HandlingOutliers

October 11, 2024

0.1 5 Number Summary

1. Minimum Value 2. Q1 - 25 Percentile 3. Median 4. Q3 - 75 Percentile 5. Maximum

0.1.1 1. np.percentile()

Definition: This function computes the "percentile" of a dataset. A percentile is the value below which a given percentage of data points fall. For example, the 50th percentile (median) means that 50% of the data points are below that value.

[2]:
```
# np.percentile(a, q, axis=None)
```

a: The input array or data (the dataset). q: The percentile to compute (a value between 0 and 100). It can be a single number (e.g., 50) or an array of percentiles (e.g., [25, 50, 75]). axis: Specifies the axis along which the percentiles are computed. If None, the entire array is flattened before computation.

0.2 2. np.quantile()

Definition: This function computes the "quantile" of a dataset. A quantile is similar to a percentile but is expressed as a fraction or decimal (between 0 and 1) instead of a percentage. For instance, the 0.5 quantile is equivalent to the 50th percentile, and the 0.25 quantile is equivalent to the 25th percentile.

[7]:
```
# np.quantile(a, q, axis=None)
```

a: The input array or data (the dataset). q: The quantile(s) to compute (a value between 0 and 1). It can be a single number (e.g., 0.5) or an array of quantiles (e.g., [0.25, 0.5, 0.75]). axis: Specifies the axis along which the quantiles are computed. If None, the entire array is flattened before computation

[12]:
```
import numpy as np

marks = [45, 35, 56, 75, 89, 54, 32, 89, 90, 87, 67, 54, 45, 98, 99, 67, 74,␣
 ↪1000, 1100]
fiveNumber = np.percentile(marks, [0, 25, 50, 75, 100])
print(fiveNumber)

# min, q1, median, q3, max = fiveNumber
# or
```

```
min, q1, q2, q3, max = np.quantile(marks, [0, 0.25, 0.5, 0.75, 1])
```

```
[  32.    54.    74.    89.5 1100. ]
```

[26]:
```
## lower fence and higher fence

iqr = q3 - q1
print(iqr)

lowerFence = q1 - (1.5 * iqr)
higherFence = q3 + (1.5 * iqr)

print('\nLower Fence :', lowerFence, '\nHigher Fence: ', higherFence)

outliers = []
for i in marks :
    if i > higherFence or i < lowerFence :
        outliers.append(i)

print('\nOutliers: ', outliers)
```

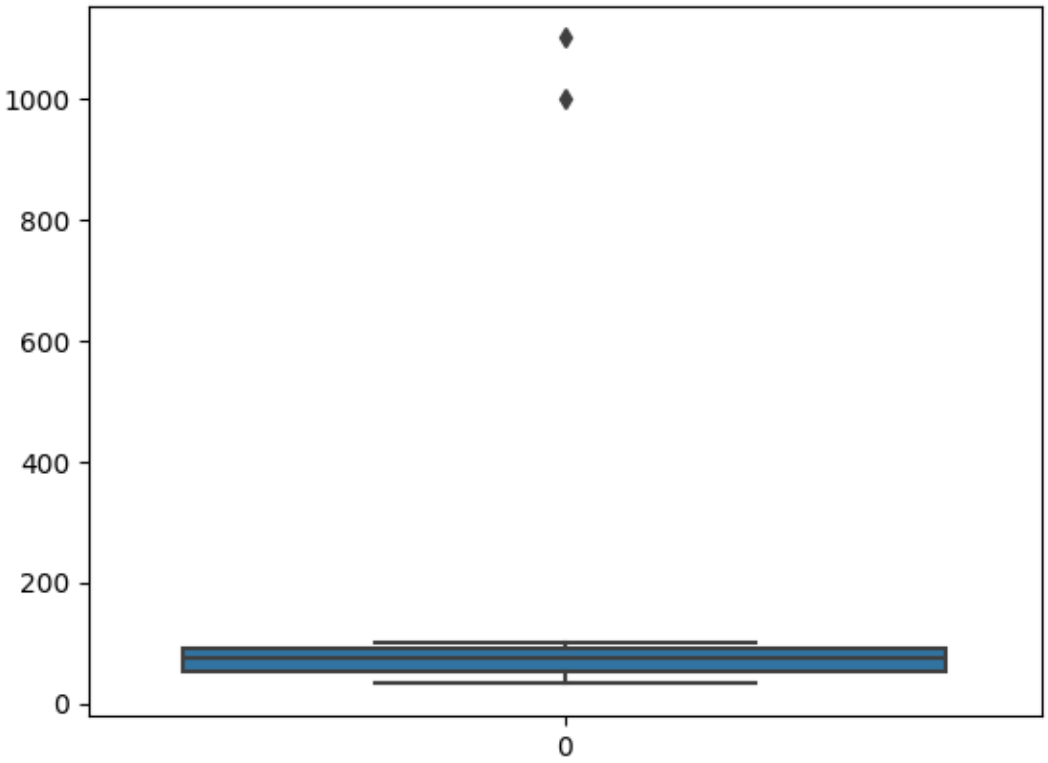
```
35.5

Lower Fence : 0.75
Higher Fence:  142.75

Outliers:  [1000, 1100]
```

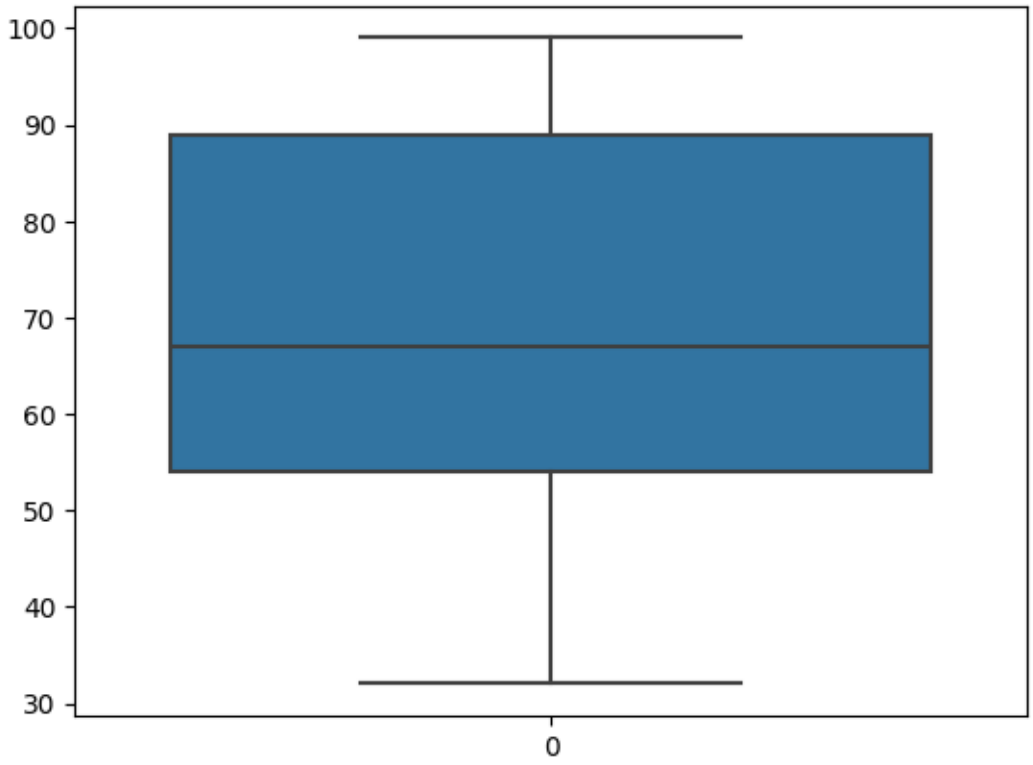
[28]:
```
import seaborn as sns
sns.boxplot(marks)
```

[28]: <Axes: >



[30]:
```
marksWithoutOutliers = [45, 35, 56, 75, 89, 54, 32, 89, 90, 87, 67, 54, 45, 98,␣
↪99, 67, 74]
sns.boxplot(marksWithoutOutliers)
```

[30]: <Axes: >


100
90
80
70
60
50
40
30
0

[ ]: