

# COURS DE SCIENCE DES DONNÉES ## École Nationale de Commerce et de Gestion (ENCG) - 4ème Année --- # PARTIE 1 : FONDAMENTAUX DE LA SCIENCE DES DONNÉES --- ## MODULE 1 : INTRODUCTION À LA SCIENCE DES DONNÉES ET AU BIG DATA ### 1.1 Définition et Contexte La **science des données** (Data Science) est un domaine interdisciplinaire qui combine les statistiques, l'informatique et l'expertise métier pour extraire des connaissances et des insights à partir de données structurées et non structurées.

**Pourquoi la science des données est-elle cruciale en entreprise ?** - **Prise de décision éclairée** : Remplacer l'intuition par des analyses basées sur les données - **Personnalisation client** : Adapter les offres et services aux besoins individuels - **Optimisation opérationnelle** : Réduire les coûts, améliorer l'efficacité - **Avantage concurrentiel** : Anticiper les tendances du marché - **Innovation** : Développer de nouveaux produits et services

**Exemples d'applications en entreprise** : - **Marketing** : Segmentation client, prédiction du churn, recommandations produits - **Finance** : Détection de fraude, scoring de crédit, prévision de revenus - **Logistique** : Optimisation de routes, gestion des stocks - **Ressources Humaines** : Prédiction du turnover, recrutement intelligent ### 1.2 Le Cycle de Vie des Données Le processus de science des données suit généralement un cycle itératif : 1. **Compréhension du problème métier** - Définir les objectifs et les KPIs - Identifier les parties prenantes 2. **Collecte des données** - Sources internes (bases de données, CRM, ERP) - Sources externes (API, web scraping, données publiques) 3. **Exploration et nettoyage des données (Data Wrangling)** - Traiter les valeurs manquantes - Détecter et gérer les outliers - Transformer et normaliser les données 4. **Analyse exploratoire (EDA - Exploratory Data Analysis)** - Statistiques descriptives - Visualisations - Identification de patterns 5. **Modélisation** - Sélection des algorithmes appropriés - Entraînement des modèles - Optimisation des hyperparamètres 6. **Évaluation** - Mesure de la performance - Validation croisée - Tests sur données de production 7. **Déploiement et monitoring** - Mise en production - Suivi des performances - Maintenance et mises à jour ### 1.3 Types de Données **Données structurées** : - Format tabulaire (lignes et colonnes) - Bases de données relationnelles (SQL) - Fichiers CSV, Excel - Facilement analysables **Données semi-structurées** : - JSON, XML - Logs système - Structure flexible **Données non structurées** : - Textes libres (emails, documents) - Images, vidéos, audio - Publications sur réseaux sociaux - Nécessitent des techniques spécifiques (NLP, Computer Vision) ### 1.4 Introduction au Big Data Le **Big Data** se caractérise par les **5 V** : 1. **Volume** : Quantité massive de données (téraoctets, pétaoctets) 2. **Vélocité** : Vitesse de génération et de traitement des données 3. **Variété** : Diversité des formats et sources de données 4. **Véracité** : Qualité et fiabilité des données 5. **Valeur** : Capacité à extraire des insights actionnables **Technologies Big Data** : - **Hadoop** : Framework de traitement distribué - **Spark** : Moteur de traitement rapide en mémoire - **NoSQL** : Bases de données non relationnelles (MongoDB, Cassandra, Redis)

**Différences SQL vs NoSQL** :

Aspect	SQL	NoSQL
Structure	Schéma fixe	Schéma flexible
Scalabilité	Verticale	Horizontale
Transactions	ACID	BASE
Cas d'usage	Données structurées	Big Data, données non structurées

### 1.5 Environnement Python pour la Data Science **Installation des bibliothèques essentielles** :

```
python # Installation via pip
pip install pandas numpy matplotlib seaborn scikit-learn jupyter #
Ou via conda
conda install pandas numpy matplotlib seaborn scikit-learn jupyter
```

**Structure d'un projet Data Science** :

```

projet_data_science/
├── data/
│   ├── raw/ # Données brutes
│   ├── processed/ # Données nettoyées
│   └── external/ # Données externes
├── notebooks/ # Jupyter notebooks pour exploration
├── src/ # Code source
├── data/ # Scripts de traitement des données
├── features/ # Feature engineering
├── models/ # Modèles ML
├── visualization/ # Visualisations
├── models/ # Modèles entraînés
├── reports/ # Rapports et analyses
└── requirements.txt # Dépendances
```

--- ## MODULE 2 : STATISTIQUES POUR LA SCIENCE DES DONNÉES ### 2.1 Statistiques Descriptives Les statistiques descriptives résument et décrivent les principales caractéristiques d'un ensemble de données.

**Mesures de tendance centrale** :

- Moyenne** :  $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$
- Médiane** : Valeur centrale qui divise les données en deux parties égales
- Mode** : Valeur la plus fréquente

**Mesures de dispersion** :

- Variance** :  $s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$
- Écart-type** :  $s = \sqrt{s^2}$
- Étendue** : Maximum - Minimum
- Écart interquartile (IQR)** :  $Q3 - Q1$

**Code Python** -

```

Statistiques descriptives : ** ```python import pandas as pd import numpy as np import
matplotlib.pyplot as plt import seaborn as sns # Charger un dataset en ligne (exemple : ventes
d'une entreprise) url = "https://raw.githubusercontent.com/datasets/gdp/master/data/gdp.csv" df
= pd.read_csv(url) # Afficher les premières lignes print(df.head()) # Informations sur le dataset
print(df.info()) # Statistiques descriptives print(df.describe()) # Statistiques pour une colonne
spécifique colonne = df['Value'] # Adapter selon vos données print(f'Moyenne:
{colonne.mean():.2f}') print(f'Médiane: {colonne.median():.2f}') print(f'Mode: {colonne.mode()
[0]:.2f}') print(f'Écart-type: {colonne.std():.2f}') print(f'Variance: {colonne.var():.2f}') print(f'Min:
{colonne.min():.2f}') print(f'Max: {colonne.max():.2f}') # Quartiles print(f'Q1 (25%):
{colonne.quantile(0.25):.2f}') print(f'Q2 (50% - Médiane): {colonne.quantile(0.50):.2f}') print(f'Q3
(75%): {colonne.quantile(0.75):.2f}') print(f'IQR: {colonne.quantile(0.75) -
colonne.quantile(0.25):.2f}') ```

### 2.2 Visualisation des Statistiques Descriptives ```python #
Configuration du style sns.set_style("whitegrid") plt.rcParams['figure.figsize'] = (12, 8) # Créer
une figure avec plusieurs sous-graphiques fig, axes = plt.subplots(2, 2, figsize=(14, 10)) # 1.
Histogramme axes[0, 0].hist(colonne, bins=30, edgecolor='black', alpha=0.7) axes[0,
0].set_title('Distribution des valeurs', fontsize=14, fontweight='bold') axes[0,
0].set_xlabel('Valeur') axes[0, 0].set_ylabel('Fréquence') # Ajouter la moyenne et la médiane
axes[0, 0].axvline(colonne.mean(), color='red', linestyle='--', label=f'Moyenne:
{colonne.mean():.2f}') axes[0, 0].axvline(colonne.median(), color='green', linestyle='--',
label=f'Médiane: {colonne.median():.2f}') axes[0, 0].legend() # 2. Boxplot axes[0,
1].boxplot(colonne.dropna(), vert=True) axes[0, 1].set_title('Boxplot - Détection des outliers',
fontsize=14, fontweight='bold') axes[0, 1].set_ylabel('Valeur') # 3. Densité
colonne.plot(kind='density', ax=axes[1, 0]) axes[1, 0].set_title('Courbe de densité', fontsize=14,
fontweight='bold') axes[1, 0].set_xlabel('Valeur') # 4. QQ-plot (pour tester la normalité) from
scipy import stats stats.probplot(colonne.dropna(), dist="norm", plot=axes[1, 1]) axes[1,
1].set_title('Q-Q Plot (Test de normalité)', fontsize=14, fontweight='bold') plt.tight_layout()
plt.show() ```

### 2.3 Statistiques Inférentielles Les statistiques inférentielles permettent de faire
des inférences sur une population à partir d'un échantillon.

**Concepts clés : ** - **Population** : Ensemble complet des individus/éléments étudiés -
**Échantillon** : Sous-ensemble de la population - **Paramètre** : Caractéristique de la population ( $\mu$ ,  $\sigma$ ) - **Statistique** :
Caractéristique de l'échantillon ( $\bar{x}$ , s)

**Intervalles de confiance : ** Un intervalle de
confiance à 95% pour la moyenne :  $\bar{x} \pm t_{\alpha/2} \times \frac{s}{\sqrt{n}}$  Où : -
 $\bar{x}$  : moyenne de l'échantillon -  $t_{\alpha/2}$  : valeur critique de la distribution t de
Student - s : écart-type de l'échantillon - n : taille de l'échantillon

**Code Python - Intervalle
de confiance : ** ```python from scipy import stats def intervalle_confiance(data, confiance=0.95):
""" Calcule l'intervalle de confiance pour la moyenne """ n = len(data) moyenne = np.mean(data)
ecart_type = np.std(data, ddof=1) # ddof=1 pour échantillon erreur_standard = ecart_type /
np.sqrt(n) # Valeur critique de la distribution t t_critique = stats.t.ppf((1 + confiance) / 2, n - 1)
marge_erreur = t_critique * erreur_standard ic_inf = moyenne - marge_erreur ic_sup = moyenne
+ marge_erreur return moyenne, ic_inf, ic_sup # Exemple d'utilisation moyenne, ic_inf, ic_sup =
intervalle_confiance(colonne.dropna()) print(f'Moyenne: {moyenne:.2f}') print(f'Intervalle de
confiance à 95%: [{ic_inf:.2f}, {ic_sup:.2f}]) ```

### 2.4 Tests d'Hypothèses Les tests
d'hypothèses permettent de prendre des décisions statistiques sur une population.

**Structure
d'un test d'hypothèse : ** 1.  $H_0$  (Hypothèse nulle) : Assertion à tester 2.  $H_1$  (Hypothèse
alternative) : Ce qu'on suspecte être vrai 3. Niveau de signification ( $\alpha$ ) : Généralement 0.05
4. Statistique de test : Valeur calculée à partir des données 5. p-value : Probabilité
d'observer les données si  $H_0$  est vraie 6. Décision : Rejeter  $H_0$  si p-value <  $\alpha$ 

**Principaux
tests : **

**Test t de Student (comparaison de moyennes) : ** ```python from scipy.stats import
ttest_1samp, ttest_ind, ttest_rel # Test t pour un échantillon (comparer à une valeur) # H0: La
moyenne de la population = valeur_ref valeur_ref = 1000 t_stat, p_value =
ttest_1samp(colonne.dropna(), valeur_ref) print(f'Test t (un échantillon):') print(f'Statistique t:
{t_stat:.4f}') print(f'p-value: {p_value:.4f}') if p_value < 0.05: print("Rejet de H0: La moyenne est
significativement différente de", valeur_ref) else: print("On ne peut pas rejeter H0")

# Test t pour
deux échantillons indépendants # Exemple: Comparer les ventes de deux régions echantillon1
= np.random.normal(1000, 200, 100) echantillon2 = np.random.normal(1100, 200, 100) t_stat,
p_value = ttest_ind(echantillon1, echantillon2) print(f'\nTest t (deux échantillons
indépendants):') print(f'Statistique t: {t_stat:.4f}') print(f'p-value: {p_value:.4f}') ```

**Test du Chi-

```

```

carré (variables catégorielles) :** ```python from scipy.stats import chi2_contingency # Exemple:
Relation entre genre et préférence produit # Créer un tableau de contingence data = { 'Homme':
[120, 80, 50], 'Femme': [90, 110, 70] } tableau = pd.DataFrame(data, index=['Produit A', 'Produit
B', 'Produit C']) print("Tableau de contingence:") print(tableau) # Test du Chi-carré chi2, p_value,
dof, expected = chi2_contingency(tableau) print(f"\nTest du Chi-carré:") print(f"Chi2: {chi2:.4f}")
print(f"p-value: {p_value:.4f}") print(f"Degrés de liberté: {dof}") if p_value < 0.05: print("Il existe
une relation significative entre genre et préférence") else: print("Pas de relation significative") ```
### 2.5 Corrélation et Régression **Corrélation : ** La corrélation mesure la relation linéaire
entre deux variables. **Coefficient de corrélation de Pearson : ** 
$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

$ - $r$ ∈ [-1, 1] $ - $r = 1$ :
corrélation positive parfaite - $r = -1$ : corrélation négative parfaite - $r = 0$ : pas de corrélation
linéaire ```python # Charger un dataset avec plusieurs variables url =
"https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv" df_iris =
pd.read_csv(url) # Matrice de corrélation correlation_matrix = df_iris.corr() print("Matrice de
corrélation:") print(correlation_matrix) # Visualisation de la matrice de corrélation
plt.figure(figsize=(10, 8)) sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
center=0, square=True, linewidths=1) plt.title('Matrice de Corrélation', fontsize=16,
fontweight='bold') plt.tight_layout() plt.show() # Nuage de points avec régression
plt.figure(figsize=(10, 6)) sns.regplot(x='sepal_length', y='sepal_width', data=df_iris,
scatter_kws={'alpha':0.5}) plt.title('Relation entre longueur et largeur du sépale', fontsize=14,
fontweight='bold') plt.xlabel('Longueur du sépale (cm)') plt.ylabel('Largeur du sépale (cm)')
plt.show() # Calculer le coefficient de corrélation from scipy.stats import pearsonr r, p_value =
pearsonr(df_iris['sepal_length'], df_iris['sepal_width']) print(f"\nCoefficient de corrélation de
Pearson: {r:.4f}") print(f"p-value: {p_value:.4f}") ``` --- ## TRAVAUX PRATIQUES 1 : ANALYSE
EXPLORATOIRE D'UN DATASET D'ENTREPRISE ### Objectif Réaliser une analyse
exploratoire complète d'un dataset de ventes d'une entreprise en ligne, en appliquant les
concepts statistiques vus précédemment. ### Dataset Nous utiliserons un dataset de ventes en
ligne disponible publiquement. ```python import pandas as pd import numpy as np import
matplotlib.pyplot as plt import seaborn as sns from scipy import stats # Charger le dataset url =
"https://raw.githubusercontent.com/plotly/datasets/master/26k-consumer-goods.csv" df_ventes =
pd.read_csv(url) print("=" * 80) print("ANALYSE EXPLORATOIRE DES DONNÉES - VENTES E-
COMMERCE") print("=" * 80) # 1. EXPLORATION INITIALE print("\n1. APERÇU DES
DONNÉES") print("-" * 80) print(f"Nombre de lignes: {len(df_ventes)}") print(f"Nombre de
colonnes: {len(df_ventes.columns)}") print(f"\nPremières lignes:") print(df_ventes.head())
print(f"\nInformations sur les colonnes:") print(df_ventes.info()) # 2. TRAITEMENT DES
VALEURS MANQUANTES print("\n2. VALEURS MANQUANTES") print("-" * 80)
valeurs_manquantes = df_ventes.isnull().sum() pourcentage_manquant = (valeurs_manquantes
/ len(df_ventes)) * 100 missing_df = pd.DataFrame({ 'Valeurs manquantes':
valeurs_manquantes, 'Pourcentage': pourcentage_manquant })
print(missing_df[missing_df["Valeurs manquantes"] > 0]) # 3. STATISTIQUES DESCRIPTIVES
print("\n3. STATISTIQUES DESCRIPTIVES") print("-" * 80) print(df_ventes.describe()) # 4.
ANALYSE PAR CATÉGORIE if 'Category' in df_ventes.columns: print("\n4. RÉPARTITION PAR
CATÉGORIE") print("-" * 80) print(df_ventes['Category'].value_counts()) # 5. VISUALISATIONS
fig, axes = plt.subplots(2, 2, figsize=(16, 12)) fig.suptitle('Analyse Exploratoire des Ventes',
fontsize=16, fontweight='bold') # Histogramme des prix (si colonne existe) if 'Price' in
df_ventes.columns: axes[0, 0].hist(df_ventes['Price'].dropna(), bins=50, edgecolor='black',
alpha=0.7) axes[0, 0].set_title('Distribution des Prix') axes[0, 0].set_xlabel('Prix') axes[0,
0].set_ylabel('Fréquence') axes[0, 0].axvline(df_ventes['Price'].mean(), color='red', linestyle='--',
label=f"Moyenne: {df_ventes['Price'].mean():.2f}") axes[0, 0].legend() # Boxplot par catégorie if
'Category' in df_ventes.columns and 'Price' in df_ventes.columns:
df_ventes.boxplot(column='Price', by='Category', ax=axes[0, 1]) axes[0, 1].set_title('Prix par
Catégorie') axes[0, 1].set_xlabel('Catégorie') axes[0, 1].set_ylabel('Prix') # Top 10 des
catégories/produits if 'Category' in df_ventes.columns: top_categories =
df_ventes['Category'].value_counts().head(10) axes[1, 0].barh(range(len(top_categories)),
top_categories.values) axes[1, 0].set_yticks(range(len(top_categories))) axes[1,
0].set_yticklabels(top_categories.index) axes[1, 0].set_title('Top 10 des Catégories') axes[1,
0].set_xlabel('Nombre de produits') axes[1, 0].invert_yaxis() # Matrice de corrélation pour

```

```

colonnes numériques numeric_cols = df_ventes.select_dtypes(include=[np.number]).columns
if len(numeric_cols) > 1: corr_matrix = df_ventes[numeric_cols].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0, ax=axes[1, 1], square=True)
axes[1, 1].set_title('Matrice de Corrélation')
plt.tight_layout()
plt.show()
print("\n" + "=" * 80)
print("ANALYSE TERMINÉE")
print("=" * 80)
```

```

**#### Exercices pratiques**  
**\*\*Exercice 1 :\*\***  
 Identifier les outliers dans les prix et proposer une stratégie de traitement.

**\*\*Exercice 2 :\*\*** Tester l'hypothèse : "Le prix moyen est différent selon les catégories" (utiliser ANOVA).

**\*\*Exercice 3 :\*\*** Créer un rapport exécutif de 2 pages présentant les insights clés découverts.

--- **\*Suite du cours dans la Partie 2 : Apprentissage Automatique et Applications Avancées\***