IoT & Sensor Data

Shabab Akhter
External Lecturer
RUC

Contents

Contents

Intro to IoT

The Internet of Things (IoT) refers to a network of interconnected devices and objects embedded with sensors and software that communicate and share data.

They are mostly used for applications such as:

Smart Homes: IoT enables automation of home devices like lights, thermostats, and security cameras, improving energy efficiency, security, and convenience.

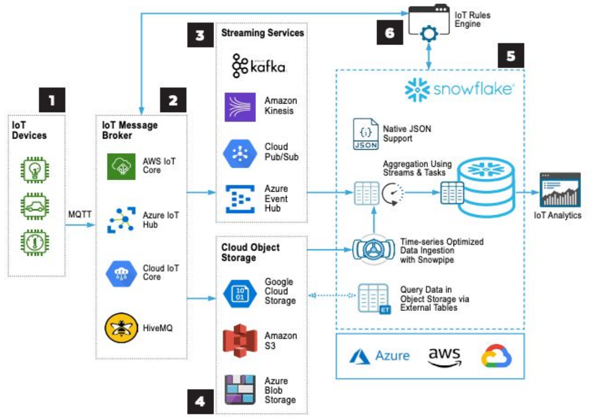
Healthcare & Remote Monitoring: IoT devices track patient health data in real time, supporting better disease management and reducing hospital visits.

Industrial IoT (IIoT): In industries, IoT monitors machines, tracks inventory, and optimizes production, increasing efficiency and reducing downtime.

Smart Cities: IoT helps manage traffic, waste, and energy systems, reducing congestion, improving services, and saving resources.

Key Technologies:

- Sensors
- Data acquisition
- Cloud/Edge computing
- Machine learning for predictive analytics



IoT Computing Layers

Cloud computing layer

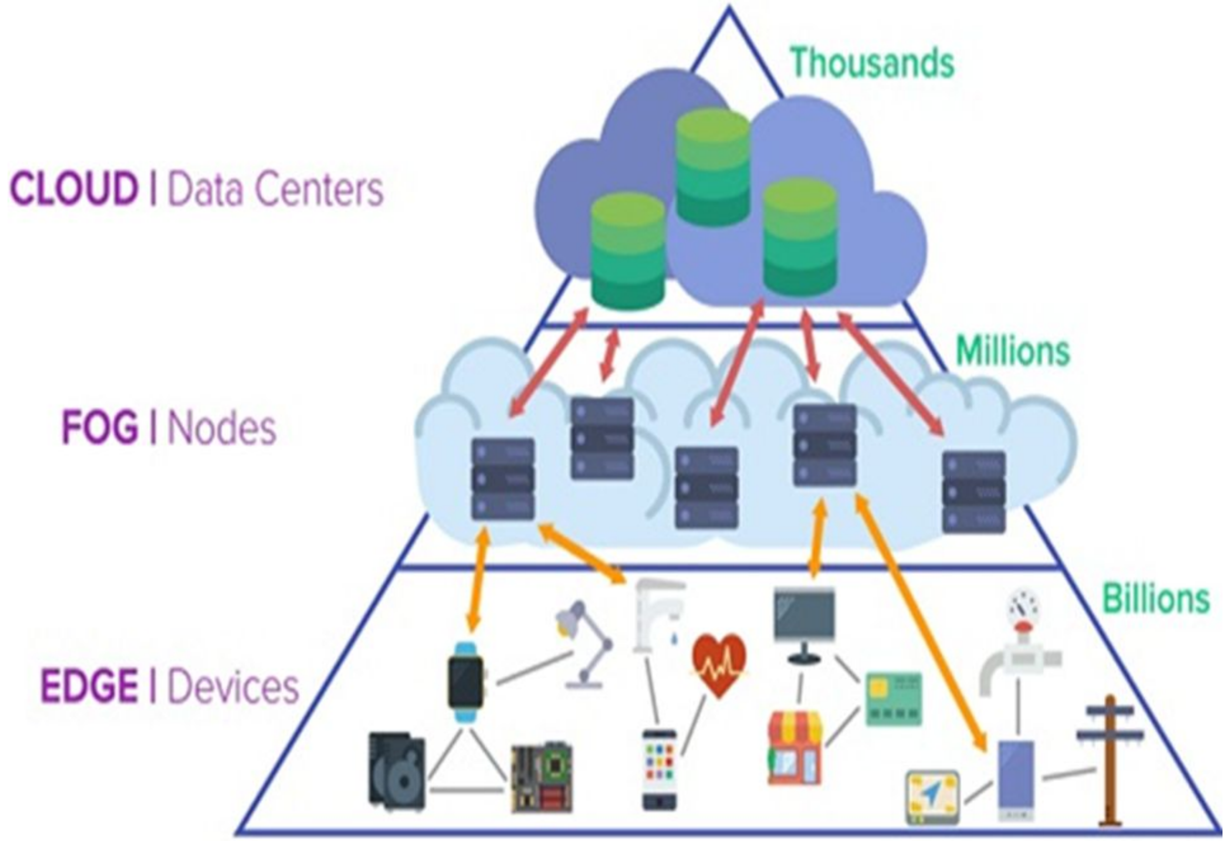
- Aggregates data summaries from multiple fog nodes
- Performs deeper analysis on large data sets and sends application rules to fog nodes

Fog computing layer

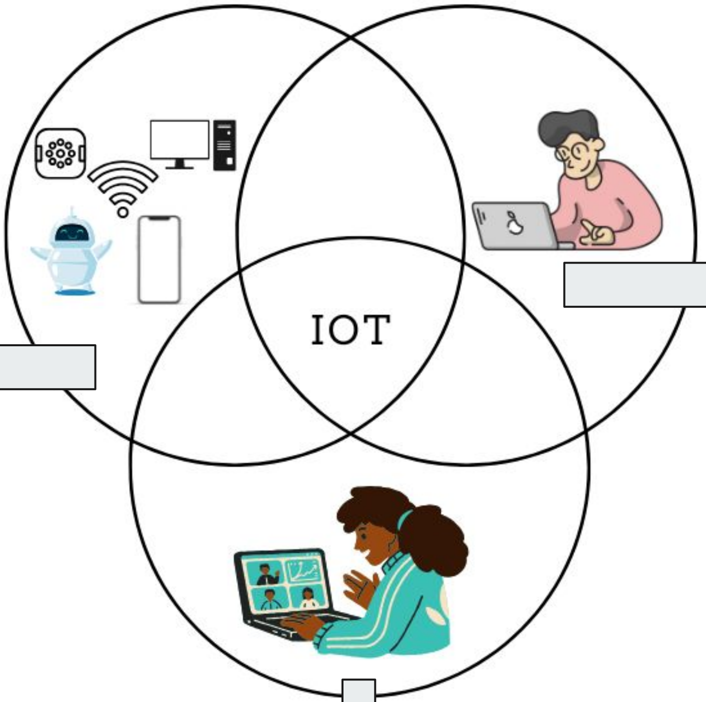
- Transient storage for immediate data
- Real-time analytics and control based on application rules provided by cloud layer

Edge computing layer

- Capture user interactions and send feeds to Fog node
- Performs intelligent actions based on real-time control signals from fog nodes



Connection Types in IoT

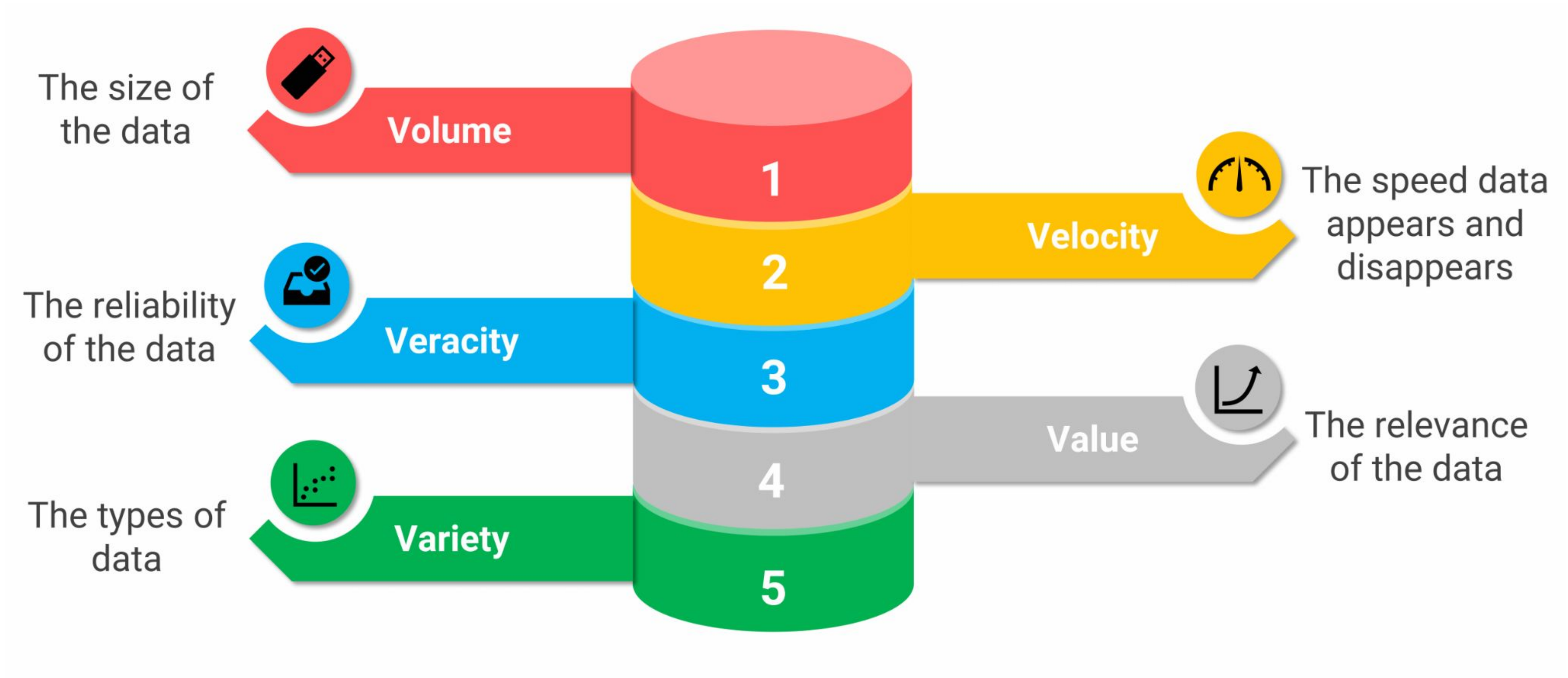


Machine-to-machine (M2M): Connect 'things' over a network, including sensors, robots, computers, and mobile devices.

Machine-to-people (M2P): Connection for transferring information between a thing and a person.

People-to-People (P2P): Connections between one person and another through video, mobile devices, and social networks, called collaboration.

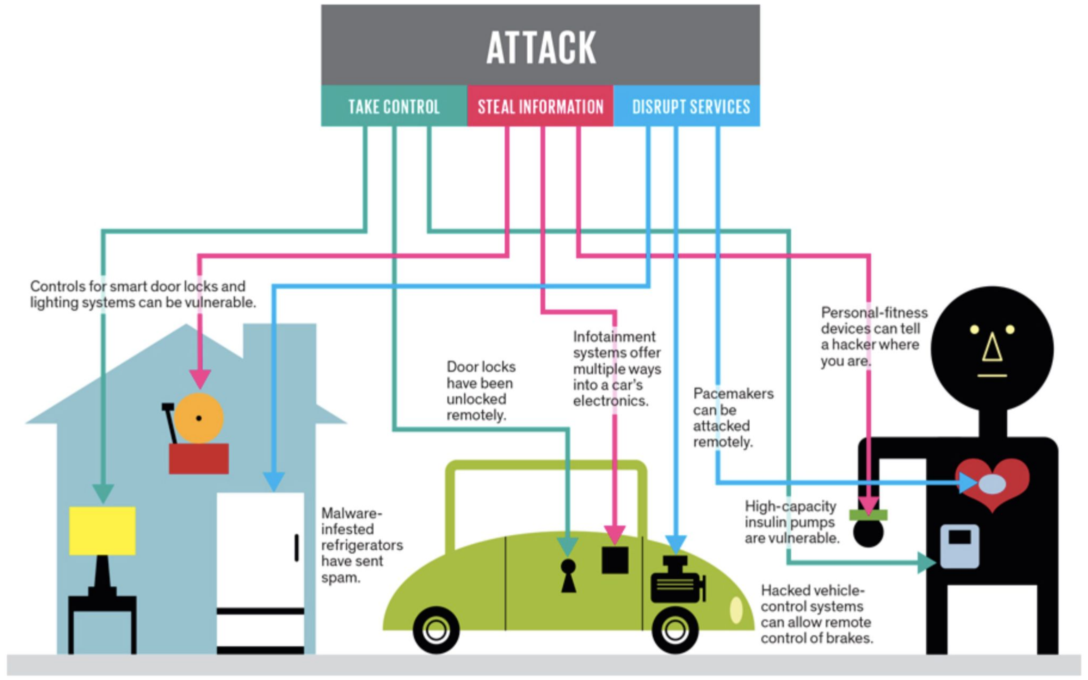
Characteristics of IoT Data



The size of the data
Volume
1
Velocity
2
The speed data appears and disappears
The reliability of the data
Veracity
3
Value
4
The relevance of the data
The types of data
Variety
5

The Challenge with IoT Data

- **Data Volume:** IoT devices generate vast amounts of data, making storage and processing a challenge.
- **Data Quality:** Raw sensor data may be noisy, incomplete, or erroneous, affecting analysis accuracy.
- **Data Security:** IoT devices are vulnerable to cyberattacks, posing privacy and security risks.
- **Data Integration:** Devices may use different data formats and protocols, requiring complex integration.
- **Real-Time Processing:** Many applications require real-time data processing, demanding efficient systems.
- **Scalability:** IoT systems must handle millions of devices and vast amounts of data.

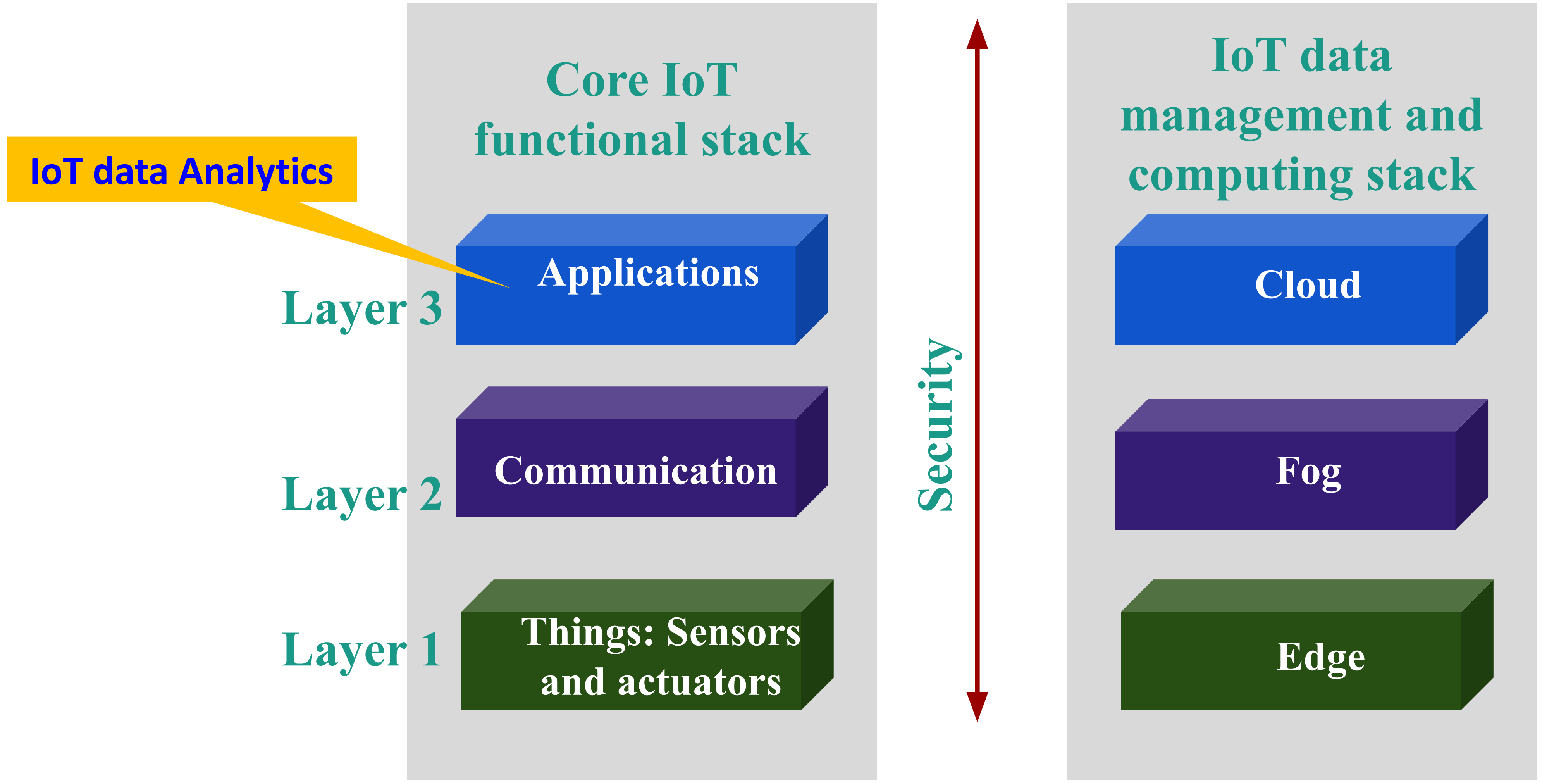

ATTACK
TAKE CONTROL
STEAL INFORMATION
DISRUPT SERVICES
Controls for smart door locks and lighting systems can be vulnerable.
Door locks have been unlocked remotely.
Infotainment systems offer multiple ways into a car's electronics.
Pacemakers can be attacked remotely.
Personal-fitness devices can tell a hacker where you are.
Malware-infested refrigerators have sent spam.
High-capacity insulin pumps are vulnerable.
Hacked vehicle-control systems can allow remote control of brakes.

Contents

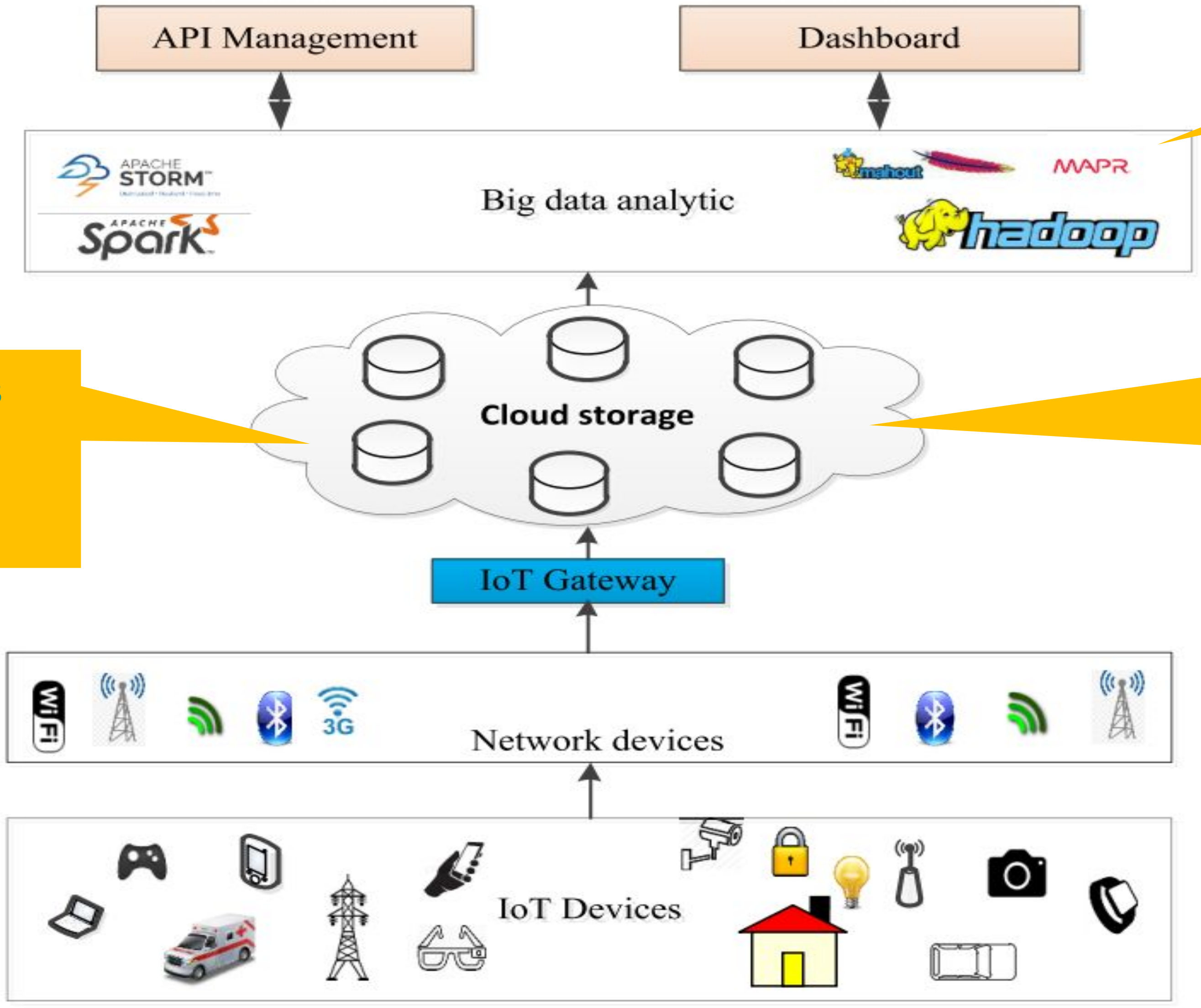
Connectivity is required to perform IoT Data Analytics

- Purpose: To harvest value out of IoT data
- Requirements for IoT applications
  - Real-time monitoring of what “things” are doing, their operating conditions, their issues, their load, their configuration, etc.
  - Predictive maintenance: fix before “things” break.
  - Optimization: configuration, interaction with humans, energy efficiency, most suitable “things” for different uses.
  - Analytics to find out the current shortcomings to design next versions of “things”.

Simplified IoT Layer Architecture


IoT data Analytics
Core IoT functional stack
Layer 3
Applications
Layer 2
Communication
Layer 1
Things: Sensors and actuators
Security
IoT data management and computing stack
Cloud
Fog
Edge

Cloud Based IoT Data Analytics



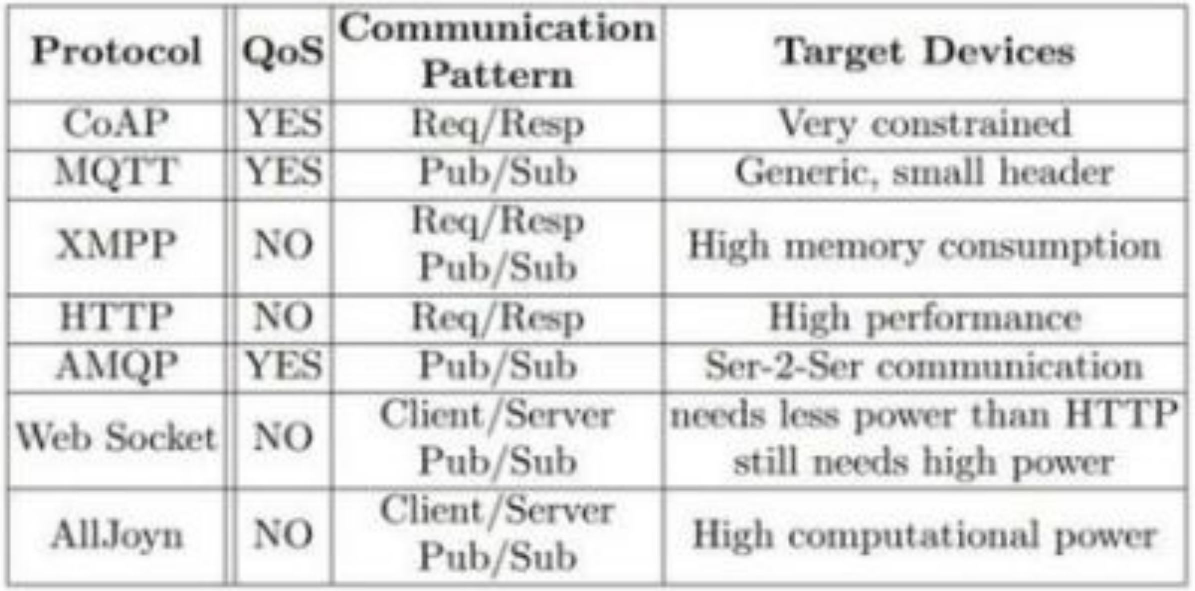
Many more available

Types of clouds
- Public
- Private
- Hybrid

Services
- Google IoT Core
- IBM Cloud
- Amazon EC2
- MS Azure
- …

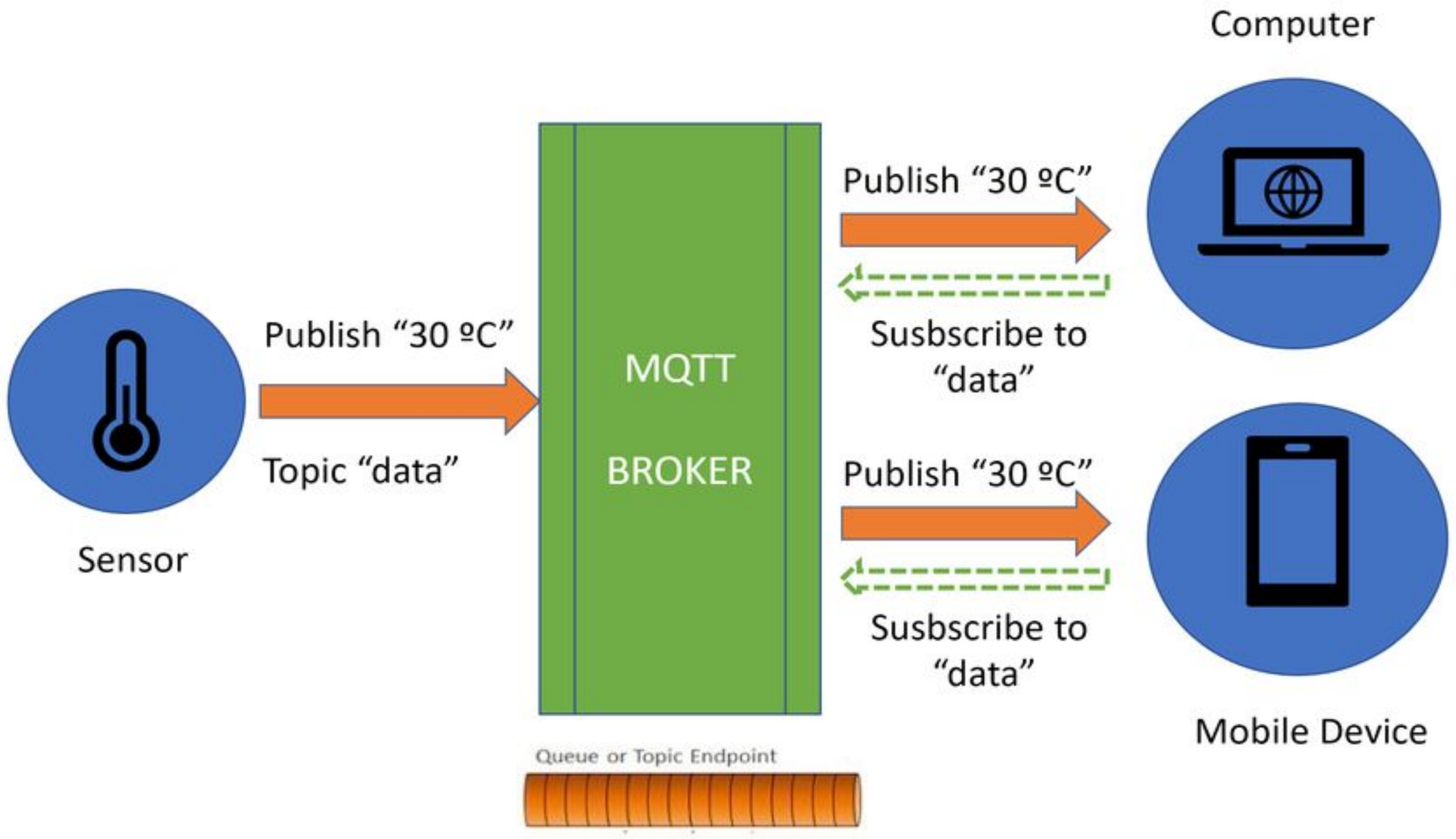
Typical IoT Application Protocols

- REST
  - An architectural style for developing web services.
- CoAP
  - Constrained Application Protocol
- MQTT
  - Message Queuing Telemetry Transport
- XMPP
  - Extensible Messaging and Presence Protocol
- AMQP
  - Advanced Message Queuing Protocol
- WebSocket
  - Computer Communications Protocol



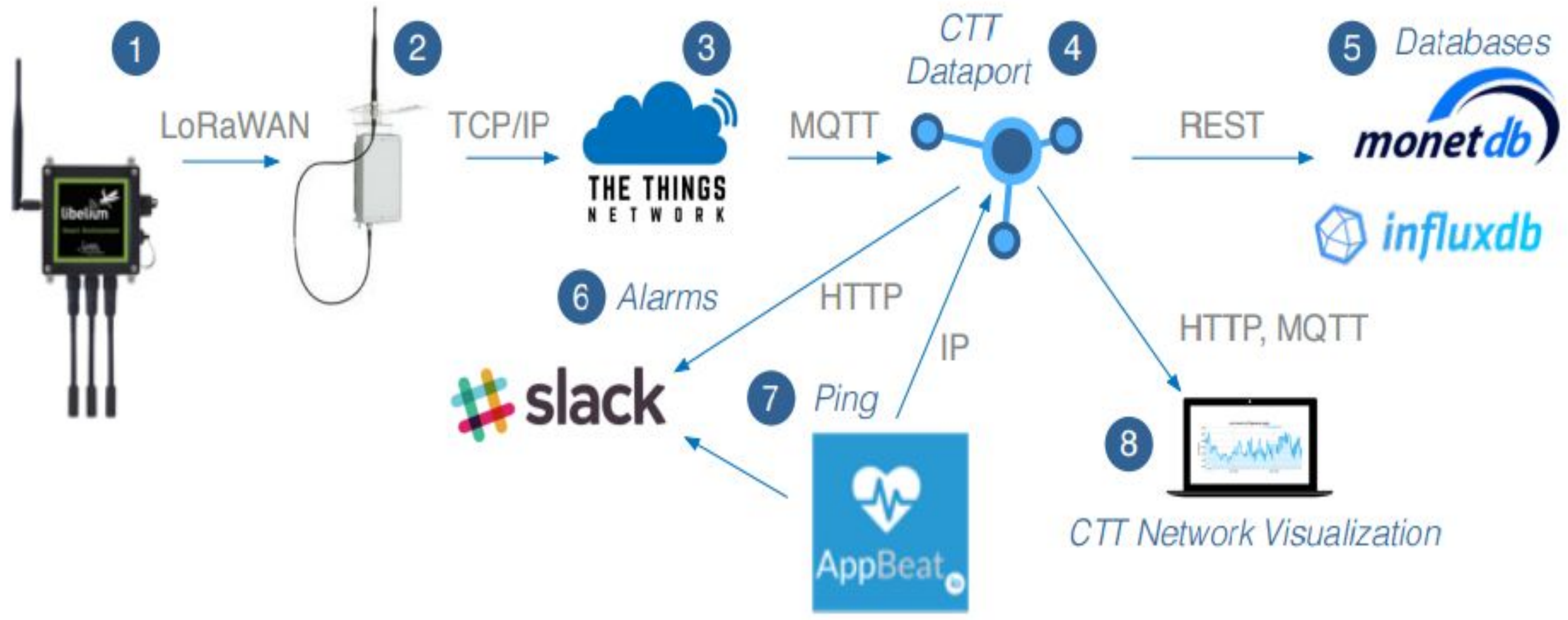
Protocol	QoS	Communication Pattern	Target Devices
CoAP	YES	Req/Resp	Very constrained
MQTT	YES	Pub/Sub	Generic, small header
XMPP	NO	Req/Resp Pub/Sub	High memory consumption
HTTP	NO	Req/Resp	High performance
AMQP	YES	Pub/Sub	Ser-2-Ser communication
Web Socket	NO	Client/Server Pub/Sub	needs less power than HTTP still needs high power
AllJoyn	NO	Client/Server Pub/Sub	High computational power

Examples of Messaging Technologies: MQTT



Topic: temperature

Protocols Used in A Full IoT Application



Exercise – use MQTT brokers

1. Let us see how to setup an MQTT broker and send messages
2. Now, try yourself:

   Use the diabetes.csv dataset to do the following:

   1. Select the following 4 attributes (3 features + 1 class label) :
      - Glucose, BloodPressure, Insulin, Outcome
   2. Normalize Glucose, BloodPressure and Insulin to [0, 1] using MinMax.
   3. Store the new data (3 normalized features + 1 class label) in another dataset ***S***.
   4. Modify the MQTT example to do the following:
      - The publisher publishes records in ***S*** continuously. When it reaches the end of ***S***, it continues to send from the beginning again.
      - The subscriber continuously receives the data. For each latest record *r* received, apply the 3NN classification to the last 5 records before r, and compare the classification result with the Outcome label in *r*.
        - Repeat this for 1000 times, and report the number of correct classifications.

Contents

Predictive Maintenance

Predictive Maintenance (PdM) refers to the practice of using data from IoT devices and sensors to predict when equipment or machinery will fail, allowing for proactive maintenance before any issues arise. This approach helps avoid unplanned downtime and reduces maintenance costs.

They are mostly used for applications such as:

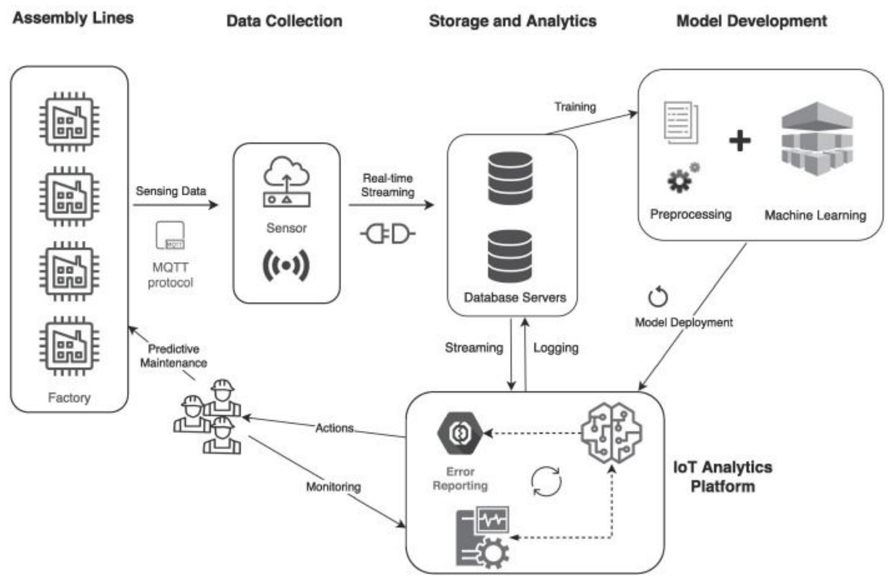
Automotive: IoT devices track the performance of vehicles, including engine health, tire pressure, and battery status. Predictive maintenance in this sector helps avoid breakdowns by predicting component failure and recommending timely repairs, extending vehicle lifespan and improving safety.

Energy & Utilities: In power plants and energy grids, IoT sensors monitor turbines, transformers, and other equipment. Predictive maintenance predicts potential failures in critical infrastructure, ensuring smooth operations and preventing energy outages.

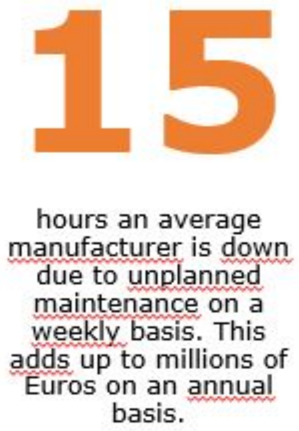
Aerospace: Aircraft and engine components are equipped with IoT sensors that monitor conditions such as pressure, temperature, and vibration. Predictive maintenance in aviation ensures timely inspections and repairs, enhancing safety and reducing operational disruptions.

Key Technologies:

- IoT Sensors
- Data acquisition
- Cloud/Edge computing
- Machine learning for predictive analytics



Predictive maintenance is critical in manufacturing



15

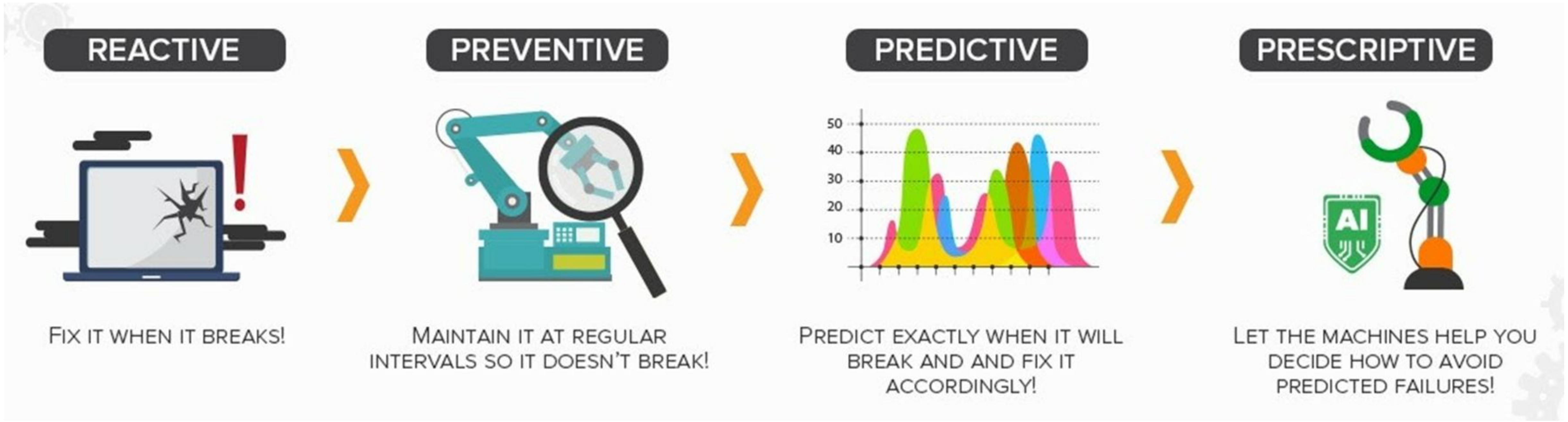
hours an average manufacturer is down due to unplanned maintenance on a weekly basis. This adds up to millions of Euros on an annual basis.

Sensor data combined with historical **maintenance/downtime data** can be used to **learn failure patterns** and hence predict failures in advance

VIBRATIONS
TEMPERATURE
PRESSURE
OIL
NOISE

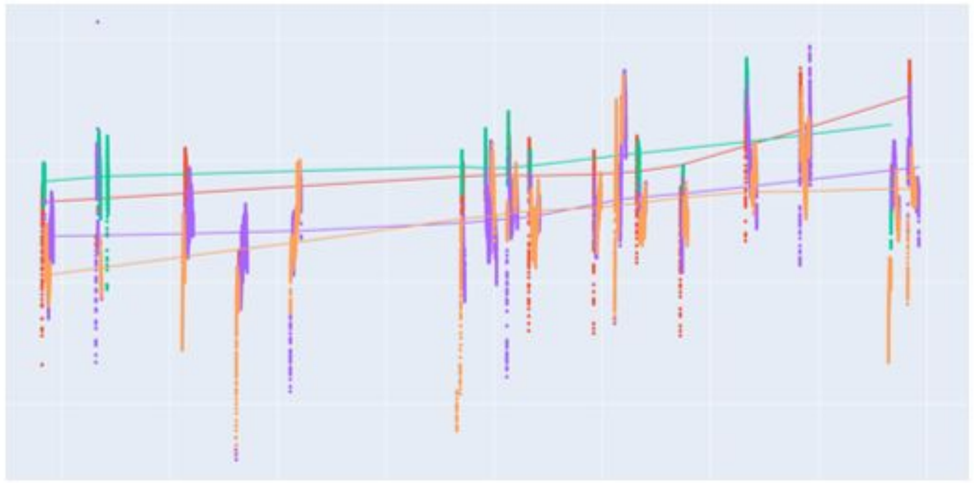
Predicting failures in advance will enable saving millions of Euros annually

Different types of maintenance

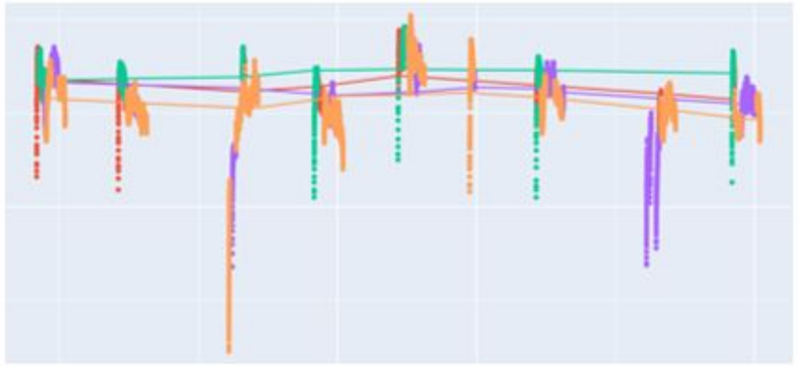


REACTIVE
PREVENTIVE
PREDICTIVE
PRESCRIPTIVE
50
40
30
20
10
AI
FIX IT WHEN IT BREAKS!
MAINTAIN IT AT REGULAR INTERVALS SO IT DOESN'T BREAK!
PREDICT EXACTLY WHEN IT WILL BREAK AND AND FIX IT ACCORDINGLY!
LET THE MACHINES HELP YOU DECIDE HOW TO AVOID PREDICTED FAILURES!

Real life sensor examples

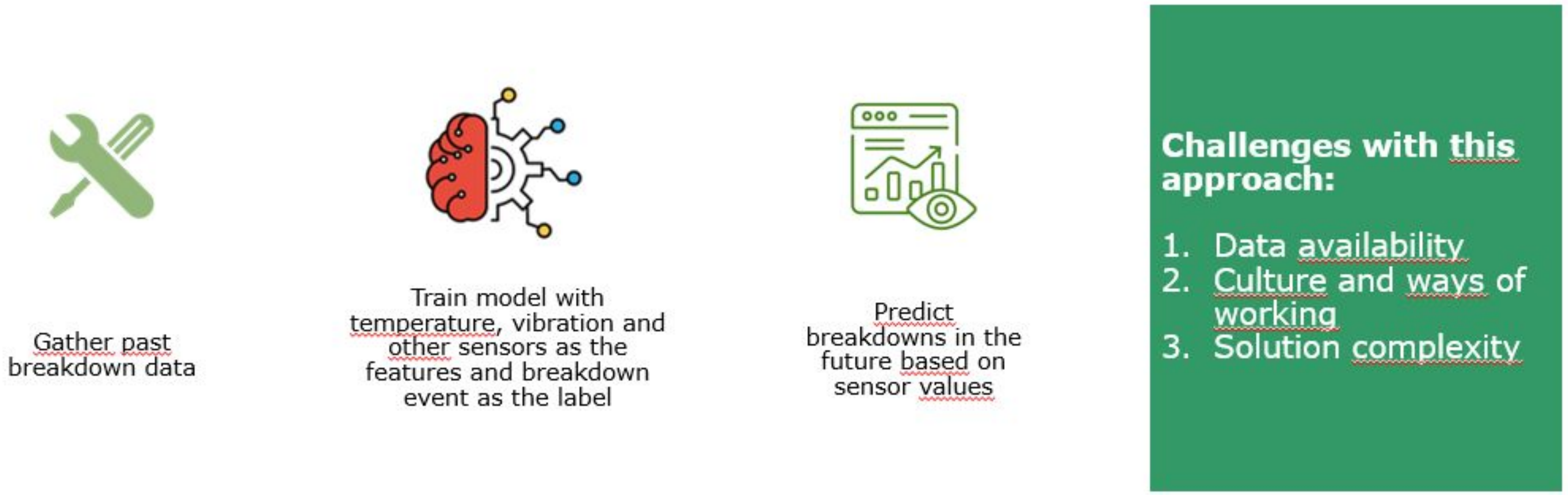


Trend line indicates sensor value is getting higher over time across regimes.
Indicates possibility to predict replacement in advance.



Trend line indicates sensor values is relatively stable over time across regimes.
Indicates possibility of stopping unnecessary early replacements setup based on predefined schedules.

Reality is challenging



Gather past breakdown data

Train model with temperature, vibration and other sensors as the features and breakdown event as the label

Predict breakdowns in the future based on sensor values

Challenges with this approach:

1. Data availability
2. Culture and ways of working
3. Solution complexity

We can use unsupervised methods instead

First the data needs to be treated using methods such as Dimensionality Reduction

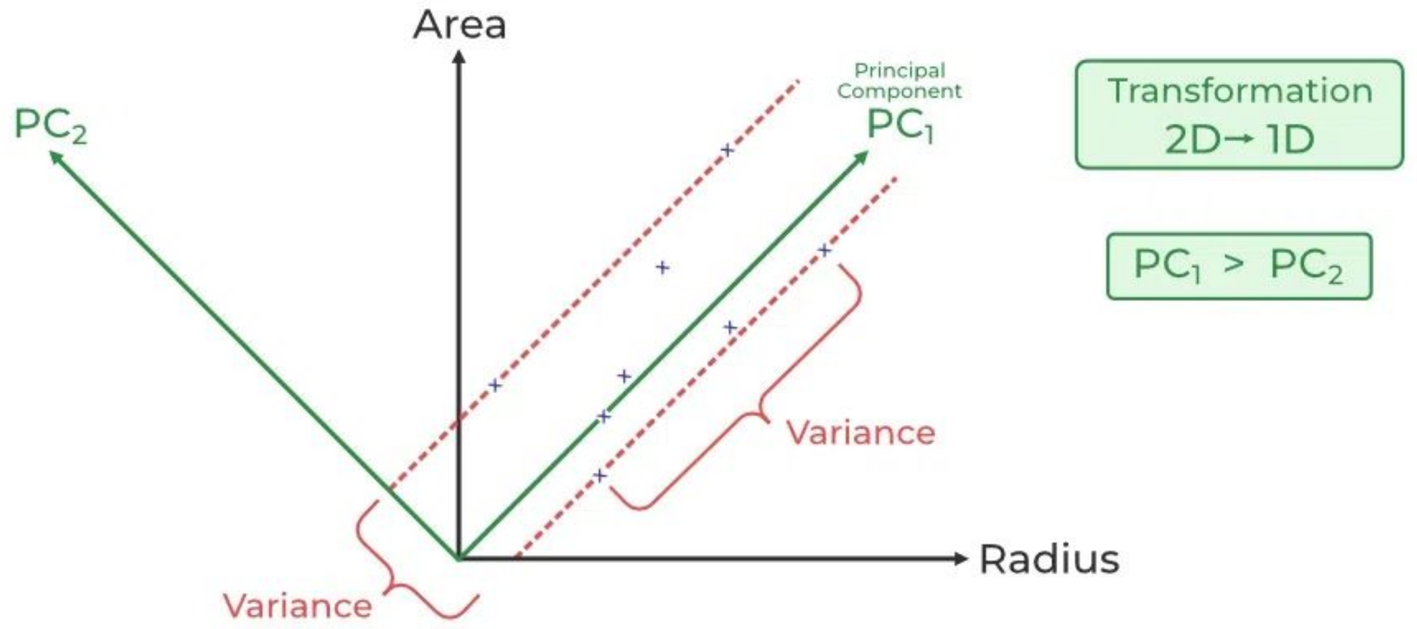
Principal Component Analysis (PCA): Reducing the number of features while retaining most of the variance in the data.

t-SNE or UMAP: Visualizing high-dimensional data or reducing dimensionality in a way that highlights important patterns.

Importance of Dimensionality Reduction

- **Reduces Computational Complexity**: Minimizes processing time and resource usage by working with fewer features.
- **Mitigates Overfitting:** Helps models generalize better by removing irrelevant or redundant features.
- **Enhances Interpretability:** Simplifies data, making it easier to identify key patterns and relationships.

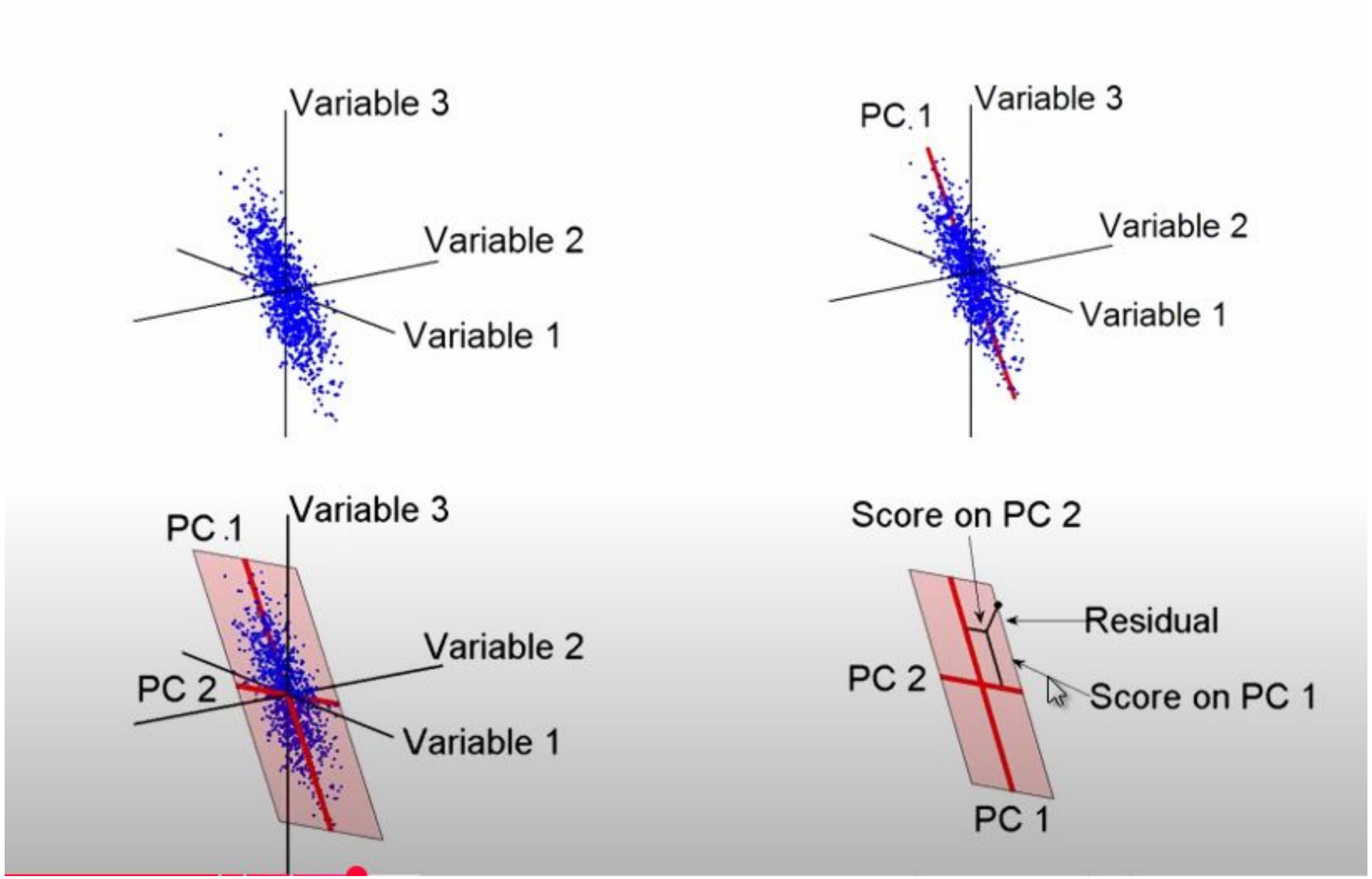
Principal component analysis



1. **Dimensionality Reduction:** PCA is used to reduce the dimensionality of the data (e.g., from 2D to 1D in this example) by focusing on the components that capture the most variance (PC1). This makes the data easier to analyze while minimizing information loss.

2. **Feature Transformation:**The data is reoriented along new axes (principal components), which are linear combinations of the original features.

PCA – another example



Try to draw a straight line that capture most of the variance in the data

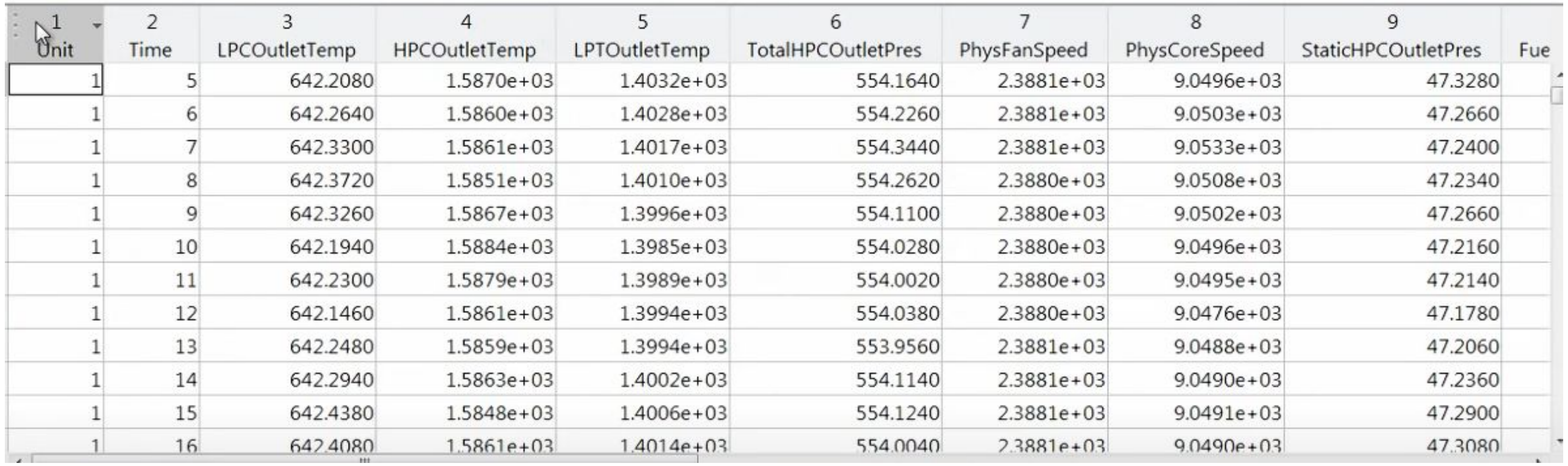
Then draw another line orthogonal to the first line that again captures most of the variance in the data

The two lines form a new 2D plane which is used as the new converted dataset

PCA in action

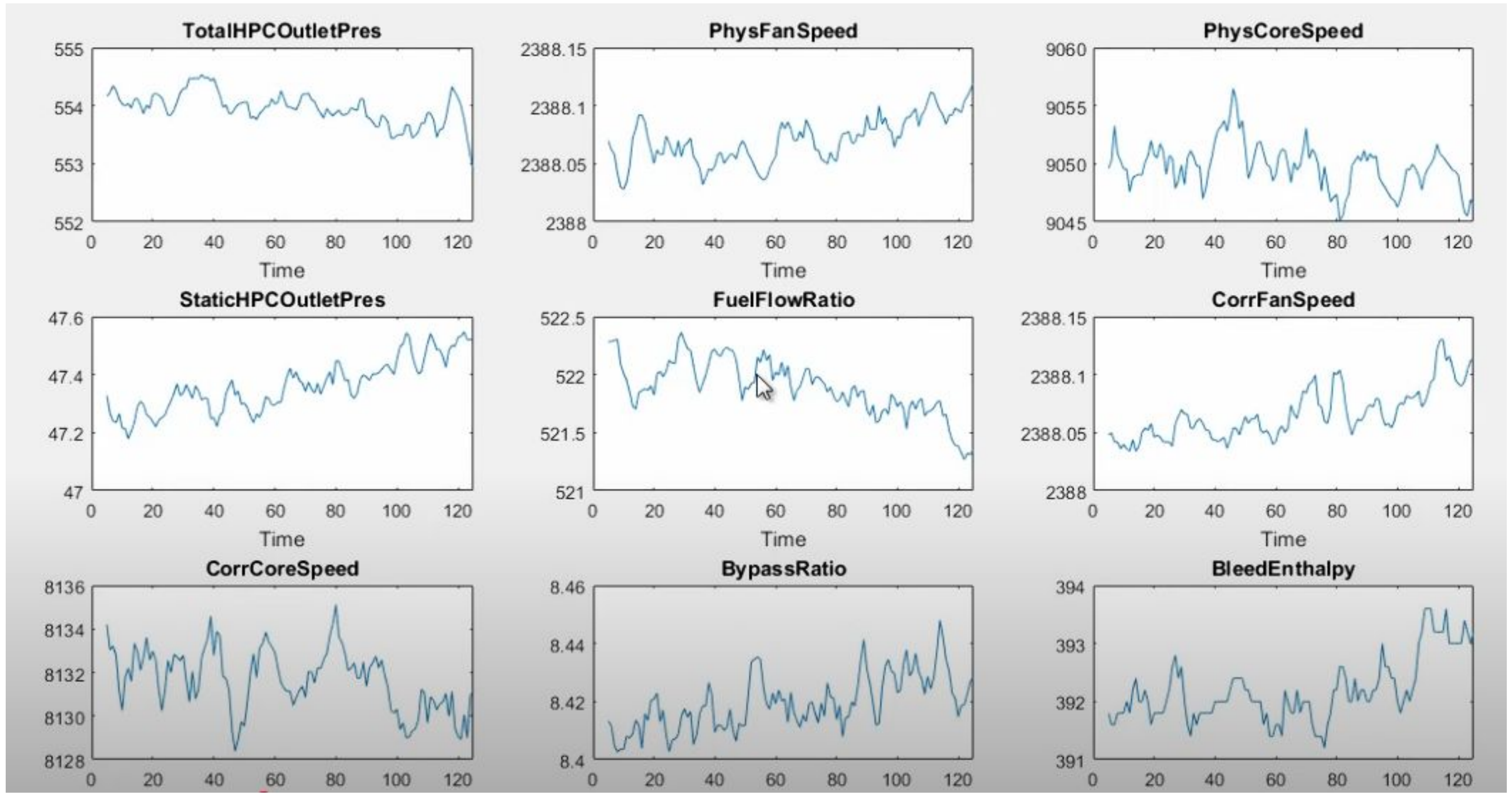
Dataset containing sensors values from sensors in a plane engine. There are a total of 14 sensors that are sending values every second. The goal is to be able to predict engine failure based on these sensor values.

The hard part: **there are no labels!**



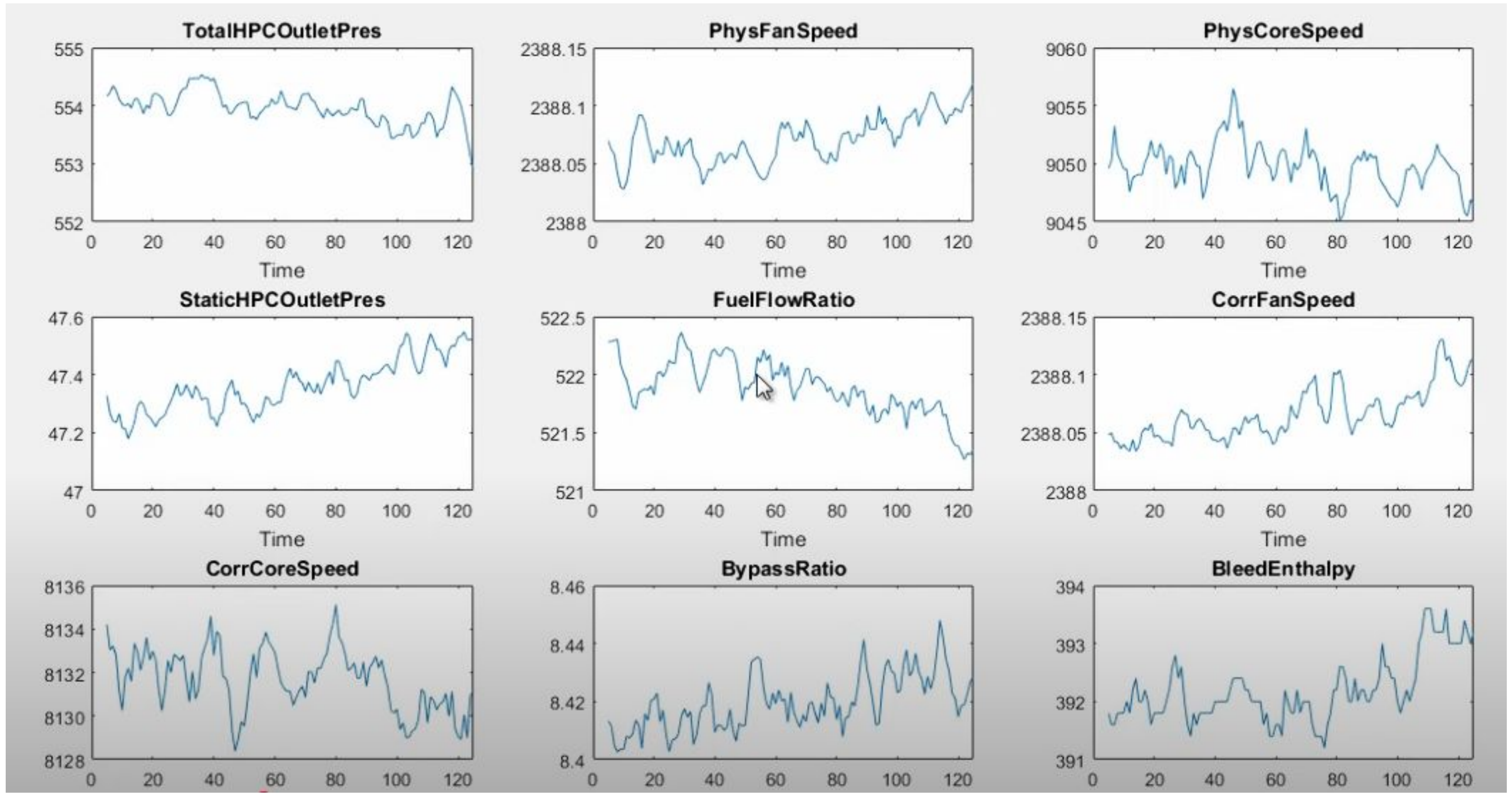
1 Unit	2 Time	3 LPCOutletTemp	4 HPCOutletTemp	5 LPTOutletTemp	6 TotalHPCOutletPres	7 PhysFanSpeed	8 PhysCoreSpeed	9 StaticHPCOutletPres	Fue
1	5	642.2080	1.5870e+03	1.4032e+03	554.1640	2.3881e+03	9.0496e+03	47.3280	
1	6	642.2640	1.5860e+03	1.4028e+03	554.2260	2.3881e+03	9.0503e+03	47.2660	
1	7	642.3300	1.5861e+03	1.4017e+03	554.3440	2.3881e+03	9.0533e+03	47.2400	
1	8	642.3720	1.5851e+03	1.4010e+03	554.2620	2.3880e+03	9.0508e+03	47.2340	
1	9	642.3260	1.5867e+03	1.3996e+03	554.1100	2.3880e+03	9.0502e+03	47.2660	
1	10	642.1940	1.5884e+03	1.3985e+03	554.0280	2.3880e+03	9.0496e+03	47.2160	
1	11	642.2300	1.5879e+03	1.3989e+03	554.0020	2.3880e+03	9.0495e+03	47.2140	
1	12	642.1460	1.5861e+03	1.3994e+03	554.0380	2.3880e+03	9.0476e+03	47.1780	
1	13	642.2480	1.5859e+03	1.3994e+03	553.9560	2.3881e+03	9.0488e+03	47.2060	
1	14	642.2940	1.5863e+03	1.4002e+03	554.1140	2.3881e+03	9.0490e+03	47.2360	
1	15	642.4380	1.5848e+03	1.4006e+03	554.1240	2.3881e+03	9.0491e+03	47.2900	
1	16	642.4080	1.5861e+03	1.4014e+03	554.0040	2.3881e+03	9.0490e+03	47.3080	

We start with EDA & visualizations (ofcourse)



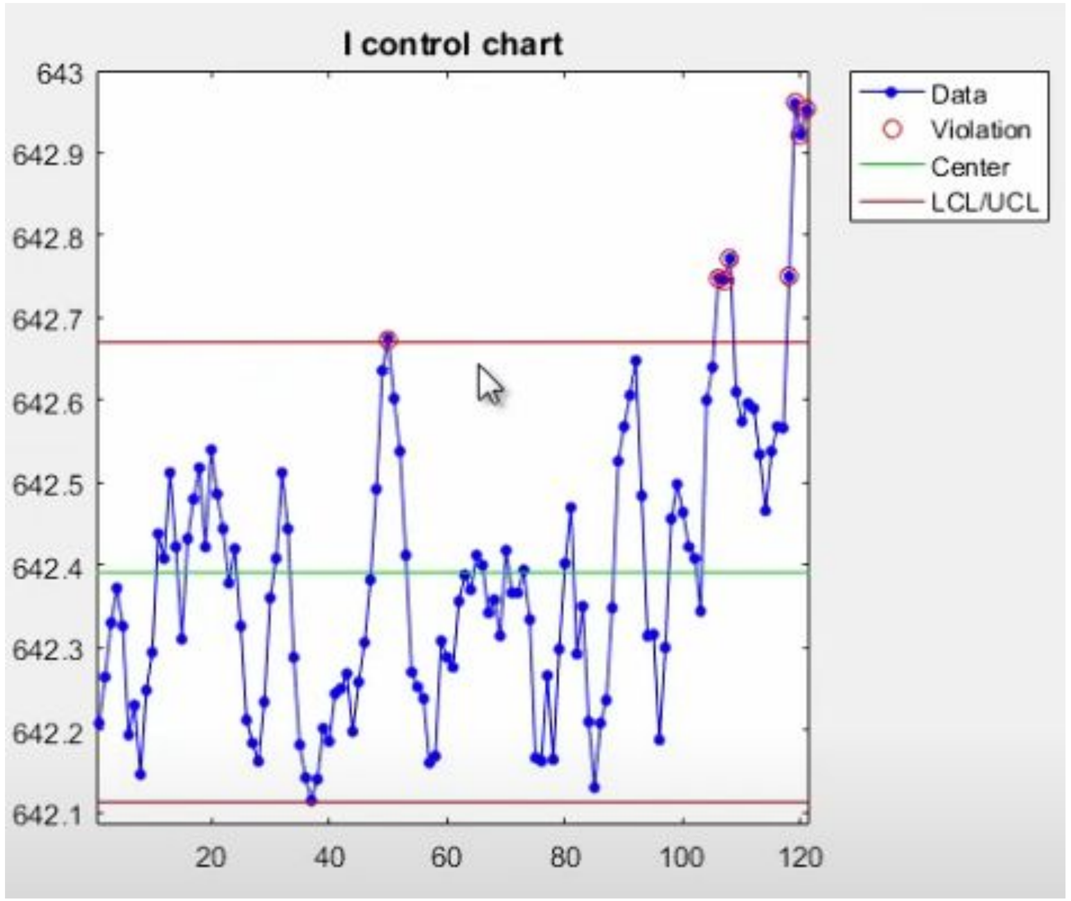
Nice to see general view of the data and does give some insights – but can you really predict anything with this?

We start with EDA & visualizations (ofcourse)



Nice to see general view of the data and does give some insights – but can you really predict anything with this?

We can use control charts



Old school method but works very well if you have small datasets.

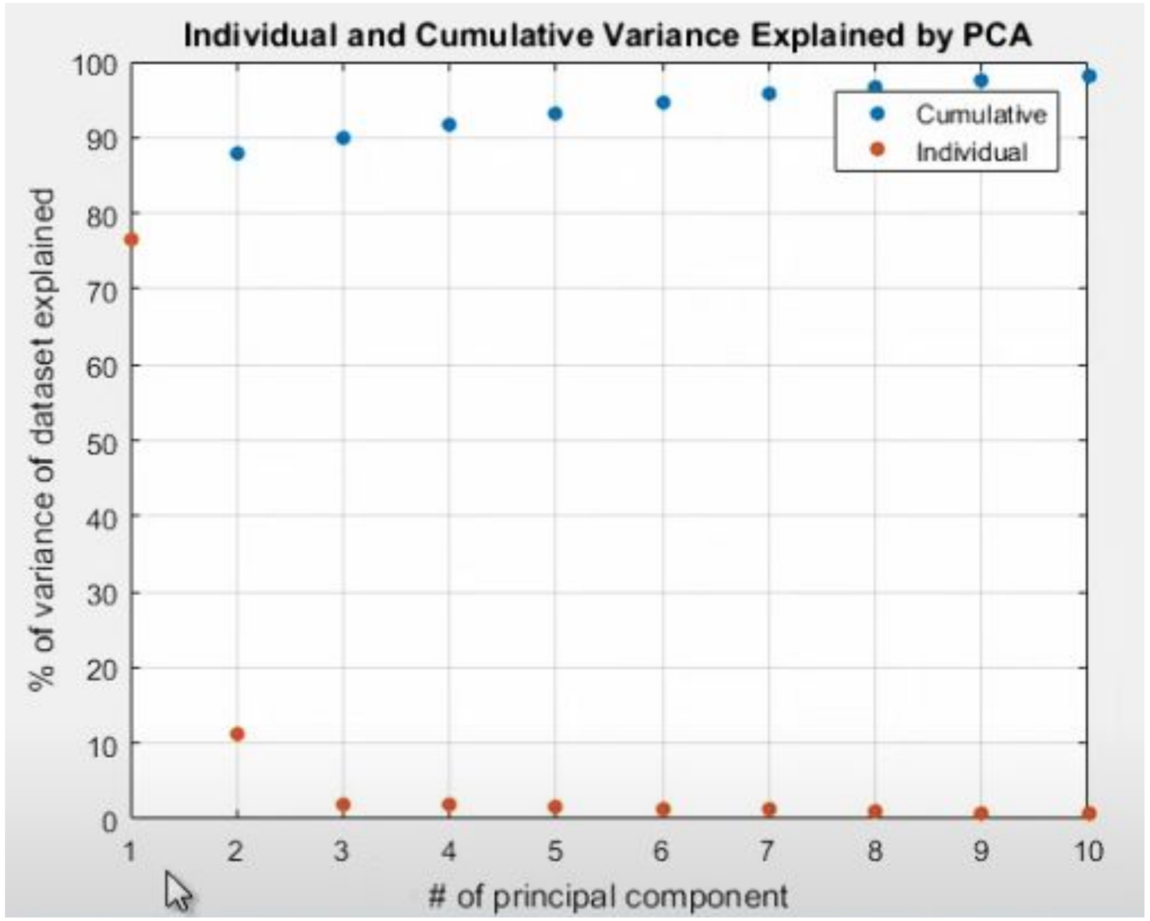
Concept is simple:

1. Find a control level, i.e. what are the most common values of the data

2. Anything below or under this means that something is potentially wrong

What do you think could be challenges with this method, particular to our dataset?

This is where we can apply PCA



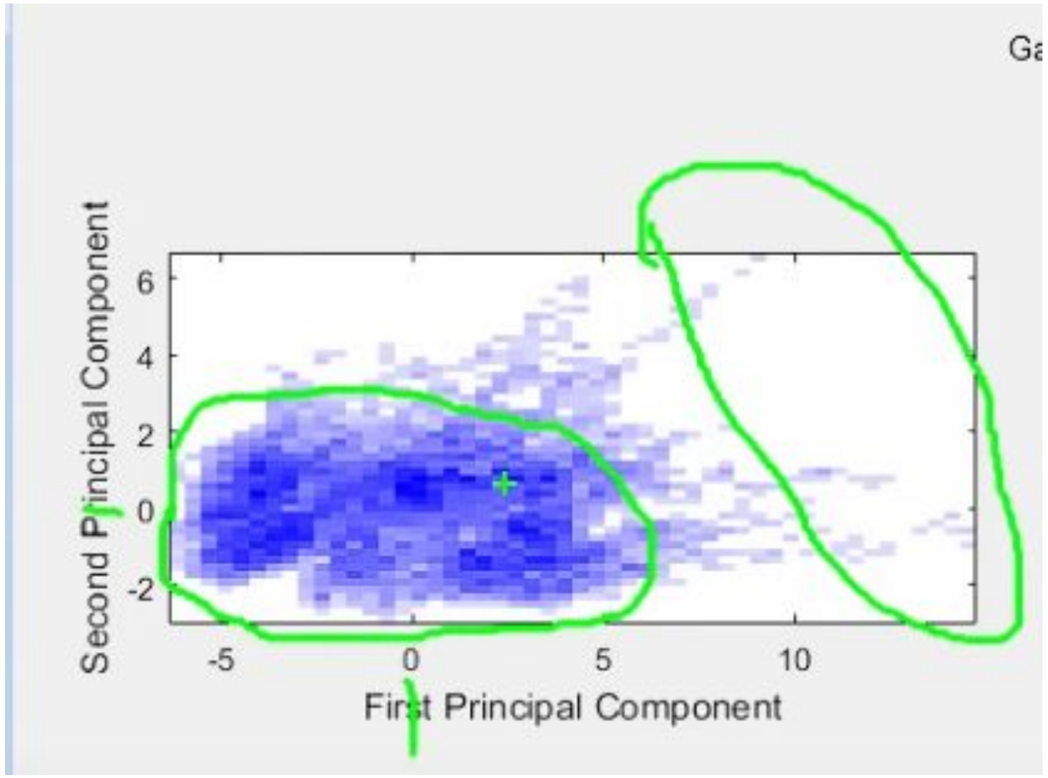
After applying PCA, we can check how much of our variance is explained by each principal component.

More than 75% is explained with just one component instead of the 14 variables.

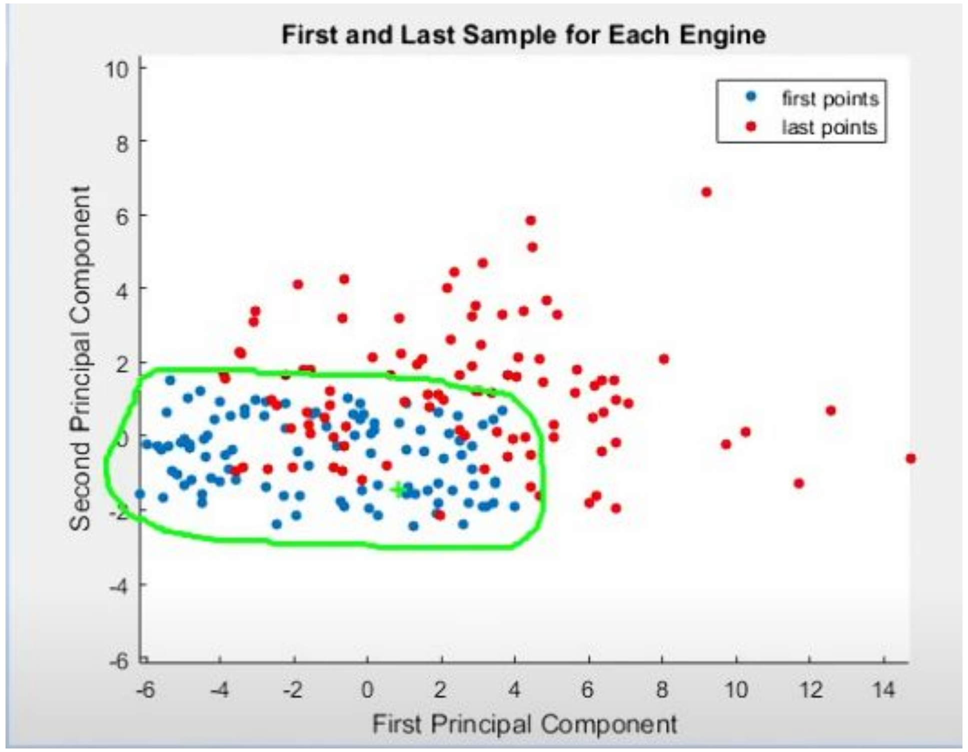
More than 80% explained by 4 components.

This is where we can apply PCA

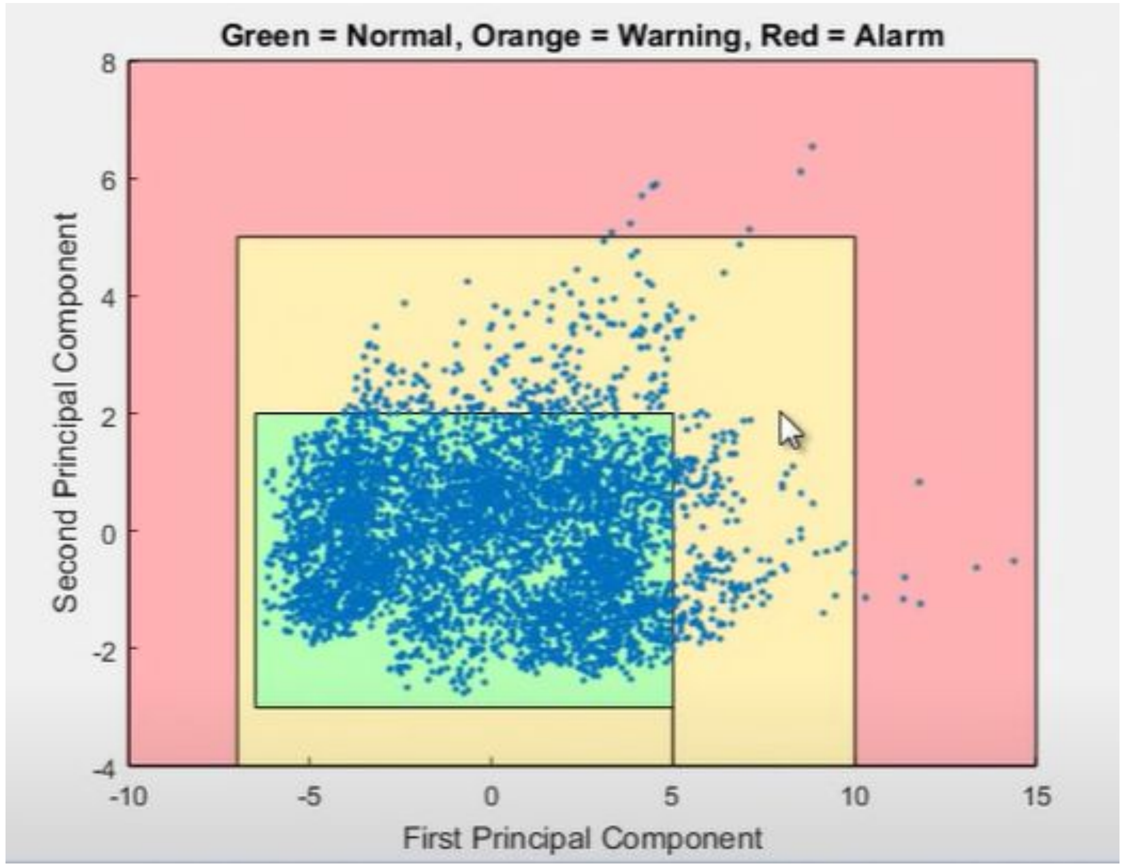
After plotting principal components (3 components were selected), the graph gives valuable insights



We can validate our theory by visualizing the data points based on time



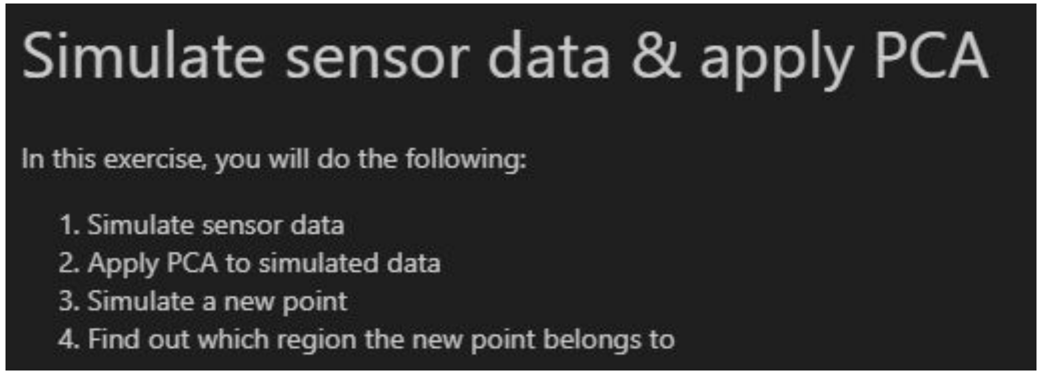
In the end, we can create something like this



Once we have this model, once new sensor values come in, we can get its principal components and put it in this plot.

Based on where the point(s) fall in this plot, we can decide if there is risk of engine failure

Exercise – try to do the same yourself



Simulate sensor data & apply PCA

In this exercise, you will do the following:

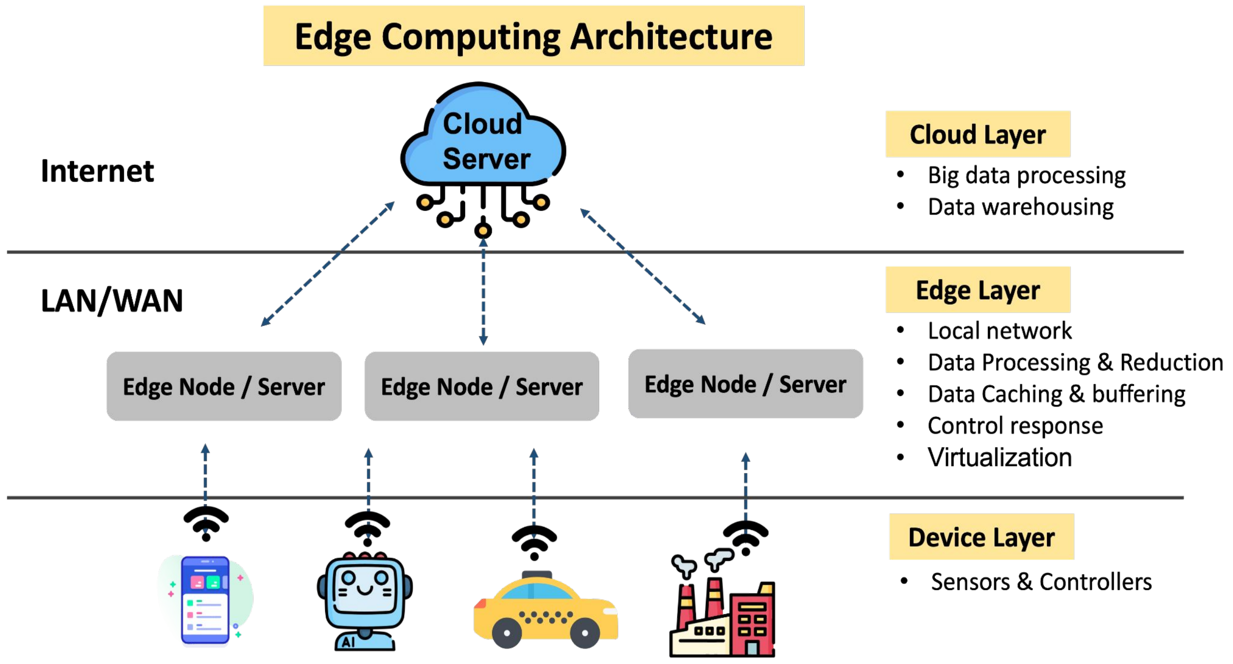
1. Simulate sensor data
2. Apply PCA to simulated data
3. Simulate a new point
4. Find out which region the new point belongs to

Contents

IoT and edge computing

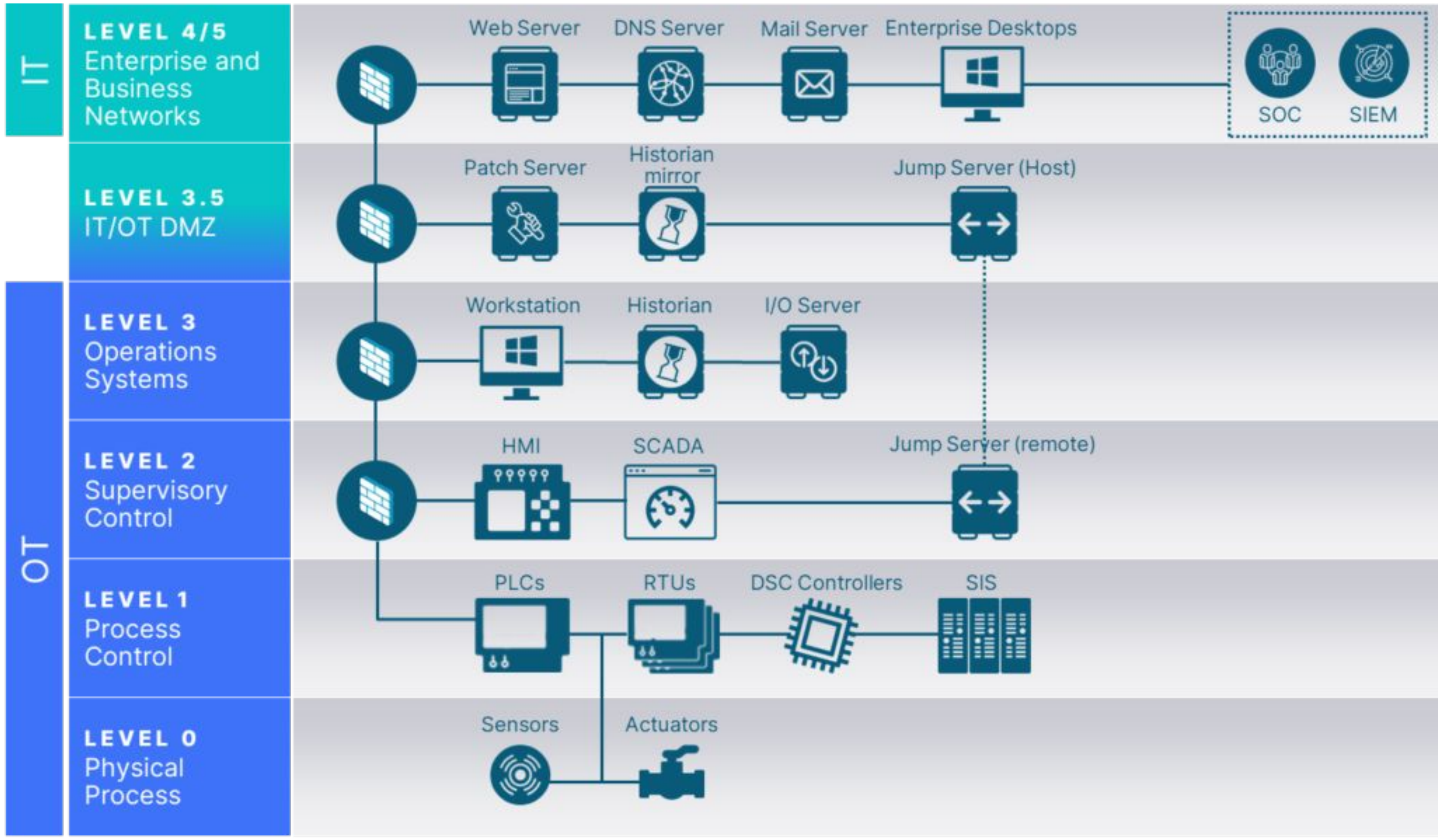
IoT infrastructure refers to the hardware and software architecture that supports the deployment and management of IoT devices and sensors.

Edge computing refers to processing data closer to where it is generated (at the "edge" of the network), reducing reliance on cloud computing.



Figure：Edge computing architecture overview
Source：The research team

Industry standard – the Purdue model


IT
LEVEL 4/5
Enterprise and Business Networks
Web Server
DNS Server
Mail Server
Enterprise Desktops
SOC
SIEM
LEVEL 3.5
IT/OT DMZ
Patch Server
Historian mirror
Jump Server (Host)
OT
LEVEL 3
Operations Systems
Workstation
Historian
I/O Server
LEVEL 2
Supervisory Control
HMI
SCADA
Jump Server (remote)
LEVEL 1
Process Control
PLCs
RTUs
DSC Controllers
SIS
LEVEL 0
Physical Process
Sensors
Actuators