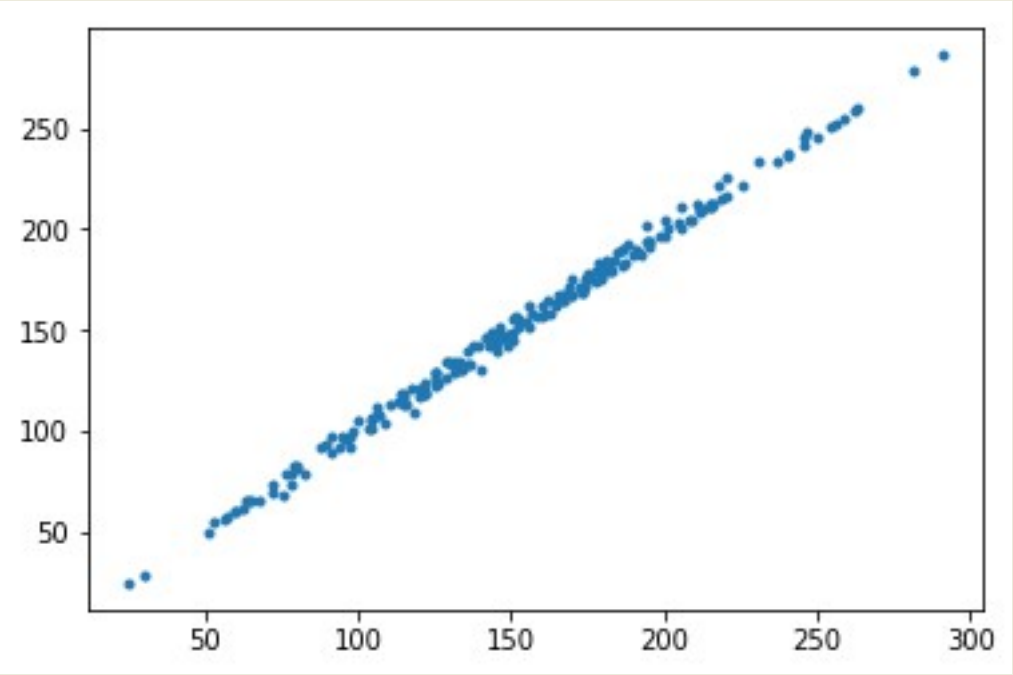
ML_CLASIFICACION_RN_02	Red Neuronal para sumar tres números	ML

En esta práctica se construye una red neuronal que aprende a sumar tres números.

Se crea de forma aleatoria un conjunto de datos ('dataset') formado por la matriz 'X' que contiene los 'inputs' (X1, X2, X3) de la capa de entrada y la matriz 'y' que contiene los 'outputs' (y=X1+X2+X3).

Se utiliza la librería 'sklearn' para entrenar y testar la red neuronal. Se divide 'X' para ejecutar el entrenamiento (X_train,y_train) y la validación o test de la red neuronal (X_test,y_test).

Como resultado se obtiene una red neuronal capaz de sumar tres números.



SOLUCIÓN

Importar las librerías necesarias para realizar la práctica.

```
# Librerías
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPRegressor
from sklearn import model_selection
```

Generar aleatoriamente los inputs (x) y calcular el output (y)

```
# Generar 3 secuencias de números enteros aleatorios X1, X2 y X3 (inputs) y
los sumamos (y, output)
ndatos=1000
X1=np.round(np.random.uniform(size=ndatos)*100)
X2=np.round(np.random.uniform(size=ndatos)*100)
X3=np.round(np.random.uniform(size=ndatos)*100)
# Lo pasamos a forma matricial
X=np.transpose([X1,X2,X3])

# Calcular el output (suma de los tres números)
y=X1+X2+X3

print("Dimensiones: X ",np.shape(X),"   y ",np.shape(y))
```

Dividir los datos para entrenamiento y test

```
# Utilizar la librería sklearn para entrenamiento
# Seleccionar los que queremos utilizar para entrenamiento y para test
# Dividir para entrenamiento y test (80 % para entrenamiento y 20 % para
validación)
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y,
test_size=0.2, random_state=7)
```

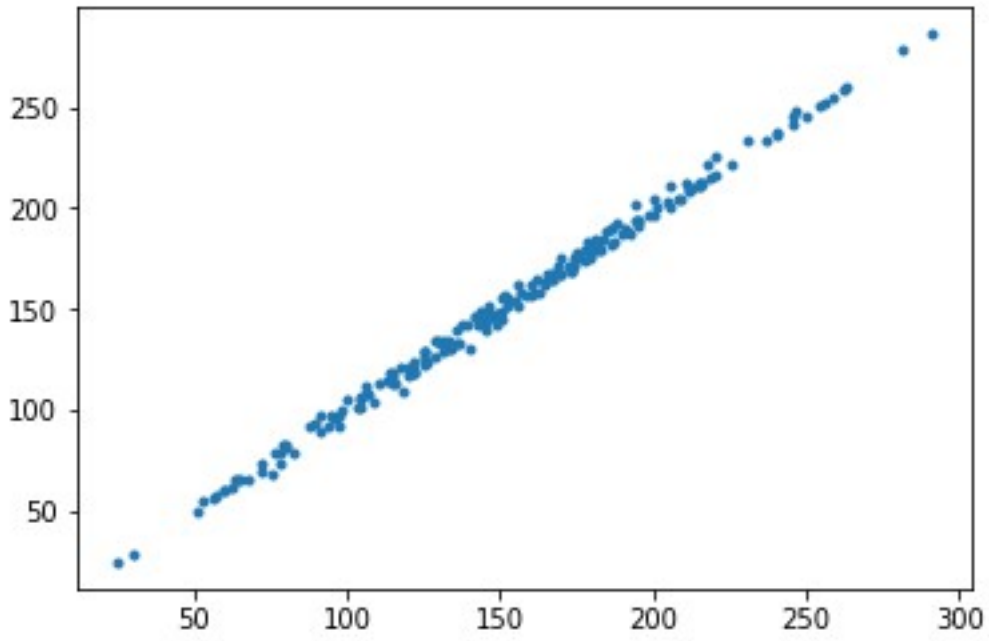
Crear la Red Neuronal y ajustarla a los datos de entrenamiento

```
# Utilizar el método de red neuronal de la librería sklearn
# definimos el número de capas ocultas a utilizar
mlp = MLPRegressor(hidden_layer_sizes=(10),max_iter=500,verbose=True)

# Entrenamiento de la red neuronal
mlp.fit(X_train,y_train)
```

Probar para los datos de test, evaluar la red y visualizar resultados

```
# Predecir para los datos de test
predictions = mlp.predict(X_test)

# Evaluar la red neuronal calculando el error
print("Correlación: ",np.corrcoef(predictions,y_test))
```

```
# Visualizar los resultados
plt.plot(predictions,y_test,'.')
```



Predecir para tres números

```
X_pred=[[5,12,6]]
y_pred=mlp.predict(X_pred)
print('La suma es', y_pred)
```

```
…
Iteration 495, loss = 0.31210478
Iteration 496, loss = 0.31160233
Iteration 497, loss = 0.31217012
Iteration 498, loss = 0.31087689
Iteration 499, loss = 0.31014266
Iteration 500, loss = 0.30982952
Correlación:  [[1.         0.99980753]
 [0.99980753 1.        ]]
La suma es [23.45007787]
```