```
!pip install numdifftools
```

```
Requirement already satisfied: numdifftools in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scipy>=0.8 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: statsmodels>=0.6 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: algopy>=0.4 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: numpy>=1.9 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: patsy>=0.4.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from pa
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from numpy.linalg import norm
from numdifftools import Gradient
import copy
from sklearn.datasets import make_regression
```

Regression

```
class Regression:
  def __init__(self,X):
    self.w = np.random.randn(X.shape[1]+1, 1)

  def model(self,X, w):
    if X.shape[1] !=w.shape[0]:
      x = np.hstack((X, np.ones((X.shape[0],1))))
    else:
      x = X

    return x.dot(w)

  def lost_function(self,X, y, w):
    m = len(y)
    if X.shape[1] !=w.shape[0]:
      x = np.hstack((X, np.ones((X.shape[0],1))))
    else:
      x = X
    return 1/(2*m) * np.sum((self.model(x, w) - y)**2)

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

  def gradient_descent(self,X, y, w, alpha=0.001,e = 0.001,n_iter = 10000):

    cost_history = [] # création d'un tableau de stockage pour enregistrer l'évolution du
    i = 0
    while norm(self.grad(X,y,w).T) >e :
        w = w - alpha * self.grad(X, y, w)
        # mise a jour du parametre w (formule du gradient descent)
        cost_history.append(self.lost_function(X, y, w)) # on enregistre la valeur du Cout
        i+=1
        if i>n_iter:
          break
    return w, cost_history

  def fit(self,X,y,alpha = 0.001,e = 0.001,n_iter = 100000):
    x = np.hstack((X, np.ones((X.shape[0],1))))
    self.w,self.cost_history = self.gradient_descent(x,y,self.w,alpha,e,n_iter)

  def lost_courbe(self):
    plt.plot(range(len(self.cost_history)), self.cost_history)

  def coef_determination(self,X,y):
    u = ((y - self.model(X,self.w))**2).sum()
    v = ((y - y.mean())**2).sum()
    return 1 - u/v
```

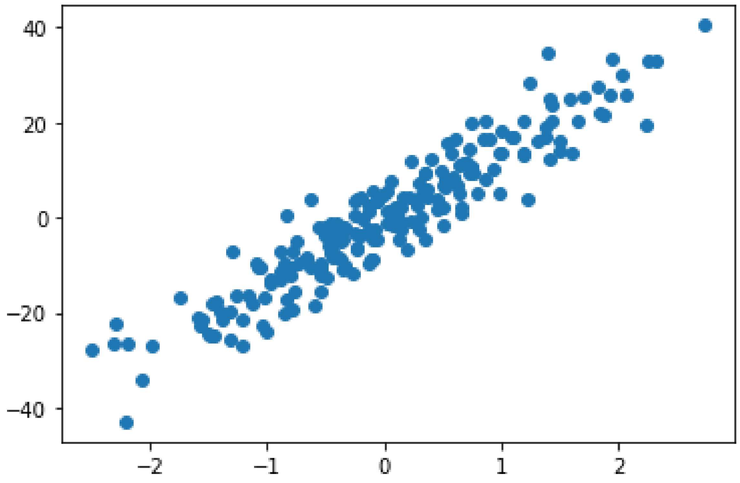
Dataset to test the algorithme

```
x,y = make_regression(n_samples=200,n_features=1,noise=5)
from sklearn.model_selection import train_test_split
X_train, y_train = x[:150], y[:150]
X_test, y_test = x[150:], y[150:]
print(X_train.shape)
print(X_test.shape)
```

```
(150, 1)
(50, 1)
```

```
plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x7f2d3c6e6950>
```



```
print(X_train.shape)
y_train = y_train.reshape((150,1))
y_test = y_test.reshape((50,1))
print(y_train.shape)
```

```
(150, 1)
(150, 1)
```

```
R = Regression(X_train)
```

```
R.w
```

```
array([[0.52003023],
       [0.33285384]])
```

```
#Erreur initail
R.lost_function(X_train,y_train,R.w)
```

```
105.6957445171955
```

```
R.fit(X_train,y_train)
```

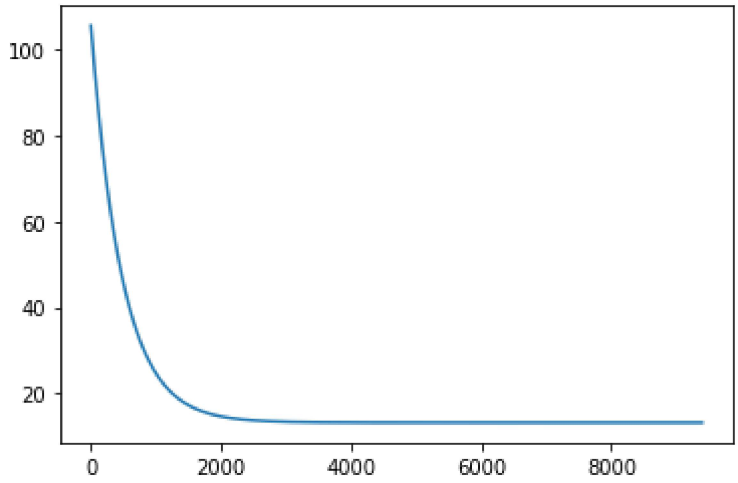
```
R.lost_function(X_train,y_train,R.w)
```

```
13.19936832629869
```

```
R.w
```

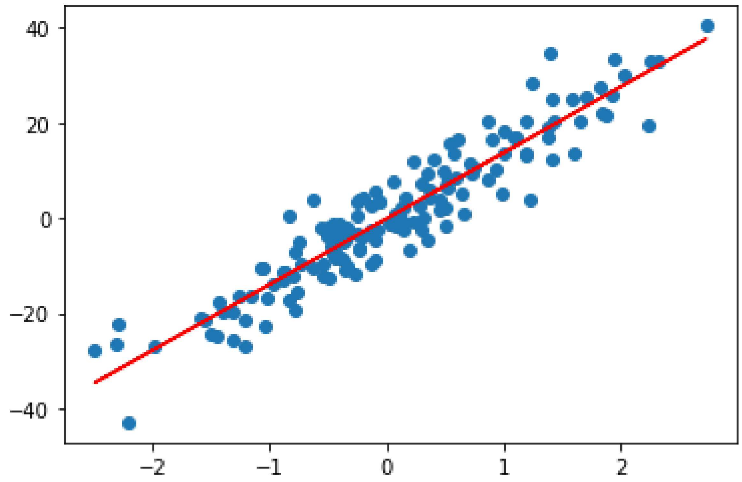
```
array([[13.81204828],
       [-0.114275  ]])
```

```
R.lost_courbe()
```



```
plt.scatter(X_train,y_train)
plt.plot(X_train,R.model(X_train,R.w),c = 'r')
```

```
[<matplotlib.lines.Line2D at 0x7f2d3c11aed0>]
```

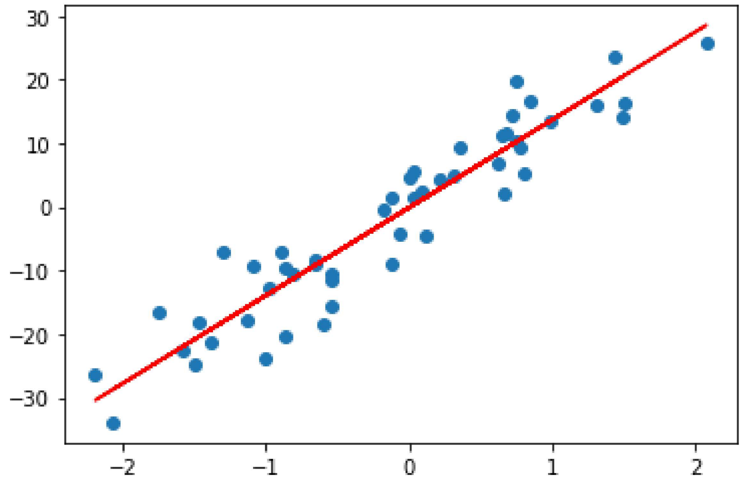


```
R.coef_determination(X_train,y_train)
```

```
0.8831910855472059
```

```
plt.scatter(X_test,y_test)
plt.plot(X_test,R.model(X_test,R.w),c = 'r')
X_test.shape
```

```
(50, 1)
```



```
R.lost_function(X_test,y_test,R.w)
```

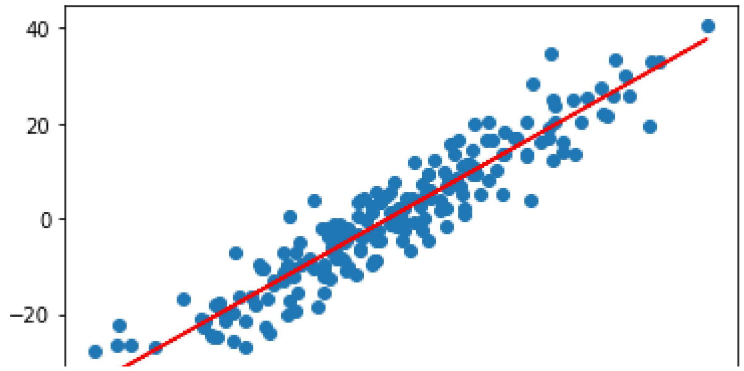
```
11.898624185183492
```

```
R.coef_determination(X_test,y_test)
```

```
0.8858715121110685
```

```
plt.scatter(x,y)
plt.plot(x,R.model(x,R.w),c = 'r')
```

```
[<matplotlib.lines.Line2D at 0x7f2d3c030ad0>]
```



Exercice 1

```
from google.colab import drive
drive.mount("/content/drive/")
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive
```

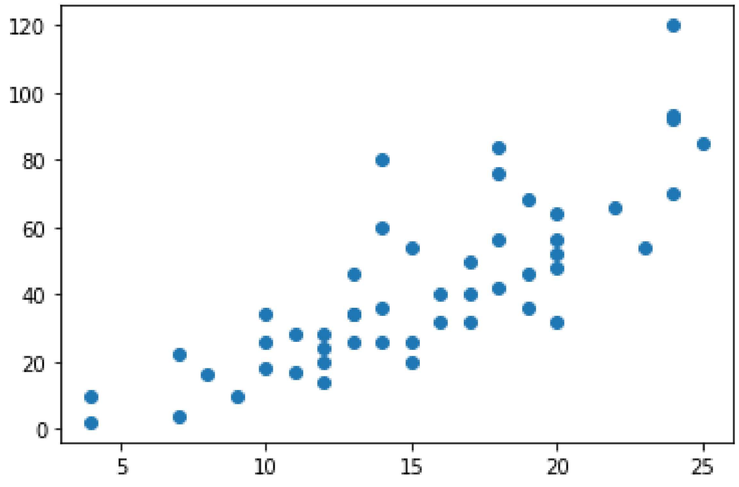
```
data_cars = pd.read_csv('/content/drive/My Drive/TP_machine_learning/cars.csv')
```

```
data_cars.head()
```

	Unnamed: 0	speed	dist
0	1	4	2
1	2	4	10
2	3	7	4
3	4	7	22
4	5	8	16

```
plt.scatter(data_cars['speed'],data_cars['dist'])
```

```
<matplotlib.collections.PathCollection at 0x7f2d3c14ad10>
```



```
X = np.array(data_cars['speed'])
y = np.array(data_cars['dist'])
```

```
print(X.shape)
print(y.shape)
```

```
(50,)
(50,)
```

```
X = X.reshape((X.shape[0],1))
y = y.reshape((y.shape[0],1))
X.shape
```

```
(50, 1)
```

```
R1 = Regression(X)
```

```
R1.w
```

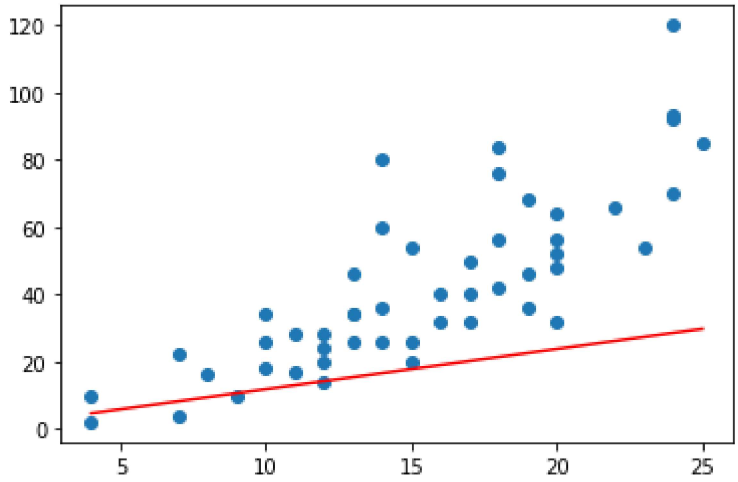
```
array([[ 1.19573531],
       [-0.18256351]])
```

```
R1.lost_function(X,y,R1.w)
```

```
522.3774213198348
```

```
plt.scatter(X,y)
plt.plot(X,R1.model(X,R1.w),c = 'r')
```

```
[<matplotlib.lines.Line2D at 0x7f2d3bfc5290>]
```



```
R1.fit(X,y)
```

```
R1.w
```

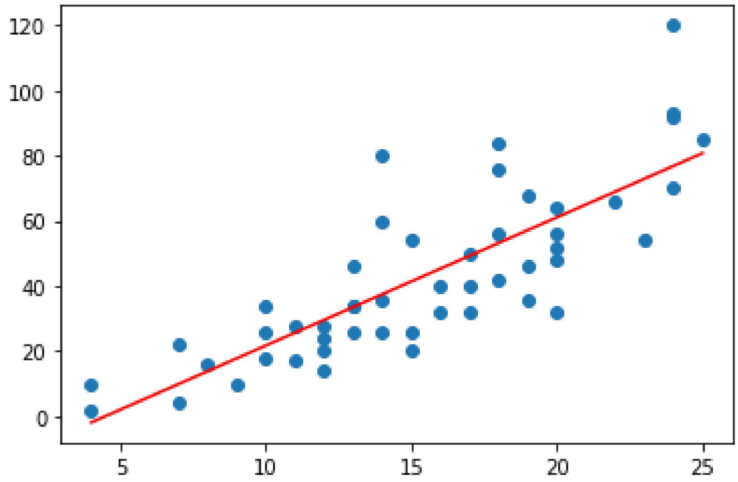
```
array([[  3.9318456 ],
           [-17.56942403]])
```

```
R1.lost_function(X,y,R1.w)
```

```
    113.53521535409273
```

```
plt.scatter(X,y)
plt.plot(X,R1.model(X,R1.w),c = 'r')
```

```
    [<matplotlib.lines.Line2D at 0x7f2d3bcf6410>]
```



```
R1.coef_determination(X,y)
```

```
    0.6510793658741216
```

Exercice 3

```
data_pops = pd.read_excel('/content/drive/My Drive/TP_machine_learning/pop.xlsx')
```

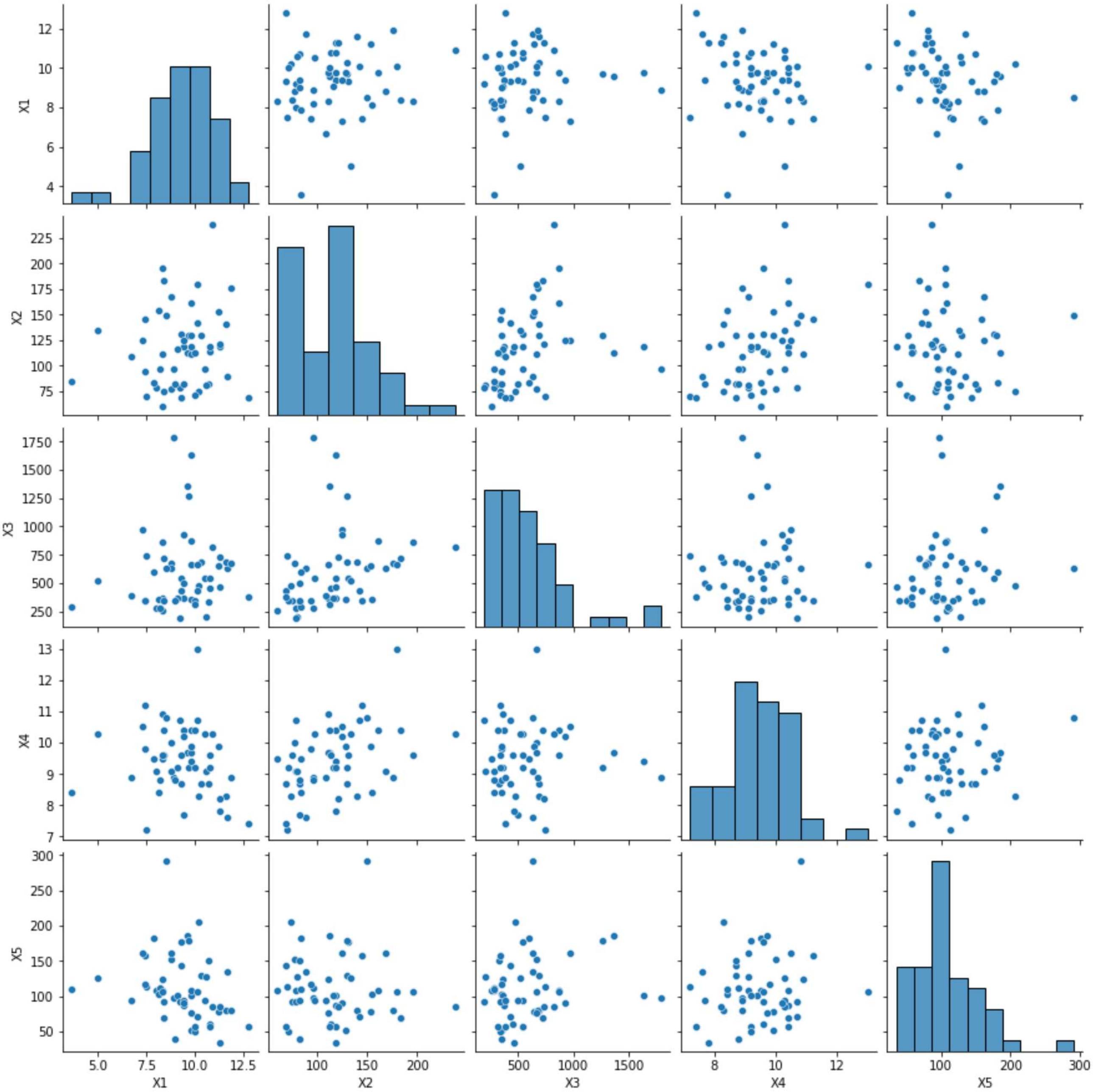
```
data_pops.head()
```

	X1	X2	X3	X4	X5
0	8.0	78	284	9.1	109
1	9.3	68	433	8.7	144
2	7.5	70	739	7.2	113
3	8.9	96	1792	8.9	97
4	10.2	74	477	8.3	206

```
import seaborn as sbs
```

```
sbs.pairplot(data_pops, size=2.5);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:2076: UserWarning: The `si
  warnings.warn(msg, UserWarning)
```



```
X = np.array(data_pops[['X1','X2','X3','X4']])
y = np.array(data_pops['X5'])
```

```
print(X.shape)
print(y.shape)
```

```
(53, 4)
(53,)
```

```
y = y.reshape((53,1))
y.shape
```

```
(53, 1)
```

```
R3 = Regression(X)
```

```
R3.w
```

```
array([[0.79979181],
       [0.31788753],
       [0.13969132],
       [0.63706308],
       [0.00935675]])
```

```
R3.lost_function(X,y,R3.w)
```

```
2265.638482805931
```

```
R3.fit(X,y,alpha=0.001,e=0.1,n_iter= 3000)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:19: RuntimeWarning: over
/usr/local/lib/python3.7/dist-packages/numpy/core/fromnumeric.py:87: RuntimeWarning:
  return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:30: RuntimeWarning: inva
```

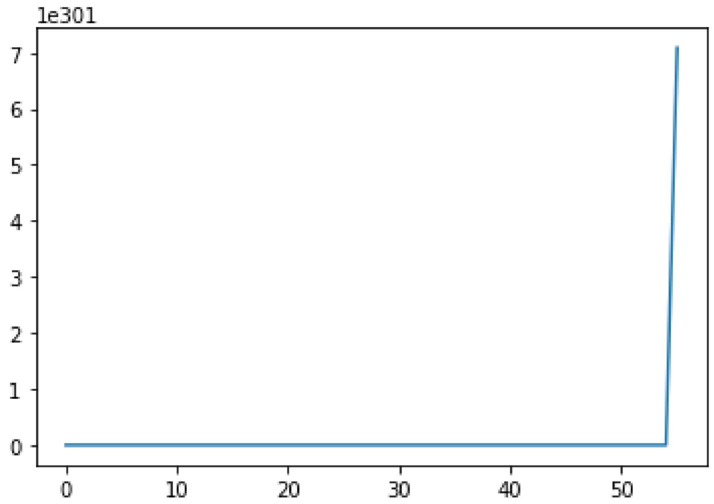
```
R3.w
```

```
array([[nan],
       [nan],
       [nan],
       [nan],
       [inf]])
```

```
R3.lost_function(X,y,R3.w)
```

```
nan
```

```
R3.lost_courbe()
```



✓ 0 s terminée à 16:32