DA 231o: Data Engineering at Scale

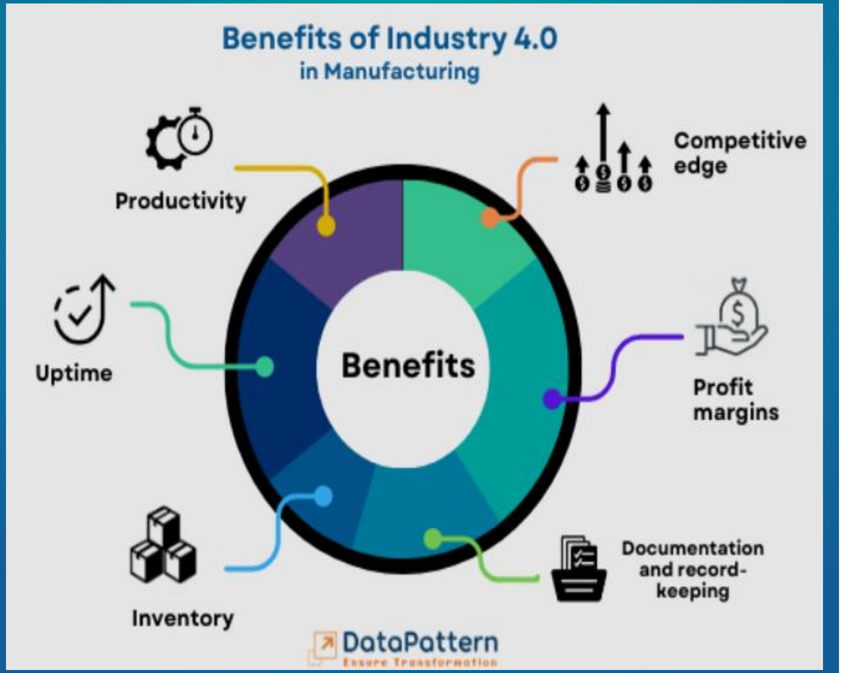
Course Project Presentation

IoT Driven Real Time Predictive Maintenance

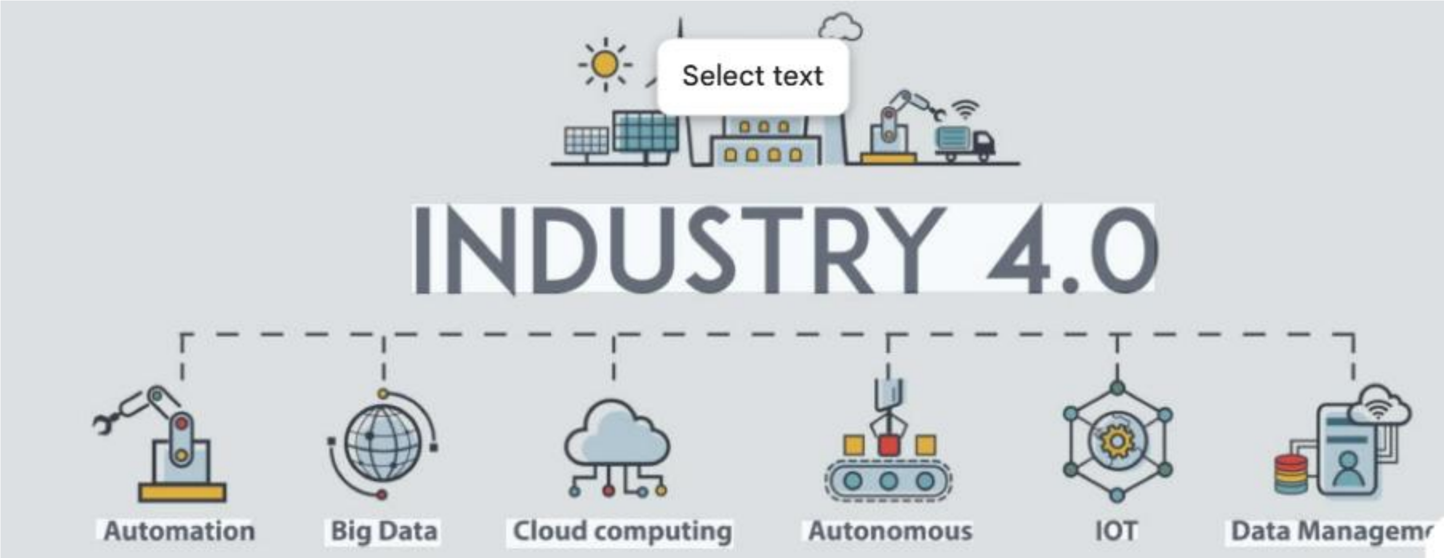
Abhilasha Kawle, abhilashak@iisc.ac.in
Sangram Kumar Y, sangramyerra@iisc.ac.in
Siddaraju D H, siddarajuh@iisc.ac.in
Venturi Naveen, naveen1@iisc.ac.in

Image source: Internet

Problem Definition



- Industry 4.0 Boosts
  - Faster, high quality production with flexibility and hi-efficiency in process
- Promotes use of technologies such as IoT, Big data, Automation
  - Rela-time monitoring – reduce downtime, Ops cost
  - Faster decisions through data driven insights - Optimization of Operations

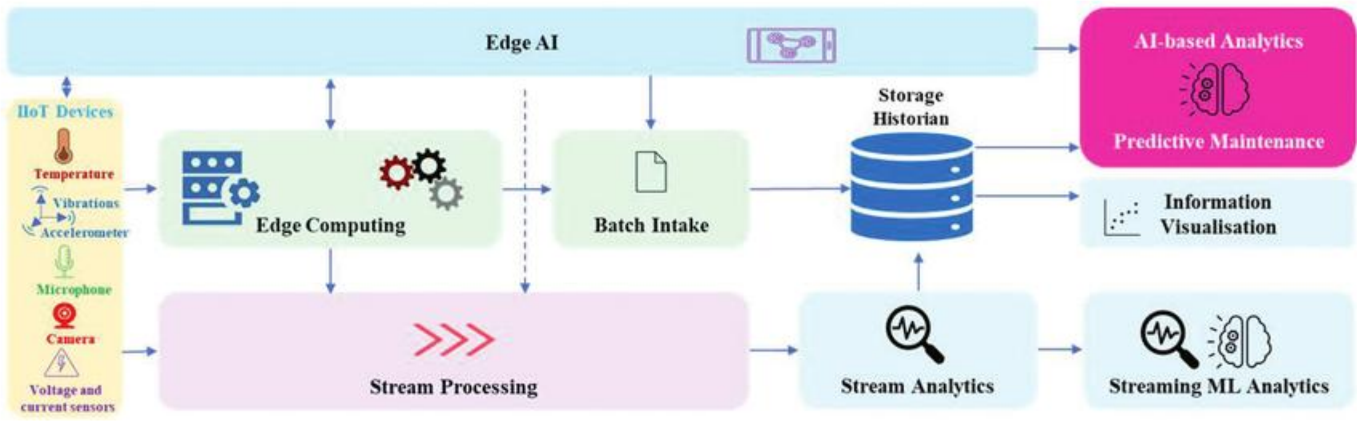


Problem Motivation

Predictive maintenance is essential in asset-heavy industries

Equipment failure leads to severe operational, financial losses

- Thousands of IoT sensors → high-speed, real-time data streams
- AI and machine learning models analyze this continuous data to detect anomalies and predict failures before they occur
- Growing adoption across diverse industries, predictive maintenance has become a **scalable Big Data storage and processing challenge**.



Project Goals

Scalable Architecture for Storage, streaming and Processing IoT sensors data

Application targeted : **Water pump health monitoring**

- Data Set : https://www.kaggle.com/datasets/anseldsouza/water-pump-rul-predictive-maintenance?select=rul_hrs.csv
- Dataset with 50 sensors on each water pump. And the data is tracked for 7 machines
- Predicting healthy hours of the water pump before its failure.
- Goal : set “FAILURE” alert at 24th hours of remaining healthy.

Proposed Methodology

- Model
  - Data Preprocessing
  - ML training
- Kafka Env
  - Kafka producers streaming IoT sensor data
  - Scales Vertically and Horizontally
- Spark Cluster
  - Reads ML Model and Input for the prediction
  - Scales as per load
- Dashboard
  - Real time monitoring of healthy hours
  - Publish predicted healthy hours and alerts

- Models from spark ML
- Outputs
  - Regression output – ‘hours_remaining_healthy”
  - Classifier output – “machine_failure_in_24hrs”
- Data Models
  - Linear Regression,
  - Randomforest regressor,
  - GBTRegressor,
  - Randomforest classifier
- Success Matrix
  - R2 > 95% for Regression;
  - Accuracy > 95% for Classifier
- Kafka Cluster of Producer for streaming the IoT sensor data
- Spark Cluster consumes the kafka ingesting data, & does ML model inferencing
- Results published on Dashboard predicted healthy hours and alters for any machine failure in 24hrs

Implementation Plan

Tools:

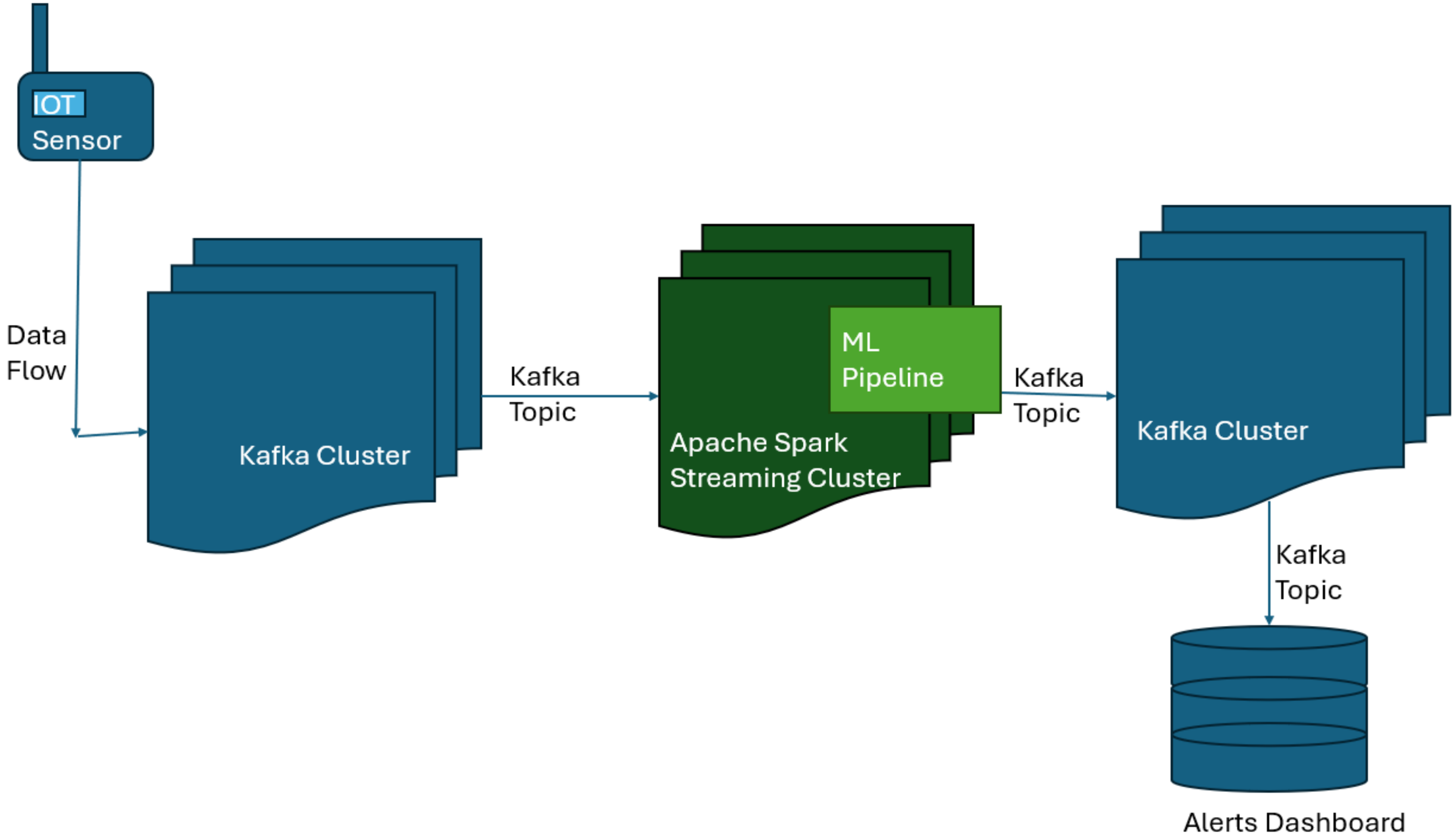
Spark

SparkML

Kafka

SparkStream

- Architecture Block Diagram



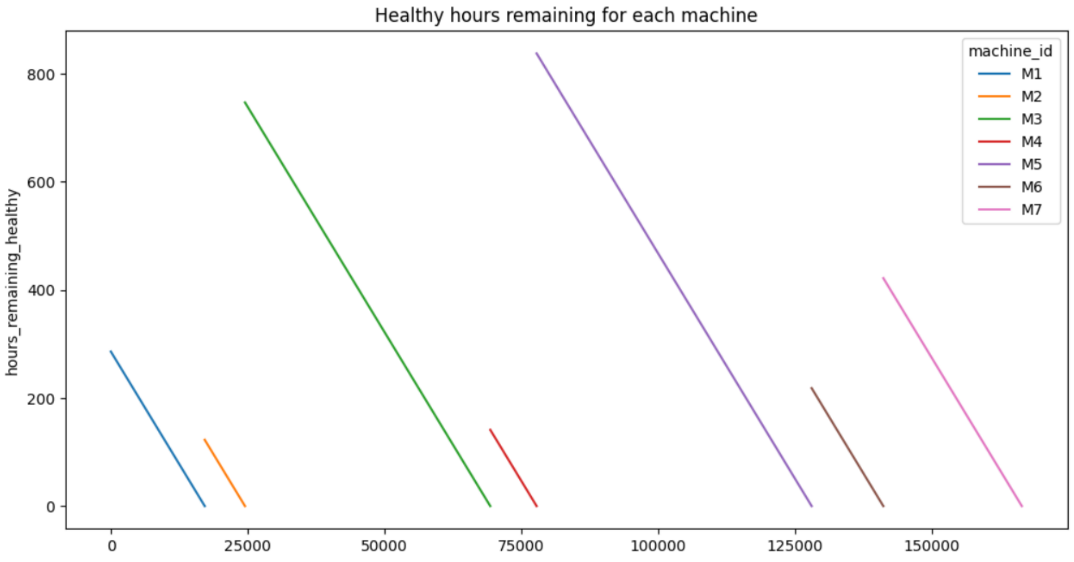
Step 1: Data Collection and Preparation

- Data converted to parquet (columnar format) for faster processing
  - Data Compression of parquet leveraged
- The data inspected for NULLS – No NULLs were present
- The datatype for sensors was “string”, Converted to ‘double’ for use into ML training for prediction regression.
- All the sensor data was rounded to 3 decimals for easy readability and computation
- Added “machine_failure_in_24hrs” = 0 for hours_remaining_healthy > 24hrs

  = 1 for hours_remaining_healthy <=24hrs

Step 2: Data Exploration

- Data for “Healthy hours remaining” for M1 to M7
  - Linear decrease from normal hrs to ‘0’
- Correlation to Target column - .corr([col])
  - Maximum correlation (-0.27) - sensor 13 with negative sign.
  - This means the sensor data is increasing as remaining healthy hours decrease.
- Skew was analyzed – skewness()
  - Some sensors showed strong skewness > 10
  - But the correlation factor was low, the skew transformation was not applied



Top 5 Sensors with high correlation	Correlation factor	Skew
Sensor 13	(-)0.276	1.55
Sensor 29	0.225	-0.9
Sensor 37	0.177	-0.22
Sensor 41	0.1439	8.5
Sensor 05	(-)0.136	-2.688

Step 3 : Model Development

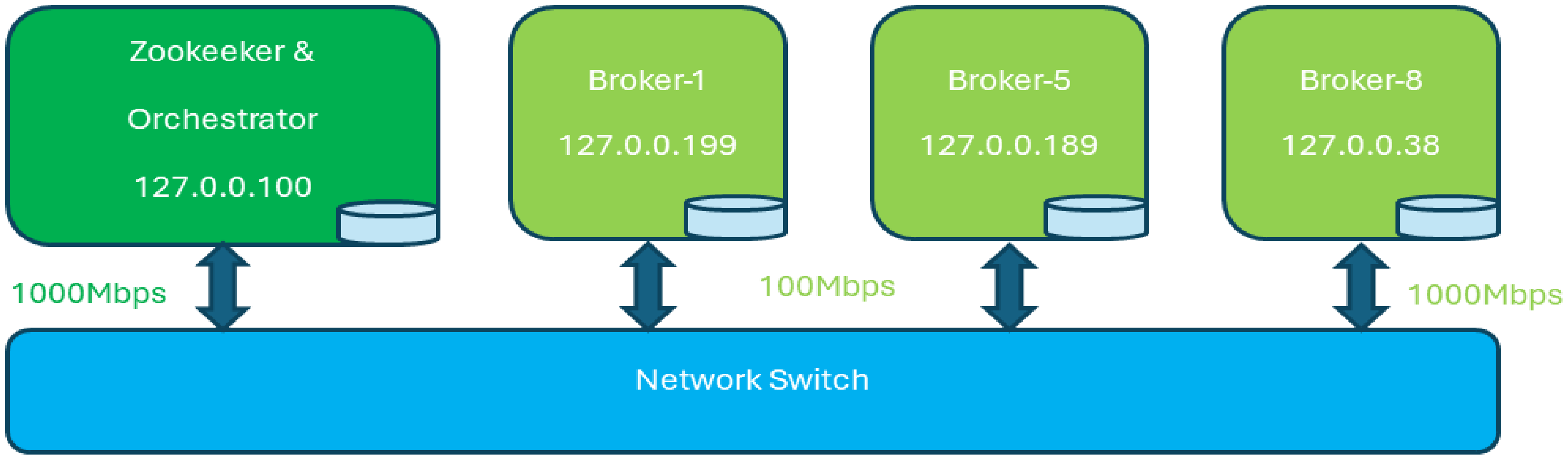
- ML Transformation
  - Vector assembler of numerical columns containing sensor data.
  - There is no categorical feature in this dataset.
  - Standard scaler on vector – For normalizing the sensor data as we don't have information on type of sensors in this dataset.

- Model Performance

Spark ML Model	R2 / Accuracy
Ridge L2 Regression	0.444
Random Forest Regressor (Trees=50, Depth=12)	0.991
GBT Regressor (Depth=5)	0.895
Random Forest classifier (Trees=50, Depth=12)	0.997

- For the given dataset, the target column shows linear characteristics. This could be the reason for R2/accuracy ~ 0.99
- This sparkML model was saved for loading into kafka topics for prediction over streaming data

Kafka Cluster Setup



Systemd service for resiliency

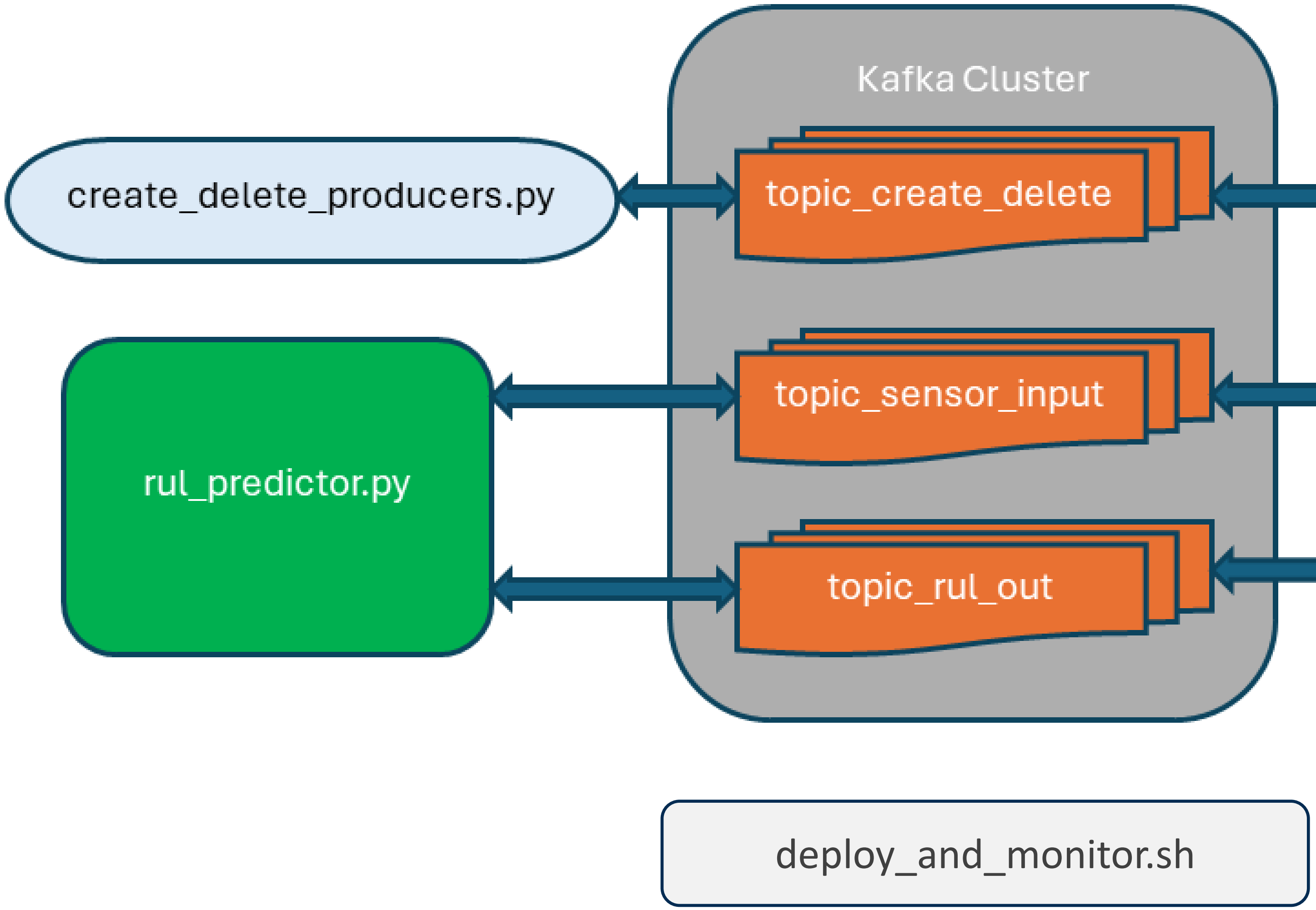
tmux session for live monitoring and debug

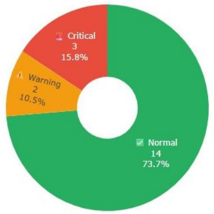
7days log retention

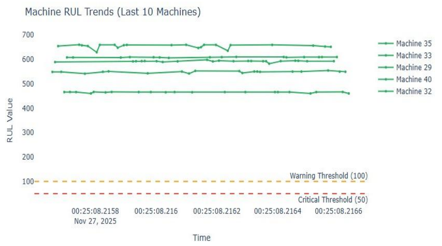
Replication factor = 3

Orchestrator: deploy_and_ monitor.sh (or K8s)

Kafka Cluster & Environment for message Streaming







Kafka producer-consumer: App design strategy

- Producer
  - A standalone multi-threaded deployable instance for each or more motor producer
  - Connects to the Kafka cluster
  - Check if the identified partition ID already exists, if not, create a new one
  - Collate and send the sensor data

- Consumer
  - A multithreaded deployable application
  - Subscribes to the producer topic
  - Registers for rebalance listener events
  - Registers to a common consumer group
  - Partition alloc: spans new threads
  - Partition revoke: Stops the running thread
  - max_poll_record=1 for realtime

- Scaling & resiliency
  - Vertical scaling: Exploits threading
  - Horizontal scaling: Parallel instance
  - Resiliency: App relaunch from an orchestrator

Spark Streaming

- Read Kafka Streaming Source
- Replace null values with 0 for the batch
- Assemble Features
- Load Models inside batch from HDFS
- Predict using Model Transform
- Clean Columns in DF from both Models
- Join the Predictions
- Create Outgoing kakfa topic if not present
- Write Predictions to Kafka Topic

Spark Scalability & Fault Tolerance

- Multi-machine worker deployment enables parallel processing and improved system resilience
- ML Models deployed as re-usable . Can be upgraded without system downtime
- Platform easily configures to diverse industry requirements and integrates new sensor types without architectural changes
- Computation is horizontally Scalable across worker nodes
- Spark Streaming & Kafka ensures at-least-once delivery
- Checkpointing is enabled for recovery

Spark Scaling

Spark 3.5.7 **Spark Master at spark://**

URL: spark://
Alive Workers: 3
Cores in use: 6 Total, 6 Used
Memory in use: 12.0 GiB Total, 3.0 GiB Used
Resources in use:
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (3)

Worker Id	Address	State	Cores	Memory	Resources
worker-2025112705434		ALIVE	2 (2 Used)	4.0 GiB (1024.0 MiB Used)	
worker-20251127111348-		ALIVE	2 (2 Used)	4.0 GiB (1024.0 MiB Used)	
worker-20251127111407-		ALIVE	2 (2 Used)	4.0 GiB (1024.0 MiB Used)	

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20251127111420-0000 (kill)	PredictiveMaintenance	6	1024.0 MiB		2025/11/27 11:14:20	iisc_naveen	RUNNING	26 s

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

- Throughput: 400 rows/sec
- Batch Latency: 1k->2.5s
- Executor Memory: 1GB/ Executor

Dashboard Overview

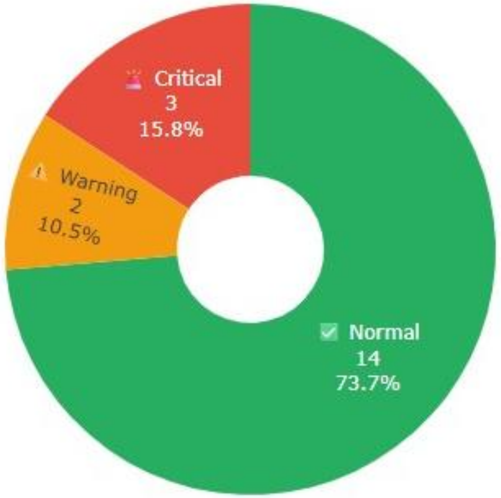
- **Purpose**

Real-time monitoring and visualization of Remaining Useful Life (RUL) predictions for industrial machine fleet, enabling proactive maintenance decisions and operational efficiency.

- **Key Features**
  - **Real-time Data Ingestion:** Consumes Kafka messages every 3 seconds
  - **Fleet Health Visualization:** Interactive pie chart showing critical/warning/normal status distribution
  - **Trend Analysis:** RUL progression over time for individual machines
  - **Prioritized Alerts:** Color-coded table sorted by criticality (lowest RUL first)
  - **Automated Status Classification:** Critical (<50), Warning (50-100), Normal (≥100)

Dashboard Components

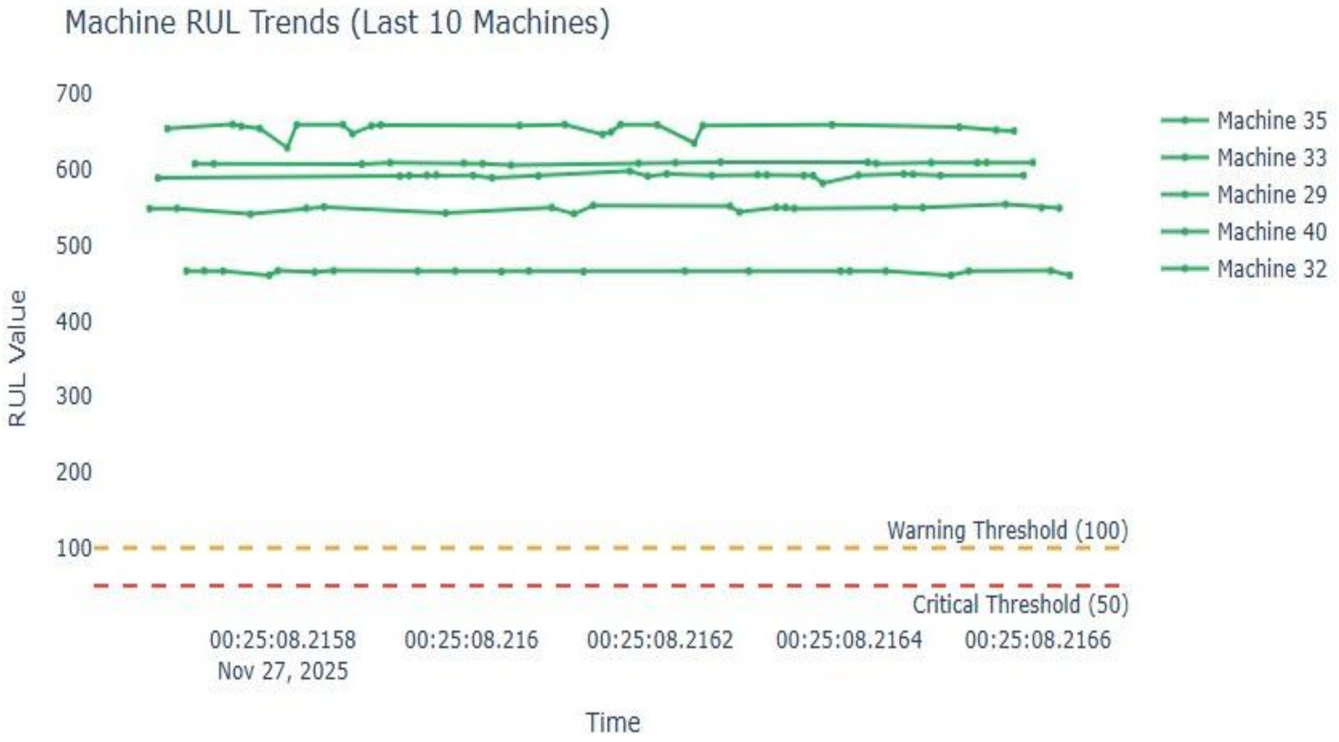
- **Fleet Health Status (Pie Chart)**



- **Visual**: chart with status distribution
- **Colors:** Critical, Warning, Normal
- **Special Case:** Full green when all machines normal

- **Machine RUL Trends (Line Chart)**

  - **Visual:** Multi-line time series (last 10 machines)
  - **Features:** Color-coded by status, threshold lines at 50 & 100
  - **Purpose:** Identify deteriorating vs improving machines



Dashboard Components

- ## Machine Status Table

  •**Data**: Machine ID, RUL value, status, timestamp
  •**Sorting**: Automatic by RUL (critical first)
  •**Highlighting**: Row colors match status severity

Machine Status Details

⇕Machine ID	⇕RUL Value	⇕Status	⇕Last Update
Machine 23	32.84	critical	2018-04-17 07:40:00
Machine 21	51.67	warning	2018-04-15 22:30:00
Machine 22	55.09	warning	2018-04-16 14:46:00
Machine 20	61.63	warning	2018-04-15 05:47:00
Machine 40	473.41	normal	2018-04-29 02:19:00
Machine 38	515.32	normal	2018-04-27 17:37:00
Machine 37	516.2	normal	2018-04-27 00:27:00
Machine 39	522.39	normal	2018-04-28 10:17:00
Machine 35	550.39	normal	2018-04-25 15:35:00

- ## Statistics Bar

  •**Metrics**: Total machines, critical count, warning count, message count
  •**Updates**: Real-time every 3 seconds

📊 Total: 19 | 🚨 Critical: 2 | ⚠️ Warning: 2 | ✅ Normal: 15 | 📈 Messages: 5959 | ⏰ Last: 00:20:56

Role and Responsibilities

- Abhilasha Kawle:
  - Data Preprocessing
  - ML model training
- Siddaraju D H:
  - KAFKA Cluster [Horizontal & Vertical Scale Setup]
  - Transmit Sensor data to Spark Engine
- Venturi Naveen:
  - Spark Cluster Setup
  - Spark Pre-procesing and ML Inferencing
- Sangram Kumar Yerra:
  - Efforts to get Kafka setup in Collab
  - Dashboard

Conclusions

- Business Value

- **Cost Savings**: Prevent catastrophic failures
- **Operational Efficiency**: Real-time visibility into fleet health
- **Data-Driven Decisions**: Replace reactive with predictive maintenance
- **Asset Life Extension :** Through proper maintenance timing