David Alvarez 1112791148

In [19]:

```
import numpy as np
from sklearn import preprocessing
# Datos de prueba
input_data = np.array([[5.1, -2.9, 3.3],
[-1.2, 7.8, -6.1],
[3.9, 0.4, 2.1],
[7.3, -9.9, -4.5]])

print(input_data)
```

```
[[ 5.1 -2.9  3.3]
 [-1.2  7.8 -6.1]
 [ 3.9  0.4  2.1]
 [ 7.3 -9.9 -4.5]]
```

In [7]:

```
data_binarized = preprocessing.Binarizer(threshold=2.1).transform(input_data)
print("\nDatos binarizados:\n", data_binarized)
```

```

Datos binarizados:
 [[1. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
 [1. 0. 0.]]
```

In [20]:

```
print("\nANTES:")
print("Media =", input_data.mean(axis=0))
print("Desviación estándar =", input_data.std(axis=0))
```

```

ANTES:
Media = [ 3.775 -1.15  -1.3  ]
Desviación estándar = [3.12039661 6.36651396 4.0620192 ]
```

In [21]:

```
data_scaled = preprocessing.scale(input_data)
print("\nDESPUÉS:")
print("Media =", data_scaled.mean(axis=0))
print("Desviación estándar =", data_scaled.std(axis=0))
```

```

DESPUÉS:
Media = [1.11022302e-16 0.00000000e+00 2.77555756e-17]
Desviación estándar = [1. 1. 1.]
```

```
In [22]: data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0,
         1))
         data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
         print("\nMin max escalamiento de datos:\n", data_scaled_minmax)
```

```
Min max escalamiento de datos:
 [[0.74117647 0.39548023 1.        ]
 [0.         1.         0.        ]
 [0.6        0.5819209  0.87234043]
 [1.         0.         0.17021277]]
```

```
In [23]: data_normalized_l1 = preprocessing.normalize(input_data,
         norm='l1')
         data_normalized_l2 = preprocessing.normalize(input_data,
         norm='l2')
         print("\nL1 dato normalizado:\n", data_normalized_l1)
         print("\nL2 dato normalizado:\n", data_normalized_l2)
```

```
L1 dato normalizado:
 [[ 0.45132743 -0.25663717  0.2920354 ]
 [-0.0794702   0.51655629 -0.40397351]
 [ 0.609375    0.0625      0.328125  ]
 [ 0.33640553 -0.4562212  -0.20737327]]

L2 dato normalizado:
 [[ 0.75765788 -0.43082507  0.49024922]
 [-0.12030718  0.78199664 -0.61156148]
 [ 0.87690281  0.08993875  0.47217844]
 [ 0.55734935 -0.75585734 -0.34357152]]
```

In [26]:

```
import numpy as np
from sklearn import preprocessing

input_labels = ['red', 'black', 'red', 'green', 'black', 'yellow',
'white']

encoder = preprocessing.LabelEncoder()
encoder.fit(input_labels)

print("\nMapeo de etiquetas:")
for i, item in enumerate(encoder.classes_):
    print(item, '-->', i)

test_labels = ['green', 'red', 'black']
encoded_values = encoder.transform(test_labels)
print("\nLabels =", test_labels)
print("Encoded values =", list(encoded_values))

encoded_values = [3, 0, 4, 1]
decoded_list = encoder.inverse_transform(encoded_values)
print("\nEncoded values =", encoded_values)
print("Decoded labels =", list(decoded_list))
```

```

Mapeo de etiquetas:
black --> 0
green --> 1
red --> 2
white --> 3
yellow --> 4

Labels = ['green', 'red', 'black']
Encoded values = [1, 2, 0]

Encoded values = [3, 0, 4, 1]
Decoded labels = ['white', 'black', 'yellow', 'green']
```