

In [3]:



Detecting and Handling Outliers

Lecture 02

Prepared by
Dr. Hager Saleh



What is an Outlier?

- An **outlier** is a data point that significantly differs from other values in a dataset. Outliers can be either much larger or much smaller than most of the other data points.

Example

[160, 162, 165, 168, 170, 163, 164, 166, 167, 300]

- In this dataset, most values are around 160-170 cm, but **300 cm** is an extreme value compared to the rest of the data. This value is likely an **outlier**.

Mean

- The **mean**, often referred to as the **average**, is a measure of central tendency that gives an idea of the typical value in a dataset.

$$\bar{x} = \frac{1}{n}(x_1 + x_2 + \dots + x_n) = \frac{1}{n} \sum_{i=1}^n x_i$$

Where

n is the sample size

Σ is the summation symbol

Example: Mean

1792 1666 1362 1614 1460 1867 1439

$$\begin{aligned}\bar{x} &= \frac{1792 + 1666 + 1362 + 1614 + 1460 + 1867 + 1439}{7} \\ &= \frac{11,200}{7} \\ &= 1600\end{aligned}$$

Median (M)

- The median is the middle value of a data set when the numbers are arranged in order.
- If the data set has an odd number of values, the median is the middle number.
- If the data set has an even number of values, the median is the average of the two middle numbers
- Example 1: 2 4 6

Median = 4

- Example 2: 2 4 6 8
Median = 5 (average of 4 and 6)

- Example 3: 6 2 4
Median $\neq$ 2

(Values must first be **ordered** first 2 4 6 , Median = 4)

Interquartile range (IQR)

- The **interquartile range (IQR)** is a measure of the spread of the middle 50% of the data. It is calculated as the difference between the **third quartile (Q3)** and the **first quartile (Q1)**.
 - **Q1 (First Quartile)**: The median of the lower half of the data (25th percentile).
 - **Q3 (Third Quartile)**: The median of the upper half of the data (75th percentile).
- **Formula:**

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

Steps to Calculate IQR:

1. Arrange the data in ascending order.
2. Split the data into lower and upper halves.
 1. If the number of data points is odd, exclude the median when splitting the data.
3. Find Q1 (the median of the lower half) and Q3 (the median of the upper half).
4. Subtract Q1 from Q3 to find the IQR.

Steps to Detect Outliers Using IQR

Data set: 3, 5, 7, 8, 12, 13, 14, 18, 19, 21

Step-by-Step Process:

Step 1: Arrange the data in ascending order: 3, 5, 7, 8, 12, 13, 14, 18, 19, 21

Step 2: Find the first quartile (Q1) and third quartile (Q3):

Q1 is the median of the lower half:

Lower half = 3, 5, 7, 8, 12

Q1 = 7 (middle value of the lower half).

Q3 is the median of the upper half:

Upper half = 13, 14, 18, 19, 21

Q3 = 18 (middle value of the upper half).

Step 3: Calculate the IQR = Q3-Q1=18-7=11

Step 4: Determine the outlier boundaries:

- Lower Bound: Lower Bound=Q1-1.5×IQR=7-1.5×11=7-16.5=-9.5
- Upper Bound: Upper Bound=Q3+1.5×IQR= 34.5

Step 5: Identify the outliers:

Any value less than -9.5 or greater than 34.5 is considered an outlier.

In this case, no values are below -9.5 or above 34.5, so there are no outliers.

z-score

The **z-score** is a statistical measure that tells how many standard deviations a data point is from the mean of the dataset. It is commonly used to detect **outliers**, which are data points that deviate significantly from the rest of the dataset.

Formula for the z-score:

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

- z is the z-score,
- x is the value of the data point,
- μ is the mean of the dataset,
- σ is the standard deviation of the dataset.

Steps to compute the z-score for outlier detection:

1. Calculate the mean of your dataset.
2. Calculate the standard deviation of your dataset.
3. Compute the z-score for each data point using the formula above.
4. Set a threshold: Common thresholds are |z|>3 or |z|>2.5, depending on how strict you want to be in identifying outliers.
 1. If |z|>3 the data point is considered an outlier.

Coding

```
In [2]: import numpy as np
import pandas as pd

df=pd.read_csv("Data-Melbourne_F_fixed.csv")
df.head()
```

Out[2]:

	Average Outflow	Average Inflow	Energy Consumption	Ammonia	Biological Oxygen Demand	Chemical Oxygen Demand	Total Nitrogen	Average Temperature
0	2.941	2.589	175856	27.0	365.0	730.0	60.378	19
1	2.936	2.961	181624	25.0	370.0	740.0	60.026	17
2	2.928	3.225	202016	42.0	418.0	836.0	64.522	16
3	2.928	3.354	207547	36.0	430.0	850.0	63.000	14
4	2.917	3.794	202824	46.0	508.0	1016.0	65.590	13

Common Methods to Detect Outliers

Using IQR (Interquartile Range):

The IQR method uses quartiles to determine the range of normal data. An outlier is typically defined as a point that lies below $Q1 - 1.5 * IQR$ or above $Q3 + 1.5 * IQR$.

```
In [3]: data=df.copy()
def detect_outliers_iqr(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Return a boolean series indicating if the value is an outlier
    return (column < lower_bound) | (column > upper_bound)

# Apply the IQR method to each feature in the DataFrame
for col in data.columns:
    data[f'{col}_Outlier'] = detect_outliers_iqr(data[col])

# Filter the DataFrame to display rows where any outlier flag is True
outliers_only = data[data.filter(like='_Outlier').any(axis=1)]

# Display the filtered DataFrame with outliers
outliers_only
```

Out[3]:

	Average Outflow	Average Inflow	Energy Consumption	Ammonia	Biological Oxygen Demand	Chemical Oxygen Demand	Total Nitrogen	Average Temperature
9	3.912	3.425	202411	42.0	530.0	830.0	66.354	
10	2.971	3.775	171123	47.0	435.0	870.0	65.531	
11	3.833	3.763	193509	43.0	295.0	590.0	59.548	
12	2.984	3.487	205761	38.0	600.0	1200.0	68.820	
15	2.935	3.481	173825	47.0	470.0	980.0	76.000	
...	...	...	...	...	...	...	...	...
1342	0.002	5.208	257362	49.0	330.0	780.0	62.160	
1358	5.502	4.453	285131	26.0	190.0	610.0	48.000	
1359	5.392	4.930	328307	24.0	200.0	560.0	53.295	
1362	4.846	4.219	343734	19.0	170.0	640.0	59.797	
1364	3.954	4.907	259208	33.0	180.0	720.0	61.190	

291 rows × 16 columns

Using Visualization:

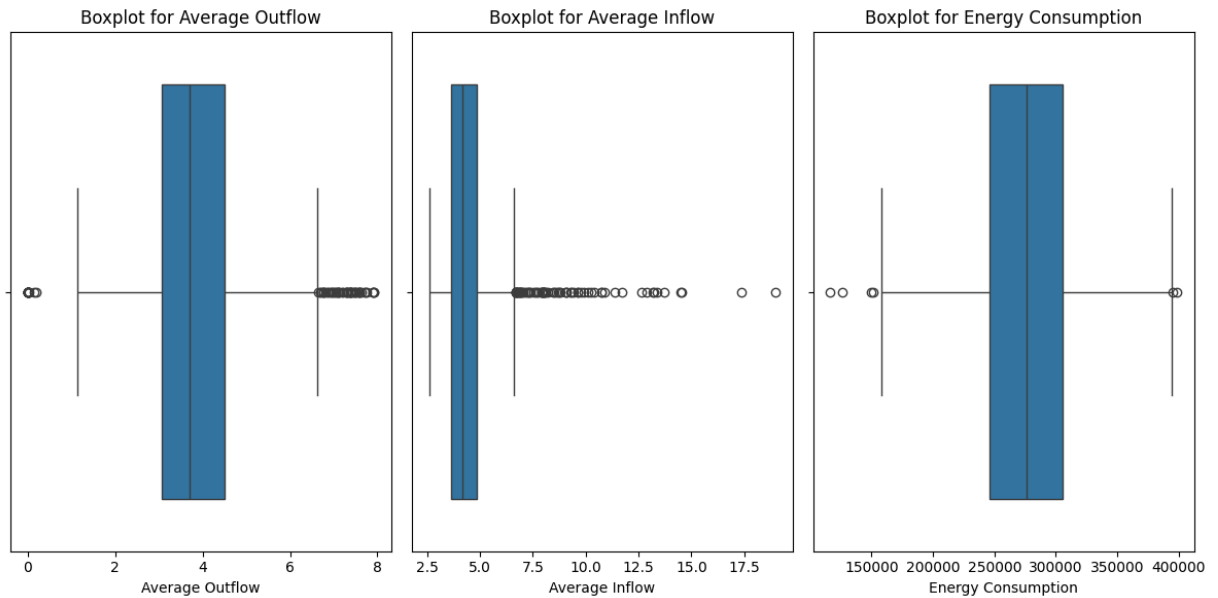
Visual methods like box plots and scatter plots can help in identifying outliers visually.

```
In [4]: import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 6))

for i, col in enumerate(df.columns[:3]): # Only plotting the first 3 features
    plt.subplot(1, 3, i+1) # Create subplots for each feature
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot for {col}")

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



Using Z-Score:

The Z-score measures how many standard deviations a data point is from the mean. A common threshold is a Z-score of ± 3 , which indicates that the data point is an outlier.

```
In [5]: # Function to detect outliers in a column using Z-Score
data=df.copy()
def detect_outliers_zscore(column, threshold=3):
    mean = np.mean(column)
    std_dev = np.std(column)
    z_scores = (column - mean) / std_dev
    # Return a boolean series where True indicates an outlier
    return np.abs(z_scores) > threshold

# Apply the Z-Score method to each feature in the DataFrame
for col in data.columns:
    data[f'{col}_Outlier'] = detect_outliers_zscore(data[col])

# Filter the DataFrame to display rows where any outlier flag is True
outliers_only = data[data.filter(like='_Outlier').any(axis=1)]

outliers_only
```

Out[5]:

	Average Outflow	Average Inflow	Energy Consumption	Ammonia	Biological Oxygen Demand	Chemical Oxygen Demand	Total Nitrogen	Av Temper
9	3.912000	3.425	202411	42.0	530.0	830.0	66.354	
10	2.971000	3.775	171123	47.0	435.0	870.0	65.531	
11	3.833000	3.763	193509	43.0	295.0	590.0	59.548	
15	2.935000	3.481	173825	47.0	470.0	980.0	76.000	
22	2.997000	3.526	214635	43.0	344.0	688.0	63.521	
...	...	...	...	...	...	...	...	
1320	0.000004	5.556	232271	35.0	410.0	1000.0	64.477	
1321	0.000004	5.299	263513	35.0	280.0	720.0	61.866	
1341	0.000004	6.372	269271	43.0	320.0	880.0	61.000	
1342	0.002000	5.208	257362	49.0	330.0	780.0	62.160	
1358	5.502000	4.453	285131	26.0	190.0	610.0	48.000	

127 rows × 16 columns

Handling Outliers

Replace outliers with NaN, then use methods for handling missing values to fill the NaN values.

```
In [6]: data=df.copy()
def replace_outliers_with_nan(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Replace outliers with NaN
    return column.where((column >= lower_bound) & (column <= upper_bound), np.nan)

# Replace outliers with NaN for each feature
for col in data.columns:
    data[col] = replace_outliers_with_nan(data[col])
```

```
In [7]: null_counts = data.isnull().sum()
print("Sum of null values in each column:")
print(null_counts)
```

Sum of null values in each column:

Average Outflow	74
Average Inflow	69
Energy Consumption	6
Ammonia	11
Biological Oxygen Demand	58
Chemical Oxygen Demand	44
Total Nitrogen	103
Average Temperature	8

dtype: int64

We use K-Nearest Neighbors (KNN) Imputation to handle missing values

```
In [8]: from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=2)
data_imputed_knn = pd.DataFrame(imputer.fit_transform(data), columns=data.columns)
data_imputed_knn.isnull().sum()
```

```
Out[8]: Average Outflow      0
Average Inflow      0
Energy Consumption   0
Ammonia              0
Biological Oxygen Demand  0
Chemical Oxygen Demand  0
Total Nitrogen       0
Average Temperature  0
dtype: int64
```

Remove outliers

```
In [9]: df=pd.read_csv("Data-Melbourne_F_fixed.csv")
```

```
Out[9]: 1382
```

Using the Interquartile Range (IQR) Method

```
In [10]: data=df.copy()

def remove_outliers_iqr(data):
    # Calculate Q1 (25th percentile) and Q3 (75th percentile) for each feature
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1

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

    # Filter the dataset to remove rows with outliers in any column
    data_cleaned = data[~((data < lower_bound) | (data > upper_bound)).any(axis=1)]

    return data_cleaned
```

```
cleaned_data = remove_outliers_iqr(data)

print("Original Data:")
print(len(data))
print("\nCleaned Data (without outliers):")
print(len(cleaned_data))
```

Original Data:
1382

Cleaned Data (without outliers):
1091

Using the Z-Score Method

```
In [15]: data = df.copy() # Create a copy to avoid modifying the original dataframe

def remove_outliers_zscore(data, threshold=3):

    # Boolean DataFrame indicating if a value is an outlier
    outliers = pd.DataFrame(False, index=df.index, columns=df.columns)

    # Iterate through each numeric column
    for column in data.columns:
        if data[column].dtype in ['float64', 'int64']: # Only apply to numeric columns
            mean = data[column].mean()
            std_dev = data[column].std()
            z_scores = (data[column] - mean) / std_dev

            # Mark True in outliers DataFrame where abs(z_score) > threshold
            outliers[column] = abs(z_scores) > threshold

    # Filter the rows that have any True in the outliers DataFrame (indicating an outlier)
    rows_to_keep = ~outliers.any(axis=1)

    # Return the DataFrame with outlier rows removed
    return data[rows_to_keep]

# Example usage:
cleaned_data = remove_outliers_zscore(data, threshold=3)
print("Original Data:")
print(len(data))
print("\nCleaned Data (without outliers):")
print(len(cleaned_data))
```

Original Data:
1382

Cleaned Data (without outliers):
1257

In []: