Intelligent techniques for processing large and structured data

Lecture 7

Faculty of Mathematics and Computer Science
Babeș-Bolyai University

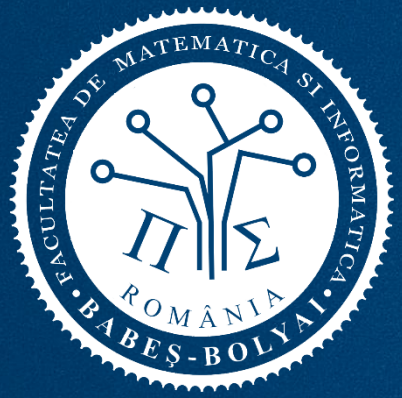
Sergiu Limboi, PhD Teaching Assistant

Motto:“In big data, the rare event is the most valuable signal.”

Anomaly Detection at Scale: Finding Rare Signals in Massive and Streaming Data

AGENDA

- Warm-Up
- Real-World Motivation
- What is an Anomaly?
- Types of Anomalies
- Anomaly Detection Methods
- Teamwork Time
- Scaling Anomaly Detection
- Key Takeaways



Warm-Up

Faculty of Mathematics and Computer Science

Warm-Up

Go to www.menti.com and enter the code **2658 9187**

or use the QR code

Real-World Motivation

Real-World Motivation

- We are interested in **rare, strange, unexpected events**

Think about this scenario:

Your bank processes 500 million transactions per day.
Only 0.01% are fraudulent.

How would you find those 50,000 fraudulent transactions
without manually reviewing all 500 million?

Real-World Motivation

Finance

Credit card fraud

Money laundering

Insider trading

Cybersecurity

Intrusion detection

DDoS attacks

Data exfiltration

Manufacturing

Equipment failure

Quality control

Sensor monitoring

Anomaly detection is one of the highest-value applications in data mining.

The Cost of NOT Detecting Anomalies

$32 billion

Global card fraud losses (2024)

$4.88 million

Average cost of a data breach (2024)

$1.5 trillion

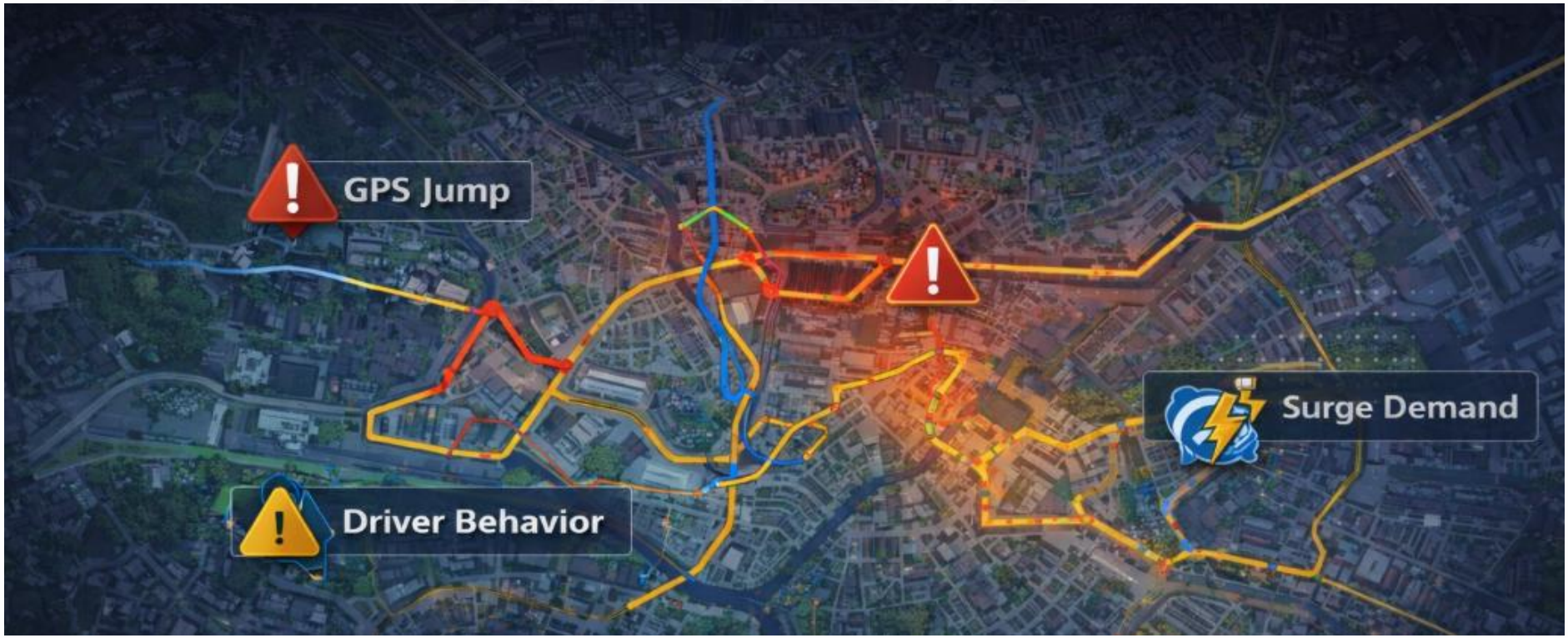
Annual cost of equipment failures (US)

272 days

Avg. time to identify a breach

Uber Anomaly Detection

Uber continuously analyses GPS streams from drivers and users to detect anomalies in real time — because anomalies directly impact pricing, safety, and user experience



Uber Anomaly Detection

- Examples of anomalies:
  - A driver appears suddenly far away from previous location
  - GPS jump: Cluj → Bucharest in 2 minutes
  - A ride that usually takes 10 minutes suddenly takes 40
  - A driver cancels 90% of rides
  - Suddenly 500 people request rides in 1 area

Uber Anomaly Detection

- Why anomaly detection is critical for Uber?
  - Business
    - Detect demand spikes → adjust pricing
    - Avoid revenue loss
  - Security
    - Detect fake drivers or accounts
    - Prevent fraud
  - Operations
    - Improve routing
    - Identify traffic issues

Industry/IoT Anomaly Detection

- In modern factories, machines are equipped with **sensors** that continuously send data:
  - 🌡️ Temperature
  - 🔊 Vibration
  - ⚡ Energy consumption
  - 🔄 Rotation speed (RPM)
  - 🛢️ Pressure

- These sensors generate **data streams (real-time)**

Industry/IoT Anomaly Detection

- Examples of anomalies:
  - Temperature suddenly jumps from 70°C → 120°C
  - 90°C may be normal during production, but abnormal at night (machine should be idle)
  - Vibration gradually increases over 2 days
  - Machine consumes ~100 kWh per hour. Sudden increase → 180 kWh
  - Pipeline pressure should be stable. Sudden drop

Industry/IoT Anomaly Detection

- Why anomaly detection is critical?
  - Cost impact
    - Machine failure = production stops
    - Can cost millions per hour
  - Maintenance optimization
    - Instead of: Fixed schedule
    - We do: Predictive maintenance

What is an Anomaly?

What is an Anomaly?

An anomaly (or outlier) is a data point that deviates significantly from the expected behaviour of the majority of the data.

Key insight: "Normal" must be defined before "abnormal" can be detected.

Context matters:

A €5,000 purchase is normal for a CEO, anomalous for a junior.

A temperature of 35°C is normal in July, anomalous in January.

100 login attempts/min is normal for a server, anomalous for a user.

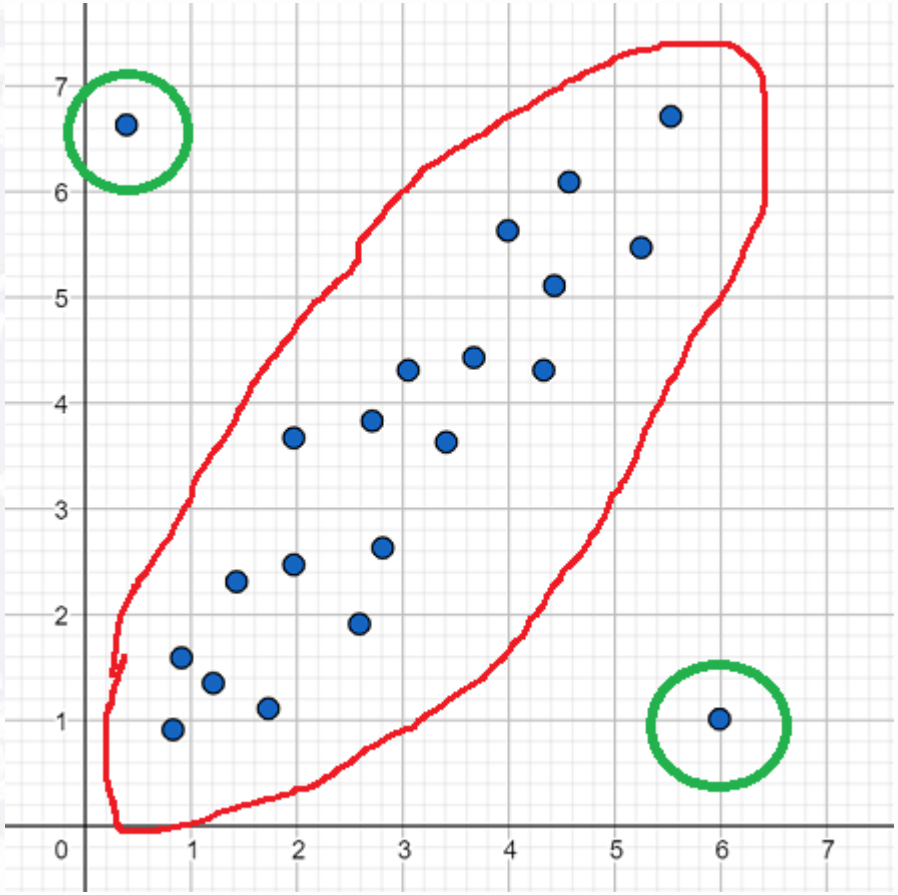
What is an Anomaly?

- A point that is:
  - rare
  - different
  - unexpected

- Clusters = groups of similar points
- Anomalies = points that do NOT belong to any cluster

What is an Anomaly?

- Normal data → dense region
- Anomalies → isolated points



Why Anomaly Detection is Hard?

- No labels
- Imbalanced data (0.1% anomalies)
- Context matters
- Noise vs anomaly confusion

- Example:
  - typo in data → noise
  - fraud → anomaly

Why Anomaly Detection is Hard in Large Data?

- Rarity
  - anomalies = <1% of data
- Imbalance problem
  - 1 fraud in 10,000 transactions
- Noise vs anomaly
  - Not all weird = anomaly
- Scale
  - billions of events → cannot check manually
- High dimensionality
  - many features → distance becomes tricky

Anomaly vs. Noise vs. Novelty

Concept	Definition	Action	Example
Anomaly	Meaningful deviation from expected pattern	Investigate & act	Fraudulent transaction
Noise	Random error in data	Filter & ignore	Sensor measurement jitter
Novelty	New, but legitimate pattern	Learn & adapt model	New purchasing behaviour

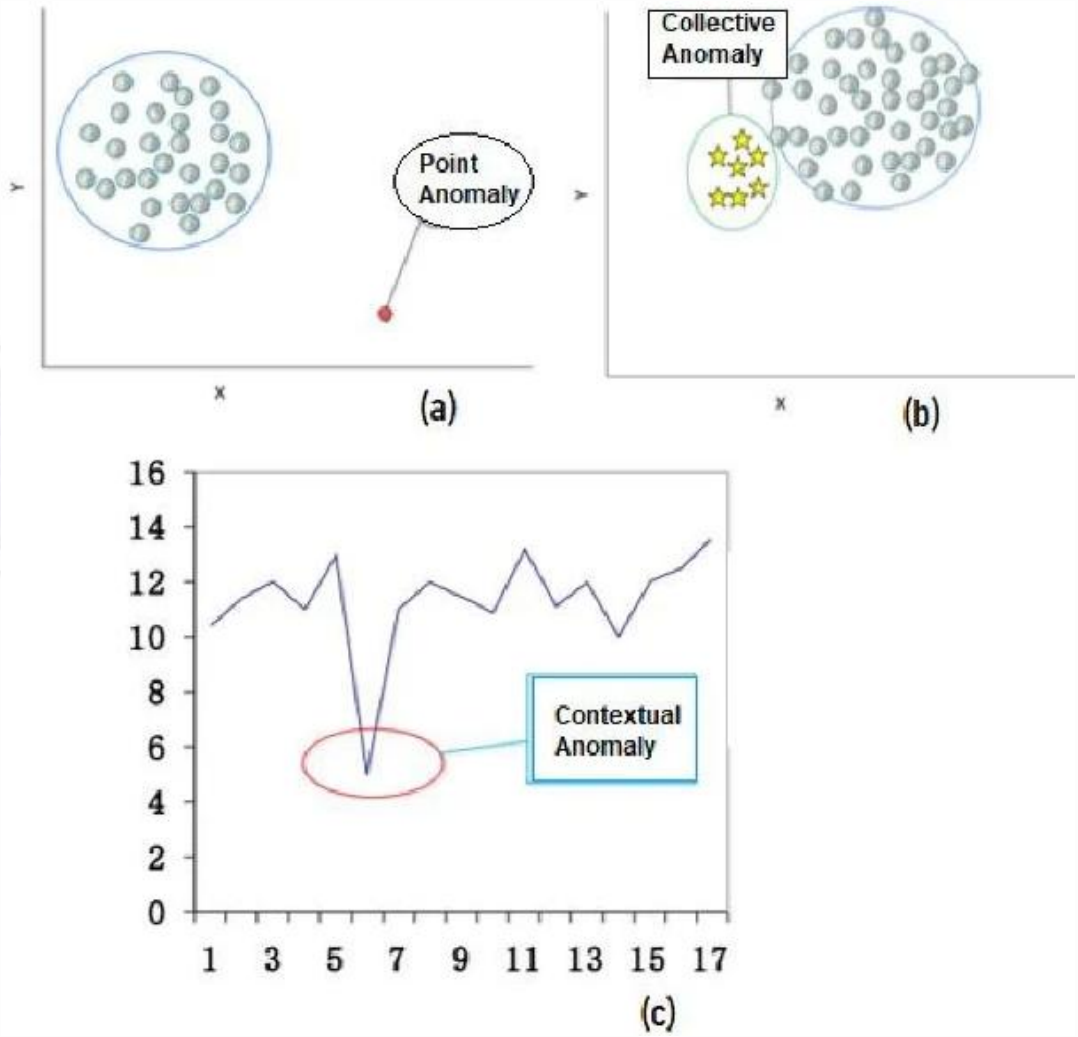
The hardest part of anomaly detection: distinguishing real anomalies from noise.

Types of Anomalies

Types of Anomalies

Point Anomaly	Contextual Anomaly	Collective Anomaly
A single data point that is very different from the rest. Example: A transaction of €50,000 when the average is €100.	Abnormal in a specific context, normal otherwise. Example: High AC usage in winter (normal in summer).	A group of points that are anomalous together. Example: 100 small transfers to the same account in 1 hour.

Types of Anomalies



Anomaly Detection Methods

Anomaly Detection Methods

- Most anomaly detection methods rely on one core idea:

Normal points are close to their neighbors. Anomalies are far away.

This requires measuring:

1. Distance — how far apart are two points?
2. Density — how many neighbours does a point have nearby?
3. Isolation — how easy is it to separate a point from others?

Method 1: Statistical — Z-Score

- Assume data follows a distribution (usually Gaussian)
  - most values near mean
  - anomalies far from mean

- We compute:
  - mean (μ)
  - standard deviation (σ)

Idea: measure how many standard deviations a point is from the mean.

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

If |Z| > 3 → likely anomaly (assuming normal distribution)

Advantages:

Simple, fast, interpretable

Limitations:

Assumes normal distribution

Only works for univariate data (one feature at a time)

Fails with multimodal data or complex patterns

Method 1: Statistical — Z-Score

- Dataset: student exam scores [70,75, 80, 85, 90, 95, 30]
- Mean = 75
- 30 is far anomaly

```
import numpy as np

data = np.array([70, 75, 80, 85, 90, 95, 30])

mean = np.mean(data)
std = np.std(data)

z_scores = (data - mean) / std

print(z_scores)
```

[-0.25 0. 0.25 0.5 0.75 1. -2.25]

- Large absolute z-score → anomaly
- Here |-2.25| = 2.25 → 30 is, indeed, the anomaly

Method 1: Statistical — IQR Method

- It measures how spread out the “middle” of the data is
- IQR = Interquartile Range
- Instead of using mean (which is sensitive to outliers),we use quartiles (robust to noise)
- We split the data into 4 parts:
  - **Q1 (25%)** → lower quartile
  - **Q2 (50%)** → median
  - **Q3 (75%)** → upper quartile

```
IQR = Q3 - Q1
Lower bound = Q1 - 1.5 × IQR
Upper bound = Q3 + 1.5 × IQR
```

Anything outside the bounds is flagged as an anomaly.

Advantage: Works with skewed data, no distribution assumption.

Limitation: Still univariate. Cannot capture complex relationships between features.

Method 2: Tree-based- Isolation Forest

Anomalies are few and different → they are easier to isolate.

How it works:

1. Randomly select a feature
2. Randomly select a split value between min and max of that feature
3. Divide data into two groups
4. Repeat until every point is isolated

This builds a random tree

Key insight: Anomalies need FEWER splits to be isolated.

Normal points need MANY splits because they are surrounded by similar points.

Method 2: Isolation Forest- Anomaly Score

Build an ensemble of random isolation trees (forest).

For each point:

Measure the average path length across all trees.

Short path → Easy to isolate → **Anomaly**

Long path → Hard to isolate → **Normal**

Imagine:

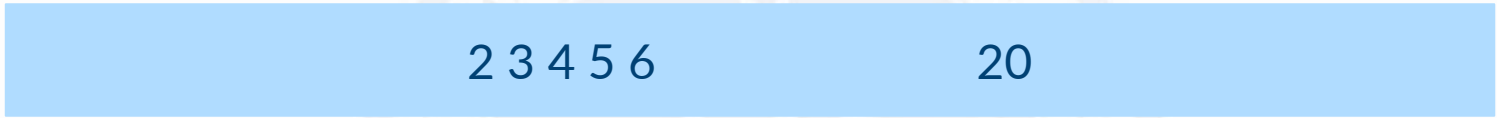
100 normal points clustered
1 anomaly far away

Random splits:

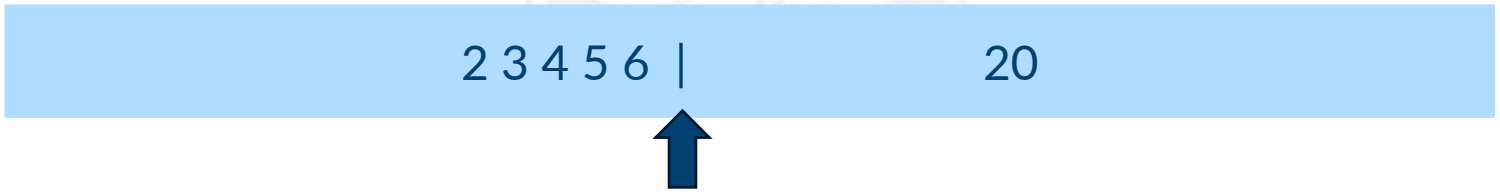
anomalies get separated very quickly
normal → many splits

Method 2: Isolation Forest

- Step 1: start with all data
  - Dataset [2,3,4,5,6,20]

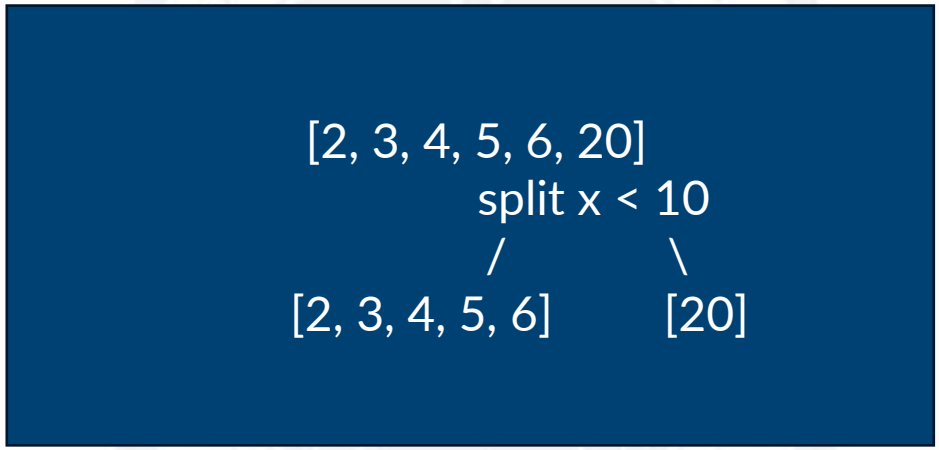


- Step 2: choose a random split
  - Let's say the algorithm randomly chooses x < 10
  - Then the data is divided into
    - Left side: [2,3,4,5,6]
    - Right side: [20]



Method 2: Isolation Forest

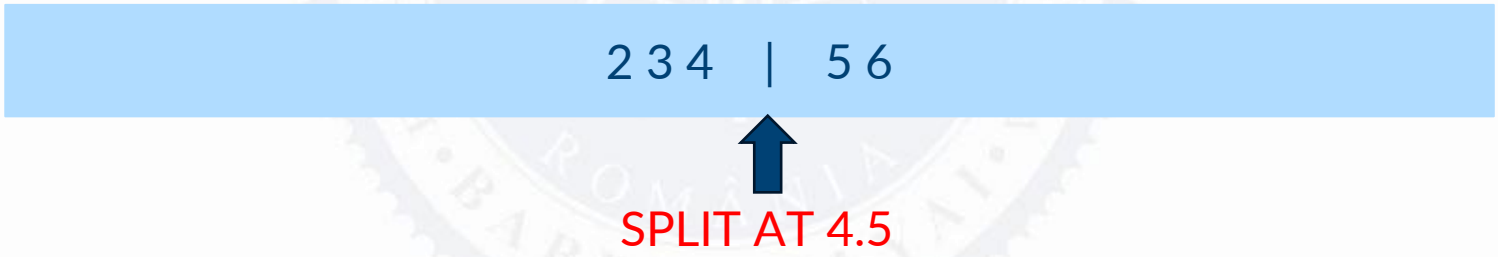
- 20 is already alone after just 1 split
- the others are still grouped together



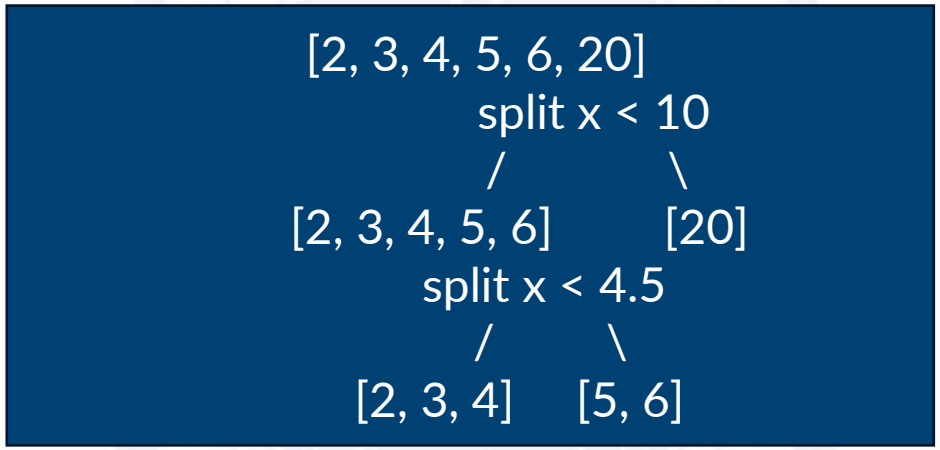
Point 20 is isolated immediately
Path length=1

Method 2: Isolation Forest

- Step 3: Continue with the normal points
- Now we continue splitting the left group:[2,3,4,5,6]
  - Suppose next random split is : x < 4.5
  - Then left: [2,3,4]
  - Right : [5,6]

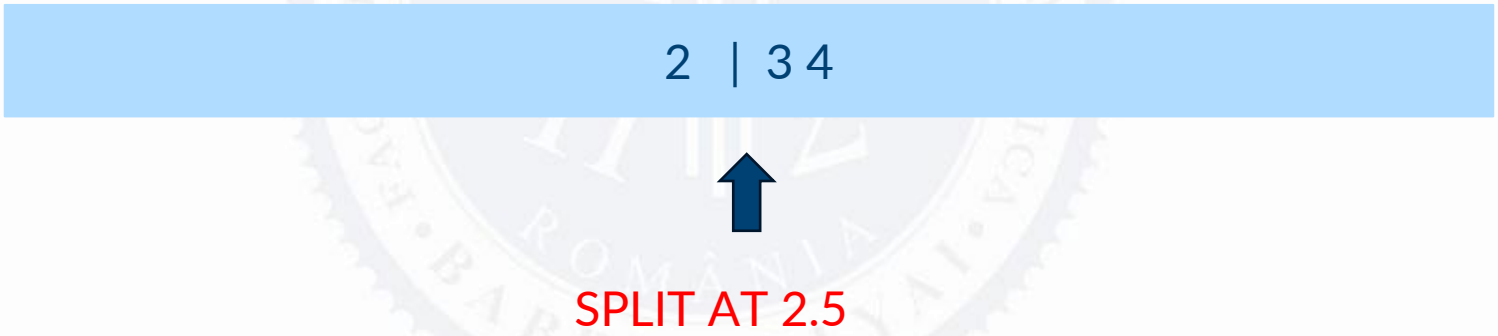


Method 2: Isolation Forest



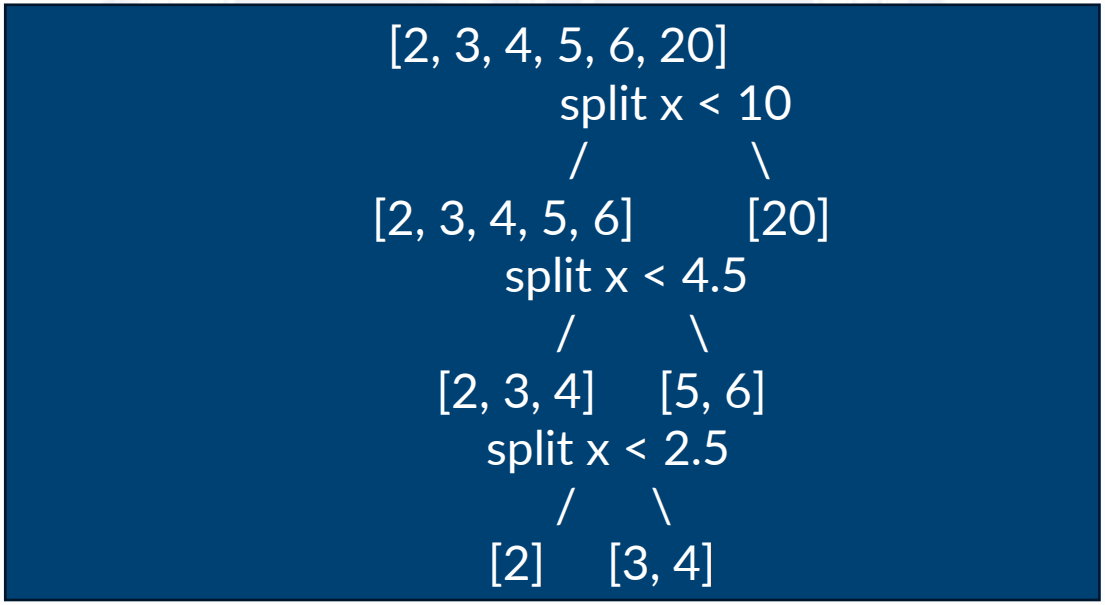
Method 2: Isolation Forest

- Step 4: split again
  - Take the group [2,3,4]
  - Suppose the next split is: x < 2.5
    - Left: [2]
    - Right: [3,4]



Method 2: Isolation Forest

- Now 2 becomes isolated, but it needed **3 splits** total:
  1. X<10
  2. X<4.5
  3. X<2.5
- So, path length for 2 is 3

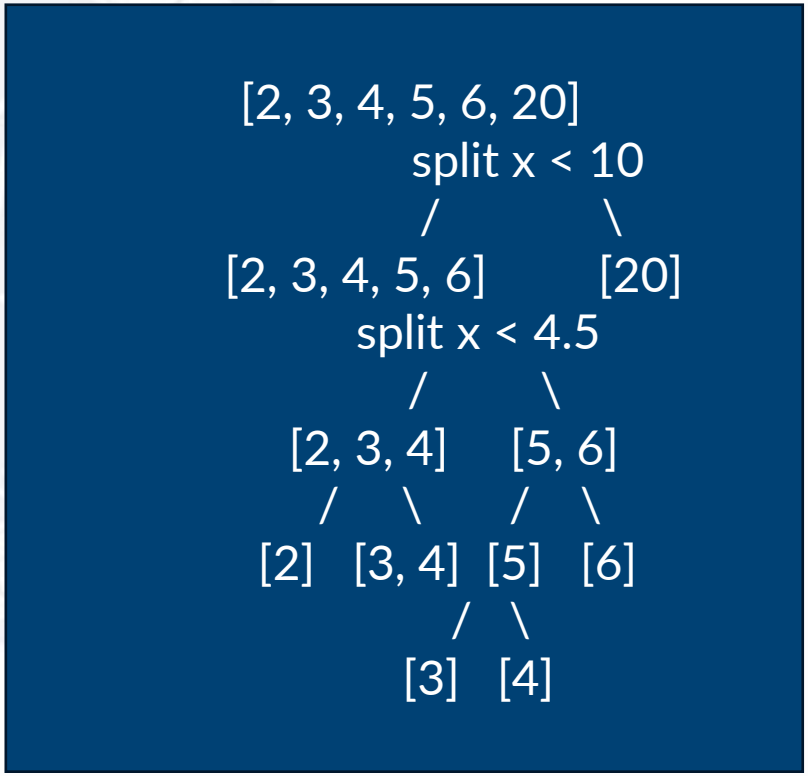


Method 2: Isolation Forest

- Step 5: Continue until all points are isolated
  - You could keep splitting:
    - [3,4]
    - [5,6]
    - Example:
      - Split [3,4] at 3.4 → isolates 3 and 4
      - Split [5,6] at 5.5 → isolates 5 and 6

Path lengths in this tree

- 20→1 split
- 2→3 splits
- 3→4 splits
- 4→4 splits
- 5→3 splits
- 6→3 splits



Method 2: Isolation Forest

Point	Path length	Meaning
20	1	Very easy to isolate
2	3	More normal
3	4	Normal
4	4	Normal
5	3	More normal
6	3	More normal

Since 20 is isolated much faster than the others, it is likely an anomaly.

Method 2: Isolation Forest

```
from sklearn.ensemble import IsolationForest
import numpy as np

X = [[1,1],[1,2],[2,1],[2,2],[10,10]]

model = IsolationForest(contamination=0.2)
model.fit(X)

pred = model.predict(X)
print(pred)
```

- Total points = 5
- contamination = 0.2
- 0.2×5=1
- The model assumes: ~1 point is anomalous
- Isolation Forest computes an anomaly score for each point

Contamination = expected proportion of anomalies in the data

```
pred = [1, 1, 1, 1, -1]
```

-1 anomaly
1 normal

Method 3: Distance-based: k-NN

- k-Nearest Neighbours Idea
- For any data point, we can measure how "crowded" its neighbourhood is:

Step 1:

Pick k (number of neighbours to consider).

Step 2:

Find the k closest points to each data point.

Step 3:

Compute the average distance to those k neighbours.

Result:

Points with HIGH average distance = potential anomalies.

Dense region
→ Normal
→ Low avg distance

Sparse region
→ Anomalous
→ High avg distance

Method 3: Distance-based: k-NN

- Normal points: close to neighbours
- Anomalies: far from neighbours

Limitations:

- expensive for large datasets
- sensitive to scaling

```
from sklearn.neighbors import NearestNeighbors
import numpy as np

X = np.array([[1,1],[1,2],[2,1],[2,2],[10,10]])

nbrs = NearestNeighbors(n_neighbors=2).fit(X)
distances, _ = nbrs.kneighbors(X)

print(distances)
```

Why n_neighbors=2?

Because:

- 1 neighbor = the point itself
- 2nd neighbor = actual closest point

Large score → anomaly

Method 4: Density-based-DBSCAN

- Core idea: anomalies are points in low-density regions

- Normal data:
  - points are **close together**

- Anomalies:
  - points are isolated

Density ≈ **how many neighbors are close to a point**

Method 4: Density-based-DBSCAN

- Two key parameters (DBSCAN idea)
  - ε (epsilon)→ radius around a point
  - min_samples→ minimum number of points inside that radius

- Classification of points
  - Core points
    - Many neighbours nearby
    - High density→ normal
  - Border points
    - Few neighbours, but near dense region
    - still considered normal
  - Noise points (anomalies) ! Not the same concept like the noise from the beginning of the lecture
    - Very few neighbours
    - Isolated
    - Low density

Method 4: Density-based-DBSCAN

- Cluster: (1,1), (1,2), (2,1), (2,2)
- Outlier: (10,10)

- Step 1: choose the parameters
  - ε = 2
  - min_samples = 3

- Step 2: check the neighbors
  - Point (1,1)
    - Distances: (used Euclidean distance)
      - To (1,2) =1
      - To (2,1)= 1
      - To (2,2)=1.4
    - All are inside ε = 2
    - neighbors found : (1,2), (2,1), (2,2) , so 3 neighbors, 3 ≥ min_samples (3) → CORE POINT

Method 4: Density-based-DBSCAN

- Step 2
  - Point (10,10)
  - Distances
    - To (2,2)=11.21
    - To (1,1)=12.73
    - To (1,2)=12.04
    - To (2,1)=12.04
  - None inside ε = 2
  - No neighbour → <3 anomaly

```
from sklearn.cluster import DBSCAN
import numpy as np

X = np.array([[1,1],[1,2],[2,1],[2,2],[10,10]])

model = DBSCAN(eps=2, min_samples=3)
labels = model.fit_predict(X)

print(labels)
```

[0 0 0 0 -1]

0 cluster
-1 anomaly

Method 4: Density-based-DBSCAN

- In dense regions:
  - many neighbors → normal
- In sparse regions:
  - few neighbors → anomaly

Advantage:

- No assumption about distribution
- Works well for irregular shapes
- Detects clusters AND anomalies together

Limitation:

- Choosing ε is hard
- Struggles in high dimensions
- Sensitive to scale

Distance-based vs. Density-based methods

Aspect	Distance-based	Density-based
Core idea	distance to neighbors	number of neighbors
Focus	absolute distance	local structure
Example	kNN	DBSCAN
Anomaly	far from others	in sparse region

Distance-based

“Is this point far from others?”

Large distance → anomaly

Density-based

“Are there enough neighbors around this point?”

Few neighbors → anomaly

Anomaly Detection Methods

Method	Idea
Statistical	Far from mean
Distance	Far from neighbors
Density	Low density
Isolation Forest	Easy to isolate

Teamwork Time

Teamwork Time- Choose Your Weapon

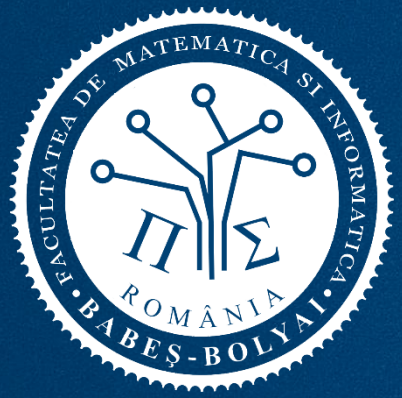
Scenario:
You work at a telecom company. Your network generates 10 million events per hour.

Each event has: timestamp, source_ip, dest_ip, bytes_transferred, protocol, duration, packet_count.

You need to detect:
a) DDoS attacks (sudden spike in connections from many IPs to one target)
b) Data exfiltration (unusually large data transfers at unusual hours)
c) Port scanning (one IP connecting to many ports in rapid succession)

For each type of anomaly:
1. Which type of anomaly is it? (Point / Contextual / Collective)
2. Which method would you use and why?
3. What features would you engineer

15 minutes — discuss in groups of 3-4



Scaling Anomaly Detection

Scaling Anomaly Detection

What changes when data goes from 10K → 10B rows?

Challenge	Small data	Large data
Pairwise distances	Fast (seconds)	Impossible
Memory	Fits in RAM	Needs distributed storage
Model training	Single machine	Needs parallelism
"Normal" changes	Static model OK	Concept drift!

Solution: we need methods designed for scale from the ground up.

Scaling Isolation Forest

Isolation Forest is already designed for scale. Here's why:

1. **Subsampling works by design**
   256 samples per tree is often enough! Anomalies are rare — a small sample captures them.

2. **Trees can be parallelized**
   Each tree is independent → train on different nodes with MapReduce.

3. **Scoring is embarrassingly parallel**
   Each point scores independently → distribute scoring across the cluster.

4. **Works with distributed frameworks**
   PySpark, Dask, and Ray all support parallel Isolation Forest.

Distributed Isolation Forest with PySpark-DEMO

- See Lecture7.ipynb

The Concept Drift Problem

What is "normal" today may not be "normal" tomorrow.

Examples of concept drift:

E-commerce: Black Friday spending looks like fraud to a model trained on normal days.
Network traffic: A new service deployment changes normal traffic patterns.
Manufacturing: Machine wear gradually shifts sensor readings over months.

Solutions:

Sliding window retraining — retrain model on recent data periodically.
Online learning — update model with each new data point.
Ensemble approach — combine models from different time periods.

Streaming Anomaly Detection

Requirements:

- Process each event in real-time (milliseconds)
- Cannot store the full stream — use fixed memory
- Model must adapt over time

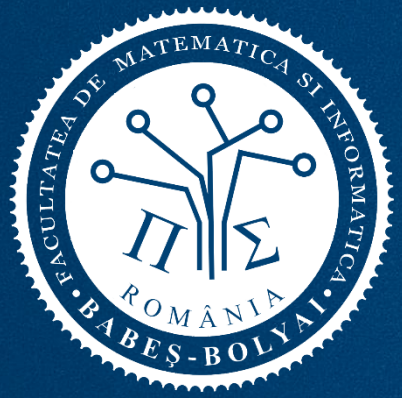
Approaches:

- Half-Space Trees: like Isolation Forest, but updates incrementally
- Windowed statistics: maintain rolling Z-scores and IQR over a time window
- Sketch-based methods: probabilistic data structures (like from Lecture 5/6)

Streaming Anomaly Detection

```python
import numpy as np
from collections import deque

class StreamingZScore:
    """Rolling Z-Score anomaly detector for data streams."""

    def __init__(self, window_size=1000, threshold=3.0):
        self.window = deque(maxlen=window_size)
        self.threshold = threshold

    def update(self, value):
        self.window.append(value)

        # Need minimum samples
        if len(self.window) < 30:
            return False, 0.0

        mean = np.mean(self.window)
        std = np.std(self.window)

        if std == 0:
            return False, 0.0

        z_score = (value - mean) / std
        is_anomaly = abs(z_score) > self.threshold

        return is_anomaly, z_score


# Usage: process events one by one
detector = StreamingZScore(window_size=5000)

for transaction in stream:  # Each arrives in real-time
    is_anomaly, score = detector.update(transaction.amount)

    if is_anomaly:
        alert(transaction, score)
```

See also the example from Lecture 5-6 (LIVE demo)



Key Takeaways

Key Takeaways

- Anomaly ≠ just rare → must be **different + meaningful**
- Domain knowledge is key
- Many approaches:
  - statistical
  - distance
  - density
  - Isolation forest (tree-based)

- Anomaly detection is *not only* unsupervised.
- But most commonly, yes, it is used in an unsupervised way.
  - It can be also semi-supervised

Thank you for your attention — questions, thoughts, or challenges?

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
BABEȘ-BOLYAI UNIVERSITY

1 Mihail Kogălniceanu Street,
Cluj-Napoca, Cluj, România

www.cs.ubbcluj.ro