

NOMS:
CLASSE:
EQUIPE:

TP1 : Régression Linéaire simple

L'objectif de ce TP est de vous faire maîtriser le **workflow complet** d'un projet de régression linéaire simple. Vous allez implémenter chaque étape essentielle, de l'importation des données à la prédiction finale.

Nous allons utiliser un jeu de données très simple et intuitif qui contient 30 observations sur l'expérience professionnelle et le salaire.

Voici le dictionnaire des données que nous allons utiliser :

Variable	Description	Type
Salary	Le salaire annuel de l'employé.	Numérique
YearsExperience	Le nombre d'années d'expérience professionnelle.	Numérique

Partie 1 — Téléchargement et chargement du dataset

```
import pandas as pd

# 1) dataset
fichier_local = "Salary_Data.csv"

# 2) Chargement
data = pd.read_csv(fichier_local)

# 3) Aperçu
print("Aperçu :")
print(data.head())

# 4) Informations
print("\nInfos :")
print(data.info())

# 5) Statistiques descriptives
print("\nDescribe :")
print(data.describe())
```

Question : Quelles sont les colonnes présentes ?
Que signifient-elles ?

Réponse : _____

Question : Combien d'observations contient le dataset ?

Réponse : _____

Partie 2 — Visualisation initiale

```
import matplotlib.pyplot as plt

plt.figure(figsize=(7,5))
plt.scatter(data['YearsExperience'],
            data['Salary'])
plt.title("Salaire en fonction des années
d'expérience")
plt.xlabel("Années d'expérience")
plt.ylabel("Salaire")
plt.grid(True)
plt.show()
```

Question : La relation semble-t-elle linéaire ?
Expliquez en une phrase.

Réponse :

Partie 3 — Préparation des données (train/test)

```
from sklearn.model_selection import
train_test_split

X = data[['YearsExperience']] # variable
explicative
y = data['Salary']           # variable cible

X_train, X_test, y_train, y_test =
train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("Taille train :", X_train.shape, " | Taille
test :", X_test.shape)
```

Question : Pourquoi séparer les
données en apprentissage et test ?

Réponse :

Partie 4 — Entraînement du modèle de régression linéaire

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)

print("Pente (a) :", model.coef_[0])
print("Ordonnée à l'origine (b) :",
model.intercept_)
```

Question : Interprétez les valeurs de a et
b dans le contexte du problème.

Réponse :

Partie 5 — Visualisation de la droite de régression (sur l'entraînement)

```
# Prédiction sur l'entraînement
y_pred_train = model.predict(X_train)

plt.figure(figsize=(7,5))
plt.scatter(X_train, y_train, label='Données
(train)')
plt.plot(X_train, y_pred_train, color='red',
linewidth=2, label='Droite de régression')
plt.title("Ajustement du modèle sur les données
d'entraînement")
plt.xlabel("Années d'expérience")
plt.ylabel("Salaire")
plt.legend()
plt.grid(True)
plt.show()
```

Question : La droite suit-elle bien les points ?
Que remarquez-vous ?

Réponse : _____

Partie 6 — Évaluation sur l'ensemble de test

```
from sklearn.metrics import
mean_squared_error, r2_score

y_pred_test = model.predict(X_test)
mse = mean_squared_error(y_test,
y_pred_test)
r2 = r2_score(y_test, y_pred_test)

print("MSE :", mse)
print("R² :", r2)
```

Question : Que signifie un R^2 proche de 1 ? Et s'il est faible ?

Réponse :

Question : Pourquoi évaluer sur l'ensemble de test plutôt
que sur l'entraînement ?

Réponse :

Partie 7 — Prédiction sur une nouvelle valeur

```
# Exemple : prédire le salaire pour
8.5 années d'expérience
experience_nouvelle = [[8.5]]
salaire_prevu =
model.predict(experience_nouvelle)
print(f"Salaire prédit pour 8.5 années
d'expérience :
{salaire_prevu[0]:.2f}")
```

Question : Donnez une phrase d'interprétation de cette
prédiction.

Réponse :

TP2 : Régression Linéaire Multiple

Ce TP marque la transition de la régression linéaire simple (une seule variable) vers la régression linéaire multiple (plusieurs variables).

L'objectif est d'étudier la relation entre le prix moyen des logements à Boston et *différents* facteurs socio-économiques simultanément. Nous allons utiliser un jeu de données classique en Machine Learning. Il contient des informations sur 506 quartiers de la région de Boston.

Voici le dictionnaire des données que nous allons utiliser :

Variable	Description	Type
MEDV	Valeur médiane des maisons occupées par leur propriétaire (en milliers de \$).	Numérique
CRIM	Taux de criminalité par habitant.	Numérique
ZN	Proportion de terrains résidentiels zonés pour de grands lots.	Numérique
INDUS	Proportion de la surface dédiée à des activités industrielles.	Numérique
CHAS	Variable binaire : 1 si le quartier longe la rivière Charles, 0 sinon.	Binaire
NOX	Concentration en oxyde d'azote (pollution de l'air).	Numérique
RM	Nombre moyen de pièces par logement.	Numérique
AGE	Proportion de logements construits avant 1940.	Numérique
DIS	Distance pondérée aux grands centres d'emploi.	Numérique
RAD	Indice d'accessibilité aux autoroutes principales.	Numérique
TAX	Taux de la taxe foncière.	Numérique
PTRATIO	Ratio élèves/enseignants dans les écoles de la ville.	Numérique
B	$1000(B_k - 0.63)^2$ où B_k est la proportion d'habitants noirs.	Numérique
LSTAT	Pourcentage de la population considérée comme de "statut inférieur".	Numérique

Partie 1 — Téléchargement et chargement du dataset

```
from sklearn.datasets import fetch_openml
import pandas as pd
# Chargement du dataset boston =
fetch_openml(name='boston', version=1,
as_frame=True)
data = boston.frame
# Affichage des premières lignes
print(data.head())
# Informations générales
print(data.info())
# Statistiques descriptives
print(data.describe())
```

1. Combien de variables explicatives contient le jeu de données ?-----

2. Quelle est la variable cible ? -----

3. Que fait la ligne de code ci-dessous ?

```
data = data.drop('B', axis=1)
-----
```

4. Expliquez la nécessité de la question 3.

```
-----
-----
```

Partie 2 — Exploration et visualisation des données

1. Afficher les statistiques descriptives.
2. Tracer des graphiques : histogramme de MEDV, RM vs MEDV, LSTAT vs MEDV.

Partie 3 — Préparation des données

1. Définir X comme toutes les colonnes sauf MEDV.
2. Définir y comme la colonne MEDV.
3. Diviser le jeu en apprentissage (80%) et test (20%).

Partie 4 — Construction du modèle

1. Créer un modèle LinearRegression.
2. L'entraîner sur l'ensemble d'apprentissage.
3. Afficher l'interception et les coefficients.

Partie 5 — Évaluation du modèle

1. Prédire les prix sur le jeu de test.
2. Calculer le MSE et le R^2 .
3. Interpréter les résultats.

Partie 6 — Visualisation des résultats

1. Tracer y_test vs y_pred avec la diagonale y=x.
2. Tracer les résidus