Premier modèle IA

Julien Ronan

Rappel sur les régressions

En mathématiques, la **régression** recouvre plusieurs méthodes d'analyse statistique permettant d'approcher une variable à partir d'autres qui lui sont corrélées.

Une régression linéaire cherche à établir une relation linéaire entre une variable, dite expliquée, et une ou plusieurs variables, dites explicatives.

La régression linéaire multiple étend la simple, pour décrire les variations d'une variable endogène associée aux variations de plusieurs variables exogènes.

La régression polynomiale est une régression multiple faite en utilisant une variable aléatoire explicative. Les observations sont construite à partir des puissances de cette seule variable.

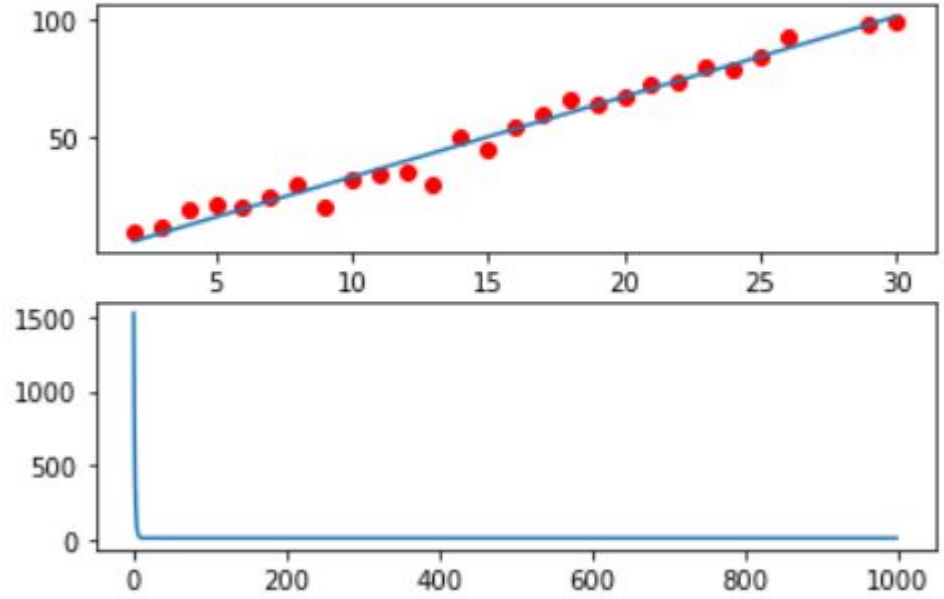
Fonctions :

```python
def model(X, theta):
    return X.dot(theta)

def function_count(X, y, theta):
    m = len(y)
    return 1/(2*m) * np.sum((model(X, theta) - y)**2)

def grad(X, y, theta):
    m = len(y)
    return 1/m * X.T.dot(model(X, theta) - y)

def gradient_descent(X, y, theta, learning_rate, n_iterations):
    # création d'un tableau de stockage pour enregistrer l'évolution du Cout du modele
    cost_history = np.zeros(n_iterations)

    for i in range(0, n_iterations):
        theta = theta - learning_rate * grad(X, y, theta) # mise a jour du parametre theta (formule du gradient descent)
        cost_history[i] = function_count(X, y, theta) # on enregistre la valeur du Cout au tour i dans cost_history[i]

    return theta, cost_history

def correlation_coeff(actual,predict):
    corr_matrix = np.corrcoef(actual, predict)
    corr = corr_matrix[0,1]
    return corr**2
```

Resultats :

Regression linéaire

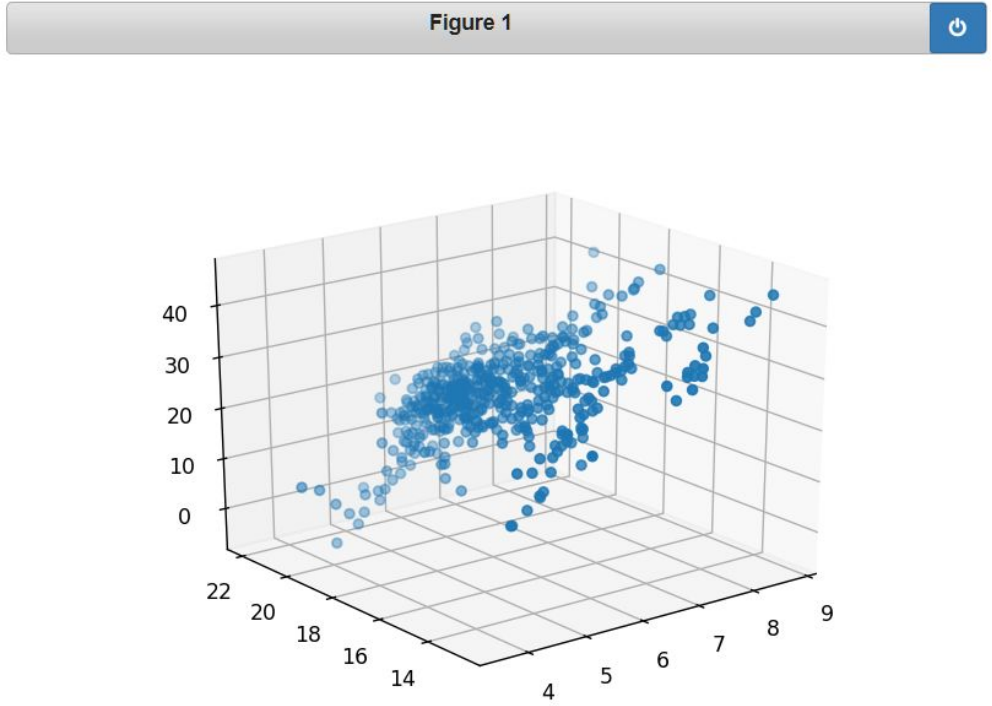
```
MEAN_squared :
20.005563133528515
```



Resultats

Regression linéaire multiple

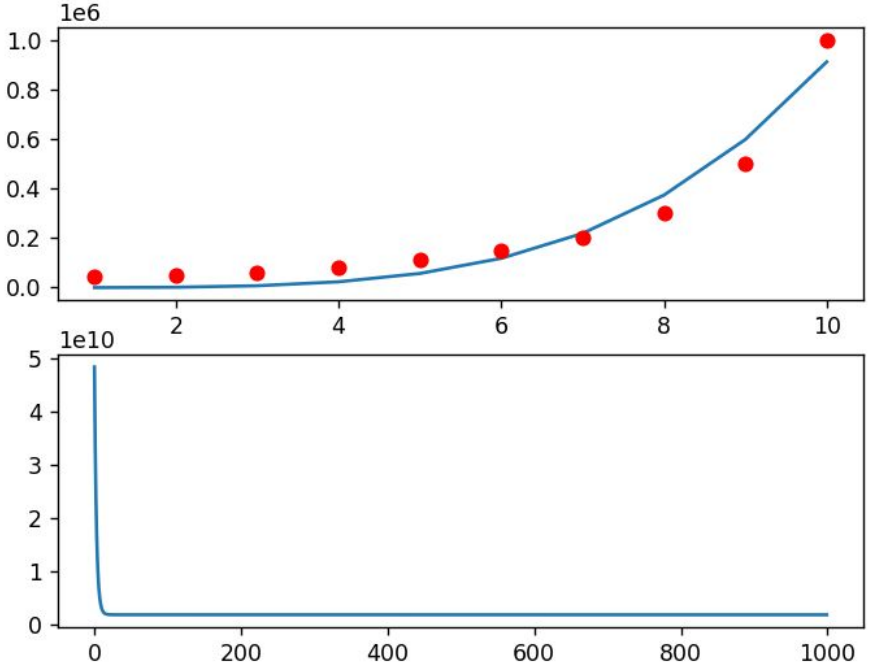
```
MEAN_squared :
27.88930165516501
```



Resultats

Regression polynomiale

```
MEAN_squared :
3220704741.757154
```



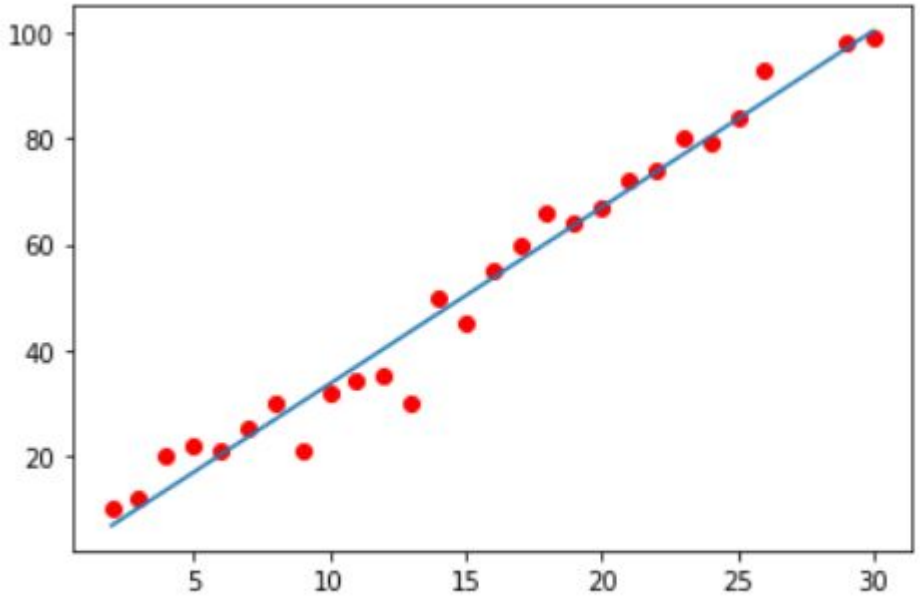
Analyse :

Regression linéaire : Coeff R2 : 0.9733202350434533

Regression linéaire multiple: Coeff R2 : 0.6663272451163134

Regression polynomiale: Coeff R2 : 0.9599718386130459

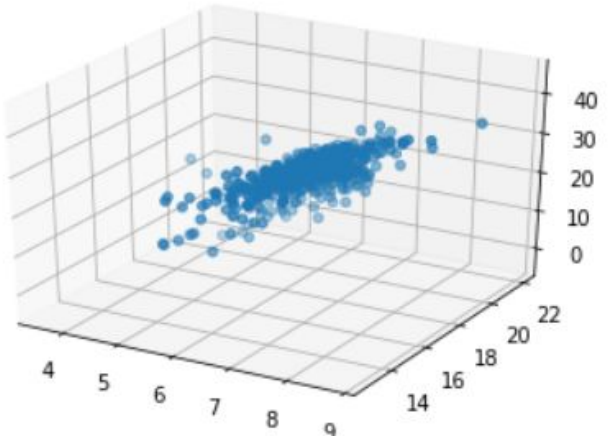
Presentation avec Scikit-learn




```
0.9732132549497465
20.461930334507777
```

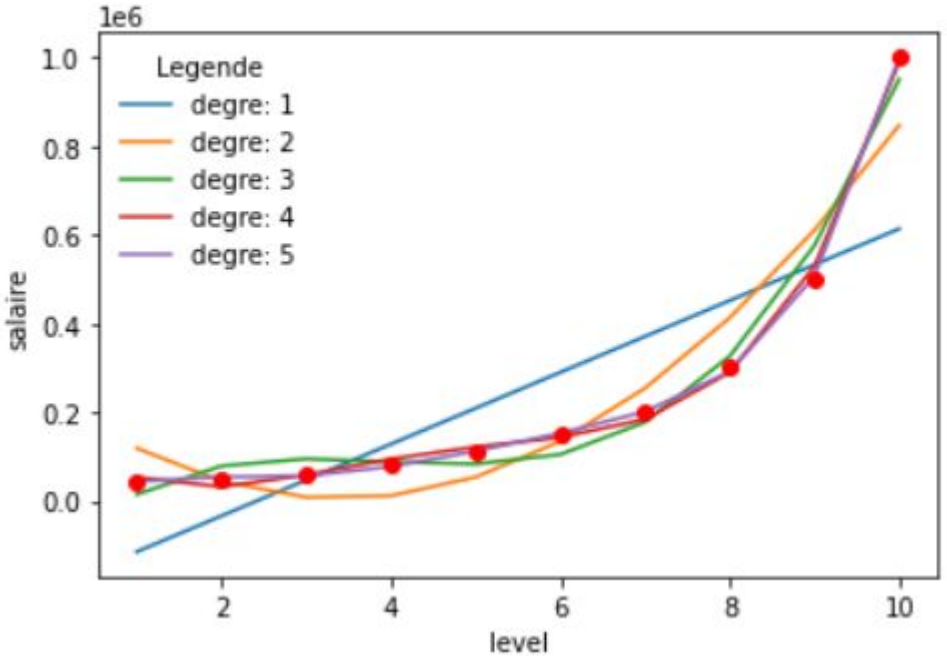
Regression multiple avec Scikit Learn

```
0.6892120408933597
26.745087563008017

<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x26605b66d60>
```



Regression polynomiale avec scikit learn

```
0.9997969027099755
```



Comparaisons :

coeff R2	Méthode normale	Scikit Learn
Linéaire	97%	97%
multiple	66%	68%
polynomiale	95%	99%

Conclusion

Qu’avez-vous appris ?

Comment ?

Des difficultés ?

Comment vous vous sentez après ce projet ?