Scikit-Learn

Réalisé par :

Ayoub Oukchiren

Encadré par :

Mr. Mohammed Reda

Plan

Introduction

Définition

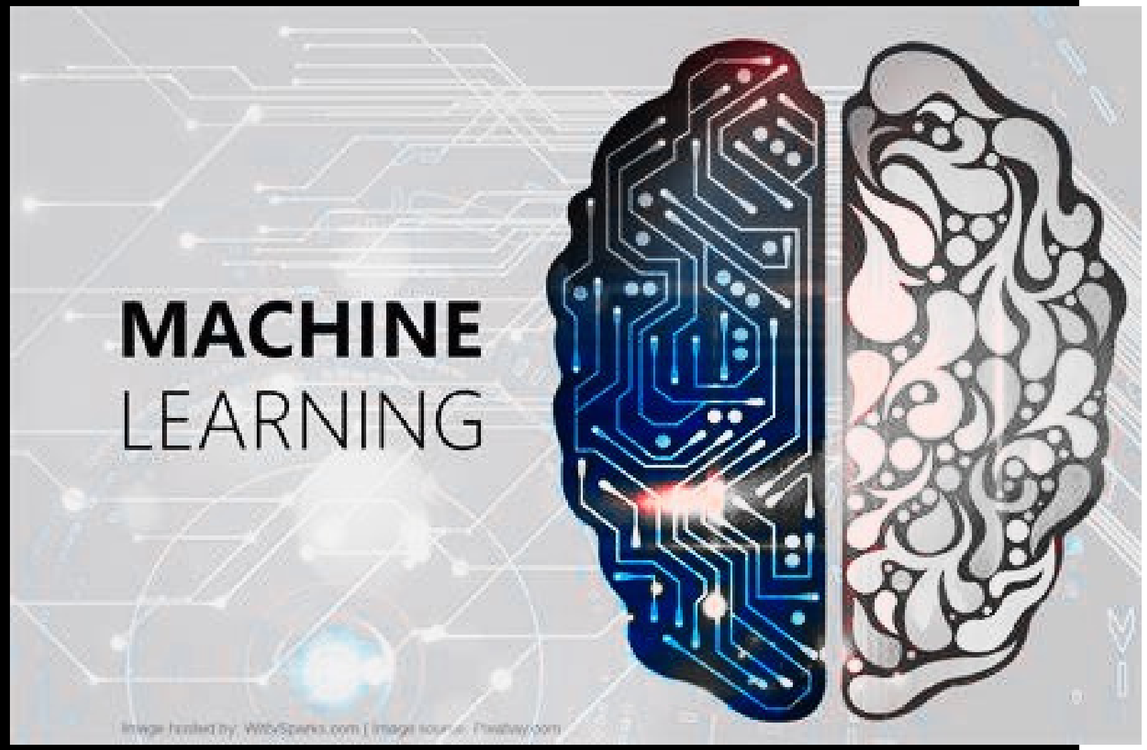
Installation

Fonctionnalités

Application

Conclusion

Introduction



Maching learning est une application de l'intelligence artificielle (IA) qui fournit aux systèmes la capacité d'apprendre et de s'améliorer automatiquement à partir de l'expérience sans être explicitement programmés.
parmi les bibliothèques **ML** les plus populaires on trouve **scikit-learn**

Définition

c'est quoi ?

scikit learn(sklearn) est la bibliothèque la plus puissante pour le machine learning en Python, elle contient de nombreux outils efficaces pour l'apprentissage automatique et la modélisation statistique, notamment la classification, la régression et le clustering.

Installation

prérequis :

Installing Scikit-Learn


```
python3 --version
```

La sortie devrait ressembler à :

```
Python 3.8.2
```

on tape la commande :

```
pip install scikit-learn
```

2eme methode :

```
pip install -U scikit-learn
```

pour verifier l'installation

```
C:\>python3 -m pip show scikit-learn
Name: scikit-learn
Version: 0.24.0
Summary: A set of python modules for machine learning and data mining
Home-page: http://scikit-learn.org
Author: None
Author-email: None
License: new BSD
Location: c:\python38\lib\site-packages
Requires: numpy, scipy, threadpoolctl, joblib
Required-by:
```

- NumPy 1.13.3+
- SciPy 0.19.1+
- Joblib 0.11+
- threadpoolctl 2.0.0+

Fonctionnalités

Scikit-learn propose de nombreuses fonctionnalités pour aider les utilisateurs à résoudre des problèmes de machine learning

sklearn propose des outils pour normaliser, standardiser, encoder, et nettoyer les données pour les préparer à l'utilisation avec des algorithmes de machine learning.

Modèles de machine learning

sklearn propose une variété de modèles de machine learning tels que les régressions linéaires et logistiques, les arbres de décision, les k plus proches voisins, les réseaux de neurones, etc.

Interfaçage avec d'autres librairies

sklearn est compatible avec de nombreuses autres librairies de machine learning et de traitement de données, comme NumPy, pandas, et TensorFlow.

les types de modèles de machine learning disponibles dans sklearn

1- Régressions: y compris la régression linéaire, la régression Ridge, la régression Lasso, la régression polynomiale, etc.

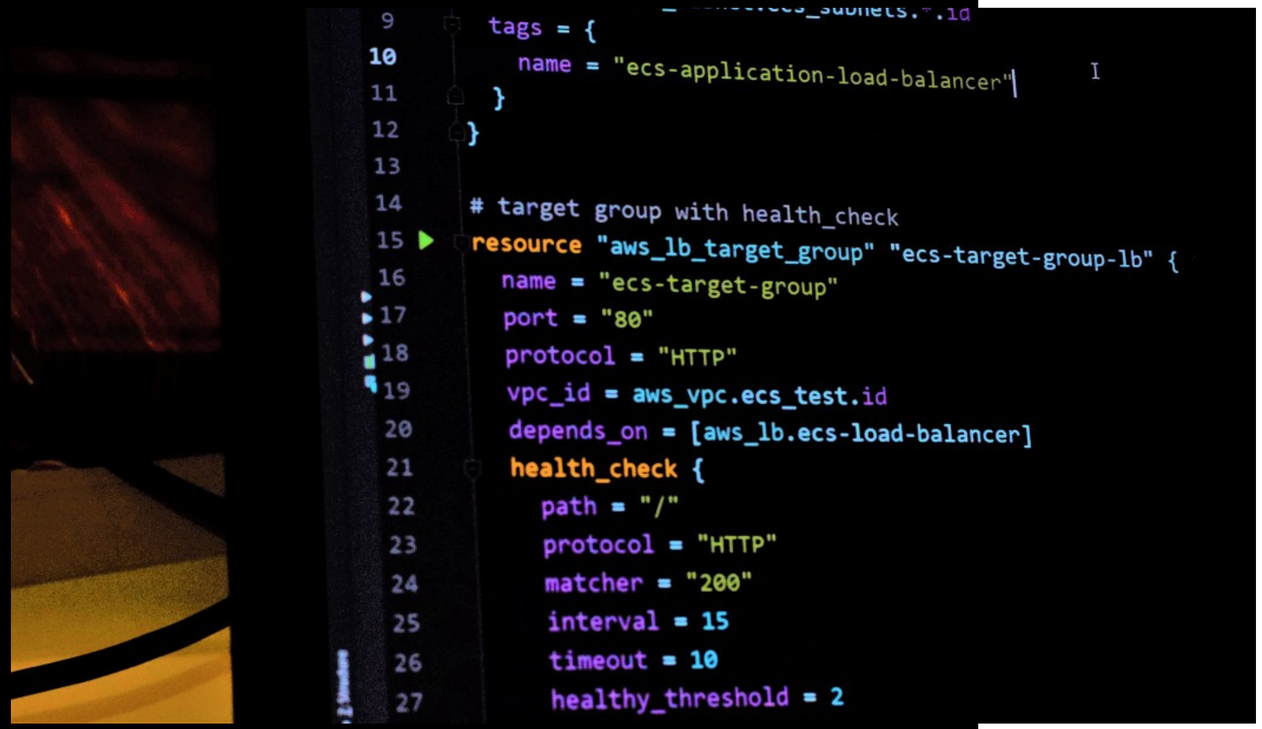
2- **Arbres de décision:** y compris l'arbre de décision, le random forest et l'extra tree.

3- k-moyennes: pour l'agrégation de données en groupes (clustering)

4- **k-plus proches voisins (k-NN)**

5- **Réseaux de neurones :** MLP, RBM

etc.



Application

la problématique

comment prédire les valeurs de la colonne "salary" dans le fichier (SalaireData1.csv) en se basant sur le dataset de fichier (Salary_Data.csv) en utilisant sklearn?

	A	B
1	YearsExperie	Salary
2	1.1	39343.00
3	1.3	46205.00
4	1.5	37731.00
5	2.0	43525.00
6	2.2	39891.00
7	2.9	56642.00
8	3.0	60150.00
9	3.2	54445.00
10	3.2	64445.00
11	3.7	57189.00
12	3.9	63218.00
13	4.0	55794.00

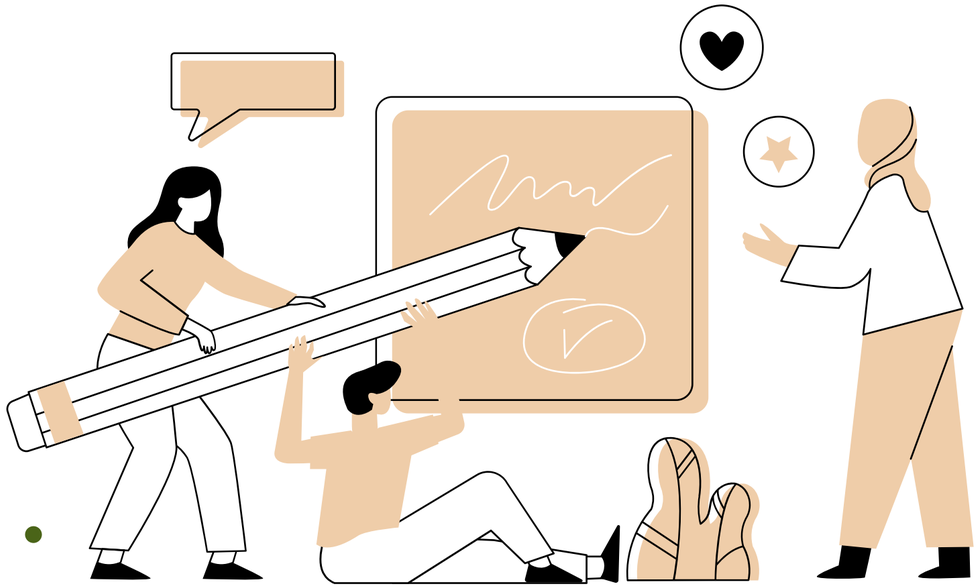
Salairy_Data.csv

A	B
YearsExperie	Salary
3	
5	
7	
9	
11	
13	
15	
17	
19	
21	
23	
25	

SalaireData1.csv

l'objectif :

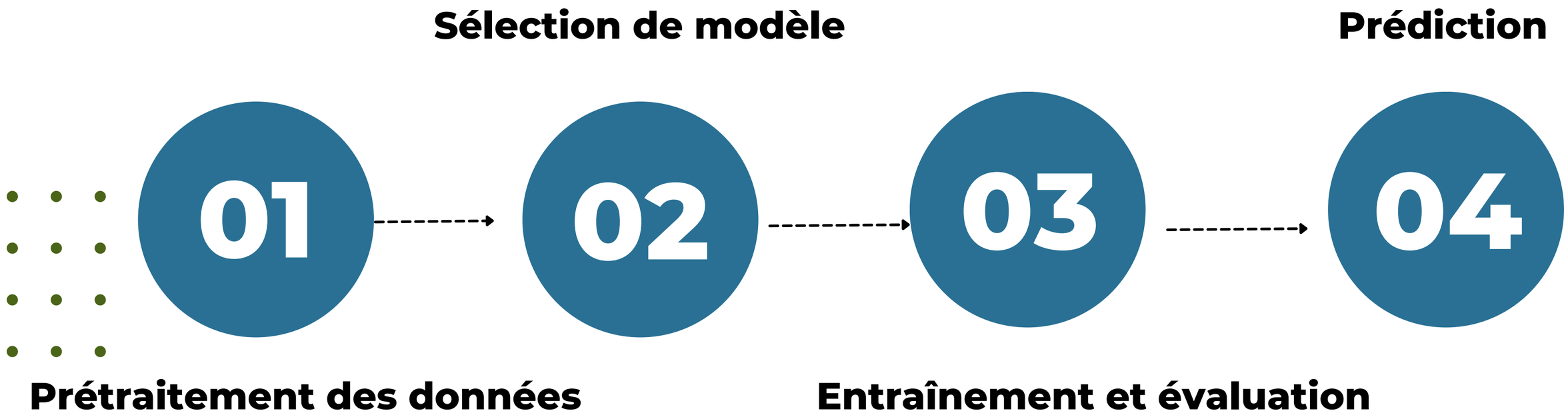
est de construire un bon modèle d'apprentissage automatique capable de prédire les salaires en fonction nombre d'années d'experiences



A	B
YearsExperie	Salary
3	
5	
7	
9	
11	
13	
15	
17	
19	
21	
23	
25	

SalaireData1.csv

LES ETAPES SUIVIES :

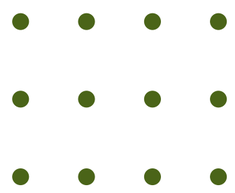


Importer les paquets

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

Importer le dataset

```
data = pd.read_csv("Salary_Data.csv")
```



afficher le dataset (Salary_data.csv)

```
data.head()
```

	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

```
data.info()
```

```
<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: 608.0 bytes
```

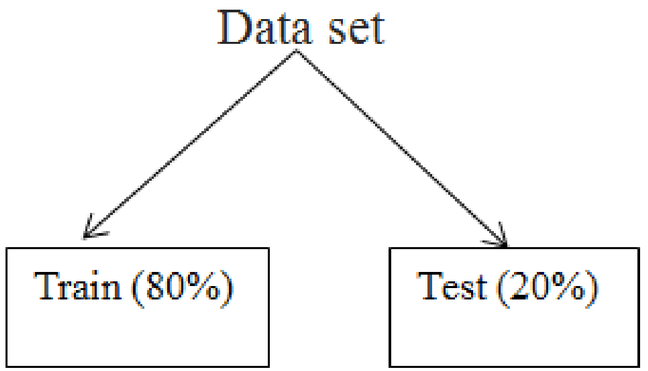
Prétraitement des données

```
x = data.iloc[:,0].values
y = data.iloc[:,1].values
```

x stock toutes les lignes de la première colonne de data (YearsExperience)
y stock toutes les lignes de la deuxième colonne de data (Salary)

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state=3)
```

train_test_split permet de diviser un ensemble de données en deux parties : un ensemble d'entraînement et un ensemble de test. tel que 20% des données seront utilisées pour tester le modèle (x_test,y_test) et les 80% pour l'entraînement (x_train,y_train)



Définir le modèle de régression linéaire

```
ml = LinearRegression()
```

La variable **ml** est un objet de la classe de régression linéaire que on va utiliser pour entraîner un modèle sur les données d'entraînement, puis effectuer des prédictions sur les données de test.

Entraîner le modèle sur les données d'entraînement

```
ml.fit(x_train, y_train)
```

```
LinearRegression()
```

fit() est une méthode de l'objet de régression linéaire ml, qui permet d'entraîner le modèle sur des données d'entraînement. Les arguments passés à la méthode fit sont les données d'entraînement x_train et les étiquettes correspondantes y_train

Faire des prédictions sur les données du test

```
y_pred = ml.predict(x_test)
```

La méthode predict prend en entrée les données pour lesquelles on souhaite effectuer des prédictions et renvoie les prédictions sous forme d'un tableau de valeurs.

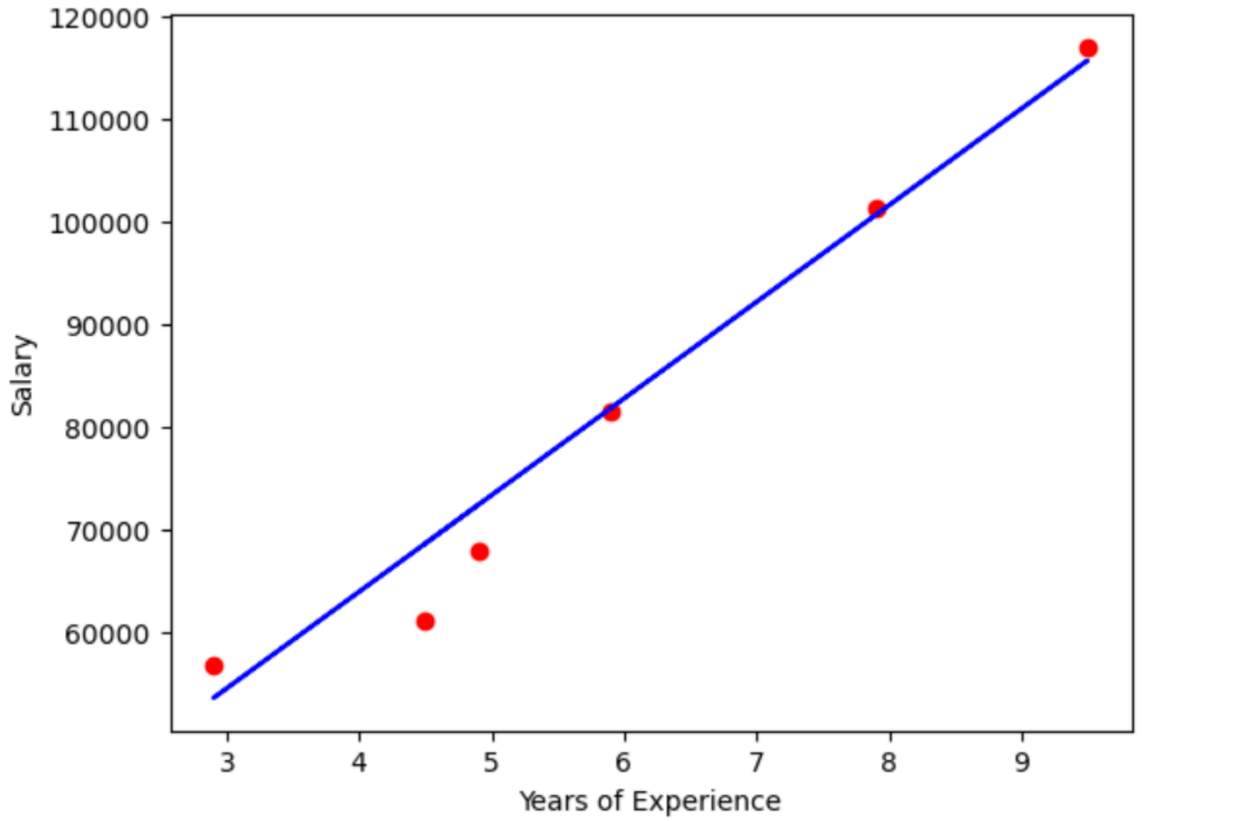
y_pred est un tableau qui contient les prédictions du modèle pour les données de test

Comparer les résultats de la prédiction

- le nuage de points rouge correspond aux données réelles
- la ligne bleue correspond aux valeurs prédites
- on constate que les nuages de points sont proches de la droite blue donc on peut dire que **notre modèle généralise assez bien**

```
plt.scatter(x_test,y_test,color= 'red')
plt.plot(x_test,y_pred,color='blue')
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
```

```
Text(0, 0.5, 'Salary')
```



calculer la performance du modèle avec score()

```
ml.score(x_test,y_test)
```

```
0.9695039421049821
```

score() : renvoie un score de performance qui varie entre 0 et 1, on a trouvé un score proche de 1 signifie que notre modèle est performant donc nous pouvons utiliser ce modèle pour prédire les valeurs manquantes dans la colonne Salary du fichier salaireData1.csv

afficher le dataset (SalaireData1.csv)

```
data2 = pd.read_csv("SalaireData1.csv")
```

```
data2.head()
```

	YearsExperience	Salary
0	3	NaN
1	5	NaN
2	7	NaN
3	9	NaN
4	11	NaN

utiliser le modèle pour prédire les salaires

```
x_predict = data2[["YearsExperience"]].values
```

```
y_predict= ml.predict(x_predict)
```

x_predict contient les années d'expérience pour lesquelles on souhaite effectuer des prédictions de salaires

y_predict contient les prédictions de salaires

résultat

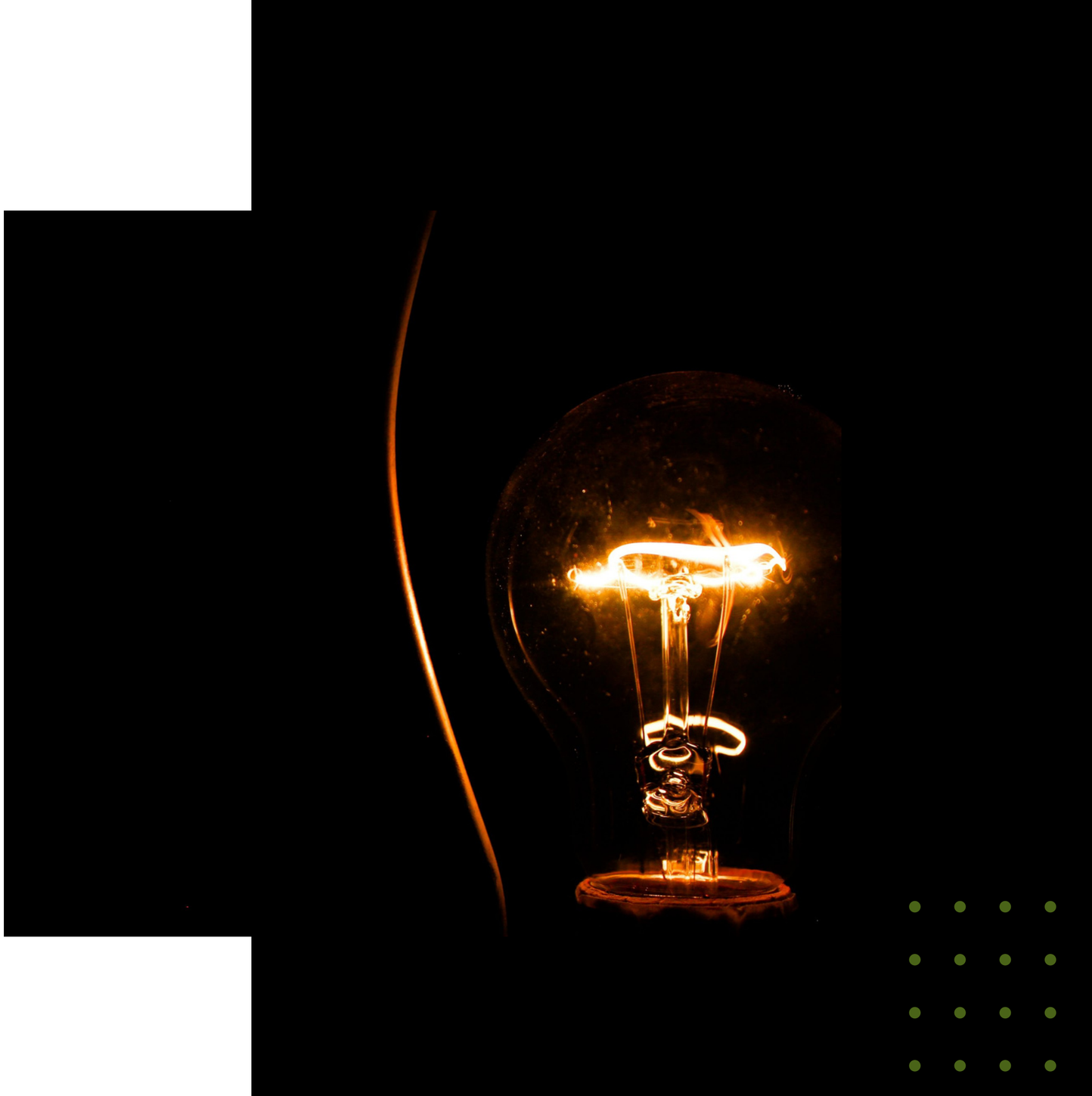
```
y_predict.shape
df=pd.DataFrame(data2[["YearsExperience"]])
df['Salary']=y_predict
```

```
df
```

	YearsExperience	Salary
0	3	54453.467948
1	5	73290.601896
2	7	92127.735843
3	9	110964.869791
4	11	129802.003738
5	13	148639.137685
6	15	167476.271633
7	17	186313.405580
8	19	205150.539527
9	21	223987.673475
10	23	242824.807422
11	25	261661.941370
12	27	280499.075317

Conclusion

scikit-learn rest un outil incontournable pour tous les chercheurs et les développeurs d'IA qui cherchent à résoudre des problèmes d'apprentissage automatique



MERCI

pour votre attention