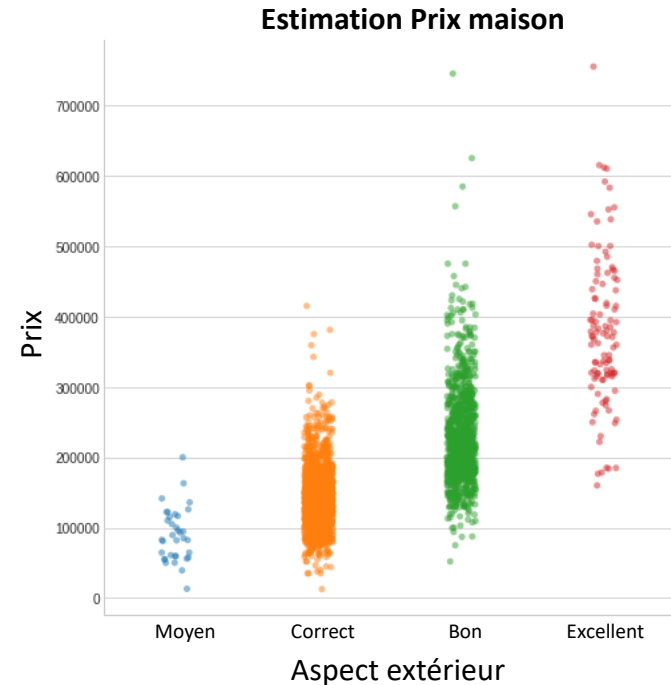


# Information Mutuelle

- Comment la connaissance de la feature réduit l'incertitude sur le label ?

$$MI(\text{feature}) = \text{Entropy}(\text{label}) - \text{Entropy}(\text{label} | \text{feature})$$

- Assez proche de l'algorithme CART pour les arbres de décision
- Pas d'information mutuelle  
=>  $MI = 0$ .
- Plus MI est grand, plus le feature est important. 2 est une grande valeur.
- Contrairement à la corrélation, l'information mutuelle peut mesurer tout type d'information.



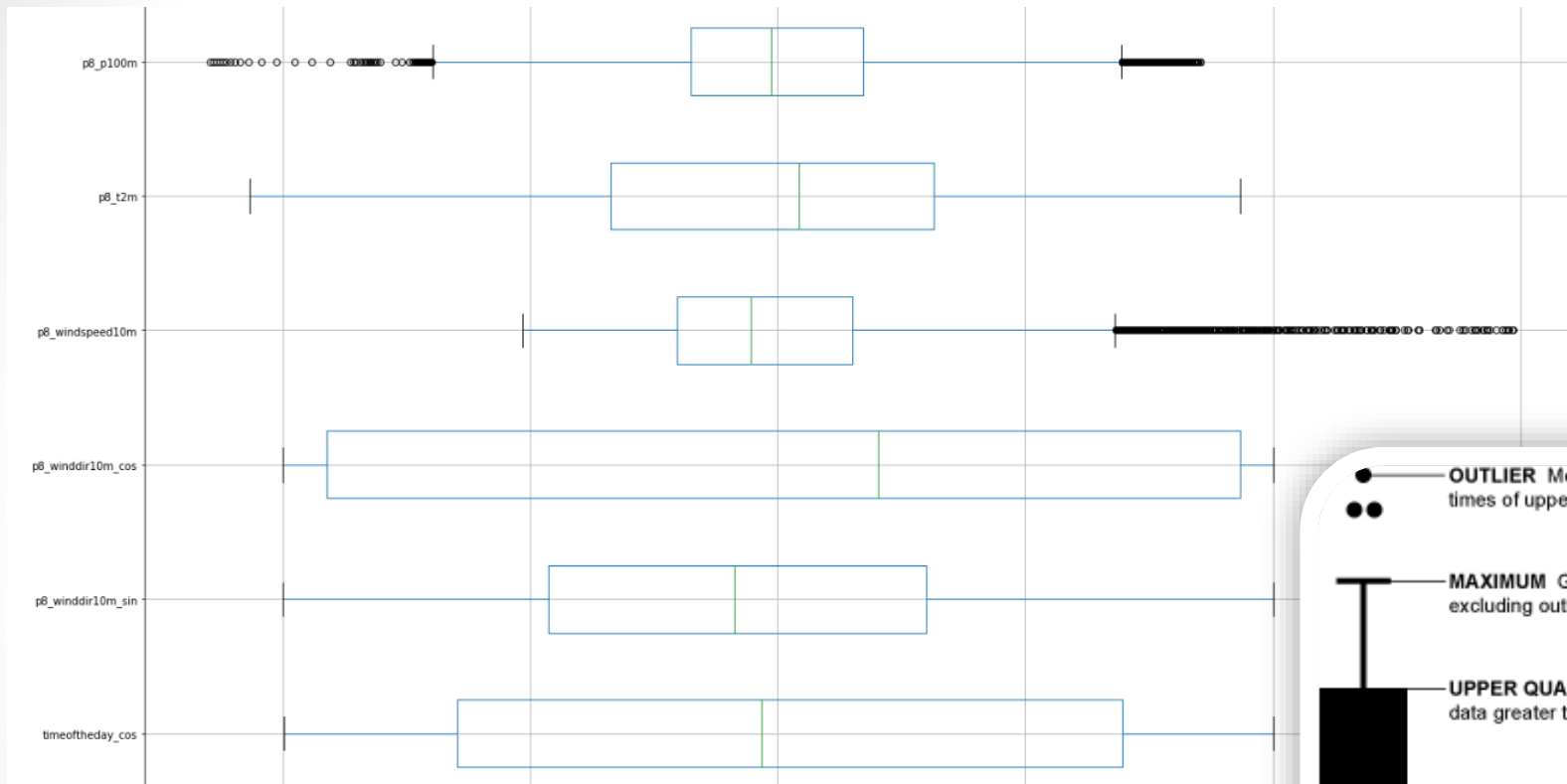
La connaissance de l'Aspect extérieur (feature), permet de mieux estimer le prix de la maison (label).

Fonctionne aussi sur des features numériques.

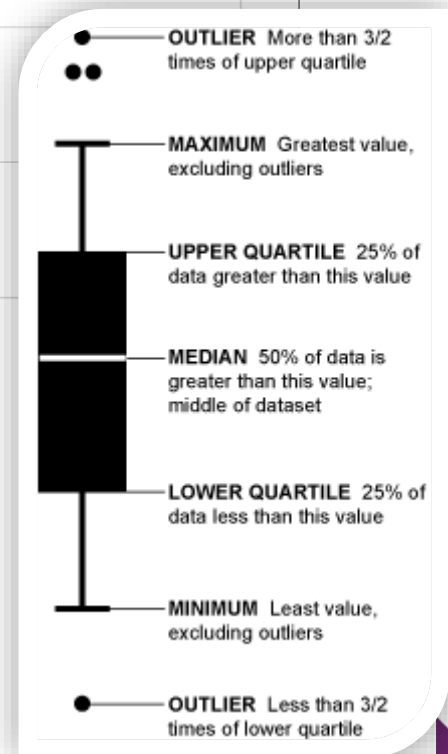
=> Exemples

# Corriger ses données

# Filtrage



- Suppression des outliers
- Dépend de la nature des données...
- Méthode `plot.box()`
- Peut être inutile en Big Data (les erreurs sont noyées dans la masse).





# Besoin de Normalisation



- Les features peuvent avoir des ordres de grandeurs très différents.
- Exemple : Pour une maison, le nombre de chambre (1 à 5), la surface (100 à 3000 m<sup>2</sup>), le dernier prix de vente (100 000 à 1000000 €).
- Pour que l'apprentissage ne surpondère pas certaine variables, il est d'usage de normaliser les features en -1 et 1.
- Dans les problèmes de régression, Les labels sont normalisées également.



# Techniques de Normalisation



- La formule généralement utilisée pour la valeur normalisée est :

$$\bar{x} = 2 \frac{x - x_{moy}}{x_{max} - x_{min}}$$

- L'écart type peut aussi être utilisé à la place de  $x_{max} - x_{min}$



Ne pas oublier de normaliser aussi avant l'inférence, avec les valeurs statistiques du Training Set.





# Distribution des labels

- Pour que le training ne soit pas faussé les classes ou valeurs des labels doivent être uniformément distribués.
- Par exemple, distribution parfaite de MNIST entre 0 et 9.
- Dans la vraie vie c'est rarement le cas. On peut accepter un certain niveau d'hétérogénéité, mais les classes rares risquent d'être ignorées.
- Quelques exemples :
  - Fraudes.
  - Grains de beauté problématiques
  - Pannes
  - ...
- La classe rare est souvent celle qui nous intéresse.
- Hors le modèle pourrait apprendre que la meilleure fonction de coût est obtenue en « oubliant » simplement la classe rare.



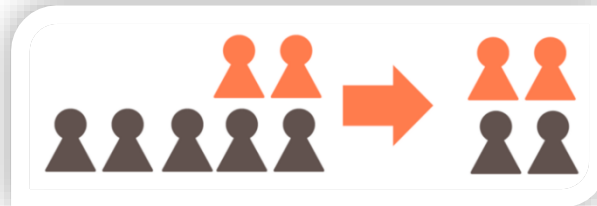
# Distribution des labels : solutions



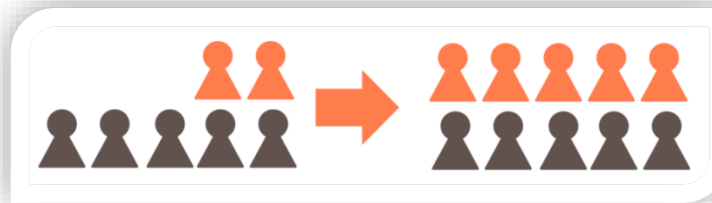
- Regrouper plusieurs classes rares



- Supprimer des exemples de la classe majoritaire



- Dupliquer des exemples de la classe rare



- Pour un problème de régression, on peut parfois prendre le log de la valeur.





# Traitement des données manquantes

- Pas de solution idéale.
- Essayer de comprendre pourquoi les données manquent. Corriger si possible.
- Deux stratégies : supprimer ou compléter.
- La suppression est possible à deux conditions:
  - La proportion du volume de données à supprimer est faible.
  - Les données manquantes sont distribuées aléatoirement (MAR, Missing At Random).
- Dans les autres cas, il est préférable de compléter les valeur manquantes (imputation) avec deux cas de figures:
  - Série Temporelle
  - Valeurs quelconques => Exemples

