**Outlier Treatment Using IQR and Box Plot**

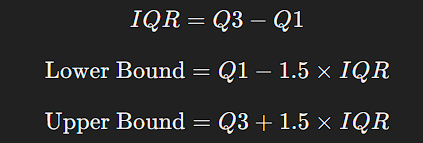
### **Outlier Treatment Using IQR and Box Plot**

Outliers are extreme values that differ significantly from the rest of the dataset. They can affect statistical analyses and machine learning models, so it’s important to identify and handle them properly.

### **Step 1: Identifying Outliers Using IQR**

The **Interquartile Range (IQR)** method is commonly used to detect outliers.

#### **Formula for Outlier Detection:**



* **Any value below the lower bound or above the upper bound is considered an outlier.**

#### **Example:**

Consider the dataset: **{10, 12, 15, 18, 22, 25, 30, 100}**

* Q1 = 13.5
* Q3 = 27.5
* IQR = 27.5 - 13.5 = 14
* Lower Bound = **13.5 - (1.5 × 14) = -7.5**
* Upper Bound = **27.5 + (1.5 × 14) = 48.5**
* **Outlier:** 100 (because it is greater than 48.5)

### **Step 2: Visualizing Outliers Using Box Plot**

Box plots visually represent outliers:

* Outliers appear as **individual points** beyond the whiskers.
* The **whiskers extend** to the smallest and largest values within the allowed range.

#### **Python Code to Generate a Box Plot and Detect Outliers:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

# Sample dataset

data = [10, 12, 15, 18, 22, 25, 30, 100]

# Convert to DataFrame

df = pd.DataFrame(data, columns=['Values'])

# Calculate Q1, Q3, and IQR

Q1 = df['Values'].quantile(0.25)

Q3 = df['Values'].quantile(0.75)

IQR = Q3 - Q1

# Define outlier bounds

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

# Detect outliers

outliers = df[(df['Values'] < lower\_bound) | (df['Values'] > upper\_bound)]

print("Outliers:\n", outliers)

# Box plot visualization

plt.boxplot(df['Values'])

plt.title("Box Plot for Outlier Detection")

plt.ylabel("Values")

plt.show()

### **Step 3: Treating Outliers**

Once identified, outliers can be handled in different ways:

**Removing Outliers (If Data is Large & Outliers Are Errors)**  
df\_filtered = df[(df['Values'] >= lower\_bound) & (df['Values'] <= upper\_bound)]

1. ✅ Best for **erroneous or incorrect data points**
2. **Capping Outliers (Winsorization)**
   * Replace extreme values with the upper/lower bound.

df['Values'] = np.where(df['Values'] > upper\_bound, upper\_bound, df['Values'])

df['Values'] = np.where(df['Values'] < lower\_bound, lower\_bound, df['Values'])

1. ✅ Best for **keeping data but reducing impact of extreme values**

**Transforming Data (Log or Square Root Transform)**  
df['Values'] = np.log1p(df['Values']) # Log transformation

1. ✅ Useful for **skewed distributions**

### **Conclusion**

* **IQR & Box Plots help in outlier detection.**
* **Outliers can be removed, capped, or transformed depending on the use case.**
* **Handling outliers improves statistical accuracy and model performance.**