DEPARTMENT OF BIOMEDICAL ENGINEERING

24AI671- FUNDAMENTALS OF DATA SCIENCE

UNIT II

DATA COLLECTION, CLEANING AND PREPROCESSING

4. OUTLIER DETECTION

An **outlier** is a data point that **differs significantly from the other observations** in the dataset – it lies unusually far from the general pattern. For example, in a class where marks are between 40 and 95, a recorded mark of 950 (a typing mistake) or in a salary list where most people earn ₹ 20,000 – ₹ 60,000 a person earning ₹ 50 lakh are outliers.

Outlier detection is important because outliers can **distort the mean and standard deviation, mislead the regression line, reduce the accuracy of models** such as linear regression, KNN and K-means, and make scaling unreliable. At the same time, some outliers are **valuable** – they may represent frauds, machine faults or rare diseases, which is the whole purpose of **anomaly detection**.

How One Outlier Disturbs the Statistics?

Take the nine values 10, 12, 12, 13, 14, 15, 16, 17, 18. Their **mean is 14.1** and the **median is 14**. Now add one wrong entry 95. The mean jumps to **22.2** – higher than every one of the original nine values – while the median moves only slightly to **14.5**. This shows that the **mean and standard deviation are sensitive to outliers whereas the median and IQR are robust**. In a regression the effect is even stronger: a single far-away point can tilt the fitted line towards itself, and in K-Means one extreme point can pull a cluster centre away from the real group.

4.1 Causes of Outliers

- **Data entry or measurement errors** – wrong decimal point, sensor spikes, unit mix-up (cm vs m).
- **Data processing errors** – mistakes in merging or converting data.
- **Natural variation** – genuine extreme values such as a billionaire's income or a record-breaking flood.
- **Sampling error** – a sample accidentally includes people from another population.
- **Novelty or fraud** – unusual behaviour such as a stolen card used for a very high purchase.

4.2 Types of Outliers

Type	Description	Example
Point (global) outlier	A single value far from the whole data	Age = 210 in a list of ages
Contextual outlier	Normal in general but unusual in a particular context (time, place)	30 °C is normal in May but an outlier in January in Ooty
Collective outlier	A group of values that together are unusual, though each value may look normal	A sudden flat line of identical readings in a heartbeat signal

4.3 Methods of Outlier Detection

4.3.1 Visual methods

- **Box plot** – shows the median, quartiles and plots points beyond the whiskers as outliers (Fig 4.1).
- **Histogram** – isolated bars far from the main group indicate outliers.
- **Scatter plot** – useful for two variables; points away from the cloud are outliers (Fig 4.3).

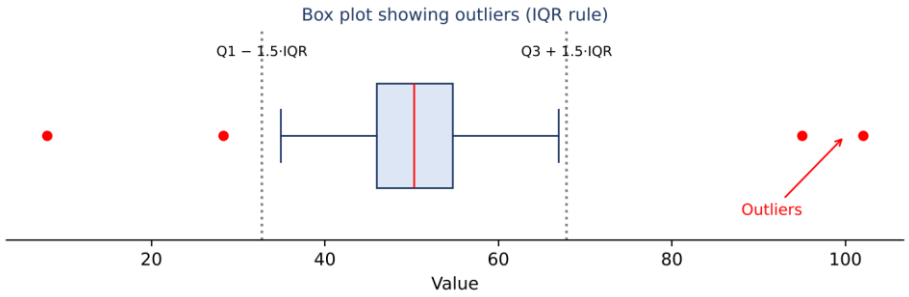

Fig 4.1 Box plot – points outside Q1 − 1.5·IQR and Q3 + 1.5·IQR are outliers

4.3.2 Z-score (standard score) method

The Z-score tells **how many standard deviations a value is away from the mean**. For data that is approximately **normally distributed**, about 99.7% of values lie within ±3 standard deviations, so a point with **|Z| > 3** (some people use 2.5 or 2) is treated as an outlier.

$$Z = (x - \mu) / \sigma$$

where x is the value, μ is the mean and σ is the standard deviation of the data.

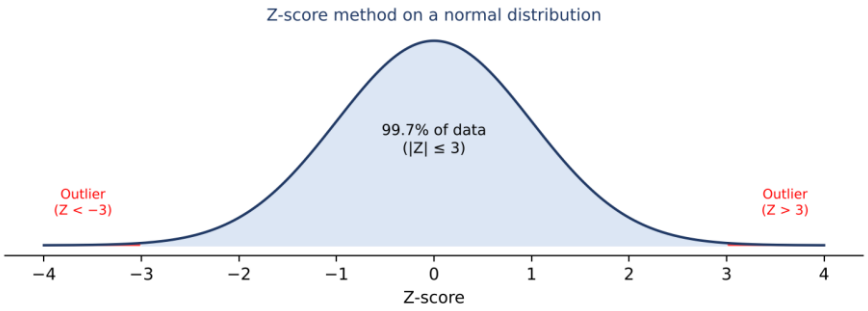

Fig 4.2 Z-score method – values beyond ±3 standard deviations are outliers

Limitation: the mean and standard deviation are themselves affected by the outliers (called the **masking effect**), and the method assumes normal data. Consider the data **10, 12, 12, 13, 14, 15, 16, 17, 18, 95**: mean = 22.2 and σ = 24.38, so for the value 95 the Z-score is (95 − 22.2) / 24.38 = **2.99**, which is just below 3 – the extreme value almost escapes detection because it inflated σ itself!

4.3.3 Modified Z-score (using median and MAD)

To avoid the masking effect, the **median** and the **Median Absolute Deviation (MAD)** are used because they are **robust** to outliers. A value with a modified Z-score greater than **3.5** is considered an outlier.

$$\text{Modified } Z = 0.6745 \times (x - \text{median}) / \text{MAD}, \quad \text{MAD} = \text{median}(| x_i - \text{median} |)$$

For the same data: median = 14.5, MAD = 2.5, so for 95 the modified Z = 0.6745 × (95 − 14.5) / 2.5 = **21.7**, clearly an outlier.

Understanding quartiles. Quartiles divide sorted data into four equal parts. **Q1 (25th percentile)** is the value below which 25% of the data lies, **Q2 is the median (50%)**, and **Q3 (75th percentile)** has 75% of the data below it. The middle half of the data lies between Q1 and Q3, and its width is the **IQR**. In a box plot, the box spans Q1 to Q3, the line inside is the median, and the 'whiskers' extend to the last data points inside the fences; points outside are drawn individually as outliers.

4.3.4 IQR (Inter-Quartile Range) method

The IQR method does not assume a normal distribution and is the rule used by box plots. Arrange the data in ascending order and find the **first quartile Q1 (25th percentile)**, **third quartile Q3 (75th percentile)** and **IQR = Q3 − Q1**. Any value **below the lower fence or above the upper fence** is an outlier.

Lower fence = Q1 − 1.5 × IQR	**Upper fence = Q3 + 1.5 × IQR**

Worked example: Data (sorted): 10, 12, 12, 13, 14, 15, 16, 17, 18, 95 (n = 10).

Step	Calculation	Result
1. Q1 (median of lower half 10,12,12,13,14)	middle value	12
2. Q3 (median of upper half 15,16,17,18,95)	middle value	17
3. IQR	17 − 12	5
4. Lower fence	12 − 1.5 × 5	4.5
5. Upper fence	17 + 1.5 × 5	24.5
6. Decision	95 > 24.5	95 is an outlier

Why the factor 1.5? John Tukey, who invented the box plot, chose 1.5 as a practical compromise. For normally distributed data the fences Q1 − 1.5·IQR and Q3 + 1.5·IQR are at about **±2.7 standard deviations** from the mean, so roughly **99.3%** of normal data falls inside them. A smaller factor

would flag too many ordinary points, while a larger factor (3 × IQR is used for ‘extreme’ outliers) would miss real ones.

Point	Z-score method	IQR method
Uses	Mean and standard deviation	Quartiles (median based)
Assumption	Roughly normal data	None – works for skewed data
Sensitive to the outliers themselves?	Yes (masking effect)	No – robust
Typical rule	$\|Z\| > 3$	Outside Q1 − 1.5·IQR … Q3 + 1.5·IQR
Best for	Large, bell-shaped data	Small or skewed data; quick screening

4.3.5 Percentile method

Values below a chosen low percentile (e.g. 1st) or above a high percentile (e.g. 99th) are treated as outliers. It is simple and often used for **winsorising** (capping).

4.3.6 Machine-learning based methods

Method	Idea	Remark
Isolation Forest	Builds random trees; outliers are separated (isolated) in very few splits	Fast, works for many features, no distribution assumption
DBSCAN	Density-based clustering; points not belonging to any dense cluster are noise	Finds arbitrary-shaped clusters; needs eps and min_samples
LOF (Local Outlier Factor)	Compares the local density of a point with that of its neighbours	Finds local outliers; sensitive to K
K-Means distance	Points far from every cluster centre are outliers	Assumes round clusters

Method	Idea	Remark
One-Class SVM	Learns a boundary around normal data	Needs careful tuning
Autoencoders	Neural network that fails to rebuild abnormal data (high reconstruction error)	For complex, high-dimensional data

How Isolation Forest works. The algorithm builds many random trees. In each tree, a random feature and a random split value are chosen repeatedly until every point is alone in its own leaf. A normal point lies in a crowded region and needs **many splits** to be isolated, whereas an outlier lies far away and is isolated after **very few splits**. The average path length over all trees is converted to an **anomaly score** – a short path means a high chance of being an outlier. The parameter contamination states the expected fraction of outliers.

How DBSCAN works. DBSCAN needs two parameters: **eps** (radius of the neighbourhood) and **min_samples**. A point with at least min_samples points inside its eps-radius is a **core point**; points inside the radius of a core point but not themselves core are **border points**; every other point is **noise (outlier)** and is labelled −1. It does not need the number of clusters in advance.

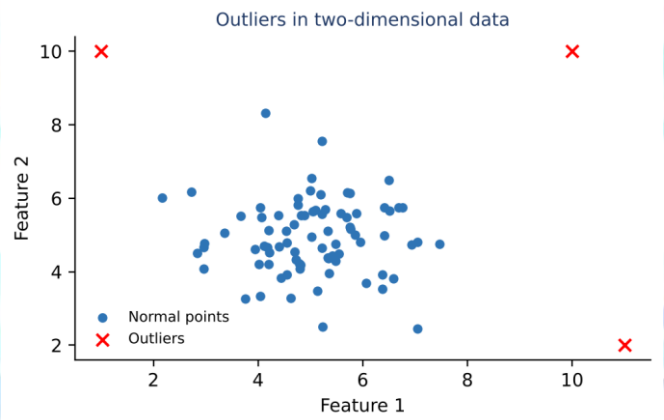

Fig 4.3 Outliers in two-dimensional data

4.4 Python Implementation

Code

```
import numpy as np, pandas as pd
x = pd.Series([10, 12, 12, 13, 14, 15, 16, 17, 18, 95])

# ---- IQR method ----
q1, q3 = x.quantile(0.25), x.quantile(0.75)
iqr = q3 - q1
low, high = q1 - 1.5 * iqr, q3 + 1.5 * iqr
print("fences:", low, high)
print("outliers:", x[(x < low) | (x > high)].tolist())

# ---- Z-score method ----
z = (x - x.mean()) / x.std(ddof=0)
print("z of 95:", round(z.iloc[-1], 2))

# ---- Modified Z-score ----
med = x.median(); mad = (x - med).abs().median()
mz = 0.6745 * (x - med) / mad
print("modified z of 95:", round(mz.iloc[-1], 2))
```

Output

```
fences: 5.5 23.5
outliers: [95]
z of 95: 2.99
modified z of 95: 21.72
```

Isolation Forest

Code

```
from sklearn.ensemble import IsolationForest

X = x.to_numpy().reshape(-1, 1)              # sklearn needs a 2-D array
iso = IsolationForest(contamination=0.1, random_state=42)
labels = iso.fit_predict(X)                  # 1 = normal, -1 = outlier
```

```
print(labels.tolist())
```

Output

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, -1]
```

Should an outlier be removed? – a simple guide

1. **Is it an obvious error?** (age = 210, negative height) → correct it, or remove / impute it.
2. **Is it genuine but rare?** (a very rich customer, a real fault) → keep it; use robust methods, transformation or analyse it separately.
3. **Does the model suffer?** Compare the model accuracy with and without the outlier before deciding.
4. **Always record** what was removed or changed, and why.

4.5 Treating Outliers

Detection is followed by a decision. **Never remove an outlier automatically** – first find out whether it is an error or a genuine value.

Treatment	Description	When to use
Remove (trimming)	Delete the record	Clear data-entry error; large dataset
Capping / Winsorising	Replace values beyond the fences by the fence value (clip)	Keep the record but reduce its influence
Transformation	Apply log, square-root or Box-Cox to compress large values	Right-skewed data (income, population)
Imputation	Treat as missing and replace with mean / median	Value is definitely wrong
Separate model	Analyse outliers separately	They are frauds, faults or special cases
Use robust methods	Median, MAD, Huber regression, tree-based models	Outliers are genuine and must stay

Code

```
# Capping (winsorising) using the IQR fences
x_capped = x.clip(lower=low, upper=high)
print(x_capped.tolist())

# Log transform to reduce the effect of large values
x_log = np.log1p(x)       # log(1 + x), safe for zero
```

Output

```
[10.0, 12.0, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 23.5]
```

Outlier vs noise vs anomaly

Term	Meaning
Outlier	A data point far from the rest; may be an error or a genuine extreme
Noise	Random error spread over many values (next section) – not a single extreme point
Anomaly	An outlier that is meaningful – fraud, intrusion, machine fault; the goal of anomaly detection

5. NOISE REDUCTION

Noise is **random error or meaningless variation in the data** that hides the true pattern. If a temperature sensor shows 30.1, 29.8, 30.4, 29.9 and 30.2 °C when the real temperature is steady at 30 °C, the small random differences are noise. A model trained on noisy data may learn the noise instead of the real pattern (**overfitting**), so noise must be reduced during pre-processing.

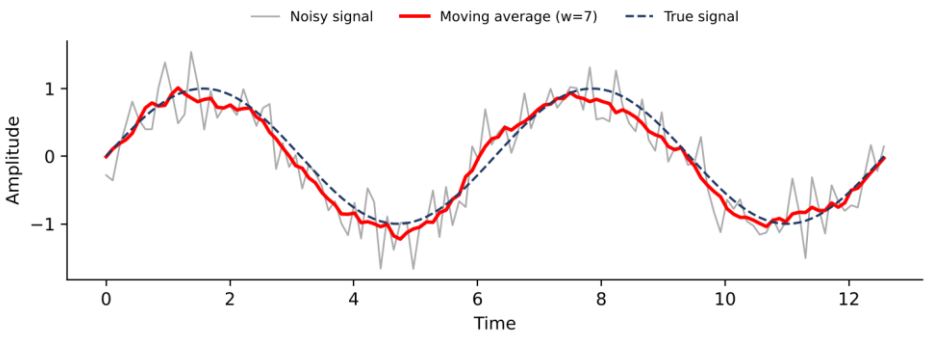

Fig 5.1 Noise reduction – a moving average (red) recovers the true signal (dashed) from a noisy signal (grey)

Understanding Noise – Signal and Noise

Every measurement can be written as **observed value = true value (signal) + noise**. The signal is the real pattern we want to find – for instance the actual temperature of a furnace – and the noise is the unwanted random variation added by the sensor, the environment or the human recorder. A model that is trained on noisy data may try to explain even the random wiggles, which is **overfitting**: high accuracy on the training data but poor results on new data. Noise reduction therefore tries to **increase the signal-to-noise ratio** without destroying the real pattern.

The central difficulty is the **trade-off**: too little smoothing leaves the noise, while too much smoothing removes real changes and important peaks. The right amount depends on the application and is usually found by experiment and by looking at plots.

5.1 Sources and Types of Noise

Type of noise	Description	Example
Random (measurement) noise	Small unpredictable fluctuations in readings	Electrical interference in a sensor
Systematic noise (bias)	Constant error in one direction	A thermometer always reads 2 °C high
Attribute noise	Wrong or inconsistent values in feature columns	Typing mistakes, ‘Chenai’ for ‘Chennai’
Class (label) noise	Wrong target labels	A spam mail labelled ‘not spam’

Type of noise	Description	Example
Duplicate and irrelevant data	Repeated records and unnecessary columns	Same customer entered twice
Format noise	Inconsistent formats and units	'12/03/26' vs '2026-03-12'; kg vs g

Noise vs outliers

Point	Noise	Outlier
Nature	Small random error spread over many values	A few values that are very far from the rest
Effect	Blurs the pattern and reduces accuracy	Pulls the mean and model line towards itself
Remedy	Smoothing, filtering, averaging	Detect and remove, cap or transform

5.2 Techniques for Noise Reduction

5.2.1 Binning (smoothing by bins)

In **binning**, the sorted data is divided into small groups (**bins**) and each value is replaced by a value representing its bin – thus the data is made smooth by consulting its neighbours. Bins may be **equal-width** (same range) or **equal-frequency** (same number of values).

Example: Sorted data: 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34. Using equal-frequency bins of size 4:

Bin	Original values	Smoothing by means	Smoothing by medians	Smoothing by boundaries
Bin 1	4, 8, 9, 15	9, 9, 9, 9	8.5, 8.5, 8.5, 8.5	4, 4, 4, 15
Bin 2	21, 21, 24, 25	22.75 (all four)	22.5 (all four)	21, 21, 25, 25
Bin 3	26, 28, 29, 34	29.25 (all four)	28.5 (all four)	26, 26, 26, 34

In **smoothing by boundaries**, each value is replaced by the nearer of the two bin boundaries (minimum or maximum of that bin).

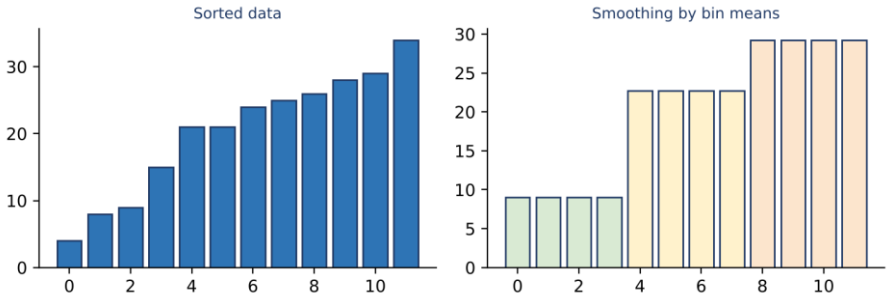

Fig 5.2 Smoothing by bin means (bins coloured)

Code

```
import pandas as pd
v = pd.Series([4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34])
bins = pd.qcut(v, q=3, labels=False)            # 3 equal-frequency bins
smooth = v.groupby(bins).transform("mean")      # replace by bin mean
print(smooth.round(2).tolist())
```

Output

```
[9.0, 9.0, 9.0, 9.0, 22.75, 22.75, 22.75, 22.75, 29.25, 29.25, 29.25, 29.25]
```

Choosing the bin method and size. Equal-width bins are easy to understand but can leave some bins almost empty when the data is skewed; equal-frequency bins keep the same number of values in every bin and are usually preferred. Smoothing by **means** gives the smoothest result, by **medians** is less affected by extreme values, and by **boundaries** keeps the minimum and maximum of each bin and therefore preserves the range of the data. Fewer, larger bins give stronger smoothing but lose more detail.

5.2.2 Moving average (rolling mean)

Each value is replaced by the **average of the values in a window** around it. It is the simplest smoothing method for time-series data. A larger window gives a smoother curve but reacts more slowly to real changes.

$$\mathbf{MA_t = (x_t + x_{t-1} + \ldots + x_{t-k+1}) / k \qquad (k = window\ size)}$$

Example: readings 5, 7, 6, 9, 8 with window k = 3 give (5+7+6)/3 = **6.00**, (7+6+9)/3 = **7.33** and (6+9+8)/3 = **7.67**.

Choosing the window size. In a moving average, a small window (e.g. 3) follows the original data closely and removes only slight noise, while a large window (e.g. 15) gives a very smooth line but reacts late to real changes and flattens peaks. A centred window (values on both sides) avoids the time delay, but it can only be used when future values are already available, i.e. in offline analysis. For real-time streaming, a **trailing window** (only past values) or exponential smoothing is used.

5.2.3 Exponential smoothing

Gives **more weight to recent readings** and exponentially smaller weights to older ones. The smoothing factor α ($0 < \alpha \leq 1$) controls the memory: large α follows the data closely, small α smooths more.

$$S_t = \alpha \cdot x_t + (1 - \alpha) \cdot S_{t-1}, \quad S_1 = x_1$$

5.2.4 Median filter

Replaces each value by the **median of its neighbours**. It removes sudden spikes (impulse or 'salt-and-pepper' noise) while keeping edges sharp, which a moving average cannot do. It is widely used in image processing and sensor data.

Code

```
import pandas as pd
from scipy.signal import medfilt

v = pd.Series([5, 7, 6, 9, 8, 12, 10])
# moving average
print(v.rolling(3, center=True, min_periods=1).mean().round(2).tolist())
print(v.ewm(alpha=0.5, adjust=False).mean().round(2).tolist())  # exponential
# median filter
print(medfilt([5, 7, 40, 9, 8, 12, 10], kernel_size=3).tolist())
```

Output

```
[6.0, 6.0, 7.33, 7.67, 9.67, 10.0, 11.0]
[5.0, 6.0, 6.0, 7.5, 7.75, 9.88, 9.94]
[5, 7, 9, 9, 9, 10, 10]
```

In the third line the spike '40' is completely removed by the median filter.

5.2.5 Regression and curve-fitting smoothing

A regression line or polynomial curve is fitted to the data and the **fitted values replace the noisy values**. Techniques such as **LOWESS** and the **Savitzky–Golay filter** (scipy.signal.savgol_filter) fit small local polynomials and preserve the shape of peaks better than a plain moving average.

5.2.6 Frequency-domain filtering

For signals such as vibration, ECG or audio, noise is often at high frequencies. A **low-pass filter** (e.g. Butterworth) removes the high-frequency part, while a **band-pass** or **notch filter** removes only a chosen band such as 50 Hz mains hum. The Fourier Transform is used to find which frequencies contain the noise.

5.2.7 Clustering and outlier removal

Values that do not fall into any cluster (e.g. in DBSCAN) are treated as noise and removed. Outlier-detection methods of Section 4 are therefore also noise-removal tools.

5.2.8 Dimensionality reduction (PCA)

Principal Component Analysis keeps the components with large variance and discards those with small variance, which are often noise. Reconstructing the data from the main components gives a **de-noised** version of the data.

5.2.9 Ensemble methods and regularisation

Averaging many models (bagging, Random Forest) reduces the effect of noise on predictions, and **L1 / L2 regularisation** prevents a model from fitting noise. For **label noise**, relabelling by experts, majority voting among several annotators or removing suspicious samples is used.

5.3 Cleaning Noisy Records (Inconsistent and Duplicate Data)

Not all noise is numerical. Text and categorical columns need cleaning too:

- **Remove duplicates** – drop_duplicates().
- **Standardise text** – remove extra spaces, make the case uniform, correct spelling (str.strip(), str.lower(), str.title(), replace()).
- **Standardise formats and units** – convert dates using pd.to_datetime(), convert all weights to kg.
- **Fix data types** – convert numbers stored as text using astype() or pd.to_numeric(errors="coerce").

- **Validate ranges** – e.g. marks between 0 and 100; age between 0 and 120.

Code

```
d = pd.DataFrame({"name": [" Asha", "asha ", "RAVI", "Ravi"],
                  "age":  [21, 21, 23, 23]})
d["name"] = d["name"].str.strip().str.title()     # ' Asha','asha ' -> 'Asha'
print(d.drop_duplicates())                        # remove repeated records
```

Output

```
   name  age
0  Asha   21
2  Ravi   23
```

Summary of noise-reduction techniques

Technique	Best for	Advantage	Limitation
Binning	Numeric attribute values	Simple and quick	Loses detail
Moving average	Time-series trends	Easy to implement	Lags; blurs sharp changes
Exponential smoothing	Streaming / forecasting	Recent data gets more weight	Needs choice of α
Median filter	Spikes, images	Preserves edges	Slower for large windows
Regression / Savitzky–Golay	Smooth curves	Preserves peaks	Assumes a model shape
Low-pass / band filters	Signals (ECG, vibration)	Removes specific frequencies	Needs signal-processing knowledge
PCA	High-dimensional data	Removes low-variance noise	Components are less interpretable