Data Pipeline

De la Donnée Brute au Modèle Performant

Plan du Cours

Introduction & Théorie des Données

Nature Fondamentale des Données

- Les modèles de Machine Learning sont essentiellement des **moteurs mathématiques**.
- Ils ne "savent" pas que :
  - *Rouge* est une couleur.
  - *20°C* est une température.
- **Réalité :** Ils ne voient que des nombres.

Pourquoi la Théorie des Données est Cruciale ?

Le Problème

Comprendre la théorie des données nous assure de choisir les bonnes transformations.

Objectif :

- Que les nombres que nous fournissons au modèle représentent fidèlement la réalité.

Les 4 Types de Variables

L'acronyme à retenir : **NOIR**

1. **N**ominal
2. **O**rdinal
3. **I**ntervalle
4. **R**atio

En descendant la liste, la quantité d'information augmente.

1. Nominal

- **Définition :** Catégories distinctes sans ordre ni classement. De simples étiquettes.
- **Opérations Mathématiques :** Égalité ($=$) et Inégalité ($\neq$).
  - Vous ne pouvez pas les additionner, les soustraire ou les classer.
- **Exemples :**
  - Couleurs : $\{\text{Rouge}, \text{Bleu}, \text{Vert}\}$
  - Type d'animal : $\{\text{Chien}, \text{Chat}, \text{Oiseau}\}$
  - Numéros d'ID : ID Utilisateur 89402 vs 10293.
  - *Note :* Même s'ils ressemblent à des nombres, ce sont des étiquettes.

Nominal : Implication ML

- Vous ne pouvez pas les fournir directement à une régression ou un réseau de neurones.
- Le modèle ne doit pas supposer de relation mathématique entre "Chien" et "Chat".
- Il faut encoder l'information (par exemple via One-Hot Encoding).

2. Ordinal

- **Définition :** Catégories ayant un ordre logique clair, mais la *distance* entre les catégories est inconnue ou inégale.
- **Opérations :** Comparaison ($<, >$) et Égalité.
- **Exemples :**
  - Niveau d'éducation : {Lycée, Licence, Master, Doctorat}
  - Satisfaction client : {Mécontent, Neutre, Content}
  - Taille de T-shirt : $\{S, M, L, XL\}$

Ordinal : Implication ML

- Bien que nous sachions que $L > S$, nous ne pouvons pas dire que la différence entre Small et Medium est mathématiquement la même que la différence entre Medium et Large.
- $(M - S) \neq (L - M)$.

3. Intervalle

- **Définition :** Données numériques continues où les intervalles entre les valeurs sont cohérents et égaux, mais il n'y a **pas de vrai point zéro**.
- Zéro ne signifie pas "rien" ; c'est juste un autre point sur l'échelle.
- **Opérations :** Addition $(+)$ et Soustraction $(-)$. Les ratios n'ont pas de sens.
- **Exemples :**
  - **Température :** $0°C$ est froid, ce n'est pas "pas de température".
  - **Années calendaires :** L'an 0 n'est pas le début du temps.

4. Ratio

- **Définition :** Données numériques continues avec intervalles égaux et un **vrai point zéro absolu**.
- Zéro signifie l'absence de la variable.
- **Opérations :** Toutes $(+, -, \times, \div)$.
- **Exemples :**
  - **Revenu :** 0\$ signifie aucun revenu. $100k$\$ est exactement le double de $50k$\$.
  - **Distance/Poids :** $0kg$ signifie aucun poids.

Ratio : Implication ML

- Ce sont les variables les plus flexibles.
- La standardisation (Standard Scaling), les transformations logarithmiques et l'ingénierie de caractéristiques basée sur des ratios fonctionnent le mieux ici.

Philosophie de la Pipeline ML

Tout est Transformation

- Au sens mathématique, tout est une fonction $f(X) \rightarrow X'$.
- **Scikit-Learn** est construit sur cette vision : `.fit()` / `.transform()`.
- Bien que le code soit le même, nous distinguons les opérations par leur **INTENTION** :

Les 4 Types de Transformations

1. Réparatrices (Cleaning)

- *But :* Restaurer la réalité.
- *Philo :* "La donnée est cassée."
- *Ex :* Imputation (Manquants).

2. Adaptatives (Scaling)

- *But :* Traduire pour l'algorithme.
- *Philo :* "L'ordi ne parle pas cette langue."
- *Ex :* StandardScaler, OneHot.

3. Créatives (Engineering)

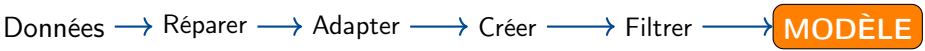
- *But :* Révéler des signaux cachés.
- *Philo :* "Info trop subtile pour le modèle."
- *Ex :* Ratios, Dates.

4. Sélectives (Selection)

- *But :* Focaliser l'attention.
- *Philo :* "Trop d'info tue l'info."
- *Ex :* VarianceThreshold.

La Vision Pipeline

Une Pipeline n'est qu'une chaîne de transformations successives.



Données Manquantes (Réparatrice)

Gestion des Données Manquantes

- Les algorithmes standards (Régression, SVM, Réseaux de Neurones) ne gèrent pas mathématiquement les `NaN`.
- `NaN` dans une multiplication matricielle = Plantage.
- **Première étape :** Comprendre l'approche rudimentaire (La Suppression).

Approche A : Suppression par Liste (Lignes)

- Il faut supprimer tout exemple contenant au moins un `NaN`.
- Vous réduisez la taille de votre jeu de données N à $N_{complet}$.
- **Commande :** `df.dropna(axis=0)`
- **Condition de sécurité :**
  1. Faible volume ($< 5\%$).
  2. Aléatoire.

Avantages & Inconvénients (Lignes)

- (+) Simple. Préserve la vraie distribution pour les points restants. N'invente pas de données.
- (-) Réduction drastique de la taille de l'échantillon. Risque de biais si l'absence n'est pas aléatoire (ex : supprimer les riches qui cachent leur revenu).

Approche B : Suppression de Caractéristiques (Colonnes)

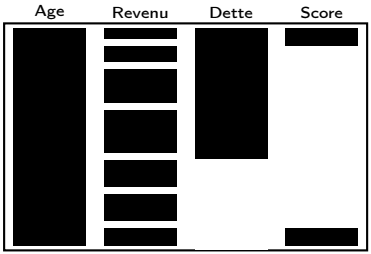
- Si une variable est remplie de trous, elle ne vaut peut-être pas la peine d'être sauvée.
- Il est préférable de retirer entièrement la variable.
- **Commande :** `df.drop(columns=['col'])`
- **Condition de sécurité :**
  1. Forte absence ($> 60\%$).
  2. Non-pertinence (faible corrélation).

Avantages & Inconvénients (Colonnes)

- (+) Supprime le bruit.
- (-) Risque de perdre un prédicteur critique. Parfois, l'absence *est* l'information (ex : Score Crédit manquant = Risque).

Visualiser les Trous : La Matrice de Nullité

- **Outil :** `missingno.matrix(df)`
- Chaque colonne = une variable.
- **Coup d'œil rapide :**
  - **Noir** = Donnée présente.
  - **Blanc** = Donnée manquante.
- Permet de repérer les corrélations d'absence (MNAR).



Exemple de Matrice de Nullité

Le Compromis de la Suppression

Règle d'or : *Ne supprimez jamais à l'aveugle.*

Données Manquantes : Avancé
(Réparatrice)

Pourquoi ne pas simplement supprimer ?

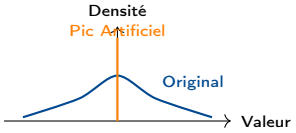
Le Problème de `dropna()`

Supprimer des lignes réduit la taille de votre échantillon (N diminue).

- **Perte de Puissance Statistique :** Moins de données = Modèle moins performant.
- **Biais :** Si l'absence n'est pas aléatoire, supprimer crée un échantillon biaisé qui ne représente plus la réalité.

Technique 1 : Moyenne / Médiane

- **Action :** Remplacer le trou par la moyenne (si normal) ou la médiane (si asymétrique).
- **Avantage :** Rapide, simple, conserve la moyenne globale.
- **Inconvénient : Détruit la variance** et les corrélations. Crée un pic artificiel qui peut tromper le modèle.



Technique 2 : Valeur Arbitraire

- **Action :** Remplacer par `-1`, `-999` ou `"Inconnu"`.
- **Cible :** Modèles à base d'arbres (Random Forest, XGBoost).
- **Avantage :** L'arbre peut isoler cette valeur ("Si Valeur est -1 alors...") et donc *apprendre* le motif de l'absence.
- **Inconvénient :** Catastrophique pour Régression Linéaire ou KNN (qui vont croire que -999 est une vraie mesure).

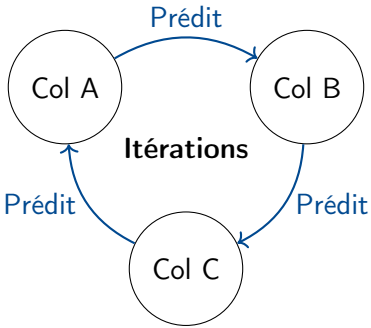
Technique 3 : KNN (K-Nearest Neighbors)

- **Action :** Trouver les K "voisins" les plus proches et faire leur moyenne.
- **Avantage :** Très précis. Respecte la structure locale des données.
- **Inconvénient :**
  1. Lent sur gros volumes de données.
  2. Nécessite de mettre à l'échelle (StandardScaler) avant.

Technique 4 : MICE (Equations Chaînées) - Concept

- **Le problème :** KNN est lent et la moyenne détruit la variance.
- **L'idée MICE :** Traiter chaque valeur manquante comme un problème de Machine Learning en soi.
- **Algorithme :**
  1. Remplir tous les trous avec la moyenne (temporaire).
  2. Pour la colonne A : On efface le remplissage temporaire (elle redevient cible).
  3. Entraîner une régression : $Colonnes(B, C) \rightarrow Colonne(A)$.
  4. Prédire les manquants de A.
  5. Répéter pour B, puis C... (C'est un cycle !)

MICE - Le Cycle



Scikit-Learn : `IterativeImputer` (Expérimental, doit être activé explicitement)

Résumé Imputation

Technique	Bon pour...	Défaut Fatal
Moyenne/Médiane	Solutions rapides	Déforme variance & corrélations.
Arbitraire	Arbres (Random Forest)	Casse modèles linéaires.
KNN	Petits datasets	Lent. Nécessite Scaling.
MICE	Précision max	Complexe.

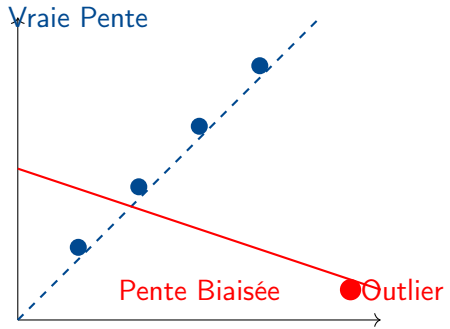
Gestion des Outliers (Réparatrice)

Détection et Traitement des Données Aberrantes

- **Outlier :** Point qui diffère significativement des autres.
- **Double tranchant :**
  1. **Erreur :** Mesure fausse (Taille = 20m).
  2. **Or :** Signal précieux (Fraude).

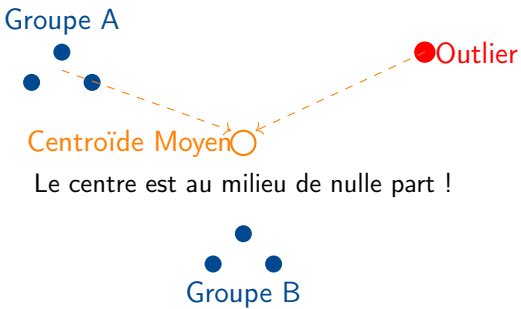
Impact Outliers : Régression Linéaire

Le piège des moindres carrés (MSE)



Impact Outliers : K-Means

Centroïdes déplacés et clusters fusionnés



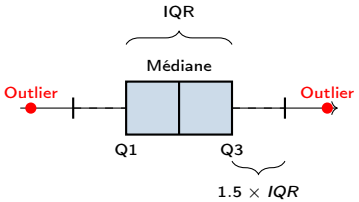
1. Z-Score (Paramétrique)

- Suppose une **Distribution Normale**.
- $$z = \frac{x - \mu}{\sigma}$$

  (où μ = moyenne, σ = écart-type)
- **Règle :** Si $|z| > 3$, c'est un outlier.
- *Défaut :* La moyenne et sigma sont eux-mêmes influencés par l'outlier.

2. Méthode de Tukey / IQR (Box Plot)

- **Robuste** (Non-paramétrique).
- **IQR** $= Q3 - Q1$.
- **Limites :** $1.5 \times IQR$.



Détection d'Anomalies (Multivariée)

- Quand X est normal et Y est normal, mais la combinaison (X, Y) est impossible.
  - *Ex : Âge 10 ans avec Salaire 50k$.*

Algorithmes d'Anomalies

Isolation Forests

- Basé sur Random Forest.
- Isole les anomalies car elles demandent moins de "coupes" dans l'arbre (branches courtes).
- Idéal pour haute dimension.

LOF (Local Outlier Factor)

- Basé sur la densité locale vs voisins.

Que faire des Outliers ?

1. **Élagage (Suppression) :** Si erreur certaine ou volume infime.
2. **Plafonnement (Winsorisation) :** Ramener les valeurs extrêmes au 99ème centile. Garde l'info sans détruire la pente.
3. **Traiter comme Manquant :** Mettre `NaN` puis Imputer.
4. **Transformation :** Logarithme (compresse l'échelle).

Scaling & Transformation (Adaptative)

Pourquoi Mettre à l'Échelle ?

- Âge (0-100) vs Revenu (0-1 000 000).
- Pour l'ordinateur, Revenu est 10 000x plus important.

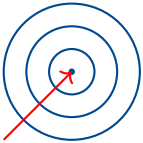
Impact Scaling : Descente de Gradient

Non Scalé

Convergence Lente (Zigzag)

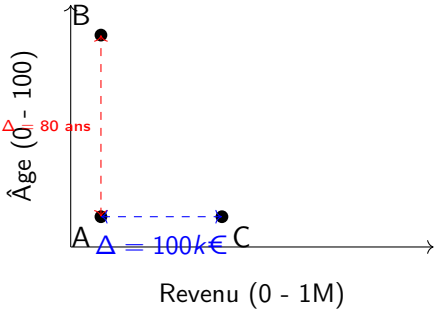
Scalé

Convergence Rapide (Directe)



Impact Scaling : Distances (KNN / K-Means)

L'axe dominant écrase l'information



- **Visuellement :** C semble proche de A (sur le papier).
- **Réalité :** $\Delta_{AC} \gg \Delta_{AB}$ car l'échelle du Revenu écrase tout.
- **Conséquence :** L'algo ne "voit" que le Revenu.

1. Standard Scaling (Z-Score)

- $$z = \frac{x - \mu}{\sigma}$$

  (où μ = moyenne, σ = écart-type)
- **Résultat :** Moyenne = 0, Variance = 1.
- **Caractéristiques :** Ne borne pas les données. Préserve les outliers.
- **Usage :** ML Général, SVM, Réseaux de Neurones.

2. Min-Max Scaling

- $$x' = \frac{x - \min}{\max - \min}$$
- **Résultat :** Compressé entre [0, 1].
- **Caractéristiques :** Très sensible aux outliers (écrase les données normales).
- **Usage :** Images (pixels), Deep Learning exigeant [0,1].

3. Robust Scaling

- $$x' = \frac{x - \text{median}}{IQR}$$
- **Résultat :** Scalé sur les 50% du milieu.
- **Usage :** Données avec beaucoup d'outliers.

Quelle méthode choisir ?

Scénario	**Technique**
ML Général (SVM, Régression)	`StandardScaler`
Deep Learning / Images	`MinMaxScaler`
Beaucoup d'Outliers	`RobustScaler`
Asymétrie (Revenus)	`PowerTransformer` (Log/Yeo)

Encodage Catégoriel (Adaptative)

Transformer les Mots en Nombres

- On ne peut pas multiplier "Rouge" par 0.5.

1. One-Hot Encoding (OHE)

- **Pour :** Nominal (Non ordonné).
- **Méthode :** Une colonne binaire par catégorie.
- **Attention :**
  - **Dummy Trap :** Supprimer une colonne pour éviter la multicolinéarité parfaite (`drop_first=True`).
  - **Malédiction dimensionnalité :** Explose la mémoire si trop de catégories.

2. Encodage Ordinal

- **Pour :** Ordinal (Ordonné).
- **Méthode :** Entier par rang (Petit=0, Moyen=1, Grand=2).
- **Risque :** Ne pas utiliser sur du Nominal (crée un faux ordre mathématique).

Feature Engineering (Créative)

Ingénierie des caractéristiques

"Le ML appliqué est essentiellement de l'ingénierie de caractéristiques." - *Andrew Ng*

- Utiliser la connaissance métier pour créer de *nouvelles* variables.
- Rendre le problème plus facile pour le modèle.

1. Interactions

- **Arithmétiques :**
  - Somme (Revenu Foyer).
  - Ratio (Dette / Limite). *Très puissant.*
- **Polynomiales :**
  - Carrés, Cubes, Produits croisés ($a \times b$).
  - Permet aux modèles linéaires de capturer des courbes.

2. Date et Heure

- **Extraction :** Heure, Jour, Mois, Weekend (0/1).

3. Géospatial (Lat/Lon)

- Coordonnées brutes difficiles à exploiter.
- **Distances :** Calculer la distance vers des points clés (ex : Centre-Ville, Gares).
- **Clustering :** K-Means sur Lat/Lon pour créer des "Quartiers" (ID_Cluster).

4. Texte (Dans données tabulaires)

- **Bag-of-Words :** Compte des mots.
- **TF-IDF :** Fréquence pondérée par la rareté. Pénalise les mots courants ("le", "et").
- **Embeddings (Word2Vec, BERT) :** Vecteurs denses. Capture le **sens** et le contexte. (État de l'art).

Sélection de Features (Sélective)

La Malédiction de la Dimensionnalité

- Trop de features $\rightarrow$ Le volume explose $\rightarrow$ Les points deviennent isolés (Sparsity).
- Conséquences : Surapprentissage, lenteur, mauvaise généralisation.
- **Solution :** Sélectionner le signal, jeter le bruit.

1. Méthodes de Filtre (Statistiques)

- Indépendant du modèle. Rapide.
- **Variance Threshold :** Supprimer les constantes (Variance = 0).
- **Corrélation :**
  - Feature vs Cible (On veut élevé).
  - Feature vs Feature (Multicolinéarité - On veut faible).
- **Information Mutuelle :** Capture les dépendances non-linéaires.

2. Méthodes Wrapper (Itératives)

- Essai-Erreur avec un modèle.
- **Forward Selection :** Ajouter 1 par 1.
- **RFE (Recursive Feature Elimination) :** Tout mettre, puis élaguer les plus faibles récursivement.
- *Coûteux mais efficace.*

3. Méthodes Intégrées (Embedded)

- Sélection pendant l'entraînement.
- **Lasso (L1) :** Annule les coefficients des features inutiles.
- **Importance des Arbres :** Random Forest calcule le gain d'impureté de chaque feature.

L'Entonnoir de Sélection

1. **Enlever les Déchets :** Variance nulle.
2. **Enlever les Doublons :** Haute corrélation.
3. **Sélection Fine :** RFE ou Importance Arbre.

Données Déséquilibrées (Réparatrice)

Données Déséquilibrées

- Classe Majoritaire (99%) vs Minoritaire (1%).
- *Exemples :* Fraude, Cancer, Défaut Usine.
- Le modèle "bête" prédit tout en Majoritaire.

Le Piège de la Métrique

- **Accuracy :** 99% (sur un modèle inutile).
- **Les Bonnes Métriques :**
  - **Précision :** Qualité des positifs.
  - **Rappel :** Quantité de positifs trouvés.
  - **F1-Score :** Moyenne harmonique.

Techniques de Rééchantillonnage

1. **Sous-échantillonnage (Majorité) :**
   - Supprimer des lignes majoritaires.
   - *Risque :* Perte d'info. Bon pour gros datasets.
2. **Sur-échantillonnage (Minorité) :**
   - **Aléatoire :** Dupliquer (Risque d'overfitting).
   - **SMOTE :** Créer des points synthétiques par interpolation entre voisins.

Poids des Classes (Cost-Sensitive)

- Ne pas toucher aux données, changer la pénalité.
- `class_weight='balanced'`
- Erreur sur Minorité coûte 100x plus cher que sur Majorité.
- *Recommandation : Essayer ceci en premier.*

Pipelines & Leakage

Data Leakage

- Utiliser des infos du futur (Test) pour entraîner le modèle.
- Résultat : Scores excellents en dev, catastrophiques en prod.

Types de Fuites

1. **Prétraitement :** Calculer la moyenne (pour imputation/scaling) sur tout le dataset AVANT le split.
2. **Temporelle :** Splitter aléatoirement une série temporelle (apprendre du futur).

Validation Croisée (Stratégie)

Pourquoi un simple Split ne suffit pas ?

- **Le hasard est dangereux :**
  - Si votre Test Set est "facile" $\rightarrow$ Vous êtes trop optimiste.
  - Si votre Test Set est "difficile" (cas rares) $\rightarrow$ Vous sous-estimez votre modèle.
- **Gaspillage de données :**
- En faisant un split 80/20, vous perdez 20% de l'information pour l'entraînement.
- **Solution :** La Validation Croisée (Cross-Validation).

Le Concept de K-Fold Cross-Validation

- On divise les données en K parties (plis).
- On entraîne K fois le modèle.
- Chaque pli sert une fois de Test Set et $K - 1$ fois de Train Set.

Cross-Validation : Avantages et Inconvénients

Avantages (+)

- **Robustesse :** La moyenne des scores est une bien meilleure estimation de la réalité.
- **Utilisation Totale :** Chaque point sert à tester et à entraîner.

Inconvénients (-)

- **Coût :** Entraîner K modèles prend K fois plus de temps.
- *Ex : 5-Fold = 5x plus long.*

Quand l'utiliser ? *Toujours*, sauf si votre dataset est gigantesque ($>$ 1M lignes) ou si vous faites du Deep Learning (trop lent).

Le Workflow Idéal

1. **Split :** Isoler le Test Set.
2. **Pipeline :** Définir la chaîne.
3. **Cross-Validation :** `cross_val_score(pipeline)`. Le pipeline protège contre les fuites dans chaque pli.
4. **Fit Final :** Sur tout le Train.
5. **Predict :** Sur Test.

Résumé du Cours

Nous avons couvert la chaîne complète :

1. **Théorie :** Types de variables.
2. **Nettoyage :** Manquants & Outliers.
3. **Préparation :** Scaling & Encodage.
4. **Création :** Feature Engineering & Sélection.
5. **Robustesse :** Pipelines & Gestion du déséquilibre.