

COMPTE RENDU DU TP3

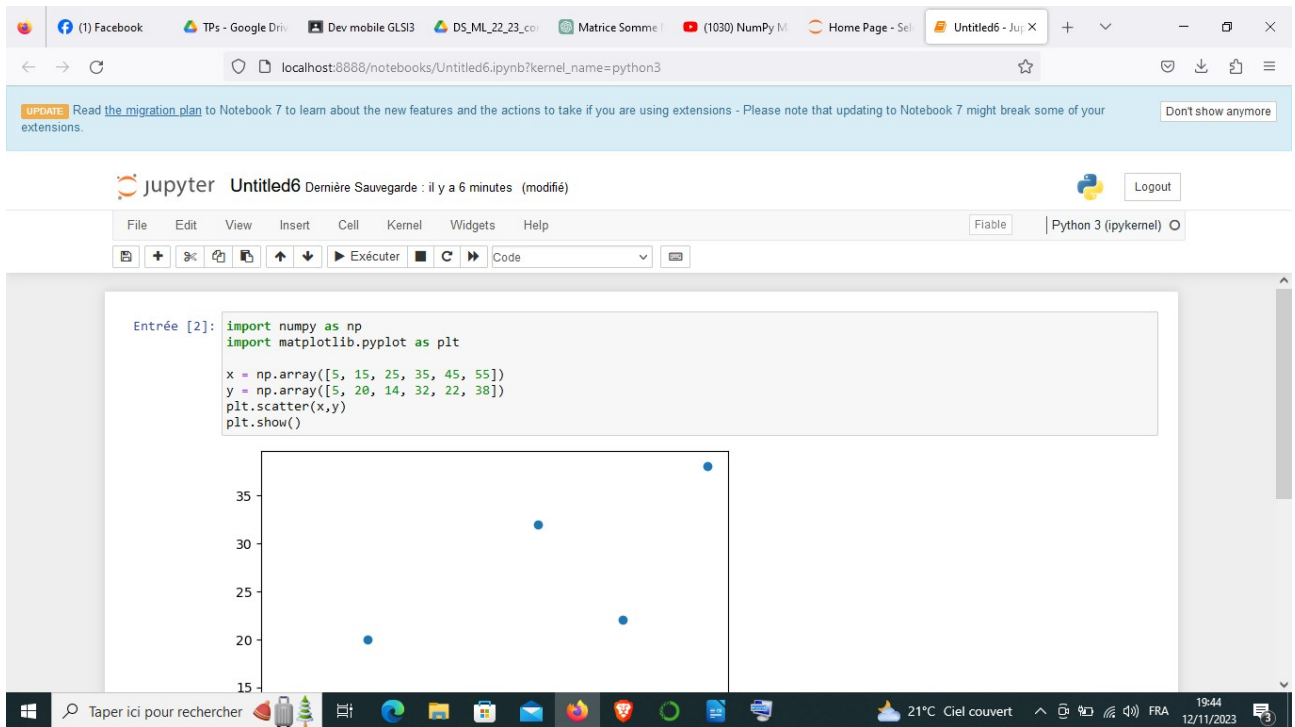
Realisé par Barry Alpha Mamadou Douah

Année Universitaire 2023-2024

partie1

1) Creation de deux tableaux x et y :

2) representation des nuages de point :

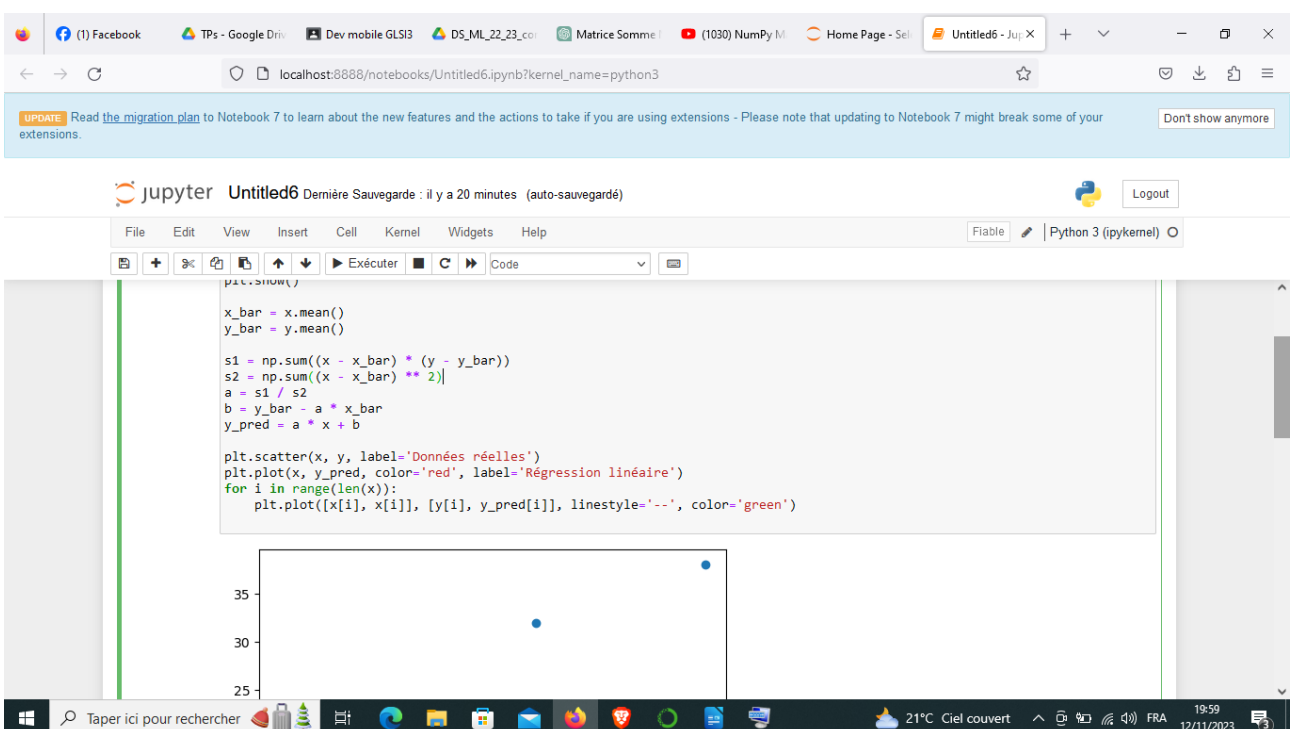


3) Calculer et afficher les valeurs a et b en utilisant la méthode des moindres carrés.

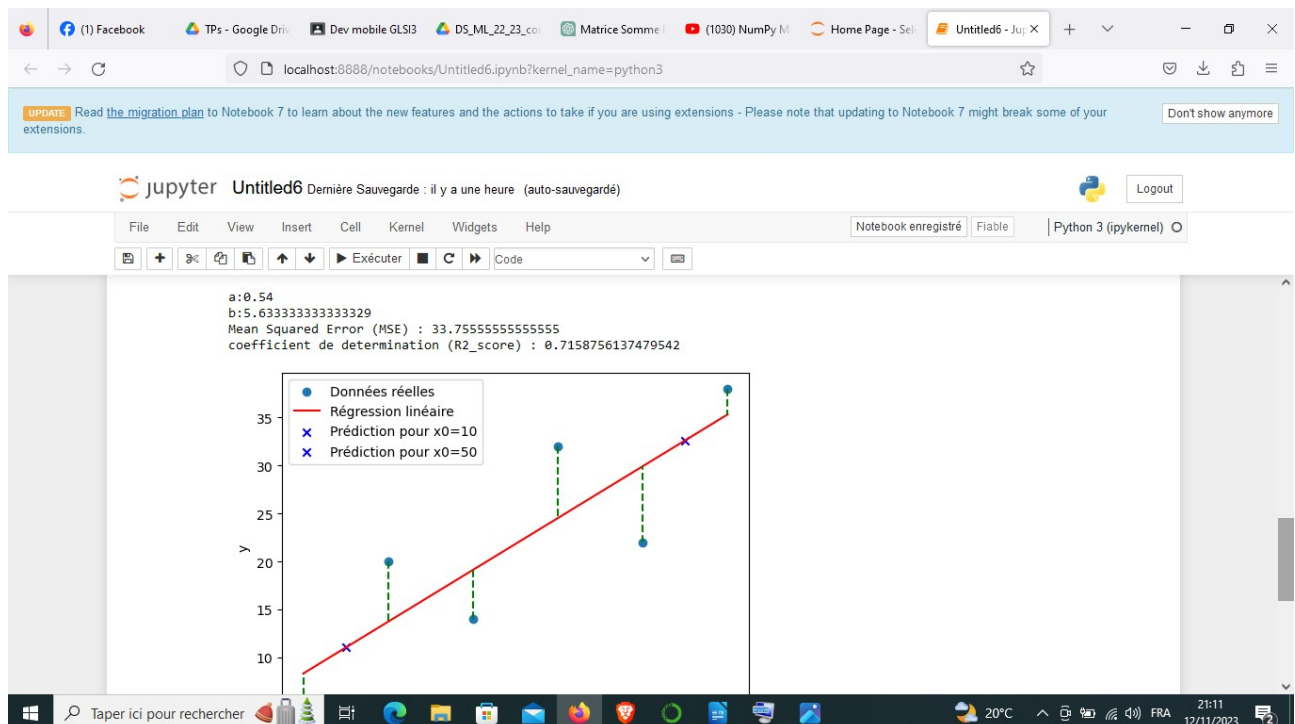
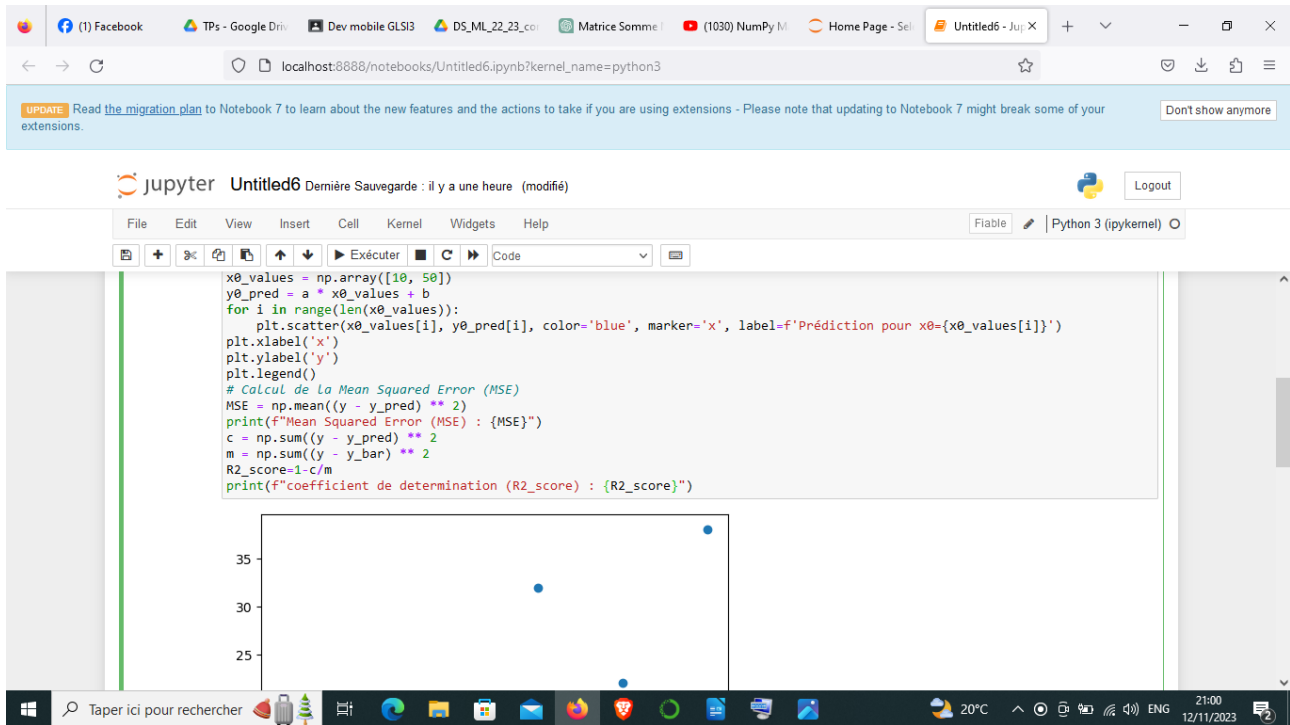
4) Calculer et afficher les valeurs prédites (\hat{y}_i) dans un tableau ypredit.

5) Représenter la droite de régression dans le même graphique.

6) Représenter les valeurs prédites sur la droite de régression.

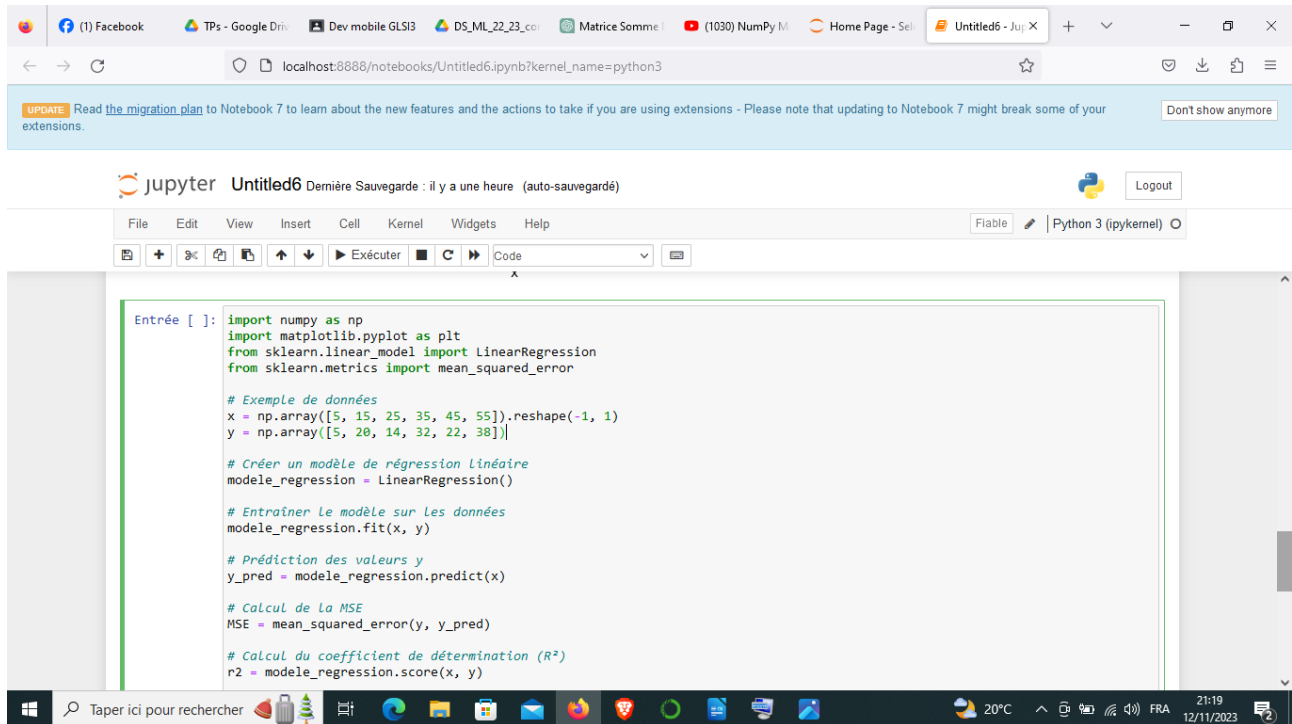


- 7) Calculer et afficher la prédiction pour $x_0=10$ et $x_0=50$.
- 8) Ajouter une légende à votre graphique.
- 9) Calculer et afficher la valeur MSE (Mean Squared Error) ainsi que le coefficient de détermination R^2



Partie2

Implementation de la partie 1 avec sk learn :



```
Entrée [ ]: import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Exemple de données
x = np.array([5, 15, 25, 35, 45, 55]).reshape(-1, 1)
y = np.array([5, 20, 14, 32, 22, 38])

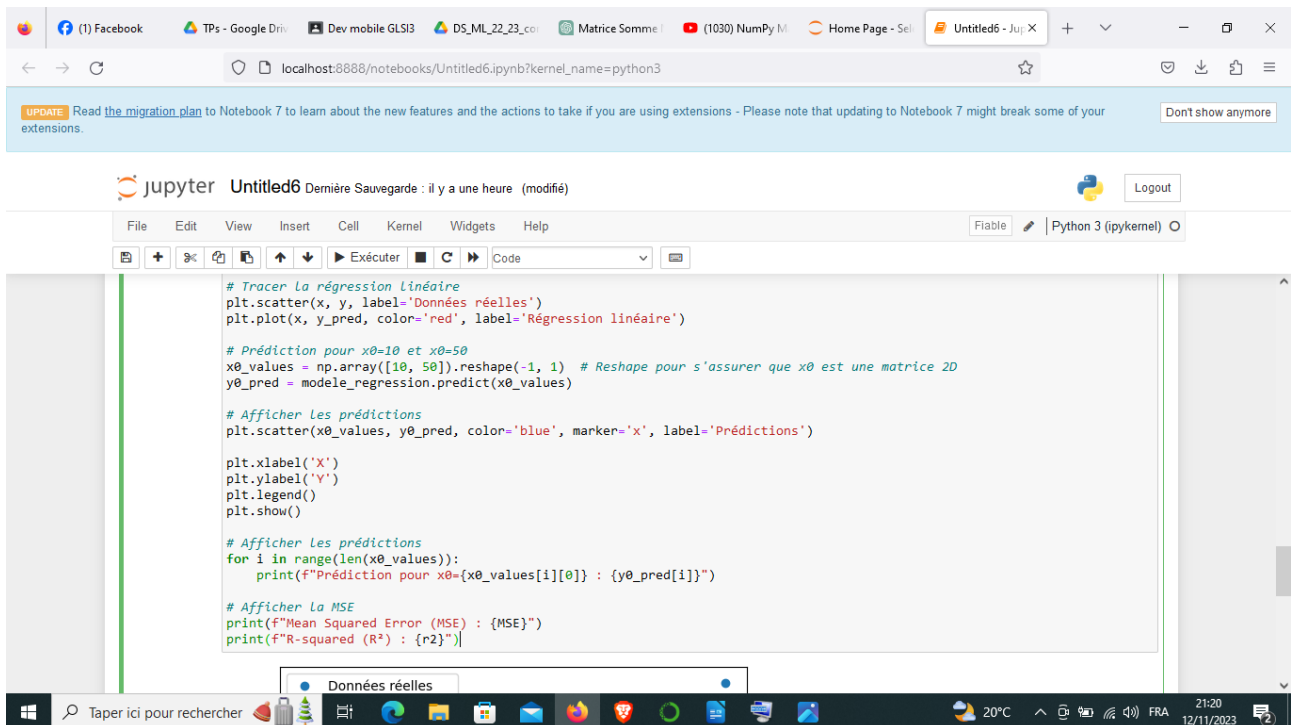
# Créer un modèle de régression linéaire
modele_regression = LinearRegression()

# Entraîner le modèle sur les données
modele_regression.fit(x, y)

# Prédiction des valeurs y
y_pred = modele_regression.predict(x)

# Calcul de la MSE
MSE = mean_squared_error(y, y_pred)

# Calcul du coefficient de détermination (R²)
r2 = modele_regression.score(x, y)
```



```
# Tracer la régression linéaire
plt.scatter(x, y, label='Données réelles')
plt.plot(x, y_pred, color='red', label='Régression linéaire')

# Prédiction pour x0=10 et x0=50
x0_values = np.array([10, 50]).reshape(-1, 1) # Reshape pour s'assurer que x0 est une matrice 2D
y0_pred = modele_regression.predict(x0_values)

# Afficher les prédictions
plt.scatter(x0_values, y0_pred, color='blue', marker='x', label='Prédictions')

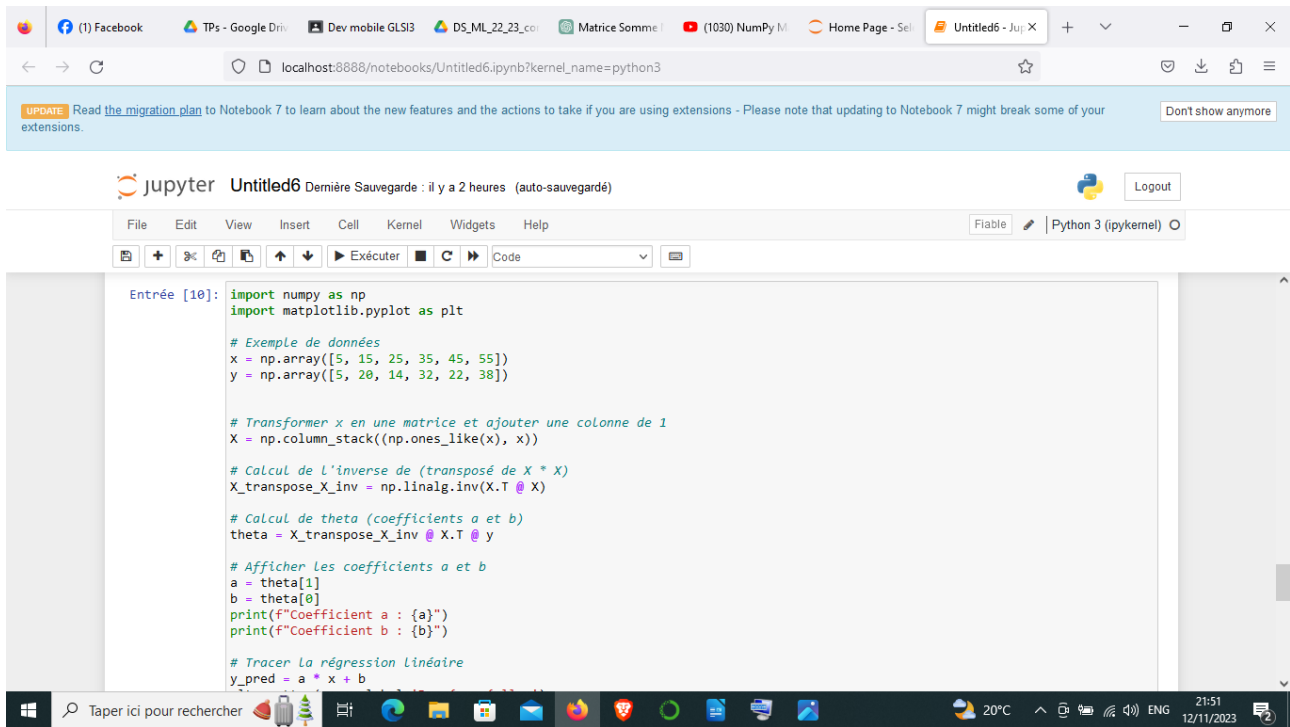
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()

# Afficher les prédictions
for i in range(len(x0_values)):
    print(f"Prédiction pour x0={x0_values[i][0]} : {y0_pred[i]}")

# Afficher la MSE
print(f"Mean Squared Error (MSE) : {MSE}")
print(f"R-squared (R²) : {r2}")
```

Partie3

Implementation par l'équation normale



The screenshot shows a Jupyter Notebook titled 'Untitled6' with a Python 3 kernel. The code in the cell is as follows:

```
Entrée [10]: import numpy as np
import matplotlib.pyplot as plt

# Exemple de données
x = np.array([5, 15, 25, 35, 45, 55])
y = np.array([5, 20, 14, 32, 22, 38])

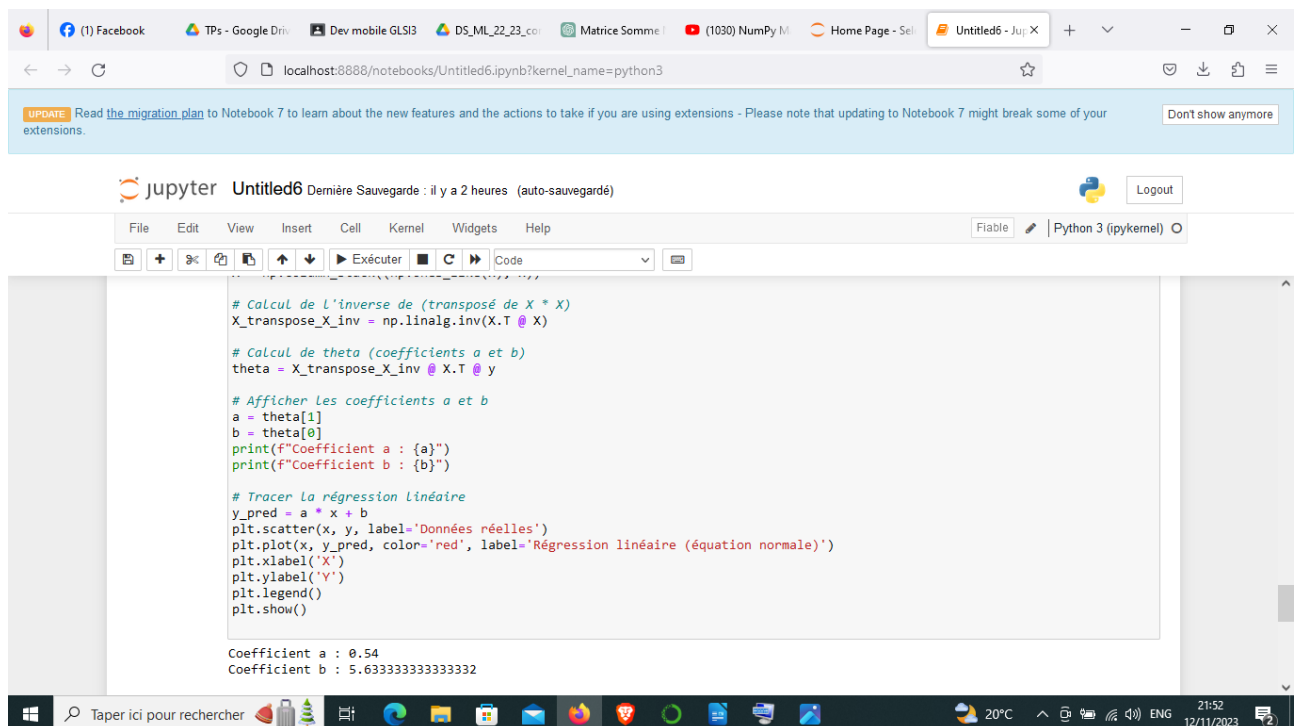
# Transformer x en une matrice et ajouter une colonne de 1
X = np.column_stack((np.ones_like(x), x))

# Calcul de l'inverse de (transposé de X * X)
X_transpose_X_inv = np.linalg.inv(X.T @ X)

# Calcul de theta (coefficients a et b)
theta = X_transpose_X_inv @ X.T @ y

# Afficher Les coefficients a et b
a = theta[1]
b = theta[0]
print(f"Coefficient a : {a}")
print(f"Coefficient b : {b}")

# Tracer La régression linéaire
y_pred = a * x + b
```



The screenshot shows the same Jupyter Notebook after execution. The code is identical to the previous one, but it now includes the plotting and printing of the results:

```
# Calcul de l'inverse de (transposé de X * X)
X_transpose_X_inv = np.linalg.inv(X.T @ X)

# Calcul de theta (coefficients a et b)
theta = X_transpose_X_inv @ X.T @ y

# Afficher Les coefficients a et b
a = theta[1]
b = theta[0]
print(f"Coefficient a : {a}")
print(f"Coefficient b : {b}")

# Tracer La régression linéaire
y_pred = a * x + b
plt.scatter(x, y, label='Données réelles')
plt.plot(x, y_pred, color='red', label='Régression linéaire (équation normale)')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()
```

The output of the cell is displayed below the code:

```
Coefficient a : 0.54
Coefficient b : 5.633333333333332
```