In [27]:
```
#PRESENTADO POR:Orfilia Castillo Maturana
#codigo:1003853896
#COMPUTACIÓN BLANDA - Sistemas y Computación
# ------------------------------------------------------
#                     RESUMEN
# En el siguiente documento lo que se realizara
# es la creacion de arreglos o tambien conocidas comomatices
# se manejan estas matrices y se utilizan propiedades
# de estadisticas para sacar de estos arreglos matriciales
# los valores de la media y la desviacion estandar.
# se realiza tambien el escalamiento de esta matrix
# y se normaliza con funciones de preprocesamiento.
# ------------------------------------------------------
#   Introducción
#
# para esta leccion utilizaremos las librerias numpy: es un programa
# que provee python con arreglos
# multidimensionales de alta eficiencia  y diseñados
# con el fin de que se utilizaran en calculos cientificos
# un arreglo puede contener:
#    *tiempos discretos de un experimento o simulación.
#    *señales grabadas por un instrumento de medida.
#    *pixeles de una imagen entre otras cosas.
# y las funciones de preprocesamiento.
# ------------------------------------------------------

# 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([[7.1, -4.9, 5.3],
[-3.2, 9.8, -8.1],
[5.9, -2.4, 4.1],
[9.3, 11.9, -6.5]])
print(input_data)
```

```
[[ 7.1 -4.9  5.3]
 [-3.2  9.8 -8.1]
 [ 5.9 -2.4  4.1]
 [ 9.3 11.9 -6.5]]
```

In [28]:
```
# Binarizar los datos
data_binarized = preprocessing.Binarizer(threshold=3.4).transform(input_data)
print("\nDatos binarizados:\n", data_binarized)
```

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

In [29]:
```
# 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 = [ 4.775  3.6   -1.3  ]
Desviación estándar = [4.76307411 7.34132141 6.04152299]
```

In [30]:
```
# 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 = [-2.77555756e-17  0.00000000e+00  0.00000000e+00]
Desviación estándar = [1. 1. 1.]
```

In [31]:
```
# 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:
 [[0.824      0.         1.        ]
 [0.         0.875      0.        ]
 [0.728      0.14880952 0.91044776]
 [1.         1.         0.11940299]]
```

In [36]:
```
# 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:
 [[ 0.41040462 -0.28323699  0.30635838]
 [-0.15165877  0.46445498 -0.38388626]
 [ 0.47580645 -0.19354839  0.33064516]
 [ 0.33574007  0.42960289 -0.23465704]]

L2 dato normalizado:
 [[ 0.70125381 -0.48396389  0.52347115]
 [-0.24407577  0.74748205 -0.6178168 ]
 [ 0.77888213 -0.31683341  0.54125707]
 [ 0.56561349  0.723742   -0.39532126]]
```

In [40]:
```
# Manejo de etiquetas
import numpy as np
from sklearn import preprocessing
# Se definen algunas etiquetas simples
input_labels = ['red', 'black', 'red', 'green', 'black', 'yellow',
'white']
# 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 = ['white', 'yellow', 'black']
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 = [2, 1, 4, 0]
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 = ['white', 'yellow', 'black']
Encoded values = [3, 4, 0]

Encoded values = [2, 1, 4, 0]
Decoded labels = ['red', 'green', 'yellow', 'black']
```

In [ ]: