Informations sur l'étudiant:

- Nom :
- Prénom(s) :
- Classe :
- Numéro :

Analyse de données avec Régression Linéaire

Jeu de données : California Housing (prix des logements en Californie)

Ce notebook présente un workflow complet d'analyse de données en utilisant un modèle de **régression linéaire** pour prédire le **prix médian des logements** en Californie à partir de caractéristiques géographiques et démographiques.

1. Chargement et inspection des données

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Configuration graphique
sns.set_style("whitegrid")
plt.rcParams['figure.figsize'] = (10, 6)

print("Bibliothèques importées avec succès")
```

```
# Charger le jeu de données California Housing
housing = fetch_california_housing(as_frame=True)
df = housing.frame

print(f"Dimensions du dataset : {df.shape[0]} lignes × {df.shape[1]} colonnes")
print(f"\nVariable cible : MedHouseVal (valeur médiane du logement en centaines de milliers de $)")
print(f"\nDescription des variables :")
print("-" * 60)
descriptions = {
    'MedInc': 'Revenu médian du quartier',
    'HouseAge': 'Âge médian des logements',
    'AveRooms': 'Nombre moyen de pièces par logement',
    'AveBedrms': 'Nombre moyen de chambres par logement',
    'Population': 'Population du quartier',
    'AveOccup': 'Nombre moyen d\'occupants par logement',
    'Latitude': 'Latitude géographique',
    'Longitude': 'Longitude géographique',
    'MedHouseVal': 'Prix médian du logement (cible)'
}
for col, desc in descriptions.items():
    print(f"  {col:15s} → {desc}")
```

```
# Afficher les premières lignes avec .head()
# pour avoir un aperçu visuel des données
```

```
# Utiliser .info() pour vérifier les types de colonnes
# et détecter les éventuelles valeurs manquantes
```

```
# Utiliser .describe() pour obtenir les statistiques descriptives :
# moyenne, écart-type, min, max, quartiles de chaque colonne
```

2. Nettoyage des données

```
# Vérifier les valeurs manquantes avec .isnull().sum()
# Si présentes : soit les supprimer (.dropna), soit les imputer (.fillna)
```

```
# Vérifier et supprimer les doublons avec .duplicated() et .drop_duplicates()
```

```
# Détecter les valeurs aberrantes (outliers) à l'aide de boxplots
# Les boxplots montrent visuellement la distribution et les points extrêmes
```

```
# Filtrer les outliers extrêmes pour AveRooms, AveBedrms et AveOccup
# On fixe des seuils raisonnables (ex: AveRooms < 50, AveOccup < 20)
# pour éviter que ces valeurs extrêmes faussent le modèle
```

3. Analyse exploratoire (EDA)

```
# Tracer la distribution de la variable cible (MedHouseVal)
# avec un histogramme + courbe de densité (kde)
# et un boxplot pour repérer la médiane et les extrêmes
```

```
# Calculer et afficher la matrice de corrélation sous forme de heatmap
# Cela permet d'identifier quelles variables sont le plus liées au prix :
#   - corrélation proche de +1 → relation positive forte
#   - corrélation proche de -1 → relation négative forte
#   - corrélation proche de  0 → pas de relation linéaire
```

```
# Scatter plots : croiser les variables les plus corrélées avec la cible
# Ex : MedInc vs MedHouseVal → on s'attend à une relation positive forte
# (plus le revenu est élevé, plus le logement est cher)
```

```
# Carte géographique : afficher chaque quartier (Latitude, Longitude)
# coloré selon le prix médian, pour visualiser la répartition spatiale
# des prix en Californie (zones côtières = plus cher)
```

4. Ingénierie des variables

```
# Créer de nouvelles variables dérivées, par exemple :
#   PiecesParChambre = AveRooms / AveBedrms  (ratio pièces/chambres)
#   PopParLogement   = Population / AveOccup  (estimation du nb de logements)
# Ces nouvelles features peuvent capturer des informations supplémentaires
```

```
# Séparer les données en :
#   X = variables explicatives (features) → les 10 colonnes d'entrée
#   y = variable cible → MedHouseVal (le prix à prédire)
```

5. Séparation entraînement / test

```
# Diviser X et y en deux sous-ensembles avec train_test_split :
#   - 80% pour l'entraînement (le modèle apprend sur ces données)
#   - 20% pour le test (on évalue la qualité des prédictions)
# Le paramètre random_state=42 garantit la reproductibilité
```

6. Construction du modèle de régression linéaire

Le modèle cherche à trouver la relation :

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

où β_0 est l'ordonnée à l'origine et β_i sont les coefficients de chaque variable.

```
# Instancier LinearRegression() et entraîner avec .fit(X_train, y_train)
# Le modèle calcule les coefficients β optimaux en minimisant
# la somme des carrés des erreurs (méthode des moindres carrés)

# Afficher :
#   - model.intercept_ → l'ordonnée à l'origine (β₀)
#   - model.coef_ → les coefficients (β₁, β₂, ..., βₙ)
# Un coefficient positif = la variable augmente le prix
# Un coefficient négatif = la variable diminue le prix
```

7. Prédiction et évaluation

```
# Prédire les prix sur l'ensemble de test avec model.predict(X_test)
# Puis calculer les métriques d'évaluation :
#
#   MSE  = moyenne des (réel - prédit)²     → erreur quadratique moyenne
#   RMSE = √MSE                             → plus interprétable (même unité que y)
#   MAE  = moyenne des |réel - prédit|       → erreur absolue moyenne
#   R²   = 1 - (SS_res / SS_tot)            → proportion de variance expliquée
#
# R² = 1.0 → prédiction parfaite
# R² = 0.0 → le modèle ne fait pas mieux que la moyenne
```

8. Visualisation des résultats

```
# Graphique 1 : Valeurs réelles vs Prédites
# Un scatter plot avec la ligne diagonale (prédiction parfaite)
# Plus les points sont proches de la diagonale, meilleur est le modèle
```

```
# Graphique 2 : Analyse des résidus (résidu = réel - prédit)
#   a) Histogramme des résidus → devrait suivre une distribution normale centrée sur 0
#   b) Résidus vs Prédictions → les points doivent être dispersés aléatoirement
#      (un pattern visible indiquerait que le modèle rate quelque chose)
```

```
# Graphique 3 : Barplot des coefficients du modèle
# Permet de visualiser l'importance relative de chaque variable :
#   🟢 Coefficient positif = contribue à augmenter le prix
#   🔴 Coefficient négatif = contribue à diminuer le prix
```

9. Export des résultats

```
# Créer un DataFrame de résultats contenant :
#   - les features du test
#   - le prix réel
#   - le prix prédit
#   - le résidu (écart)
#   - l'erreur en pourcentage
# Puis exporter en CSV avec .to_csv("resultats_regression.csv")
```

10. Analyse et recommandations

Double-click (or enter) to edit