```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

#Função do cáculo da sigmóide
def sigmoid(x):
    return 1/(1+np.exp(-x))


DataSet = pd.read_csv('Data.csv')


DataSet.head()
```

	Input1	Input2	Input3	Output1	Output2
0	0.93	0.23	0.73	0.41	0.42
1	0.49	0.85	0.50	0.41	0.81
2	0.86	0.04	0.68	0.35	0.22
3	0.71	0.29	0.30	0.24	0.67
4	0.96	0.78	0.82	0.56	0.89

```
DataSet.columns
```

```
Index(['Input1', 'Input2', 'Input3', 'Output1', 'Output2'], dtype='object')
```

```
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(DataSet.drop(['Output1', 'Output2'],
                                                    DataSet[['Output1', 'Output2']], test
                                                     random_state=101)


#Tamanho do DataSet de Treinamento
n_records, n_features = X_train.shape

#Arquitetura da MPL
N_input = 3
N_hidden = 4
N_output = 2
learnrate = 0.5
```

```
#Pesos da Camada Oculta (Inicialização Aleatória)
weights_input_hidden = np.random.normal(0, scale=0.1, size=(N_input, N_hidden))
print('Pesos da Camada Oculta:')
print(weights_input_hidden)

#Pesos da Camada de Saída (Inicialização Aleatória)
weights_hidden_output = np.random.normal(0, scale=0.1, size=(N_hidden, N_output))
print('Pesos da Camada de Saída:')
print(weights_hidden_output)
```

```
Pesos da Camada Oculta:
[[-0.1535729   0.03669718  0.01458018  0.020465  ]
 [-0.024315   -0.01097997 -0.11258423  0.08495635]
 [ 0.09787241  0.02826119 -0.02539654  0.04007731]]
Pesos da Camada de Saída:
[[ 0.08681665  0.25369894]
 [-0.04201222 -0.1100921 ]
 [-0.00992037 -0.04105967]
 [-0.00309508 -0.09717834]]
```

+ Código + Texto

```
epochs = 5000
last_loss=None
EvolucaoError=[]
IndiceError=[]

for e in range(epochs):
    delta_w_i_h = np.zeros(weights_input_hidden.shape)
    delta_w_h_o = np.zeros(weights_hidden_output.shape)
    for xi, yi in zip(X_train.values, y_train.values):

# Forward Pass
        #Camada oculta
        #Calcule a combinação linear de entradas e pesos sinápticos
        hidden_layer_input = np.dot(xi, weights_input_hidden)
        #Aplicado a função de ativação
        hidden_layer_output = sigmoid(hidden_layer_input)

        #Camada de Saída
        #Calcule a combinação linear de entradas e pesos sinápticos
        output_layer_in = np.dot(hidden_layer_output, weights_hidden_output)

        #Aplicado a função de ativação
        output = sigmoid(output_layer_in)
        #print('As saídas da rede são',output)
#-----------------------------------------

# Backward Pass
        ## TODO: Cálculo do Erro
        error = yi - output
```

```
    # TODO: Calcule o termo de erro de saída (Gradiente da Camada de Saída)
    output_error_term = error * output * (1 - output)

    # TODO: Calcule a contribuição da camada oculta para o erro
    hidden_error = np.dot(weights_hidden_output,output_error_term)

    # TODO: Calcule o termo de erro da camada oculta (Gradiente da Camada Oculta)
    hidden_error_term = hidden_error * hidden_layer_output * (1 - hidden_layer_outpu

    # TODO: Calcule a variação do peso da camada de saída
    delta_w_h_o += output_error_term*hidden_layer_output[:, None]

    # TODO: Calcule a variação do peso da camada oculta
    delta_w_i_h += hidden_error_term * xi[:, None]

#Atualização dos pesos na época em questão
weights_input_hidden += learnrate * delta_w_i_h / n_records
weights_hidden_output += learnrate * delta_w_h_o / n_records


# Imprimir o erro quadrático médio no conjunto de treinamento

if  e % (epochs / 20) == 0:
    hidden_output = sigmoid(np.dot(xi, weights_input_hidden))
    out = sigmoid(np.dot(hidden_output,
                         weights_hidden_output))
    loss = np.mean((out - yi) ** 2)

    if last_loss and last_loss < loss:
        print("Erro quadrático no treinamento: ", loss, " Atenção: O erro está aumen
    else:
        print("Erro quadrático no treinamento: ", loss)
    last_loss = loss

    EvolucaoError.append(loss)
    IndiceError.append(e)

 Erro quadrático no treinamento:  0.20410022942903783
 Erro quadrático no treinamento:  0.2798016577130815  Atenção: O erro está aumentan
 Erro quadrático no treinamento:  0.20813843515173683
 Erro quadrático no treinamento:  0.09290389487029696
 Erro quadrático no treinamento:  0.053345222185684285
 Erro quadrático no treinamento:  0.03906006277907365
 Erro quadrático no treinamento:  0.032642046854614706
 Erro quadrático no treinamento:  0.029179284339553243
 Erro quadrático no treinamento:  0.027031385921146733
 Erro quadrático no treinamento:  0.025550219677239926
 Erro quadrático no treinamento:  0.02444839596261962
 Erro quadrático no treinamento:  0.023586090388449846
```

```
Erro quadrático no treinamento:  0.022888583918094334
Erro quadrático no treinamento:  0.02231177131085136
Erro quadrático no treinamento:  0.021826894951755917
Erro quadrático no treinamento:  0.021413490100043574
Erro quadrático no treinamento:  0.02105604652417263
Erro quadrático no treinamento:  0.020742411863392513
Erro quadrático no treinamento:  0.0204629995093066
Erro quadrático no treinamento:  0.02021033864915133
```

```
plt.plot(IndiceError, EvolucaoError, 'r') # 'r' is the color red
plt.xlabel('')
plt.ylabel('Erro Quadrático')
plt.title('Evolução do Erro no treinamento da MPL')
plt.show()
```


```
# Calcule a precisão dos dados de teste
n_records, n_features = X_test.shape
MSE_Output1=0
MSE_Output2=0

for xi, yi in zip(X_test.values, y_test.values):

# Forward Pass
        #Camada oculta
        #Calcule a combinação linear de entradas e pesos sinápticos
        hidden_layer_input = np.dot(xi, weights_input_hidden)
        #Aplicado a função de ativação
        hidden_layer_output = sigmoid(hidden_layer_input)

        #Camada de Saída
        #Calcule a combinação linear de entradas e pesos sinápticos
        output_layer_in = np.dot(hidden_layer_output, weights_hidden_output)
```

```
        #Aplicado a função de ativação
        output = sigmoid(output_layer_in)

#-----------------------------------------

#Cálculo do Erro
        ## TODO: Cálculo do Erro
        error = yi - output
        MSE_Output1 += (yi[0] - output[0])**2
        MSE_Output2 += (yi[1] - output[1])**2

#Erro Quadrático Médio
MSE_Output1/=n_records
MSE_Output2/=n_records

print('Erro Quadrático Médio da Saída Output1 é: ',MSE_Output1)
print('Erro Quadrático Médio da Saída Output2 é: ',MSE_Output2)
```

```
Erro Quadrático Médio da Saída Output1 é:  0.015555117159104506
Erro Quadrático Médio da Saída Output2 é:  0.004128980575364591
```