In [15]:

```
# PROCESAMIENTO DIGITAL
# Se importan las librería numpy y las funciones de preprocesamiento
import numpy as np
from sklearn import preprocessing
# Datos de prueba
input_data = np.array([[-2.2, 6.4, -3.3], [0.2, -5.8, 2.1], [3.9, -0.4, 2.1], [-7.3, -9.9,
-4.5]])
print(input_data)
```

```
[[-2.2  6.4 -3.3]
 [ 0.2 -5.8  2.1]
 [ 3.9 -0.4  2.1]
 [-7.3 -9.9 -4.5]]
```

In [16]:

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

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

In [17]:

```
# Imprimir la media y la desviación estándar
print("\nANTES:")
print("Media =", input_data.mean(axis=0))
print("Desviación estándar =", input_data.std(axis=0))
```

```
ANTES:
Media = [-1.35  -2.425 -0.9  ]
Desviación estándar = [4.06478782 6.1083447  3.02985148]
```

In [18]:

```
# Remover la media
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 = [ 5.55111512e-17 -1.11022302e-16  0.00000000e+00]
Desviación estándar = [1. 1. 1.]
```

In [19]:

```
# Escalamiento Min Max
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:
 [[0.45535714 1.         0.18181818]
 [0.66964286 0.25153374 1.        ]
 [1.         0.58282209 1.        ]
 [0.         0.         0.        ]]
```

In [20]:

```
# Normalización de datos
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:
 [[-0.18487395  0.53781513 -0.27731092]
 [ 0.02469136 -0.71604938  0.25925926]
 [ 0.609375   -0.0625      0.328125  ]
 [-0.33640553 -0.4562212  -0.20737327]]

L2 dato normalizado:
 [[-0.29219276  0.8500153  -0.43828914]
 [ 0.03240593 -0.93977201  0.34026228]
 [ 0.87690281 -0.08993875  0.47217844]
 [-0.55734935 -0.75585734 -0.34357152]]
```

In [21]:

```
# Manejo de etiquetas
import numpy as np
from sklearn import preprocessing
# Se definen algunas etiquetas simples
input_labels = ['rojo', 'negro', 'rojo', 'verde', 'azul', 'violeta', 'blanco']
# Se crea un codificador de etiquetas y se ajustan las etiquetas
encoder = preprocessing.LabelEncoder()
encoder.fit(input_labels)
# Se imprime el mapeo entre palabras y números
print("\nMapeo de etiquetas:")
for i, item in enumerate(encoder.classes_):
    print(item, '-->', i)
# Codificar un conjunto de etiquetas con el codificador
test_labels = ['rojo', 'azul', 'violeta']
encoded_values = encoder.transform(test_labels)
print("\nLabels =", test_labels)
print("Encoded values =", list(encoded_values))
# Decodificar un conjunto de valores usando el codificador
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:
azul --> 0
blanco --> 1
negro --> 2
rojo --> 3
verde --> 4
violeta --> 5

Labels = ['rojo', 'azul', 'violeta']
Encoded values = [3, 0, 5]

Encoded values = [3, 0, 4, 1]
Decoded labels = ['rojo', 'azul', 'verde', 'blanco']
```

In [ ]: