MACHINE LEARNING

LAB WORK 3

Name: Arjun Unnikrishnan USN: 22BTRAD004

Boston Housing Dataset

Question 1. Load a dataset with (features of different scales) Boston Housing Dataset.

Code:

```
import pandas as pd

# Load the CSV file into a pandas DataFrame

boston_df = pd.read_csv('HousingData.csv')

# Display the first few rows of the DataFrame

print(boston_df.head())
```

Output:

Name: Arjun Unnikrishnan

USN: 22BTRAD004

Lab 3

Question 1. Load a dataset with (features of different scales) Boston Housing Dataset.

In [2]:
```
import pandas as pd
# Load the CSV file into a pandas DataFrame
boston_df = pd.read_csv('HousingData.csv')
# Display the first few rows of the DataFrame
print(boston_df.head())
```

```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222     18.7

        B  LSTAT  MEDV
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90    NaN  36.2
```

Question 2. Apply Min Max scaling to dataset

Code:

```
from sklearn.preprocessing import MinMaxScaler

# Apply Min-Max scaling

scaler = MinMaxScaler()

boston_scaled = pd.DataFrame(scaler.fit_transform(boston_df),
columns=boston_df.columns)

# Display the first few rows of the scaled DataFrame

print(boston_scaled.head())
```

Output:

Question 2. Apply Min Max scaling to dataset

In [3]:
```
from sklearn.preprocessing import MinMaxScaler
# Apply Min-Max scaling
scaler = MinMaxScaler()
boston_scaled = pd.DataFrame(scaler.fit_transform(boston_df), columns=boston_df.columns)
# Display the first few rows of the scaled DataFrame
print(boston_scaled.head())
```

```
       CRIM    ZN     INDUS  CHAS       NOX        RM       AGE       DIS  \
0  0.000000  0.18  0.067815   0.0  0.314815  0.577505  0.641607  0.269203
1  0.000236  0.00  0.242302   0.0  0.172840  0.547998  0.782698  0.348962
2  0.000236  0.00  0.242302   0.0  0.172840  0.694386  0.599382  0.348962
3  0.000293  0.00  0.063050   0.0  0.150206  0.658555  0.441813  0.448545
4  0.000705  0.00  0.063050   0.0  0.150206  0.687105  0.528321  0.448545

        RAD       TAX   PTRATIO         B     LSTAT      MEDV
0  0.000000  0.208015  0.287234  1.000000  0.089680  0.422222
1  0.043478  0.104962  0.553191  1.000000  0.204470  0.368889
2  0.043478  0.104962  0.553191  0.989737  0.063466  0.660000
3  0.086957  0.066794  0.648936  0.994276  0.033389  0.631111
4  0.086957  0.066794  0.648936  1.000000       NaN  0.693333
```

Question 3. Apply Standardization to dataset.

Code:

```
from sklearn.preprocessing import StandardScaler

# Apply Standardization

scaler = StandardScaler()

boston_standardized = pd.DataFrame(scaler.fit_transform(boston_df),
columns=boston_df.columns)

# Display the first few rows of the standardized DataFrame
```

print(boston_standardized.head())

Output:

```
In [4]: from sklearn.preprocessing import StandardScaler
        # Apply Standardization
        scaler = StandardScaler()
        boston_standardized = pd.DataFrame(scaler.fit_transform(boston_df), columns=boston_df.columns)
        # Display the first few rows of the standardized DataFrame
        print(boston_standardized.head())
```

```
       CRIM        ZN     INDUS      CHAS       NOX        RM       AGE  \
0 -0.413898  0.290525 -1.284840 -0.274265 -0.144217  0.413672 -0.118643
1 -0.411488 -0.479864 -0.587798 -0.274265 -0.740262  0.194274  0.371156
2 -0.411491 -0.479864 -0.587798 -0.274265 -0.740262  1.282714 -0.265225
3 -0.410908 -0.479864 -1.303877 -0.274265 -0.835284  1.016303 -0.812226
4 -0.406697 -0.479864 -1.303877 -0.274265 -0.835284  1.228577 -0.511911

        DIS       RAD       TAX   PTRATIO         B     LSTAT      MEDV
0  0.140214 -0.982843 -0.666608 -1.459000  0.441052 -1.082105  0.159686
1  0.557160 -0.867883 -0.987329 -0.303094  0.441052 -0.500165 -0.101524
2  0.557160 -0.867883 -0.987329 -0.303094  0.396427 -1.215000  1.324247
3  1.077737 -0.752922 -1.106115  0.113032  0.416163 -1.367479  1.182758
4  1.077737 -0.752922 -1.106115  0.113032  0.441052       NaN  1.487503
```

Question 5. Apply Robust scaling to the dataset.

Code:

```
from sklearn.preprocessing import RobustScaler

# Apply Robust scaling

scaler = RobustScaler()

boston_robust_scaled = pd.DataFrame(scaler.fit_transform(boston_df),
columns=boston_df.columns)

# Display the first few rows of the robust scaled DataFrame

print(boston_robust_scaled.head())
```

Output:

Question 5. Apply Robust scaling to the dataset.

```
In [5]: from sklearn.preprocessing import RobustScaler
        # Apply Robust scaling
        scaler = RobustScaler()
        boston_robust_scaled = pd.DataFrame(scaler.fit_transform(boston_df), columns=boston_df.columns)
        # Display the first few rows of the robust scaled DataFrame
        print(boston_robust_scaled.head())
```

```
       CRIM    ZN     INDUS  CHAS       NOX        RM       AGE       DIS  \
0 -0.071124  1.44 -0.571650   0.0  0.000000  0.496612 -0.237705  0.285777
1 -0.065090  0.00 -0.202943   0.0 -0.394286  0.287940  0.043033  0.569789
2 -0.065095  0.00 -0.202943   0.0 -0.394286  1.323171 -0.321721  0.569789
3 -0.063635  0.00 -0.581720   0.0 -0.457143  1.069783 -0.635246  0.924391
4 -0.053090  0.00 -0.581720   0.0 -0.457143  1.271680 -0.463115  0.924391

    RAD       TAX   PTRATIO         B     LSTAT      MEDV
0 -0.20 -0.087855 -1.339286  0.261902 -0.656155  0.351097
1 -0.15 -0.227390 -0.446429  0.261902 -0.232960  0.050157
2 -0.15 -0.227390 -0.446429  0.066675 -0.752798  1.692790
3 -0.10 -0.279070 -0.125000  0.153016 -0.863683  1.529781
4 -0.10 -0.279070 -0.125000  0.261902       NaN  1.880878
```

Question 5. Assess the impact of scaling on the dataset.

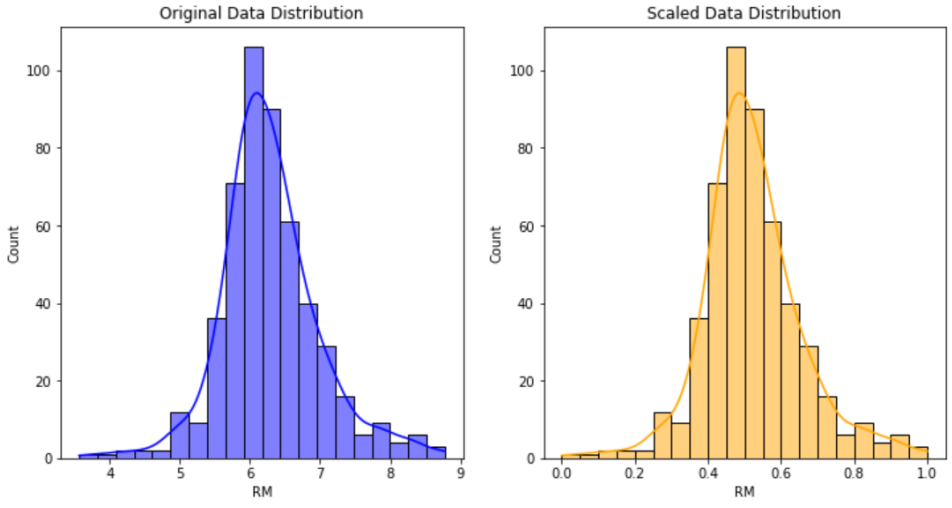
Code:

```
#Data Distribution Visualisation
import matplotlib.pyplot as plt
import seaborn as sns
# Histograms for original and scaled datasets
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(boston_df['RM'], bins=20, kde=True, color='blue')
plt.title('Original Data Distribution')
plt.subplot(1, 2, 2)
sns.histplot(boston_scaled['RM'], bins=20, kde=True, color='orange')
plt.title('Scaled Data Distribution')
plt.show()
```

Output:

Question 5. Assess the impact of scaling on the dataset.

In [7]:
```
#Data Distribution Visualisation
import matplotlib.pyplot as plt
import seaborn as sns
# Histograms for original and scaled datasets
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(boston_df['RM'], bins=20, kde=True, color='blue')
plt.title('Original Data Distribution')
plt.subplot(1, 2, 2)
sns.histplot(boston_scaled['RM'], bins=20, kde=True, color='orange')
plt.title('Scaled Data Distribution')
plt.show()
```



GitHub Link: https://github.com/arj1-1n/ML