Régression Linéaire

Introduction:

La régression linéaire simple est une méthode statistique fondamentale utilisée pour analyser la relation entre deux variables quantitatives : une **variable indépendante** et **une variable dépendante**. Son principal objectif est de modéliser et de prédire les valeurs de la variable dépendante en fonction des variations de la variable indépendante. Cette méthode repose sur l'idée que la relation entre les deux variables peut être représentée par une droite dans un plan cartésien, permettant ainsi d'identifier des tendances et d'effectuer des prévisions.

Les quatre notions essentielles pour le développement d'un modèle de régression linéaire sont :

1. **Le Dataset** : Dans l'apprentissage supervisé, le dataset est l'ensemble de données sur lequel le modèle est entraîné. Il est divisé en deux parties : les caractéristiques (features), qui sont les variables indépendantes, et la cible (target), qui est la variable dépendante que l'on souhaite prédire.
2. **Le Modèle** : À partir de ce jeu de données, on crée un modèle, représenté par une fonction mathématique de la forme $y=f(x)=ax+b$. Les coefficients de cette fonction (comme aaa et bbb) sont les paramètres du modèle que l'on ajuste pour obtenir la meilleure approximation possible.
3. **La Fonction Coût** : Après avoir testé notre modèle sur le dataset, celui-ci génère des erreurs, qui représentent la différence entre les valeurs prédites et les valeurs réelles. La fonction coût mesure ces erreurs et sert à quantifier la performance du modèle.
4. **L'Algorithme d'Apprentissage** : L'objectif de l'entraînement du modèle est de trouver les paramètres qui minimisent les erreurs (la fonction coût). Pour cela, on utilise des algorithmes d'apprentissage tels que la descente de gradient, qui ajustent les paramètres pour améliorer les prédictions du modèle.

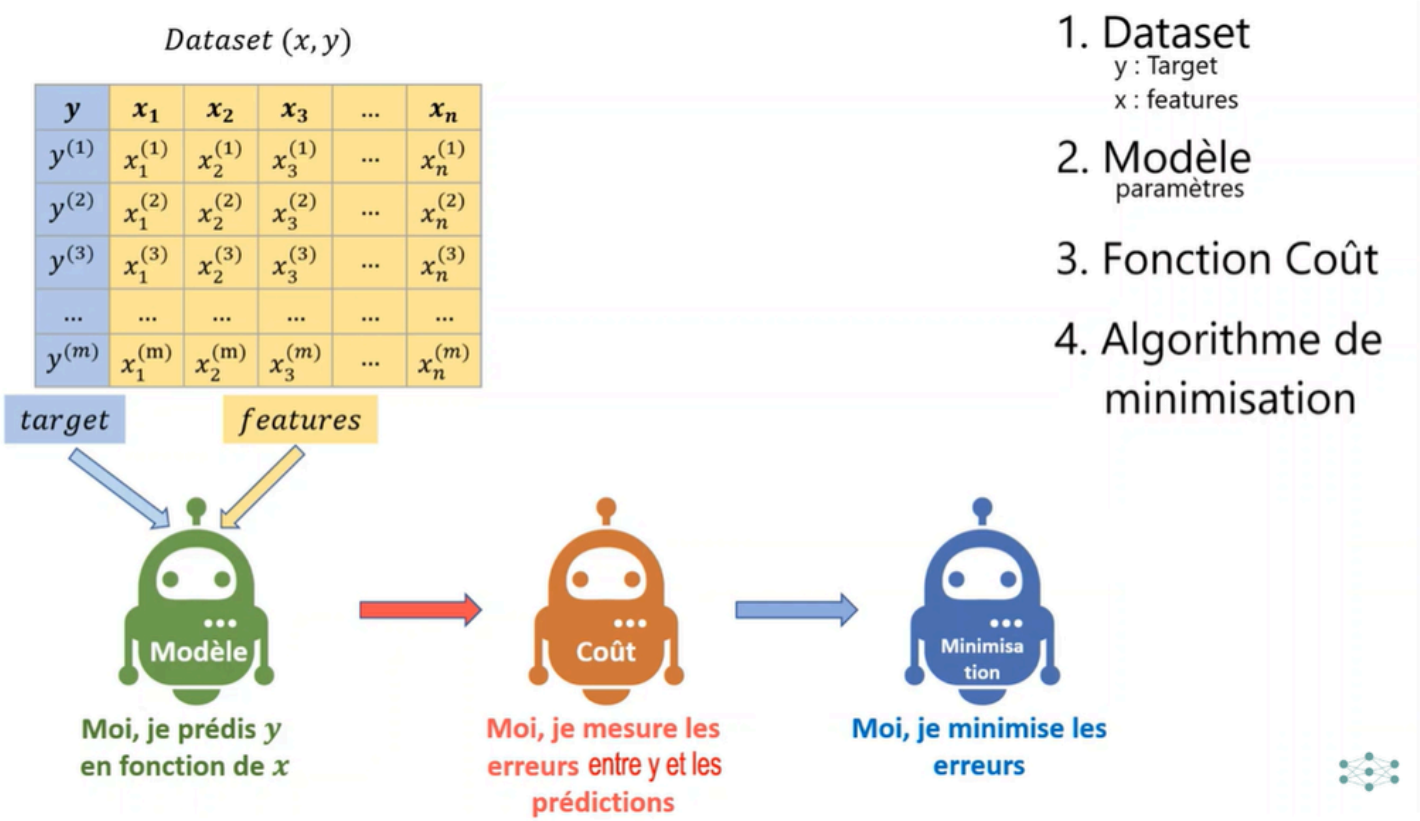


Figure1:Illustration des étapes de développement d'un modèle de machine learning(**image de Machine Learnia**)

Les étapes pour développer une Régression linéaire:

Etape 1 **: Importer les librairies**

Tout d'abord, nous allons commencer par l'importation des bibliothèques et des fonctions suivantes :

- **NumPy** : pour manipuler le dataset comme une matrice.
- **Matplotlib** : pour visualiser nos données.
- **make_regression** :pour générer un nuage de points pour notre dataset.
- **SGDRegressor** :l'algorithme de régression utilisant la descente de gradient stochastique . Il inclut la fonction coût, le calcul du gradient, et l'algorithme de minimisation.

```
import  numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_regression
from sklearn.linear_model import SGDRegressor
```

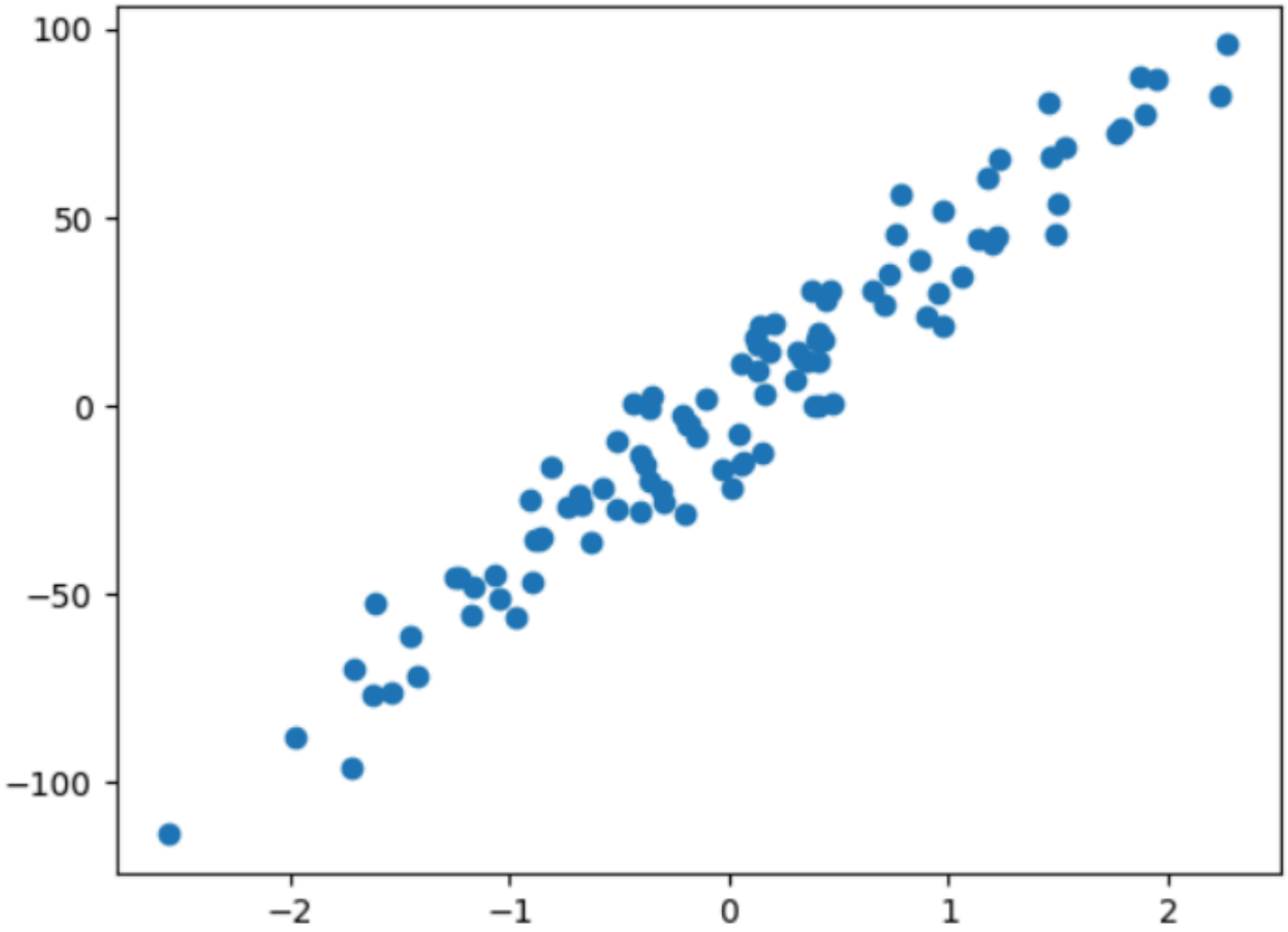
Etape 2 **: Créer Un Dataset**

Un tableau de données (x, y) aléatoires sera généré à l'aide de la fonction make_regression, qui prend comme arguments le nombre d'échantillons, le nombre de variables et le bruit.

Pour **la visualisation**, la fonction **plt.scatter(x, y)** sera utilisée.

```
np.random.seed(0)
x, y = make_regression(n_samples=100, n_features=1, noise=10)
plt.scatter(x, y)
```

Le résultat obtenu est le suivant :



Etape 3: **Développement et entraînement du modèle**

Pour créer un modèle, une variable nommée model est définie en utilisant le générateur SGDRegressor. Cela se fait en spécifiant le nombre d'itérations ainsi que le taux d'apprentissage (learning rate).

La fonction **fit** est utilisée pour entraîner notre modèle.

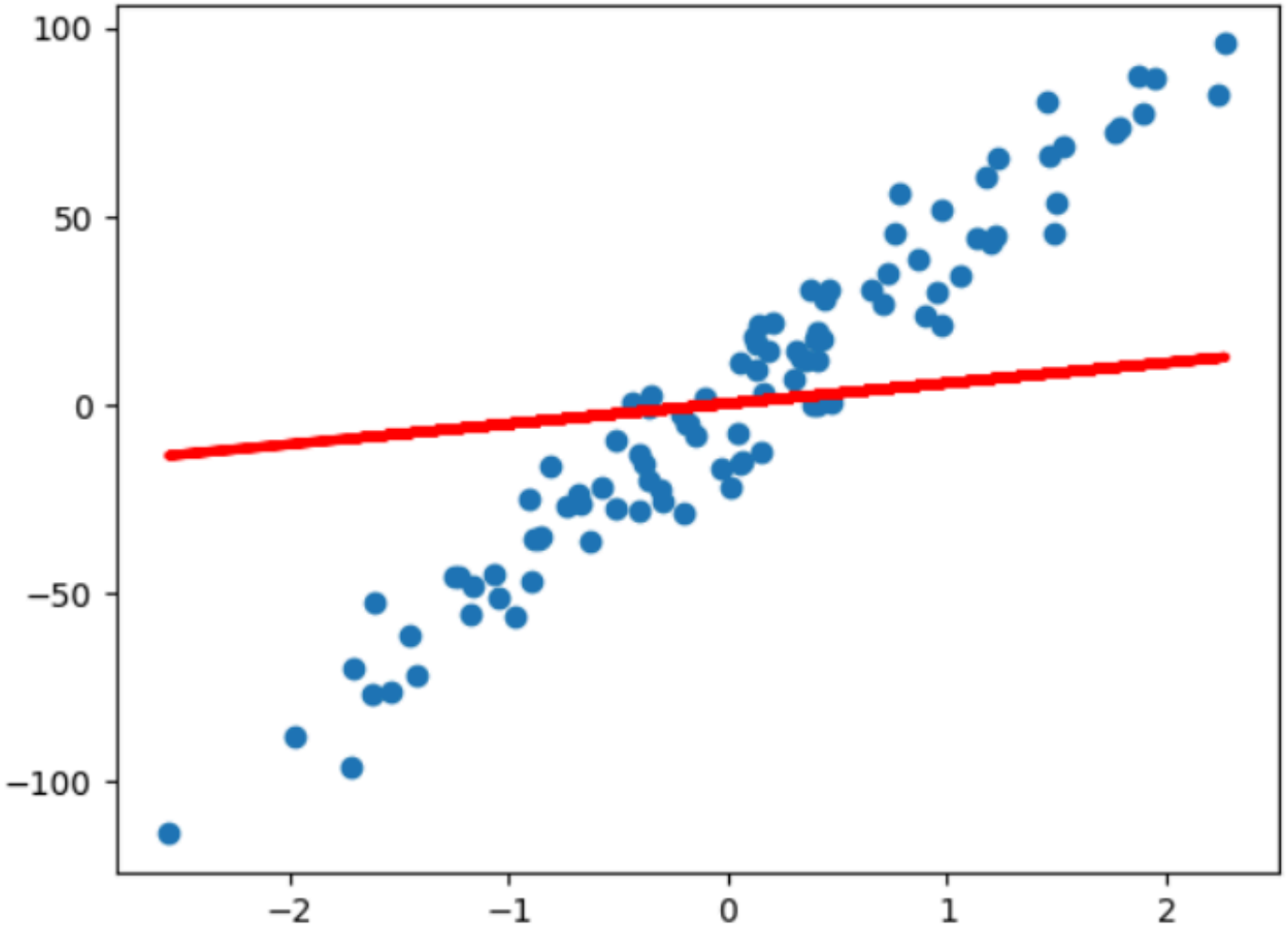
```
model=SGDRegressor(max_iter=100,eta0=0.0001)
model.fit(x,y)
```

Avec la fonction R^2 qui calcule le **coefficient de détermination** entre le modèle et les valeurs y de notre dataset, nous pouvons observer la précision de notre modèle.

Nous pouvons également utiliser notre modèle pour effectuer de nouvelles prédictions avec la fonction predict et visualiser les résultats à l'aide de la fonction plt.plot

```
print('Coeff R2 =', model.score(x, y))
plt.scatter(x, y)
plt.plot(x, model.predict(x), c='red', lw = 3)
```

```
Coeff R2 = 0.22318690191350676
```



Nous avons obtenu de mauvais résultats parce que le modèle n'a pas été entraîné suffisamment longtemps et que le taux d'apprentissage était trop faible. Ce n'est pas un problème, il est possible de le réentraîner avec de meilleurs hyperparamètres

En Machine Learning, les valeurs qui donnent généralement de bons résultats pour la majorité des entraînements sont les suivantes :

- Nombre d'itérations : 1000
- Taux d'apprentissage : 0.001

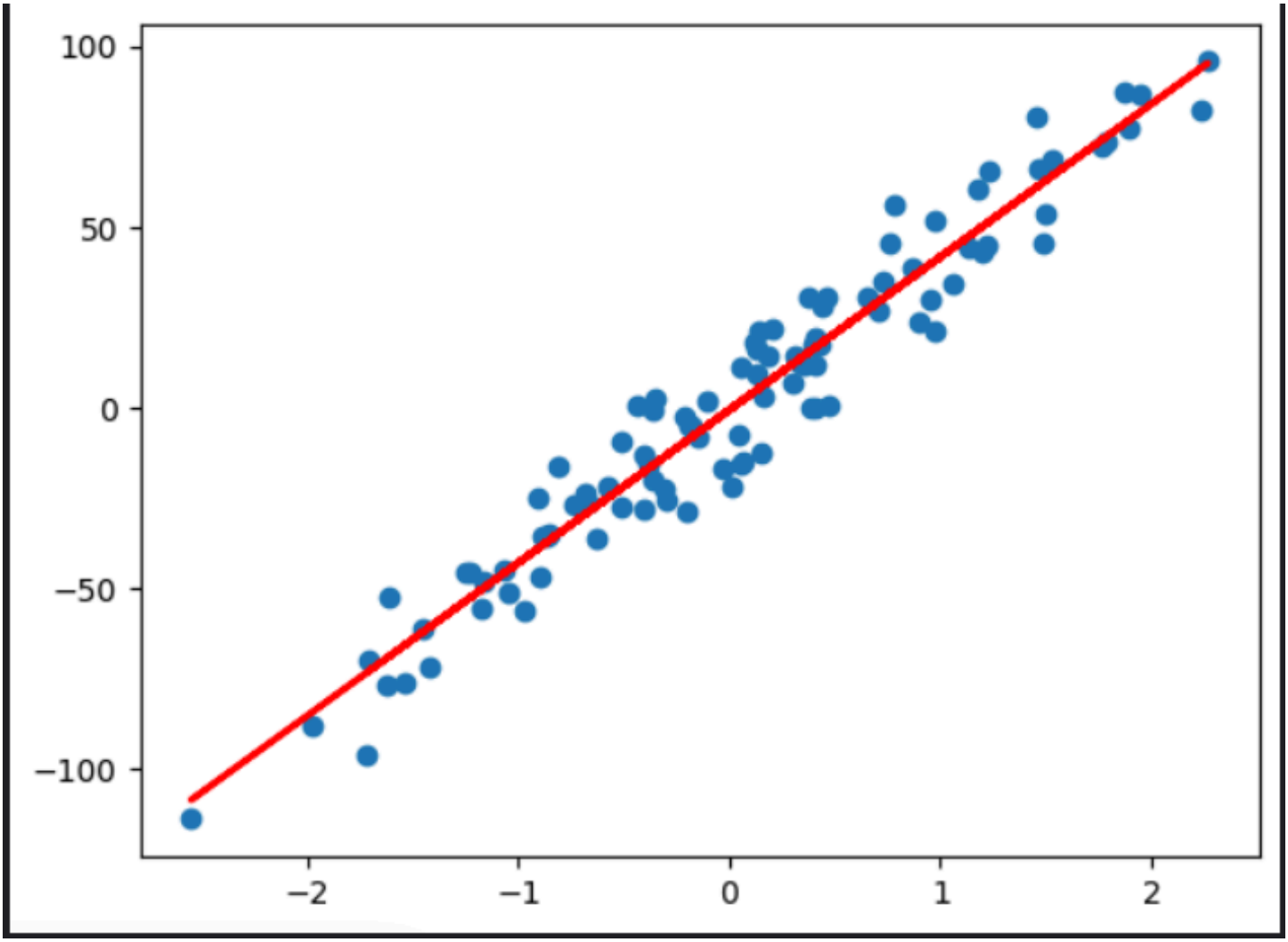
```
model= SGDRegressor(max_iter=1000,eta0=0.0001,learning_rate='constant')
model.fit(x,y)
```

```
print('Coeff R2 =', model.score(x, y))

plt.scatter(x, y)
plt.plot(x, model.predict(x), color='red', linewidth=2)
```

```
Coeff R2 = 0.9416838606962155
```

Notre modèle fonctionne très bien avec un coefficient de détermination R^2 de 94 %.



Résumé : pour développer un modèle en machine learning :

1/ **Sélectionner un modèle** : model = LinearRegression(). Cela signifie choisir un algorithme adapté à la nature des données et au problème à résoudre, ici la régression linéaire.

2/ **Entraîner le modèle sur les données X, Y** : model.fit(X, Y). Cela consiste à ajuster le modèle aux données d'entraînement pour qu'il puisse apprendre les relations entre les variables indépendantes (X) et la variable dépendante (Y).

3/ **Évaluer le modèle** : model.score(X, Y). Cette étape permet de mesurer la performance du modèle en calculant son score de précision sur les données d'entraînement.

4/ **Utiliser le modèle** : model.predict(X). Une fois le modèle entraîné, on l'utilise pour prédire les valeurs de Y à partir de nouvelles données X.