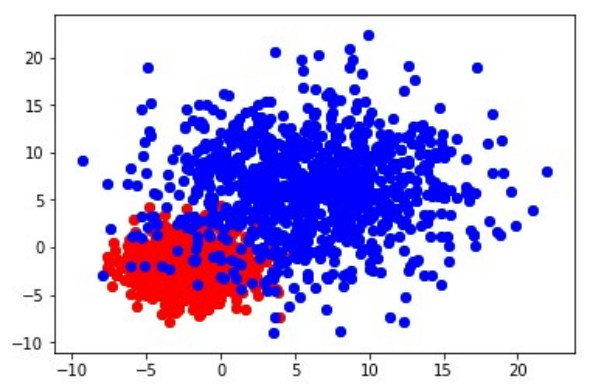
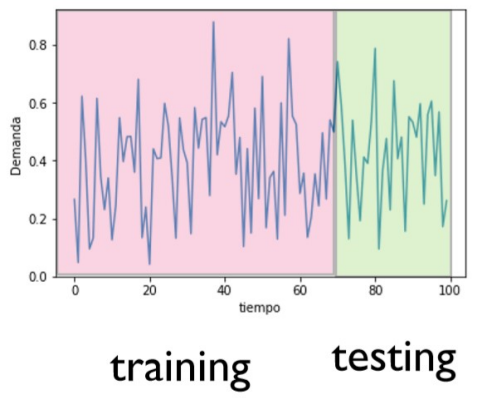
ML_CLASIFICACION_RN_03	Red Neuronal para clasificación de datos	ML

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).

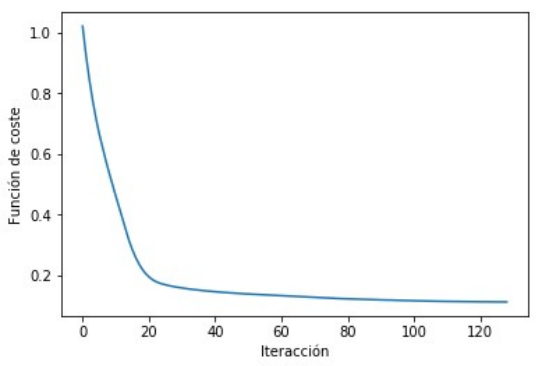
Se divide 'X' para ejecutar el entrenamiento (X_train,y_train) y la validación o test de la red neuronal (X_test,y_test). Se utiliza la librería 'sklearn' para entrenar y testar la red neuronal.



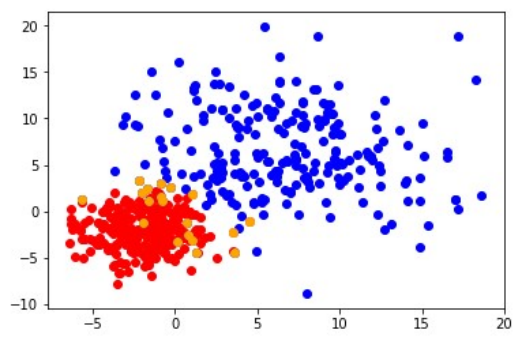
Conjunto de datos según clase (rojo -> 1, azul -> 0)



Del conjunto de datos, utilizamos una parte para entrenamiento y la otra para validación



Evaluamos la convergencia (reducción del error) en función del número de iteraciones



Detectamos los falsos positivos (en naranja), del conjunto de validación (test) para los que la red neuronal no funciona, no clasifica bien

SOLUCIÓN

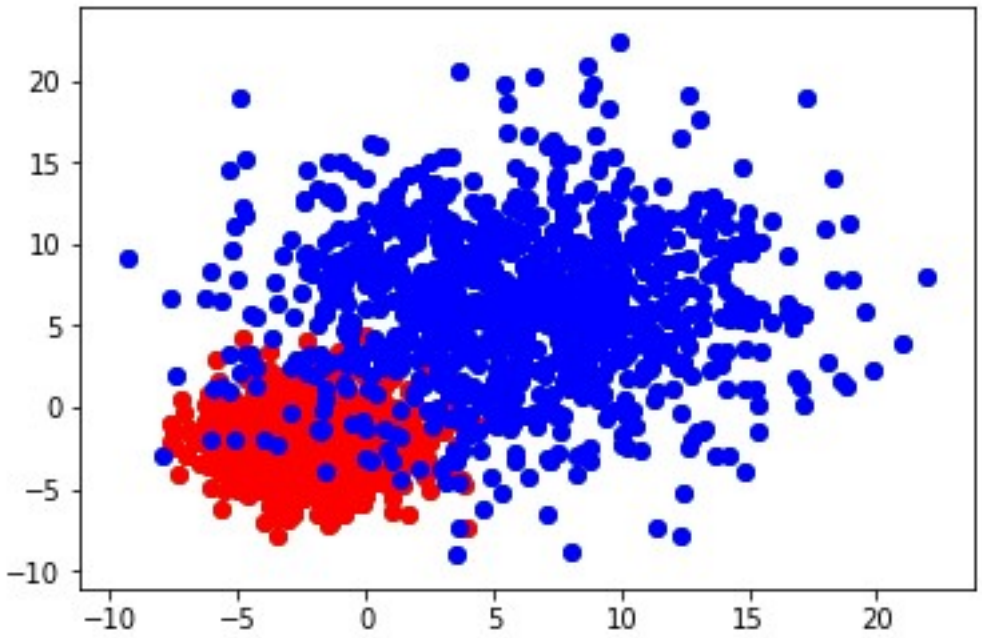
Importar las librerías necesarias para realizar la práctica.

```
# Librerías
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

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[:,0], X[:,1], c='g')   # todos los datos en verde
plt.scatter(X[y>0,0], X[y>0,1], c=colors[0])    # en rojo los '1'
plt.scatter(X[y==0,0], X[y==0,1], c=colors[1])  # en azul los '0'
plt.show()
```



Dividir los datos para entrenamiento y test

```
# Dividir los datos 'X' para entrenamiento y validación(test)
ntrain=int(3*len(y)/4)
X_train=X[:ntrain,:];y_train=y[:ntrain]
X_test=X[ntrain:,:];y_test=y[ntrain:]
print ("Entrenamiento:   X ",np.shape(X_train),"   y ",np.shape(y_train))
print ("Test:   X ",np.shape(X_test),"   y ",np.shape(y_test))

Entrenamiento:   X  (1500, 2)    y  (1500,)
Test:   X  (500, 2)    y  (500,)
```

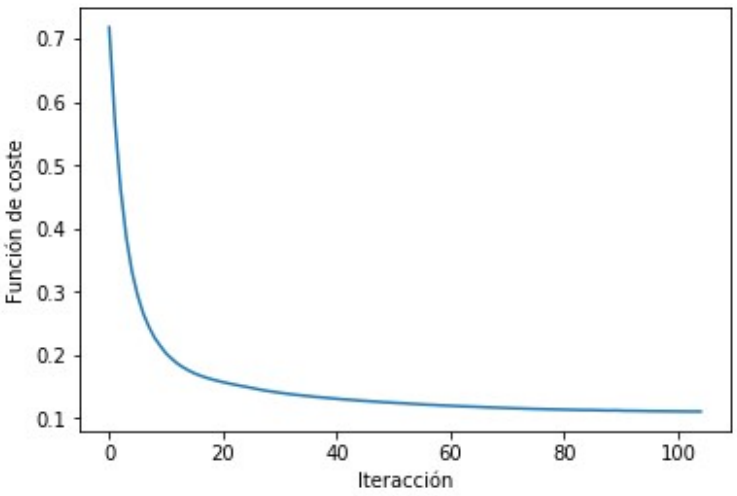
Crear la Red Neuronal y ajustarla a los datos de entrenamiento

```
# Crear el clasificador basado en Redes Neuronales
# definir numero de capas y de hidden nodes, función de activación
mlp =
MLPClassifier(hidden_layer_sizes=(10,10,10),activation='relu',max_iter=500)

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

Visualizar la función de coste

```
# Gráfica para ver rendimiento del entranamiento
plt.plot(mlp.loss_curve_)
# mlp.loss_curve nos da una idea de como converge con los datos de
entrenamiento
plt.xlabel('Iteracción')
plt.ylabel('Función de coste')
plt.show()
```



Predecir para los datos de test

```
# Validar (test) la red neuronal
y_pred = mlp.predict(X_test)
```

Evaluar la red neuronal con la matriz de confusión y con el informe de clasificación

```
# Evaluar la red neuronal
print("Matriz de confusión")
print(confusion_matrix(y_test,y_pred))
# C_ii Positivos verdaderos, C_ij!=0 Falsos Positivos
print()
print("Falsos positivos (0): ",confusion_matrix(y_test,y_pred)[0][1])
print("Falsos positivos (1): ",confusion_matrix(y_test,y_pred)[1][0])
print()

# Aplicar métrica para clasificar los datos usados para validación (test)
print("Clasificación de los resultados de la Validación(test)")
print("precision = num detecciones correctas / numero detecciones")
print("recall = num detecciones correctas / numero total de objetos en esa
clase")
print()
print(classification_report(y_test,y_pred))
```

```
Matriz de confusión
[[222  15]
 [  4 259]]

Falsos positivos (0):  15
Falsos positivos (1):  4

Clasificación de los resultados de la Validación(test)
precision = num detecciones correctas / numero detecciones
recall = num detecciones correctas / numero total de objetos en esa clase

             precision    recall  f1-score   support

        0.0       0.98      0.94      0.96       237
        1.0       0.95      0.98      0.96       263

avg / total       0.96      0.96      0.96       500
```

Visualizar los falsos positivos

```
# Gráfica para comprobar datos de validación que son falsos positivos
plt.scatter(X_test[y_test==1,0],X_test[y_test==1,1],c='r')
plt.scatter(X_test[y_test==0,0],X_test[y_test==0,1],c='b')
plt.scatter(X_test[(y_test==1)&(y_pred==0),0],X_test[(y_test==1)&(y_pred==0),1
],c='orange')
```

```
plt.scatter(X_test[(y_test==0)&(y_pred==1),0],X_test[(y_test==0)&(y_pred==1),1
],c='orange')
plt.show()
```