Outliers Detection Using IQR, Z-score

What are Inliers and Outliers?

Outliers are values that seem excessively different from most of the rest of the values in the given dataset.

Outliers could normally exist due to new inventions **(true outliers)**, development of new patterns/phenomena, experimental errors, rarely occurring incidents, anomalies, incorrectly fed data due to typographical mistakes, failure of data recording systems/components, etc. However, all outliers are not bad; some reveal new information.

Inliers are all the data points that are part of the distribution except outliers.

O*utliers are* values in the data set that are very large or small compared to the majority of the values in the data set.

For example, 367 is an outlier in the following data set (3,6,1,4,5,3,2,1, 367).

Why is it important to learn?
The machine learning model performance depends on how clean your data is. Even the best machine learning algorithm will not perform well if outliers are present in the data. Outliers affect the training process of a machine learning model, and results can be inaccurate.

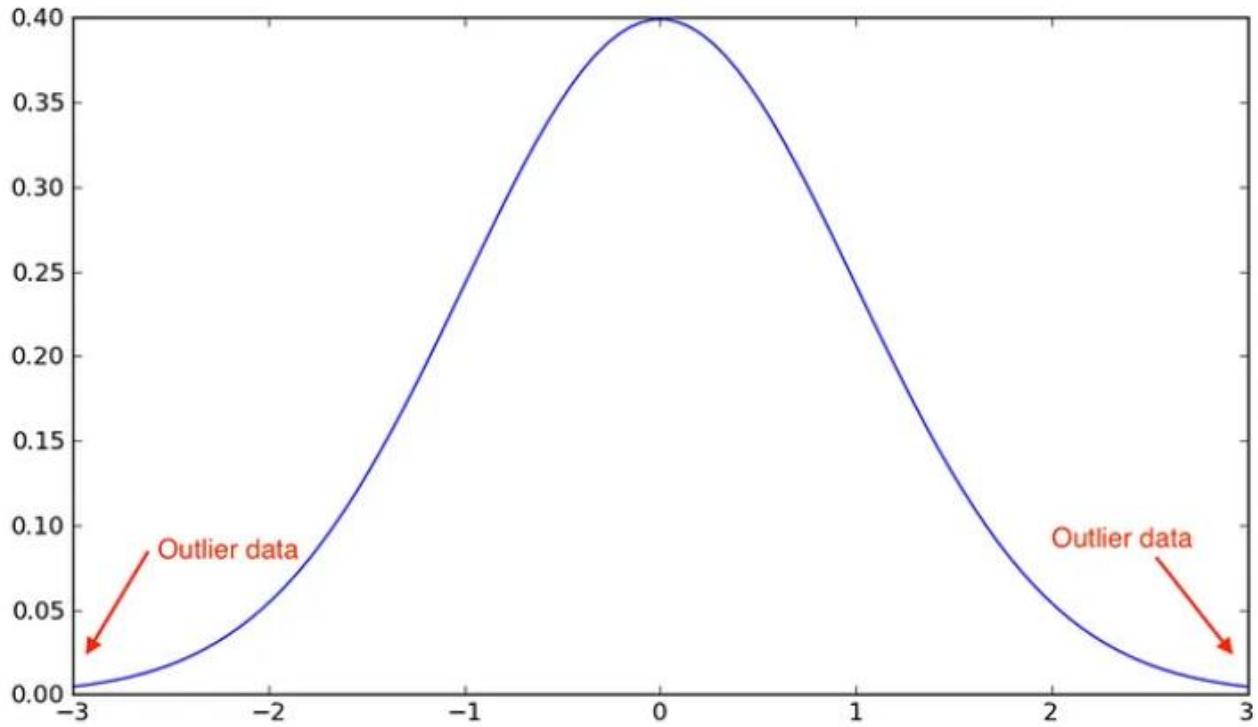
Detecting outliers with Visualization

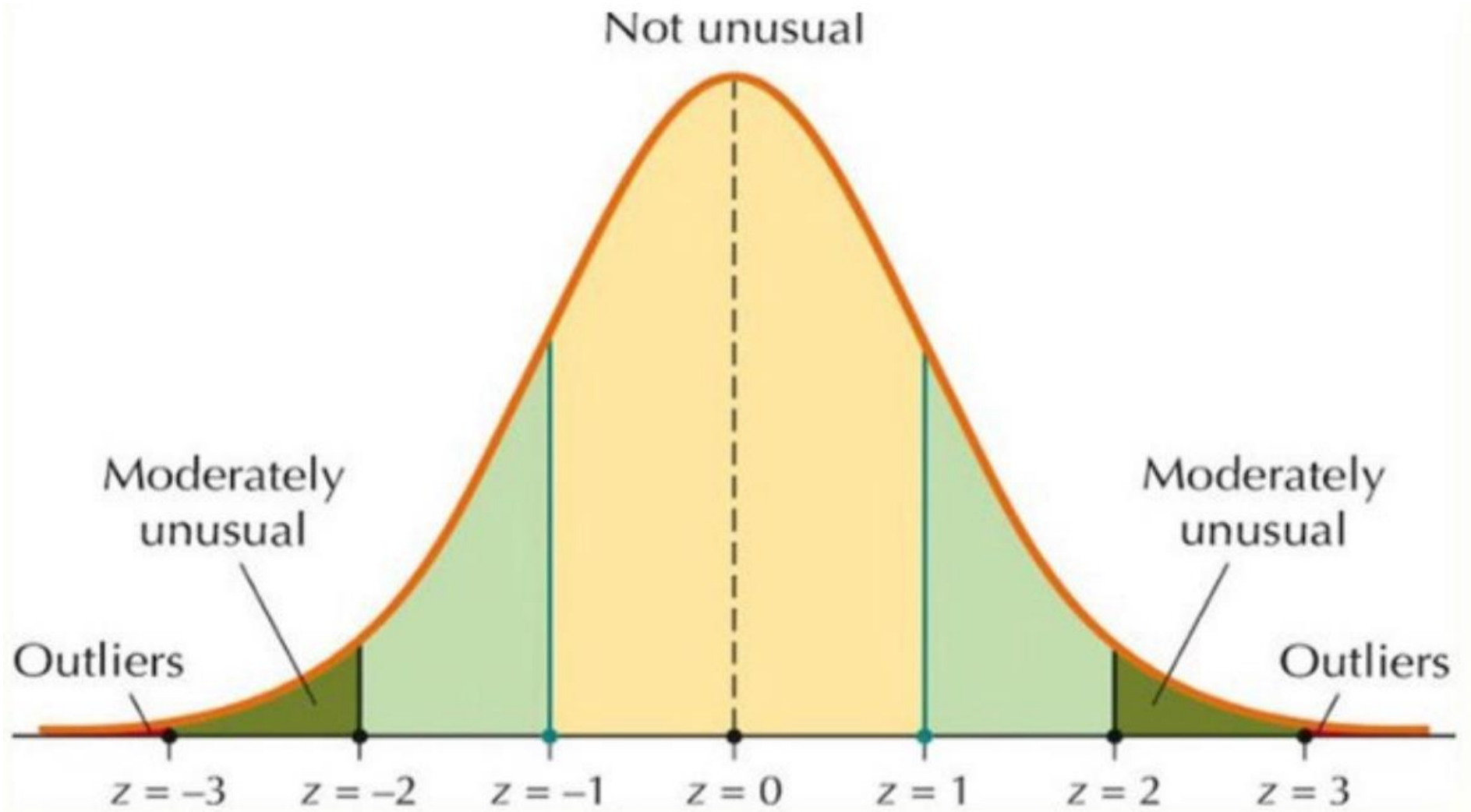
The easiest way to see the outliers is by using a box plot. A box has two main components a '**box**' and two '**whiskers**'. The whiskers represent the data outside the middle 50% of the data. Data values that lie outside the whiskers are considered outliers.

Detecting outliers with Z score

In statistics, Z-score is used to find outliers in a data set. It is a statistical measurement that helps us understand how close or far we are from the data set's arithmetic mean. For people who don't have a statistics background, all you have to remember is " any value that greater than **+3** and less than **-3** standard deviation is an outlier value."

The best way to visualize this concept is by looking at a normal distribution. The x-axis in the figure below represents the standard deviation (a measure of variability in statistics). Observe the data outside the standard deviation of +3 and -3 is considered as outliers.




Not unusual
Moderately unusual
Moderately unusual
Outliers
Outliers
z = –3
z = –2
z = –1
z = 0
z = 1
z = 2
z = 3

Limitations of Z-Score

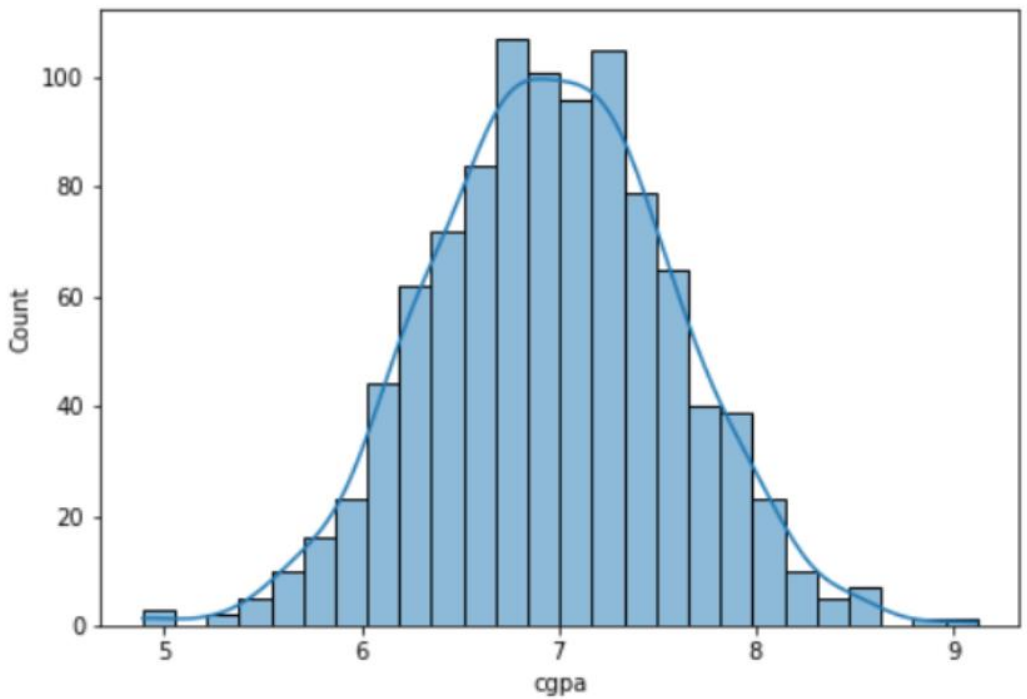
Though Z-Score is a highly efficient way of detecting and removing outliers, we cannot use it with every data type. When we said that, we mean that it only works with the data which is completely or close to **normally distributed,** which in turn stimulates that this method is not for **skewed** data, either **left skew** or **right skew. For** the other data, we have something known as **Inter quartile range (IQR)** method,

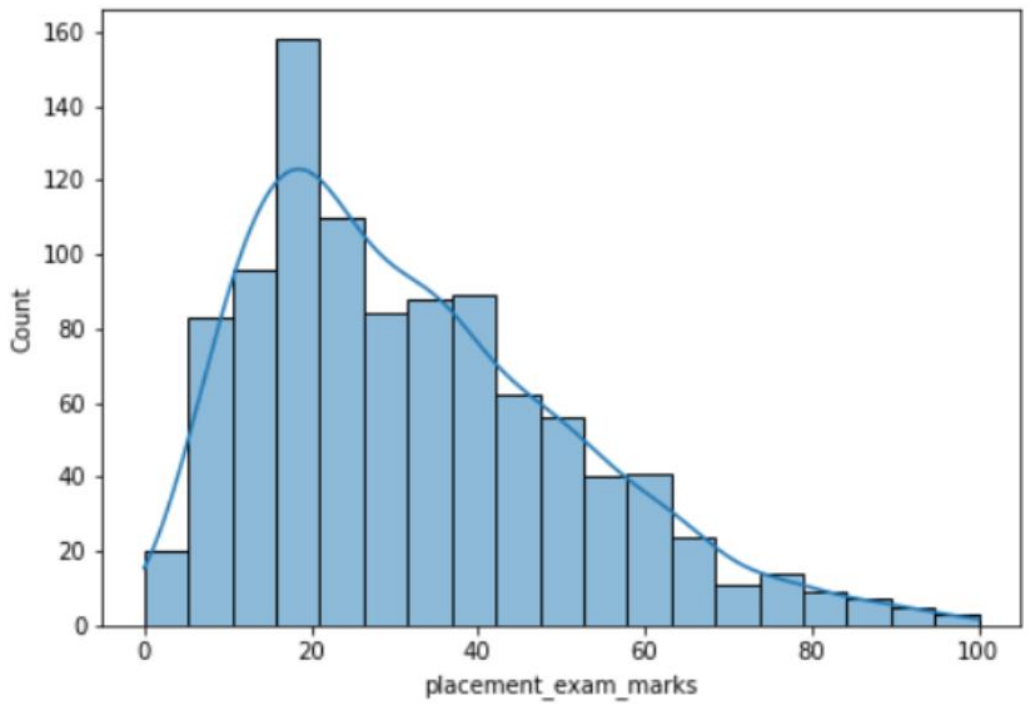
	cgpa	placement_exam_marks	placed
0	7.19	26.0	1
1	7.46	38.0	1
2	7.54	40.0	1
3	6.42	8.0	1
4	7.23	17.0	0

```
plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.histplot(df_org['cgpa'], kde=True)

plt.subplot(1,2,2)
sns.histplot(df_org['placement_exam_marks'], kde=True)

plt.show()
```





From the graph, we can see that CGPA is almost the right fit for the normal distribution, or we can say it is forming the correct **bell curve**. Other one is slightly **skewed** towards the right.

Hence, take CGPA for our further analysis.

```
df_org['placement_exam_marks'].skew()
```

Output:

```
0.8356419499466834
```

```
df_org['cgpa'].skew()
```

Output:

```
-0.01452993892931491 8
```

Here,we have got the difference between a normal distribution and skewed distribution from the above graph, but sometimes, the graph might not give a clear understanding that we have the skew function from pandas which will give a higher positive value if the distribution seems to be skewed (placement_exam_marks) otherwise it will return the quite lower value even in negative if it is not skewed at all (CGPA).

```
print("Mean value of cgpa",df_org['cgpa'].mean())
print("Standard deviation of cgpa",df_org['cgpa'].std())
print("Minimum value of cgpa",df_org['cgpa'].min())
print("Maximum value of cgpa",df_org['cgpa'].max())
```

Output:

```
Mean value of cgpa 6.96124000000001
Standard deviation of cgpa 0.6158978751323894
Minimum value of cgpa 4.89
Maximum value of cgpa 9.12
```

Here are some different statistical measures for CGPA, which are printed out to compare the original values (from original data) to when the data will be free of any outlier.

Approach for Outliers

Approach to deal with bad data using Z-Score:

- The very first step will be setting the upper and lower limit.
- This range stimulates that every data point will be regarded as an outlier out of this range.

Formulae for both upper and lower limits.

Upper: Mean + 3 * standard deviation.

Lower: MEan – 3 * standard deviation.

```
print("Upper limit",df_org['cgpa'].mean() + 3*df_org['cgpa'].std())
print("Lower limit",df_org['cgpa'].mean() - 3*df_org['cgpa'].std())
```

Output:
Upper limit 8.808933625397177
Lower limit 5.113546374602842

The highest value is 8.80 while the lowest value is 5.11.
Any value out of this range is the bad data point

The second step is to detect how many outliers are there in the dataset based on the upper and lower limit that we set up just

```
df_org[(df_org['cgpa'] > 8.80) | (df_org['cgpa'] < 5.11)]
```

Output:

	cgpa	placement_exam_marks	placed
485	4.92	44.0	1
995	8.87	44.0	1
996	9.12	65.0	1
997	4.89	34.0	0
999	4.90	10.0	1

The third and the last step, where we will finally remove the detected outliers in step 2 using two techniques; we can either go for Trimming or Capping.

Trimming for Outliers

The first technique for dealing with outliers is trimming, and this is regardless of what kind of data distribution are working with, trimming is an applicable and proven technique for most data types.
We pluck out all the outliers using the filter condition in this technique.

```
new_df_org = df_org[(df_org['cgpa']  5.11)]
new_df_org
```

We applied that condition where all the data should be in the range of our upper and lower limit. As a result, instead of 1000 rows, it returned 995 rows which indicate that we have successfully removed 5 outliers.

Output:

	cgpa	placement_exam_marks	placed
0	7.19	26.0	1
1	7.46	38.0	1
2	7.54	40.0	1
3	6.42	8.0	1
4	7.23	17.0	0
...	...	...	...
991	7.04	57.0	0
992	6.26	12.0	0
993	6.73	21.0	1
994	6.48	63.0	0
998	8.62	46.0	1

Capping

- Capping is another technique for dealing with bad data points.
- It is useful when we have many outliers, and removing a good amount of data from the dataset is not good.
- In that case, capping comes into the picture as it won’t remove them.
- Instead, it brings back those data points within the range we specified according to our Z-Score value.

```
upper_limit = df_org['cgpa'].mean() + 3*df_org['cgpa'].std()
lower_limit = df_org['cgpa'].mean() - 3*df_org['cgpa'].std()
```

The first thing is setting the upper and lower bound. Then comes the part of np.
Where the function that will help to implement the logic behind capping, the basic syntax is as, np.where(condition, True, False) i.e., if the condition is true, then that data point will be getting the upper limit value (within range) if not, it will go to check the lower limit, and if that's true, then it will give that data point lower limit value (within range).

```
df_org['cgpa'] = np.where(
    df_org['cgpa']>upper_limit,
    upper_limit,
    np.where(
        df_org['cgpa']<lower_limit,
        lower_limit,
        df_org['cgpa']
    )
)
```

```
df_org.shape
```

Output:

```
(1000, 3)
```

On capping, no data was lost, and we still have 1000 rows.

df_org['cgpa'].describe()

Output:

```
count    1000.000000
mean        6.961499
std         0.612688
min         5.113546
25%         6.550000
50%         6.960000
75%         7.370000
max         8.808934
Name: cgpa, dtype: float64
```

if we compare the minimum and maximum values before outlier removal and after, we can see that **the minimum value is increased** and **the maximum value is decreased**.

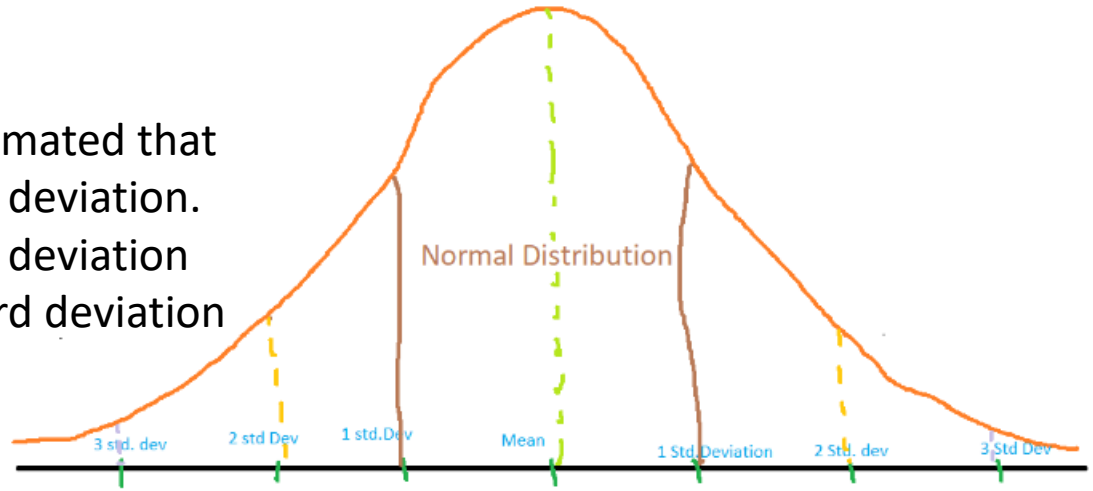
Treating/Cleaning Outlier

So far, we have detected the outliers, and we will overwrite them with a higher outlier.
In the **Age** column we will use the Z score **upper_limit** of 72 to replace the outliers.

Z score is an important concept in statistics. Z score is also called standard score. This score helps to understand if a data value is greater or smaller than mean and how far away it is from the mean. More specifically, Z score tells how many standard deviations away a data point is from the mean.

Z score = (x -mean) / std. deviation

A normal distribution is shown below and it is estimated that
68% of the data points lie between +/- 1 standard deviation.
95% of the data points lie between +/- 2 standard deviation
99.7% of the data points lie between +/- 3 standard deviation



Detecting outliers with Interquartile Range (IQR)

In statistics, the interquartile range is a measure of statistical dispersion, which describes the variability in data.

To find an **IQR**, we calculate **Q1** and **Q3** values then find the **IQR** by subtracting **Q3** from **Q1**. **Q3** is the 75th percentile, and **Q1** is the 25th percentile.

Use the outlier formula, 1.5 is the constant used to determine outliers.

Lower Outlier = Q1-(1.5 x IQR)
Upper Outlier =Q3-(1.5 x IQR)

A resulting value from these formulas that is greater than a higher outlier, and less than a lower outlier is considered as an outlier.

Outliers are highly useful in anomaly detection like fraud detection where the fraud transactions are very different from normal transactions.

What is Interquartile Range IQR?

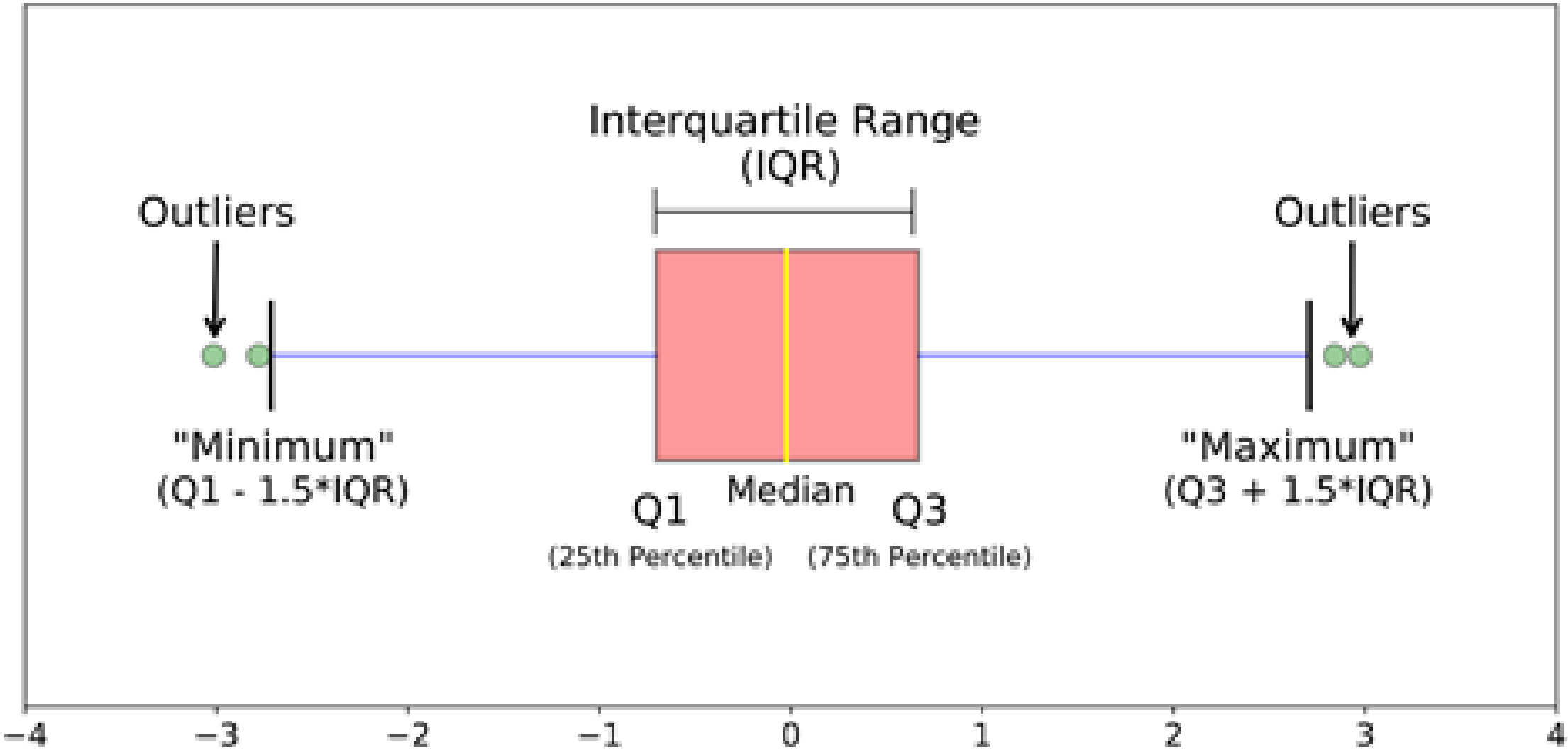
IQR is used to measure variability by dividing a data set into quartiles. The data is sorted in ascending order and split into 4 equal parts. Q1, Q2, Q3 called first, second and third quartiles are the values which separate the 4 equal parts.

Q1 represents the 25th percentile of the data.
Q2 represents the 50th percentile of the data.
Q3 represents the 75th percentile of the data.

If a dataset has 2n or 2n+1 data points, then
Q2 = median of the dataset.
Q1 = median of n smallest data points.
Q3 = median of n highest data points.

IQR is the range between the first and the third quartiles namely Q1 and Q3: IQR = Q3 – Q1. The data points which fall below Q1 – 1.5 IQR or above Q3 + 1.5 IQR are outliers.

Detecting Outliers using the Inter Quantile Range(IQR)



Criteria: data points that lie 1.5 times of IQR above Q3 and below Q1 are outliers.

Steps

- Sort the dataset in ascending order
- Calculate the 1st and 3rd quartiles(Q1, Q3)
- Compute IQR=Q3-Q1
- Compute lower bound = (Q1–1.5*IQR), upper bound = (Q3+1.5*IQR)
- Loop through the values of the dataset and check for those who fall below the lower bound and above the upper bound and mark them as outlier treatment in python

```
outliers = []
def detect_outliers_iqr(data):
    data = sorted(data)
    q1 = np.percentile(data, 25)
    q3 = np.percentile(data, 75)

    # print(q1, q3)
    IQR = q3-q1
    lwr_bound = q1-(1.5*IQR)
    upr_bound = q3+(1.5*IQR)

    # print(lwr_bound, upr_bound)
    for i in data:
        if (i<lwr_bound or i>upr_bound):
            outliers.append(i)
    return outliers

# Driver code
sample_outliers = detect_outliers_iqr(sample)
print("Outliers from IQR method: ", sample_outliers)
```

code outputs: Outliers from IQR method: [101]

How to Handle Outliers?

Methods of treating the outliers:

Step 1: Trimming/Remove the outliers

In this technique, we remove the outliers from the dataset. Although it is not a good practice to follow.

Python code to delete the outlier treatment and copy the rest of the elements to another array.

```
# Trimming
for i in sample_outliers:
    a = np.delete(sample, np.where(sample==i))
    print(a) # print(len(sample), len(a))
```

The outlier '101' is deleted and the rest of the data points are copied to another array 'a'.

Step 2: Quantile Based Flooring and Capping

In this technique, the outlier is capped at a certain value above the 90th percentile value or floored at a factor below the 10th percentile value.

Python code to delete the outlier and copy the rest of the elements to another array.

```
# Computing 10th, 90th percentiles and replacing the outlier treatment in python
tenth_percentile = np.percentile(sample, 10)
ninetieth_percentile = np.percentile(sample, 90)
# print(tenth_percentile, ninetieth_percentile)
b = np.where(sample<tenth_percentile, tenth_percentile, sample)
b = np.where(b>ninetieth_percentile, ninetieth_percentile, b)
# print("Sample:", sample)
print("New array:",b)
```

The above code outputs: New array: [15, 20.7, 18, 7.2, 13, 16, 11, 20.7, 7.2, 15, 10, 9]

The data points that are lesser than the 10th percentile are replaced with the 10th percentile value and the data points that are greater than the 90th percentile are replaced with 90th percentile value.

Step 3: Mean/Median Imputation

As the mean value is highly influenced by the outlier treatment, it is advised to replace the outliers with the median value.

Python Code:

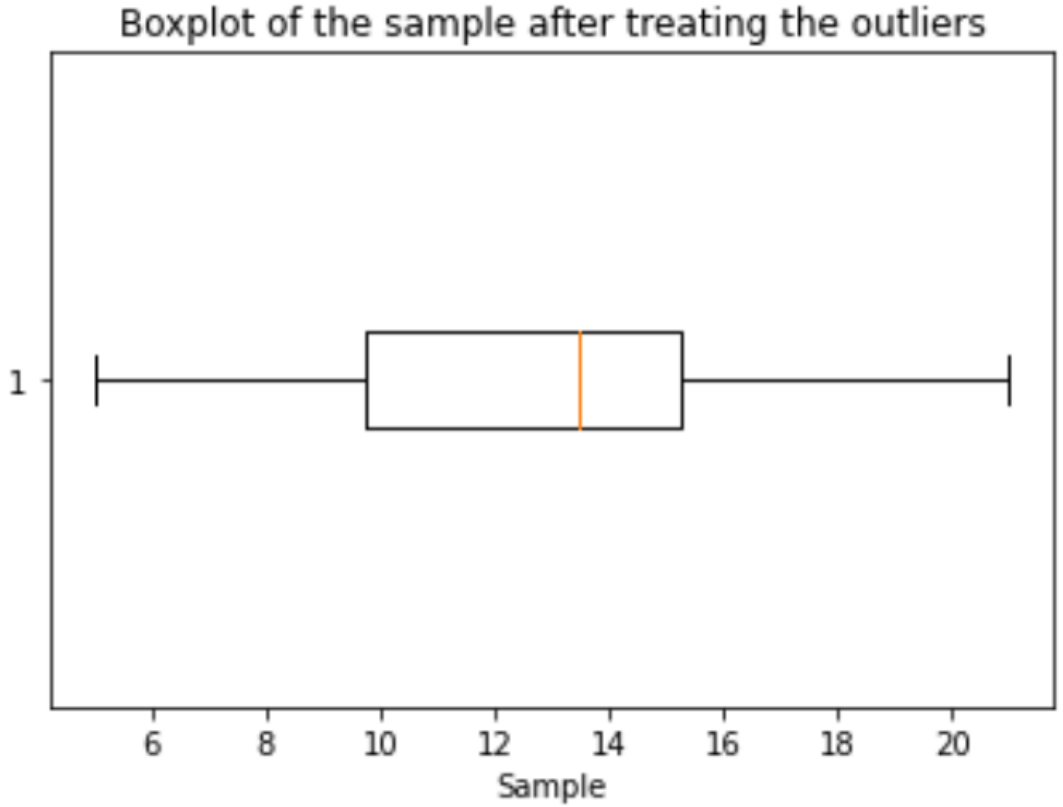
```
median = np.median(sample)

# Replace with median

for i in sample_outliers:
    c = np.where(sample==i, 14, sample)
    print("Sample: ", sample)
    print("New array: ",c)
    # print(x.dtype)
```

Step 5: Visualizing the Data after Treating the Outlier

```
plt.boxplot(c, vert=False)
plt.title("Boxplot of the sample after treating the outliers")
plt.xlabel("Sample")
```



Data after treating Outliner