Detect and Remove Outliers using Pandas

February 14, 2025

1 Detect and Remove Outliers using Python (Pandas)

https://www.geeksforgeeks.org/detect-and-remove-the-outliers-using-python/ An Outlier is a data item/object that deviates significantly from the rest of the (so-called normal) objects. Identifying outliers is important in statistics and data analysis because they can have a significant impact on the results of statistical analyses. The analysis for outlier detection is referred to as outlier mining.

Outliers can skew the mean (average) and affect measures of central tendency, as well as influence the results of tests of statistical significance.

1.1 Using barplot graph

1.1.1 Outlier detection

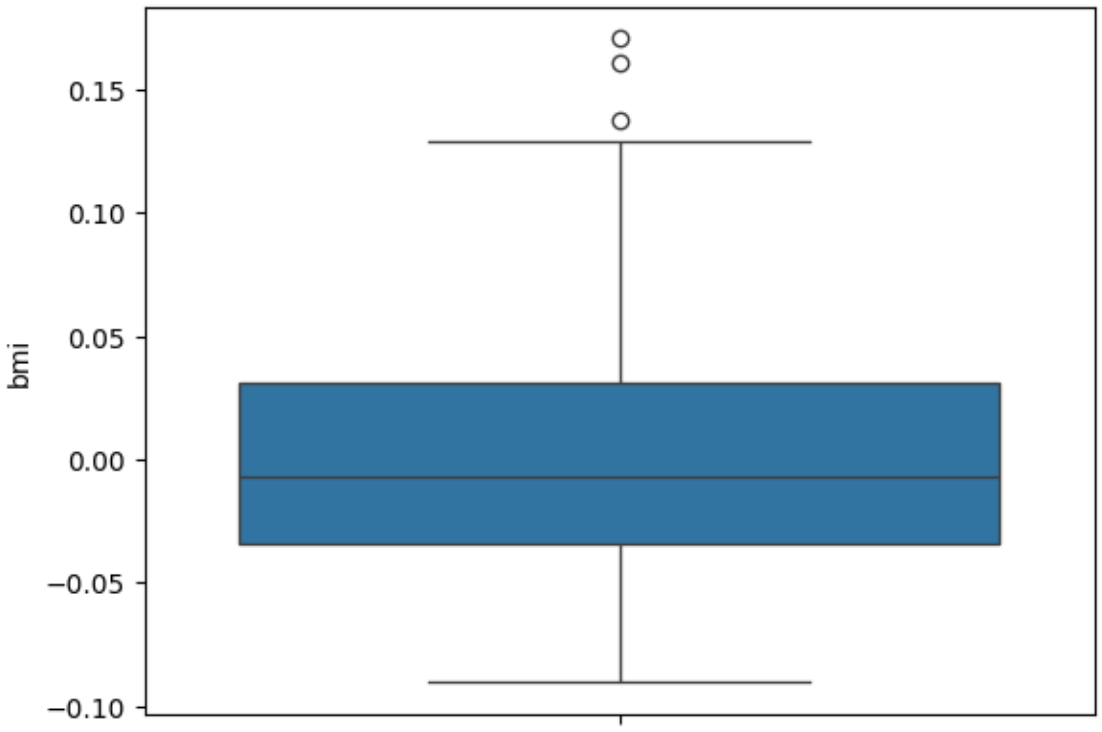
[1]:
```
import sklearn
from sklearn.datasets import load_diabetes
import pandas as pd
import matplotlib.pyplot as plt

diabetes = load_diabetes()

column_name = diabetes.feature_names
df_diabetes = pd.DataFrame(diabetes.data)
df_diabetes.columns = column_name
print(df_diabetes.head())
print(df_diabetes.info())
```

```
        age       sex       bmi        bp        s1        s2        s3  \
0  0.038076  0.050680  0.061696  0.021872 -0.044223 -0.034821 -0.043401
1 -0.001882 -0.044642 -0.051474 -0.026328 -0.008449 -0.019163  0.074412
2  0.085299  0.050680  0.044451 -0.005670 -0.045599 -0.034194 -0.032356
3 -0.089063 -0.044642 -0.011595 -0.036656  0.012191  0.024991 -0.036038
4  0.005383 -0.044642 -0.036385  0.021872  0.003935  0.015596  0.008142

         s4        s5        s6
0 -0.002592  0.019907 -0.017646
1 -0.039493 -0.068332 -0.092204
2 -0.002592  0.002861 -0.025930
3  0.034309  0.022688 -0.009362
```

```
4 -0.002592 -0.031988 -0.046641
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 442 entries, 0 to 441
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   age     442 non-null    float64
 1   sex     442 non-null    float64
 2   bmi     442 non-null    float64
 3   bp      442 non-null    float64
 4   s1      442 non-null    float64
 5   s2      442 non-null    float64
 6   s3      442 non-null    float64
 7   s4      442 non-null    float64
 8   s5      442 non-null    float64
 9   s6      442 non-null    float64
dtypes: float64(10)
memory usage: 34.7 KB
None
```

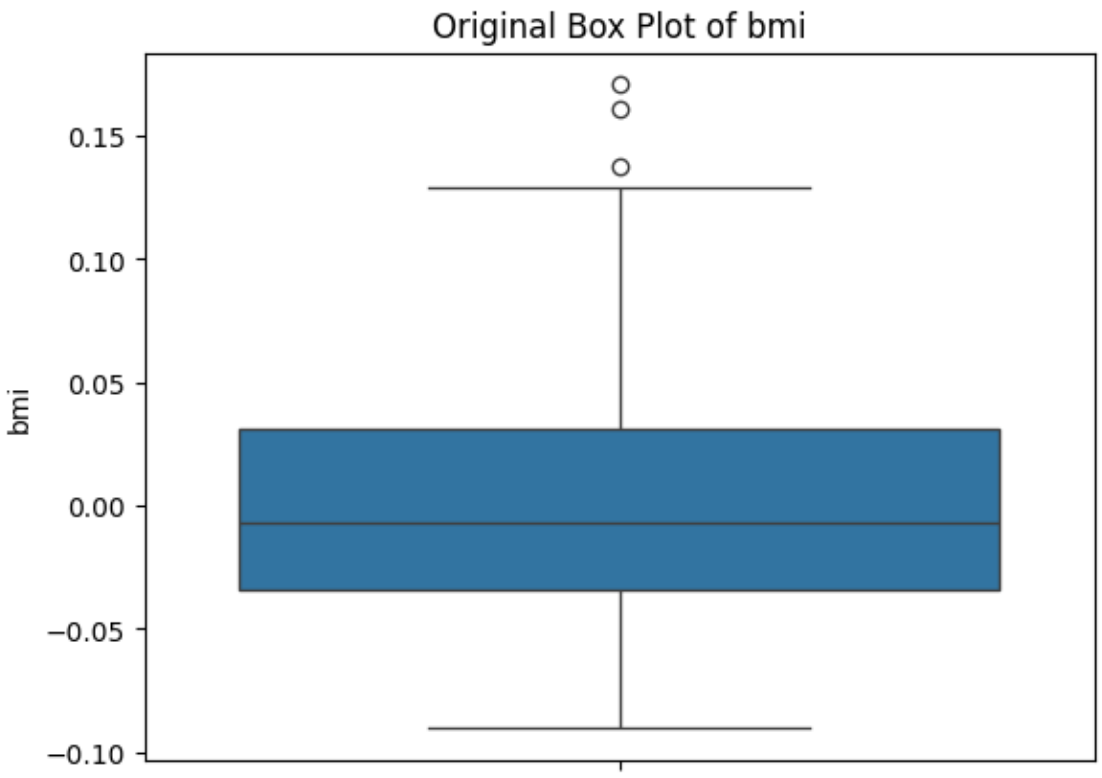
[2]:
```
import seaborn as sns
sns.boxplot(df_diabetes['bmi'])
```

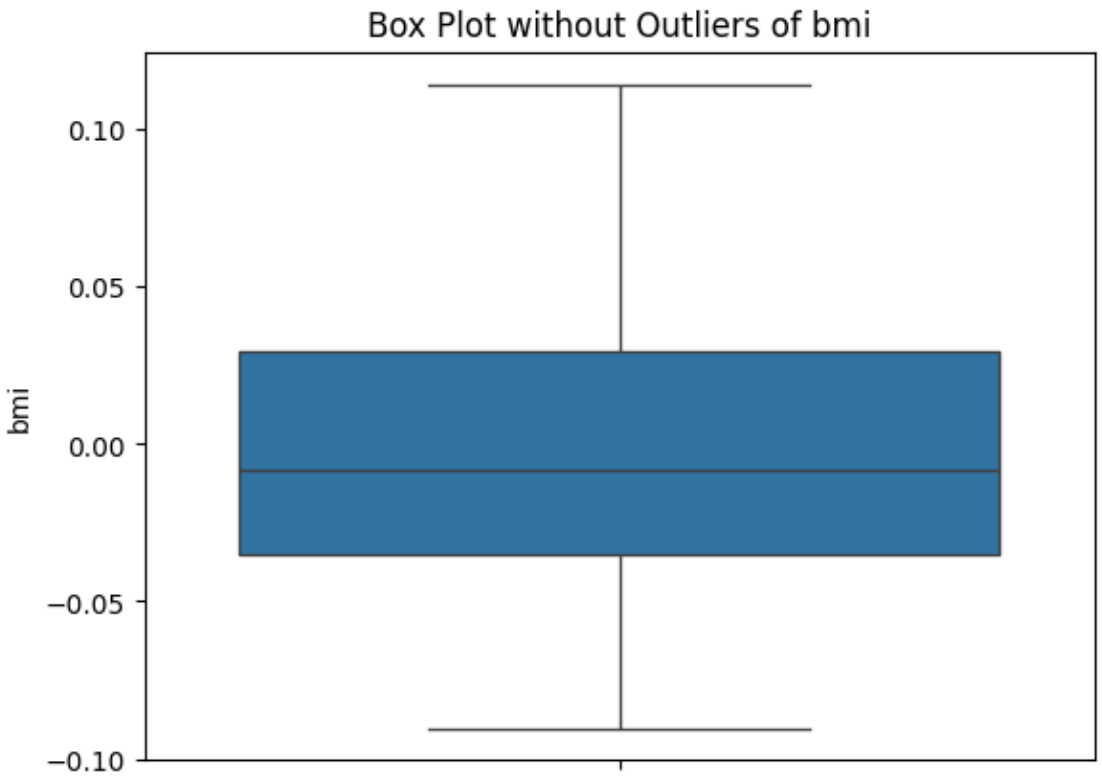
[2]: <Axes: ylabel='bmi'>



1.1.2 Outlier removal

[3]:
```
import seaborn as sns
import matplotlib.pyplot as plt

def removal_box_plot(df, column, threshold):
    sns.boxplot(df[column])
    plt.title(f'Original Box Plot of {column}')
    plt.show()

    removed_outliers = df[df[column] <= threshold]

    sns.boxplot(removed_outliers[column])
    plt.title(f'Box Plot without Outliers of {column}')
    plt.show()
    return removed_outliers

threshold_value = 0.12

no_outliers = removal_box_plot(df_diabetes,'bmi',threshold_value)
```

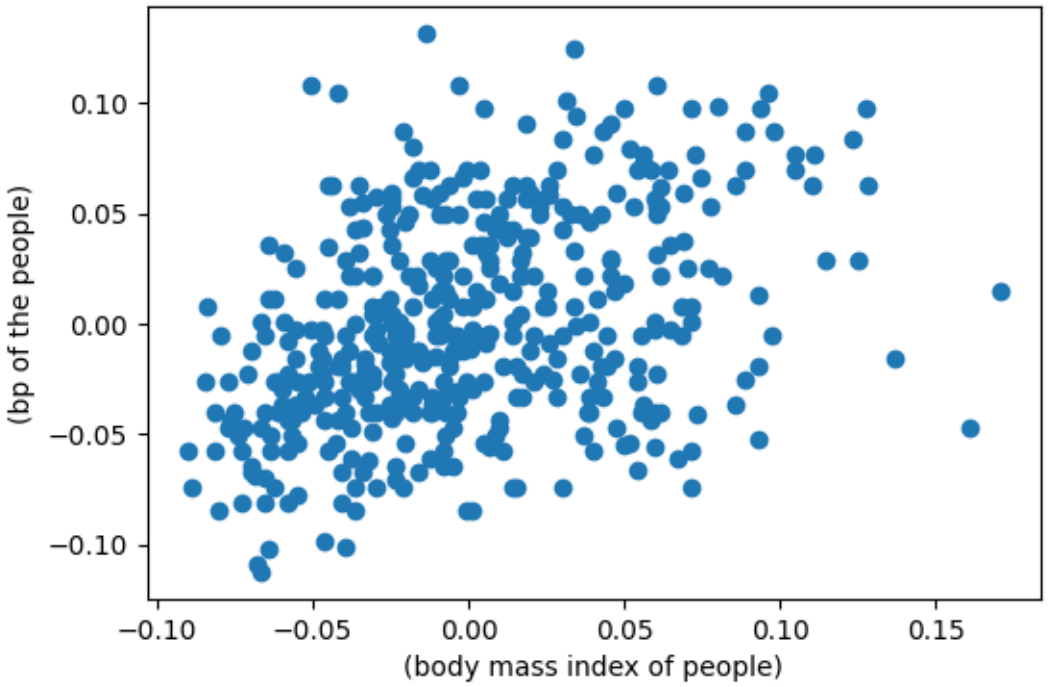




1.2 Using scatterplot graph

1.2.1 Outlier Detection

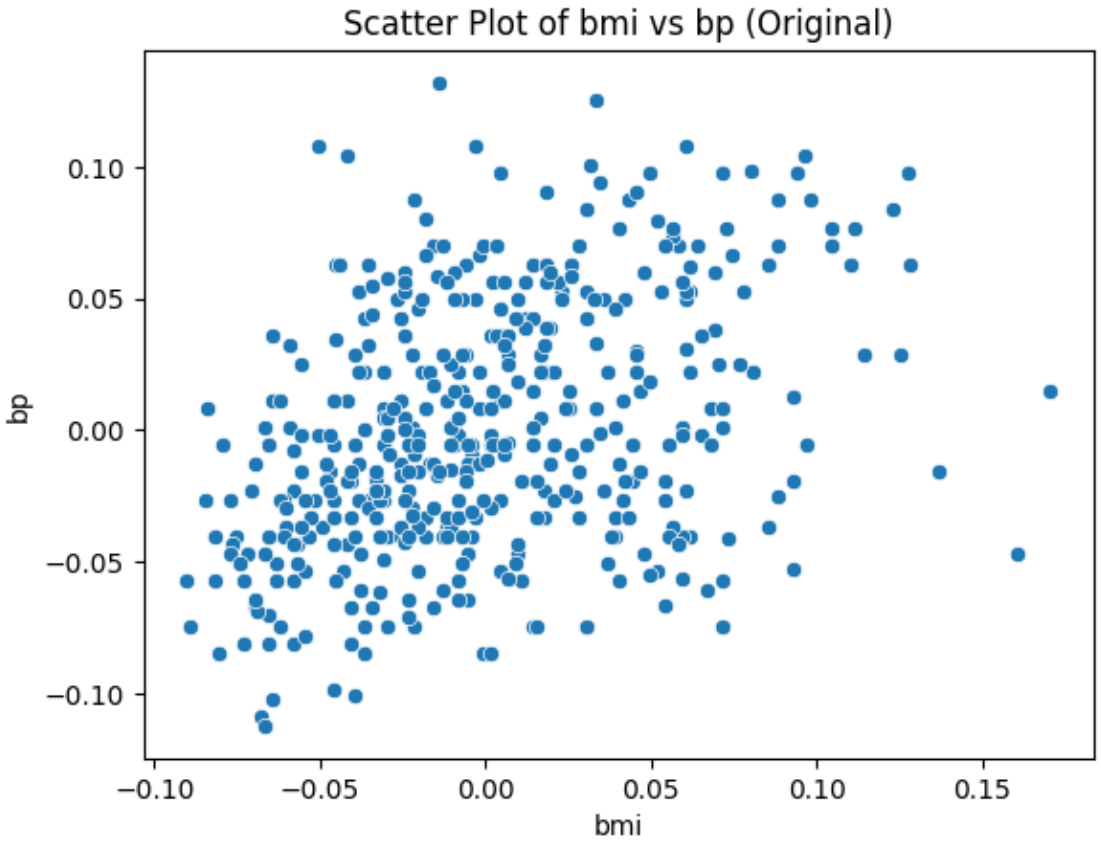
[4]:
```
fig, ax = plt.subplots(figsize=(6,4))
ax.scatter(df_diabetes['bmi'], df_diabetes['bp'])
ax.set_xlabel('(body mass index of people)')
ax.set_ylabel('(bp of the people)')
plt.show()
```

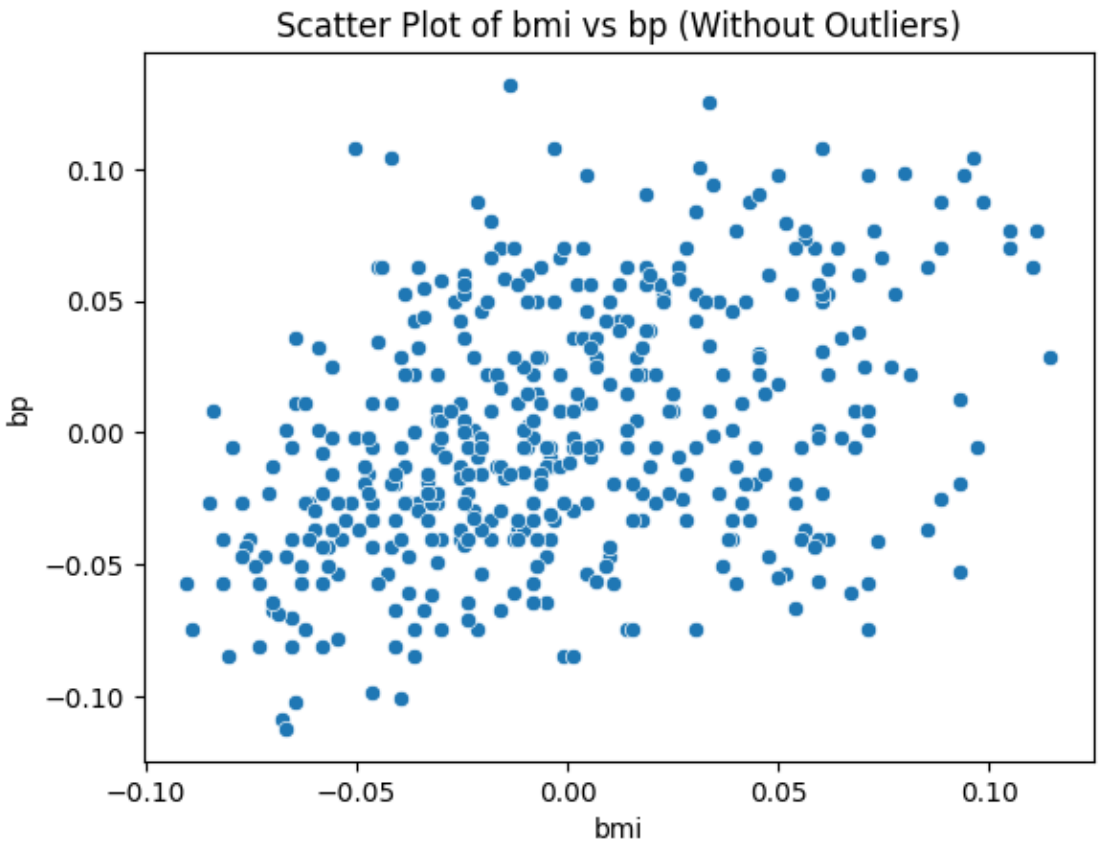


1.2.2 Outlier Removal

[10]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

def removal_scatter_plot(df, column1, column2):
    # Scatter plot before removing outliers
    sns.scatterplot(x=df[column1], y=df[column2])
    plt.title(f'Scatter Plot of {column1} vs {column2} (Original)')
    plt.xlabel(column1)
    plt.ylabel(column2)
    plt.show()

    # Removing outliers using the specified condition
    outlier_indices = np.where((df[column1] > 0.12) & (df[column2] < 0.8))
    removed_outliers = df.drop(outlier_indices[0])

    # Check if there are any data points left after outlier removal
    if removed_outliers.empty:
        print(f"No data points remain after removing outliers with the given␣
 ↪condition.")
    else:
        # Scatter plot after removing outliers
```

```
        sns.scatterplot(x=removed_outliers[column1],␣
 ↪y=removed_outliers[column2])
        plt.title(f'Scatter Plot of {column1} vs {column2} (Without Outliers)')
        plt.xlabel(column1)
        plt.ylabel(column2)
        plt.show()

    return removed_outliers

# Apply the function
no_outliers = removal_scatter_plot(df_diabetes, 'bmi', 'bp')
```





1.3 Using Z-score

Zscore = (data_point - mean) / std. deviation

1.3.1 Outlier detection

[29]:
```
from scipy import stats
import numpy as np

z = np.abs(stats.zscore(df_diabetes['age']))
print(z)
#print(df_diabetes['age'].info())
```

```
[0.80050009 0.03956713 1.79330681 1.87244107 0.11317236 1.94881082
 0.9560041  1.33508832 0.87686984 1.49059233 2.02518057 0.57139085
 0.34228161 0.11317236 0.95323959 1.1087436  0.11593688 1.48782782
 0.80326461 0.57415536 1.03237385 1.79607132 1.79607132 0.95323959
 1.33785284 1.41422259 2.25428981 0.49778562 1.10597908 1.41145807
 1.26148309 0.49778562 0.72413034 0.6477606  0.34228161 1.02960933
 0.26591186 0.19230663 0.03956713 0.03956713 0.11317236 2.10155031
 1.26148309 0.41865135 0.95323959 0.57139085 1.18511334 1.64333183
```

1.41145807 0.87963435 0.72413034 1.25871858 1.1087436 0.19230663
1.03237385 0.87963435 0.87963435 0.57415536 0.87686984 1.33508832
1.49059233 0.87963435 0.57415536 0.72689486 1.41145807 0.9560041
0.19230663 0.87686984 0.80050009 0.34228161 0.03956713 0.03956713
1.33508832 0.26591186 0.26591186 0.19230663 0.65052511 2.02518057
0.11317236 2.17792006 1.48782782 0.26591186 0.34504612 0.80326461
0.03680262 0.95323959 1.49059233 0.95323959 1.1087436 0.9560041
0.26591186 0.95323959 0.42141587 1.03237385 1.64333183 1.49059233
1.18234883 0.57415536 0.03680262 0.03956713 0.34228161 0.34228161
1.94881082 1.25871858 0.57415536 0.4950211 2.02518057 0.57139085
0.41865135 0.80050009 0.87686984 0.41865135 1.79607132 0.41865135
0.4950211 0.65052511 1.02960933 1.25871858 1.18511334 0.34228161
1.03237385 1.33508832 1.02960933 0.11317236 0.11593688 0.11593688
1.87244107 0.72413034 1.1087436 0.18954211 1.33785284 2.02518057
0.34228161 0.87963435 1.56696208 0.11593688 1.94881082 0.11317236
0.72413034 0.4950211 0.87686984 0.57415536 0.87686984 0.65052511
0.6477606 0.87963435 0.65052511 1.18511334 1.26148309 1.03237385
0.4950211 0.03680262 0.72689486 0.87686984 1.41145807 0.57415536
0.34504612 0.03956713 0.26867637 0.11593688 0.19230663 0.9560041
1.1087436 0.34228161 0.95323959 0.87963435 1.18511334 1.48782782
0.03680262 0.03956713 0.4950211 0.42141587 0.87686984 1.33785284
0.34228161 1.41145807 0.95323959 1.02960933 0.87686984 0.49778562
0.80326461 1.02960933 0.95323959 0.95323959 0.34228161 1.56696208
1.71970158 1.41422259 0.11317236 0.03956713 0.18954211 0.11593688
1.18234883 0.18954211 1.41422259 0.57139085 0.49778562 1.02960933
1.1087436 0.87686984 1.18234883 0.72689486 1.71693706 0.03956713
2.32789504 0.65052511 0.03680262 0.18954211 0.6477606 0.80050009
0.18954211 1.9460463 1.41145807 0.03680262 0.6477606 0.57139085
0.26591186 1.56419757 0.87963435 1.87244107 0.4950211 0.9560041
0.49778562 2.10155031 0.57415536 0.6477606 2.17792006 1.41145807
1.1087436 0.57415536 0.80326461 0.18954211 0.26591186 1.41145807
0.95323959 1.41145807 0.57139085 1.18234883 0.72413034 0.4950211
1.02960933 0.6477606 2.17792006 0.34228161 1.26148309 0.57415536
0.87686984 1.71970158 0.87963435 0.26867637 1.41145807 1.1087436
0.11317236 1.71693706 0.6477606 0.03680262 1.03237385 0.57415536
1.64056731 0.26591186 0.87686984 1.02960933 0.34504612 1.56696208
0.72413034 0.72689486 1.1087436 1.25871858 1.33508832 0.18954211
0.11317236 0.80050009 0.26591186 1.56419757 0.34228161 0.11593688
0.26591186 0.72689486 1.41145807 0.80050009 0.18954211 1.94881082
1.48782782 0.34504612 0.87686984 0.26591186 0.80326461 0.95323959
1.48782782 1.56696208 1.25871858 1.56419757 0.18954211 1.49059233
0.4950211 1.1087436 1.41145807 0.03680262 0.4950211 0.80050009
0.34228161 0.03956713 0.26591186 1.56419757 0.87686984 0.19230663
0.18954211 1.41145807 0.03680262 0.19230663 0.11593688 2.02241605
1.56696208 1.25871858 0.49778562 0.18954211 0.34228161 0.41865135
1.86967656 0.41865135 0.49778562 2.02241605 0.4950211 1.48782782
0.6477606 0.03956713 0.95323959 1.56419757 0.80326461 0.26867637
0.18954211 1.71693706 0.6477606 0.57139085 1.26148309 0.11317236

```
 0.42141587 0.41865135 1.33785284 0.57139085 0.34504612 0.6477606
 1.18234883 0.42141587 2.25428981 1.71693706 0.11317236 0.80050009
 0.6477606  0.03680262 0.57415536 1.79607132 0.26591186 1.1087436
 0.49778562 1.56696208 0.11593688 1.26148309 0.42141587 0.80050009
 0.34228161 0.87686984 0.41865135 1.03237385 0.03680262 0.72413034
 0.9560041  0.19230663 0.34504612 0.19230663 0.41865135 1.10597908
 0.57415536 1.56696208 2.25428981 0.95323959 0.03956713 0.41865135
 0.34228161 0.03956713 0.34228161 1.49059233 1.02960933 0.11317236
 0.72413034 0.4950211  0.41865135 0.9560041  1.10597908 0.11593688
 0.18954211 0.49778562 0.87963435 1.56696208 0.72413034 1.26148309
 1.79607132 1.10597908 0.26591186 1.25871858 0.49778562 0.34228161
 2.32789504 0.42141587 0.34504612 1.02960933 1.18511334 0.57139085
 1.33508832 1.1087436  0.19230663 0.11317236 1.56419757 1.1087436
 1.71693706 0.11593688 0.57415536 1.1087436  0.18954211 0.42141587
 0.4950211  0.80050009 1.64333183 0.18954211 0.03680262 1.64333183
 0.6477606  0.72689486 1.02960933 0.87963435 0.19230663 1.48782782
 0.18954211 0.57415536 0.34228161 0.26867637 1.18511334 0.87686984
 0.11593688 0.87686984 0.9560041  0.9560041 ]
```

1.3.2 Identifying Threshold value using z-score

[32]:
```
from scipy import stats
import numpy as np

# Calculate z-scores for the 'age' column
z = np.abs(stats.zscore(df_diabetes['age']))

# Define the threshold using the 95th percentile
threshold = np.percentile(z, 95)

# Print the threshold value
print("Threshold value:", threshold)

# Identify outliers
outliers = np.where(z > threshold)

# Print the results
print("Z-scores:", z)
print("Outliers:", outliers)
```

```
Threshold value: 1.8724410718097853
Z-scores: [0.80050009 0.03956713 1.79330681 1.87244107 0.11317236 1.94881082
 0.9560041  1.33508832 0.87686984 1.49059233 2.02518057 0.57139085
 0.34228161 0.11317236 0.95323959 1.1087436  0.11593688 1.48782782
 0.80326461 0.57415536 1.03237385 1.79607132 1.79607132 0.95323959
 1.33785284 1.41422259 2.25428981 0.49778562 1.10597908 1.41145807
 1.26148309 0.49778562 0.72413034 0.6477606  0.34228161 1.02960933
 0.26591186 0.19230663 0.03956713 0.03956713 0.11317236 2.10155031
```

```
1.26148309 0.41865135 0.95323959 0.57139085 1.18511334 1.64333183
1.41145807 0.87963435 0.72413034 1.25871858 1.1087436  0.19230663
1.03237385 0.87963435 0.87963435 0.57415536 0.87686984 1.33508832
1.49059233 0.87963435 0.57415536 0.72689486 1.41145807 0.9560041
0.19230663 0.87686984 0.80050009 0.34228161 0.03956713 0.03956713
1.33508832 0.26591186 0.26591186 0.19230663 0.65052511 2.02518057
0.11317236 2.17792006 1.48782782 0.26591186 0.34504612 0.80326461
0.03680262 0.95323959 1.49059233 0.95323959 1.1087436  0.9560041
0.26591186 0.95323959 0.42141587 1.03237385 1.64333183 1.49059233
1.18234883 0.57415536 0.03680262 0.03956713 0.34228161 0.34228161
1.94881082 1.25871858 0.57415536 0.4950211  2.02518057 0.57139085
0.41865135 0.80050009 0.87686984 0.41865135 1.79607132 0.41865135
0.4950211  0.65052511 1.02960933 1.25871858 1.18511334 0.34228161
1.03237385 1.33508832 1.02960933 0.11317236 0.11593688 0.11593688
1.87244107 0.72413034 1.1087436  0.18954211 1.33785284 2.02518057
0.34228161 0.87963435 1.56696208 0.11593688 1.94881082 0.11317236
0.72413034 0.4950211  0.87686984 0.57415536 0.87686984 0.65052511
0.6477606  0.87963435 0.65052511 1.18511334 1.26148309 1.03237385
0.4950211  0.03680262 0.72689486 0.87686984 1.41145807 0.57415536
0.34504612 0.03956713 0.26867637 0.11593688 0.19230663 0.9560041
1.1087436  0.34228161 0.95323959 0.87963435 1.18511334 1.48782782
0.03680262 0.03956713 0.4950211  0.42141587 0.87686984 1.33785284
0.34228161 1.41145807 0.95323959 1.02960933 0.87686984 0.49778562
0.80326461 1.02960933 0.95323959 0.95323959 0.34228161 1.56696208
1.71970158 1.41422259 0.11317236 0.03956713 0.18954211 0.11593688
1.18234883 0.18954211 1.41422259 0.57139085 0.49778562 1.02960933
1.1087436  0.87686984 1.18234883 0.72689486 1.71693706 0.03956713
2.32789504 0.65052511 0.03680262 0.18954211 0.6477606  0.80050009
0.18954211 1.9460463  1.41145807 0.03680262 0.6477606  0.57139085
0.26591186 1.56419757 0.87963435 1.87244107 0.4950211  0.9560041
0.49778562 2.10155031 0.57415536 0.6477606  2.17792006 1.41145807
1.1087436  0.57415536 0.80326461 0.18954211 0.26591186 1.41145807
0.95323959 1.41145807 0.57139085 1.18234883 0.72413034 0.4950211
1.02960933 0.6477606  2.17792006 0.34228161 1.26148309 0.57415536
0.87686984 1.71970158 0.87963435 0.26867637 1.41145807 1.1087436
0.11317236 1.71693706 0.6477606  0.03680262 1.03237385 0.57415536
1.64056731 0.26591186 0.87686984 1.02960933 0.34504612 1.56696208
0.72413034 0.72689486 1.1087436  1.25871858 1.33508832 0.18954211
0.11317236 0.80050009 0.26591186 1.56419757 0.34228161 0.11593688
0.26591186 0.72689486 1.41145807 0.80050009 0.18954211 1.94881082
1.48782782 0.34504612 0.87686984 0.26591186 0.80326461 0.95323959
1.48782782 1.56696208 1.25871858 1.56419757 0.18954211 1.49059233
0.4950211  1.1087436  1.41145807 0.03680262 0.4950211  0.80050009
0.34228161 0.03956713 0.26591186 1.56419757 0.87686984 0.19230663
0.18954211 1.41145807 0.03680262 0.19230663 0.11593688 2.02241605
1.56696208 1.25871858 0.49778562 0.18954211 0.34228161 0.41865135
1.86967656 0.41865135 0.49778562 2.02241605 0.4950211  1.48782782
0.6477606  0.03956713 0.95323959 1.56419757 0.80326461 0.26867637
```

```
 0.18954211 1.71693706 0.6477606  0.57139085 1.26148309 0.11317236
 0.42141587 0.41865135 1.33785284 0.57139085 0.34504612 0.6477606
 1.18234883 0.42141587 2.25428981 1.71693706 0.11317236 0.80050009
 0.6477606  0.03680262 0.57415536 1.79607132 0.26591186 1.1087436
 0.49778562 1.56696208 0.11593688 1.26148309 0.42141587 0.80050009
 0.34228161 0.87686984 0.41865135 1.03237385 0.03680262 0.72413034
 0.9560041  0.19230663 0.34504612 0.19230663 0.41865135 1.10597908
 0.57415536 1.56696208 2.25428981 0.95323959 0.03956713 0.41865135
 0.34228161 0.03956713 0.34228161 1.49059233 1.02960933 0.11317236
 0.72413034 0.4950211  0.41865135 0.9560041  1.10597908 0.11593688
 0.18954211 0.49778562 0.87963435 1.56696208 0.72413034 1.26148309
 1.79607132 1.10597908 0.26591186 1.25871858 0.49778562 0.34228161
 2.32789504 0.42141587 0.34504612 1.02960933 1.18511334 0.57139085
 1.33508832 1.1087436  0.19230663 0.11317236 1.56419757 1.1087436
 1.71693706 0.11593688 0.57415536 1.1087436  0.18954211 0.42141587
 0.4950211  0.80050009 1.64333183 0.18954211 0.03680262 1.64333183
 0.6477606  0.72689486 1.02960933 0.87963435 0.19230663 1.48782782
 0.18954211 0.57415536 0.34228161 0.26867637 1.18511334 0.87686984
 0.11593688 0.87686984 0.9560041  0.9560041 ]
Outliers: (array([  5,  10,  26,  41,  77,  79, 102, 106, 131, 136, 204, 211,
223,
       226, 242, 281, 311, 321, 344, 374, 402]),)
```

1.3.3 Outlier Removal

[28]:
```python
import numpy as np

def removal_outliers_zscore(df, column, z_column, threshold):
    # Calculate outlier indices based on z-score threshold for the specified␣
 ↪column
    outlier_indices = np.where(z_column > threshold)[0]
    removed_outliers = df.drop(outlier_indices)

    # Print DataFrame shapes
    print(f"Original DataFrame Shape: {df.shape}")
    print(f"DataFrame Shape after Removing Outliers: {removed_outliers.shape}")

    return removed_outliers

# Define the z-score threshold
threshold_z = 2

# Calculate z-scores for the 'age' column
z = np.abs((df_diabetes['age'] - df_diabetes['age'].mean()) /␣
 ↪df_diabetes['age'].std())

# Apply the function
```

```
no_outliers = removal_outliers_zscore(df_diabetes, 'age', z, threshold_z)
```

```
Original DataFrame Shape: (442, 10)
DataFrame Shape after Removing Outliers: (426, 10)
```

1.4 Using IQR (Inter Quartile Range)

IQR = Quartile3 – Quartile1

Syntax: numpy.percentile(arr, n, axis=None, out=None) Parameters: arr :input array. n : percentile value.

[34]:
```
Q1 = np.percentile(df_diabetes['bmi'], 25, method='midpoint')
Q3 = np.percentile(df_diabetes['bmi'], 75, method='midpoint')
IQR = Q3 - Q1

print(IQR)
```

```
0.06520763046978838
```

To define the outlier base value is defined above and below dataset's normal range namely Upper and Lower bounds, define the upper and the lower bound (1.5*IQR value is considered):

upper = Q3 +1.5IQR *lower* = *Q1* – *1.5*IQR

In the above formula as according to statistics, the 0.5 scale-up of IQR (new_IQR = IQR + 0.5*IQR) is taken, to consider all the data between 2.7 standard deviations in the Gaussian Distribution.

[37]:
```
upper = Q3 + 1.5 * IQR
upper_array = np.array(df_diabetes['bmi'] >= upper)
print("Upper Bound:", upper)
print(upper_array.sum())

lower = Q1 - 1.5 * IQR
lower_array = np.array(df_diabetes['bmi'] <= lower)
print("Lower Bound:", lower)
print(lower_array.sum())
```

```
Upper Bound: 0.12879000811776306
3
Lower Bound: -0.13204051376139045
0
```

1.5 Outlier removal in Dataset using IQR

[5]:
```
import sklearn
from sklearn.datasets import load_diabetes
import pandas as pd
import numpy as np
```

```
diabetes = load_diabetes()

column_name = diabetes.feature_names
df_diabetes = pd.DataFrame(diabetes.data)
df_diabetes.columns = column_name
df_diabetes.head()
print("Old Shape:", df_diabetes.shape)

Q1 = df_diabetes['bmi'].quantile(0.25)
Q3 = df_diabetes['bmi'].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR

upper_array = np.where(df_diabetes['bmi'] >= upper)[0]
lower_array = np.where(df_diabetes['bmi'] <= lower)[0]

df_diabetes.drop(index=upper_array, inplace=True)
df_diabetes.drop(index=lower_array, inplace=True)

print("New Shape:", df_diabetes.shape)
```

```
Old Shape: (442, 10)
New Shape: (439, 10)
```

[ ]: