8_Feature_Scaling

January 14, 2025

[ ]:
```
'''
    Feature Scaling -->

    Feature scaling is a preprocessing technique used in machine learning to␣
↪standardize
    the range of independent variables (features) in the data. It ensures that␣
↪no feature
    dominates others purely due to differences in scale, which can negatively␣
↪impact certain
    models (like gradient-based algorithms).
'''
```

[17]:
```
#   Common types of feature scaling -->

#   Normalization (Min-Max Scaling) : Rescales the values to a fixed range,␣
↪usually [0, 1]
#   Standardization (Z-score Scaling) : Centers the data around the mean and␣
↪scales it according
#   to the standard deviation, giving it a mean of 0 and a standard deviation␣
↪of 1
```

[ ]:
```
#   Standardization -->

#   z = (x- )/

#   Where,

#   x = Original value
#     = Mean of dataset
#     = Standard deviation of dataset
```

[ ]:
```
#   Normalization -->

#   x  = x-min(x)/max(x)-min(x)

#   Where,
#   x = Original value
#   min(x) = minimum value in dataset
#   max(x) = maximum value in dataset
```

[18]:
```
#   Feature scaling is particularly important for algorithms like -->

#   K-nearest neighbors (KNN)
#   Support Vector Machines (SVM)
#   Gradient Descent-based models (e.g., logistic regression, neural networks)

#   Some algorithms, like decision trees and random forests, are less sensitive␣
 ↪to feature scaling
```

[19]:
```
#   Done After Splitting !
```

[ ]:
```
#   Let's Do Pre-Processing From Scratch !
```

[20]:
```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
```

[21]:
```
data = pd.read_csv('Data/Data.csv')
data
```

[21]:
```
   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
```

[22]:
```
x_data = data.iloc[:, :-1].values
y_data = data.iloc[:, -1].values
```

[23]:
```
impute = SimpleImputer(missing_values = np.nan, strategy = 'mean')
impute.fit(x_data[:, 1:3])
x_data[:, 1:3] = impute.transform(x_data[:, 1:3])
```

```
x_data
```

```
[23]: 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)
```

```
[24]: clt = ColumnTransformer(transformers = [('encoder', OneHotEncoder(), [0])],␣
       ↪remainder = 'passthrough')
      x_data = np.array(clt.fit_transform(x_data))
      x_data
```

```
[24]: 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)
```

```
[25]: encode = LabelEncoder()
      y_data = encode.fit_transform(y_data)
      y_data
```

```
[25]: array([0, 1, 0, 0, 1, 1, 0, 1, 0, 1])
```

```
[26]: x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size =␣
       ↪0.2, random_state = 1)
      print(x_train)
```

```
[[0.0 0.0 1.0 38.77777777777778 52000.0]
 [0.0 1.0 0.0 40.0 63777.77777777778]
 [1.0 0.0 0.0 44.0 72000.0]
 [0.0 0.0 1.0 38.0 61000.0]
 [0.0 0.0 1.0 27.0 48000.0]
 [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 35.0 58000.0]]
```

[27]:
```
sc = StandardScaler()
x_train[:, 3:] = sc.fit_transform(x_train[:, 3:])
x_test[:, 3:] = sc.transform(x_test[:, 3:])
```

[28]:
```
x_train
```

[28]:
```
array([[0.0, 0.0, 1.0, -0.19159184384578545, -1.0781259408412425],
       [0.0, 1.0, 0.0, -0.014117293757057777, -0.07013167641635372],
       [1.0, 0.0, 0.0, 0.566708506533324, 0.633562432710455],
       [0.0, 0.0, 1.0, -0.30453019390224867, -0.30786617274297867],
       [0.0, 0.0, 1.0, -1.9018011447007988, -1.420463615551582],
       [1.0, 0.0, 0.0, 1.1475343068237058, 1.232653363453549],
       [0.0, 1.0, 0.0, 1.4379472069688968, 1.5749910381638885],
       [1.0, 0.0, 0.0, -0.7401495441200351, -0.5646194287757332]],
      dtype=object)
```

[29]:
```
x_test
```

[29]:
```
array([[0.0, 1.0, 0.0, -1.4661817944830124, -0.9069571034860727],
       [1.0, 0.0, 0.0, -0.44973664397484414, 0.2056403393225306]],
      dtype=object)
```