FUNDAMENTALS OF MACHINE LEARNING

LAB ASSIGNMENT - 3

NAME - Kaparotu Venkata Surya Tharani
USN - 22BTRAD018
BRANCH - AI & DE

Questions -

1. Load a dataset with outliers values (Boston Housing Dataset).

CODE :

```
# importing modules
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

# loading the data
f1=pd.read_csv("HousingData.csv")
print(f1.head())
```

Output :

```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO       B  LSTAT  MEDV
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900    1  296     15.3  396.90   4.98  24.0
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242     17.8  396.90   9.14  21.6
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242     17.8  392.83   4.03  34.7
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222     18.7  394.63   2.94  33.4
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222     18.7  396.90    NaN  36.2
PS C:\Users\kvsth\Desktop\Term 7\Fundamentals of ML\Module 2>
```

2. Apply Min-Max scaling to dataset.

- Min-Max scaling is a data pre-processing technique where the minimum of feature is made equal to zero and the maximum of feature equal to one.
- MinMax Scaler shrinks the data within the given range, usually of 0 to 1.
- It transforms data by scaling features to a given range.
- It scales the values to a specific value range without changing the shape of the original distribution.

Code :

```
# import module
from sklearn.preprocessing import MinMaxScaler

# scale features
scaler = MinMaxScaler()
model=scaler.fit_transform(f1)
scaled_data=pd.DataFrame(model,columns=f1.columns)

#printing the data
print("Unnormalized Data \n",f1.head())
print("Min-Max normalized \n",scaled_data.head())
```

Output :

```
Unnormalized Data
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO      B  LSTAT  MEDV
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900    1  296     15.3  396.90   4.98  24.0
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242     17.8  396.90   9.14  21.6
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242     17.8  392.83   4.03  34.7
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222     18.7  394.63   2.94  33.4
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222     18.7  396.90    NaN  36.2
Min-Max normalized
       CRIM    ZN     INDUS  CHAS       NOX        RM  ...       RAD       TAX   PTRATIO        B     LSTAT      MEDV
0  0.000000  0.18  0.067815   0.0  0.314815  0.577505  ...  0.000000  0.208015  0.287234  1.000000  0.089680  0.422222
1  0.000236  0.00  0.242302   0.0  0.172840  0.547998  ...  0.043478  0.104962  0.553191  1.000000  0.204470  0.368889
2  0.000236  0.00  0.242302   0.0  0.172840  0.694386  ...  0.043478  0.104962  0.553191  0.989737  0.063466  0.660000
3  0.000293  0.00  0.063050   0.0  0.150206  0.658555  ...  0.086957  0.066794  0.648936  0.994276  0.033389  0.631111
4  0.000705  0.00  0.063050   0.0  0.150206  0.687105  ...  0.086957  0.066794  0.648936  1.000000       NaN  0.693333

[5 rows x 14 columns]
```

3. Apply Standardization to dataset.

- The standard deviation is converted to 1 and the mean to 0 through standardization.
- Standardized and rescaled data are produced by subtracting the mean from each data point and dividing the resulting value by the standard deviation.
- When features in the input data set are measured in multiple units (e.g., pounds, meters, miles, etc.) or when there are significant discrepancies between their ranges, data normalization becomes necessary.
- Many machine learning models encounter difficulties as a result of these variations in the ranges of initial features.
- For instance, in models based on distance computation, the distance will be determined by a specific feature if it has a wide range of values.

Code :

```
# import module
from sklearn.preprocessing import StandardScaler

# scale features
scaler = StandardScaler()
model=scaler.fit_transform(f1)
standardized_data=pd.DataFrame(model,columns=f1.columns)

#printing the data
print(standardized_data.head())
```

Output :

```
      CRIM    ZN     INDUS  CHAS      NOX       RM  ...   RAD       TAX   PTRATIO        B     LSTAT     MEDV
0 -0.071124  1.44 -0.571650   0.0  0.000000  0.496612  ... -0.20 -0.087855 -1.339286  0.261902 -0.656155  0.351097
1 -0.065090  0.00 -0.202943   0.0 -0.394286  0.287940  ... -0.15 -0.227390 -0.446429  0.261902 -0.232960  0.050157
2 -0.065095  0.00 -0.202943   0.0 -0.394286  1.323171  ... -0.15 -0.227390 -0.446429  0.066675 -0.752798  1.692790
3 -0.063635  0.00 -0.581720   0.0 -0.457143  1.069783  ... -0.10 -0.279070 -0.125000  0.153016 -0.863683  1.529781
4 -0.053090  0.00 -0.581720   0.0 -0.457143  1.271680  ... -0.10 -0.279070 -0.125000  0.261902       NaN  1.880878

[5 rows x 14 columns]
PS C:\Users\kvsth\Desktop\Term 7\Fundamentals of ML\Module 2> & C:/Users/kvsth/AppData/Local/Programs/Python/Python311/pyth
```

4. Apply Robust Scaling to the dataset.

- Sometimes an input variable may have outlier values.
- A probability distribution including outliers might be skewed, which makes it challenging to scale data using standardization since the outliers will affect the computed mean and standard deviation.
- Using the computed values to scale the variable after excluding the outliers from the mean and standard deviation computation is one method of standardizing input variables when there are outliers, is known as robust scaling.
- This scaling algorithm removes medians and scales the data by quantile range.
- This method removes the median and scales the data between the 1st and 3rd quartiles. that is, between the 25th and 75th quantiles. This range is also called the interquartile range.
- The median and interquartile range are then stored for use in future data using a transformation method.
- When the data set has outliers, the median and interquartile range perform better and outperform the sample mean and variance.
- Using RobustScaler() we can remove outliers and then use StandardScaler or MinMaxScaler to preprocess the data.

Code :

```
# import module
from sklearn.preprocessing import RobustScaler

# scale features
scaler = RobustScaler()
model=scaler.fit_transform(f1)
robust_scaleddata=pd.DataFrame(model,columns=f1.columns)

#printing the data
print(robust_scaleddata.head())
```

Output :

```
       CRIM    ZN     INDUS  CHAS       NOX        RM  ...   RAD       TAX   PTRATIO        B     LSTAT      MEDV
0 -0.071124  1.44 -0.571650   0.0  0.000000  0.496612  ... -0.20 -0.087855 -1.339286  0.261902 -0.656155  0.351097
1 -0.065090  0.00 -0.202943   0.0 -0.394286  0.287940  ... -0.15 -0.227390 -0.446429  0.261902 -0.232960  0.050157
2 -0.065095  0.00 -0.202943   0.0 -0.394286  1.323171  ... -0.15 -0.227390 -0.446429  0.066675 -0.752798  1.692790
3 -0.063635  0.00 -0.581720   0.0 -0.457143  1.069783  ... -0.10 -0.279070 -0.125000  0.153016 -0.863683  1.529781
4 -0.053090  0.00 -0.581720   0.0 -0.457143  1.271680  ... -0.10 -0.279070 -0.125000  0.261902       NaN  1.880878

[5 rows x 14 columns]
PS C:\Users\kvsth\Desktop\Term 7\Fundamentals of ML\Module 2>
```

5. Assess the impact of scaling on the dataset.

Assessing the impact of scaling involves looking at the scaled datasets and considering factors such as the range of values, the centering of data around zero, and the robustness to outliers.Impact of scaling gives information about how the data is changed and impacted after the different types of scaling.

Code :

```
# Statistical measures before scaling
print('Statistical measures before scaling:\n')
print(f1.describe(),'\n')

# Statistical measures after Min-Max scaling
print('Statistical measures after Min-Max scaling: \n')
print(pd.DataFrame(scaled_data).describe(),'\n')

# Statistical measures after standardization
print('Statistical measures after standardization: \n')
print(pd.DataFrame(standardized_data).describe(),'\n')
```

```
# Statistical measures after robust scaling
print('Statistical measures after robust scaling:','\n')
print(pd.DataFrame(robust_scaleddata).describe())
```

Output :

Before scaling :

```
Statistical measures before scaling:

             CRIM          ZN       INDUS        CHAS  ...     PTRATIO           B       LSTAT        MEDV
count  486.000000  486.000000  486.000000  486.000000  ...  506.000000  506.000000  486.000000  506.000000
mean     3.611874   11.211934   11.083992    0.069959  ...   18.455534  356.674032   12.715432   22.532806
std      8.720192   23.388876    6.835896    0.255340  ...    2.164946   91.294864    7.155871    9.197104
min      0.006320    0.000000    0.460000    0.000000  ...   12.600000    0.320000    1.730000    5.000000
25%      0.081900    0.000000    5.190000    0.000000  ...   17.400000  375.377500    7.125000   17.025000
50%      0.253715    0.000000    9.690000    0.000000  ...   19.050000  391.440000   11.430000   21.200000
75%      3.560263   12.500000   18.100000    0.000000  ...   20.200000  396.225000   16.955000   25.000000
max     88.976200  100.000000   27.740000    1.000000  ...   22.000000  396.900000   37.970000   50.000000

[8 rows x 14 columns]
```

Impacts after Min-Max Scaling :

```
Statistical measures after Min-Max scaling:

             CRIM          ZN       INDUS        CHAS  ...     PTRATIO           B       LSTAT        MEDV
count  486.000000  486.000000  486.000000  486.000000  ...  506.000000  506.000000  486.000000  506.000000
mean     0.040526    0.112119    0.389443    0.069959  ...    0.622929    0.898568    0.303130    0.389618
std      0.098013    0.233889    0.250583    0.255340  ...    0.230313    0.230205    0.197458    0.204380
min      0.000000    0.000000    0.000000    0.000000  ...    0.000000    0.000000    0.000000    0.000000
25%      0.000850    0.000000    0.173387    0.000000  ...    0.510638    0.945730    0.148869    0.267222
50%      0.002781    0.000000    0.338343    0.000000  ...    0.686170    0.986232    0.267660    0.360000
75%      0.039945    0.125000    0.646628    0.000000  ...    0.808511    0.998298    0.420116    0.444444
max      1.000000    1.000000    1.000000    1.000000  ...    1.000000    1.000000    1.000000    1.000000

[8 rows x 14 columns]
```

Impacts after Standardization :

```
Statistical measures after standardization:

              CRIM            ZN          INDUS          CHAS  ...       PTRATIO             B         LSTAT          MEDV
count  4.860000e+02  4.860000e+02  4.860000e+02  4.860000e+02  ...  5.060000e+02  5.060000e+02  4.860000e+02  5.060000e+02
mean   2.924044e-17 -5.665336e-17  1.023415e-16  1.462022e-17  ... -4.212704e-16 -7.442444e-16 -1.023415e-16 -5.195668e-16
std    1.001030e+00  1.001030e+00  1.001030e+00  1.001030e+00  ...  1.000990e+00  1.000990e+00  1.001030e+00  1.000990e+00
min   -4.138979e-01 -4.798643e-01 -1.555749e+00 -2.742649e-01  ... -2.707379e+00 -3.907193e+00 -1.536745e+00 -1.908226e+00
25%   -4.052217e-01 -4.798643e-01 -8.631004e-01 -2.742649e-01  ... -4.880391e-01  2.050715e-01 -7.820421e-01 -5.994557e-01
50%   -3.854983e-01 -4.798643e-01 -2.041324e-01 -2.742649e-01  ...  2.748590e-01  3.811865e-01 -1.798183e-01 -1.450593e-01
75%   -5.924715e-03  5.512847e-02  1.027405e+00 -2.742649e-01  ...  8.065758e-01  4.336510e-01  5.930706e-01  2.685231e-01
max    9.799358e+00  3.800078e+00  2.439061e+00  3.646110e+00  ...  1.638828e+00  4.410519e-01  3.532846e+00  2.989460e+00

[8 rows x 14 columns]
```

Impacts after Robust Scaling :

Statistical measures after robust scaling:

	CRIM	ZN	INDUS	CHAS	...	PTRATIO	B	LSTAT	MEDV
count	4.860000e+02	486.000000	486.000000	486.000000	...	506.000000	506.000000	486.000000	506.000000
mean	9.654425e-01	0.896955	0.107978	0.069959	...	-0.212309	-1.667632	0.130766	0.167123
std	2.506982e+00	1.871110	0.529504	0.255340	...	0.773195	4.379176	0.727962	1.153242
min	-7.112398e-02	0.000000	-0.714950	0.000000	...	-2.303571	-18.761003	-0.986775	-2.031348
25%	-4.939537e-02	0.000000	-0.348567	0.000000	...	-0.589286	-0.770476	-0.437945	-0.523511
50%	-7.982438e-18	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000
75%	9.506046e-01	1.000000	0.651433	0.000000	...	0.410714	0.229524	0.562055	0.476489
max	2.550697e+01	8.000000	1.398141	1.000000	...	1.053571	0.261902	2.699898	3.611285

[8 rows x 14 columns]