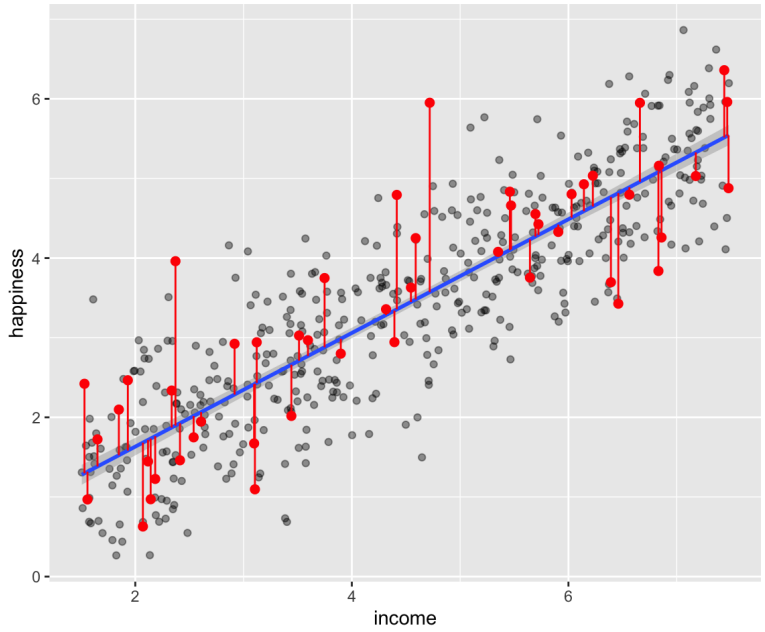
Université Abdelmalek Essaadi
Faculté des Sciences et Techniques de Tanger
Département Génie Informatique

Atelier 1 :
« Regression »



Encadré par :
Prof. ELAACHAK LOTFI

Realisé par :
BARBYCH Mohamed

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

Session d'octobre 2024

Table des matières

1 Introduction

L'objectif de cet atelier est de comprendre et de mettre en pratique les concepts fondamentaux de la regression lineaire simple et multiple, ainsi que la regression polynomiale, en s'appuyant sur des données reelles. Ce rapport documente les etapes, les analyses, et les conclusions tirees de ces experiences. La regression est un outil cle dans l'apprentissage automatique, permettant de modeliser des relations entre des variables et de faire des predictions precises. Nous avons choisi des datasets varies pour mieux comprendre l'impact des modèles et techniques sur des données reelles, en explorant differentes approches analytiques et de visualisation.

2 Introduction aux bibliothèques utilisees

Pour mener à bien cette etude, plusieurs bibliothèques Python essentielles ont ete importees.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error,
    r2_score
import seaborn as sns
```

Voici une brève description de chacune d'elles :

- **pandas** : Utilisee pour la manipulation et l'analyse des données, notamment pour lire et traiter les ensembles de données sous forme de DataFrame.
- **numpy** : Fournit des fonctions mathematiques avancees pour les calculs numeriques et est souvent utilise pour travailler avec des tableaux multidimensionnels.
- **matplotlib.pyplot** : Une bibliothèque de visualisation pour generer des graphiques simples ou complexes.
- **pandas.plotting.scatter_matrix** : Un outil de `pandas` permettant de generer des matrices de graphiques de dispersion pour analyser les relations entre plusieurs variables.
- **sklearn.model_selection.train_test_split** : Permet de diviser les ensembles de données en ensembles d'entraînement et de test pour la modelisation.
- **sklearn.linear_model.LinearRegression** : Implemente des modèles de regression lineaire pour la prediction des valeurs continues.

— **sklearn.preprocessing.PolynomialFeatures et StandardScaler** : Utilisees respectivement pour transformer les caracteristiques en polynômes et pour standardiser les données en les centrant et en les reduisant.

— **sklearn.metrics.mean_squared_error, mean_absolute_error et r2_score** : Fournissent des outils pour evaluer la performance des modèles, notamment à l'aide des erreurs quadratiques moyennes, des erreurs absolues moyennes et du coefficient de determination R^2.

— **seaborn** : Une bibliothèque de visualisation statistique basee sur `matplotlib`, facilitant la creation de graphiques plus esthetiques et informatifs.

Ces outils combines permettent une exploration approfondie des données, une modelisation performante et une evaluation robuste des resultats obtenus.

3 Partie 1 : Visualisation des données

3.1 Exploration des jeux de données

Les jeux de données utilises sont les suivants :

— **Experience et Salaire :** Ce dataset relie les annees d'experience professionnelle à un salaire annuel.

— **Assurance :** Ce dataset inclut des données demographiques et des informations sur les coûts d'assurance medicale.

— **China GDP :** Ce dataset montre l'evolution du PIB chinois sur plusieurs decennies.

Le code ci-dessous permet de charger et d'explorer les données :

```
# Chargement des datasets
experience_salary = pd.read_csv("./Salary_Data.csv")
insurance = pd.read_csv("./insurance.csv")
china_gdp = pd.read_csv("./china_gdp.csv")


# Partie 1 : Exploration et visualisation
# Question 1 : Explorer les donn es
# Exploration des donn es des deux datasets
print("------------- Exploration du dataset Experience-Salaire :
    -------------")
print(experience_salary.info())
print(experience_salary.head())
```

```
13
14 print("\n------------- Exploration du dataset Assurance : -------------")
15 print(insurance.info())
16 print(insurance.head())
```

Les resultats de cette etape sont illustres dans la Figure 1.

```
------------- Exploration du dataset Expérience-Salaire : -------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 612.0 bytes
None
   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

------------- Exploration du dataset Assurance : -------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
None
   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
```

FIGURE 1 – Exploration des jeux de données.

3.2 Resumé statistique

Pour mieux comprendre la structure des données, voici le Resumé statistique :

```
# Question 2 : Resum  statistique avec interpretation
# Resum  statistique pour le dataset Experience-Salaire
print("------------ Resum  statistique du dataset Experience-Salaire :
    ------------")
print(experience_salary.describe())

# Resum  statistique pour le dataset Assurance
print("\n------------ Resum  statistique du dataset Assurance :
    ------------")
print(insurance.describe())
```

Les resultats de cette etape sont illustres dans la Figure 2.

```
------------ Résumé statistique du dataset Expérience-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

------------ Résumé statistique 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
```

FIGURE 2 – Resumé statistique des jeux de données.

3.3 Nuage des points du premier data set « Experience / Salaire »

Voici les visualisations demandees :

```
# Question 3 : Nuage des points pour le dataset Experience-Salaire
experience_salary.plot(kind='scatter', x='YearsExperience', y='Salary',
    title='Experience vs Salaire')

plt.xlabel('Annees d\'experience')
plt.ylabel('Salaire')
plt.title('Nuage de points : Experience vs Salaire')
plt.show()
plt.show()
```

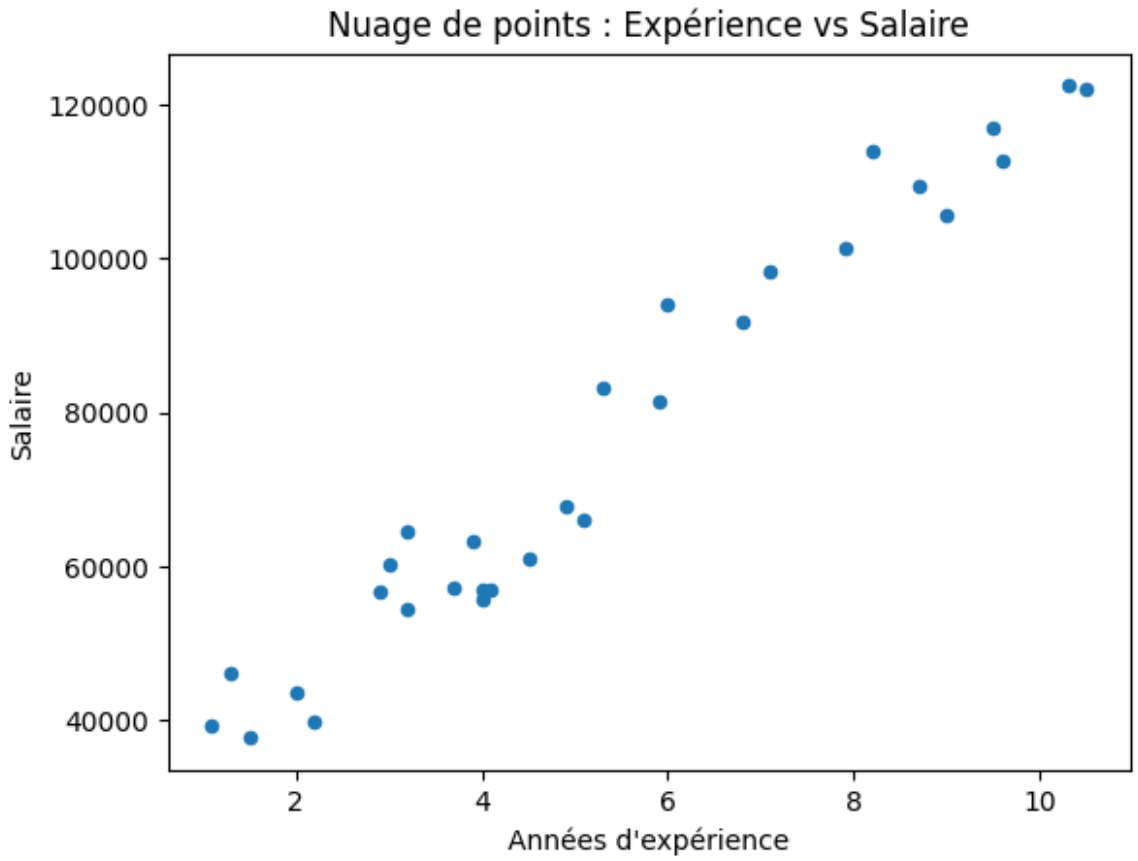


FIGURE 3 – Nuage des points pour le dataset Experience-Salaire

3.4 Nuages des points du deuxième data set selon les proprietes « Features »

Voici les visualisations demandees :

```
# Question 4 : Nuages des points pour le dataset assurance
# Selection des colonnes numeriques pour le scatter matrix
features = ['age', 'bmi', 'children', 'charges']
scatter_matrix(insurance[features], figsize=(10, 8), diagonal='hist', alpha
    =0.7)

plt.suptitle('Scatter Matrix : Assurance', fontsize=16)
plt.show()
```

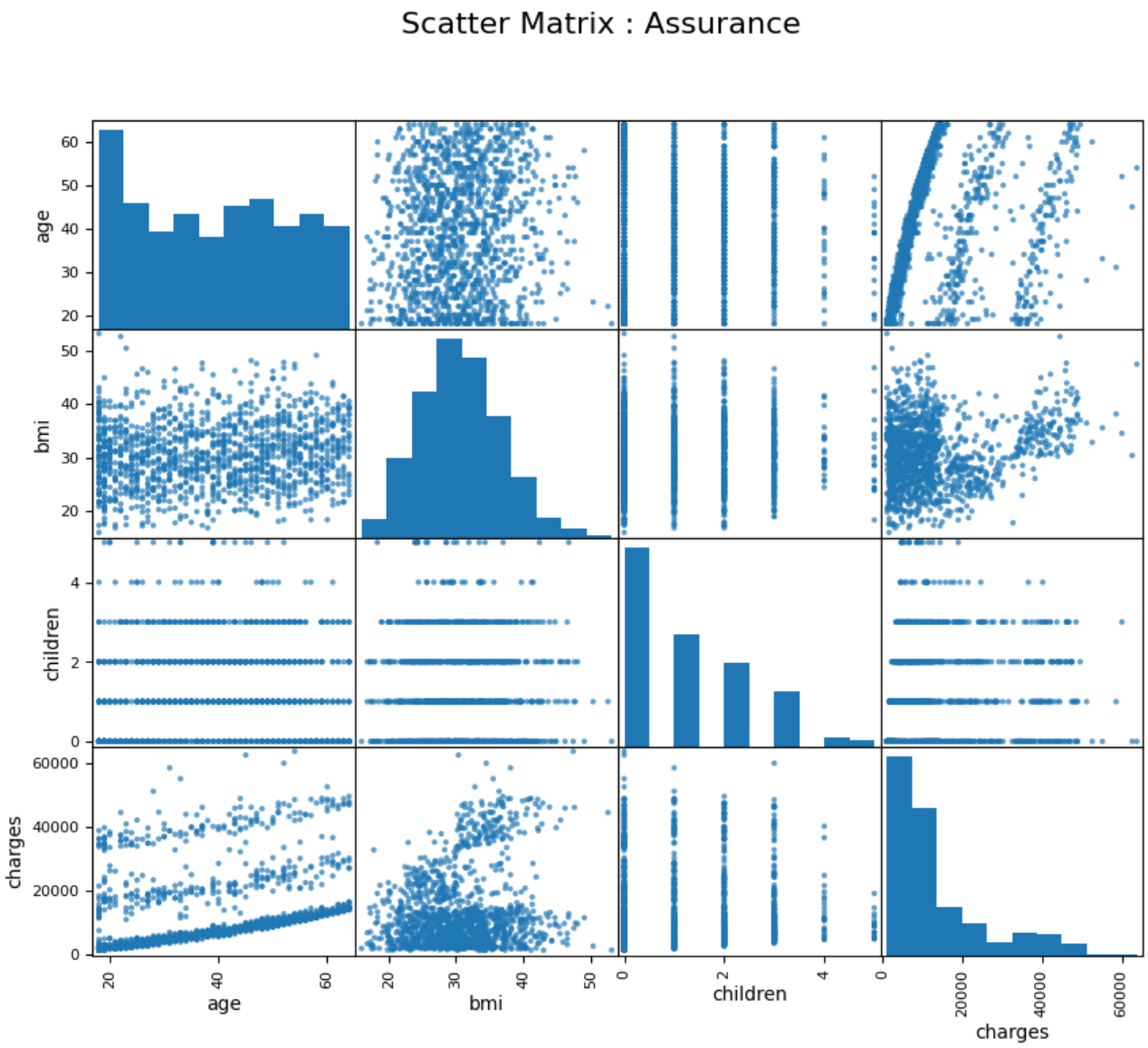


FIGURE 4 – Nuages des points du deuxième data set selon les proprietes « Features »

4 Partie 2 : Regression lineaire simple (Experience vs Salaire)

4.1 Entraînement de modèle

Pour predire les salaires en fonction des annees d'experience, nous avons applique une regression lineaire simple :

```
# Partie 2 : Regression lineaire simple (Experience vs Salaire)
# Question 1 : Entra ner le mod le

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

# Separation des donn es en variables explicatives (X) et cible (y)
X_exp = experience_salary[['YearsExperience']]
y_exp = experience_salary['Salary']

# Division des donn es en ensembles d'entra nement et de test
X_train, X_test, y_train, y_test = train_test_split(X_exp, y_exp, test_size
    =0.2, random_state=42)

# Creation et entra nement du mod le
model = LinearRegression()
model.fit(X_train, y_train)

# Affichage des coefficients du mod le
print(f"Coefficient de la regression : {model.coef_[0]}")
print(f"Intercept de la regression : {model.intercept_}")
```

Les visualisations et resultats d'evaluation sont inclus dans la suite du rapport.

— **Coefficient de la regression :** 9423.8153

— **Intercept de la regression :** 25321.5830

4.2 Prediction sur le dataset de test

```
# Question 2 : Prediction sur le dataset de test

y_pred = model.predict(X_test)

# Affichage des predictions
print("Predictions sur le dataset de test :")
print(y_pred)
```

— **Predictions sur le dataset de test :** [115790.21011287 71498.27809463 102596.86866063 75267.80422384 55477.79204548 60189.69970699]

4.3 Visualisation des resultats de la regression

```
# Question 3 : Visualisation des resultats de la regression
plt.scatter(X_test, y_test, color='blue', label='donn es reelles')
plt.plot(X_test, y_pred, color='red', label='Regression lineaire')
plt.title('Regression lineaire : Experience vs Salaire')
plt.xlabel('Annees d\'experience')
plt.ylabel('Salaire')
plt.legend()
plt.show()
```

FIGURE 5 – Visualisation des resultats de la regression

4.4 Evaluation du model

```python
# Question 4 : Evaluation du model
from sklearn.metrics import mean_squared_error, mean_absolute_error

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

# Affichage des resultats
print("-------- evaluation du mod le : --------")
print(f"Mean Squared Error (MSE) : {mse}")
print(f"Root Mean Squared Error (RMSE) : {rmse}")
print(f"Mean Absolute Error (MAE) : {mae}")
```

```
-------- Évaluation du modèle : --------
Mean Squared Error (MSE) : 49830096.85590839
Root Mean Squared Error (RMSE) : 7059.04362190151
Mean Absolute Error (MAE) : 6286.453830757749
```

FIGURE 6 – Evaluation du model

4.4.1 Interpretation

Les metriques calculees pour evaluer les performances du modèle sont les suivantes :

— **Mean Squared Error (MSE)** : 49,830,096.86

— **Root Mean Squared Error (RMSE)** : 7,059.04

— **Mean Absolute Error (MAE)** : 6,286.45

Ces resultats indiquent un ecart notable entre les predictions du modèle et les valeurs reelles. Une optimisation ou des ajustements supplementaires sont necessaires pour ameliorer la performance du modèle.

5 Partie 3 : Regression multiple (Assurance)

5.1 EDA (Exploratory Data Analysis)

EDA (Exploratory Data Analysis) :

```
# Question 1 : EDA  (Exploratory Data Analysis)

# Aper u des donn es
print("-------- Aper u des donn es : --------")
print(insurance.head())

# Informations generales sur les donn es
print("\n-------- Informations generales : --------")
print(insurance.info())
```

```
-------- Aperçu des données : --------
   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

-------- Informations générales : --------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
None
```

```
# Verification des valeurs nulles
print("\n-------- Nombre de valeurs nulles par colonne : --------")
print(insurance.isnull().sum())

# Statistiques descriptives
print("\n-------- Statistiques descriptives : --------")
print(insurance.describe())
```

```
-------- Nombre de valeurs nulles par colonne : --------
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64

-------- Statistiques descriptives : --------
                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
```

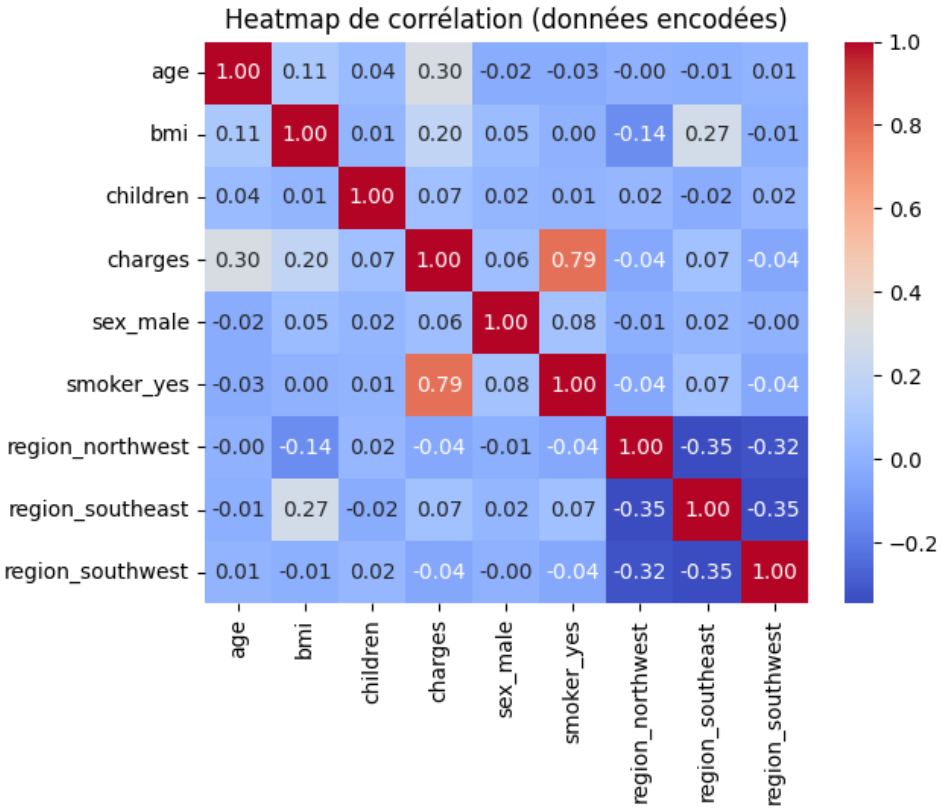
```python
# Analyse des variables categoriques
print("\n-------- Distribution des colonnes categoriques : --------")
print(insurance['sex'].value_counts())
print(insurance['smoker'].value_counts())
print(insurance['region'].value_counts())
```

```
-------- Distribution des colonnes catégoriques : --------
sex
male      676
female    662
Name: count, dtype: int64
smoker
no     1064
yes     274
Name: count, dtype: int64
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

```python
# Encoder les colonnes categoriques en variables numeriques
insurance_encoded = pd.get_dummies(insurance, drop_first=True)

# Matrice de correlation pour les variables numeriques
corr_matrix = insurance_encoded.corr()
print("\n-------- Matrice de correlation : --------")
print(corr_matrix)
```

```
8
9  # Visualisation : Heatmap de la matrice de correlation
10 sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
11 plt.title("Heatmap de correlation (donn es encodees)")
12 plt.show()
```

```
-------- Matrice de corrélation : --------
                        age       bmi  children   charges  sex_male  \
age                1.000000  0.109272  0.042469  0.299008 -0.020856
bmi                0.109272  1.000000  0.012759  0.198341  0.046371
children           0.042469  0.012759  1.000000  0.067998  0.017163
charges            0.299008  0.198341  0.067998  1.000000  0.057292
sex_male          -0.020856  0.046371  0.017163  0.057292  1.000000
smoker_yes        -0.025019  0.003750  0.007673  0.787251  0.076185
region_northwest  -0.000407 -0.135996  0.024806 -0.039905 -0.011156
region_southeast  -0.011642  0.270025 -0.023066  0.073982  0.017117
region_southwest   0.010016 -0.006205  0.021914 -0.043210 -0.004184

                  smoker_yes  region_northwest  region_southeast  \
age                -0.025019         -0.000407         -0.011642
bmi                 0.003750         -0.135996          0.270025
children            0.007673          0.024806         -0.023066
charges             0.787251         -0.039905          0.073982
sex_male            0.076185         -0.011156          0.017117
smoker_yes          1.000000         -0.036945          0.068498
region_northwest   -0.036945          1.000000         -0.346265
region_southeast    0.068498         -0.346265          1.000000
region_southwest   -0.036945         -0.320829         -0.346265

                  region_southwest
age                       0.010016
bmi                      -0.006205
children                  0.021914
charges                  -0.043210
sex_male                 -0.004184
smoker_yes               -0.036945
region_northwest         -0.320829
region_southeast         -0.346265
region_southwest          1.000000
```



5.2 Selectionner 3 proprietes selon leurs degree d'importance

```
# Question 2 : Selectionner 3 proprietes selon leurs degree  d importance


# 1. Calcul de la correlation avec la variable cible
correlations = insurance_encoded.corr()
charges_corr = correlations['charges'].sort_values(ascending=False)
print("------ Correlation avec 'charges' : ------")
print(charges_corr)

# 2. Affichage des 3 proprietes les plus correlees
selected_features = charges_corr.index[1:4]  # Exclut 'charges' lui-m me
print("\n------ Proprietes selectionnees selon la correlation : ------",
    selected_features.tolist())

# 3. Justification par Feature Importance avec un mod le
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import f_regression

X = insurance_encoded[selected_features]
y = insurance_encoded['charges']
```

```
20
21 # Entra nement du mod le
22 model = LinearRegression()
23 model.fit(X, y)
24
25 # Importance des variables
26 importances = pd.DataFrame({'Feature': selected_features, 'Importance':
       model.coef_})
27 print("\n------ Importance des variables selon la regression : ------")
28 print(importances)
```

```
------ Corrélation avec 'charges' : ------
charges             1.000000
smoker_yes          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

------ Propriétés sélectionnées selon la corrélation : ------ ['smoker_yes', 'age', 'bmi']

------ Importance des variables selon la régression : ------
      Feature    Importance
0  smoker_yes  23823.684495
1         age    259.547492
2         bmi    322.615133
```

5.3 Technique de standardisation

```
1 # Question 3 : technique de standardisation
2
3 from sklearn.preprocessing import StandardScaler
4
5 # Proprietes selectionnees
6 selected_features = ['age', 'bmi', 'smoker_yes']
7
8 # Standardisation des proprietes choisies
9 scaler = StandardScaler()
10 X_standardized = scaler.fit_transform(insurance_encoded[selected_features])
11
12 # Creation d'un DataFrame pour visualisation des resultats
13 X_standardized_df = pd.DataFrame(X_standardized, columns=selected_features)
14 print("\n--------- Proprietes standardisees : ---------")
15 print(X_standardized_df.head())
```

```
--------- Propriétés standardisées : ---------
        age       bmi  smoker_yes
0 -1.438764 -0.453320    1.970587
1 -1.509965  0.509621   -0.507463
2 -0.797954  0.383307   -0.507463
3 -0.441948 -1.305531   -0.507463
4 -0.513149 -0.292556   -0.507463
```

5.4 Entraîner le modèle

```python
# Question 4 : Entra ner le mod le

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

# Selection des variables explicatives et de la cible
X = X_standardized  # Variables standardisees selectionnees ('age', 'bmi', '
    smoker_yes')
y = insurance_encoded['charges']

# Division des 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)

# Creation et entra nement du mod le de regression lineaire
model = LinearRegression()
model.fit(X_train, y_train)

# Affichage des coefficients
print("\nCoefficients du mod le :")
for feature, coef in zip(['age', 'bmi', 'smoker_yes'], model.coef_):
    print(f"{feature}: {coef:.4f}")

print(f"\nIntercept : {model.intercept_:.4f}")
```

```
Coefficients du modèle :
age: 3643.3408
bmi: 1990.0105
smoker_yes: 9554.0342

Intercept : 13321.2930
```

5.5 Prediction

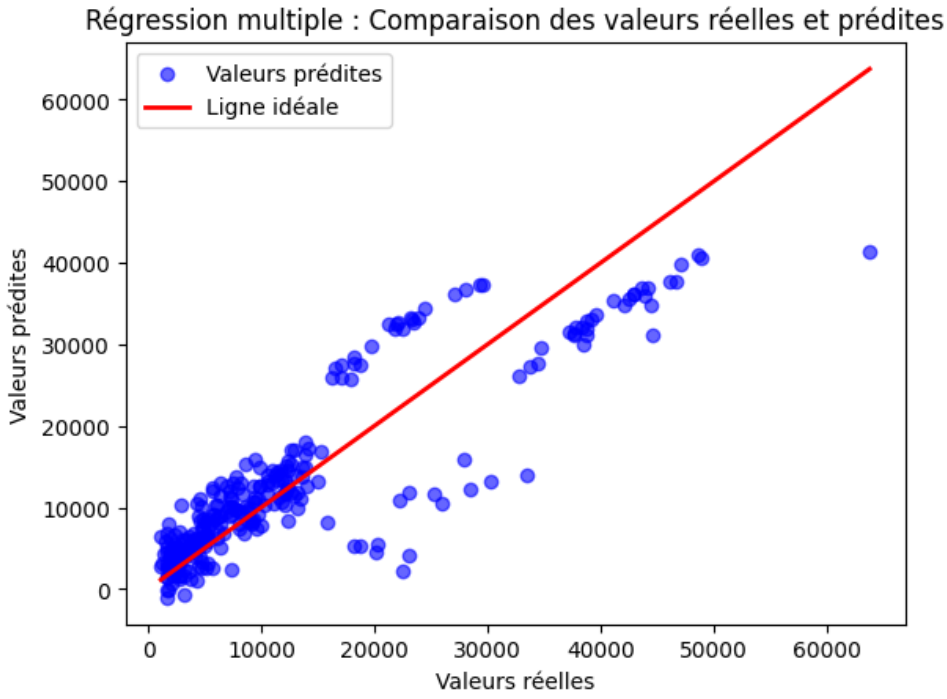
```python
# Question 5 : Predire

# Predire les donn es du dataset de test
y_pred = model.predict(X_test)

# Affichage des predictions et des valeurs reelles
predictions_df = pd.DataFrame({
    'Valeurs reelles': y_test.values,
    'Valeurs predites': y_pred
})

print("\nPredictions sur le dataset de test :")
print(predictions_df.head())
```

```
Prédictions sur le dataset de test :
   Valeurs réelles  Valeurs prédites
0       9095.06825       8184.041468
1       5272.17580       7431.001001
2      29330.98315      37346.437099
3       9301.89355       8629.528305
4      33750.29180      27316.654810
```

5.6 Visualisation

```python
# Question 6 : Visualisation

# Visualisation des valeurs reelles vs valeurs predites
plt.scatter(y_test, y_pred, alpha=0.6, color='blue', label='Valeurs predites
    ')
```

```
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', lw=2, label='Ligne ideale')
plt.title('Regression multiple : Comparaison des valeurs reelles et predites')
plt.xlabel('Valeurs reelles')
plt.ylabel('Valeurs predites')
plt.legend()
plt.show()
```



5.7 Evaluation de modèle en utilisant ces trois methodes

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np

# Calcul des metriques
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)

# Affichage des resultats
print("\n-------- evaluation du mod le : --------")
print(f"Mean Squared Error (MSE) : {mse:.2f}")
```

```
-------- Évaluation du modèle : --------
Mean Squared Error (MSE) : 34512843.88
Root Mean Squared Error (RMSE) : 5874.76
Mean Absolute Error (MAE) : 4260.56
Coefficient de détermination (R²) : 0.78
```

```python
print(f"Root Mean Squared Error (RMSE) : {rmse:.2f}")
print(f"Mean Absolute Error (MAE) : {mae:.2f}")
print(f"Coefficient de determination ( R  ) : {r2:.2f}")
```

5.7.1 Interpretation

Les metriques calculees pour evaluer les performances du modèle sont les suivantes :

— **Mean Squared Error (MSE)** : 34,512,843.88

— **Root Mean Squared Error (RMSE)** : 5,874.76

— **Mean Absolute Error (MAE)** : 4,260.56

— **Coefficient de determination (R^2)** : 0.78

Analyse des resultats : Le coefficient de determination ($R^2 = 0.78$) indique que 78% de la variance des données est expliquee par le modèle, ce qui reflète une bonne capacite predictive. Les erreurs (MSE, RMSE, MAE) sont raisonnablement faibles, mais une optimisation supplementaire pourrait encore ameliorer les predictions.

6 Partie 4 : Regression polynomiale (China GDP)

6.1 Comparaison des modèles

Nous avons entraîne et compare des modèles de regression lineaire et polynomiale pour predire l'evolution du PIB chinois :

```
# Question 1 : Entra ner les mod les
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split

# Separation des variables explicatives (X) et cible (y)
X_gdp = china_gdp[['Year']]
y_gdp = china_gdp['Value']

# Division des donn es en ensembles d'entra nement et de test
X_train, X_test, y_train, y_test = train_test_split(X_gdp, y_gdp, test_size
    =0.2, random_state=42)

# Mod le de regression lineaire
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

# Mod le de regression polynomiale (degre 4)
degree = 4
poly_features = PolynomialFeatures(degree=degree)
X_train_poly = poly_features.fit_transform(X_train)
X_test_poly = poly_features.transform(X_test)

poly_model = LinearRegression()
poly_model.fit(X_train_poly, y_train)

# Affichage des coefficients pour chaque mod le
print("\nCoefficients de la regression lineaire :")
print(f"Coef : {linear_model.coef_}, Intercept : {linear_model.intercept_}")

print("\nCoefficients de la regression polynomiale :")
print(f"Coef : {poly_model.coef_}, Intercept : {poly_model.intercept_}")
```

```
Coefficients de la régression linéaire :
Coef : [1.09727972e+11], Intercept : -216579790383883.8

Coefficients de la régression polynomiale :
Coef : [ 0.00000000e+00  6.04170555e+08  8.00117307e+11 -5.38980266e+08
  1.02111940e+05], Intercept : -5.2241796843888 93e+17
```

6.2 Prediction des données d'un data set de test pour les deux modèles

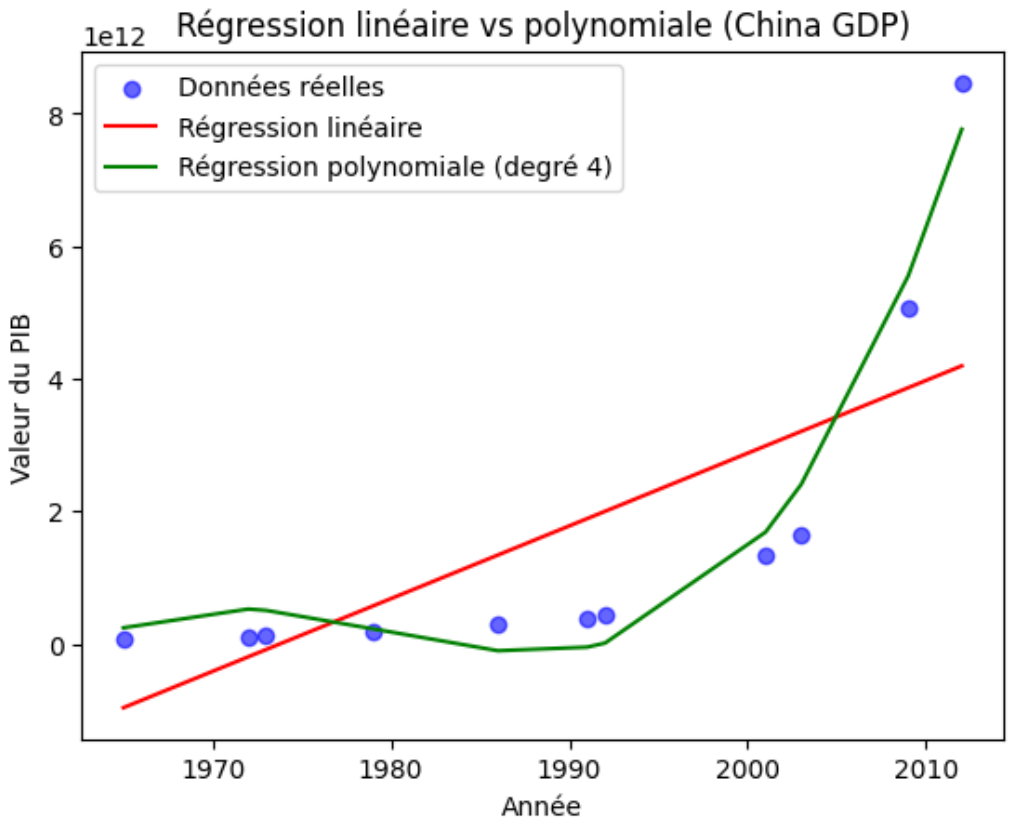
```python
# Question 2 : Prediction des donn es  d un  data set de test pour les deux
    mod les .

# Prediction avec le mod le de regression lineaire
linear_predictions = linear_model.predict(X_test)

# Prediction avec le mod le de regression polynomiale
poly_predictions = poly_model.predict(X_test_poly)

# Affichage des predictions
print("\n----- Predictions du mod le de regression lineaire : -----")
print(linear_predictions[:5])

print("\n----- Predictions du mod le de regression polynomiale : -----")
print(poly_predictions[:5])
```

```
----- Prédictions du modèle de régression linéaire : -----
[ 1.88860122e+12 -9.64326039e+11  1.99832920e+12 -8.65022655e+10
  5.71865565e+11]

----- Prédictions du modèle de régression polynomiale : -----
[-5.04822941e+10  2.41385689e+11  1.06164358e+10  5.04155164e+11
  2.24894391e+11]
```

6.3 Visualisation des resultats des deux modèles

```python
# Question 3 : Visualisation des resultats des deux mod les
# Trier les donn es de test et leurs predictions pour eviter des
   oscillations dans la visualisation
sorted_indices = np.argsort(X_test.values.flatten())
X_test_sorted = X_test.values[sorted_indices]
linear_predictions_sorted = linear_predictions[sorted_indices]
poly_predictions_sorted = poly_predictions[sorted_indices]
```

```
7
8  # Visualisation corrigee
9  plt.scatter(X_test, y_test, color='blue', label='donn es reelles', alpha
       =0.6)
10
11 # Tracer les lignes de prediction triees
12 plt.plot(X_test_sorted, linear_predictions_sorted, color='red', label='
       Regression lineaire')
13 plt.plot(X_test_sorted, poly_predictions_sorted, color='green', label='
       Regression polynomiale (degre 4)')
14
15 # Configuration du graphique
16 plt.title('Regression lineaire vs polynomiale (China GDP)')
17 plt.xlabel('Annee')
18 plt.ylabel('Valeur du PIB')
19 plt.legend()
20 plt.show()
```



6.4 Evaluation des deux modèles en utilisant ces trois methodes

```python
# Question 4 :  evaluation des deux mod les en utilisant ces trois methodes

from sklearn.metrics import r2_score

# evaluation du mod le de regression lineaire
linear_mse = mean_squared_error(y_test, linear_predictions)
linear_rmse = np.sqrt(linear_mse)
linear_mae = mean_absolute_error(y_test, linear_predictions)
linear_r2 = r2_score(y_test, linear_predictions)

# evaluation du mod le de regression polynomiale
poly_mse = mean_squared_error(y_test, poly_predictions)
poly_rmse = np.sqrt(poly_mse)
poly_mae = mean_absolute_error(y_test, poly_predictions)
poly_r2 = r2_score(y_test, poly_predictions)

# Affichage des resultats
print("\n----- evaluation du mod le de regression lineaire : -----")
print(f"Mean Squared Error (MSE) : {linear_mse:.2f}")
print(f"Root Mean Squared Error (RMSE) : {linear_rmse:.2f}")
print(f"Mean Absolute Error (MAE) : {linear_mae:.2f}")
print(f"Coefficient de determination ( R  ) : {linear_r2:.2f}")

print("\n----- evaluation du mod le de regression polynomiale : -----")
print(f"Mean Squared Error (MSE) : {poly_mse:.2f}")
print(f"Root Mean Squared Error (RMSE) : {poly_rmse:.2f}")
print(f"Mean Absolute Error (MAE) : {poly_mae:.2f}")
print(f"Coefficient de determination ( R  ) : {poly_r2:.2f}")
```

```
----- Évaluation du modèle de régression linéaire : -----
Mean Squared Error (MSE) : 2909722992049542559432704.00
Root Mean Squared Error (RMSE) : 1705791016522.70
Mean Absolute Error (MAE) : 1341446144991.78
Coefficient de détermination (R²) : 0.56

----- Évaluation du modèle de régression polynomiale : -----
Mean Squared Error (MSE) : 2075341301920927685017600.00
Root Mean Squared Error (RMSE) : 455559140169.63
Mean Absolute Error (MAE) : 413445600606.50
Coefficient de détermination (R²) : 0.97
```

6.4.1 Interpretation

Les resultats de l'evaluation des deux modèles sont les suivants :

— **Modèle de regression lineaire** :
  — **Mean Squared Error (MSE)** : 2,909,722,992,049,542,559,432,704.00
  — **Root Mean Squared Error (RMSE)** : 1,705,791,016,522.70
  — **Mean Absolute Error (MAE)** : 1,341,446,144,991.78
  — **Coefficient de determination (R^2)** : 0.56
— **Modèle de regression polynomiale** :
  — **Mean Squared Error (MSE)** : 207,534,130,192,092,768,501,760.00
  — **Root Mean Squared Error (RMSE)** : 455,559,140,169.63
  — **Mean Absolute Error (MAE)** : 413,445,600,606.50
  — **Coefficient de determination (R^2)** : 0.97

Analyse des resultats : Le modèle de regression lineaire presente un $R^2 = 0.56$, ce qui signifie qu'il explique seulement 56% de la variance des données. Ses erreurs (MSE, RMSE, MAE) sont relativement elevees, indiquant une performance modeste.

En revanche, le modèle de regression polynomiale obtient un $R^2 = 0.97$, ce qui montre qu'il explique 97% de la variance des données, ce qui est un excellent resultat. Ses erreurs sont bien plus faibles que celles du modèle lineaire, ce qui suggère qu'il offre une meilleure prediction.

7 Conclusion

Cet atelier a permis d'explorer plusieurs types de regression et d'evaluer leur performance sur differents jeux de données. Les resultats obtenus montrent l'importance de choisir le modèle approprie pour les données disponibles.

Dans le cadre de cette etude, nous avons compare deux modèles de regression : la regression lineaire et la regression polynomiale, en utilisant des metriques de performance telles que le Mean Squared Error (MSE), le Root Mean Squared Error (RMSE), le Mean Absolute Error (MAE) et le Coefficient de determination R^2. Ces metriques nous ont permis de quantifier la precision et la qualite des predictions des modèles.

Le modèle de regression lineaire a montre une capacite d'ajustement relativement faible, avec un $R^2 = 0.56$, ce qui indique que plus de 40% de la variance des données reste inexpliquee. De plus, les erreurs (MSE, RMSE, MAE) elevees suggèrent que ce modèle n'est pas suffisamment adapte aux données analysees, ce qui peut être attribue à sa simplicite et à sa tendance à sous-estimer les relations complexes entre les variables.

En revanche, la regression polynomiale a demontre une performance bien superieure, avec un $R^2 = 0.97$, indiquant que presque 97% de la variance des données a ete expliquee par ce modèle. Les erreurs plus faibles (MSE, RMSE, MAE) confirment que le modèle polynomiale est capable de capturer de manière plus precise les relations non lineaires et complexes dans les données. Cela demontre l'importance de la flexibilite du modèle polynomial, bien que sa complexite puisse entraîner un surajustement (overfitting) dans certains cas.

Ces resultats renforcent l'idee que le choix du modèle depend non seulement de la nature des données, mais aussi du compromis entre la simplicite du modèle et sa capacite à capturer les relations sous-jacentes. Il est essentiel de tester plusieurs modèles et d'evaluer leur performance sur des metriques adaptees pour determiner la meilleure approche à adopter dans un contexte donne.

Ainsi, ce travail souligne l'importance de l'analyse prealable des données et du choix judicieux du modèle afin d'obtenir des resultats fiables et pertinents.

8 References

— Becoming Human - Linear Regression in Python

— Kaggle - Tutorial on Multiple Regression

— StackAbuse - Multiple Linear Regression