Assignment 4

Title: Execute feature scalings on give dataset.

Theory:

Feature Scalling is a technique to standardize the independent feature present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitude or values or units.

If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, heigher and consider smaller values as the lower values, ~~san~~ regardless of the unit of the values.

Example:

If an algorithm is not using the feature scaling method then it can consider the value 3000 meters to be greater than 5km but that's actually not true & in this case, then algorithm will give wrong predictions. So, we use Feature scaling to bring all values to same magnitudes & thus, tackle this issue.

Technique to perform Feature scaling:

Consider two most important ones:

Min-Max Normalization: This technique re-scales a feature or observation value with distribution value bet^n 0 & 1

$$X_{norm} = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

Standardization: It is a very effective technique which re-scales a feature value so that it has distribution with 0 mean values & variance equals to 1.

$$X_{new} = \frac{x_i \; X_{mean}}{\text{Standard Deviation.}}$$

Conclusion :

Thus, we have studied feature scaling one given dataset.

In [1]:

```
# Python code explaining How to perform Feature Scaling

""" PART 1
        Importing Libraries """

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Sklearn library
from sklearn import preprocessing
```

In [2]:

```
""" PART 2
        Importing Data """

data_set = pd.read_csv(r"C:\Users\prati\Desktop\Data_for_Missing_Values.csv")
data_set.head()
print(data_set)
# here Features - Age and Salary columns
# are taken using slicing
# to handle values with varying magnitude
x = data_set.iloc[:, 1:3].values
print ("\nOriginal data values : \n", x)
```

```
   Country   Age  Salary  Purchased
0   France  44.0   72000          0
1    Spain  27.0   48000          1
2  Germany  30.0   54000          0
3    Spain  38.0   61000          0
4  Germany  40.0    1000          1
5   France  35.0   58000          1
6    Spain   NaN   52000          0
7   France  48.0   79000          1
8  Germany  50.0   83000          0
9   France  37.0   67000          1

Original data values :
 [[4.4e+01 7.2e+04]
 [2.7e+01 4.8e+04]
 [3.0e+01 5.4e+04]
 [3.8e+01 6.1e+04]
 [4.0e+01 1.0e+03]
 [3.5e+01 5.8e+04]
 [    nan 5.2e+04]
 [4.8e+01 7.9e+04]
 [5.0e+01 8.3e+04]
 [3.7e+01 6.7e+04]]
```

In [3]:

```
""" PART 4
        Handling the missing values """

from sklearn import preprocessing

""" MIN MAX SCALER """

min_max_scaler = preprocessing.MinMaxScaler(feature_range =(0, 1))

# Scaled feature
x_after_min_max_scaler = min_max_scaler.fit_transform(x)

print ("\nAfter min max Scaling : \n", x_after_min_max_scaler)
```

```
After min max Scaling :
 [[0.73913043 0.86585366]
 [0.         0.57317073]
 [0.13043478 0.64634146]
 [0.47826087 0.73170732]
 [0.56521739 0.        ]
 [0.34782609 0.69512195]
 [       nan 0.62195122]
 [0.91304348 0.95121951]
 [1.         1.        ]
 [0.43478261 0.80487805]]
```

In [4]:

```
""" Standardisation """

Standardisation = preprocessing.StandardScaler()

# Scaled feature
x_after_Standardisation = Standardisation.fit_transform(x)

print ("\nAfter Standardisation : \n", x_after_Standardisation)
```

```
After Standardisation :
 [[ 0.71993143  0.66527061]
 [-1.62367514 -0.43586695]
 [-1.21009751 -0.16058256]
 [-0.10722383  0.16058256]
 [ 0.16849459 -2.59226136]
 [-0.52080146  0.02294037]
 [        nan -0.25234403]
 [ 1.27136827  0.98643574]
 [ 1.54708669  1.16995867]
 [-0.24508304  0.43586695]]
```

In [ ]: