

FEATURE SCALING

Exp :05

Date: 26-08-2025

Aim:

To **understand the importance of feature scaling** by implementing and comparing two common methods: **Standardization (StandardScaler)** and **Normalization (MinMaxScaler)**.

Algorithm:

1. Load the dataset and preprocess it by imputing missing values and encoding categorical data to create the **final feature set (X)**.
2. Initialize the **StandardScaler**; fit and transform the feature set (X) to create **standardized data**.
3. Calculate **Standardization (Z-score)** by subtracting the mean and dividing by the standard deviation, resulting in a mean of 0 and a standard deviation of 1.
4. Initialize the **MinMaxScaler**; fit and transform the feature set (X) to create **normalized data**.
5. Calculate **Normalization** by scaling the features to a specific range (e.g., 0 to 1).
6. Display the outputs of both methods to verify the data transformations.

Code:

```
import numpy as np
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler, MinMaxScaler
df = pd.read_csv("pre_process_datasample.csv")
df.Country.fillna(df.Country.mode()[0],inplace=True)
```

```

features=df.iloc[:, :-1].values

age=SimpleImputer(strategy="mean",missing_values=np.nan)

Salary=SimpleImputer(strategy="mean",missing_values=np.nan)

age.fit(features[:,[1]])

Salary.fit(features[:,[2]])

features[:,[1]]=age.transform(features[:,[1]])

features[:,[2]]=Salary.transform(features[:,[2]])

oh = OneHotEncoder(sparse_output=False)

Country=oh.fit_transform(features[:,[0]])

final_set=np.concatenate((Country,features[:,[1,2]]),axis=1)

sc=StandardScaler()

sc.fit(final_set)

feat_standard_scaler=sc.transform(final_set)

print("Standardized Data (StandardScaler - Z-Score):")

print(feat_standard_scaler)

mms=MinMaxScaler(feature_range=(0,1))

mms.fit(final_set)

feat_minmax_scaler=mms.transform(final_set)

print("\nNormalized Data (MinMaxScaler - 0 to 1):")

print(feat_minmax_scaler)

```

Output:

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

```

array([[1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.]])

```

```
array([[ 'France', 44.0, 72000.0],
       [ 'Spain', 27.0, 48000.0],
       [ 'Germany', 30.0, 54000.0],
       [ 'Spain', 38.0, 61000.0],
       [ 'Germany', 40.0, 63777.77777777778],
       [ 'France', 35.0, 58000.0],
       [ 'Spain', 38.77777777777778, 52000.0],
       [ 'France', 48.0, 79000.0],
       [ 'Germany', 50.0, 83000.0],
       [ 'France', 37.0, 67000.0]], dtype=object)
```

```
array([[1.0, 0.0, 0.0, 44.0, 72000.0],
       [0.0, 0.0, 1.0, 27.0, 48000.0],
       [0.0, 1.0, 0.0, 30.0, 54000.0],
       [0.0, 0.0, 1.0, 38.0, 61000.0],
       [0.0, 1.0, 0.0, 40.0, 63777.77777777778],
       [1.0, 0.0, 0.0, 35.0, 58000.0],
       [0.0, 0.0, 1.0, 38.77777777777778, 52000.0],
       [1.0, 0.0, 0.0, 48.0, 79000.0],
       [0.0, 1.0, 0.0, 50.0, 83000.0],
       [1.0, 0.0, 0.0, 37.0, 67000.0]], dtype=object)
```

```
array([[ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
        7.58874362e-01,  7.49473254e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
        -1.71150388e+00, -1.43817841e+00],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
        -1.27555478e+00, -8.91265492e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
        -1.13023841e-01, -2.53200424e-01],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
        1.77608893e-01,  6.63219199e-16],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
        -5.48972942e-01, -5.26656882e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
        0.00000000e+00, -1.07356980e+00],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
        1.34013983e+00,  1.38753832e+00],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
        1.63077256e+00,  1.75214693e+00],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
        -2.58340208e-01,  2.93712492e-01]])
```

```
array([[1.      , 0.      , 0.      , 0.73913043, 0.68571429],
       [0.      , 0.      , 1.      , 0.      , 0.      ],
       [0.      , 1.      , 0.      , 0.13043478, 0.17142857],
       [0.      , 0.      , 1.      , 0.47826087, 0.37142857],
       [0.      , 1.      , 0.      , 0.56521739, 0.45079365],
       [1.      , 0.      , 0.      , 0.34782609, 0.28571429],
       [0.      , 0.      , 1.      , 0.51207729, 0.11428571],
       [1.      , 0.      , 0.      , 0.91304348, 0.88571429],
       [0.      , 1.      , 0.      , 1.      , 1.      ],
       [1.      , 0.      , 0.      , 0.43478261, 0.54285714]])
```

Result:

The experiment successfully applied **StandardScaler (Standardization)** to set features with a mean of and standard deviation of, and **MinMaxScaler (Normalization)** to scale features into the to range. This transformation ensures all features contribute equally to model training, demonstrating the core importance of scaling. Thus, the python program was executed successfully, and the output is verified.