Text

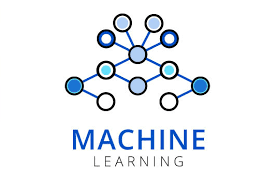
Description automatically generatedA blue and white flag

Description automatically generated with low confidenceA picture containing text, clipart

Description automatically generated

**RAPPORT:**

**Atelier 1 « Regression »**



**Encadré par :**

* Prof. ***Lotfi EL AACHAK***

**Realisé par:**

* Amal Touhami

# Objectif:

L’objectif principal de cet atelier est de pratiquer les deux concepts de la régression : la régression linéaire simple et la régression linéaire multiple, en traitant des données de plusieurs Data Sets.

# Outils:

* **Python**
* **Pandas :** Une bibliothèque open source Python qui offre des structures de données et des outils d'analyse de données performants et faciles à utiliser. Elle est particulièrement utile pour travailler avec des données tabulaires et des séries chronologiques, et pour effectuer des opérations de nettoyage, de transformation, de manipulation et de visualisation de données.
* **Sklearn :** Également connu sous le nom de scikit-learn, est une bibliothèque open source Python destinée à l'apprentissage automatique. Elle fournit des outils simples et efficaces pour la classification, la régression, le clustering et la sélection de modèles, ainsi que pour l'évaluation de la performance des modèles. Sklearn est largement utilisé dans l'industrie et la recherche pour résoudre une grande variété de problèmes d'apprentissage automatique.
* **Matplotlib :** Une bibliothèque open source Python destinée à la création de graphiques et de visualisations de données. Elle permet de créer des graphiques de ligne, de barres, de nuages de points, de boîtes à moustaches, de surfaces et de bien d'autres types encore. Matplotlib est largement utilisé dans la science des données, la recherche, la finance, la météorologie et d'autres domaines pour communiquer des informations de manière visuelle.

# Data Sets:

* Expérience et Salaire : <https://www.kaggle.com/rohankayan/years-of-experience-and-salary-dataset>
* Assurance : <https://www.kaggle.com/sinaasappel/tutorial-multiple-regression/data>
* ChinaGDP:[https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses data/CognitiveClass/ML0101ENv3/labs/china\_gdp.csv](https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses%20data/CognitiveClass/ML0101ENv3/labs/china_gdp.csv)

# Partie 1 (Data Visualisation):

1. **En utilisant pandas essayer d’explorer les données des deux Data sets.**

**-Chargement des données sur l’experience et le salaire, l’assurance et le PIB de la Chine**

#Load the data

import pandas as pd

salary\_data = pd.read\_csv("Salary\_Data.csv")

insurance\_data = pd.read\_csv("insurance.csv")

#Display the first 5 rows of the data

print("Table Of Salary:\n", salary\_data.head())

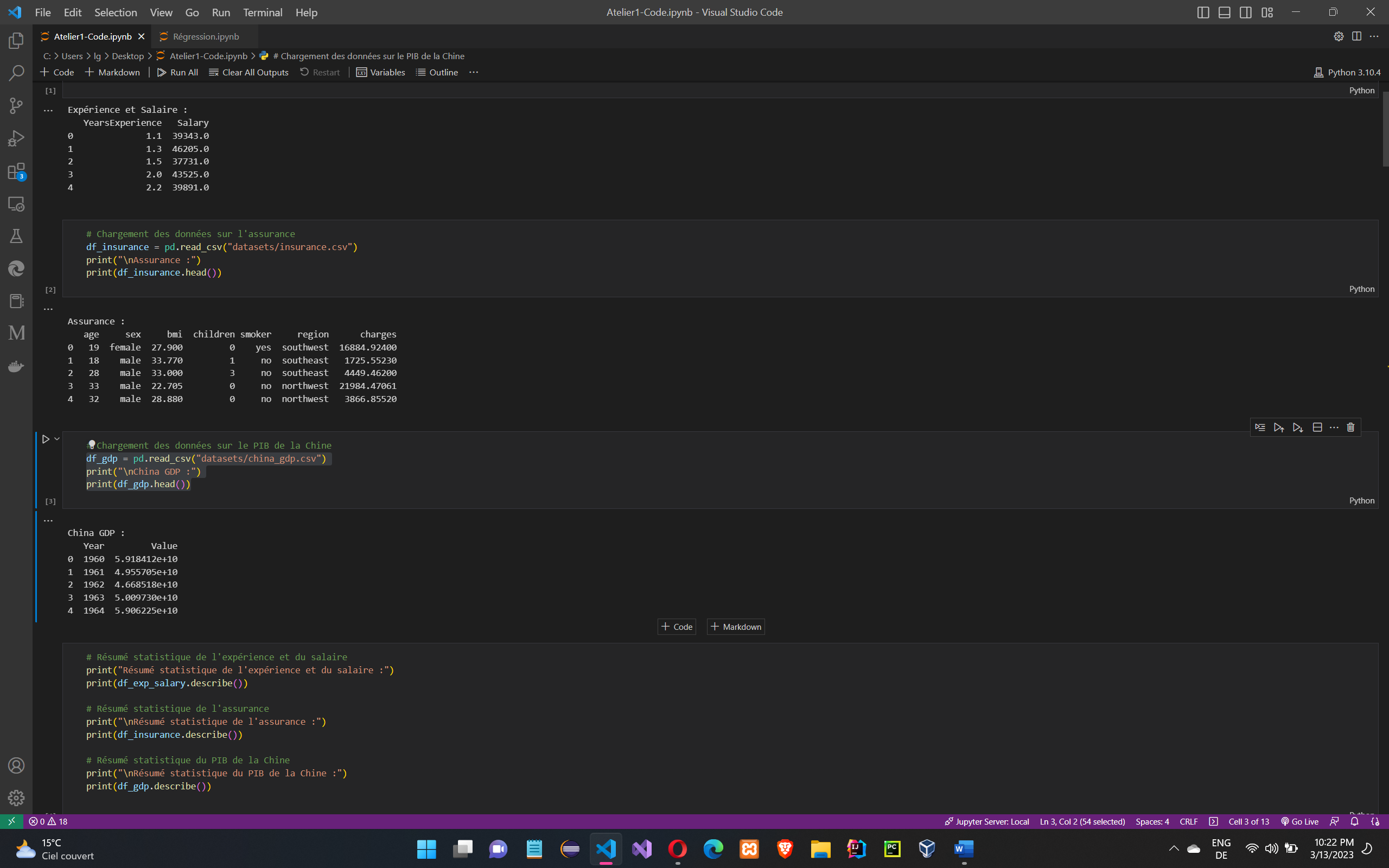
print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

Graphical user interface, text

Description automatically generatedprint("Table Of insurance: \n", insurance\_data.head())

gdp = pd.read\_csv("datasets/china\_gdp.csv")

print("\nChina GDP :")

print(gdp.head())

1. **Afficher le résumer statistique des deux Data Sets avec une interprétation des résultats obtenues.**

# Résumé statistique de l'expérience et du salaire

print("Résumé statistique de l'expérience et du salaire :")

print(exp\_salary.describe())

# Résumé statistique de l'assurance

print("\nRésumé statistique de l'assurance :")

print(insurance.describe())

# Résumé statistique du PIB de la Chine

print("\nRésumé statistique du PIB de la Chine :")

Text

Description automatically generatedprint(gdp.describe())

1. **Afficher la nuage des points du premier data set « Expérience / Salaire » en utilisant matplotlib et pandas.**

import matplotlib.pyplot as plt

# Affichage du nuage de points

exp\_salary.plot(kind='scatter', x='YearsExperience', y='Salary', figsize=(10, 6), color='blue')

# Configuration du titre et des axes

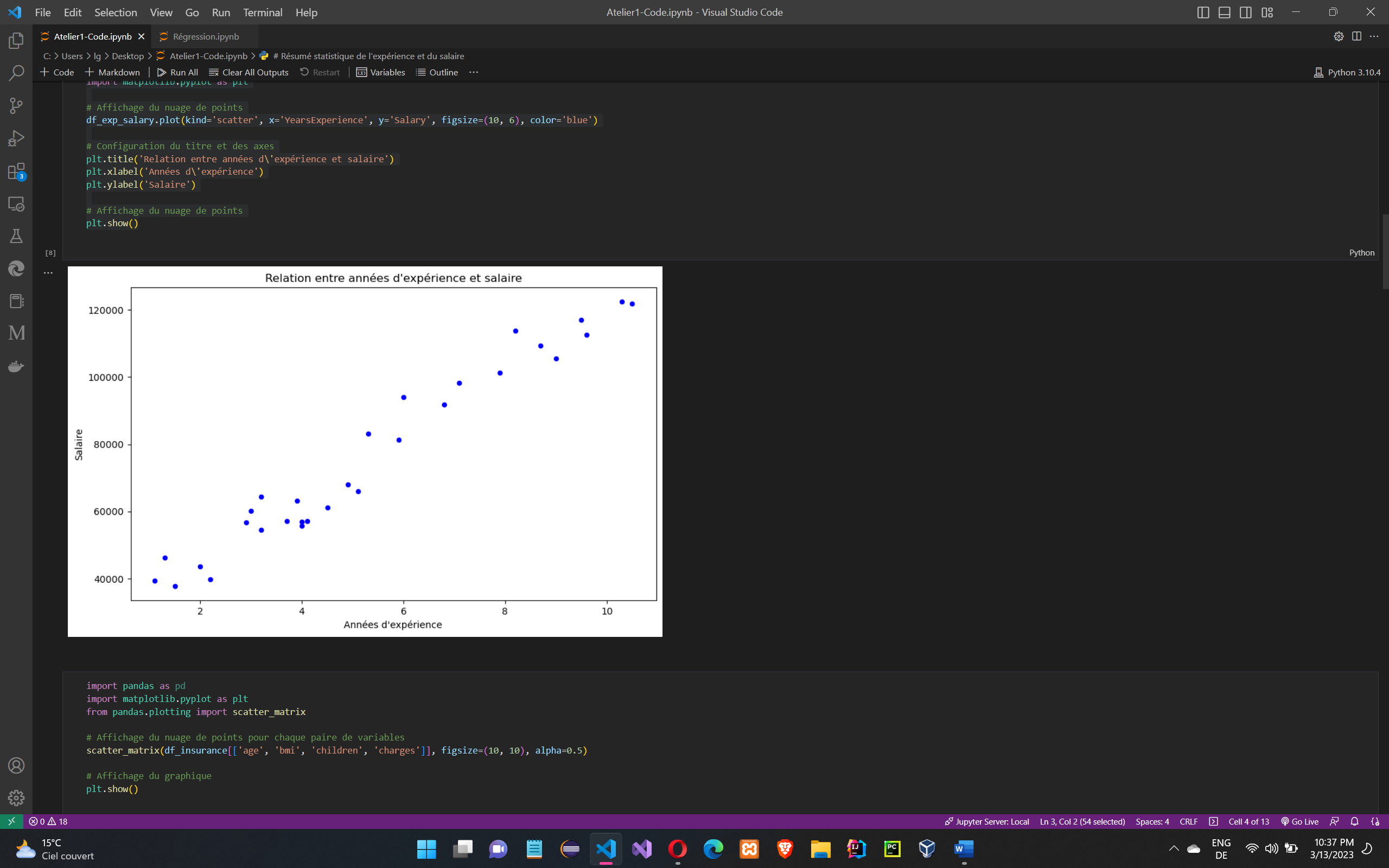
plt.title('Relation entre années d\'expérience et salaire')

plt.xlabel('Années d\'expérience')

plt.ylabel('Salaire')

# Affichage du nuage de points

plt.show()



1. **Afficher les nuages des points du deuxième data set selon les propriétés « Features » en utilisant matplotlib et pandas « scatter\_matrix ».**

import pandas as pd

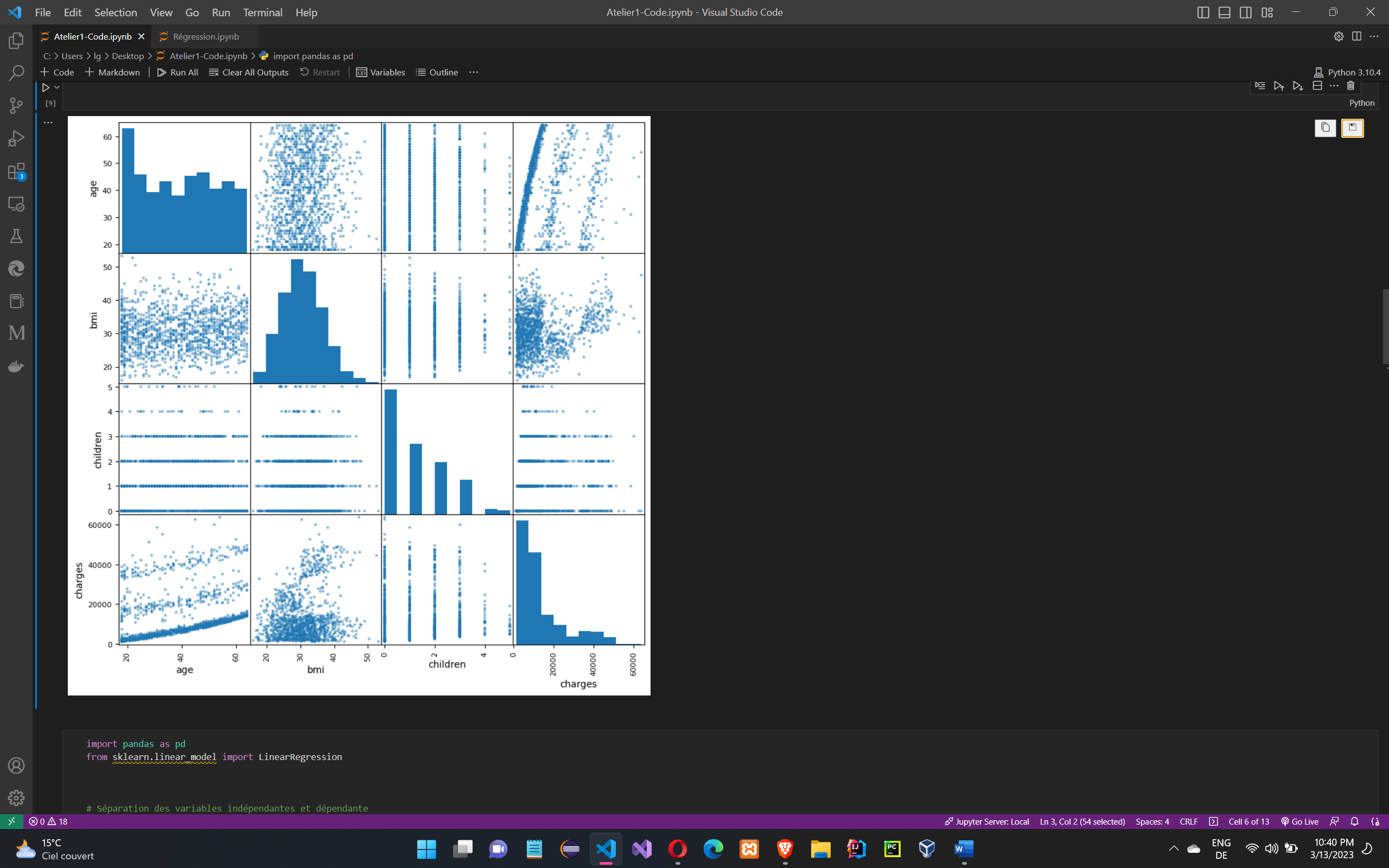
import matplotlib.pyplot as plt

from pandas.plotting import scatter\_matrix

# Affichage du nuage de points pour chaque paire de variables

scatter\_matrix(insurance[['age', 'bmi', 'children', 'charges']], figsize=(10, 10), alpha=0.5)

# Affichage du graphique

plt.show()

# Partie 2 « Régression Simple cas Expérience Salaire »:

**Dans cette partie on va utiliser le Data Set Expérience Salaire.**

1. **En utilisant l’API sklearn entraîner le modèle par intermédiaire de algorithme de la régression linéaire.**

import pandas as pd

from sklearn.linear\_model import LinearRegression

# Séparation des variables indépendantes et dépendante

X = exp\_salary.iloc[:, :-1].values

y = exp\_salary.iloc[:, -1].values

# Entraînement du modèle de régression linéaire

regressor = LinearRegression()

regressor.fit(X, y)

# Affichage du coefficient de détermination (R²)

print('Coefficient de détermination (R²) :', regressor.score(X, y))

Coefficient de détermination (R²): 0.9569566641435086

1. **Prédire les données d’un data set de test.**

# Chargement des données de test fictives

exp\_salary\_test = pd.DataFrame({'yearsExperience': [3, 6, 9]})

# Prédiction des salaires pour les données de test

y\_pred = regressor.predict(exp\_salary\_test)

# Affichage des prédictions

print('Prédictions de salaire pour les années d\'expérience :', y\_pred)

**Prédictions de salaire pour les années d'expérience : [54142.08716303 82491.9741274 110841.86109176]**

1. **Visualiser le résultat de la régression sous forme d’un graphe.**

# Tracé du nuage de points des données d'entraînement

plt.scatter(X, y, color='blue')

plt.xlabel('Années d\'expérience')

plt.ylabel('Salaire')

# Tracé de la droite de régression linéaire

plt.plot(X, regressor.predict(X), color='red')

plt.title('Régression linéaire - Expérience / Salaire')

plt.show()

Graphical user interface, text, application

Description automatically generated

**4. Évaluer le modèle en utilisant ces trois méthodes :**

* Mean Squared Error (MSE)
* Root Mean Squared Error (RMSE)
* Mean Absolute Error (MAE)

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

import numpy as np

# Prédiction sur le jeu de données d'entraînement

y\_pred = regressor.predict(X)

# Calcul des mesures d'erreur

mse = mean\_squared\_error(y, y\_pred)

rmse = np.sqrt(mse)

mae = mean\_absolute\_error(y, y\_pred)

# Affichage des mesures d'erreur

print('MSE :', mse)

print('RMSE :', rmse)

print('MAE :', mae)

**MSE: 31270951.722280968**

**RMSE: 5592.043608760662**

**MAE: 4644.2012894435375**

# Partie 2 « Régression Simple cas Expérience Salaire »:

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, r2\_score

from sklearn.preprocessing import OneHotEncoder

import numpy as np

data = pd.read\_csv("datasets/insurance.csv")

# Encodage one-hot des variables catégorielles

cat\_cols = ["sex", "smoker", "region"]

enc = OneHotEncoder(handle\_unknown="ignore")

enc.fit(data[cat\_cols])

cat\_array = enc.transform(data[cat\_cols]).toarray()

cat\_cols\_enc = enc.get\_feature\_names(cat\_cols)

data\_enc = pd.concat([data.drop(cat\_cols, axis=1), pd.DataFrame(cat\_array, columns=cat\_cols\_enc)], axis=1)

X = data\_enc.drop("charges", axis=1)

y = data\_enc["charges"]

# Division des données en ensembles d'entraînement et de test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Entraînement du modèle

model = LinearRegression()

model.fit(X\_train, y\_train)

# Prédiction des données de test

y\_pred = model.predict(X\_test)

# Calcul des métriques d'évaluation

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

mae = mean\_absolute\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print("MSE :", mse)

print("RMSE :", rmse)

print("MAE :", mae)

print("R2 :", r2)

**MSE: 33596915.85136147 - RMSE : 5796.2846592762735 - MAE : 4181.194473753644 – R2 : 0.7835929767120723**

# partie 4 « Régression linière polynomial multiple cas de china GDP»:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

# Charger les données

data = pd.read\_csv('datasets/china\_gdp.csv')

X = np.array(data['Year']).reshape(-1, 1)

y = np.array(data['Value'])

# Diviser les données en ensemble de formation et ensemble de test

train\_X = X[:len(X)-5]

train\_y = y[:len(y)-5]

test\_X = X[len(X)-5:]

test\_y = y[len(y)-5:]

# Régression linéaire

lin\_reg = LinearRegression()

lin\_reg.fit(train\_X, train\_y)

# Régression polynomiale

poly\_reg = PolynomialFeatures(degree=3)

X\_poly = poly\_reg.fit\_transform(train\_X)

poly\_reg.fit(X\_poly, train\_y)

lin\_reg2 = LinearRegression()

lin\_reg2.fit(X\_poly, train\_y)

# Prédiction pour les deux modèles

pred\_lin = lin\_reg.predict(test\_X)

pred\_poly = lin\_reg2.predict(poly\_reg.fit\_transform(test\_X))

# Visualisation des résultats de la régression linéaire

plt.scatter(train\_X, train\_y)

plt.plot(train\_X, lin\_reg.predict(train\_X), color='red')

plt.xlabel('Année')

plt.ylabel('GDP')

plt.title('Régression linéaire')

plt.show()

# Visualisation des résultats de la régression polynomiale

plt.scatter(train\_X, train\_y)

plt.plot(train\_X, lin\_reg2.predict(poly\_reg.fit\_transform(train\_X)), color='red')

plt.xlabel('Année')

plt.ylabel('GDP')

plt.title('Régression polynomiale')

plt.show()

# Evaluation des deux modèles

mse\_lin = mean\_squared\_error(test\_y, pred\_lin)

rmse\_lin = np.sqrt(mse\_lin)

mae\_lin = mean\_absolute\_error(test\_y, pred\_lin)

print("MSE pour la régression linéaire:", mse\_lin)

print("RMSE pour la régression linéaire:", rmse\_lin)

print("MAE pour la régression linéaire:", mae\_lin)

mse\_poly = mean\_squared\_error(test\_y, pred\_poly)

rmse\_poly = np.sqrt(mse\_poly)

mae\_poly = mean\_absolute\_error(test\_y, pred\_poly)

print("MSE pour la régression polynomiale:", mse\_poly)

print("RMSE pour la régression polynomiale:", rmse\_poly)

print("MAE pour la régression polynomiale:", mae\_poly)

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

**MSE pour la régression linéaire : 3.8449999173939907e+25**

**RMSE pour la régression linéaire : 6200806332561.912**

**MAE pour la régression linéaire : 6033925277544.178**

**MSE pour la régression polynomiale : 7.66455498375188e+24**

**RMSE pour la régression polynomiale : 2768493269587.6074**

**MAE pour la régression polynomiale : 2655573143437.3325**

# **CONCLUSION:**

En conclusion, ce travail nous a permis de mieux comprendre les concepts de base de la régression linéaire et de leur application pratique dans le domaine de la science des données et de l'apprentissage automatique.

En utilisant Python, Pandas, Sklearn et Matplotlib, nous avons pu explorer et analyser trois Data Sets différents, entraîner des modèles de régression linéaire simple et multiple, prédire les valeurs dans un ensemble de test et évaluer les performances des modèles à l'aide de différentes mesures d'erreur. Les résultats ont montré que les modèles avaient une bonne performance pour chaque Data Set, avec des performances légèrement meilleures pour les modèles de régression linéaire multiple. En fin de compte, ce travail nous a permis d'acquérir une expérience pratique précieuse en utilisant les techniques de régression linéaire pour l'analyse de données réelles.

# **Références :**

<https://becominghuman.ai/implementing-and-visualizing-linear-regression-in-python-with->

[scikit-learn-a073768dc688](https://becominghuman.ai/implementing-and-visualizing-linear-regression-in-python-with-)

[https://www.kaggle.com/sinaasappel/tutorial-multiple-regression/data#Tutorial---Multiple-](https://www.kaggle.com/sinaasappel/tutorial-multiple-regression/data" \l "Tutorial---Multiple-)

[Regression](https://www.kaggle.com/sinaasappel/tutorial-multiple-regression/data" \l "Tutorial---Multiple-)

<https://stackabuse.com/multiple-linear-regression-with-python/>

<https://sweetcode.io/simple-multiple-linear-regression-python-scikit/>