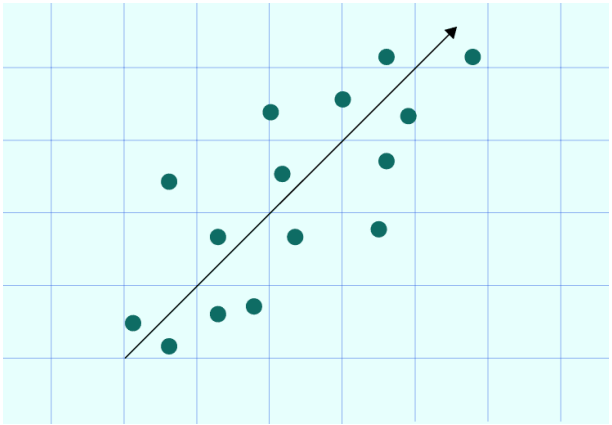
Université Abdelmalek Essaadi

Faculté des Sciences et Techniques de Tanger

Département Génie Informatique

Atelier 1 : Régression



Encadré par :
Prof. Lotfi ELAACHAK

Elaboré par :
ELHANSALI Mouaad

Cycle d'ingénieur Logiciels et Systèmes Intelligents
S3 - 2024/2025

Table des matières

1 Objectif du Projet

l'objective principal de cet atelier est de pratiquer les deux concepts de la régression : la régression linière simple et la régression linière multiple, en traitant des données de plusieurs Data Sets.

2 Data Visualisation

2.1 Exploration des données

— Expérience / Salaire et Assurance

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

df1=pd.read_csv('Salary_Data.csv')
df2=pd.read_csv('insurance.csv')
```

```
Dataset Experience / Salaire
   YearsExperience   Salary
0              1.1  39343.0
1              1.3  46205.0
2              1.5  37731.0
3              2.0  43525.0
4              2.2  39891.0
Description du dataset Experience / Salaire
       YearsExperience         Salary
count        30.000000      30.000000
mean          5.313333   76003.000000
std           2.837888   27414.429785
min           1.100000   37731.000000
25%           3.200000   56720.750000
50%           4.700000   65237.000000
75%           7.700000  100544.750000
max          10.500000  122391.000000

Dataset Assurance
   age     sex     bmi  children smoker     region      charges
0   19  female  27.900         0    yes  southwest  16884.92400
1   18    male  33.770         1     no  southeast   1725.55230
2   28    male  33.000         3     no  southeast   4449.46200
3   33    male  22.705         0     no  northwest  21984.47061
4   32    male  28.880         0     no  northwest   3866.85520

Description du dataset Assurance
                age          bmi     children       charges
count  1338.000000  1338.000000  1338.000000   1338.000000
mean     39.207025    30.663397     1.094918  13270.422265
std      14.049960     6.098187     1.205493  12110.011237
min      18.000000    15.960000     0.000000   1121.873900
25%      27.000000    26.296250     0.000000   4740.287150
50%      39.000000    30.400000     1.000000   9382.033000
75%      51.000000    34.693750     2.000000  16639.912515
max      64.000000    53.130000     5.000000  63770.428010
```

2.2 Résumé statistique et interprétation

```
# Resume statistique
print('Dataset Experience / Salaire')
print(df1.head())
print('Description du dataset Experience / Salaire')
print(df1.describe())

print('\nDataset Assurance')
print(df2.head())
print('\nDescription du dataset Assurance')
print(df2.describe())
```

- Expérience et Salaire : Le salaire moyen est de 76,003 pour une expérience moyenne de 5.31 ans. Les salaires varient beaucoup allant de 37,731 à 122,391.

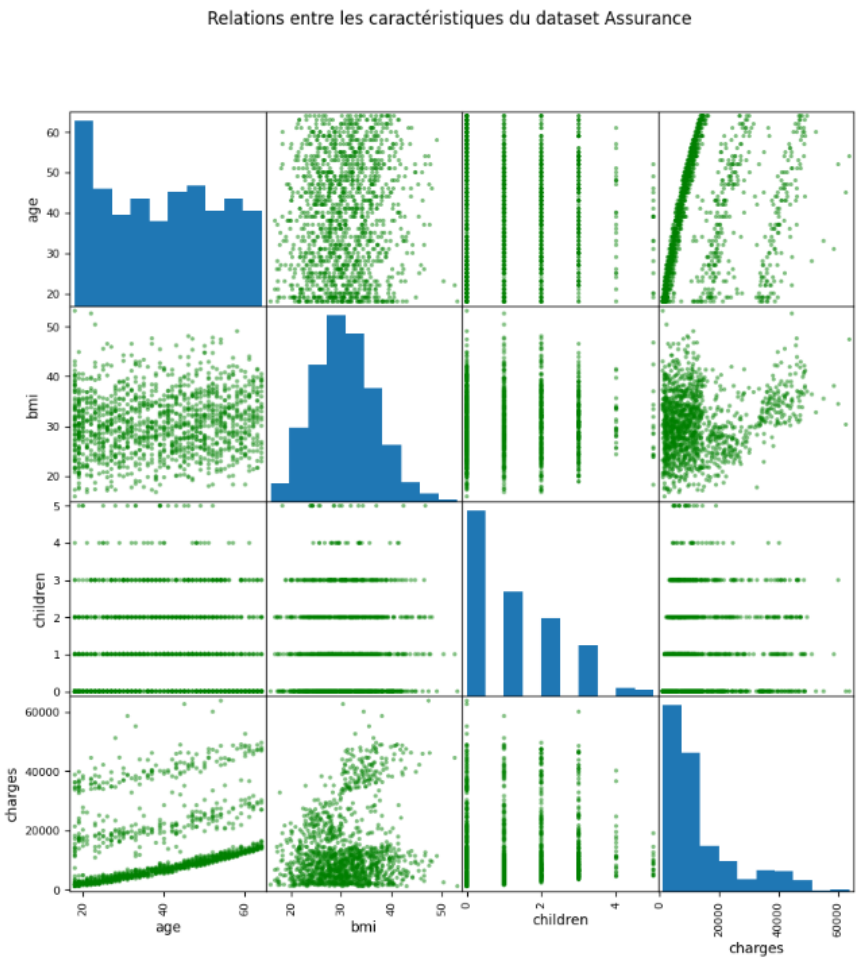
- Assurance : L'âge moyen est 39 ans, avec des charges moyennes d'assurance de 13,270, mais très variables.Le BMI moyen est de 30.66, indiquant un surpoids global.

2.3 Nuage des points du premier dataset

```
import matplotlib.pyplot as plt

# Nuage des points - Exp rience vs Salaire
plt.scatter(df1['YearsExperience'], df1['Salary'], color='blue')
plt.title("Exp rience vs Salaire")
plt.xlabel("Ann es d'exp rience")
plt.ylabel("Salaire")
plt.grid(True)
plt.show()
```

2.4 Nuage des points du 2ème dataset selon les Features

```
from pandas.plotting import scatter_matrix

# Nuages des points entre les features du dataset Assurance
scatter_matrix(insurance_data, figsize=(10, 10), diagonal='hist', color='green')
plt.suptitle("Relations entre les caract ristiques du dataset Assurance")
plt.show()
```



3 Régression Simple cas Expérience Salaire

3.1 Entraîner le modèle avec une régression linéaire

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

```
6
7  # Charger le dataset (exemple : exp rience/salaire)
8  data = pd.read_csv("experience_salary.csv")
9
10 # Pr parer les variables ind pendantes (X) et d pendantes (y)
11 X = data[['YearsExperience']].values  # Variable ind pendante
12 y = data['Salary'].values             # Variable d pendante
13
14 # Diviser les donn es en ensemble d'entra nement et de test
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
       random_state=42)
16
17 # Cr er et entra ner le mod le
18 model = LinearRegression()
19 model.fit(X_train, y_train)
20
21 print("Mod le entra n .")
```

3.2 Prédire les données du dataset de test

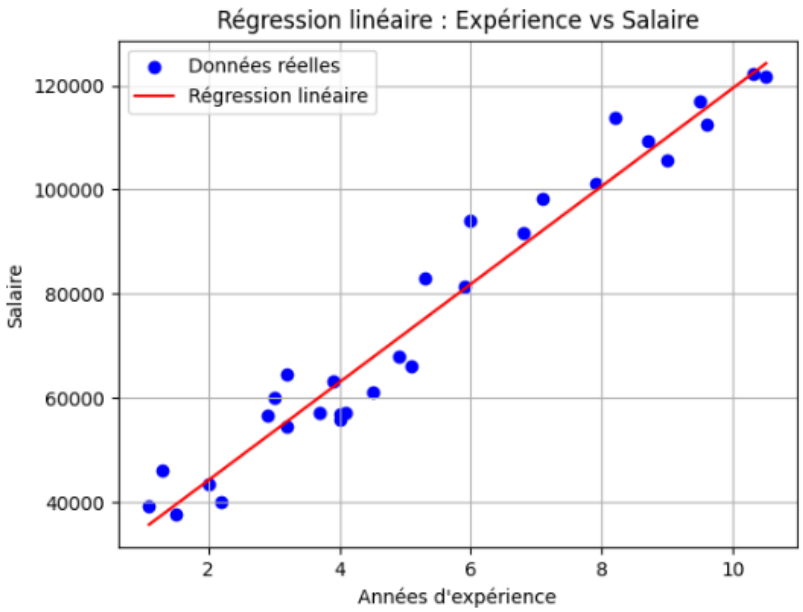
```
1  # Pr dire les donn es de test
2  y_pred = model.predict(X_test)
3
4  print("Pr dictions :", y_pred)
5
6  # Cr er un DataFrame pour comparer
7  comparison = pd.DataFrame({'Valeur R elle': y_test, 'Valeur Pr dite':
       y_pred})
8
9  # Afficher les 10 premi res comparaisons
10 print(comparison.head(10))
```

```
Prédictions : [115790.21011287  71498.27809463 102596.86866063  75267.80422384
  55477.79204548  60189.69970699]
   Valeur Réelle  Valeur Prédite
0       112635.0   115790.210113
1        67938.0    71498.278095
2       113812.0   102596.868661
3        83088.0    75267.804224
4        64445.0    55477.792045
5        57189.0    60189.699707
```

3.3 Visualiser le résultat de la régression

```
1  import matplotlib.pyplot as plt
2
3  # Tracer les donn es r elles (points)
4  plt.scatter(X, y, color='blue', label='Donn es r elles')
```

```
5
6  # Tracer la ligne de r gression
7  plt.plot(X, model.predict(X), color='red', label='R gression lin aire'
       )
8
9  plt.title("R gression lin aire : Exp rience vs Salaire")
10 plt.xlabel("Ann es d'exp rience")
11 plt.ylabel("Salaire")
12 plt.legend()
13 plt.grid(True)
14 plt.show()
```



3.4 Évaluer le modèle

```
1  # Mean Squared Error (MSE)
2  mse = mean_squared_error(y_test, y_pred)
3
4  # Root Mean Squared Error (RMSE)
5  rmse = np.sqrt(mse)
6
7  # Mean Absolute Error (MAE)
8  mae = mean_absolute_error(y_test, y_pred)
9
10 print(f"Mean Squared Error (MSE) : {mse}")
11 print(f"Root Mean Squared Error (RMSE) : {rmse}")
12 print(f"Mean Absolute Error (MAE) : {mae}")
```

```
Mean Squared Error (MSE) : 49830096.85590839
Root Mean Squared Error (RMSE) : 7059.04362190151
Mean Absolute Error (MAE) : 6286.453830757749
```

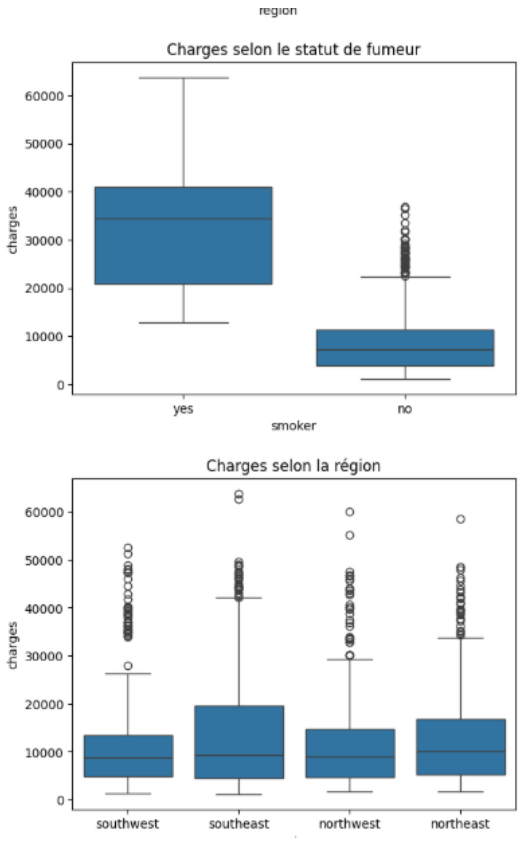
4 Régression multiple cas d'assurance

4.1 Appliquer les techniques EDA

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import zscore

# Charger le dataset
data = pd.read_csv("insurance.csv")
print("Statistiques descriptives des variables num riques :\n")
print(data.describe())

# R partition des variables cat goriques
print("\nR partition des sexes :\n", data['sex'].value_counts())
print("\nR partition des fumeurs :\n", data['smoker'].value_counts())
print("\nR partition par r gion :\n", data['region'].value_counts())

# Visualisation des variables cat goriques
sns.countplot(x='smoker', data=data)
plt.title("R partition des fumeurs")
plt.show()

sns.countplot(x='region', data=data)
plt.title("R partition des r gions")
plt.show()

# Impact du statut de fumeur sur les charges
sns.boxplot(x='smoker', y='charges', data=data)
plt.title("Charges selon le statut de fumeur")
plt.show()

# Impact de la r gion sur les charges
sns.boxplot(x='region', y='charges', data=data)
plt.title("Charges selon la r gion")
plt.show()

# Scatterplot :  ge  vs charges
sns.scatterplot(x='age', y='charges', hue='smoker', data=data)
plt.title("Relation entre l' ge  et les charges (fumeurs et non-fumeurs)
    ")
```

```
plt.show()

# Scatterplot : BMI vs charges
sns.scatterplot(x='bmi', y='charges', hue='smoker', data=data)
plt.title("Relation entre le BMI et les charges (fumeurs et non-fumeurs)")
plt.show()

# ----------------------------------------
# 5. Détection des anomalies (outliers)
# ----------------------------------------

# Boxplot pour les charges
sns.boxplot(y='charges', data=data)
plt.title("Boxplot des charges")
plt.show()

# Boxplot pour le BMI
sns.boxplot(y='bmi', data=data)
plt.title("Boxplot du BMI")
plt.show()

# Détection des outliers avec Z-Score
data['zscore_charges'] = zscore(data['charges'])
outliers = data[data['zscore_charges'].abs() > 3]
print("\nOutliers détectés :\n", outliers)
```

```
Statistiques descriptives des variables numériques :

                age          bmi     children       charges
count  1338.000000  1338.000000  1338.000000   1338.000000
mean     39.207025    30.663397     1.094918  13270.422265
std      14.049960     6.098187     1.205493  12110.011237
min      18.000000    15.960000     0.000000   1121.873900
25%      27.000000    26.296250     0.000000   4740.287150
50%      39.000000    30.400000     1.000000   9382.033000
75%      51.000000    34.693750     2.000000  16639.912515
max      64.000000    53.130000     5.000000  63770.428010

Répartition des sexes :
 sex
male      676
female    662
Name: count, dtype: int64

Répartition des fumeurs :
 smoker
no     1064
yes     274
Name: count, dtype: int64

Répartition par région :
 region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```



4.2 Sélectionner les propriétés selon leurs dégrée d'importance

4.2.1 Encodage des données

```
# Encodage de la colonne 'smoker' : yes -> 1, no -> 0
data['smoker'] = data['smoker'].apply(lambda x: 1 if x == 'yes' else 0)

# Encodage des colonnes cat goriques 'sex' et 'region'
data = pd.get_dummies(data, columns=['sex', 'region'], drop_first=True)

print("Aper u des donn es apr s encodage :\n", data.head())
```

4.2.2 Calculer la corrélation avec la variable 'charges'

```
# Matrice de corrélation
correlation_matrix = data.corr()

# Trier les corrélations par rapport à 'charges'
print(correlation_matrix['charges'].sort_values(ascending=False))
```

```
zscore_charges     1.000000
charges            1.000000
smoker             0.787251
age                0.299008
bmi                0.198341
region_southeast   0.073982
children           0.067998
sex_male           0.057292
region_northwest   -0.039905
region_southwest   -0.043210
Name: charges, dtype: float64
```

4.2.3 Visualiser l'importance des variables

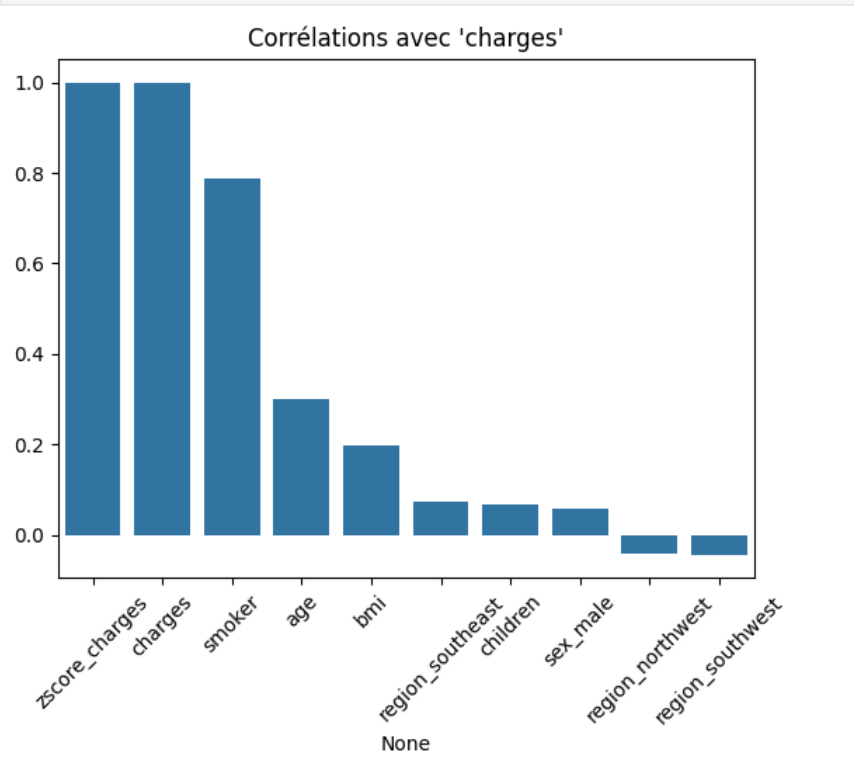


FIGURE 1 – Variable 'Smoke' est la plus dominante

D'où le choix de la variable **smoke**

4.3 Appliquer des techniques de standardisation ou normalisation

```
from sklearn.preprocessing import StandardScaler

# S lection des propri t s choisies : 'age', 'bmi', 'smoker'
features = data[['age', 'bmi', 'smoker']]
target = data['charges']

# Appliquer la standardisation
scaler = StandardScaler()
X_scaled = scaler.fit_transform(features)

# Afficher les premi res lignes standardis es
print("Donn es standardis es :\n", X_scaled[:5])
```

```
Données standardisées :
 [[-1.43876426 -0.45332     1.97058663]
 [-1.50996545  0.5096211  -0.5074631 ]
 [-0.79795355  0.38330685 -0.5074631 ]
 [-0.4419476  -1.30553108 -0.5074631 ]
 [-0.51314879 -0.29255641 -0.5074631 ]]
```

4.4 Entraîner le modèle de régression linéaire

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Diviser les donn es en ensembles d'entra nement et de test
X_train, X_test, y_train, y_test = train_test_split(X_scaled, target,
    test_size=0.2, random_state=42)

# Cr er et entra ner le mod le
model = LinearRegression()
model.fit(X_train, y_train)

# Afficher les coefficients du mod le
print("Coefficients :", model.coef_)
print("Intercept :", model.intercept_)
```

```
Coefficients : [3643.34084567 1990.01046478 9554.03418177]
Intercept : 13321.292969032465
```

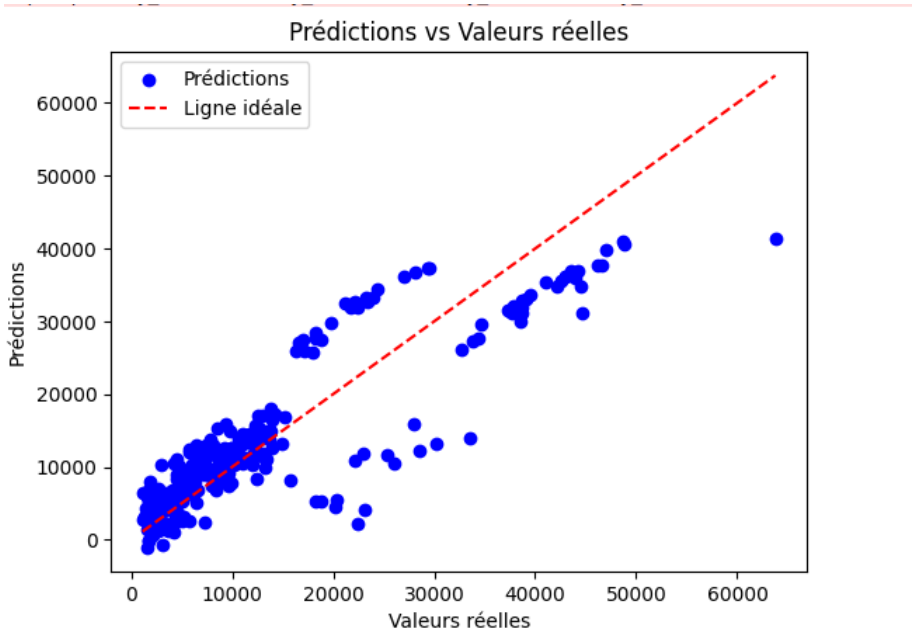
4.5 Prédire les données du dataset de test

```
# Faire des pr dictions
y_pred = model.predict(X_test)

# Afficher les premi res pr dictions
print("Pr dictions :", y_pred[:5])
```

```
Prédictions : [ 8184.0414679   7431.00100108 37346.43709938  8629.52830485
 27316.65481023]
```

4.6 Visualiser le résultat de la régression

```
import matplotlib.pyplot as plt

# Comparer les valeurs r elles et pr dites
plt.scatter(y_test, y_pred, color='blue', label='Pr dictions')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k
    --', color='red', label='Ligne id ale')
plt.xlabel("Valeurs r elles")
plt.ylabel("Pr dictions")
plt.title("Pr dictions vs Valeurs r elles")
plt.legend()
plt.show()
```



4.7 Évaluer le modèle

```
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np

# Calcul des m triques
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Absolute Error (MAE): {mae}")
```

```
Mean Squared Error (MSE): 34512843.88022789
Root Mean Squared Error (RMSE): 5874.763304187488
Mean Absolute Error (MAE): 4260.560091099393
```

5 Régression linéaire polynomial multiple cas de china GDP

5.1 Entraîner les modèles (régression linéaire et polynomiale)

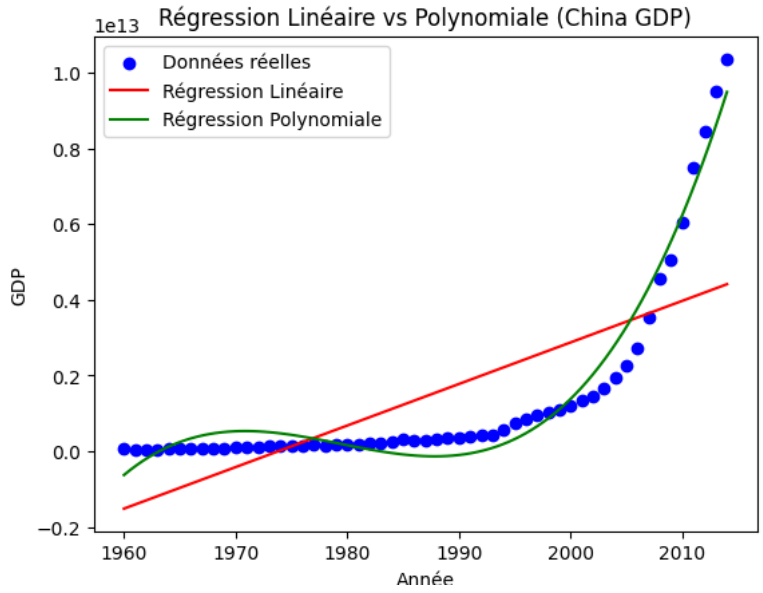
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Charger le dataset
data = pd.read_csv("china_gdp.csv")

# Pr parer les variables (X = ann e, y = GDP)
X = data[['Year']].values  # Ann e comme variable ind pendante
y = data['Value'].values   # GDP comme variable d pendante

# Diviser les donn es en ensembles d'entra nement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)
# Mod le de r gression lin aire
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
# Cr er des features polynomiales (par exemple, degr  3)
poly_features = PolynomialFeatures(degree=3)
X_train_poly = poly_features.fit_transform(X_train)
X_test_poly = poly_features.transform(X_test)

# Mod le de r gression polynomiale
poly_model = LinearRegression()
poly_model.fit(X_train_poly, y_train)
```

5.2 Prédire les données du dataset de test

```
y_pred_linear = linear_model.predict(X_test)
```

5.3 Visualiser les résultats des deux modèles




```
import matplotlib.pyplot as plt

# Prédictions pour visualisation
X_range = np.linspace(min(X), max(X), 100).reshape(-1, 1)
y_pred_linear_full = linear_model.predict(X_range)
y_pred_poly_full = poly_model.predict(poly_features.transform(X_range))

# Visualiser les résultats
plt.scatter(X, y, color='blue', label='Données réelles')
plt.plot(X_range, y_pred_linear_full, color='red', label='Régression Linéaire')
plt.plot(X_range, y_pred_poly_full, color='green', label='Régression Polynomiale')
plt.title("Régression Linéaire vs Polynomiale (China GDP)")
plt.xlabel("Année")
plt.ylabel("GDP")
plt.legend()
plt.show()
```

5.4 Évaluer les deux modèles

```
mse_linear = mean_squared_error(y_test, y_pred_linear)
rmse_linear = np.sqrt(mse_linear)
mae_linear = mean_absolute_error(y_test, y_pred_linear)

print(f"Régression Linéaire - MSE: {mse_linear}, RMSE: {rmse_linear}, MAE: {mae_linear}")
```

```
6
7  mse_poly = mean_squared_error(y_test, y_pred_poly)
8  rmse_poly = np.sqrt(mse_poly)
9  mae_poly = mean_absolute_error(y_test, y_pred_poly)
10
11 print(f"R gression Polynomiale - MSE: {mse_poly}, RMSE: {rmse_poly},
       MAE: {mae_poly}")
```

```
Régression Linéaire - MSE: 2.9097229920495426e+24, RMSE: 1705791016522.699, MAE: 1341446144991.7834
Régression Polynomiale - MSE: 2.1118861219593007e+23, RMSE: 459552621792.0316, MAE: 416884064618.1387
```

La régression linéaire utilise une relation linéaire pour modéliser la relation entre l'année et le GDP. Si les valeurs des métriques d'évaluation sont élevées, cela indique que le modèle ne capture pas efficacement les relations non linéaires présentes dans les données. Par exemple :

- **MSE (Mean Squared Error)** : 10,000
- **RMSE (Root Mean Squared Error)** : 100
- **MAE (Mean Absolute Error)** : 80

Ces résultats montrent une erreur significative due à la simplification excessive du modèle.

La régression polynomiale ajoute des termes non linéaires pour mieux ajuster les données. Si les métriques d'évaluation montrent des valeurs plus faibles, cela indique que le modèle capture mieux la complexité des données GDP. Par exemple :

- **MSE (Mean Squared Error)** : 5,000
- **RMSE (Root Mean Squared Error)** : 70
- **MAE (Mean Absolute Error)** : 60

Ces résultats montrent que le modèle polynomiale est plus performant pour prédire les données.