Data Preprocessing Techniques

In [6]:

```
import numpy as np
from sklearn import preprocessing
data=np.array([[-4,1.2,3.4,-6.6,8.8],
               [8.9,5.5,-7.0,-3.2,-8.2],
               [7.1,-6.4,1.2,-8.6,5.2],
               [8.9,6.2,-6.6,-4.2,1.1],
               [4.2,-2.1,8.5,4.3,-4.4]])
```

1. Mean Removal

In [7]:

```
print(data)
print("="*70)
data_standardlize=preprocessing.scale(data)
print(data_standardlize.mean(axis=0))
print(data.mean(axis=0))
print(data_standardlize.std(axis=0))
print(data.std(axis=0))
```

```
[[-4.   1.2  3.4 -6.6  8.8]
 [ 8.9  5.5 -7.  -3.2 -8.2]
 [ 7.1 -6.4  1.2 -8.6  5.2]
 [ 8.9  6.2 -6.6 -4.2  1.1]
 [ 4.2 -2.1  8.5  4.3 -4.4]]
======================================================================
[ 1.44328993e-16 -2.22044605e-17  0.00000000e+00 -1.33226763e-16
  6.66133815e-17]
[ 5.02  0.88 -0.1  -3.66  0.5 ]
[1. 1. 1. 1. 1.]
[4.82634437 4.72499735 5.96254979 4.40254472 6.18126201]
```

2. Scaling

In [8]:

```
print(data)
print("="*70)
data_scaler=preprocessing.MinMaxScaler(feature_range=(0,1))
data_scaled=data_scaler.fit_transform(data)
print(data)
data_scaled
```

```
[[-4.   1.2  3.4 -6.6  8.8]
 [ 8.9  5.5 -7.  -3.2 -8.2]
 [ 7.1 -6.4  1.2 -8.6  5.2]
 [ 8.9  6.2 -6.6 -4.2  1.1]
 [ 4.2 -2.1  8.5  4.3 -4.4]]
======================================================================
[[-4.   1.2  3.4 -6.6  8.8]
 [ 8.9  5.5 -7.  -3.2 -8.2]
 [ 7.1 -6.4  1.2 -8.6  5.2]
 [ 8.9  6.2 -6.6 -4.2  1.1]
 [ 4.2 -2.1  8.5  4.3 -4.4]]
```

```
Out[8]: array([[0.        , 0.6031746 , 0.67096774, 0.15503876, 1.        ],
               [1.        , 0.94444444, 0.        , 0.41860465, 0.        ],
               [0.86046512, 0.        , 0.52903226, 0.        , 0.78823529],
               [1.        , 1.        , 0.02580645, 0.34108527, 0.54705882],
               [0.63565891, 0.34126984, 1.        , 1.        , 0.22352941]])
```

3. Normalization

```
In [9]: print(data)
        print("="*70)
        data_normalized=preprocessing.normalize(data,norm="l1")
        data_normalized
```

```
[[-4.   1.2  3.4 -6.6  8.8]
 [ 8.9  5.5 -7.  -3.2 -8.2]
 [ 7.1 -6.4  1.2 -8.6  5.2]
 [ 8.9  6.2 -6.6 -4.2  1.1]
 [ 4.2 -2.1  8.5  4.3 -4.4]]
======================================================================
```

```
Out[9]: array([[-0.16666667,  0.05      ,  0.14166667, -0.275     ,  0.3666666
        7],
               [ 0.27134146,  0.16768293, -0.21341463, -0.09756098, -0.25
        ],
               [ 0.24912281, -0.2245614 ,  0.04210526, -0.30175439,  0.1824561
        4],
               [ 0.32962963,  0.22962963, -0.24444444, -0.15555556,  0.0407407
        4],
               [ 0.1787234 , -0.0893617 ,  0.36170213,  0.18297872, -0.1872340
        4]])
```

4. Binarization

```
In [10]: print(data)
         print("="*70)
         data_binarized=preprocessing.Binarizer(threshold=3.4).transform(data)
         data_binarized
```

```
[[-4.   1.2  3.4 -6.6  8.8]
 [ 8.9  5.5 -7.  -3.2 -8.2]
 [ 7.1 -6.4  1.2 -8.6  5.2]
 [ 8.9  6.2 -6.6 -4.2  1.1]
 [ 4.2 -2.1  8.5  4.3 -4.4]]
======================================================================
```

```
Out[10]: array([[0., 0., 0., 0., 1.],
                [1., 1., 0., 0., 0.],
                [1., 0., 0., 0., 1.],
                [1., 1., 0., 0., 0.],
                [1., 0., 1., 1., 0.]])
```

5. Label Encoding

```
In [19]: #Label Encoding
         label_encoder = preprocessing.LabelEncoder()
         input_classes = ['audi', 'ford', 'audi', 'toyota', 'ford', 'bmw']
```

```
label_encoder.fit(input_classes)
print ("\nClass mapping:")
for i, item in enumerate(label_encoder.classes_):
    print(item, '-->', i)

labels = ['toyota', 'ford', 'audi']
encoded_labels = label_encoder.transform(labels)
print("\nLabels =", labels)
print("Encoded labels =", list(encoded_labels))
```

```
Class mapping:
audi --> 0
bmw --> 1
ford --> 2
toyota --> 3

Labels = ['toyota', 'ford', 'audi']
Encoded labels = [3, 2, 0]
```

In [14]:

```
#one hot encoding
encoder = preprocessing.OneHotEncoder()
data1 = [[0, 2, 1, 12], [1, 3, 5, 3], [2, 3, 2, 12], [1, 2, 4,3]]
encoder.fit(data1)
encoded_vector = encoder.transform(data1).toarray()
print(encoded_vector)
```

```
[[1. 0. 0. 1. 0. 1. 0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 1. 0. 0. 0. 1. 1. 0.]
 [0. 0. 1. 0. 1. 0. 1. 0. 0. 0. 1.]
 [0. 1. 0. 1. 0. 0. 0. 1. 0. 1. 0.]]
```

In [ ]:

```

```