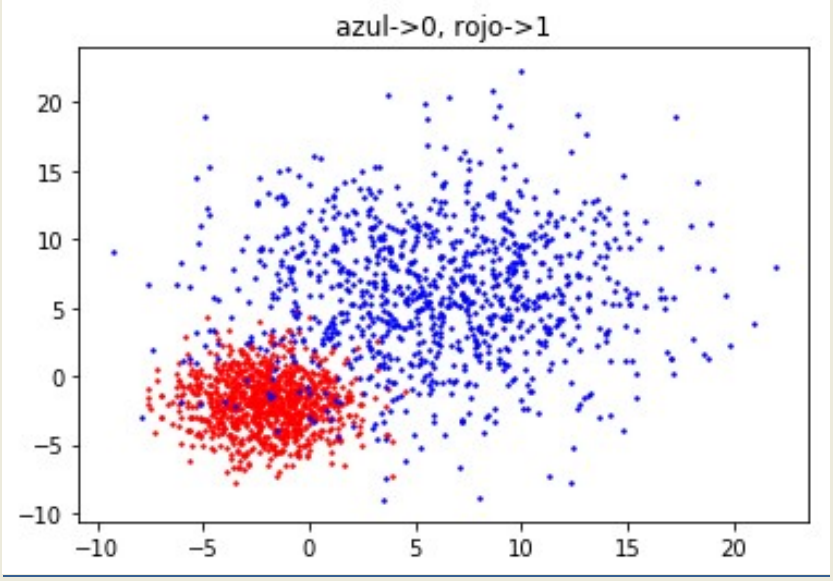
ML_CLASIFICACION_RN_04	Red Neuronal para clasificación de datos	ML

En esta práctica se construye una red neuronal que aprende a clasificar elementos según su clase. El conjunto de datos ('dataset') se lee directamente desde un fichero. El dataset está formado por la matriz 'X' que contiene los 'inputs' (secuencia de datos en 2 array), y la matriz 'y' que contiene los 'outputs' (clasificación por clase (0 -> azul, 1 -> rojo).

En esta práctica se construye la red neuronal de forma secuencial. En concreto, se define una capa de entrada con 2 inputs, una capa oculta con 4 neuronas, y una capa de salida o output.



Se utiliza también la red neuronal para predecir valores.

SOLUCIÓN

Importar las librerías necesarias para realizar la práctica.

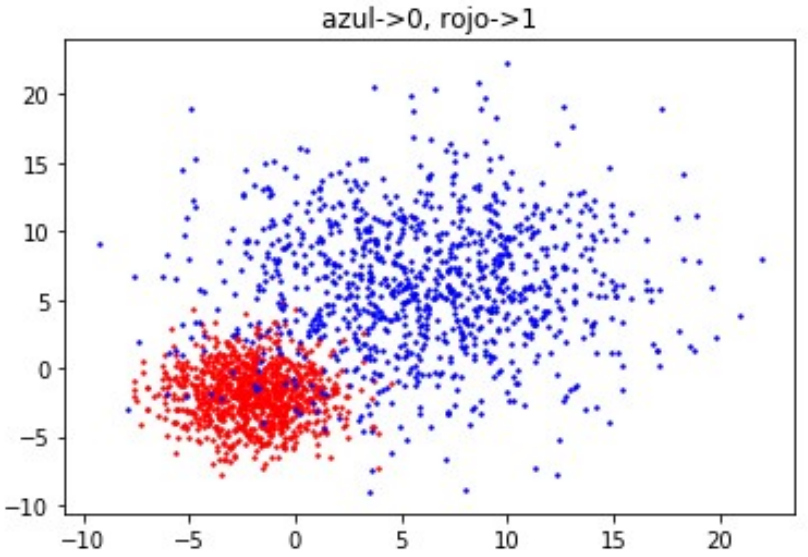
```
# Librerías

import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense
```

Cargar los datos y generar inputs (x) y outputs (y)

```
# Cargar los datos y generar inputs (X) y outputs (y) de la Red Neuronal
data=np.loadtxt('Datos.txt')
X=data[:,1:]
y=data[:,0]

# Mostrar los datos
colors = ['red', 'blue']
plt.scatter(X[y>0,0], X[y>0,1], c=colors[0],s=2)    # en rojo los '1'
plt.scatter(X[y==0,0], X[y==0,1], c=colors[1],s=2)  # en azul los '0'
plt.title("azul->0, rojo->1")
plt.show()
```



Crear la red neuronal de forma secuencial. Compilarla y ajustarla

```
# Crear la Red Neuronal
# input_dim  especifica el número de neuronas de la capa de entrada
# activation: 'relu', 'sigmoid' 'tanh'
# Definir modelo de red neuronal de forma secuencial
model = Sequential()
# Crar capa de entrada de 2 inputs (input_dim) y capa oculta de 4 neuronas
model.add(Dense(4, input_dim=2, activation='relu'))
# Crear la capa de salida con 1 neurona
model.add(Dense(1, activation='sigmoid'))
```

```
# Compilar la Red Neuronal
# binary_crossentropy: pérdida logarítmica
# adam: algoritmo del descenso de gradiente
model.compile(loss='binary_crossentropy', optimizer='adam'
,metrics=['accuracy'])

# Ajustar la red neuronal
model.fit(X, y, epochs=10, batch_size=128)
```

Evaluar la red neuronal

```
# Evaluar la red neuronal
test_loss, test_acc = model.evaluate(X, y)
print('test_acc:', test_acc)
print('test_loss:', test_loss)
```

Predecir para nuevos valores

```
# Predecir para un nuevo valor
X_pred = np.array([[5, 5]])
y_pred = model.predict_classes(X_pred)
# Mostrar la entrada (input) y la salida (output) resultado de la red neuronal

print("X=%s, Predicción=%s" % (X_pred[0], y_pred[0]))

X_pred = np.array([[-5, -5]])
y_pred = model.predict_classes(X_pred)
# Mostrar la entrada (input) y la salida (output) resultado de la red neuronal

print("X=%s, Predicción=%s" % (X_pred[0], y_pred[0]))


X=[5 5], Predicción=[0]
X=[-5 -5], Predicción=[1]
```