Apprentissage automatique Chapitre 1 - Les données

Halim Djerroud

LISV
Laboratoire d'ingénierie
des systèmes de Versailles

UVSQ
université PARIS-SACLAY

révision : 02/12/2025

Plan du cours et déroulement

Plan du cours :

1. **Introduction.**
2. **Les Données.**
3. Apprentissage supervisé et non supervisé.
4. Les réseaux de neurones.

Déroulement :

- 18 heures de cours 6 séances de 3 heures.
- Deux contrôles continus (QCM).
- Un projet à faire en binôme.
- Un examen écrit.

Les types d'Intelligence Artificielle

IA Statistique (Machine Learning) :

- Apprentissage à partir de données
- Reconnaissance de motifs
- Approches probabilistes
- **Exemples :**
  - MLP (Multi-Layer Perceptron)
  - Réseaux de neurones
  - Arbres de décision

IA Symbolique :

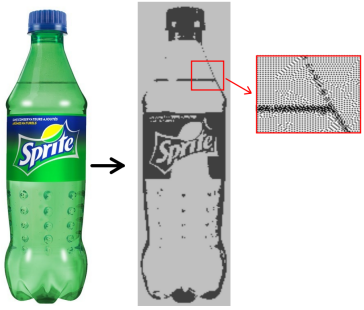
- Raisonnement logique
- Représentation explicite des connaissances
- Règles et symboles
- **Exemples :**
  - Systèmes experts
  - Systèmes multi-agents
  - Planification automatique

Dans ce cours

Nous nous concentrerons principalement sur l'**IA statistique** et l'**apprentissage automatique**.

Exemple 1

Écrire un programme qui reconnaît des bouteilles ?

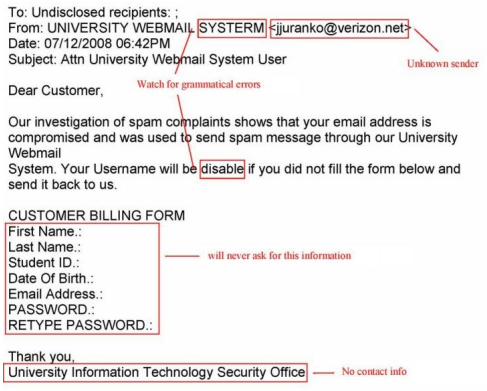



```
if(img[x,y] == PLEIN) && ...
if(img[x,y+1] == PLEIN) && ...
if(img[x,y+2] == VIDE) && ...
...
{
    return TRUE;
}
else{
    return FALSE;
}
```

- À votre avis, est-il possible d'écrire ce programme ?

Exemple 2

Écrire un programme qui reconnaît les Spams ?




```
if(is_autorized_senders(list)) && .
if(text_content(list_words) &&...
if(spell-check(email)<0.8) && ...
...
{
    return SPAM;
}
else{
    return NOT-SPAM;
}
```

LISV Laboratoire d'Ingénierie des systèmes de Versailles | UVSQ université PARIS-SACLAY

Idée de l'Intelligence Artificielle

- Développer des programmes capables de créer d'autres programmes.
- Faire en sorte que les données génèrent les programmes automatiquement.
- Cela consiste à concevoir des programmes génériques qui sont ajustés en fonction des données fournies.
- Ces programmes ont la capacité d'apprendre à partir des données.
- Les données utilisées pour cet apprentissage sont appelées les *données d'apprentissage*.

On appelle ce type de programmation : **Apprentissage Automatique**

Comment les programmes apprennent ?

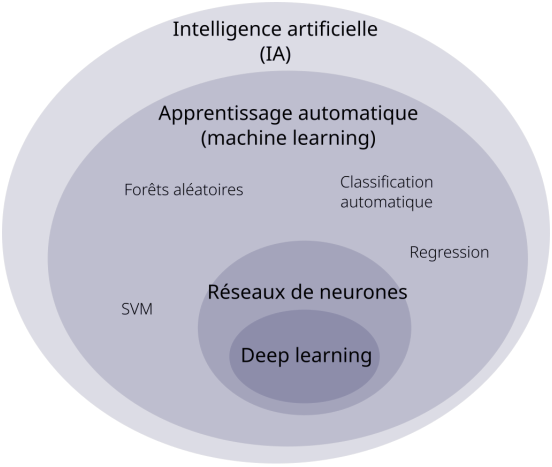


Figure: Source wikipedia

Qu'est-ce que l'apprentissage automatique

Wikipédia (octobre 2024)

L'apprentissage automatique (en anglais : machine learning, litt. "apprentissage machine"), apprentissage artificiel ou apprentissage statistique est un champ d'étude de l'intelligence artificielle qui se fonde sur des approches mathématiques et statistiques pour donner aux ordinateurs la capacité d'"*apprendre*" à partir de données, c'est-à-dire d'améliorer leurs performances à résoudre des tâches sans être explicitement programmés pour chacune... On parle d'apprentissage statistique car l'apprentissage consiste à créer un modèle dont l'erreur statistique moyenne est la plus faible possible.

Comment les programmes apprennent ?

- L'apprentissage se fait à partir de la **reconnaissance de motifs** dans les données.
- On utilise des **modèles** qui ajustent leurs **paramètres** pour mieux prédire ou classifier de nouvelles données.
- Le processus se divise en trois phases :
  1. **Entraînement** : le modèle est formé sur un ensemble de données d'apprentissage pour trouver les meilleurs paramètres.
  2. **Validation** : on évalue la performance du modèle sur un ensemble de données différent.
  3. **Test** : le modèle est évalué sur un ensemble de données encore inédit pour mesurer sa performance finale.
- L'objectif est de créer un modèle **généraliste** capable de bien prédire sur de nouvelles données.

Comment les programmes apprennent ?

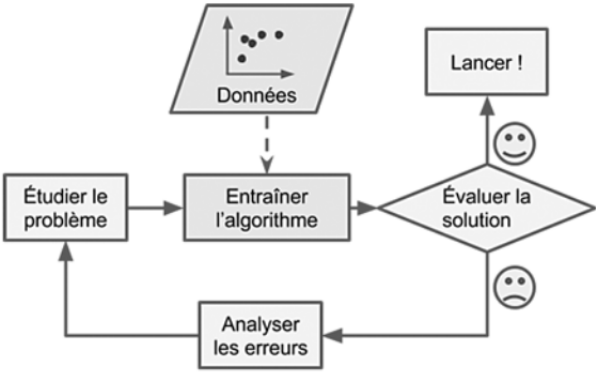


Figure: Source livre Machine Learning avec Scikit-Learn

Les données

Les Données

Objectifs du chapitre :

1. Comprendre l'importance des données dans les projets d'IA.
2. Identifier les différentes sources et types de données.
3. Apprendre les étapes du traitement des données avant l'entraînement des modèles.
4. Maîtriser les bonnes pratiques en matière de gestion des données.

Qu'est-ce qu'une donnée ?

- **Définition** : Informations **brutes**, **quantitatives** ou **qualitatives**, utilisées pour entraîner un modèle d'IA.
- **Caractéristiques des bonnes données** :
  - **Pertinence** : Doit être liée à la tâche à accomplir.
  - **Fiabilité** : Doit être exempte de bruit ou d'erreurs.
  - **Représentativité** : Couvrir l'ensemble des cas possibles.
  - **Quantité** : Un volume suffisant de données est nécessaire.

Données Brutes, Quantitatives et Qualitatives

1. Données Brutes : Données collectées directement, non traitées, contenant souvent du bruit, des doublons ou des valeurs manquantes.

- **Exemples** : Relevés de capteurs, fichiers CSV bruts, vidéos de surveillance.
- **Problèmes** : Données manquantes, anomalies, bruit.
- **Solution** : Nettoyage, normalisation, filtrage.

2. Données Quantitatives : Données numériques permettant des calculs mathématiques (discrètes ou continues).

- **Exemples** : Température (36,5°C), nombre de clics, distances (2,35 m).
- **Sous-types** :
  - **Discrètes** : Nombre d'enfants (1, 2, 3, ...).
  - **Continues** : Température, distance, durée (2,35 min).

Données Brutes, Quantitatives et Qualitatives

3. **Données Qualitatives** : Données non numériques qui classent ou catégorisent des objets.
   - **Exemples** : Couleurs (rouge, bleu), avis clients (positif, neutre, négatif), type d'appareil (smartphone, tablette).
   - **Types** :
     - **Catégoriques** : Aucune notion d'ordre (ex : couleur, genre).
     - **Ordinales** : Classées par ordre (ex : avis client : mauvais, moyen, bon).

Comparatif

- **Données Brutes** : Non traitées, bruitées, non nettoyées.
- **Données Quantitatives** : Numériques, mesurables (discrètes ou continues).
- **Données Qualitatives** : Catégories, classes ou labels (ordinales ou nominales).

Types et Sources de Données

Types de Données

- **Données Numériques** : Température, distance, age.
- **Données Catégoriques** : Genre, statut (actif/inactif), labels de classes.
- **Données Textuelles** : Avis clients, résumés de textes.
- **Données Visuelles** : Images, vidéos.
- **Séries Temporelles** : Prix des actions, données de capteurs.

Sources de Données

- **Sources Publiques** : Kaggle, UCI Machine Learning Repository.
- **Sources Internes** : Bases de données d'entreprise, CRM, ERP.
- **Données Simulées** : Données générées par des simulations.
- **Données du Web** : Collecte (web scraping) respectant les règles RGPD.

Qualité des Données

- **Problèmes courants** :
  - Données manquantes.
  - Valeurs aberrantes.
  - Données dupliquées.
  - Biais des données.
- **Solutions** :
  - Imputation des valeurs manquantes.
  - Filtrage des valeurs aberrantes.
  - Suppression des doublons.

Préparation des Données

- **Nettoyage** : Supprimer les valeurs manquantes, corriger les erreurs.
- **Transformation** : Encodage des catégories, normalisation.
- **Augmentation** : Rotation, zoom, translation (surtout pour les images).
- **Réduction de Dimension** : PCA, t-SNE, UMAP.

Étapes de Préparation des Données

1. **Importer les données** : Chargement à partir de fichiers CSV, BD, API, etc.
2. **Examiner la structure des données** : Type des colonnes, valeurs manquantes, type de données, statistiques descriptives.
3. **Explorer et visualiser les données** : Boîtes à moustaches, histogrammes, nuages de points, outliers, etc.
4. **Nettoyer les données** : Gestion des valeurs manquantes, des valeurs aberrantes, suppression des doublons.
5. **Gérer les variables qualitatives** : Application des techniques d'encodage (One-Hot Encoding, Label Encoding).
6. **Normaliser et mettre à l'échelle les variables** : Standardisation, normalisation des variables numériques pour aligner les plages de valeurs.
7. **Rechercher des corrélations** : Matrice de corrélation pour identifier les relations entre les variables.
8. **Expérimenter avec des combinaisons de variables** : Feature Engineering, PCA, réduction de la dimensionnalité.

Éthique et RGPD

Pourquoi l'éthique des données est essentielle ?

Garantir une IA fiable, équitable et conforme aux lois en protégeant les utilisateurs.

- **Protection des données personnelles**
  - Collecte minimale (principe de minimisation).
  - Consentement explicite et utilisation transparente.
- **Anonymisation et Sécurisation**
  - Pseudonymisation, floutage, agrégation.
  - Stockage sécurisé (chiffrement, contrôle d'accès).
- **Équité et absence de biais**
  - Détection et réduction des biais dans les données.
  - Évaluation régulière des impacts sur les utilisateurs.
- **Traçabilité et responsabilité**
  - Documenter l'origine des données et les transformations.
  - Garantir l'auditabilité des modèles et des décisions.

Dataset

Importer les données

Objectifs du chapitre :

1. Les différentes façons d'importer des données
2. Utiliser des dataset existants

Pandas : Introduction et Import de Données

Bibliothèque d'analyse de données

Pandas permet de manipuler et analyser des données structurées (tableaux, séries temporelles)

Import depuis différentes sources

```python
import pandas as pd

# CSV (local ou URL)
df = pd.read_csv('fichier.csv')

# Excel
df = pd.read_excel('fichier.xlsx', sheet_name='Feuille1')

# Base SQL
import sqlite3
conn = sqlite3.connect('base.db')
df = pd.read_sql_query("SELECT * FROM table", conn)

# API JSON
import requests
df = pd.DataFrame(requests.get('https://api.exemple.com/data').json())
```

Pandas : Exploration et Manipulation

Explorer les données

```python
# Apercu des donnees
print(df.head())         # 5 premières lignes
print(df.info())         # Types et valeurs manquantes
print(df.describe())     # Statistiques descriptives

# Selection et filtrage
df['colonne']            # Selectionner une colonne
df[['col1', 'col2']]     # Plusieurs colonnes
df[df['age'] > 25]       # Filtrer les lignes
```

Manipuler les données

```python
# Ajouter/modifier des colonnes
df['nouvelle_col'] = df['col1'] + df['col2']

# Supprimer des colonnes ou lignes
df.drop(columns=['col1'], inplace=True)
df.dropna()  # Supprimer les lignes avec valeurs manquantes
```

Pandas : Analyse et Export

Opérations d'agrégation

```
# Grouper et agreger
df.groupby('categorie')['valeur'].mean()
df.groupby('categorie').agg({'valeur': ['mean', 'sum', 'count']})

# Trier les donnees
df.sort_values(by='colonne', ascending=False)

# Fusionner des DataFrames
df_merged = pd.merge(df1, df2, on='cle', how='inner')
```

Exporter les résultats

```
# Exporter vers differents formats
df.to_csv('resultat.csv', index=False)
df.to_excel('resultat.xlsx', sheet_name='Resultats')
df.to_sql('table_name', conn, if_exists='replace')
```

Importer des Données avec Pandas

```python
import pandas as pd
import sqlite3
import requests

# CSV (local ou URL)
df = pd.read_csv('fichier.csv')  # ou 'https://exemple.com/fichier.csv'

# Excel
df = pd.read_excel('fichier.xlsx', sheet_name='Feuille1')

# Base de donnees SQL
conn = sqlite3.connect('base.db')
df = pd.read_sql_query("SELECT * FROM table_name", conn)

# API Web (JSON)
response = requests.get('https://api.exemple.com/donnees')
df = pd.DataFrame(response.json())

# Afficher les premiEres lignes
print(df.head())
```

Introduction aux Datasets

- Un **dataset** est un ensemble de données utilisé pour entraîner, valider et tester les modèles d'apprentissage automatique.
- Les datasets sont essentiels pour la modélisation et la généralisation des modèles de Machine Learning.
- **Exemples de datasets courants** :
  - Iris (classification des fleurs)
  - MNIST (chiffres manuscrits)
  - Boston Housing (prédiction des prix des maisons)

Utilisation des Datasets avec sklearn

- La bibliothèque **scikit-learn (sklearn)** propose de nombreux datasets standards pour l'entraînement des modèles.
- Les datasets peuvent être chargés via `sklearn.datasets`.
- Les types de datasets incluent :
  - **Datasets intégrés** (Iris, Boston, etc.)
  - **Datasets de flux** (fetch_*) pour des jeux de données plus volumineux
  - `https://scikit-learn.org/1.5/api/sklearn.datasets.html`

Dataset Iris

- **Iris** : Dataset de classification des fleurs en trois espèces basées sur la longueur et la largeur des sépales et des pétales.

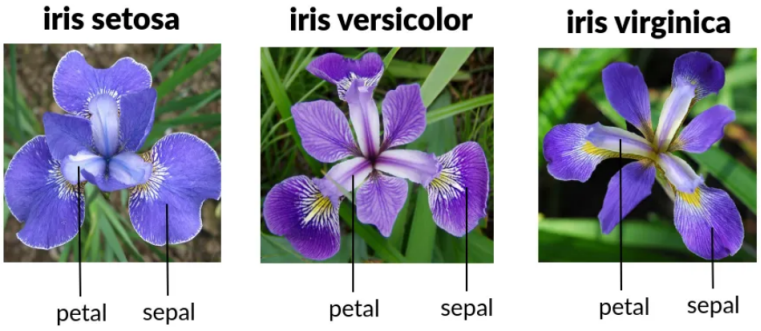


Figure: Source web

Chargement du Dataset Iris

```python
from sklearn.datasets import load_iris
import pandas as pd

# Charger le dataset Iris
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Afficher les 5 premieres lignes
print(df.head())
```

Chargement du Dataset Boston Housing

- **Boston Housing** : Données sur les prix des maisons de Boston en fonction de plusieurs caractéristiques socio-économiques.

```
from sklearn.datasets import load_boston
import pandas as pd

# Charger le dataset Boston Housing
boston = load_boston()
df = pd.DataFrame(boston.data, columns=boston.feature_names)

# Afficher les 5 premières lignes
print(df.head())
```

Dataset MNIST

- **MNIST** : Images de chiffres manuscrits (0-9) utilisées pour la reconnaissance de chiffres.

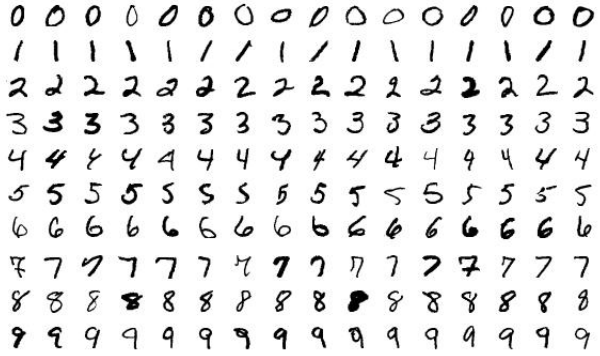


Figure: Source web

Chargement du Dataset MNIST

```python
from sklearn.datasets import fetch_openml

# Charger le dataset MNIST
mnist = fetch_openml('mnist_784', version=1)

# Afficher les noms des colonnes et le nombre d'echantillons
print(mnist.data.shape)
```

Structure des Datasets sklearn

- Les datasets de **sklearn** sont généralement des objets de type `Bunch`, similaires à des dictionnaires Python.
- Les principaux éléments d'un dataset **sklearn** incluent :
  - **data** : Tableau 2D (n_échantillons x n_caractéristiques) contenant les données d'entrée.
  - **target** : Tableau 1D ou 2D contenant les étiquettes ou les cibles.
  - **feature_names** : Noms des caractéristiques ou des colonnes (optionnel).
  - **target_names** : Noms des classes ou des cibles (pour les tâches de classification).
  - **DESCR** : Description textuelle du dataset.
- Ces composants permettent de comprendre et de manipuler facilement les données.

Exemple de Structure d'un Dataset sklearn

```python
from sklearn.datasets import load_iris

iris = load_iris()

# Afficher les cles du dataset
print(iris.keys())

# Acceder a la description
print(iris['DESCR'])

# Afficher les 5 premieres lignes des donnees
print(iris['data'][:5])
```

Explication des Principaux Attributs

- **data** :
  - Contient les données d'entrée sous forme de matrice 2D.
  - Chaque ligne correspond à un échantillon, chaque colonne à une caractéristique.
- **target** :
  - Contient les étiquettes ou les cibles associées à chaque échantillon.
- **feature_names** :
  - Liste des noms des colonnes ou caractéristiques.
- **DESCR** :
  - Contient la description complète du dataset, son origine, et les détails sur ses colonnes.

Les Datasets

- Les datasets sont essentiels à l'entraînement des modèles de Machine Learning.
- Scikit-learn fournit des datasets intégrés pour faciliter l'entraînement et la validation.
- Il est possible de charger des datasets plus volumineux via les fonctions `fetch_*` de sklearn.

Structure des données

Examiner la structure des données

Objectifs du chapitre :

1. Type des colonnes, valeurs manquantes, type de données, statistiques descriptives.

Examiner la Structure des Données

- Comprendre la nature des données, identifier les problèmes potentiels et de choisir les bonnes stratégies de prétraitement.
- Les principales actions incluent :
  - **Visualisation des premières lignes** : Vérification du contenu des premières lignes.
  - **Types de données** : Vérification des types de colonnes (numériques, catégoriques, etc.).
  - **Statistiques descriptives** : Calcul de la moyenne, médiane, écart-type, etc.
  - **Recherche des valeurs manquantes** : Identification des colonnes contenant des valeurs manquantes.
  - **Détection des valeurs aberrantes** : Identification des valeurs extrêmes ou anomalies.

Affichage des Premières Lignes

```python
import pandas as pd

# Charger le fichier CSV
df = pd.read_csv('chemin/vers/fichier.csv')

# Afficher les 5 premieres lignes
print(df.head())
```

- **Objectif** : Obtenir un aperçu rapide du contenu du dataset.
- Permet de détecter les incohérences, les erreurs de format et les colonnes inutiles.

Vérification des Types de Données

```
# Verifier les types des colonnes
print(df.dtypes)
```

- **Objectif** : Identifier si les colonnes sont de type numérique, catégorique, texte, etc.
- Les types incorrects (par exemple, des nombres lus comme du texte) doivent être corrigés.

Statistiques Descriptives

```
# Afficher les statistiques descriptives
print(df.describe())
```

- **Objectif** : Résumer les colonnes numériques en calculant la moyenne, la médiane, l'écart-type, les quantiles, etc.
- Ces statistiques permettent d'identifier les éventuelles anomalies dans les colonnes numériques.

Recherche des Valeurs Manquantes

```
# Rechercher les valeurs manquantes
print(df.isnull().sum())
```

- **Objectif** : Identifier les colonnes contenant des valeurs manquantes.
- Cela permet de décider de la stratégie de traitement des valeurs manquantes (imputation, suppression, etc.).

Détection des Valeurs Aberrantes

```
import matplotlib.pyplot as plt

# Afficher un boxplot pour detecter les valeurs aberrantes
df['colonne_numerique'].plot(kind='box')
plt.show()
```

- **Objectif** : Identifier les valeurs extrêmes ou aberrantes.
- Ces valeurs peuvent fausser la distribution des données et doivent être traitées.

Structure des données

Explorer et Visualiser les Données

Objectifs du chapitre :

1. Graphiques : Boîtes à moustaches, histogrammes, nuages de points, outliers, etc.

Introduction à l'Exploration et la Visualisation des Données

- Mieux comprendre la distribution des données et les relations entre les variables.
- La visualisation facilite la détection des anomalies, des valeurs aberrantes et des relations cachées.
- Les principales techniques incluent :
  - Histogrammes et diagrammes à barres
  - Boîtes à moustaches (Boxplots) pour détecter les valeurs aberrantes
  - Nuages de points (scatter plots) pour étudier les relations entre les variables
  - Matrices de corrélation

Histogrammes

```
import matplotlib.pyplot as plt

# Afficher un histogramme pour une colonne specifique
df['colonne_numerique'].hist(bins=30)
plt.xlabel('Valeurs')
plt.ylabel('Frequence')
plt.title('Histogramme de colonne_numerique')
plt.show()
```

- Permet d'observer la distribution des valeurs numériques.
- Utile pour détecter la symétrie, l'asymétrie et les modes.

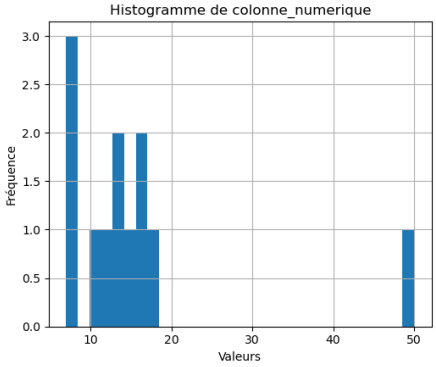
Histogrammes

```
import matplotlib.pyplot as plt
import pandas as pd

# Exemple de donnees
df = pd.DataFrame({
    'colonne_numerique': [7, 8, 8,10,
    12, 14, 14, 15, 16, 17, 18, 50]
})

# Afficher le boxplot
df['colonne_numerique'].hist(bins=30)

plt.xlabel('Valeurs')
plt.ylabel('Frequence')
plt.title('Histogramme de colonne_numeriqu
plt.show()
```



LISV Laboratoire d'Ingénierie des systèmes de Versailles | UVSQ université PARIS-SACLAY

Boîtes à Moustaches (Boxplots)

```
import matplotlib.pyplot as plt

# Afficher un boxplot pour detecter les valeurs aberrantes
df['colonne_numerique'].plot(kind='box')
plt.title('Boxplot de colonne_numerique')
plt.show()
```

- Permet de visualiser les quantiles (Q1, Q2, Q3) et les valeurs aberrantes.
- Les points au-delà des "whiskers" sont considérés comme des valeurs aberrantes.

Boîtes à Moustaches (Boxplots)

- **Définition** : Un boxplot est une méthode de visualisation graphique qui permet de résumer la distribution d'une variable numérique.
- **Composants principaux** :
  - **Médiane (Q2)** : La ligne centrale dans la boîte.
  - **Premier quartile (Q1)** et **Troisième quartile (Q3)** : Les bords de la boîte.
  - **Les "Whiskers"** (moustaches) : Les lignes s'étendant à partir de la boîte jusqu'aux points extrêmes qui ne sont pas des valeurs aberrantes.
  - **Valeurs aberrantes (outliers)** : Points situés en dehors des "whiskers".
- **Utilisation** :
  - Détection des valeurs aberrantes (outliers).
  - Visualisation de la symétrie et de la dispersion des données.
  - Comparaison de plusieurs distributions.

Exemple de Boîte à Moustaches (Boxplot)

```
import matplotlib.pyplot as plt
import pandas as pd
# Exemple de donnees
df = pd.DataFrame({
    'colonne_numerique': [7, 8, 8, 10, 12, 14, 14, 15, 16, 17, 18, 50]
})
# Afficher le boxplot
df['colonne_numerique'].plot(kind='box')
plt.title('Boxplot de colonne_numerique')
plt.show()
```

Exemple de Boîte à Moustaches (Boxplot)

- Le point au-dessus des moustaches représente une **valeur aberrante**.
- Les bordures de la boîte représentent les quartiles **Q1** et **Q3**.
- La ligne centrale représente la **médiane** (Q2) de la distribution.

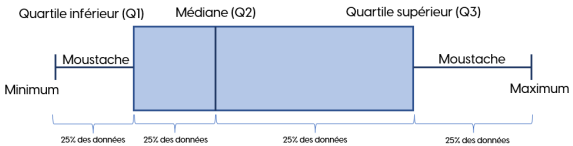
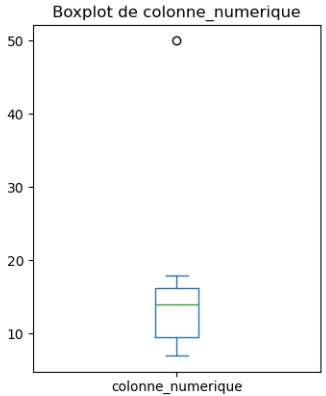


Figure: Source web



Nuages de Points (Scatter Plots)

```python
import matplotlib.pyplot as plt

# Nuage de points entre deux colonnes
plt.scatter(df['colonne1'], df['colonne2'])
plt.xlabel('Colonne 1')
plt.ylabel('Colonne 2')
plt.title('Nuage de points entre colonne1 et colonne2')
plt.show()
```

- Utile pour visualiser la relation entre deux variables numériques.
- Permet de détecter des clusters, des tendances ou des relations linéaires/non linéaires.

Matrices de Corrélation

- **Définition** : Une matrice de corrélation est une matrice montrant les coefficients de corrélation entre toutes les variables d'un dataset.
- **Objectif** : Identifier les relations linéaires entre les variables, positives ou négatives.
- **Types de corrélation** :
  - **Corrélation positive** : Lorsque l'une augmente, l'autre augmente aussi.
  - **Corrélation négative** : Lorsque l'une augmente, l'autre diminue.
  - **Corrélation nulle** : Aucune relation linéaire entre les deux variables.
- **Utilisation** :
  - Identifier les variables fortement corrélées.
  - Réduire la redondance des caractéristiques dans un modèle.
  - Sélectionner des caractéristiques pertinentes.

Exemple de Matrice de Corrélation

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.DataFrame({
'Age': [23, 45, 31, 35, 62, 45, 25, 42, 57, 36],
'Salaire': [2500, 5000, 3000, 4000, 7000, 4500, 2800, 4200, 6500, 3900],
'Experience': [1, 20, 5, 10, 30, 22, 2, 18, 25, 15]
})
# Calcul de la matrice de correlation
corr_matrix = df.corr()
# Afficher la matrice de correlation sous forme de heatmap
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Matrice de Correlation')
plt.show()
```

Exemple de Matrice de Corrélation

- La **heatmap** colorée permet de visualiser facilement les relations linéaires entre les variables.
- Les valeurs de corrélation varient de **-1** (corrélation négative forte) à **1** (corrélation positive forte).

Les Matrices de Corrélation

- **Corrélation forte** : Les valeurs proches de **1** ou **-1** indiquent une forte relation linéaire.
- **Corrélation faible** : Les valeurs proches de **0** indiquent qu'il n'existe aucune relation linéaire.
- **Utilisation des couleurs** :
  - Les tons de **bleu** indiquent une corrélation négative.
  - Les tons de **rouge** indiquent une corrélation positive.
- **Exemple** : Si la corrélation entre `Salaire` et `Experience` est proche de **0.9**, alors le salaire augmente généralement avec l'expérience.
- Les matrices de corrélation permettent de détecter les relations linéaires entre les variables numériques.
- Elles aident à identifier les caractéristiques redondantes.
- Les variables ayant une forte corrélation doivent être examinées pour réduire la multicolinéarité.

Matrice de Corrélation

```
import seaborn as sns
import matplotlib.pyplot as plt

# Calculer la matrice de correlation
corr_matrix = df.corr()

# Afficher la matrice de correlation sous forme de heatmap
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Matrice de Correlation')
plt.show()
```

- La matrice de corrélation affiche les relations linéaires entre les variables numériques.
- Les valeurs de corrélation varient de -1 (corrélation négative forte) à 1 (corrélation positive forte).

Nettoyage des Données

Nettoyage des Données

Objectifs du chapitre :

1. Avoir des données propres

Nettoyage des Données

- Le nettoyage des données consiste à identifier et corriger les problèmes présents dans les données brutes.
- Ces problèmes peuvent inclure :
  - Les valeurs manquantes.
  - Les doublons.
  - Les valeurs aberrantes (outliers).
  - Les incohérences dans les types de données.
- L'objectif est de préparer des données prêtes pour l'analyse ou l'entraînement des modèles de Machine Learning.

Gestion des Valeurs Manquantes

- Les valeurs manquantes peuvent survenir pour diverses raisons : erreurs humaines, pannes de capteurs, etc.
- **Méthodes de traitement** :
  - **Suppression** des lignes ou colonnes avec des valeurs manquantes.
  - **Imputation** : Remplacement des valeurs manquantes par la moyenne, la médiane ou une autre valeur spécifique.

```
# Identifier les valeurs manquantes
print(df.isnull().sum())

# Imputer les valeurs manquantes par la moyenne
df['colonne_numerique'].fillna(df['colonne_numerique'].mean(), inplace=True
```

Gestion des Doublons

- Les doublons sont des lignes identiques dans le dataset.
- Ils peuvent fausser les résultats de l'analyse et doivent être supprimés.
- **Méthode de traitement** : Suppression des lignes en double.

```
# Afficher le nombre de doublons
print(df.duplicated().sum())

# Supprimer les doublons
df = df.drop_duplicates()
```

Gestion des Valeurs Aberrantes

- Les valeurs aberrantes (outliers) sont des observations éloignées des autres données.
- **Méthodes de détection** :
  - Utilisation des Boxplots.
  - Utilisation de l'écart interquartile (IQR).
- **Méthodes de traitement** :
  - Suppression des outliers.
  - Transformation des valeurs.

```
import numpy as npd
# Calcul de l'IQR
Q1 = df['colonne_numerique'].quantile(0.25)
Q3 = df['colonne_numerique'].quantile(0.75)
IQR = Q3 - Q1
# Filtrer les outliers
df = df[(df['colonne_numerique'] >= Q1 - 1.5 * IQR) &
     (df['colonne_numerique'] <= Q3 + 1.5 * IQR)]
```

Gestion des Types de Données

- Parfois, les types de colonnes sont incorrects (par exemple, des nombres stockés en tant que texte).
- **Méthode de traitement** : Conversion des types de colonnes au bon format.

```
# Conversion des colonnes
df['colonne_numerique'] = pd.to_numeric(df['colonne_numerique'],
                                        errors='coerce')
df['colonne_categorique'] = df['colonne_categorique'].astype('category')
```

Nettoyage des Données

- Le nettoyage des données est une étape essentielle pour garantir la qualité des données utilisées dans les modèles d'apprentissage.
- Les principales tâches incluent la gestion des valeurs manquantes, des doublons, des valeurs aberrantes et des types de données.
- Ces actions permettent de réduire le bruit, d'améliorer la qualité des données et de renforcer les performances des modèles.

Conclusion

- L'exploration et la visualisation des données permettent de mieux comprendre la structure des données.
- Les principales techniques incluent les histogrammes, les boxplots, les scatter plots et les matrices de corrélation.
- Ces techniques facilitent la détection des anomalies, la compréhension des relations et la prise de décisions sur le prétraitement.

Gérer les variables qualitatives

Gérer les variables qualitatives

Objectifs du chapitre :

1. Avoir des données propres

Introduction à la Gestion des Variables Qualitatives

- Les variables qualitatives (ou catégoriques) représentent des valeurs discrètes non numériques.
- **Objectif** : Transformer ces variables en un format utilisable par les algorithmes de Machine Learning.
- **Exemples de variables qualitatives** :
  - `Sexe` : {Homme, Femme}
  - `Couleur` : {Rouge, Vert, Bleu}
  - `Categorie de Produit` : {Électronique, Vêtements, Alimentaire}

Méthodes de Gestion des Variables Qualitatives

- **Encodage des variables catégoriques** :
  - **Label Encoding** : Conversion des catégories en valeurs numériques (0, 1, 2, etc.).
  - **One-Hot Encoding** : Création d'une colonne binaire (0 ou 1) pour chaque catégorie.
- **Cas d'utilisation** :
  - **Label Encoding** est souvent utilisé pour les algorithmes basés sur des arbres de décision.
  - **One-Hot Encoding** est préférable pour les algorithmes linéaires et les réseaux de neurones.

Label Encoding

```python
from sklearn.preprocessing import LabelEncoder
# Exemple de dataset
df = pd.DataFrame({'Sexe': ['Homme', 'Femme', 'Femme', 'Homme', 'Homme']})
# Appliquer le Label Encoding
le = LabelEncoder()
df['Sexe_encoded'] = le.fit_transform(df['Sexe'])
print(df)
```

- **Avantage** : Simple et rapide.
- **Limite** : Peut introduire un ordre implicite non souhaité entre les catégories.

One-Hot Encoding

```python
import pandas as pd

# Exemple de dataset
df = pd.DataFrame({'Couleur': ['Rouge', 'Vert', 'Bleu', 'Rouge', 'Bleu']})
# Appliquer le One-Hot Encoding
df_encoded = pd.get_dummies(df, columns=['Couleur'])
print(df_encoded)
```

- **Avantage** : Évite d'introduire un ordre implicite entre les catégories.
- **Limite** : Peut augmenter la dimensionnalité du dataset, ce qui peut ralentir l'entraînement.

Encodage des Variables Ordinales

- Les variables ordinales ont un ordre naturel (ex : `Petit < Moyen < Grand`).
- **Approche** : On assigne des valeurs numériques reflétant cet ordre.

```python
import pandas as pd
# Exemple de dataset
df = pd.DataFrame({'Taille': ['Petit', 'Grand', 'Moyen', 'Petit', 'Grand']})
# Definir l'ordre des categories
categories = {'Petit': 1, 'Moyen': 2, 'Grand': 3}
# Appliquer l'encodage ordinal
df['Taille_encoded'] = df['Taille'].map(categories)

print(df)
```

Conclusion sur la Gestion des Variables Qualitatives

- La gestion des variables qualitatives est cruciale pour rendre les données compatibles avec les modèles d'apprentissage.
- Les méthodes principales incluent **Label Encoding**, **One-Hot Encoding** et l'encodage ordinal.
- Le choix de la méthode dépend de la nature des données et du type de modèle utilisé.

Conclusion

- L'exploration et la visualisation des données permettent de mieux comprendre la structure des données.
- Les principales techniques incluent les histogrammes, les boxplots, les scatter plots et les matrices de corrélation.
- Ces techniques facilitent la détection des anomalies, la compréhension des relations et la prise de décisions sur le prétraitement.

Recherche de Corrélations

Recherche de Corrélations

Recherche de Corrélations

Objectifs du chapitre :

1. Matrice de corrélation pour identifier les relations entre les variables

Introduction à la Recherche de Corrélations

- La recherche de corrélations vise à détecter des relations linéaires entre les variables du dataset.
- **Objectif** : Identifier les variables fortement corrélées pour éviter la redondance et la multicolinéarité.
- **Définition** : Le coefficient de corrélation mesure la force et la direction de la relation linéaire entre deux variables.
- **Types de corrélation** :
  - **Positive** : Lorsque l'une augmente, l'autre augmente aussi.
  - **Négative** : Lorsque l'une augmente, l'autre diminue.
  - **Nulle** : Aucune relation linéaire entre les deux variables.

Applications de la Recherche de Corrélation

- **Réduction de la redondance** : Éliminer les variables fortement corrélées pour réduire la multicolinéarité.
- **Sélection de variables** : Identifier les variables les plus importantes à utiliser dans le modèle.
- **Détection de relations inattendues** : Découvrir des corrélations inattendues dans les données.
- **Prévention de la multicolinéarité** : Éviter que deux variables très corrélées ne soient incluses dans un modèle linéaire.

Conclusion sur la Recherche de Corrélations

- La recherche de corrélations aide à comprendre les relations linéaires entre les variables.
- La matrice de corrélation est un outil essentiel pour détecter les variables redondantes.
- Les coefficients de corrélation permettent d'optimiser la sélection des caractéristiques et d'éviter la multicolinéarité.

Conclusion

- L'exploration et la visualisation des données permettent de mieux comprendre la structure des données.
- Les principales techniques incluent les histogrammes, les boxplots, les scatter plots et les matrices de corrélation.
- Ces techniques facilitent la détection des anomalies, la compréhension des relations et la prise de décisions sur le prétraitement.

Expérimenter avec des combinaisons de variables

Expérimenter avec des combinaisons de variables

Objectifs du chapitre :

1. Feature Engineering, PCA (Principal Component Analysis), réduction de la dimensionnalité.

Introduction à l'Expérimentation avec des Combinaisons de Variables

- L'expérimentation avec des combinaisons de variables consiste à créer de nouvelles variables ou fonctionnalités (features) en combinant les variables existantes.
- **Objectif** : Améliorer la performance des modèles d'apprentissage en ajoutant de la complexité ou en mettant en évidence des relations non linéaires.
- **Techniques courantes** :
  - Création de **nouvelles interactions** (produits de deux variables).
  - Création de **fonctions mathématiques** (logarithme, exponentielle, racine carrée, etc.).
  - **Regroupement des classes** ou **discrétisation** des variables continues.

Création d'Interactions de Variables

- **Définition** : L'interaction entre deux ou plusieurs variables est obtenue en calculant le produit de ces variables.
- **Objectif** : Introduire des relations non linéaires entre les variables.
- **Exemple** : `Revenu annuel * Age` peut capter des interactions entre l'âge et les revenus.

```python
import pandas as pd
# Exemple de dataset
df = pd.DataFrame({
    'Age': [23, 45, 31, 35, 62],
    'Revenu_Annuel': [25000, 50000, 30000, 40000, 70000]
})
# Creer une nouvelle variable basee sur l'interaction
df['Age_Revenu'] = df['Age'] * df['Revenu_Annuel']
print(df)
```

Application de Fonctions Mathématiques

- **Objectif** : Capturer des relations non linéaires entre les variables et la cible.
- **Méthodes courantes** :
  - Logarithme (`log(x)`) pour gérer les distributions asymétriques.
  - Racine carrée (`sqrt(x)`) pour réduire l'effet des grandes valeurs.
  - Exponentielle (`exp(x)`) pour amplifier les petites valeurs.

```python
import numpy as np
# Ajouter de nouvelles fonctionnalites mathematiques
df['Log_Age'] = np.log(df['Age'] + 1) # log(x + 1) pour eviter log(0)
df['Sqrt_Revenu'] = np.sqrt(df['Revenu_Annuel'])
print(df)
```

Transformation Logarithmique

Pourquoi utiliser le logarithme ?

- Réduction de l'effet des outliers (valeurs extrêmes).
- Stabilisation de la variance.
- Rendre les distributions asymétriques plus "normales".
- Facilite l'entraînement des modèles linéaires et des réseaux de neurones.
- Formule courante : $\log(x + 1)$ (pour éviter les erreurs de $\log(0)$).

Pourquoi utiliser la racine carrée ?

- Réduction de l'effet des valeurs extrêmes, mais moins agressive que le logarithme.
- Stabilisation de la variance, utile pour les distributions de type Poisson.
- Permet de réduire l'asymétrie des distributions.
- Appliqué dans le traitement des valeurs positives.
- Formule courante : $\sqrt{x}$.

Pourquoi utiliser la transformation exponentielle ?

- Inverse de la transformation logarithmique.
- Utilisée pour ramener des variables normalisées à leur échelle d'origine.

Discrétisation des Variables

- **Définition** : Transformer des variables continues en variables catégoriques (binning).
- **Objectif** : Réduire la complexité et faciliter l'interprétation.
- **Exemple** : Transformer l'âge en catégories `Jeune, Adulte, Senior`.

```
# Creer des categories basees sur des intervalles d'âge
bins = [0, 30, 50, 100]
labels = ['Jeune', 'Adulte', 'Senior']
df['Categorie_Age'] = pd.cut(df['Age'], bins=bins, labels=labels)
print(df)
```

Conclusion sur l'Expérimentation avec des Combinaisons de Variables

- L'expérimentation avec des combinaisons de variables permet de capturer des relations non linéaires et de renforcer les performances des modèles d'apprentissage.
- Les méthodes incluent :
  - Les interactions de variables (produits de deux ou plusieurs colonnes).
  - L'application de transformations mathématiques (log, sqrt, exp).
  - La discrétisation des variables continues.
- Cette étape permet d'enrichir l'ensemble des fonctionnalités (features) et d'améliorer les prédictions.

Conclusion

- L'exploration et la visualisation des données permettent de mieux comprendre la structure des données.
- Les principales techniques incluent les histogrammes, les boxplots, les scatter plots et les matrices de corrélation.
- Ces techniques facilitent la détection des anomalies, la compréhension des relations et la prise de décisions sur le prétraitement.

Séparer les jeux d'entraînement, de validation et de test

Séparer les jeux d'entraînement, de validation et de test

Objectifs du chapitre :

1. Création des ensembles pour l'entraînement et l'évaluation.

Introduction à la Séparation des Jeux de Données

- La séparation des données en trois ensembles distincts (entraînement, validation et test) est essentielle pour évaluer la performance des modèles d'apprentissage.
- **Objectifs de chaque jeu de données** :
  - **Jeu d'entraînement** : Utilisé pour ajuster les paramètres du modèle.
  - **Jeu de validation** : Utilisé pour ajuster les hyperparamètres et éviter le surapprentissage (overfitting).
  - **Jeu de test** : Utilisé pour évaluer la performance finale du modèle sur des données non vues.

Répartition des Données

- La répartition typique des données est la suivante :
  - **70%** des données pour l'entraînement.
  - **15%** des données pour la validation.
  - **15%** des données pour le test.
- Cette répartition peut varier en fonction de la taille du dataset.
- L'objectif est de garantir que le modèle généralise bien sur des données non vues.

Méthodes de Séparation des Données

- **Séparation simple** : Diviser les données de manière fixe.
- **K-Fold Cross-Validation** : Les données sont divisées en K sous-ensembles.
- **Stratification** : La distribution des classes est respectée dans chaque sous-ensemble.

Exemple de Séparation des Données

```
from sklearn.model_selection import train_test_split
# Exemple de dataset
import pandas as pd
df = pd.DataFrame({
    'Age': [23, 45, 31, 35, 62, 45, 25, 42, 57, 36],
    'Revenu': [25000, 50000, 30000, 40000, 70000, 45000, 28000,
               42000, 65000, 39000],
    'Cible': [0, 1, 0, 1, 1, 0, 0, 1, 1, 0]
})
# Diviser les donnees en jeux d'entrainement et de test
X = df[['Age', 'Revenu']]
y = df['Cible']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
print("Nombre d'exemples dans X_train :", len(X_train))
print("Nombre d'exemples dans X_test :", len(X_test))
```

Utilisation de la Validation Croisée

- La validation croisée permet de diviser les données en K sous-ensembles.
- Chaque sous-ensemble est utilisé une fois comme test, et les autres sont utilisés pour l'entraînement.
- Cela réduit le risque de dépendance à un seul jeu de validation.

```python
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
# Exemple de dataset
X = df[['Age', 'Revenu']]
y = df['Cible']
# Initialiser un modele
clf = RandomForestClassifier(random_state=42)
# Appliquer la validation croisee avec 5 folds
scores = cross_val_score(clf, X, y, cv=5)
print("Scores de validation croisee :", scores)
```

Utilisation de la Validation Croisée

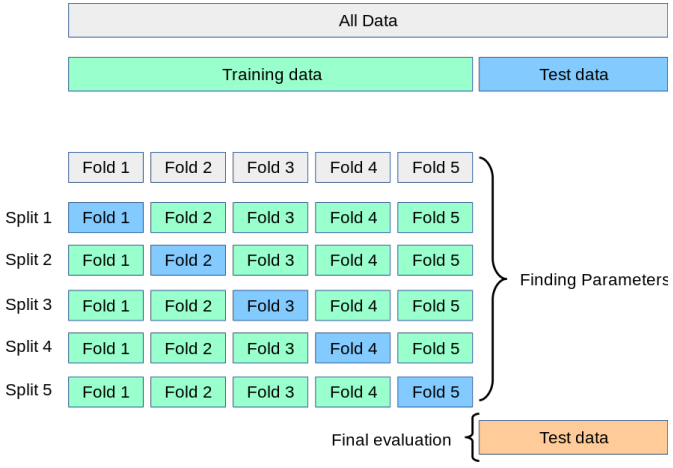


Figure: Source web

Conclusion sur la Séparation des Jeux de Données

- La séparation des données est essentielle pour éviter le surapprentissage (overfitting).
- Les jeux d'entraînement, de validation et de test permettent de tester la performance réelle du modèle.
- L'utilisation de la validation croisée renforce la robustesse de l'évaluation des modèles.

Conclusion

- L'exploration et la visualisation des données permettent de mieux comprendre la structure des données.
- Les principales techniques incluent les histogrammes, les boxplots, les scatter plots et les matrices de corrélation.
- Ces techniques facilitent la détection des anomalies, la compréhension des relations et la prise de décisions sur le prétraitement.

Conclusion

1. La qualité des données détermine la performance des modèles.
2. Chaque type de donnée demande des traitements spécifiques.
3. La préparation des données est essentielle pour assurer la qualité des entrées dans un modèle d'apprentissage.
4. Ces étapes permettent d'éviter les erreurs, de garantir la cohérence des variables et d'améliorer la précision des prédictions.
5. La séparation des ensembles d'entraînement, de validation et de test garantit une évaluation fiable des performances.
6. Respecter l'éthique et le RGPD est fondamental.