Atelier sur la Régression Linéaire avec Python

Exercice 1 : Calculer la ligne de régression à partir de zéro

Comprendre les bases de la régression linéaire en calculant la pente (mm) et l'ordonnée à l'origine (bb) manuellement.

Tâche :

- Créer deux listes : X=[1,2,3,4,5] et Y=[2,4,5,4,5]
- Calculer m et b en utilisant les formules suivantes : m=Cov(X,Y)Var(X),b=Y¯−m⋅X¯m = \frac{\text{Cov}(X, Y)}{\text{Var}(X)}, \quad b = \bar{Y} - m \cdot \bar{X}
- Afficher l'équation de la ligne y=m⋅x+by = m \cdot x + b.

Illustration Python :

```
import numpy as np

# Données
X = np.array([1, 2, 3, 4, 5])
Y = np.array([2, 4, 5, 4, 5])

# Calculer les moyennes
mean_X = np.mean(X)
mean_Y = np.mean(Y)

# Calculer la pente (m)
numerator = np.sum((X - mean_X) * (Y - mean_Y))
denominator = np.sum((X - mean_X)**2)
m = numerator / denominator

# Calculer l'ordonnée à l'origine (b)
b = mean_Y - m * mean_X

print(f"L'équation de la ligne : y = {m:.2f} * x + {b:.2f}")
```

Exercice 2 : Visualiser la ligne de régression

Tracer les données et la ligne de régression pour mieux comprendre la relation.

Tâche :

- Réutiliser les valeurs de m et b obtenues dans l'exercice précédent.
- Utiliser **Matplotlib** pour tracer :
    - Les points des données (X,Y).
    - La ligne de régression.

Illustration Python :

```
import matplotlib.pyplot as plt

# Calculer les valeurs prédites
Y_pred = m * X + b

# Tracer les points et la ligne
plt.scatter(X, Y, color="blue", label="Données")
plt.plot(X, Y_pred, color="red", label="Ligne de régression")
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.title("Régression linéaire simple")
plt.show()
```

---

3. Prédire de nouvelles valeurs

Utiliser le modèle de régression pour faire des prédictions.

Tâche :

- Écrire une fonction Python qui prédit Y en fonction de X ($Y = m \cdot X + b$).
- Tester cette fonction avec les valeurs X=[6,7,8].

Code :

```
# Fonction de prédiction
def predict(x, m, b):
    return m * x + b

# Nouvelles prédictions
new_X = np.array([6, 7, 8])
predictions = predict(new_X, m, b)
print(f"Pour X = {new_X}, les prédictions Y sont : {predictions}")
```

Exercice 4 : Utiliser `scikit-learn` pour simplifier la régression

Découvrir une approche automatisée avec `LinearRegression` de Scikit-learn.

Tâche :

- Charger les données XX et YY dans Scikit-learn.
- Entraîner un modèle de régression linéaire.
- Afficher la pente (mm) et l'ordonnée à l'origine (bb) calculées par Scikit-learn.

Illustration Python:

```
from sklearn.linear_model import LinearRegression

# Mise en forme des données
X_reshaped = X.reshape(-1, 1)  # Scikit-learn attend des colonnes 2D

# Entraîner le modèle
model = LinearRegression()
model.fit(X_reshaped, Y)

# Extraire m et b
m_sklearn = model.coef_[0]
b_sklearn = model.intercept_

print(f"Scikit-learn - Pente : {m_sklearn:.2f}, Ordonnée à l'origine : {b_sklearn:.2f}")
```

Exercice 5 : Tester le modèle avec des données réelles

Objectif : Appliquer la régression linéaire à un jeu de données réel (comme les prix des maisons, etc.).

Tâche :

- Charger un dataset public (par exemple, le dataset Boston Housing de Scikit-learn).
- Sélectionner une relation simple (par exemple, X=X = nombre de pièces, Y=Y = prix des maisons).
- Entraîner le modèle et afficher les résultats.

Illustration Python :

```
from sklearn.datasets import load_boston

# Charger les données
boston = load_boston()
X = boston.data[:, 5]  # Nombre de pièces (RM)
Y = boston.target  # Prix des maisons

# Mise en forme
X = X.reshape(-1, 1)

# Entraîner le modèle
model = LinearRegression()
model.fit(X, Y)

# Afficher les résultats
m_boston = model.coef_[0]
b_boston = model.intercept_

print(f"Boston Housing - Pente : {m_boston:.2f}, Ordonnée à l'origine : {b_boston:.2f}")
```