## 1. Outliers/Abnormalities

In machine learning, **outliers are data points that deviate significantly from the general distribution of the dataset.** They are unusual observations that don't follow the expected pattern of the majority of your data.

Anomaly detection, or outlier detection, is the identification of observations, events or data points that deviate from what is usual, standard or expected, making them inconsistent with the rest of a data set.

Other names for outliers include **deviants**, **abnormalities**, **anomalies** (or **anomalous points**), and **aberrant observations**.

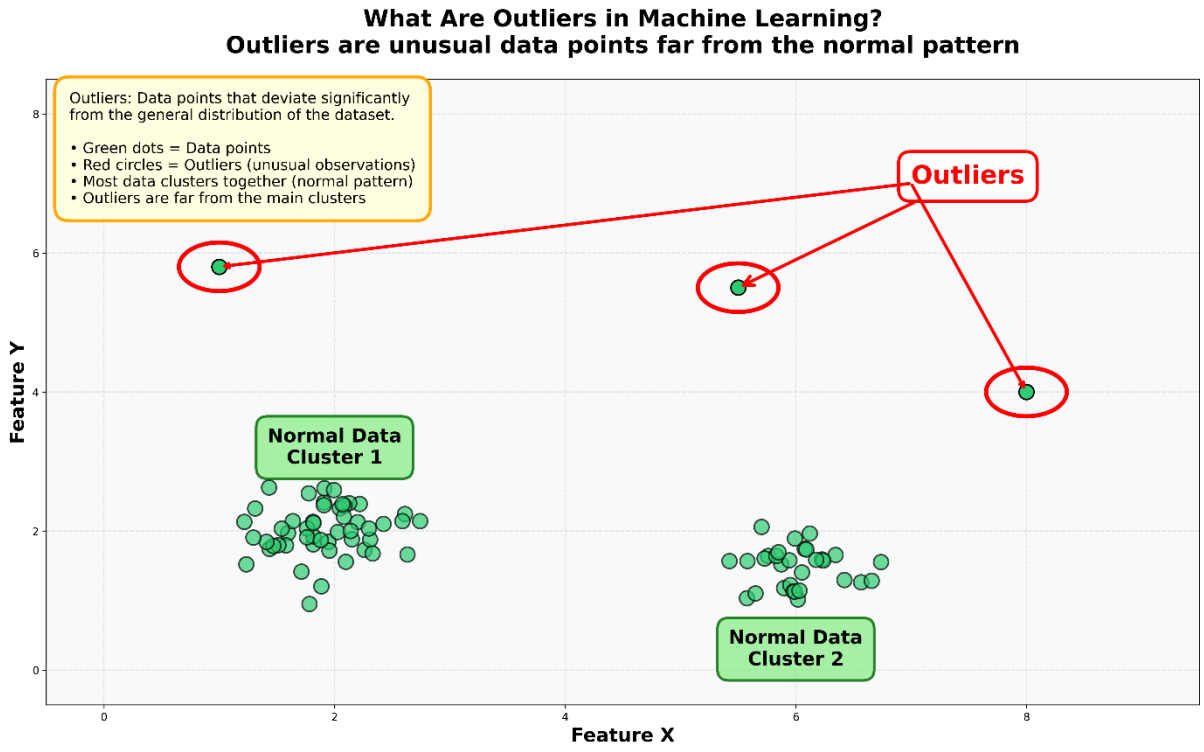



**Figure 1.1.** Visual Representation of Outliers in Machine Learning. Outliers (Red Circles) Are Data Points That Deviate Significantly from Normal Data Clusters (Green Dots).

- **Green dots** = Normal data points that follow the expected pattern
- **Cluster 1 & 2** = Most data naturally group together (normal distribution)
- **Red circles** = Outliers - data points that are far away from the main clusters
- **Red arrows** = Point to the unusual observations

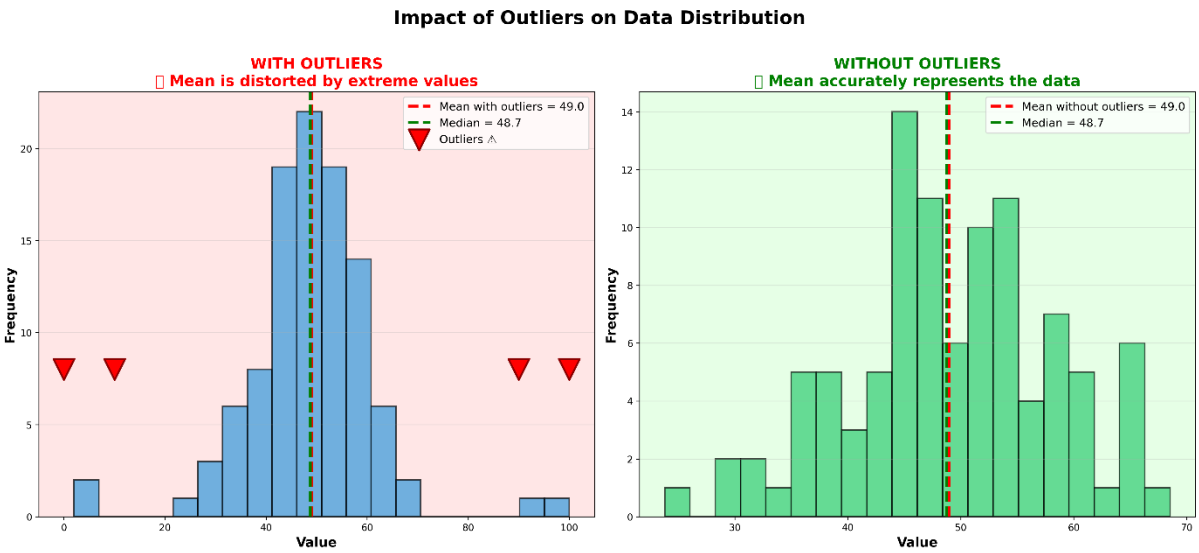



**Figure 1.2.** Impact Of Outliers on Data Distribution. Left: Outliers Distort Statistical Measures. Right: Clean Data After Outlier Removal.

**LEFT SIDE (With Outliers)**

- Shows data distribution including extreme outlier values (2, 5, 95, 100)
- Red triangles mark the outliers
- Mean is **distorted** and pulled away from the true center
- Red dashed line shows how outliers skew the average

**RIGHT SIDE (Without Outliers)**

- Shows the same data after removing outliers
- Mean and median are now close together
- Represents the data **accurately**
- Clean, reliable distribution

a. **Impact on ML models:** Can bias parameter estimation, increase variance and reduce model accuracy.

b. **Sources:** Data entry errors, measurement noise, genuine rare events.

c. **Handling methods:** Removal, transformation or robust algorithms less sensitive to outliers.

d. **Domain dependence:** What is considered an outlier in one domain (e.g., finance) may be normal in another.

Outliers are a type of data that can provide meaningful results, though they are mostly unmeaningful.

Outlier Detection | Outlier and Anomaly | Outliers Detection | Anomaly Detection

**Outliers** are data points that fall outside the expected or valid range for a specific category. In the context of age categorization, an outlier occurs when a person's age doesn't match their assigned age group.

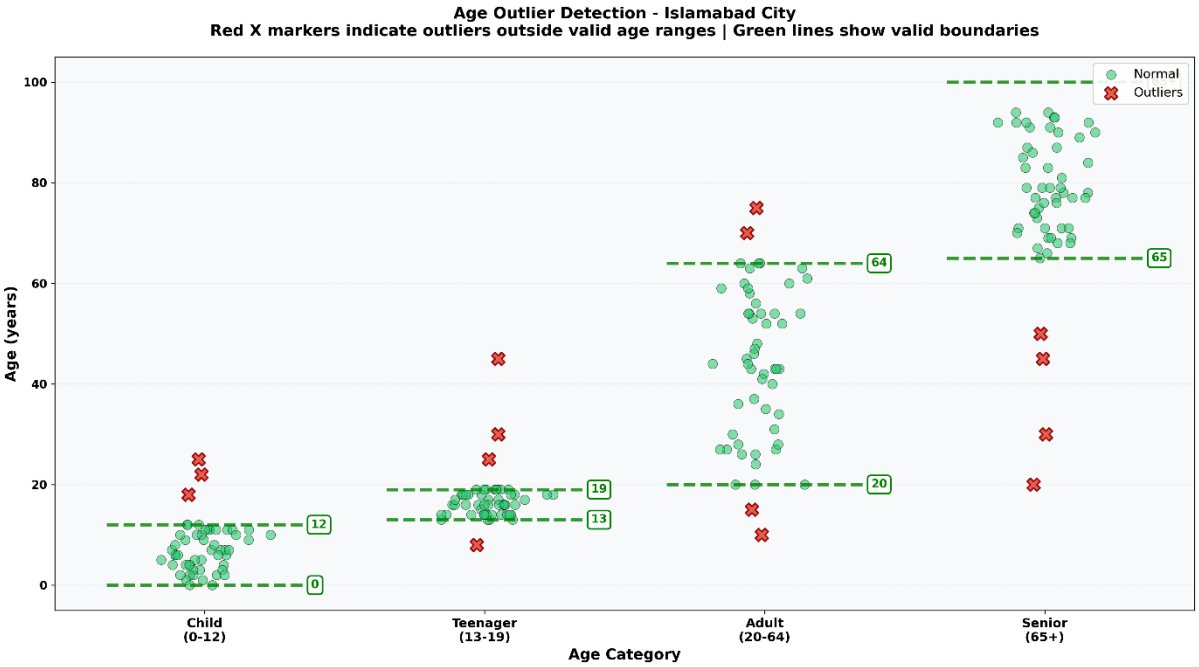



**Figure 1.3.** Age outlier detection showing normal data (green circles) and outliers (red X) across four age categories in Islamabad city.

This chart displays all four age categories on a single graph, making it easy to compare outlier patterns across categories.

- **X-axis:** Four age categories (Child, Teenager, Adult, Senior)
- **Y-axis:** Age values from 0 to 100 years (marked at intervals: 0, 20, 40, 60, 80, 100)
- **Green circles:** Normal/valid data points that fall within the correct age range
- **Red X markers:** Outliers that fall outside the valid range

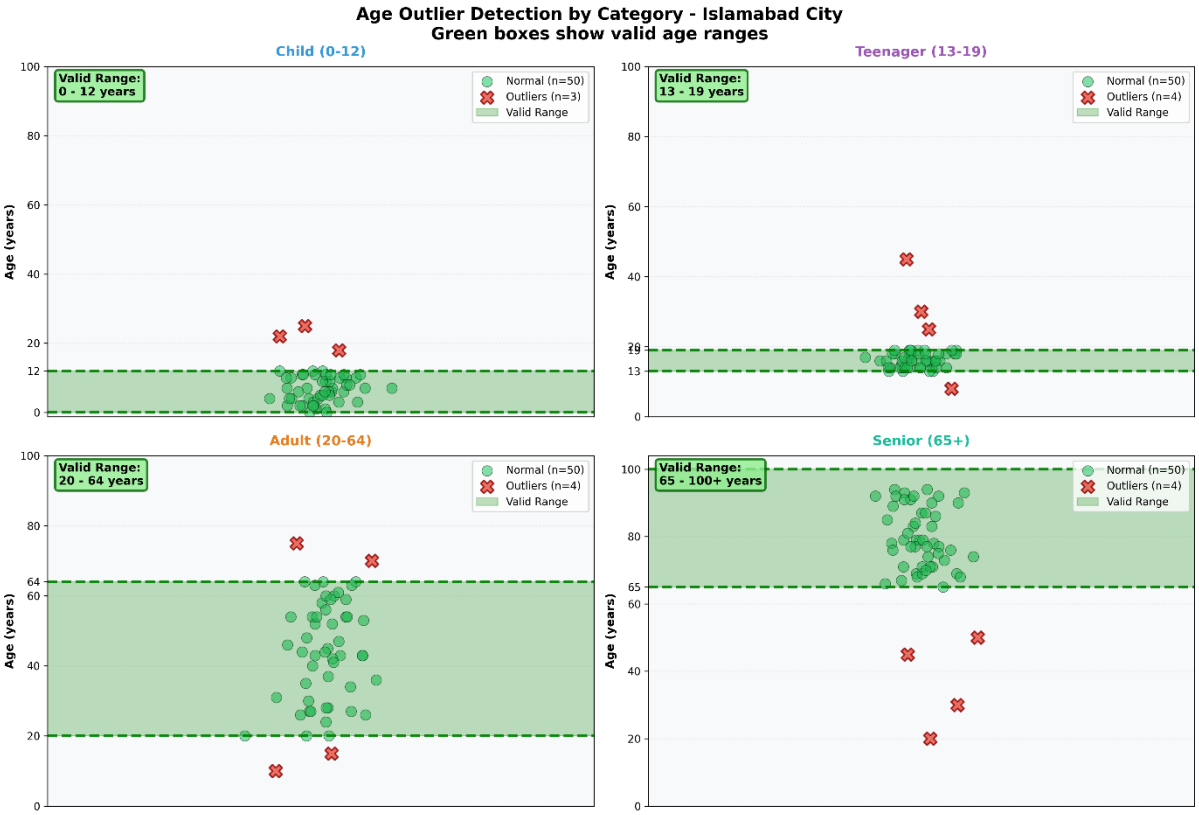



**Figure 1.4.** Category-wise age outlier breakdown with valid ranges (green areas) and outlier statistics for each age group.

This chart breaks down each category into separate panels for detailed analysis of outliers in each age group.

- **Four separate panels:** One for each age category
- **Green shaded area:** Highlights the valid age range for that category
- **Statistics shown:** Number of normal records vs. outliers for each category
- **Green box annotation:** Shows the exact valid range in years

## 1.1. Reasons That Can Cause Outliers

1. Data collection mistakes.
2. If the data is collected by a machine, the machine may have a problem or malfunction.
3. Typographical mistakes.
4. Issues with measurement or measuring instruments in the original data.
5. Data entry errors.
6. Sampling errors.
7. Natural variation in the data.
8. Incorrect data processing or coding errors.

9. Experimental errors.
10. Fraudulent or intentionally altered data.

## 1.2. Why Identifies Outliers?

1. **Machine learning models may not perform properly –** Outliers can reduce model accuracy and distort predictions.
2. **They pull the center (mean/median) toward themselves –** Extreme values can shift measures of central tendency.
3. **They create skewness in the data –** Outliers can make the distribution uneven or biased.
4. **They lead to incorrect results –** Statistical analysis may give misleading conclusions.

## 1.3. Types of Outliers

### 1.3.1. Univariable Outlier

A Univariate outlier is an extreme value that **relates to just one variable.**

These are outliers that **occur in only one variable.**

**Example:** If your data has only an age variable and someone's age is 150 years which is very unlikely, then this would be a univariate outlier.

**Table 1.3.1.1.** Univariate Outlier – Data Table

**Univariate Outlier - Data Table**
**(Only the extreme value 150 is highlighted red)**

| No. | Name | Age | Gender | Outlier? |
|---|---|---|---|---|
| 1 | Alice | 22 | Female | No |
| 2 | Brian | 25 | Male | No |
| 3 | Clara | **150** | Female | **Yes** |
| 4 | David | 24 | Male | No |
| 5 | Emma | 23 | Female | No |

*Note: Only the AGE value "150" is red — the outlier is in the variable (Age), not the person (Clara)*

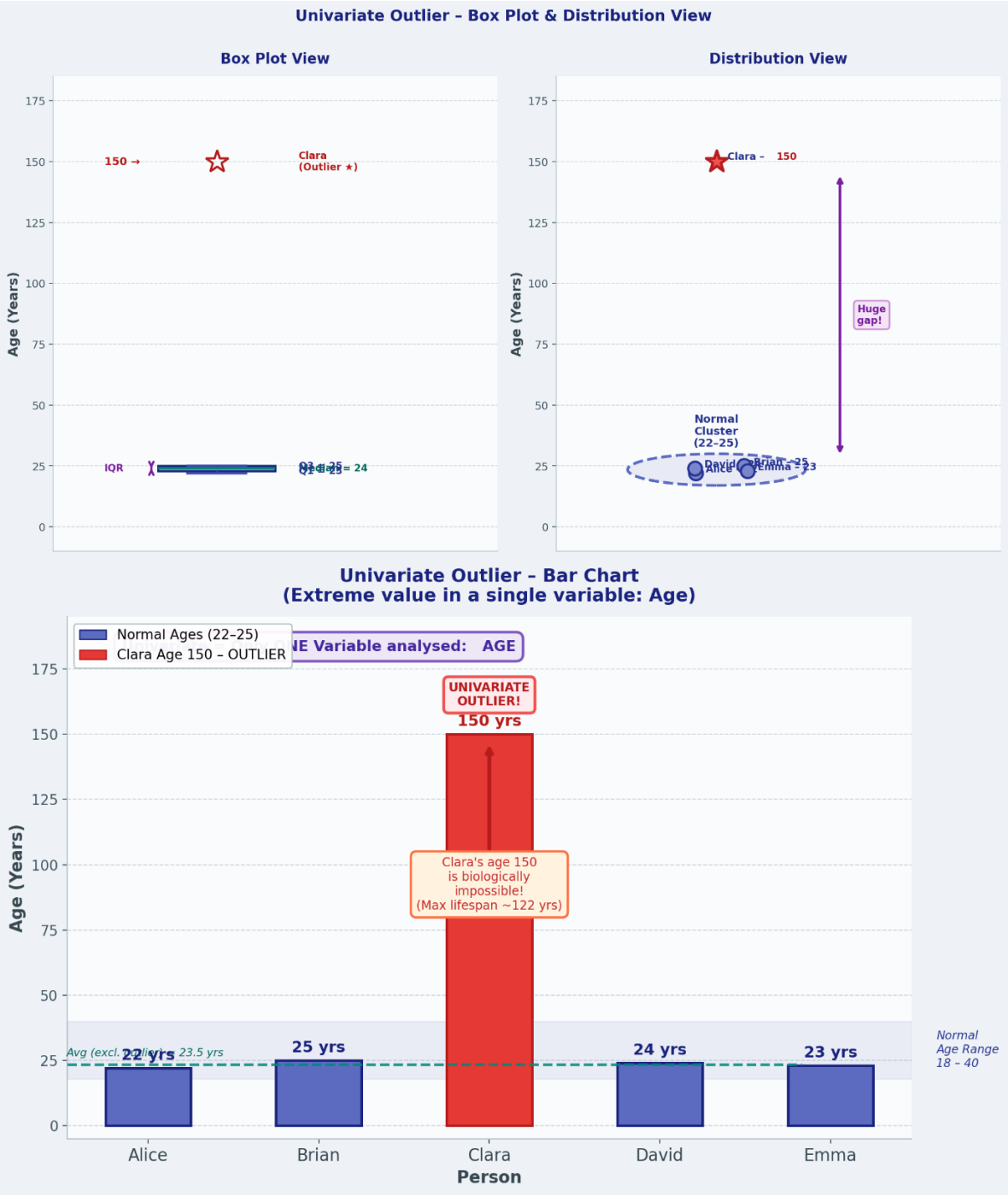



**Figure 1.3.1.1.** Univariate Outlier – Box Plot, Distribution View and Bar Chart

## 1.3.2. Multivariate Outlier

A multivariate outlier is a combination of unusual or extreme result for **at least two variables.**

These are outliers that occur in a **combination of more than one variable.**

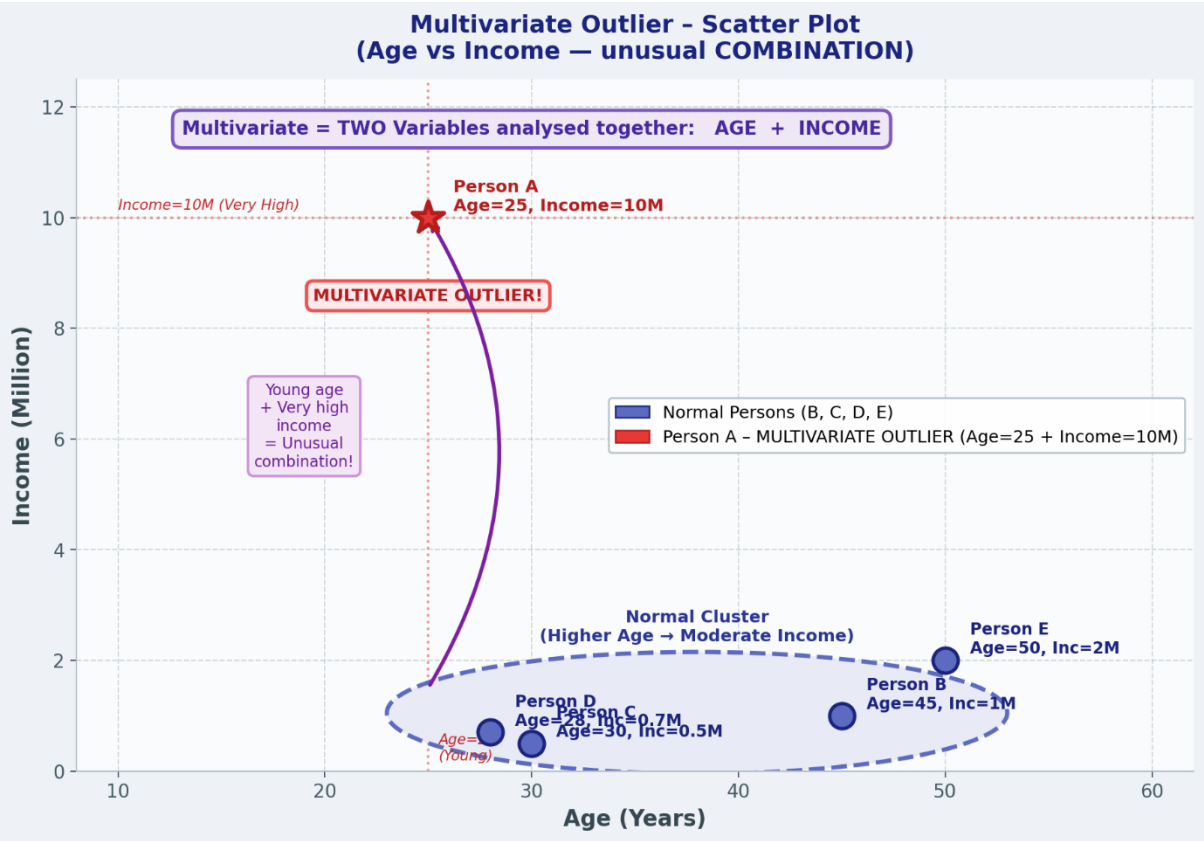
**Example:** A person's age is 25 years but income is 10 million - this combination is unusual and can be a multivariate outlier (but confirmation is necessary).

**Table 1.3.2.1.** Multivariate Outlier – Data Table

**Multivariate Outlier – Data Table**
**(Only the unusual values 25 & 10 are red — the combination is the outlier)**

| Person | Age | Income (Million) | Outlier? |
|---|---|---|---|
| A | **25** | **10** | **Yes** |
| B | 45 | 1 | No |
| C | 30 | 0.5 | No |
| D | 28 | 0.7 | No |
| E | 50 | 2 | No |

*Note: The COMBINATION of Age=25 (red) + Income=10M (red) together makes it a Multivariate Outlier*
*25 alone is not outlier | 10M alone might be borderline | Together = definitely unusual*



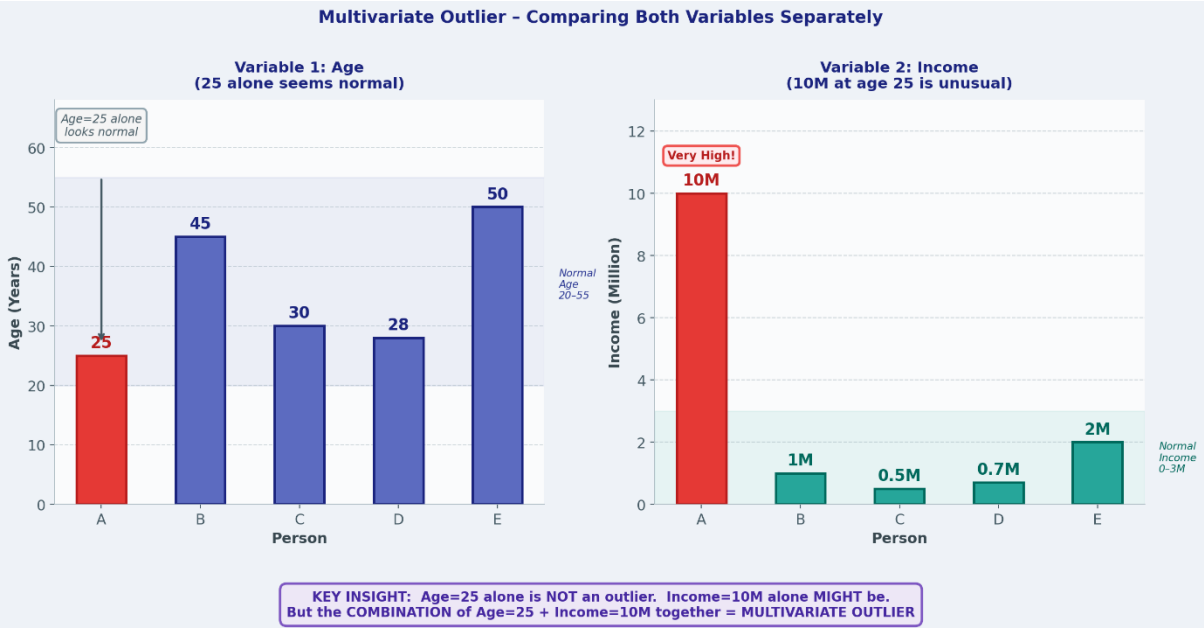



**Figure 1.3.2.1.** Multivariate Outlier – Box Plot, Distribution View and Bar Chart

## 1.3.3. Global Outlier

A global outlier is a **data point that shows an unusual or extreme result** when compared to the **entire dataset.**

These are outliers that are abnormal in the context of the entire dataset.

**Example:** If the average height of students in a school is 5 feet, and one student's height is 11 feet, then this would be a global outlier.

**Table 1.3.3.1.** Global Outlier – Data Table

**Global Outlier - Data Table**
**(Only the extreme value 11.0 is red — abnormal across the entire dataset)**

| Student | Height (Feet) | Global Outlier? |
|---|---|---|
| A | 5.1 | No |
| B | 4.9 | No |
| C | 5.0 | No |
| D | 5.2 | No |
| E | **11.0** | **Yes** |

*Note: Only the HEIGHT value "11.0" is red — it is a Global Outlier because it is extreme when compared to the ENTIRE dataset (not just a group or season)*


**Figure 1.3.3.1.** Global Outlier – Box Plot, Distribution View and Bar Chart

## 1.3.4. Local (Contextual) Outlier

A local (contextual) outlier is **a data point that shows an unusual or extreme result** only **within a specific cluster, group, or context**, but **may appear normal** when **compared to the entire dataset.**

These are outliers that are abnormal only in the context of a specific cluster or group.

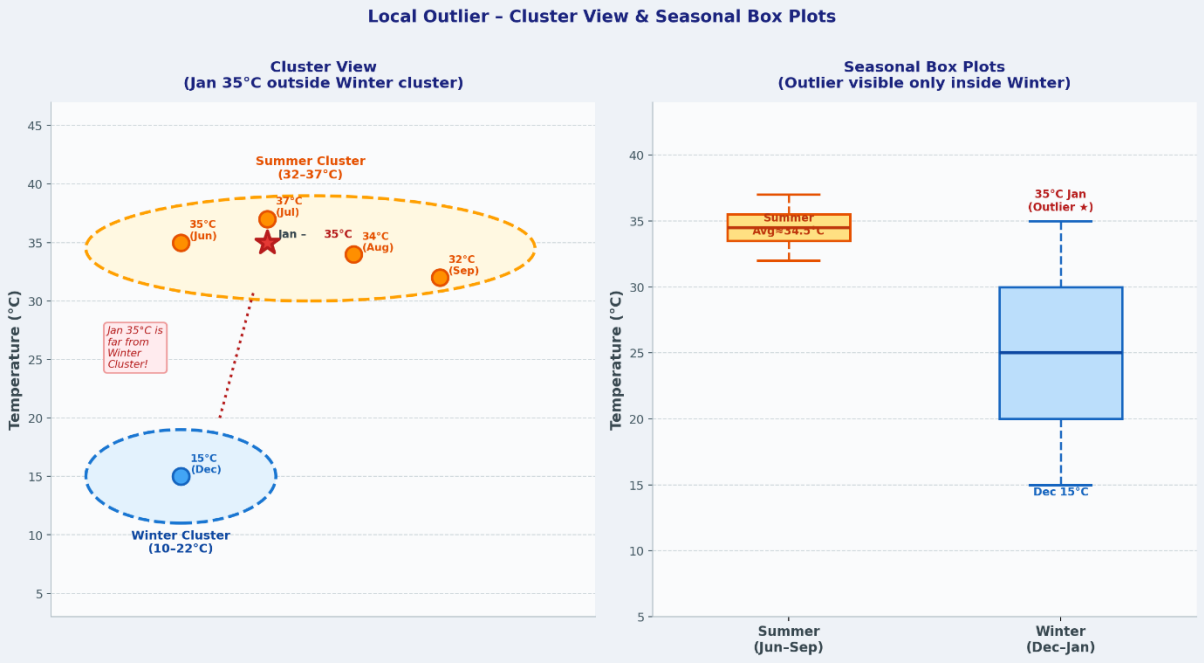
**Example:** In summer, a temperature of 35°C is normal, but 35°C in January would be a local outlier based on the winter season in Pakistan.

**Table 1.3.4.1.** Local (Contextual) Outlier – Data Table

**Local (Contextual) Outlier – Data Table**
**(Only the unusual value 35 in January is red — normal in summer, outlier in winter)**

| Month | Temperature (°C) | Local Outlier? | Reason |
|---|---|---|---|
| June | 35 | No | Normal summer temperature |
| July | 37 | No | Normal summer temperature |
| August | 34 | No | Normal summer temperature |
| September | 32 | No | Normal summer temperature |
| December | 15 | No | Normal winter temperature |
| January | **35** | **Yes** | *Unusual for winter season* |

*Note: Temperature 35°C (red) is a LOCAL OUTLIER only in January's winter context — the same value is perfectly normal in June, July, August & September (summer)*



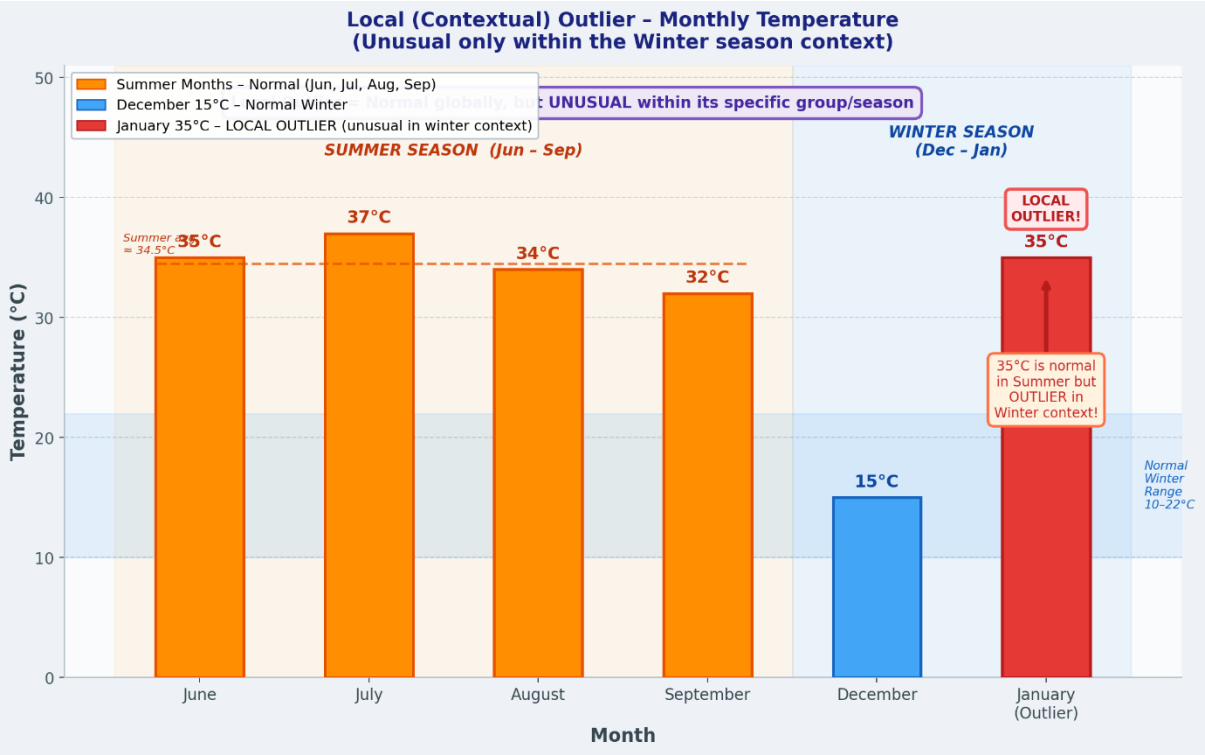



**Figure 1.3.4.1.** Local (Contextual) Outlier – Box Plot, Distribution View and Bar Chart

## 1.3.5. Point Outliers

A point outlier is a **single data point that shows an unusual or extreme value compared to the rest of the dataset**.

These are **individual data points that are completely different from the rest of the data**.

**Example:** In a class, all students' marks are between 60-80/100, but one student's marks are 2/100 - this is a point outlier.

**Table 1.3.5.1.** Point Outlier – Data Table

**Point Outlier – Data Table**
**(Only the extreme value 2 is red — single point completely different from the rest)**

| Student | Marks (/100) | Point Outlier? |
|---|---|---|
| A | 72 | No |
| B | 68 | No |
| C | 75 | No |
| D | 80 | No |
| E | 65 | No |
| F | 78 | No |
| G | 70 | No |
| H | 62 | No |
| I | 77 | No |
| J | **2** | **Yes** |

*Note: Only the MARKS value "2" is red — it is a Point Outlier because it is a single data point drastically different from all others (62–80)*


**Figure 1.3.5.1.** Point Outlier – Box Plot, Distribution View and Bar Chart

## 1.3.6. Contextual Outliers

A **contextual outlier** is a data point that appears normal in general but becomes unusual or extreme within a **specific context such as time, location, or condition**.

These are outliers that are abnormal in a **specific context**.

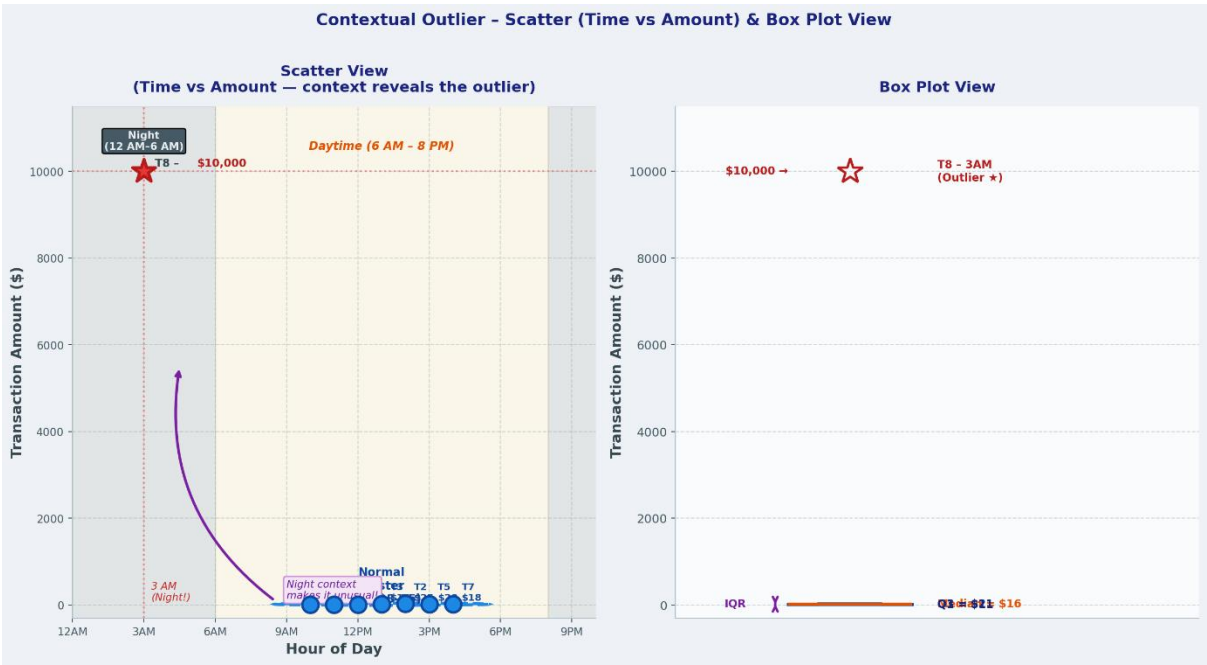
**Example:** A $10 transaction on a credit card is normal, but a $10,000 transaction at 3 AM could be a contextual outlier which banks use fraud detection applications for.

**Table 1.3.6.1.** Contextual Outlier – Data Table

**Contextual Outlier – Data Table**
**(Amount $10,000 + Time 3 AM together form the unusual context)**

| Transaction | Amount ($) | Time | Context | Outlier? |
|---|---|---|---|---|
| T1 | 10 | 10 AM | Day | No |
| T2 | 25 | 2 PM | Day | No |
| T3 | 15 | 1 PM | Day | No |
| T4 | 8 | 11 AM | Day | No |
| T5 | 20 | 3 PM | Day | No |
| T6 | 12 | 12 PM | Day | No |
| T7 | 18 | 4 PM | Day | No |
| T8 | **10,000** | **3 AM** | **Night** | **Yes** |

*Note: "$10,000" (red) is unusual. "3 AM" (red) is the context that makes it a Contextual Outlier. A $10,000 transaction at 2 PM may be normal — but at 3 AM it triggers fraud detection!*



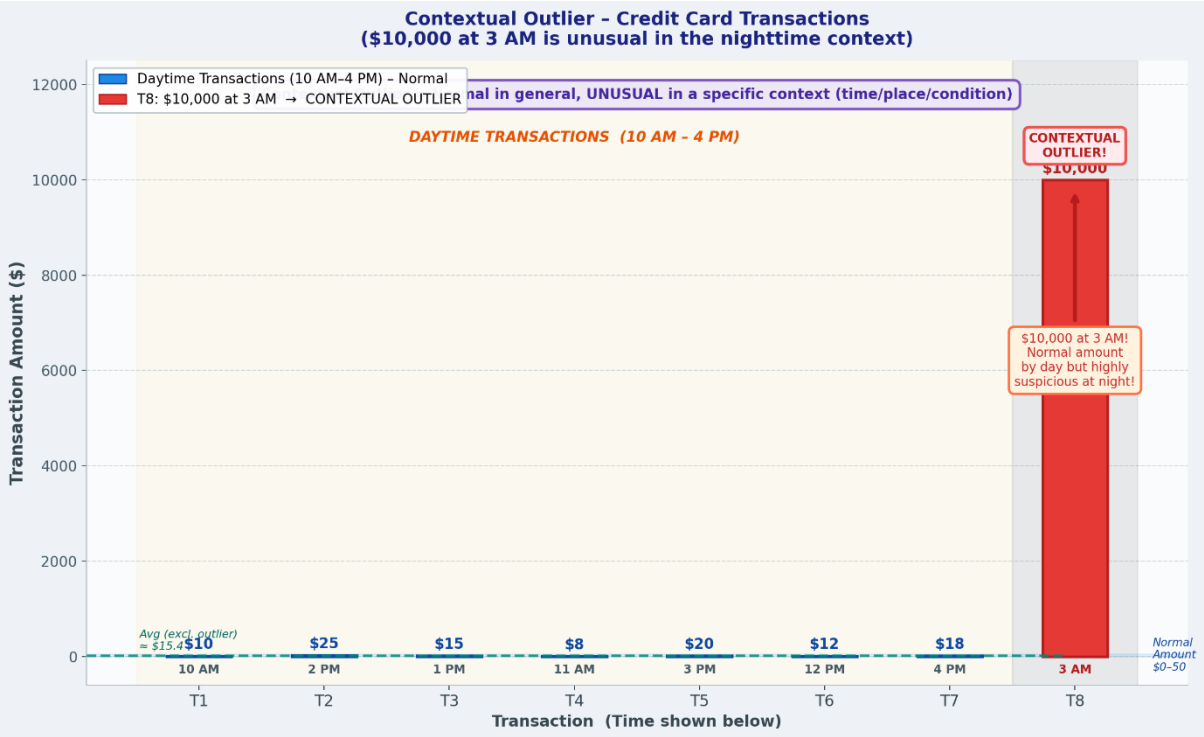



**Figure 1.3.6.1.** Contextual Outlier – Box Plot, Distribution View and Bar Chart

## 1.3.7. Collective Outliers

A **collective outlier** is a **group of data points** that **may appear normal when observed individually** but **form an unusual or abnormal pattern when considered together.**

This is a **group of data points** that may be **individually normal** but **are collectively abnormal.**

**Example:** If a website suddenly receives 1000 visits from the same IP address in one day, this is a collective outlier (possible bot attack or DDoS (Distributed Denial of Service) attack).

**Table 1.3.7.1.** Collective Outlier – Data Table

**Collective Outlier – Data Table**
**(Individually normal, but 1,000 visits from ONE IP = collectively abnormal pattern)**

| IP Address | Visits (per day) | Individually Normal? | Collective Outlier? |
|---|---|---|---|
| IP-A | 12 | Yes | No |
| IP-B | 8 | Yes | No |
| IP-C | 15 | Yes | No |
| IP-D | 10 | Yes | No |
| IP-E | 9 | Yes | No |
| IP-F | 11 | Yes | No |
| IP-G | 7 | Yes | No |
| IP-H | **1,000** | *Each visit is normal* | **Yes – Bot/DDoS Pattern** |

*Key: Each single visit by IP-H is normal on its own — but 1,000 visits TOGETHER form an abnormal GROUP pattern (Bot Attack / DDoS) — that is what makes it a Collective Outlier*

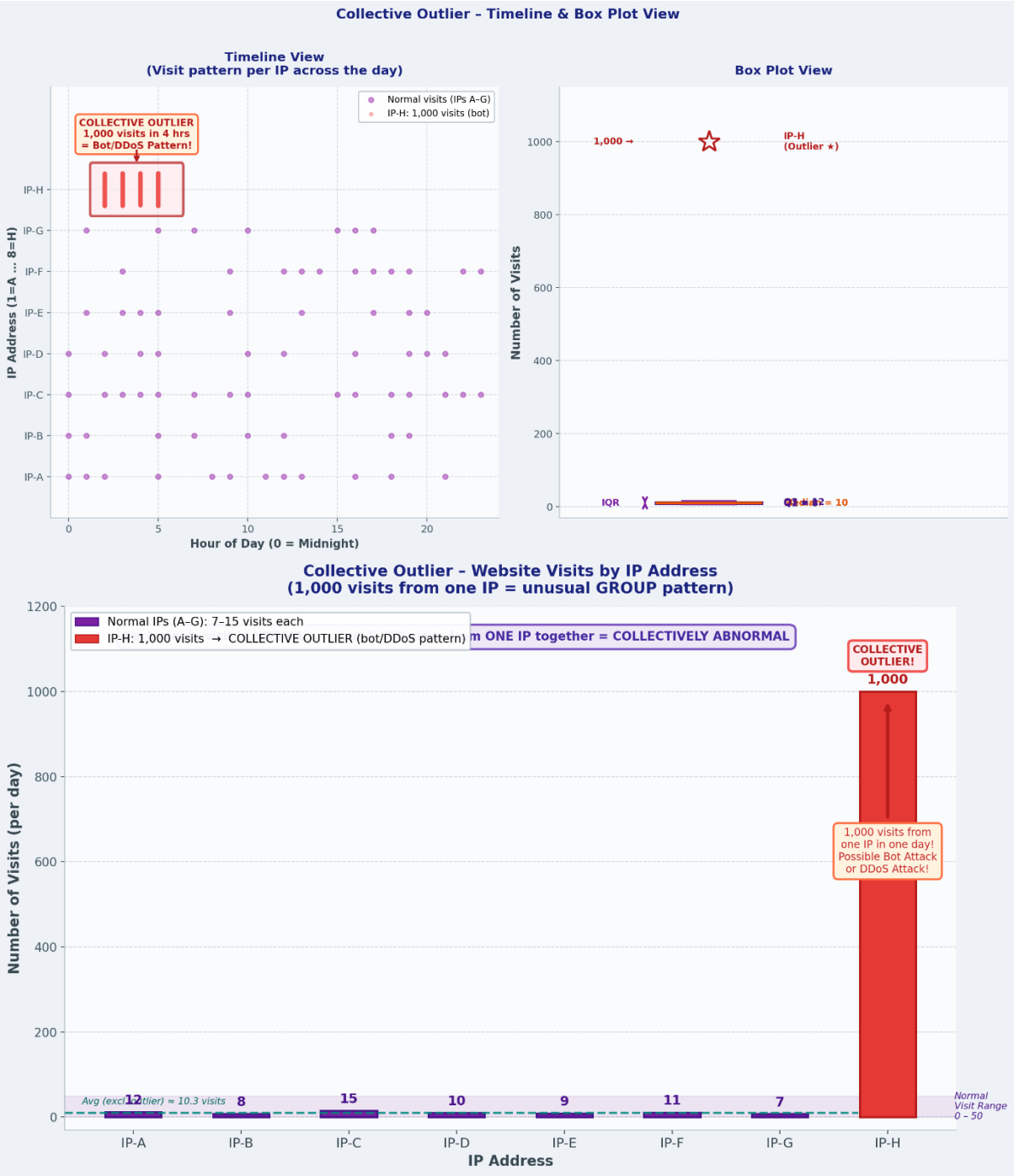



**Figure 1.3.7.1.** Collective Outlier – Box Plot, Distribution View and Bar Chart

## 1.3.8. Recurrent Outliers

A **recurrent outlier** is a data point or pattern that appears unusual but **occurs repeatedly over time in a consistent manner**.

These are outliers that **occur repeatedly**.

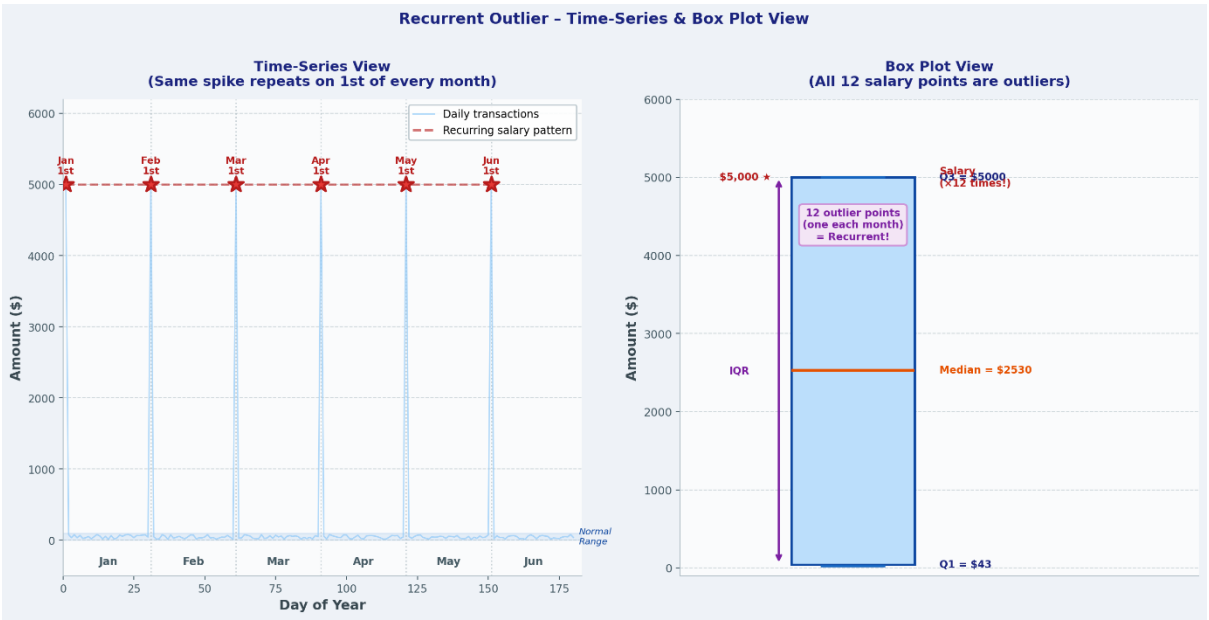
**Example:** Money comes into someone's salary account on the 1st of every month - this is a recurring pattern.

**Table 1.3.8.1.** Recurrent Outlier – Data Table

**Recurrent Outlier - Data Table**
**($5,000 salary deposit appears on 1st of EVERY month = same outlier repeating 12 times)**

| Month | Date | Normal Spend ($) | Salary Deposit ($) | Recurrent Outlier? |
|---|---|---|---|---|
| January | 1st Jan | 45 | 5,000 | Yes |
| February | 1st Feb | 30 | 5,000 | Yes |
| March | 1st Mar | 60 | 5,000 | Yes |
| April | 1st Apr | 25 | 5,000 | Yes |
| May | 1st May | 50 | 5,000 | Yes |
| June | 1st Jun | 35 | 5,000 | Yes |
| July | 1st Jul | 40 | 5,000 | Yes |
| August | 1st Aug | 55 | 5,000 | Yes |
| September | 1st Sep | 28 | 5,000 | Yes |
| October | 1st Oct | 48 | 5,000 | Yes |
| November | 1st Nov | 38 | 5,000 | Yes |
| December | 1st Dec | 52 | 5,000 | Yes |

*Note: All 5,000 values (red) are outliers compared to normal daily spend (25–60)*
*But because they REPEAT every month on the 1st — this is a RECURRENT OUTLIER pattern*



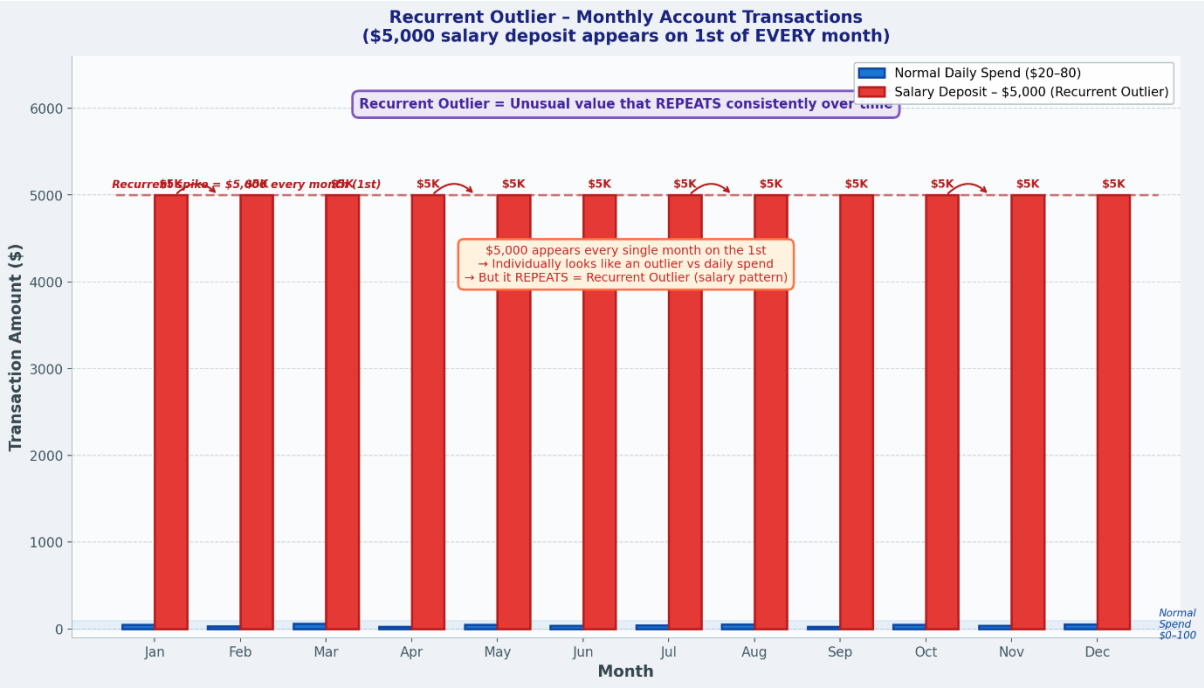



**Figure 1.3.8.1.** Recurrent Outlier – Box Plot, Distribution View and Bar Chart

## 1.3.9. Periodic Outliers

These are outliers that occur **regularly in a specific period**.

A **periodic outlier** is a data point or pattern that appears extreme but **occurs regularly during a specific time period or season**.

**Example:** On Black Friday or Eid, shopping websites have very high traffic - this is a periodic outlier and cannot be compared with regular days.

**Table 1.3.9.1.** Periodic Outlier – Data Table

**Periodic Outlier – Data Table**
**(Only 9,500 & 8,800 are red — extreme but expected every year during specific events)**

| Month | Traffic (Visits/Day) | Event | Periodic Outlier? | Reason |
|---|---|---|---|---|
| January | 1,200 | — | No | Normal daily traffic |
| February | 1,100 | — | No | Normal daily traffic |
| March | 1,300 | — | No | Normal daily traffic |
| April | 1,150 | — | No | Normal daily traffic |
| May | 1,250 | — | No | Normal daily traffic |
| June | 1,050 | — | No | Normal daily traffic |
| July | 1,180 | — | No | Normal daily traffic |
| August | 1,220 | — | No | Normal daily traffic |
| September | 1,080 | — | No | Normal daily traffic |
| October | 1,350 | — | No | Normal daily traffic |
| November | **9,500** | **Black Friday** | **Yes** | *Annual sale event spike* |
| December | **8,800** | **Eid Shopping** | **Yes** | *Annual Eid season spike* |

*Note: Traffic values "9,500" and "8,800" (red) are PERIODIC OUTLIERS — extreme but predictable*
*They occur every year during Black Friday & Eid — they CANNOT be compared with regular days*

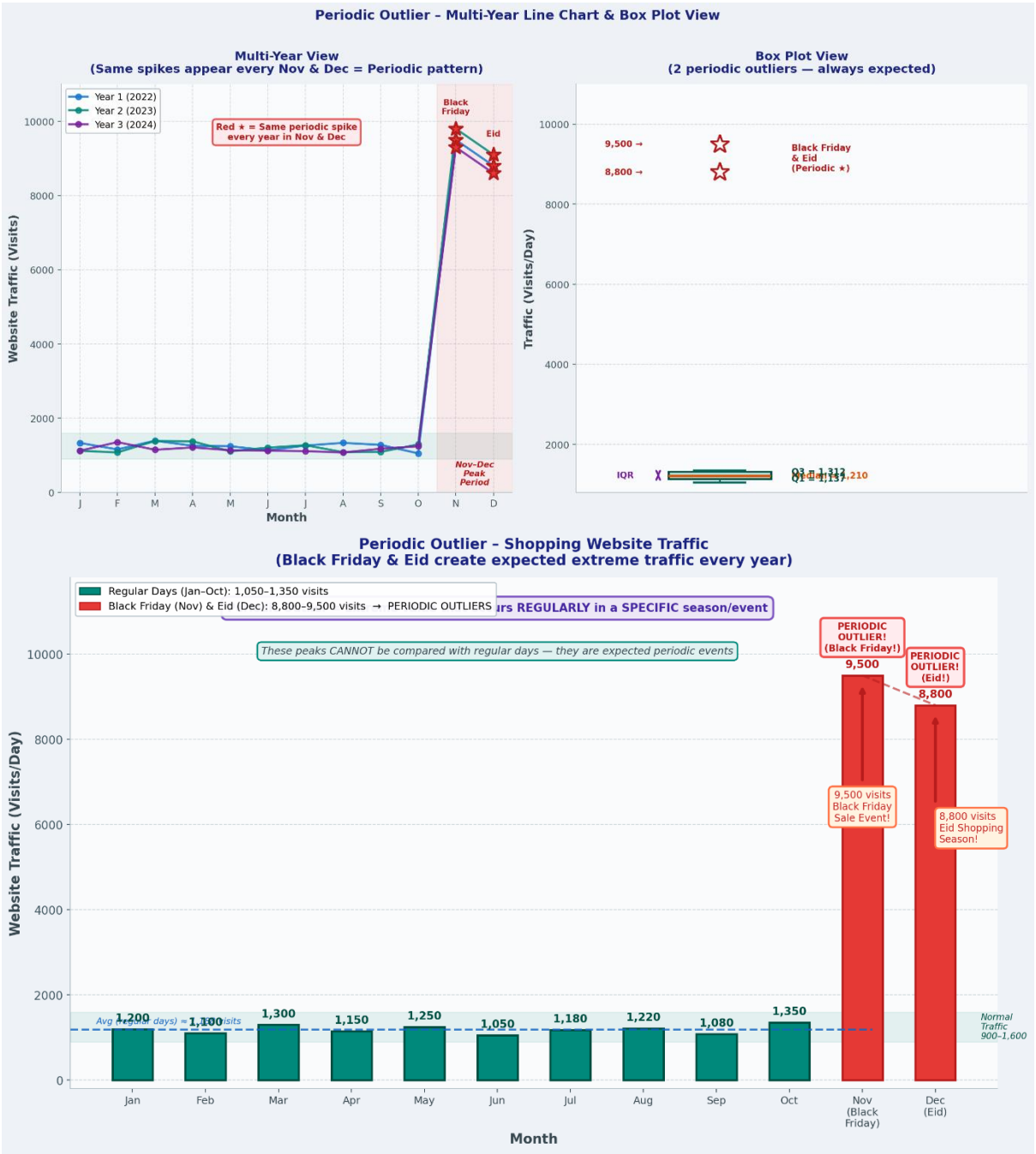



**Figure 1.3.9.1.** Periodic Outlier – Box Plot, Distribution View and Bar Chart

## 1.4. Causes of Outliers

Outliers can be caused by anything. Some common causes are given below:

### 1.4.1. Data Entry Errors

While entering data, outliers can occur due to human error.

For example, if your data has an age variable, and someone enters age as 1000 years instead of 100 years, then this would be an outlier.

### 1.4.2. Measurement Errors

While measuring data, outliers can occur due to equipment or human error.

For example, if your data has a height variable, and due to a faulty measuring tape, someone's height is measured as 50 feet instead of 5 feet, then this would be an outlier.

### 1.4.3. Experimental Errors

While conducting experiments with data, outliers can occur due to faulty procedures or equipment.

For example, if while measuring the quantity of a chemical in a lab the scale malfunctions, the readings can become outliers.

### 1.4.4. Intentional Outliers

In some cases, people intentionally enter incorrect data.

For example, in online surveys if someone enters their income as $10,000,000 instead of $10,000 to maintain privacy, then this would be an outlier.

### 1.4.5. Data Processing Errors

Outliers can appear due to technical errors while processing, transforming, or merging data.

For example, if the decimal point is placed incorrectly during currency conversion ($10050 instead of $100.50), then this becomes an outlier.

### 1.4.6. Sampling Errors

While collecting data, unusual cases can be sampled due to faulty sampling methods.

For example, if you are conducting a middle-class income survey and accidentally include a billionaire's data as well, then this would be an outlier.

### 1.4.7. Natural Outliers

In the real world, there are some genuine unusual events or cases that are naturally outliers.

For example, the sudden spike in hospital visits during the COVID-19 pandemic, or an athlete's exceptional performance that breaks records - these are natural outliers that are genuine and represent real phenomena.

## 1.5. Why should we care about Outliers?

Here you can see a complete understanding about outliers in this figure.

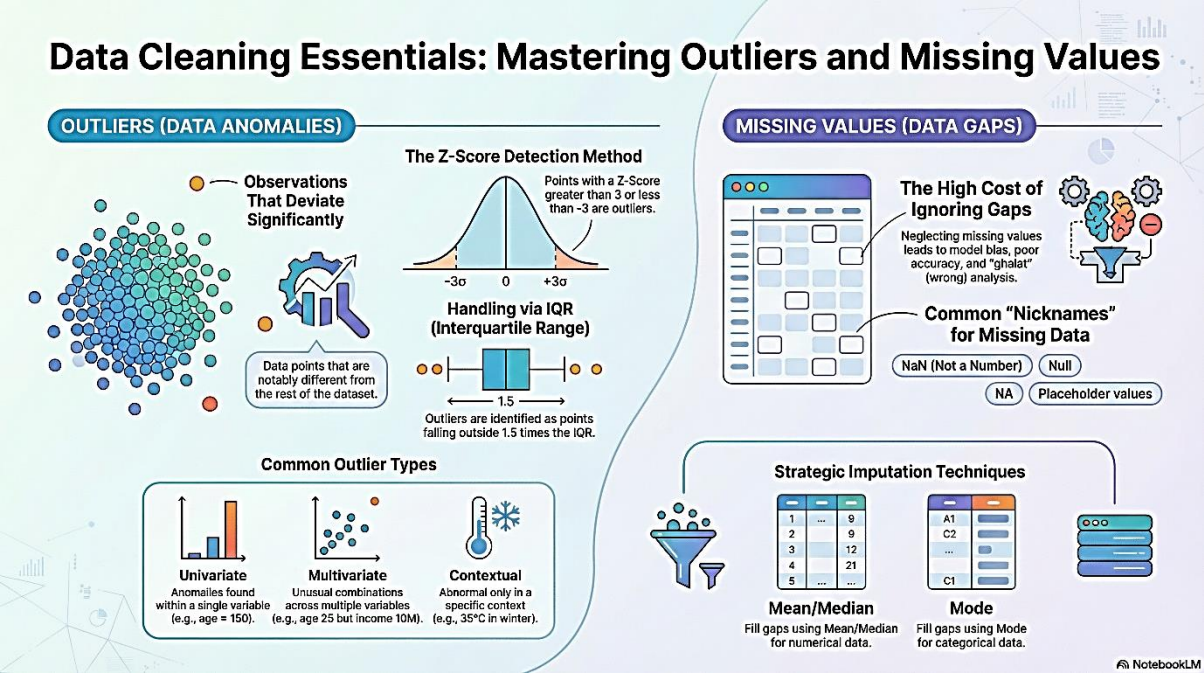



**Figure 1.5.** A comprehensive understanding of outliers and their impact on data analysis

### 1.5.1. Hidden Clues

Outliers give us hidden clues. By identifying them, we can discover hidden patterns.

Outliers are not always wrong; sometimes they can really enhance your understanding, and they can even inspire great research questions.

### 1.5.2. Data Quality

Data quality decreases due to outliers. By identifying them, we can improve data quality.

### 1.5.3. Impact Analysis

Errors occur in our analysis due to outliers. By identifying them, we can improve our analysis.

### 1.5.4. Better Decisions

Our decisions are also affected by outliers. By identifying them, we can make better decisions.

### 1.5.5. Better Models

The accuracy of our models decreases due to outliers. By identifying them, we can build better models.

### 1.5.6. Better Insights

Our insights are also affected by outliers. By identifying them, we can extract better insights.

### 1.5.7. Better Visualization

The quality of our visualizations decreases due to outliers. By identifying them, we can create better visualizations.

### 1.5.8. Better Storytelling

Our storytelling is also affected by outliers. By identifying them, we can create better stories.

### 1.5.9. Better Data Products

The quality of our data products decreases due to outliers. By identifying them, we can create better data products.

### 1.5.10. Better Data Science

The quality of our data science decreases due to outliers. By identifying them, we can do better data science.

## 1.6. Detect and Remove Outliers

To identify outliers, we use some techniques. We call these techniques 'Outlier Detection Techniques'. Some of these techniques are given below:

1. Z-Score
2. IQR
3. DBSCAN
4. Isolation Forest
5. Local Outlier Factor
6. Elliptic Envelope
7. One-Class SVM
8. Mahala-Nobis Distance
9. Robust Random Cut Forest
10. Histogram-based Outlier Score
11. K-Nearest Neighbors
12. K-Means Clustering

13. Local Correlation Integral
14. and many more…

We will only look at **Z-Score, IQR**, and **K-Means Clustering**.

[Outliers Detection Using IQR, Z-score, LOF and DBSCAN]

### 1.6.1. Z – Score Method

In the Z-Score method, we see how many standard deviations (SD) a data point is away from the mean.

The formula for Z-Score is:

$$z = \frac{x - \mu}{\sigma}$$

Where:

- Z – is the Z-Score
- x – is the data point
- μ – is the mean of the data
- σ – is the standard deviation of the data
- (x - μ): is the difference between the data point and the mean
- Z – is the difference between the data point and the mean in terms of standard deviations

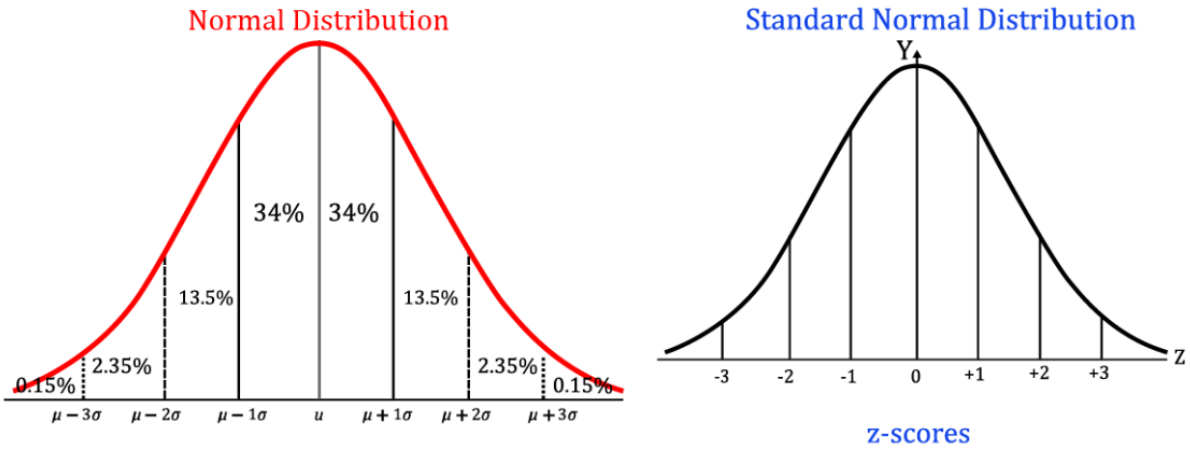



**Figure 1.6.1.1.** Normal Distribution and Z-Score Standardization

#### 1.6.1.1. Properties of Z-Score

1. The mean of Z-Score is 0 and standard deviation is 1.

2. The higher the Z-Score value, the farther the data point is from the mean.
3. The lower the Z-Score value, the closer the data point is to the mean.
4. If the Z-Score value is greater than 3 or less than -3, then the data point is an outlier.

**Table 1.6.1.1.** Z-Score Thresholds for Outlier Detection

| Z-Score | Data Point | Interpretation |
|---|---|---|
| < -3 | More than 3 SDs below the mean | Outlier |
| -3 to -2 | 2-3 SDs below the mean | Not an outlier (threshold) |
| -2 to -1 | 1-2 SDs below the mean | Not an outlier |
| -1 to 0 | 0-1 SD below the mean | Not an outlier |
| 0 | Mean | Not an outlier |
| 0 to 1 | 0-1 SD above the mean | Not an outlier |
| 1 to 2 | 1-2 SDs above the mean | Not an outlier |
| 2 to 3 | 2-3 SDs above the mean | Not an outlier (threshold) |
| > 3 | More than 3 SDs above the mean | Outlier |

### 1.6.2. Interquartile Range (IQR)

In **machine learning**, the **Interquartile Range (IQR)** method is a simple and effective statistical technique to detect **outliers** in a dataset.

The interquartile range, in short IQR, is a measure of **descriptive statistics**, which determines the range between the lower and the upper quartile, which can also be described as the middle 50% of a sample. A quartile is a type of **quantile**, which divides a sample into 4 same-sized groups of 25% each.

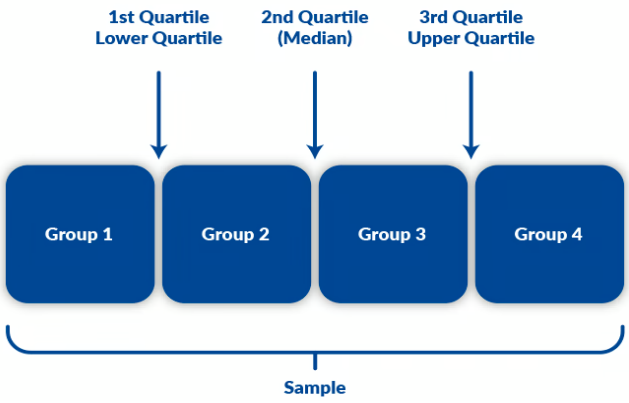



**Figure 1.6.2.1.** Quartile division of a sample illustrating Q1, median (Q2), and Q3 under the IQR framework.

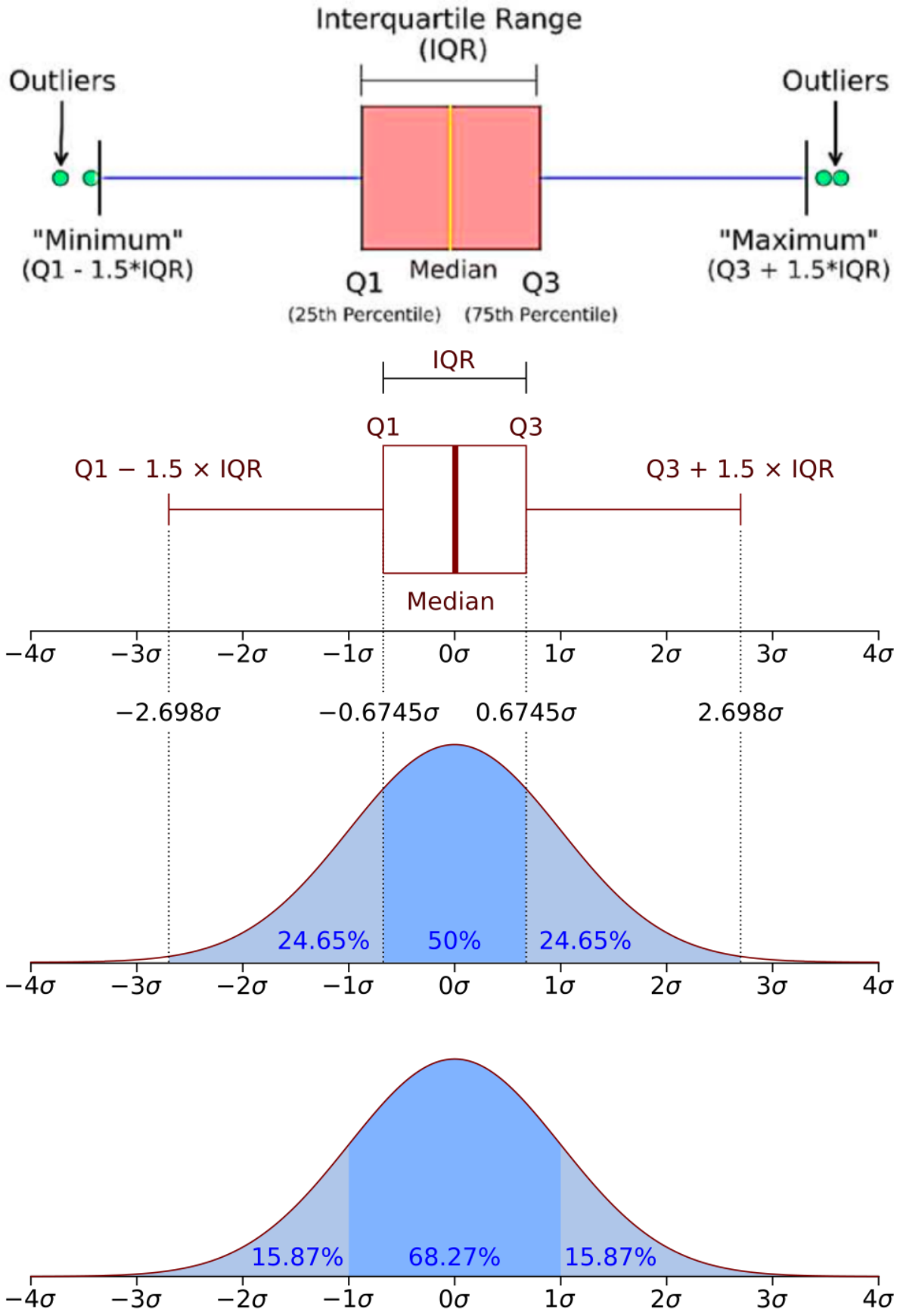



**Figure 1.6.2.2.** Boxplot with interquartile range and probability density function of a Normal $N(0, \sigma^2)$ distribution.

**Outlier Detection using IQR**

- **Lower Bound** = Q1 - 1.5 × IQR
- **Upper Bound** = Q3 + 1.5 × IQR
- Any data point **below** the Lower Bound or above the **Upper** Bound is considered an **outlier**.

**IQR** measures the spread of the middle 50% of the data.

In the IQR method, we see how many IQRs a data point is away from the median.

**The formula for IQR is**

$$\mathrm{IQR} = \mathrm{Q3} - \mathrm{Q1}$$

**Were**

- **IQR** = is the Interquartile Range
- **Q1** = 25th percentile (lower quartile i.e., first quartile)
- **Q3** = 75th percentile (upper quartile i.e., third quartile)
- **(Q3 - Q1)** = is the difference between the third quartile and the first quartile

**Outlier Rule**

- **Lower bound** = Q1 − 1.5 × IQR
- **Upper bound** = Q3 + 1.5 × IQR
- Any value **outside** these bounds is considered an outlier.

#### 1.6.2.1. Properties of IQR

1. IQR represents the range of the middle 50% of the data.
2. The higher the IQR value, the more spread out the middle 50% of the data is.
3. The lower the IQR value, the more concentrated the middle 50% of the data is around the median.

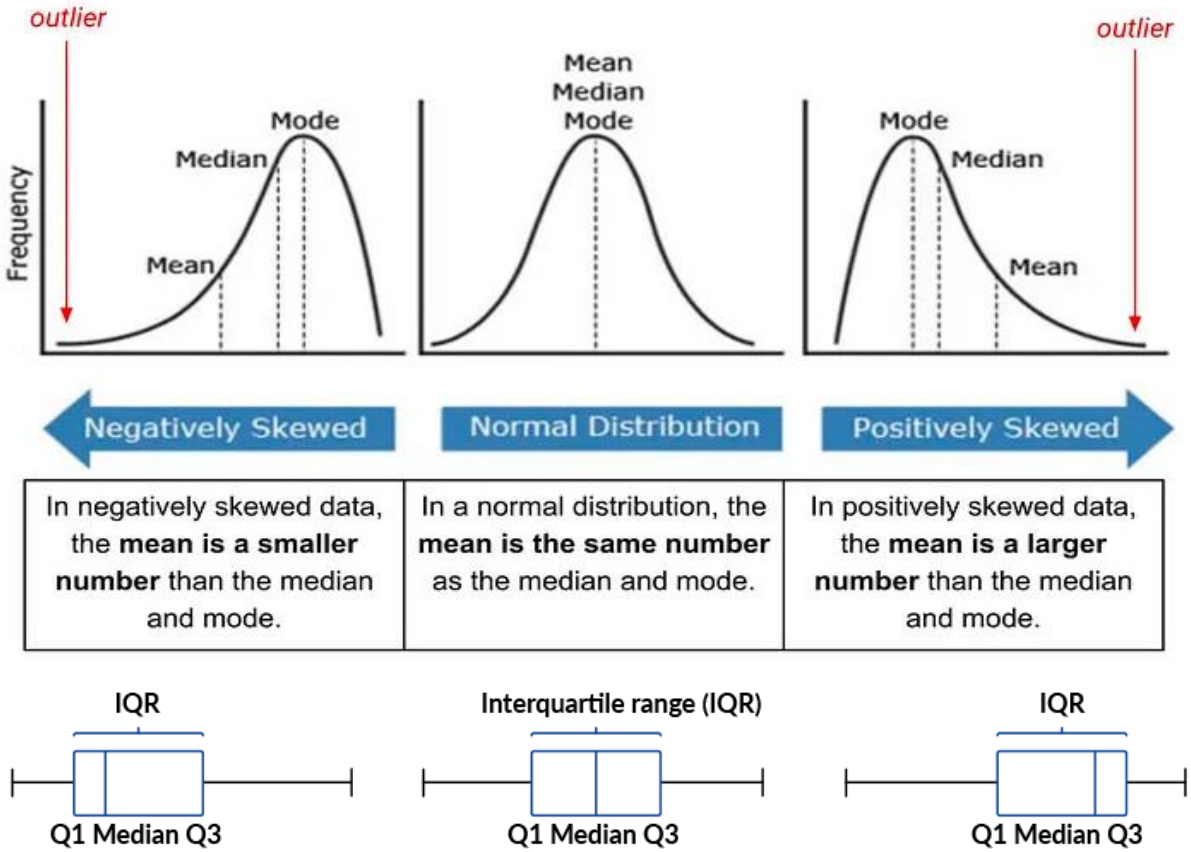



**Figure 1.6.2.3.** Boxplot showing the five-number summary and distribution skewness of the dataset.

### 1.6.3. Clustering Method (K-Means)

In the clustering method, we divide data points into clusters. This can be done using the K-Means clustering algorithm. Where we specify the number of clusters, we want to divide the data into. Then we assign each data point to a cluster.

Then we calculate the distance of each data point from the centroid of the cluster it belongs to.

Then we remove the data points that are farthest from the centroid of the cluster they belong to.

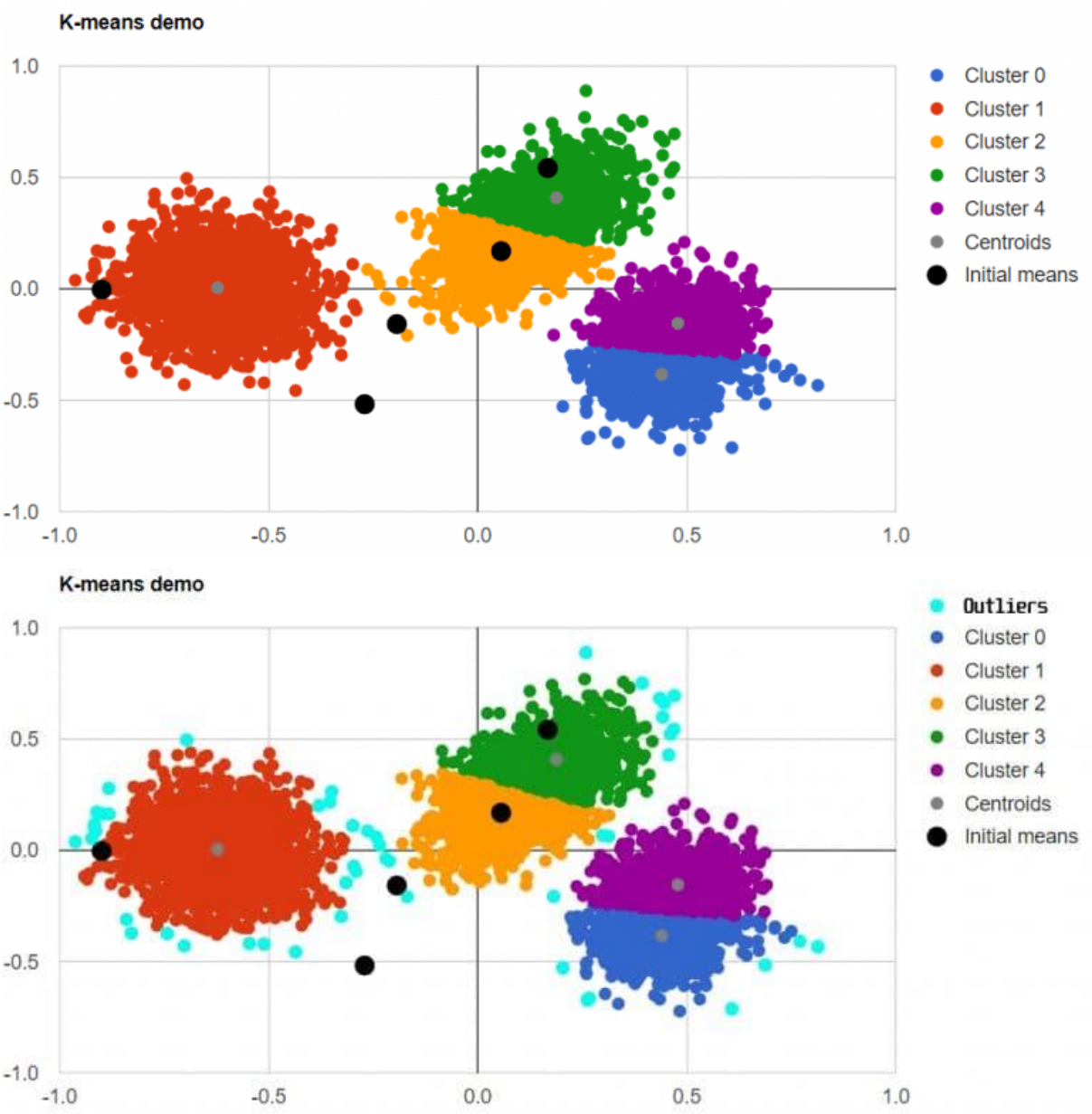



**Figure 1.6.3.1.** Outliers' detection - Illustration

## 1.7. Handling Outliers

To handle outliers, we use some techniques. We call these techniques 'Outlier Handling Techniques'. Some of these techniques are given below:

### 1.7.1. Removing the outlier

This is the most common method where all detected outliers are removed from the dataset.

### 1.7.2. Transforming and binning values

Outliers can be transformed to bring them within a range. Techniques like log transformation or square root transformation can be used.

### 1.7.3. Imputation

Outliers can also be replaced with mean, median, or mode values.

### 1.7.4. Separate treatment

In some use-cases, it's beneficial to treat outliers separately rather than removing or imputing them.

### 1.7.5. Robust Statistical Methods

Some of the statistical methods to analyze and model the data are less sensitive to outliers and provide more accurate results in the data.

## 1.8. Conclusion

1. Outliers in a dataset are observations that deviate dramatically from the rest of the data points. They might arise as a result of data gathering mistakes or abnormalities, or they can be real findings that are just infrequent or extraordinary.
2. If outliers are not appropriately accounted for, they might produce misleading, inconsistent, and erroneous findings. As a result, identifying and dealing with outliers is critical in order to produce accurate and useful data analysis findings.
3. Outliers may be detected using a variety of methods, including the percentile approach, IQR method, and z-score method. Outliers can be dealt with in a variety of methods, including removal, transformation, imputation, and so on.

## 1.9. Outlier Visualization/Detection and Removal

### 1.9.1. Visualization Method for Outlier Detections

**Box Plot** ☆☆☆☆☆

- Best for univariate outliers
- Shows quartiles, median, and outliers clearly
- Easy to interpret for non-technical audiences
- Industry standard for outlier visualization
- Library: seaborn / matplotlib

**Violin Plot** ☆☆☆☆

- Combines box plot + distribution shape
- Shows data density
- Better for understanding overall distribution

**Scatter Plot** ☆☆☆☆

- Good for seeing outliers in context
- Shows patterns and trends
- Useful for time-series or indexed data
- Library: matplotlib

**Z-score Plot** ☆☆☆☆

- Shows standardized deviations
- Clear threshold visualization (±3σ)
- Good for statistical reporting

**Histogram** ☆☆☆

- Shows distribution shape
- Less precise for outlier identification
- Better for understanding data spread
- Library: seaborn / matplotlib

**Heatmap/Correlation Matrix** ☆☆☆

- For multivariate outlier detection

- Shows relationships between variables
- Useful for complex datasets

**Pair plot:**

- Relationship-based outliers;
- Library: seaborn

### 1.9.2. LIBRARIES FOR OUTLIER DETECTION

**Table 1.9.2.1.** Essential Python libraries for outlier detection: ratings, use cases, and key functions

| Library | Rating | Category | Best For | Key Functions | When to Use |
|---|---|---|---|---|---|
| NumPy | ☆☆☆☆☆ | Computation | Array operations | percentile(), std(), mean() | Basic statistics, mathematical operations |
| Pandas | ☆☆☆☆☆ | Data Manipulation | Data preprocessing | describe(), quantile(), filter() | IQR method, data cleaning, exploration |
| SciPy | ☆☆☆☆☆ | Statistics | Statistical tests | stats.zscore(), stats.iqr() | Z-score method, advanced statistics |
| Scikit-learn | ☆☆☆☆☆ | Machine Learning | ML detection | IsolationForest, LocalOutlierFactor | Multivariate detection, production ML |
| Matplotlib | ☆☆☆☆☆ | Visualization | Custom plots | boxplot(), scatter(), hist() | Publication plots, custom visualization |
| Seaborn | ☆☆☆☆☆ | Visualization | Statistical plots | boxplot(), violinplot(), heatmap() | Quick statistical visualization |
| PyOD | ☆☆☆☆☆ | Specialized | Algorithm research | 40+ algorithms | Comparing methods, research projects |

*"Outliers are not always errors. Errors are always outliers. Know the difference."*

### 1.9.3. METHOD SELECTION MATRIX

**Table 1.9.3.1.** Method selection matrix showing recommended outlier detection approaches based on data characteristics and distribution types.

| Scenario | Best Method | Why | Libraries |
|---|---|---|---|
| **Normal distribution** | Z-Score (3σ) | Statistically sound, well-understood | scipy.stats |
| **Skewed distribution** | IQR or Modified Z-Score | Robust to skewness | pandas, scipy |
| **Multivariate data** | Isolation Forest, Mahalanobis Distance | Captures complex patterns | sklearn, scipy |
| **Time-series data** | LSTM Autoencoder, Prophet | Temporal dependencies | tensorflow, prophet |
| **High-dimensional** | Isolation Forest, PCA + Z-Score | Curse of dimensionality | Sklearn |
| **Streaming/Real-time** | Incremental algorithms, EWMA | Memory efficient | river, online-learning |
| **Small datasets (<100)** | IQR, Domain knowledge | Statistical power issues | Pandas |
| **Large datasets (>1M)** | Isolation Forest, Sampling + IQR | Computational efficiency | sklearn, dask |
| **Mixed data types** | Isolation Forest, HBOS | Handles categorical + numerical | Pyod |
| **Explainability required** | IQR, Z-Score, Domain rules | Stakeholder understanding | pandas, numpy |
| **Black-box OK** | Deep learning autoencoders, IF | Maximum performance | tensorflow, sklearn |
| **Imbalanced data** | Isolation Forest, One-Class SVM | Designed for rare events | Sklearn |
| **Clustered data** | DBSCAN, LOF | Respects local density | Sklearn |

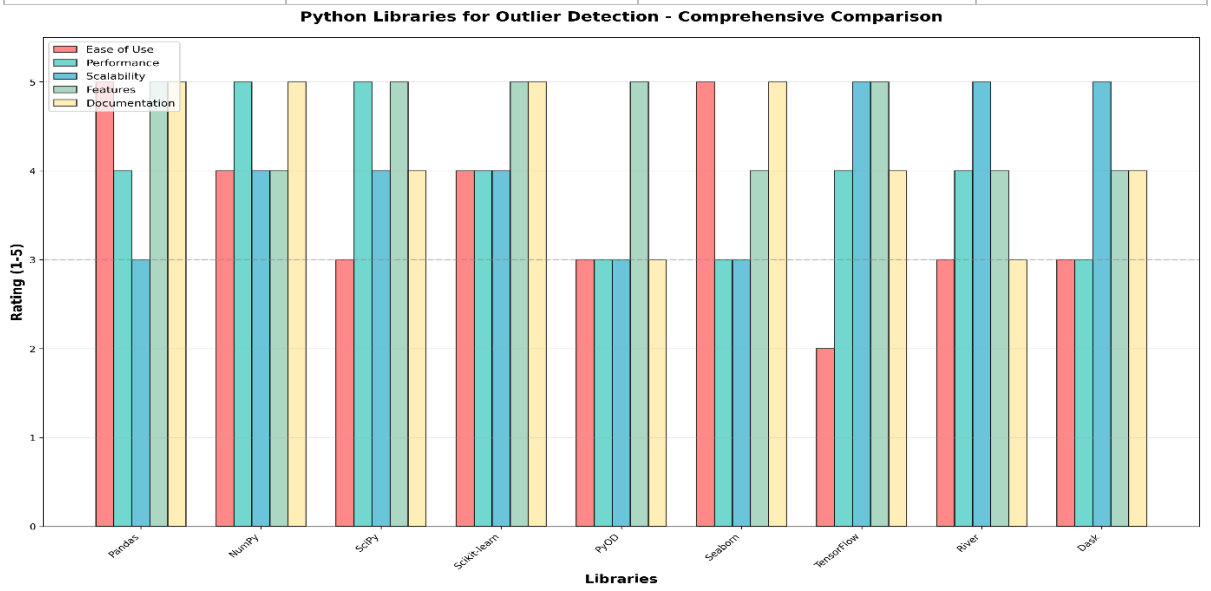



**Figure 1.9.3.1.** Comparative analysis of nine popular Python libraries for outlier detection across five key evaluation dimensions.

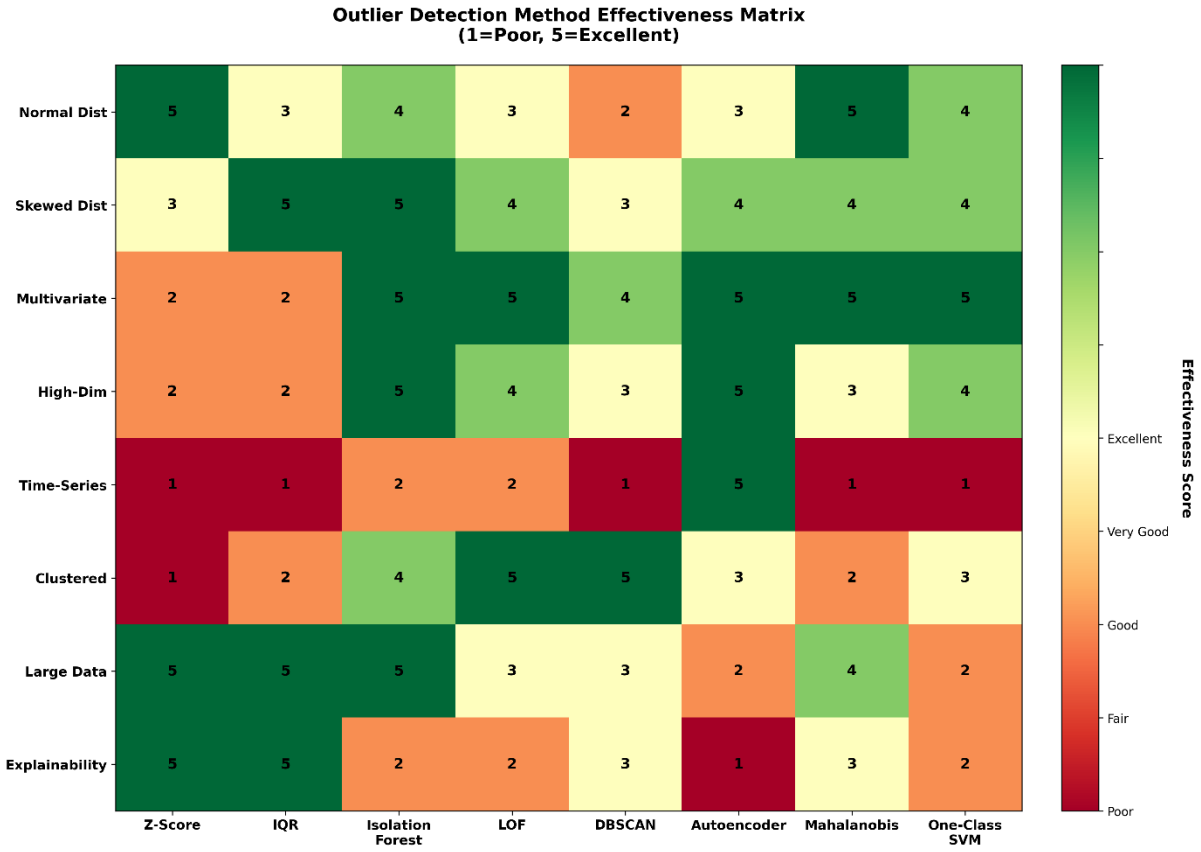



**Figure 1.9.3.2.** Effectiveness heatmap showing performance of eight outlier detection methods across eight common data scenarios.

## 1.9.4. FINAL RECOMMENDATION MATRIX

**Table 1.9.4.1.** Role-based recommendations for outlier detection tools, methods, and visualization approaches tailored to different data professional roles.

| Your Role | Primary Tool | Secondary Tool | Visualization |
|---|---|---|---|
| **Data Analyst** | IQR (Pandas) | Z-Score | Box plots, Histograms |
| **Data Scientist** | Isolation Forest | LOF, IQR | Box plots, Scatter plots |
| **ML Engineer** | Isolation Forest | Autoencoders | Dashboards, Monitoring |
| **Data Engineer** | Dask + IF | Spark + Sampling | Streaming dashboards |
| **Research Scientist** | Custom methods | Multiple ensemble | Academic plots |
| **Business Analyst** | IQR (Excel/BI) | Percentiles | Interactive dashboards |

## 1.9.5. Resources for Deep Dive

1. **PyOD**: https://github.com/yzhao062/pyod (40+ algorithms)

2. **Anomaly Detection Resources**: https://github.com/yzhao062/anomaly-detection-resources
3. **Isolation Forest Paper**: Liu et al., 2008
4. **Industry Survey**: Anomaly Detection in Practice (2023)
5. **Visualization**: Network graphs, engagement timelines

**Remember: The best method depends on:**

- Data characteristics
- Domain requirements
- Computational resources
- Explain ability needs
- Production constraints

### 1.9.6. SCALABILITY CONSIDERATIONS

**Table 1.9.6.1.** Scalability analysis of outlier detection methods showing recommended approaches, tools, and expected processing times across different dataset sizes.

| Data Size | Method | Tool | Processing Time |
|---|---|---|---|
| **< 10K rows** | Any method | Pandas + NumPy | Seconds |
| **10K - 100K** | IQR, IF, LOF | Pandas + Sklearn | Seconds - Minutes |
| **100K - 1M** | Isolation Forest | Sklearn + sampling | Minutes |
| **1M - 10M** | IF + sampling, DBSCAN | Dask, Vaex | Minutes - Hours |
| **10M - 100M** | Distributed IF | PySpark | Hours |
| **> 100M** | Sampling + streaming | Spark, Flink | Hours - Days |

## 1.10. Outlier Visualization, Detection, Validation & Document, and Removal (Coding)

Pre-reading

### 1.10.1. Interquartile Range (IQR)

*Before jumping into the code (notebook), please watch this video first. In this video, you can see how to calculate Q1, Q3, IQR, and the threshold (optional) for the given dataset, and finally determine the lower and upper bound thresholds for outlier detection from the entire dataset.*

***Inter Quartile Range (IQR) Outlier Detection in Machine Learning (Video)***

Note:

- In the IQR method, we usually think we need to arrange the dataset in ascending order, but in pandas we can compute it without sorting by using the quantile() function.
- In my code the numeric dataset **is not being sorted anywhere** before applying the IQR formula. The quantile() function in pandas does **not require sorted data**—it internally handles the computation correctly without you sorting the column.

  ```
  Q1 = numeric_df[col].quantile(0.25)
  Q3 = numeric_df[col].quantile(0.75)
  ```

  numeric_df[col].quantile(0.25) computes the **25th percentile**.

  numeric_df[col].quantile(0.75) computes the **75th percentile**.

  Pandas calculates quantiles using the values in the column; it doesn't require the column to be sorted first. Sorting is **not needed** and not done automatically in our code.

  So, our IQR calculation works fine even if the data is unsorted.

**Impute/Import Outlier Dataset from GitHub**

```
# Import necessary libraries
import pandas as pd

# Load the dataset from github (RAW link)
df =
pd.read_csv("https://raw.githubusercontent.com/tabassumgulfaraz-
```

```
ds/machine_learning_1.0/main/files_and_datasets/f_ds5_III/outlier_dataset.csv")

# Check the shape i.e., number of rows and columns also called dimensions of the dataset
print(f"Dataset Shape: {df.shape}")

df
```

***Output:***

```
Dataset Shape: (30, 5)
```

| | Age | Income | Height | Weight | Score |
|---|---|---|---|---|---|
| 0 | 25 | 45000 | 165 | 68 | 85 |
| 1 | 30 | 52000 | 170 | 72 | 78 |
| 2 | 28 | 48000 | 168 | 70 | 82 |
| 3 | 35 | 65000 | 175 | 80 | 90 |
| 4 | 22 | 38000 | 160 | 55 | 75 |
| 5 | 27 | 50000 | 172 | 75 | 88 |
| 6 | 32 | 58000 | 178 | 82 | 92 |
| 7 | 29 | 51000 | 169 | 71 | 80 |
| 8 | 26 | 47000 | 166 | 67 | 83 |
| 9 | 31 | 55000 | 174 | 78 | 87 |
| 10 | 150 | 60000 | 171 | 73 | 86 |
| 11 | 24 | 49000 | 167 | 69 | 81 |
| 12 | 33 | 62000 | 176 | 81 | 89 |
| 13 | 28 | 53000 | 170 | 74 | 84 |
| 14 | 30 | 1500000 | 173 | 77 | 91 |
| 15 | 27 | 46000 | 168 | 70 | 79 |
| 16 | 29 | 54000 | 171 | 76 | 85 |
| 17 | 25 | 44000 | 164 | 66 | 77 |
| 18 | 32 | 59000 | 177 | 83 | 93 |
| 19 | 26 | 48000 | 166 | 68 | 82 |
| 20 | 31 | 56000 | 175 | 79 | 88 |
| 21 | 28 | 50000 | 169 | 72 | 84 |
| 22 | 35 | 67000 | 180 | 350 | 90 |
| 23 | 27 | 49000 | 168 | 71 | 81 |
| 24 | 30 | 52000 | 172 | 75 | 86 |
| 25 | 24 | 1200 | 165 | 64 | 78 |

| | | | | | |
|---|---|---|---|---|---|
| 26 | 29 | 51000 | 170 | 73 | 83 |
| 27 | 26 | 47000 | 167 | 69 | 80 |
| 28 | 33 | 63000 | 176 | 82 | 2 |
| 29 | 28 | 50000 | 169 | 71 | 85 |

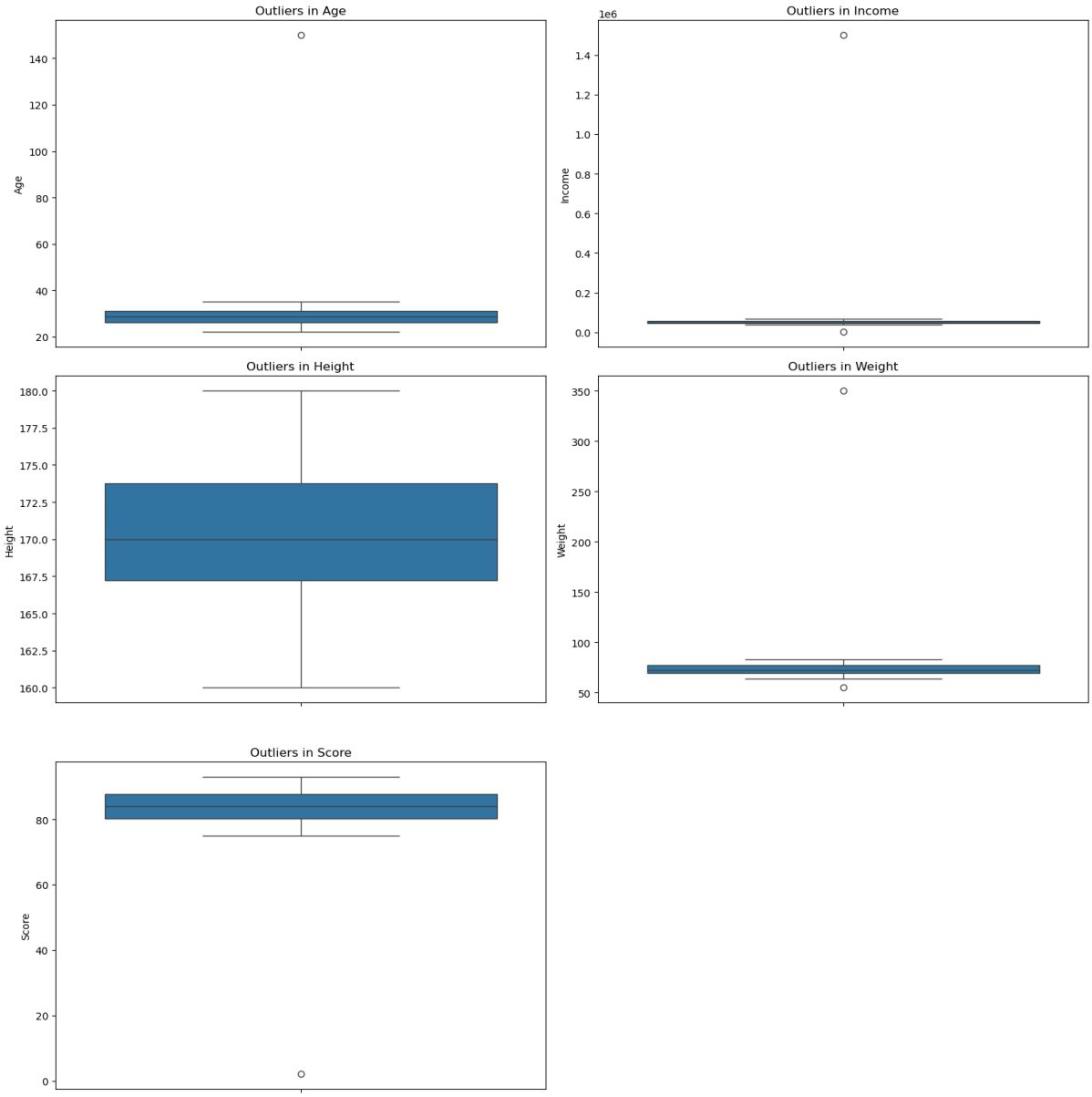
**Step 1: Select Appropriate Libraries**

```
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

**Step 2: Visualize Outliers (Boxplots)**

```
# Select numeric columns only
numeric_cols = df.select_dtypes(include=[np.number]).columns

# Set subplot layout
n_cols = 2   # plots per row
n_rows = int(np.ceil(len(numeric_cols) / n_cols))

plt.figure(figsize=(15, 5 * n_rows))

# Create boxplot for each numeric column
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(n_rows, n_cols, i)
    sns.boxplot(y=df[col])
    plt.title(f'Outliers in {col}')
    plt.ylabel(col)

plt.tight_layout()
plt.show()
```

***Output:***




**STEP 3: Detect Outliers Using IQR Method**

```
# IQR Outlier Detection

# Select numeric columns
numeric_df = df[numeric_cols]

# Create empty mask to store outlier positions
outlier_mask = pd.DataFrame(False, index=df.index,
columns=numeric_cols)

# Apply IQR formula column-wise
for col in numeric_cols:
```

```
    Q1 = numeric_df[col].quantile(0.25)
    Q3 = numeric_df[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Mark True where value is outlier
    outlier_mask[col] = (numeric_df[col] < lower_bound) |
(numeric_df[col] > upper_bound)

# Rows where ANY column has outlier
outliers = df[outlier_mask.any(axis=1)]

outliers
```

***Output:***

| | Age | Income | Height | Weight | Score |
|---|---|---|---|---|---|
| 4 | 22 | 38000 | 160 | 55 | 75 |
| 10 | 150 | 60000 | 171 | 73 | 86 |
| 14 | 30 | 1500000 | 173 | 77 | 91 |
| 22 | 35 | 67000 | 180 | 350 | 90 |
| 25 | 24 | 1200 | 165 | 64 | 78 |
| 28 | 33 | 63000 | 176 | 82 | 2 |

**Step 4: Validate and Document Result**

```
# Summary of Outliers

# Total number of outlier rows
print("Total Outlier Rows:", len(outliers))

# Percentage of dataset flagged as outlier
percentage = round((len(outliers) / len(df)) * 100, 2)
print("\nPercentage of Outliers:", percentage, "%")

# Column-wise outlier count
column_outliers = outlier_mask.sum()
```

```
print("\nColumn-wise Outlier Count:", column_outliers.to_dict())
```

***Output:***

Total Outlier Rows: 6

Percentage of Outliers: 20.0 %

Column-wise Outlier Count: {'Age': 1, 'Income': 2, 'Height': 0, 'Weight': 2, 'Score': 1}

**Step 5: Apply Removal Method**

```
# IQR Capping (Winsorization Method)
# Goal:
# Instead of removing rows that contain outliers,
# We will "cap" extreme values to a calculated boundary.
# This keeps dataset shape SAME and makes it ML-friendly.

# Create a copy of the original dataset
df_capped = df.copy()

# Loop through each numeric column
for col in numeric_cols:

    # Calculate Q1 (25th percentile)
    Q1 = df[col].quantile(0.25)

    # Calculate Q3 (75th percentile)
    Q3 = df[col].quantile(0.75)

    # Compute IQR (Interquartile Range)
    IQR = Q3 - Q1

    # Define lower and upper bounds
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Cap values below lower bound
    df_capped[col] = np.where(
```

```
        df_capped[col] < lower_bound,
        lower_bound,            # Replace with lower boundary
        df_capped[col]          # Keep original value
    )

    # Cap values above upper bound
    df_capped[col] = np.where(
        df_capped[col] > upper_bound,
        upper_bound,            # Replace with upper boundary
        df_capped[col]          # Keep original value
    )

# Check dataset shape (should be SAME as original)
print("Shape after capping:", df_capped.shape)

# Replace original dataset with capped dataset (optional)
df = df_capped.copy()

# Display dataset
df
```

***Output:***

Shape after capping: (30, 5)

| | Age | Income | Height | Weight | Score |
|---|---|---|---|---|---|
| 0 | 25.000 | 45000.0 | 165.0 | 68.0 | 85.0 |
| 1 | 30.000 | 52000.0 | 170.0 | 72.0 | 78.0 |
| 2 | 28.000 | 48000.0 | 168.0 | 70.0 | 82.0 |
| 3 | 35.000 | 65000.0 | 175.0 | 80.0 | 90.0 |
| 4 | 22.000 | 38000.0 | 160.0 | 56.5 | 75.0 |
| 5 | 27.000 | 50000.0 | 172.0 | 75.0 | 88.0 |
| 6 | 32.000 | 58000.0 | 178.0 | 82.0 | 92.0 |
| 7 | 29.000 | 51000.0 | 169.0 | 71.0 | 80.0 |
| 8 | 26.000 | 47000.0 | 166.0 | 67.0 | 83.0 |
| 9 | 31.000 | 55000.0 | 174.0 | 78.0 | 87.0 |
| 10 | 38.125 | 60000.0 | 171.0 | 73.0 | 86.0 |

| | | | | | |
|---|---|---|---|---|---|
| 11 | 24.000 | 49000.0 | 167.0 | 69.0 | 81.0 |
| 12 | 33.000 | 62000.0 | 176.0 | 81.0 | 89.0 |
| 13 | 28.000 | 53000.0 | 170.0 | 74.0 | 84.0 |
| 14 | 30.000 | 71750.0 | 173.0 | 77.0 | 91.0 |
| 15 | 27.000 | 46000.0 | 168.0 | 70.0 | 79.0 |
| 16 | 29.000 | 54000.0 | 171.0 | 76.0 | 85.0 |
| 17 | 25.000 | 44000.0 | 164.0 | 66.0 | 77.0 |
| 18 | 32.000 | 59000.0 | 177.0 | 83.0 | 93.0 |
| 19 | 26.000 | 48000.0 | 166.0 | 68.0 | 82.0 |
| 20 | 31.000 | 56000.0 | 175.0 | 79.0 | 88.0 |
| 21 | 28.000 | 50000.0 | 169.0 | 72.0 | 84.0 |
| 22 | 35.000 | 67000.0 | 180.0 | 90.5 | 90.0 |
| 23 | 27.000 | 49000.0 | 168.0 | 71.0 | 81.0 |
| 24 | 30.000 | 52000.0 | 172.0 | 75.0 | 86.0 |
| 25 | 24.000 | 33750.0 | 165.0 | 64.0 | 78.0 |
| 26 | 29.000 | 51000.0 | 170.0 | 73.0 | 83.0 |
| 27 | 26.000 | 47000.0 | 167.0 | 69.0 | 80.0 |
| 28 | 33.000 | 63000.0 | 176.0 | 82.0 | 69.0 |
| 29 | 28.000 | 50000.0 | 169.0 | 71.0 | 85.0 |

```
# Alternative: Remove detected outliers (Remove rows with ANY outlier)

# Keep only rows WITHOUT outliers
# df_clean = df[~outlier_mask.any(axis=1)]

# Reset index (optional but recommended for ML)
# df_clean = df_clean.reset_index(drop=True)

# print("Cleaned Shape:", df_clean.shape)

# Copy the cleaned dataset to a new variable for further processing
# df = df_clean.copy()
# df
```

***Output:***

Cleaned Shape: (24, 5)

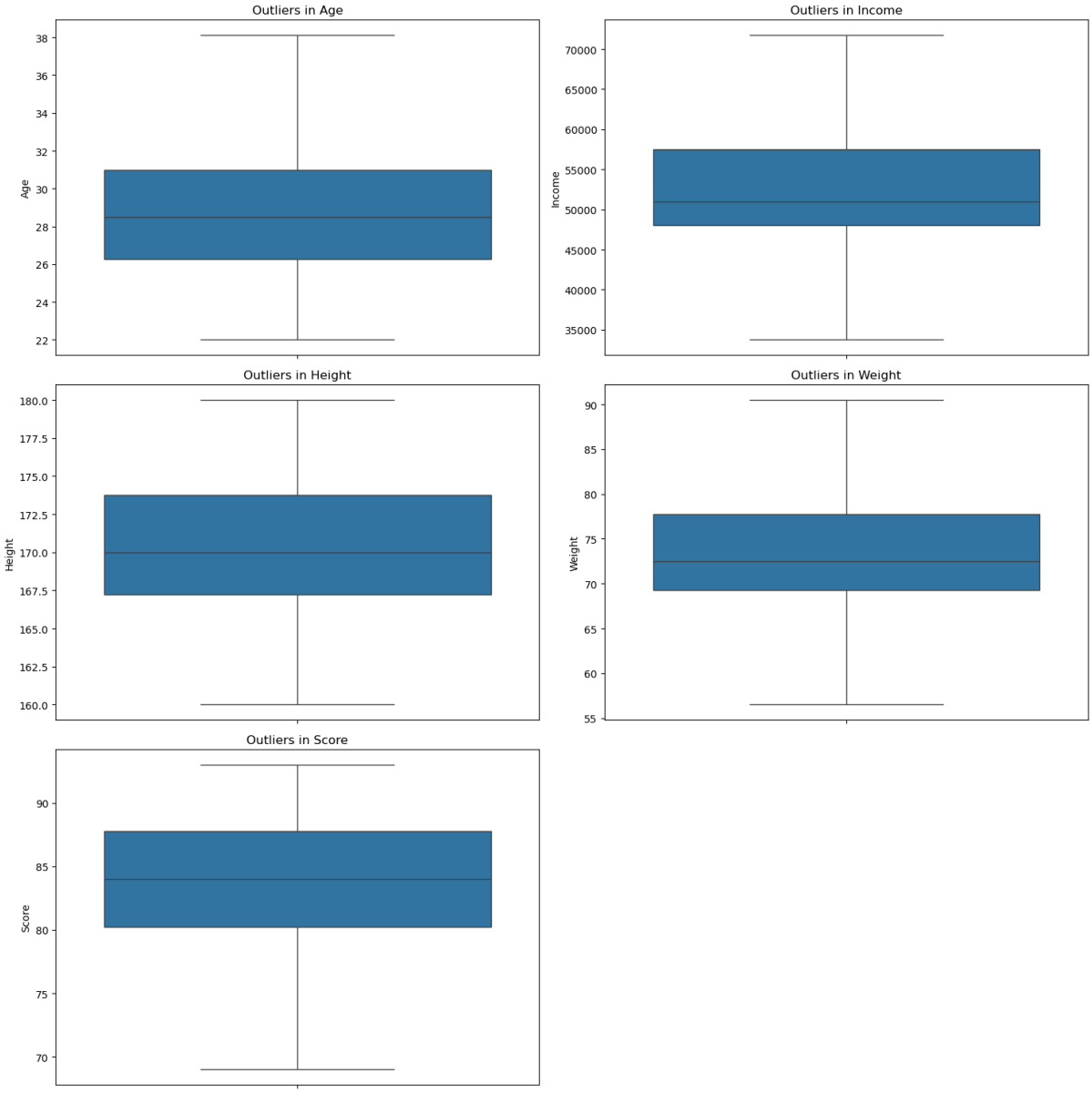
| | Age | Income | Height | Weight | Score |
|---|---|---|---|---|---|
| 0 | 25 | 45000 | 165 | 68 | 85 |
| 1 | 30 | 52000 | 170 | 72 | 78 |
| 2 | 28 | 48000 | 168 | 70 | 82 |
| 3 | 35 | 65000 | 175 | 80 | 90 |
| 4 | 27 | 50000 | 172 | 75 | 88 |
| 5 | 32 | 58000 | 178 | 82 | 92 |
| 6 | 29 | 51000 | 169 | 71 | 80 |
| 7 | 26 | 47000 | 166 | 67 | 83 |
| 8 | 31 | 55000 | 174 | 78 | 87 |
| 9 | 24 | 49000 | 167 | 69 | 81 |
| 10 | 33 | 62000 | 176 | 81 | 89 |
| 11 | 28 | 53000 | 170 | 74 | 84 |
| 12 | 27 | 46000 | 168 | 70 | 79 |
| 13 | 29 | 54000 | 171 | 76 | 85 |
| 14 | 25 | 44000 | 164 | 66 | 77 |
| 15 | 32 | 59000 | 177 | 83 | 93 |
| 16 | 26 | 48000 | 166 | 68 | 82 |
| 17 | 31 | 56000 | 175 | 79 | 88 |
| 18 | 28 | 50000 | 169 | 72 | 84 |
| 19 | 27 | 49000 | 168 | 71 | 81 |
| 20 | 30 | 52000 | 172 | 75 | 86 |
| 21 | 29 | 51000 | 170 | 73 | 83 |
| 22 | 26 | 47000 | 167 | 69 | 80 |
| 23 | 28 | 50000 | 169 | 71 | 85 |

**STEP 6: Detect Outliers Using IQR Method (step 3 repeat - Optional)**

```
# Select numeric columns only
numeric_cols = df.select_dtypes(include=[np.number]).columns

# Set subplot layout
n_cols = 2   # plots per row
n_rows = int(np.ceil(len(numeric_cols) / n_cols))

plt.figure(figsize=(15, 5 * n_rows))
```

```
# Create boxplot for each numeric column
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(n_rows, n_cols, i)
    sns.boxplot(y=df[col])
    plt.title(f'Outliers in {col}')
    plt.ylabel(col)

plt.tight_layout()
plt.show()
```

***Output:***



### 1.10.2. Z-Score

*Before jumping into the code (notebook), please watch this video first. In this video, you can see how to calculate mean (μ), standard deviation (σ), z-score for each data point (z), and set threshold* $\pm 3$ *or* $\pm 2.5$ *and finally detect the outliers for the entire dataset.* ***Z-Score Outlier Detection in Machine Learning (Video)***

| Method | Best For | Sensitive To Skew? | Outliers Detected |
|---|---|---|---|
| IQR | Skewed data | ✗ No | 6 |
| Z-Score | Normal data | ✓ Yes | 4 |

- IQR works better for skewed data
- Z-score assumes data is normally distributed (bell-shaped)
- Z-score uses mean and standard deviation
- If dataset is slightly skewed, mean shifts
- Extreme values influence standard deviation
- So fewer points cross ±3 threshold
- IQR is more robust when:
    - Data is skewed (Skewed → IQR better)
    - Small dataset
    - Heavy-tailed distribution
- Z-score works best when:
    - Data is approximately normal distribution (Normal → Z-score better)

**Impute/Import Outlier Dataset from GitHub**

```
# Import necessary libraries
import pandas as pd

# Load the dataset from github (RAW link)
df =
pd.read_csv("https://raw.githubusercontent.com/tabassumgulfaraz-ds/machine_learning_1.0/main/files_and_datasets/f_ds5_III/outlier_dataset.csv")

# Check the shape i.e., number of rows and columns
df.shape
```

***Output:***

```
Dataset Shape: (30, 5)
```

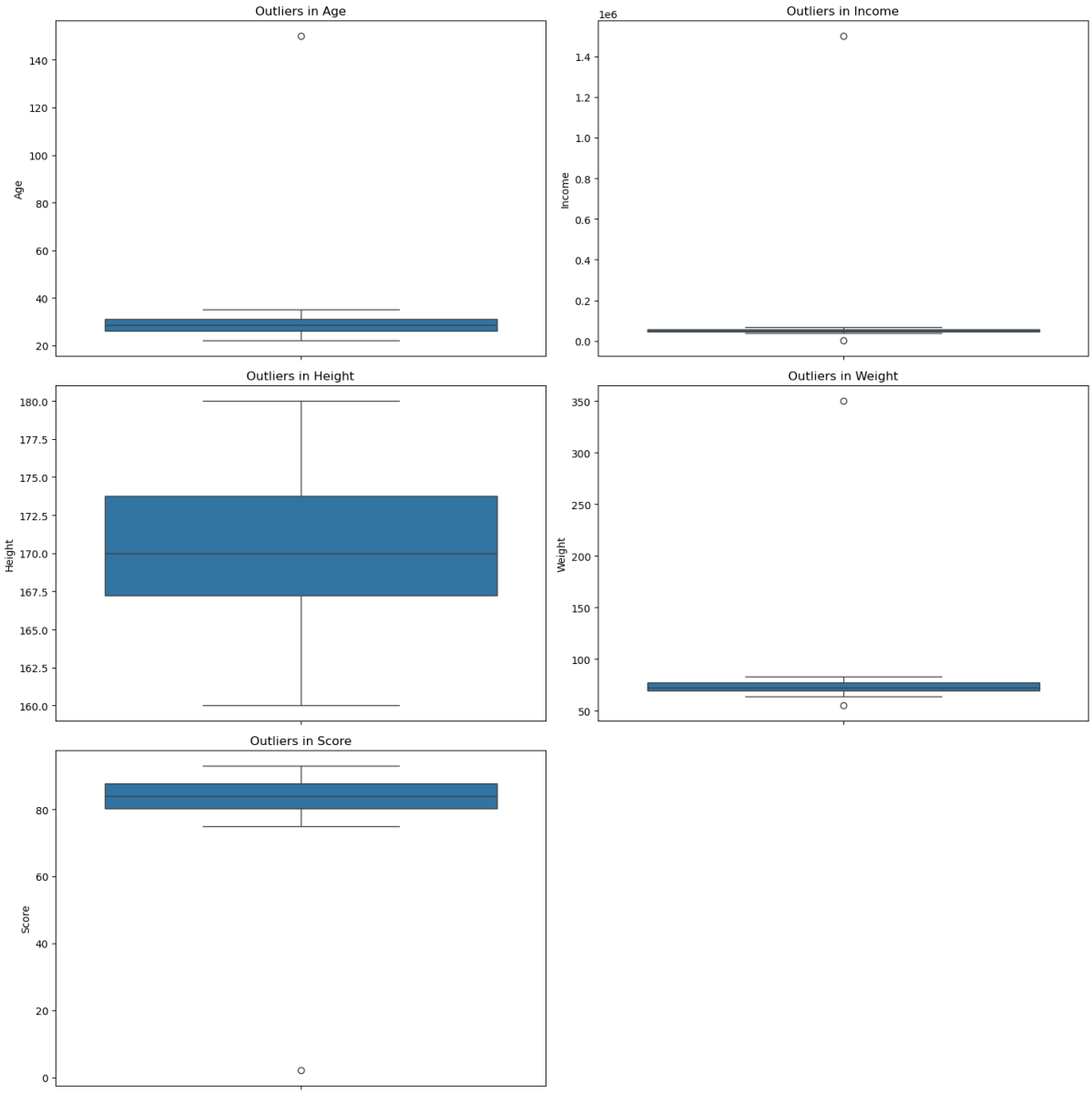
**Step 1: Select Appropriate Libraries**

```
# Import required libraries
import pandas as pd        # Data manipulation
import numpy as np         # Numerical operations
import matplotlib.pyplot as plt   # Visualization
import seaborn as sns      # Advanced visualization
from scipy import stats    # For Z-score calculation
```

**Step 2: Visualize Outliers (Boxplots Before Z-Score)**

```
# Select numeric columns only (Z-score works only on numeric data)
numeric_cols = df.select_dtypes(include=[np.number]).columns

# Set subplot layout
n_cols = 2
n_rows = int(np.ceil(len(numeric_cols) / n_cols))

plt.figure(figsize=(15, 5 * n_rows))

# Create boxplot for each numeric column
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(n_rows, n_cols, i)
    sns.boxplot(y=df[col])
    plt.title(f'Outliers in {col}')
    plt.ylabel(col)

plt.tight_layout()
plt.show()
```

***Output:***



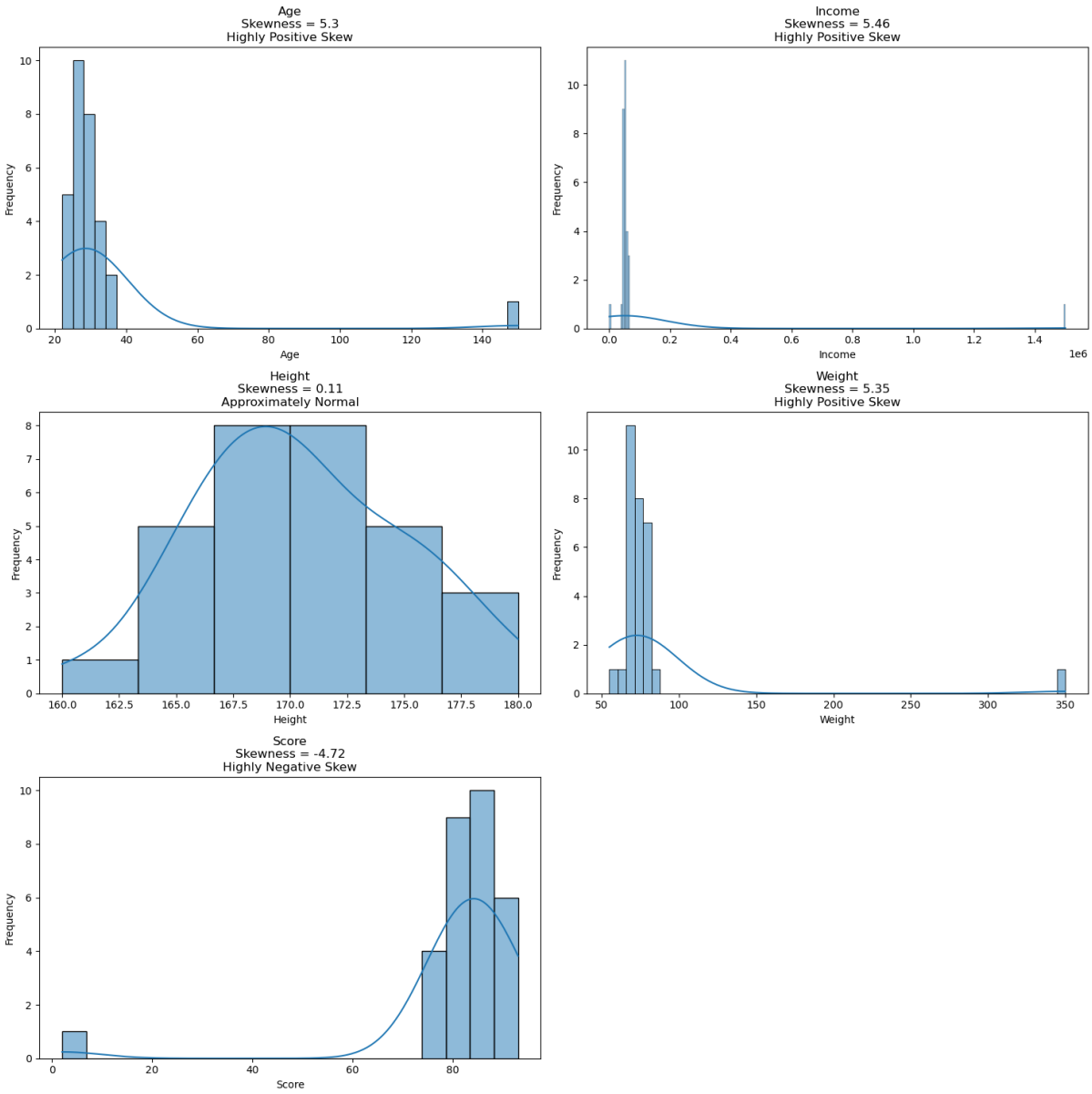

## Step 3: Distribution Skew or Normal

```
# STEP: Distribution Plot for ALL Numeric Columns
# This will:
# 1. Plot histogram + KDE curve
# 2. Calculate skewness
# 3. Automatically label distribution type inside each plot

# Select numeric columns
numeric_cols = df.select_dtypes(include=[np.number]).columns

# Set subplot layout (same logic as step 2)
n_cols = 2
```

```
n_rows = int(np.ceil(len(numeric_cols) / n_cols))

plt.figure(figsize=(15, 5 * n_rows))

# Loop through each numeric column
for i, col in enumerate(numeric_cols, 1):

    plt.subplot(n_rows, n_cols, i)

    # Plot histogram with KDE
    sns.histplot(df[col], kde=True)

    # Calculate skewness
    skew_value = df[col].skew()

    # Automatically classify distribution type
    if -0.5 <= skew_value <= 0.5:
        dist_type = "Approximately Normal"
    elif 0.5 < skew_value <= 1:
        dist_type = "Moderately Positive Skew"
    elif skew_value > 1:
        dist_type = "Highly Positive Skew"
    elif -1 <= skew_value < -0.5:
        dist_type = "Moderately Negative Skew"
    else:
        dist_type = "Highly Negative Skew"

    # Add title with skewness value
    plt.title(f"{col}\nSkewness = {round(skew_value,
              2)}\n{dist_type}")

    # Label axes
    plt.xlabel(col)
    plt.ylabel("Frequency")
```

```
plt.tight_layout()
plt.show()
```

***Output:***




## Step 4: Detect Outliers Using Z-Score Method

```
# Create a copy of numeric data
numeric_df = df[numeric_cols]

# Calculate Z-scores for each numeric column
z_scores = np.abs(stats.zscore(numeric_df))

# Convert to DataFrame (so column names remain visible)
```

```
z_scores_df = pd.DataFrame(z_scores, columns=numeric_cols)

# Create mask where True means outlier (Z > 3)
outlier_mask = z_scores_df > 3

# Extract rows where ANY column has outlier
outliers = df[outlier_mask.any(axis=1)]

# Display detected outliers
outliers
```

***Output:***

| | Age | Income | Height | Weight | Score |
|---|---|---|---|---|---|
| 10 | 150 | 60000 | 171 | 73 | 86 |
| 14 | 30 | 1500000 | 173 | 77 | 91 |
| 22 | 35 | 67000 | 180 | 350 | 90 |
| 28 | 33 | 63000 | 176 | 82 | 2 |

**Step 5: Validate and Document Result**

```
# Total number of outlier rows
print("Total Outlier Rows:", len(outliers))

# Percentage of dataset flagged as outlier
percentage = round((len(outliers) / len(df)) * 100, 2)
print("\nPercentage of Outliers:", percentage, "%")

# Column-wise outlier count
column_outliers = outlier_mask.sum()
print("\nColumn-wise Outlier Count:", column_outliers.to_dict())
```

***Output:***

```
Total Outlier Rows: 4
Percentage of Outliers: 13.33 %
Column-wise Outlier Count: {'Age': 1, 'Income': 1, 'Height': 0,
'Weight': 1, 'Score': 1}
```

**Step 6: Apply Z-Score Removal or Capping (ML Perspective)**

```
# Option A: Remove Outliers (Recommended if very few)

# Keep only rows where ALL columns have Z-score <= 3
df_clean = df[(z_scores_df <= 3).all(axis=1)]

# Reset index (important for ML models)
df_clean = df_clean.reset_index(drop=True)

print("Shape after removing outliers:", df_clean.shape)

# Replace original dataset (optional)
df = df_clean.copy()
```

***Output:***

```
Shape after removing outliers: (26, 5)
```

```
# Option B: Z-Score Capping (Winsorization using ±3)

# Create copy
df_capped = df.copy()

# Loop through each numeric column
for col in numeric_cols:

    # Calculate mean and std
    mean = df[col].mean()
    std = df[col].std()

    # Define boundaries
    upper_limit = mean + 3 * std
    lower_limit = mean - 3 * std

    # Cap values below lower limit
    df_capped[col] = np.where(
```

```
        df_capped[col] < lower_limit,
        lower_limit,
        df_capped[col]
    )

    # Cap values above upper limit
    df_capped[col] = np.where(
        df_capped[col] > upper_limit,
        upper_limit,
        df_capped[col]
    )

df_capped = df_capped.round(2)      # Round to 2 decimal places for cleaner output

# Replace original dataset (optional)
df = df_capped.copy()

df.shape
```

***Output:***

```
(30, 5)
```

### 1.10.3. K-Means Clustering

### 1.10.4. Isolation Forest