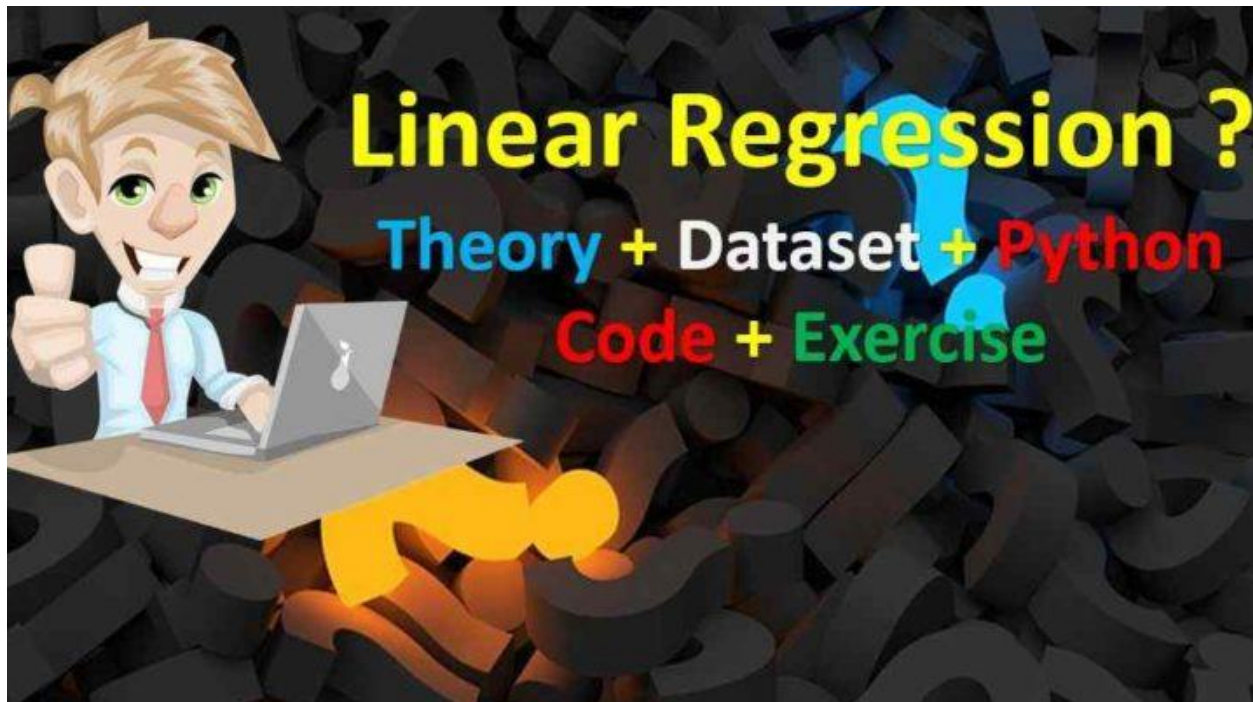


Prediction du salaire



Un Exemple de régression simple



Objectifs pédagogiques

Objectif 1 : Préparer un jeu de donnée pour l'analyse.

Objectif 2 : Construire un modèle de prédiction de salaire à partir d'un jeu de données.

Objectif 3 : interpréter la qualité du modèle de prédiction.

Objectif 4 : Prédire le salaire d'une nouvelle entrée

Prérequis

Installation de librairie *sklearn*

Introduction

La régression linéaire est une partie de l'apprentissage automatique supervisé. La régression linéaire est la meilleure ligne d'ajustement pour le point de données donné. Elle fait référence à une relation linéaire (ligne droite) entre les variables indépendantes et les variables dépendantes.

La régression linéaire simple implique deux variables où une valeur indépendante (colonne X) et une valeur dépendante (colonne Y). Dans cet article, nous essayons de prédire le salaire des employés en fonction du nombre d'années d'expérience. Nous implémentons un code python à l'aide de la bibliothèque d'apprentissage machine Scikit-Learn. Tout le code est exécuté sous Python.

Plan de travail

1. Importez la bibliothèque Python nécessaire.
2. Stocker les données (csv) dans une variable.
3. Fractionner les données en données d'apprentissage et de test.
4. Ajuster le modèle de régression aux données d'apprentissage et prévoir les données de test.
5. Visualisez le tracé entre les données d'entraînement / test par rapport à la ligne de régression.
6. Calculer l'erreur quadratique moyenne et sa racine carrée (MSE et RMSE).

Chargement de corpus

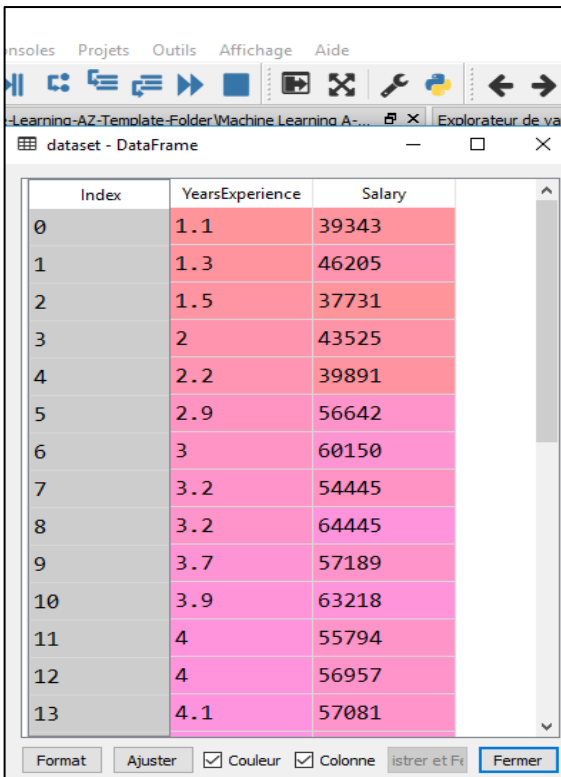
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
# Importing the dataset
```

```
dataset = pd.read_csv('Salary_Data.csv')
```

```
X = dataset.iloc[:, :-1].values
```

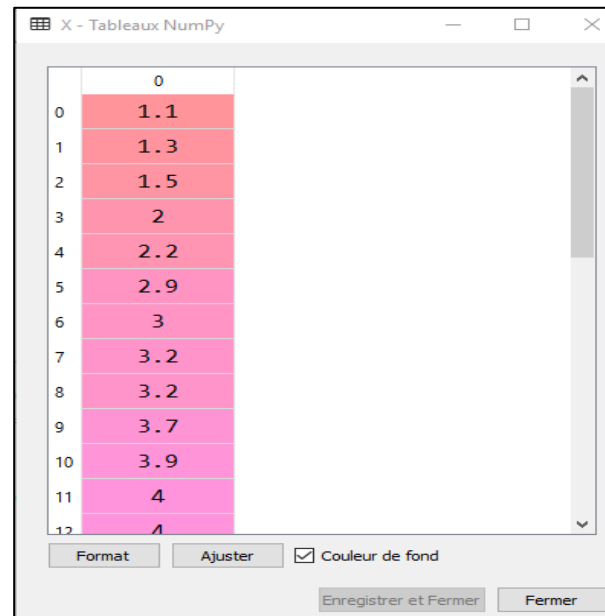
```
y = dataset.iloc[:, 1].values
```



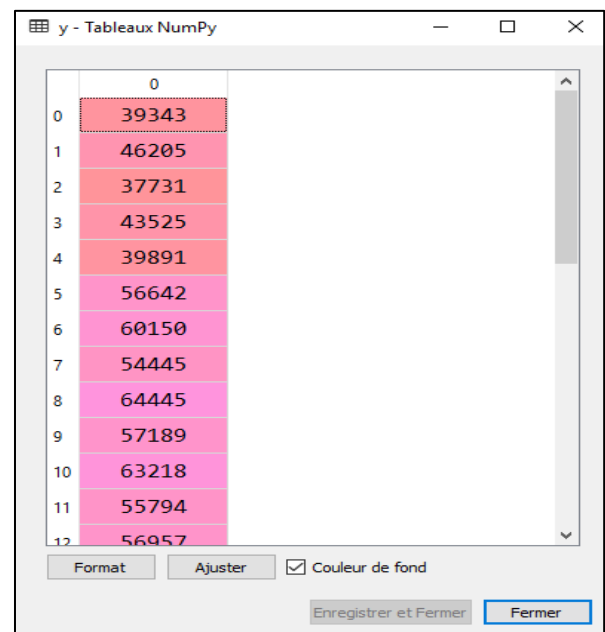
The screenshot shows a Jupyter Notebook interface with a toolbar at the top. Below the toolbar, there's a tab labeled 'dataset - DataFrame'. The main area displays a table with three columns: 'Index', 'YearsExperience', and 'Salary'. The data is as follows:

Index	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2	43525
4	2.2	39891
5	2.9	56642
6	3	60150
7	3.2	54445
8	3.2	64445
9	3.7	57189
10	3.9	63218
11	4	55794
12	4	56957
13	4.1	57081

At the bottom of the table, there are buttons for 'Format', 'Ajuster', 'Couleur', 'Colonne', 'Enregistrer et Fermer', and 'Fermer'.



The screenshot shows a window titled 'X - Tableaux NumPy'. It displays a 1D NumPy array of 'YearsExperience' values. The values are: 1.1, 1.3, 1.5, 2, 2.2, 2.9, 3, 3.2, 3.2, 3.7, 3.9, 4. The window has a scrollbar on the right and buttons at the bottom: 'Format', 'Ajuster', 'Couleur de fond' (checked), 'Enregistrer et Fermer', and 'Fermer'.



The screenshot shows a window titled 'y - Tableaux NumPy'. It displays a 1D NumPy array of 'Salary' values. The values are: 39343, 46205, 37731, 43525, 39891, 56642, 60150, 54445, 64445, 57189, 63218, 55794, 56957. The window has a scrollbar on the right and buttons at the bottom: 'Format', 'Ajuster', 'Couleur de fond' (checked), 'Enregistrer et Fermer', and 'Fermer'.

RQ : essayer de visualiser vos données avec plt

Fractionner les données

Avant d'entamer cette partie, nous allons tout d'abord répartir notre corpus en données d'apprentissage et en données de test.

1. La fonction **`train_test_split`** de librairie **`sklearn.model_selection`** fait l'affaire.

```
# Importing the dataset

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size = 1/3, random_state = 0)
```

- RQ : Essayer de visualiser vos données X_train, y_train

Ajuster le modèle de régression aux données d'apprentissage et prévoir les données de test.

Maintenant que nous avons nos variables et nos données d'apprentissage, nous pouvons former un modèle pour essayer de prédire le salaire.

Les instructions ci-dessus sert à créer un modèle de Régression Linéaire.

`Scikit-learn`, notre Framework de prédilection, fournit d'ailleurs une aide au choix de l'algorithme :

```
# Fitting Simple Linear Regression to the Training set

from sklearn.linear_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X_train, y_train)
```

- Tester le modèle et afficher y_pred, y_test

```
# Predicting the Test set results

y_pred = regressor.predict(X_test)
```

Visualiser le résultat

Visualising the Training set results

```
plt.scatter(X_train, y_train, color = 'red')  
plt.plot(X_train, regressor.predict(X_train),  
color='blue')  
plt.title('Salary vs Experience (Training set)')  
plt.xlabel('Years of Experience')  
plt.ylabel('Salary')  
plt.show()
```



Visualising the Test set results

```
plt.scatter(X_test, y_test, color = 'red')  
plt.plot(X_train, regressor.predict(X_train), color =  
'blue')  
plt.title('Salary vs Experience (Test set)')  
plt.xlabel('Years of Experience')  
plt.ylabel('Salary')  
plt.show()
```



Prédiction des nouveaux salaires

Une fois le modèle est créé, on passe à la prédiction :

```
regressor.predict([[5.6]])
```

```

Explorateur de variables | Explorateur de fichiers | Aide
Console IPython
Console 1/A
regrandexperience

In [60]: regressor.predict([[5.6]])
Out[60]: array([79153.46992552])

In [61]: regressor.predict([[11]])
Out[61]: array([129621.55911838])

```

Pour calculer la performance

```
from sklearn.metrics import r2_score
```

```
coefficient_of_dermination = mean_squared_error(y_test,
y_pred)
```

```
coefficient_of_dermination = r2_score(y_test, y_pred)
```

```
print(coefficient_of_dermination)
```