

1. Histoire du Machine Learning et contexte du Big Data

- Replacer à leur échelle les concepts d'Intelligence Artificielle, apprentissage automatique (machine learning)...
- Le lien avec les mathématiques, les statistiques (inférentielles), le data mining et la data science
- Passer de l'analyse descriptive à l'analyse prédictive puis prescriptive
- Les applications du Machine Learning (moteurs de recherche, détection des spams, lecture des chèques)
- La typologie des algorithmes de Dominique CARDON
- La communauté Data Science et les challenges Kaggle (ex. de Netflix)

Machine Learning

algorithmes capables d'apprendre à partir de données

Apprentissage supervisé
régression, classification

Apprentissage non supervisé
clustering, réduction de dimensions

Apprentissage semi-supervisé

peu d'étiquettes, beaucoup de données non-étiquetées
utile quand coût élevé d'un étiquetage

Apprentissage par renforcement

environnement cist avec bonus - malus
pour apprentissage automatique par grande quantité de tests

Applications concrètes

Recommandation produit (e-commerce)

Détection de fraude (paiement, comportements)

Analyse de churn (résiliation d'abonnement)

Email spam (filtrage automatique)

Prédiction des ventes ou des stocks

Lecture automatique de chèques (OCR)

Deep Learning

Besoin de beaucoup de données & GPU

Réseaux de neurones

Perceptron, MLP
Multi-Layer Perceptron

CNN
pour images

RNN / LSTM
pour séquences
Long Short Term Memory

Transformers
pour texte (NLP)

Applications e-commerce

NLP pour FAQ automatique ou support client

Vision par ordinateur pour catégorisation produit

Recommandations personnalisées plus fines

Analyse des avis clients (sentiment analysis)

Communauté ML

Plateformes éducatives & MOOC

Massive Open Online Course

- Coursera (Andrew Ng – ML & DL)
- Fast.ai (formation DL très concrète)
- OpenClassrooms (intro + projets ML)

Datasets célèbres

- Netflix Prize (recommandation)
- MNIST, CIFAR-10 (images)
- Titanic, House Prices (Kaggle débutants)
- Amazon Review Data, Yelp, IMDB (NLP)

Partage de notebooks

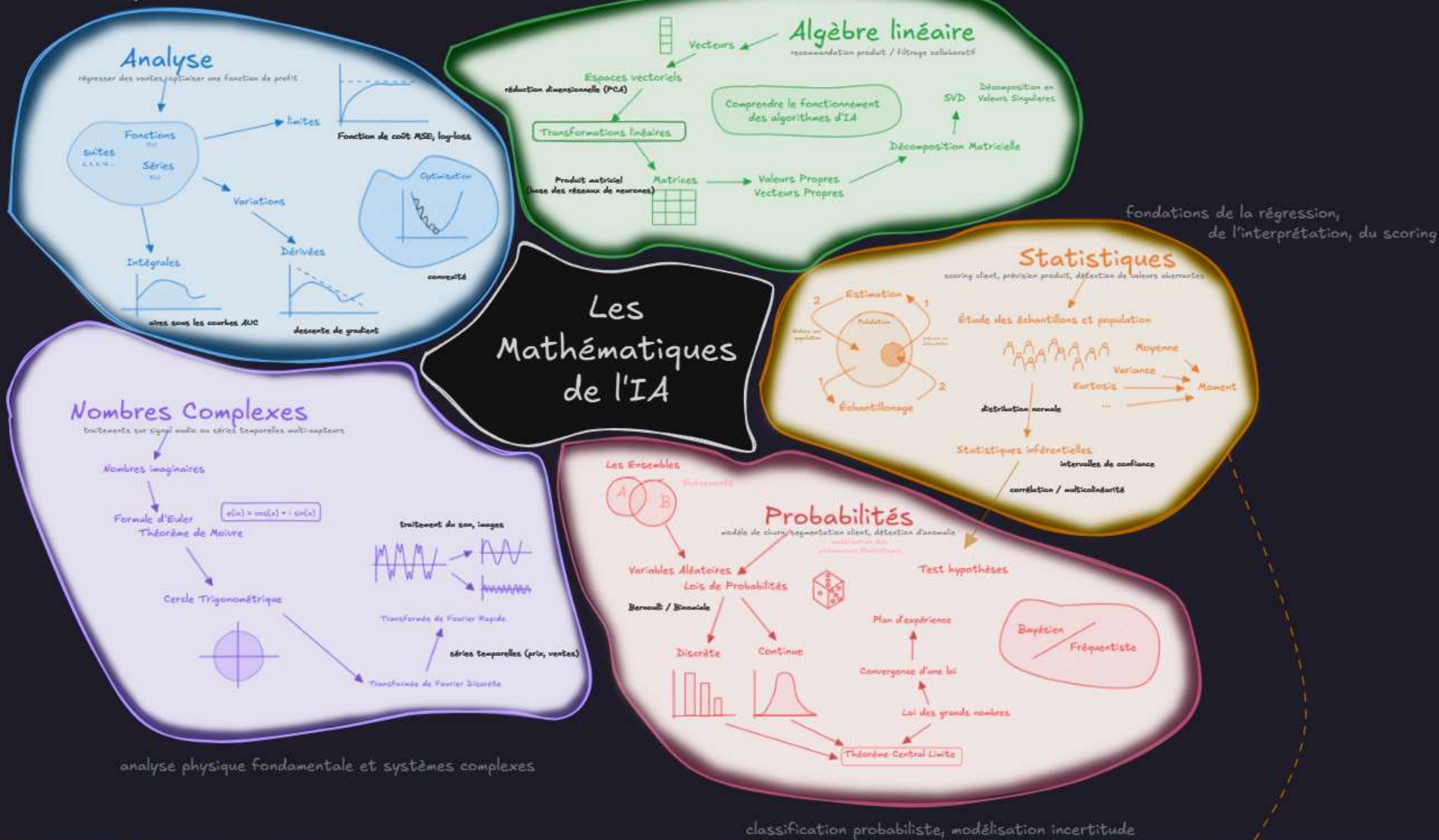
- Google Colab (très utilisé pour partager du code exécutif)
- Kaggle Kernels (équivalent notebook dans Kaggle)
- Papers With Code (liens entre articles scientifiques & code GitHub)

Compétitions

- Kaggle (le plus connu, racheté par Google)
- DrivenData (focus social & environnemental)
- Zindi Africa (compétitions pour le développement en Afrique)

comprendre les modèles de régression et les courbes de coût

base de toutes les opérations sur les matrices de données



analyse physique fondamentale et systèmes complexes


classification probabiliste, modélisation incertitude

Question logique par laquelle on admet une proposition
seulement si sa liaison avec d'autres propositions déjà tenues pour vraies.


✓ Types de statistiques utilisées en ML :

Type de statistique	But	Lien ML / Exemples
Descriptive	Résumer les données (moyenne, médiane, etc.)	Nettoyage de données, visualisations
Inférentielle	Tirer des conclusions sur une population à partir d'un échantillon	Régression, estimation des performances modèles
Exploratoire	Trouver des tendances cachées sans hypothèse	Clustering, PCA
Bayésienne	Mettre à jour les probabilités avec de l'info nouvelle	Naïve Bayes, probabilités postérieures

De l'analyse descriptive → prédictive → prescriptive

 Il s'agit des 3 niveaux d'analyse des données, qu'on peut représenter comme une évolution de la valeur :

Type d'analyse	Objectif	Exemples concrets
Descriptive	Que s'est-il passé ?	Moyenne des ventes, nb de visiteurs
Prédictive	Que va-t-il se passer ?	Prévision du churn, estimation de CA
Prescriptive	Que doit-on faire ?	Suggestion de promotions, actions ciblées

 **Prescriptif = le plus "actionnable"**. C'est ce que le ML permet d'atteindre :

- En combinant des modèles prédictifs avec des **règles métier** ou de l'**optimisation**
- Ex. dans e-commerce :
 - Prédiction du churn + **recommandation automatique** de réduction personnalisée
 - Prévion de stock + **recommandation de commande automatique**

Typologie des algorithmes selon Dominique Cardon

C'est une **classification sociologique** (et pas technique !) des algorithmes, proposée par **Dominique Cardon**, chercheur au Médialab de Sciences Po.

✳ Il distingue 4 grandes familles d'algorithmes selon la manière dont ils produisent du sens :

Famille	Type d'algorithme	Logique principale	Exemple
Statistique	Régressions, probabilités	Identifier des corrélations dans des données	ML supervisé
Connexionniste	Réseaux de neurones	Reproduire le fonctionnement du cerveau	Deep Learning
Symbolique	Systèmes experts, logique	Manipuler des règles explicites	IA des années 80
Évolutionniste	Génétique, renforcement	S'adapter à l'environnement par essai/erreur	RL, GA

💡 C'est utile pour **prendre du recul** et comprendre les **idéologies derrière les modèles** :












- Ce n'est pas juste du code, c'est **une vision du monde** (probabiliste, adaptative, logique, etc.)

2. Les données à disposition : collecte et préparation

- Données structurées, semi-structurées et non structurées
- Nature statistique des données (qualitatives ou quantitatives)
- Objets connectés (IoT) et streaming
- Opportunités et limites de l'Open Data
- Identification des corrélations, problème de la multicolinéarité
- Réduction des dimensions par Analyse des Composantes Principales
- Détection et correction des valeurs aberrantes
- Les ETL (Extract Transform Load)
- Le Web scraping

[illegible]

🌟 **Données structurées, semi-structurées, non structurées :**

-  Structurées : tableau SQL (produits, commandes, users, etc.)
 -  : facile à interroger
 -  : rigide
 -  Exemple : CSV, SQL, Excel
-  Semi-structurées : données avec balisage
 -  : souple, plus humaine
 -  : nécessite parsing
 -  Exemple : JSON, XML, YAML
-  Non structurées : texte libre, images, vidéos, son
 -  : source très riche (avis clients, visuels, chat)
 -  : nécessite NLP, vision ou audio
 -  Exemple : PDF, JPG, MP3, mails

```
"willet": "SILVERCREST-UK",  
"significant_holders_quantity": 1951161703682.176,  
"significant_holders_perc": 40.1953754783,  
"ticker": "VPERE",  
"contract_address": "0x000210B1c5543dc9c32d6be47a2d5af3d213333",  
"date": ("2023-01-04 16:10:30"),  
"network": "ERC-20",  
"unique_holders": 409015,  
"circulating_supply": 41374320787954.8023,  
"total_supply": 43868600953543.5625,  
"ecart_abs(tot_supply - circulating)": 9948178865248.4765625,  
"(tot_supply - circulating)/tot_supply": 5.61347484,  
"decimal": 18,  
"explored_pages": 41
```

[illegible]

(V.A. = Domaine + Loi)

V.A. Discrètes et Continues

Je lance un dé, et je regarde le résultat

Je tire une boule au hasard

Je reçois un email: est-il un spam ou pas

La prochaine personne qui toque à ma porte

Je regarde par la fenêtre la couleur de la première voiture qui passe

Nombre fini de résultats
(ou infini dénombrable)

Je verse de l'eau dans mon verre

La taille d'un individu

Le poids d'un individu

Le temps d'attente au téléphone

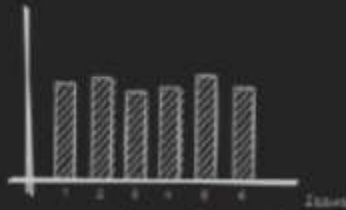
La durée de vie d'une étoile

Nombre infini indénombrable de résultats

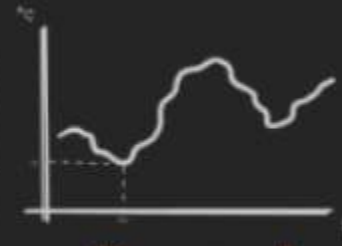
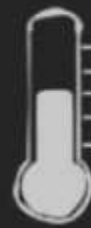
Il existe 2 grands types de Variables Aléatoires:

classe ...
Catégorie

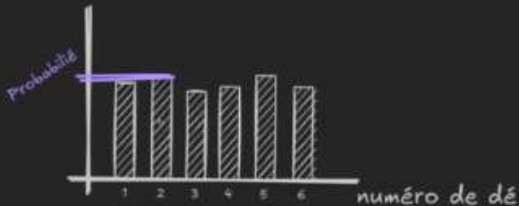
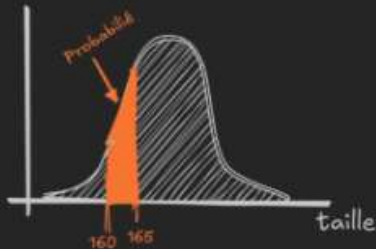
- Celles dont le domaine contient un nombre fini ou infini dénombrable de résultats
 - _____ infini indénombrable de résultats
- quantité indénombrable



"Se compte / se dénombre"



"Se mesure"

	V.A. Discrète	V.A. Continue
Domaine	fini ou infini dénombrable liste de valeurs [1, 2, 3, 4]	infini indénombrable intervalle de valeurs [0, 100]
Loi de Probabilité	<div>Fonction de Masse</div> 	<div>Fonction de Densité</div> 

Lois :
Bernouilli
Binomiale
Poisson
Hypergéométrique

Lois :
Normale
Gamma
Khi2
Student

1. Analyse Univari e

Variable Discr te

Variable Continue

Statistiques

Calcul des effectifs
 n_i

```
df['x1'].value_counts()
```

```
df['x1'].value_counts.sort_index()
```

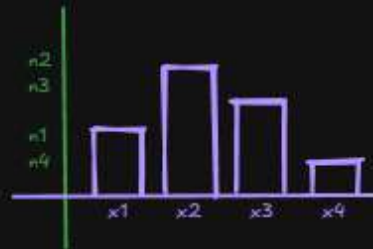
Moyenne, m diane, variance
 cart-type, min, max, Q1, Q3

```
df['x1'].describe()
```

```
.mean(), .std(), .var()  
.min(), .max(), .quantile(0.25)
```

Graphiques

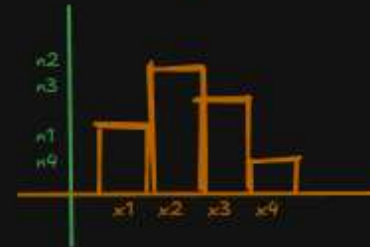
Bar Chart



```
df['x1'].value_counts().plot(kind='bar')
```

```
sns.countplot(data=df, x="x1")
```

Histogramme



```
sns.histplot(data=df, x="x1", ax=ax[0])
```

Boxplot



```
sns.boxplot(data=df, x="x1", ax=ax[1])
```


2. Analyse Multivariée

Variable Discrète

Variable Continue

Variable Discrète

Table de Contingence

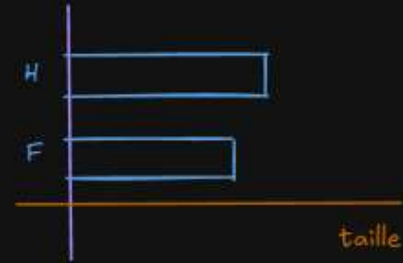
	brun	blond	noir
H	35	1	15
F	19	9	14

```
pd.crosstab(df['x1'], df['x2'], normalize)
```

Relations

```
sns.heatmap(pd.crosstab(...), annot)
```

Regroupement



```
df.groupby(v_disc)[v_cont].mean()
```

Comparaison

Variable Continue

Nuage de points

```
sns.scatterplot(data=df, x='x1', y='x2', hue)
```



Correlation

```
df.corr(method='pearson', numeric_only=True)
```

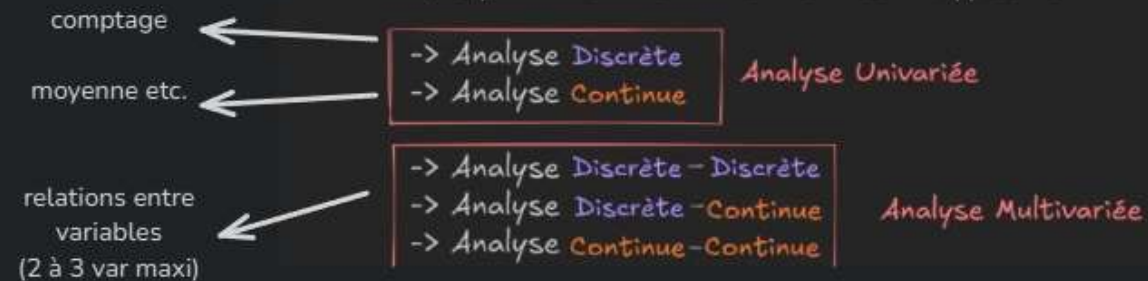
5 Analyses 🤖

Le B.A.-BA de l'analyse de Données

(Les 5 seules choses à analyser)



Que pouvons-nous faire avec ces 2 types de variables ?



- 1. Découverte python
 - exploration d'un dataset
 - TD :
 - * trouver les variables discrètes et continues
 - * graphes uni et multi variés

Analyse Exploratoire du Dataset des Maisons

Dans ce notebook, nous allons analyser un dataset contenant des informations sur des maisons, explorer les relations entre les variables, et préparer les données pour la construction d'un modèle de régression.

1. Chargement des Librairies et du Dataset

Nous allons d'abord charger les librairies nécessaires et importer le dataset.

In [1]:

```
# Lien de lancement sur Google Colab
# https://colab.research.google.com/github/Fred-Zang/AJC-ML_etat_art/blob/main/10-Regressions_v3.ipynb

# Importer les librairies nécessaires
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Lancement via Google Colab pour charger le dataset
url = "https://raw.githubusercontent.com/Fred-Zang/AJC-ML_etat_art/main/kc_house_data_modified.csv"
house_data = pd.read_csv(url)

# Charger le dataset en local
#house_data = pd.read_csv('data/kc_house_data_modified.csv')

# Afficher les premières lignes du dataset
house_data.head()
```

Out[1]:

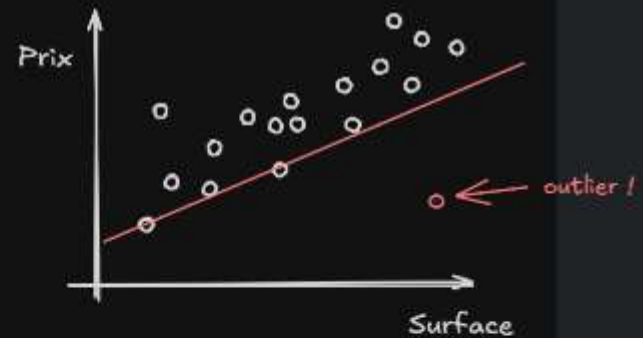
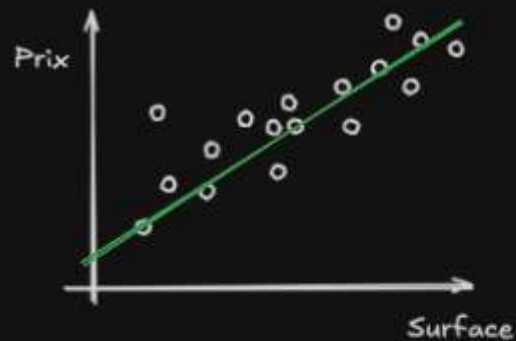
	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above
0	7129300520	20141013T000000	221900.0	3	1.00	1180.0	5650	1.0	0	0.0	...	7	1180
1	6414100192	20141209T000000	538000.0	3	2.25	2570.0	7242	2.0	0	0.0	...	7	2170
2	5631500400	20150225T000000	180000.0	2	1.00	770.0	10000	1.0	0	0.0	...	6	770
3	2487200875	20141209T000000	604000.0	4	3.00	1960.0	5000	1.0	0	0.0	...	7	1050

Outliers

Nettoyage des Valeurs Aberrantes

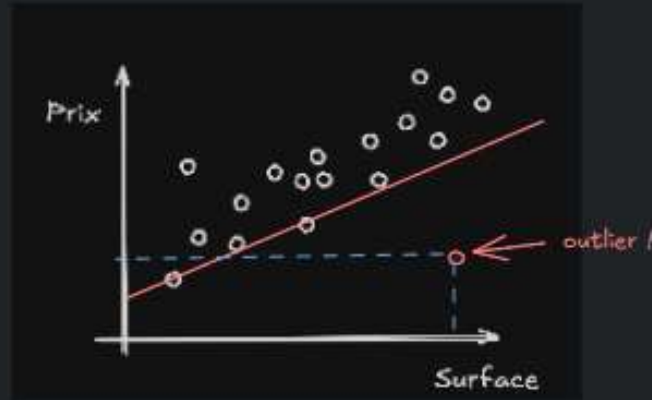
Avoir des valeurs aberrantes (outliers) dans son Train set peut grandement biaiser notre modèle de Machine learning !

sans outliers



Avant de décider si on élimine ou pas nos outliers, il faut se questionner coorectement !

est-ce normal d'avoir une grande surface pour un si petit prix ?
c'est peut-être un hangars ?



est-ce que cela vaut la peine de modéliser ce genre de valeurs ?

3 méthodes de nettoyage :

- Analyse Fondamentale c'est cette analyse qui nous permet de décider si on garde ou si on élimine certains ou tous les outliers
- Nettoyage Univarié (méthode IQR, Z-score, ...) on analyse séparément chaque variable pour détecter puis traiter les outliers
- Nettoyage Multivarié (Isolation Forest, LoF, ...) on analyse plusieurs variables (2 ou +) pour détecter puis traiter les outliers

Garbage In -> Garbage Out

Problème de la Multicolinéarité

C'est le cas où deux (ou plusieurs) variables explicatives sont fortement corrélées entre elles.

🧠 Pourquoi c'est un problème en Machine Learning ?

- 📊 Dans une **régression linéaire**, cela fausse les estimations des coefficients.
- 📊 Cela rend les modèles moins **interprétables**.
- 📊 Cela peut provoquer de l'**instabilité numérique** (matrices mal conditionnées).
- 🚫 Il devient **difficile de savoir** quelle variable influence vraiment la cible.

🌐 Exemple e-commerce (cas Neteven) :

- `prix_produit` et `prix_promo` → souvent très corrélés
- `nb_pages_vue` et `temps_passé` sur la fiche produit → idem → Dans un modèle, garder les deux sans vérification peut **nuire à la robustesse**.

Comment la détecter ?

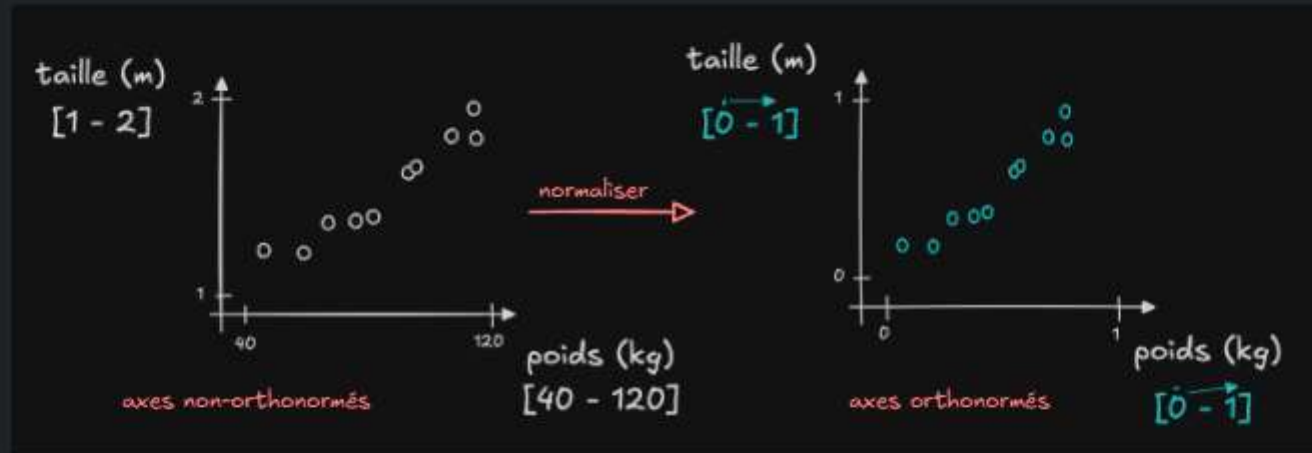
Méthode	Explication
Matrice de corrélation	Détecte les variables corrélées pair à pair
Variance Inflation Factor (VIF)	Mesure la redondance d'une variable par rapport aux autres
Régression croisée	Regarder si une variable peut être prédite par les autres

Solutions possibles

- Supprimer une des deux variables corrélées
- Fusionner (ex : créer un ratio)
- Réduire la dimension (PCA)

Normalisation

En machine learning, il est important de normaliser chaque variable, c'est-à-dire les mettre toutes sur une même échelle.



Pourquoi est-ce important ?

Pour beaucoup d'algorithmes de ML, lorsqu'une variable le est plus "grande" qu'un autre, alors celle-ci a un impact plus important sur la sortie du modèle (et donc sur ses erreurs, sa performance etc.)

par exemple un modèle $f(x_1, x_2) = w_1x_1 + w_2x_2$

NaN

Traitement des Valeurs Manquantes

NaN = "Not a Number" = champs "null" dans notre dataset

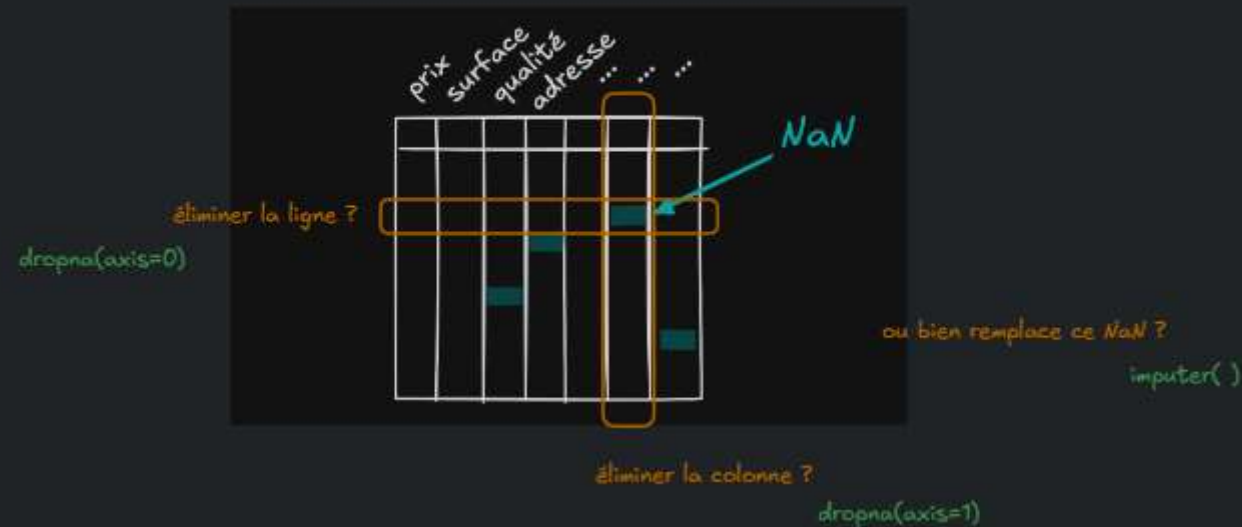
- Les NaNs sont fréquentes en Data Science
- Elles empêchent le développement d'un modèle de ML
- il faut donc les traiter

2 options :

- Les éliminer
- Les remplacer

2 questions à se poser :

- Combien y-a-t'il de NaN ?
- Comment sont-elles distribuées ?



Encoding

Transformer des valeurs non-numériques en valeurs numériques

- indispensable avant de faire du ML
 - > on ne peut pas faire des maths sur des mots
 - > mais attention au sens des catégories

```
df = sns.load_dataset('diamonds')
df.sample(5)
```

✓ 0.2s

	carat	cut	color	clarity	depth	table	price	x	y	z
34980	0.34	Ideal	F	VS2	62.1	56.0	880	4.47	4.45	2.77
6875	1.01	Good	J	VS1	60.4	64.0	4129	6.48	6.44	3.90
2743	0.65	Ideal	G	VVS1	61.9	56.0	3248	5.56	5.59	3.45
23702	0.40	Premium	J	VS1	59.7	59.0	631	4.79	4.83	2.87
16288	1.01	Premium	E	VS2	58.1	60.0	6516	6.58	6.51	3.80

2 Grandes Méthodes

- > encodage ordinal
- > encodage onehote

Astuce pour voir le noms de colonnes construites

```
encoder = OneHotEncoder(sparse_output=False, drop="first")
encoder.fit(df[["sex", "smoker", "day", "time"]])

# Récupérer les données transformées qui sont ndarray
encoded_array = encoder.transform(df[["sex", "smoker", "day", "time"]])
# Récupérer les noms des colonnes générées
encoded_column_names = encoder.get_feature_names_out(["sex", "smoker", "day", "time"])
# Créer un DataFrame avec les colonnes encodées
encoded_df = pd.DataFrame(encoded_array, columns=encoded_column_names)

# afficher
encoded_df.head()
```

✓ 0.0s: Open 'encoded_df' in Data Wrangler

	sex_Male	smoker_Yes	day_Sat	day_Sun	day_Thur	time_lunch
0	0.0	0.0	0.0	1.0	0.0	0.0
1	1.0	0.0	0.0	1.0	0.0	0.0
2	1.0	0.0	0.0	1.0	0.0	0.0
3	1.0	0.0	0.0	1.0	0.0	0.0
4	0.0	0.0	0.0	1.0	0.0	0.0

2. normalisation

- TD :
 - * trouver les variables à normaliser
 - * réaliser la normalisation

G. Normalisation

Critères pour la normalisation :

- Régression linéaire : Ce modèle est sensible à l'échelle des données. Il est donc souvent recommandé de normaliser ou standardiser les colonnes si elles ont des plages de valeurs très différentes.
- Régression par forêt aléatoire : Ce modèle est moins sensible aux différences d'échelle, mais il peut tout de même être utile de normaliser certaines colonnes pour garantir une meilleure convergence et des performances plus constantes.

In []:

```
# refaire un describe() final après notre nettoyage pour décider les features à normaliser ou non  
house_data.describe()
```

Colonnes à normaliser :

Réduction de la complexité des données

simplifier sans perdre l'essentiel

Réduction de dimension

Décomposition en valeurs singulières...

$$M = U S V^*$$

Principal Component Analysis



Variétés



UMAP

t-SNE

t-SNE

Visualization of Manifolds in High-Dimensional Data

1. Introduction à la réduction de dimension

La réduction de dimension simplifie les jeux de données en réduisant le nombre d'attributs. Elle s'appuie souvent sur des techniques mathématiques comme la décomposition en valeurs singulières (SVD). Cette méthode décompose une matrice en trois autres matrices, facilitant les calculs et les projections.

2. Analyse en composantes principales (PCA)

Le PCA est un algorithme clé pour réduire la dimensionnalité. Il projette les données sur des axes principaux afin de conserver l'essentiel des informations. Exemple : Compression d'images. Une image est projetée sur 50, 40, 30, 20, 10, ou même un seul axe. Jusqu'à 20 axes, la qualité reste bonne. En dessous de 10, l'information devient insuffisante. Le PCA réduit la taille et la complexité tout en préservant un niveau acceptable de qualité.

3. Modèles basés sur les variétés

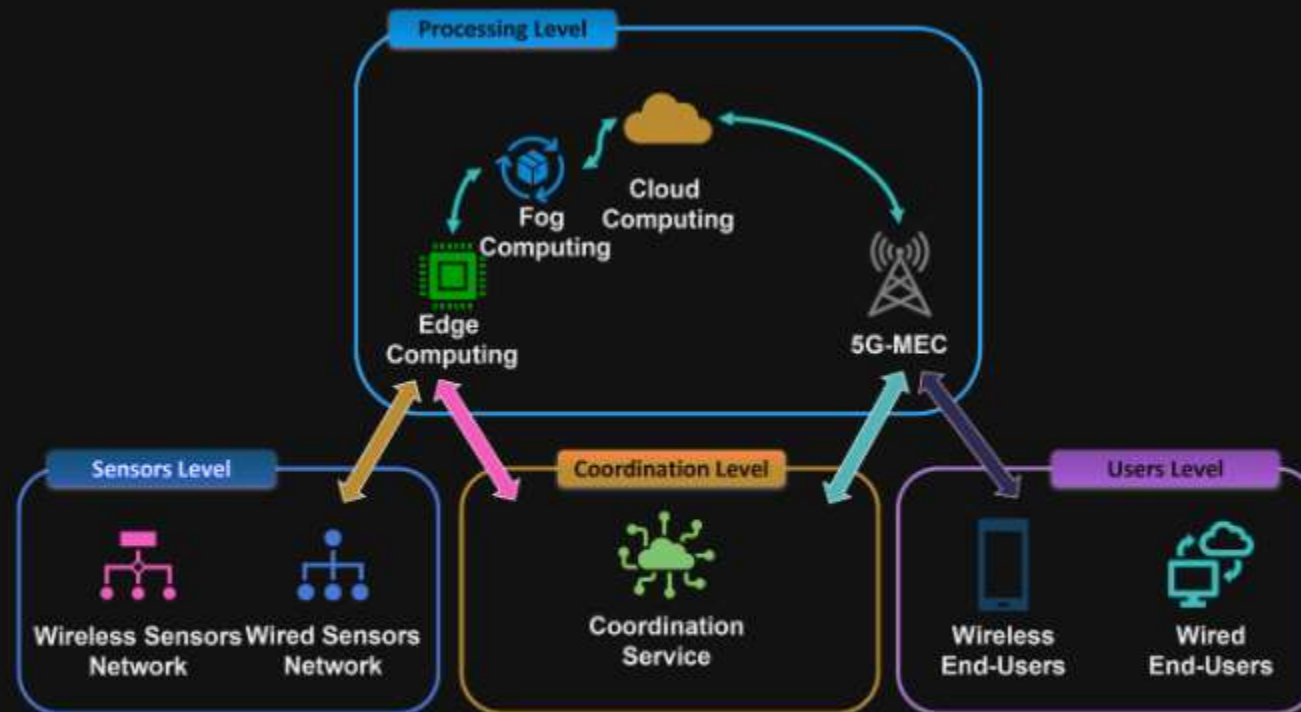
Les algorithmes de variétés apprennent des formes géométriques complexes appelées manifolds. Leur objectif est de simplifier ces formes tout en préservant leurs propriétés. Exemple : Modélisation d'un "S". Une forme complexe (comme un S) peut être réduite en une structure plus simple. Une fois modélisée, les axes d'origine deviennent inutiles. Il suffit de replacer la forme réduite dans l'espace. Algorithmes courants : t-SNE, UMAP, Isomap.

Acquisition et intégration de données

Collecte des données : sources internes, externes et en temps réel

✓ Objets connectés & streaming

- + Exemple : capteur de température, GPS colis, logs utilisateur
- + Notions : données continues, en temps réel, volume élevé
- 📶 Lien ML : prédiction de panne, recommandation en temps réel



✓ Open Data

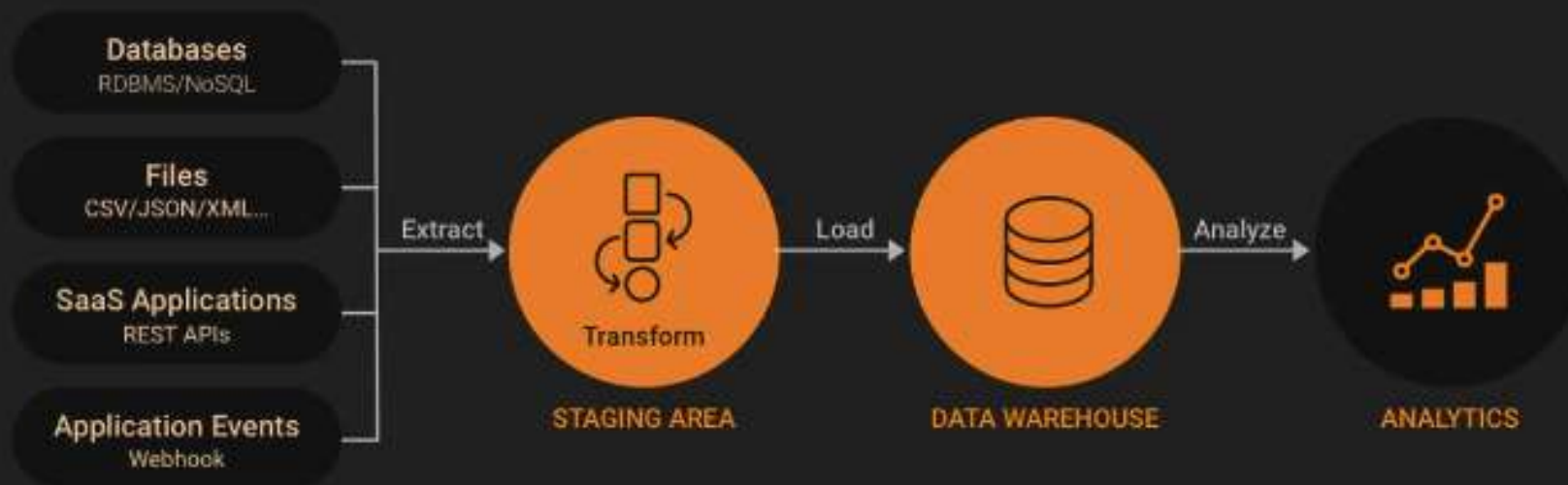
- + Avantages : gratuit, accessible, base pour POC
- — Limites : qualité variable, MAJ rares, biais potentiels
- 🔗 Exemple : data.gouv.fr, Kaggle datasets, World Bank

Opportunités vs Limites

● Opportunités	● Limites
Gratuit, accessible à tous	Qualité parfois hétérogène
Parfait pour expérimenter, POC	Données souvent incomplètes
Base de tests ou de benchmark	Mises à jour peu fréquentes
Source variée : géo, socio, éco...	Formats hétérogènes, mal documentés
Peut enrichir ses propres données internes	Risque de biais (collecte, échantillons)
Réutilisable dans des projets pédagogiques ou publics	Problèmes potentiels de cohérence ou fiabilité

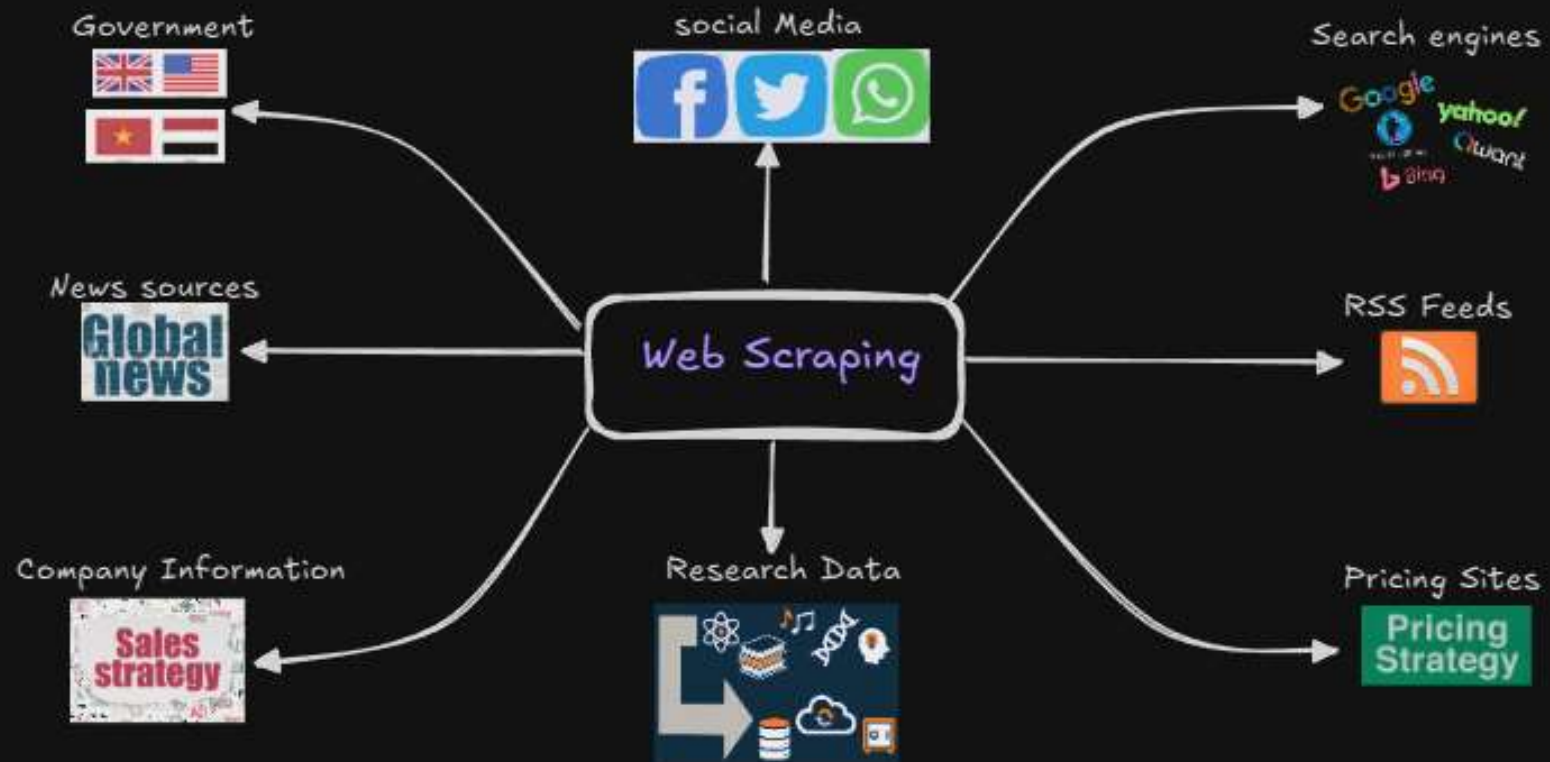
✓ ETL (Extract – Transform – Load)

ETL PROCESS



✓ Web scraping

- 🔍 Définition simple : récupération de données sur des pages web
- ⚠️ Mention légale rapide (robots.txt, respect RGPD)
- 🛠️ Outils : Python (BeautifulSoup, Selenium), Apify, Octoparse



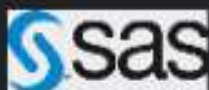
3. Les outils du marché pour le traitement de la donnée et le ML

- Les logiciels traditionnels (SAS, SPSS, Stata...) et leur ouverture à l'Open Source
- Choisir entre les deux leaders Open Source : Python et R
- Plateformes Cloud (Azure, AWS, Google Cloud Platform) et solutions SaaS (IBM Watson, Dataiku)
- Nouveaux postes en entreprises : data engineer, data scientist, data analyst, etc.
- Associer les bonnes compétences à ces différents outils
- Les API en ligne (IBM Watson, Microsoft Cortana Intelligence...)
- Les chatbots (agents conversationnels)


Les Outils du marché pour la donnée & le Machine Learning

Les logiciels traditionnels et Open Source

Ces logiciels sont utilisés depuis longtemps en entreprise, en particulier dans la recherche, la finance, le marketing et les grandes institutions publiques :



Outil	Acronyme	Description
SAS	<i>Statistical Analysis System</i>	Logiciel propriétaire très utilisé dans les grandes entreprises pour la statistique avancée , la gestion de données , et la production automatisée de rapports . Solide mais coûteux.
SPSS	<i>Statistical Package for the Social Sciences</i>	Développé à l'origine pour les sciences sociales. Très visuel, orienté analyse statistique classique . Utilisé en santé, psychologie, éducation.
Stata	<i>(Pas un acronyme officiel)</i>	Outil très populaire en économie et sciences sociales pour l' analyse de données et la régression statistique . Interface simple + langage script.

 **Tendance actuelle** : ces outils s'ouvrent aux **langages Open Source** comme **Python** et **R**, avec intégration directe dans leurs interfaces (SAS Viya, SPSS Modeler...).

Les leaders Open Source : Python et R

Langage	Description & usage
Python	Langage polyvalent très utilisé en Data Science , Machine Learning , IA et développement web . Très riche en bibliothèques : <code>pandas</code> , <code>scikit-learn</code> , <code>matplotlib</code> , <code>tensorflow</code> , etc. Facile d'accès pour débiter, très prisé en entreprise.
R	Langage statistique pur, excellent pour les visualisations , analyses statistiques complexes , et la modélisation bayésienne . Très utilisé en recherche, biostatistiques et par certains analystes.

Critère	Python	R	SAS / SPSS
Facilité d'apprentissage	✅ Oui	⚠ Statistique pure	❌ Menu mais coûteux
Communauté & support	✅ Enorme	✅ Très forte en stats	❌ Fermé / payant
Visualisation	✅ Bonne	✅ Excellente	❌ Moyenne
Machine Learning	✅ Top (sklearn, keras...)	✅ Oui (mais moins scalable)	❌ Possible mais rigide
Coût	✅ Gratuit	✅ Gratuit	❌ Payant
Évolutivité (cloud, API, etc.)	✅ Très bonne	❌ Moyenne	❌ Faible à moyenne

Nous utiliserons le langage Python plus pertinent et populaire dans le monde pro.

Reste à connaître vos moyens informatiques disponibles pour chacun d'entre vous, afin de déterminer un choix d'IDE, packages à installer ou par cloud avec Google Colab.

Plateformes Cloud, Solutions SaaS & APIs pour le Machine Learning



☁ 1. Plateformes Cloud généralistes

Les 3 leaders du cloud public proposent des solutions puissantes pour le ML, l'IA et la gestion de données :

Plateforme	Nom complet	Description & services liés au ML
AWS	<i>Amazon Web Services</i>	Propose SageMaker (service complet pour entraîner, déployer, monitorer des modèles ML). Aussi : Redshift, Lambda, S3, Athena...
Azure	<i>Microsoft Azure</i>	Propose Azure Machine Learning Studio (interface visuelle + SDK Python), Data Factory (ETL), Power BI...
GCP	<i>Google Cloud Platform</i>	Propose Vertex AI (successeur d'AI Platform), BigQuery, AutoML, Looker...

✍ Ces plateformes offrent des **outils cloud-native** pour tout le cycle de vie du ML : stockage, traitement, entraînement, déploiement.



2. Solutions SaaS spécialisées (Software as a Service)

Solution	Description
IBM Watson Studio	Plateforme SaaS complète pour le ML et l'IA. Inclut : entraînement de modèles, visualisation, NLP (Natural Language Processing), outils d'explicabilité, AutoML.
Dataiku	Plateforme française no-code/low-code pour la collaboration data , le prétraitement , et le déploiement de modèles ML . Très adaptée aux équipes mixtes (data + métier) .
H2O.ai	Plateforme open-source & cloud pour le machine learning automatisé (AutoML) , très performante en industrie.

Pour Neteven, Watson a de bons services IA prêt à l'emploi et Dataiku est idéal pour rapprocher tech & métier



3. APIs IA / ML prêtes à l'emploi

Fournisseur	Exemple d'API	Usage
IBM Watson	NLP, analyse de sentiments, visual recognition	Analyse des avis clients, chatbot
Microsoft Azure	Cortana Intelligence / Azure Cognitive Services	Vision, speech-to-text, traduction, détection d'objets, analyse de texte
Google Cloud	Vision API, Translation API, AutoML NLP	Extraction d'infos depuis du texte/image sans entraînement manuel

Ces API sont souvent **consommables via des requêtes web**, avec un **coût à l'usage** (pay-as-you-go).

Métiers de la donnée et leurs outils

Métier	Rôle principal	Compétences clés
Data Analyst	Analyse des données, création de dashboards, support décisionnel	SQL, Excel, Power BI, Tableau, statistiques, Python (pandas)
Data Scientist	Conçoit, entraîne et évalue des modèles ML, interprète les résultats	Python (pandas, sklearn, XGBoost), stats, ML, visualisation, storytelling
Data Engineer	Construit et maintient les pipelines de données (ETL, bases, cloud)	SQL, Spark, cloud (AWS/GCP), Python, Airflow, Docker
ML Engineer	Industrialise et déploie les modèles en production	DevOps, CI/CD, APIs, TensorFlow, cloud, MLOps
AI Product Manager	Pilote un produit data/IA, fait le lien entre métier, tech, client	Connaissance métier + tech, gestion projet agile, UX, communication
Data Steward / Architect	Garant de la qualité, gouvernance, structuration des données	Modélisation, RGPD, gestion de catalogue, data warehouse

Les rôles ne sont pas figés : un analyste peut monter en data science

Un bon projet ML repose sur la collaboration de plusieurs profils

Le choix d'outil dépend du niveau de technique et de l'objectif métier

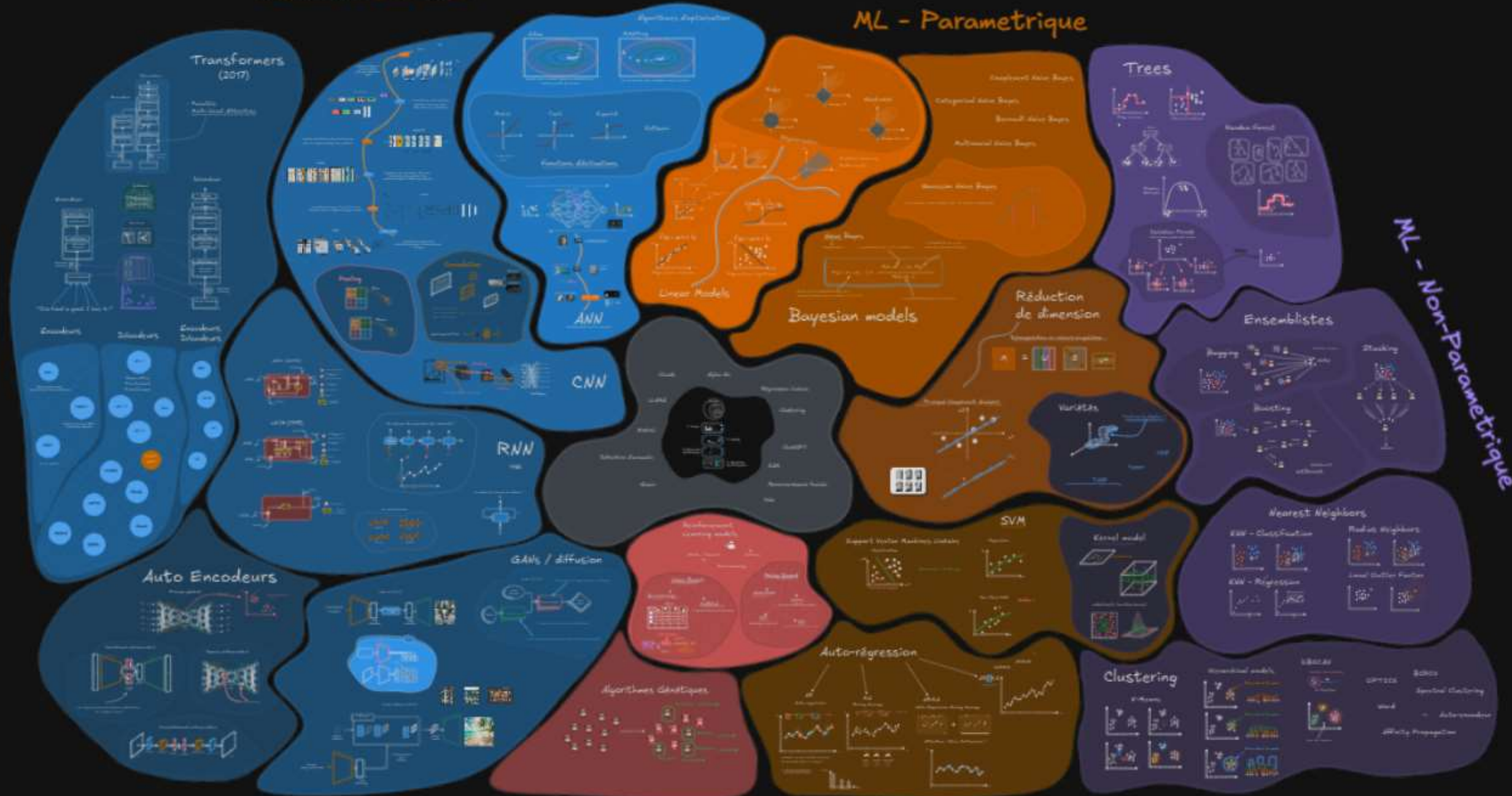
4. Tour d'horizon de tous les algorithmes ML et DL

- Apprentissage supervisé : répéter un exemple
- Apprentissage non supervisé : découvrir les données
- Online (Machine) Learning par opposition aux techniques batch
- Reinforcement learning : optimisation d'une récompense
- Autres types d'apprentissage (par transfert, séquentiel, actif...)
- Illustrations (moteurs de recommandation...)
- Régression linéaire simple et multiple. Limites des approches linéaires
- Régression polynomiale (LASSO) Séries temporelles
- Régression logistique et applications en scoring
- Classification hiérarchique et non hiérarchique (KMeans)
- Classification par arbres de décision ou approche Naïve Bayes
- Random Forest (développement des arbres de décision)
- Gradient Boosting. Réseaux de neurones. Machine à support de vecteurs
- Deep Learning : exemples et raisons du succès actuel
- Text Mining : analyse des corpus de données textuelles

Neural Networks

ML - Parametrique

ML - Non-Paramétrique



3. Choix d'un modèle de Régression

- TD :
 - * donner quelques modèles ML possibles à utiliser
 - * choix d'un modèle et réalisation

6. modèle de régression Gradient Boosting

- Un modèle de Gradient Boosting avec 100 arbres de décision (n_estimators=100) et une profondeur maximale de 3 (max_depth=3) a été utilisé. Ces paramètres peuvent être ajustés pour améliorer les performances.
- A. Diviser les données en ensembles d'entraînement et de test.

```
[15] # Importer les bibliothèques nécessaires
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import matplotlib.pyplot as plt
```

```
[16] house_data.describe()
```

	date	price	bedrooms	bathrooms	floors	condition	grade	yr_built	lat	long	total_s
count	16917	1.691700e+04	16917.000000	16917.000000	16917.000000	16917.000000	16917.000000	16917.000000	16917.000000	16917.000000	1.691700e
mean	2014-10-29 21:32:18.854406400	4.637523e+05	3.288172	2.012369	1.476355	3.407401	7.459538	1970.387835	47.560508	-122.227839	-9.744394e
min	2014-05-02 00:00:00	7.800000e+04	2.000000	0.500000	1.000000	1.000000	4.000000	1900.000000	47.155900	-122.503000	-2.231076e
25%	2014-07-22 00:00:00	3.086250e+05	3.000000	1.500000	1.000000	3.000000	7.000000	1950.000000	47.474100	-122.336000	-6.756471e
50%	2014-10-17 00:00:00	4.250000e+05	3.000000	2.000000	1.000000	3.000000	7.000000	1972.000000	47.570100	-122.266000	-4.087637e

5. Procédure d'entraînement et d'évaluation des algorithmes

- Séparation du jeu de données : entraînement, test et validation
- Techniques de bootstrap (bagging)
- Exemple de la validation croisée
- Définition d'une métrique de performance
- Descente de gradient stochastique (minimisation de la métrique)
- Courbes ROC et de lift pour évaluer et comparer les algorithmes
- Matrice de confusion : faux positifs et faux négatifs



4. Elaboration d'un programme d'entraînement et d'évaluation

- TD :
 - * quelles mesures d'évaluation pouvons-nous prendre pour notre modèle ?
 - * choix de 2 mesures et mise en pratique

```
# Séparer les features (X) et la cible (y)
X = house_data_selected.drop(columns=['price'])
y = house_data_selected['price']

# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Créer et entraîner le modèle de Gradient Boosting
gb_model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)
gb_model.fit(X_train, y_train)

# Évaluer la performance du modèle
y_pred_gb = gb_model.predict(X_test)

# Calculer le coefficient de détermination R² et le RMSE
r2_gb = r2_score(y_test, y_pred_gb)
rmse_gb = np.sqrt(mean_squared_error(y_test, y_pred_gb))

# Afficher les résultats
print(f"Coefficient de détermination R² (Gradient Boosting) : {r2_gb:.4f}")
print(f"Root Mean Squared Error (RMSE - Gradient Boosting) : {rmse_gb:.2f} USD")
```

```
⇒ Coefficient de détermination R² (Gradient Boosting) : 0.5734
Root Mean Squared Error (RMSE - Gradient Boosting) : 133346.41 USD
```

5. Faire des prédictions

- construire 3 exemples sans cohérents selon les datas existantes
- réaliser la prédiction des prix

```
# Faire des prédictions sur de nouveaux exemples
new_samples = pd.DataFrame({
    'bedrooms' : [5,6,8],
    'bathrooms': [2, 3, 4],
    'floors' : [1, 2, 2],
    'condition' : [2, 3, 4],
    'grade': [7, 9, 10],
    'yr_built': [1990, 2005, 2015],
    'total_sqft': [11800, 12500, 13500],
    'neighbor_avg_sqft': [5000, 8000, 12000]

})

# Normaliser les nouvelles données
new_samples[columns_to_normalize[:6]] = scaler.transform(new_samples[columns_to_normalize[:6]])

# Faire la prédiction avec le modèle Gradient Boosting
new_predictions = gb_model.predict(new_samples)

# Afficher les résultats
for i, pred in enumerate(new_predictions):
    print(f"Exemple {i+1} :")
    print(new_samples.iloc[i])
    print(f"Prix prédit : {pred:.2f} USD\n")
```

⇒ Exemple 1 :

bedrooms	5.000000
bathrooms	2.000000
floors	1.000000
condition	2.000000
grade	7.000000

6. Mise en production d'un algorithme de Machine Learning

- Description d'une plateforme Big Data
- Principe de fonctionnement des API
- Du développement à la mise en production
- Stratégie de maintenance corrective et évolutive
- Evaluation du coût de fonctionnement en production

Mise en production d'un algorithme de Machine Learning

le ML ne s'arrête pas à l'entraînement d'un modèle, et sa mise en production implique des choix techniques, métiers, et économiques.

1. Description d'une plateforme Big Data

Élément	Description
Définition	Une plateforme Big Data est un ensemble d'outils permettant de collecter, stocker, traiter et analyser de très grandes quantités de données.
Exemples	<ul style="list-style-type: none">- Hadoop (stockage distribué HDFS + calcul MapReduce)- Spark (traitement rapide en mémoire)- Databricks (cloud + notebooks collaboratifs)- Snowflake (entrepôt cloud moderne)- GCP BigQuery, Azure Synapse
Cas d'usage	<ul style="list-style-type: none">- Traitement des logs web- Prédiction de ventes à partir de millions de lignes- Analyse en temps réel d'événements (fraude, churn...)

2. Principe de fonctionnement des API

Rappel court (déjà vu dans le bloc précédent, à connecter ici) :

Une API permet d'exposer un modèle ML comme un service consultable à distance (via requêtes HTTP, REST, JSON).

💡 En pratique :

Un modèle entraîné avec `sklearn` ou `tensorflow` est souvent déployé via :

- FastAPI / Flask en local
- Docker pour le packager
- Kubernetes / Cloud Run / Lambda pour l'exécuter à l'échelle



Terme	Type	Sert à quoi ?	Exemple
HTTP	Protocole	Transporter des requêtes et réponses web	<code>GET /index.html</code> <code>HTTP/1.1</code>
REST	Architecture API	Organiser les URL/API via HTTP	<code>GET /users/5</code>
JSON	Format de données	Structurer les données échangées	<code>{"nom": "Alice"}</code>

Une requête HTTP est une demande envoyée par un client (navigateur, script, etc.) à un serveur web via le protocole HTTP.

Caractéristiques : Transmet des en-têtes (headers) et parfois un corps (body).

Peut transporter du HTML, XML, JSON, fichiers, etc.

REST est un style d'architecture d'API web qui repose sur les méthodes HTTP pour manipuler des ressources identifiées par des URL.

Caractéristiques : Utilise les méthodes HTTP (GET, POST, PUT, DELETE) pour représenter les actions.

Les ressources sont représentées via des URL lisibles : `GET /users/123` → Récupérer l'utilisateur avec l'id 123.

Chaque requête est indépendante et le format des données est souvent en JSON, mais pas obligé (peut être XML, HTML...).






REST - protocole, c'est un ensemble de principes de conception pour des API.


JSON est un format léger de données, lisible par l'homme et facilement analysable par les machines.

Caractéristiques : Utilisé dans le corps (body) des requêtes ou réponses HTTP, souvent dans des API REST.

Pour échanger des données structurées entre client et serveur.

3. Du développement à la mise en production

Étapes	Description
 Développement local	Entraînement du modèle sur Jupyter, test de performance
 Packaging	Encapsuler le modèle avec ses dépendances (Docker, MLflow, joblib...)
 Déploiement	Le rendre accessible via une API (REST ou gRPC)
 Monitoring	Suivre la performance (latence, erreurs, dérive du modèle)
 Amélioration continue	Re-entraînement avec nouvelles données, versioning

 on peut visualiser ça avec un pipeline de type CI/CD ML :
dev → test → prod → logs + monitoring

🔧 4. Maintenance corrective et évolutive

Type	Description
Corrective	Corriger un bug dans l'API ou un calcul erroné
Évolutive	Reprendre le modèle car les données ont changé (drift, concept shift)
Préventive	Ajouter des tests automatiques, alertes, logs

💡 Exemple : modèle de scoring qui surévalue les clients depuis 2 mois → drift → réentraînement avec données plus récentes.

💰 5. Évaluation du coût en production

Composante	Exemple de coût
Stockage	Données dans S3, Azure Blob, BigQuery
Calcul	GPU/CPU pour réentraînement, prédictions
Appels API	Pay-as-you-go (Watson, GCP AutoML, Hugging Face)
Surveillance	Coût d'un service MLOps (Weights & Biases, MLflow sur cloud)

💡 Le coût doit être mis en regard :

- des bénéfices business (meilleure prédiction, rétention client..)
- du **choix d'architecture** (on-premise vs cloud)

7. Aspects éthiques et juridiques liés à l'Intelligence Artificielle


- Missions de la CNIL et évolutions à venir
- Question du droit d'accès aux données personnelles
- Question de la propriété intellectuelle des algorithmes
- Nouveaux rôles dans l'entreprise : Chief Data Officer et Data Protection Officer
- Question de l'impartialité des algorithmes
- Attention au biais de confirmation
- Les secteurs et les métiers touchés par l'automatisation

Éthique & Législation autour de l'IA

Faire prendre conscience que l'IA n'est pas neutre, et que sa conception, utilisation et régulation impliquent des responsabilités éthiques et juridiques

1. Missions de la CNIL (Commission Nationale de l'Informatique et des Libertés)

Mission	Description
Encadrer l'usage des données personnelles	Veille à la conformité au RGPD, contrôle les traitements automatisés
Informier et sensibiliser	Publie des guides IA & éthique, bons usages des données
Sanctionner en cas d'abus	Peut infliger des amendes lourdes pour non-respect du RGPD

 La CNIL propose un **cadre de développement de l'IA respectueux des libertés**, basé sur :

- la transparence des algorithmes
- la minimisation des données
- le respect des droits des personnes

👁 2. Droit d'accès aux données personnelles

✳️ RGPD (Règlement Général sur la Protection des Données) :

- Toute personne peut :
 - accéder à ses données
 - demander rectification ou suppression
 - refuser un traitement automatisé

⚖ 3. Propriété intellectuelle des algorithmes

Problème	Réflexion
Modèle développé en interne	Appartient à l'entreprise
Modèle basé sur un modèle pré-entraîné (GPT, BERT, etc.)	Sujets de licence, partage et droits d'usage
Dataset utilisé pour l'entraînement	Peut être soumis à des droits d'auteur ou à des restrictions d'usage (→ Open Data ≠ open licence)

⚠ Attention aux usages non autorisés de jeux de données publics dans des contextes commerciaux.

🧑💼 4. Nouveaux rôles en entreprise

Rôle	Fonction
Chief Data Officer (CDO)	Définit la stratégie data globale de l'entreprise
Data Protection Officer (DPO)	Veille à la conformité RGPD, travaille avec la CNIL, conseille les équipes projet

💡 Ce sont des **acteurs transverses** qui interagissent avec data scientists, ingénieurs et décideurs métier.

⚖️ 5. Impartialité des algorithmes

★ Un algorithme peut reproduire (ou amplifier) des **biais** présents dans les données d'entraînement :

Exemple de biais	Risque
Genre, origine, géographie	Discrimination indirecte
Historique métier biaisé	Reproduction d'injustices passées
Biais de sélection	Modèle non généralisable à de nouveaux cas

💡 **Auditer, expliquer, réguler** sont essentiels dans tout projet IA.

⚠ 6. Biais de confirmation

Tendance à chercher ce qui valide nos hypothèses au lieu de tester objectivement.

🧠 En ML, cela se traduit par :

- des analyses orientées
- des modèles choisis pour confirmer un scénario
- un manque d'AB testing ou de contre-modèles

🏭 7. Métiers touchés par l'automatisation

Secteurs en mutation	Impact IA
Logistique	Prévision de demande, routage intelligent
RH / recrutement	Tri de CV automatisé, scoring de profils
Marketing	Segmentation, recommandations produits
Service client	Chatbots, analyse des avis
Industrie	Maintenance prédictive, vision industrielle
Finance	Détection de fraude, scoring crédit

⚠ Il ne s'agit pas toujours de **remplacer** : souvent, l'IA **augmente** ou **assiste** les tâches.