***Sensor Failure Prediction and Maintenance Analytics***

***1. Introduction***

In the context of modern industrial systems, predictive maintenance has emerged as a crucial technique to reduce unplanned equipment downtime, optimize maintenance costs, and enhance operational efficiency. Instead of relying on fixed schedules, predictive maintenance analyzes sensor data from machinery to anticipate failures before they happen.

This project focuses on simulating a real-world predictive maintenance scenario using the CMAPSS (Commercial Modular Aero-Propulsion System Simulation) dataset – a widely recognized benchmark dataset developed by NASA. The dataset contains multivariate time-series measurements from jet engines, including temperature, pressure, and vibration-related sensors.

By applying rule-based anomaly detection, signal processing techniques (such as rolling averages and z-scores), and machine learning modeling (XGBoost), the goal of this project is to detect early signs of system failure. The approach aligns closely with real use cases in logistics, warehouse operations, and IoT-enabled system monitoring — similar to systems operated by technology-driven logistics platforms like Lazada.

***2. Objectives***

The main objectives of this project are as follows:

1. **Sensor Behavior Analysis:**  
   Investigate and visualize the behavior of critical sensors (e.g., vibration, temperature, pressure) across time for each engine unit to identify degradation trends.
2. **Anomaly Detection:**  
   Apply statistical techniques such as rolling mean and z-score to detect abnormal sensor behavior, and flag potential failure signals before they escalate.
3. **MTTF Estimation:**  
   Estimate the Mean Time To Failure (MTTF) per unit to understand operational life cycles and to provide planning input for maintenance scheduling.
4. **Machine Learning for Failure Prediction:**  
   Train a binary classification model (XGBoost) to predict whether a given time point represents an anomaly, using multivariate sensor inputs.
5. **Visualization & Reporting:**  
   Create intuitive visualizations such as sensor timelines, heatmaps, and distribution plots to communicate insights effectively to technical and non-technical stakeholders.
6. **Real-world Application Simulation:**  
   Simulate a real-world use case in logistics and operations, providing a data-driven foundation for preventive system control strategies in environments such as automated warehouses or IoT-enabled facilities.

***3. Dataset Description***

The dataset used in this project is derived from the **CMAPSS (Commercial Modular Aero-Propulsion System Simulation)** dataset, originally developed by NASA and made publicly available through Kaggle. It simulates degradation behavior in jet engines based on various operating conditions and sensor signals.

**📦 Source:**

* **Name:** CMAPSS Jet Engine Simulated Data
* **Link:** https://www.kaggle.com/datasets/palbha/cmapss-jet-engine-simulated-data

**📊 Dataset Highlights:**

* **Engine Units:** 100+
* **Time Steps (Cycles):** Over 20,000 rows of multivariate time-series data
* **Variables:**
  + unit\_number: Engine ID
  + cycle: Operating time step
  + op\_setting\_1 to op\_setting\_3: Operational settings
  + sensor\_1 to sensor\_21: Real-valued sensor readings for temperature, vibration, pressure, etc.

**✅ Key Features Used:**

In this project, only a subset of the most relevant sensors was selected for analysis and modeling:

* **sensor\_2** – typically indicates temperature fluctuations
* **sensor\_3** – often linked to core pressure readings
* **sensor\_7** – typically reflects vibration or engine speed

These sensors were chosen due to their high correlation with degradation behavior and potential failure patterns as observed in the initial data exploration phase.

📌 *The raw data was originally provided as a .txt file and was preprocessed into a structured .csv format with clearly named columns for further analysis.*

***4. Methodology***

This section outlines the step-by-step methodology followed in the project, from data preprocessing to anomaly detection and machine learning modeling.

**4.1 Data Preprocessing**

* **File Conversion:**  
  The original data in .txt format was converted to .csv using pandas.read\_csv() with whitespace separators.
* **Column Naming:**  
  26 columns were assigned meaningful names, including:
  + unit\_number, cycle, op\_setting\_1 to op\_setting\_3, and sensor\_1 to sensor\_21
* **Feature Selection:**  
  Sensor signals with the highest variance and strongest visual trends (e.g., sensor\_2, sensor\_3, sensor\_7) were selected for analysis.
* **Rolling Mean & Z-Score Calculation:**  
  A 10-cycle rolling mean was calculated for smoothing. Z-score normalization was applied to highlight deviations from the mean.

df['sensor\_2\_roll\_mean'] = df.groupby('unit\_number')['sensor\_2'].transform(lambda x: x.rolling(window=10, min\_periods=1).mean())

df['sensor\_2\_z'] = (df['sensor\_2'] - df['sensor\_2'].mean()) / df['sensor\_2'].std()

**4.2 Anomaly Flagging Logic**

* A data point was flagged as an anomaly if the **absolute z-score > 2.5** for any of the selected sensors.
* If **3 or more sensor z-scores exceeded the threshold** within a 10-cycle window, the unit was considered to be at high risk of failure.

df['anomaly\_flag'] = np.where(

(df['sensor\_2\_z'].abs() > 2.5) |

(df['sensor\_3\_z'].abs() > 2.5) |

(df['sensor\_7\_z'].abs() > 2.5), 1, 0)

**4.3 Data Visualization**

Multiple plots were generated to uncover insights and communicate patterns:

| **Plot Type** | **Purpose** |
| --- | --- |
| **Sensor timeline** | Show sensor trends over cycles and highlight anomalies |
| **Heatmap** | Reveal correlations between sensor readings |
| **Anomaly bar chart** | Count of anomalies per engine unit |
| **MTTF distribution** | Show cycle lifespan per engine |

✅ *These visuals were saved in the /img/ directory and included in this report.*

**4.4 Machine Learning: XGBoost Classifier**

A machine learning model was trained to predict whether a time step is anomalous (anomaly\_flag = 1):

* **Model Used:** XGBClassifier
* **Features:** All 21 sensor columns
* **Target:** anomaly\_flag
* **Train-Test Split:** 80/20
* **Evaluation Metrics:** Accuracy, Precision, Recall, F1-score, Confusion Matrix

from xgboost import XGBClassifier

model = XGBClassifier()

model.fit(X\_train, y\_train)

The trained model achieved solid predictive performance and produced a well-structured confusion matrix.

***5. Key Findings***

The project yielded the following key insights from both statistical analysis and machine learning modeling:

**5.1 Sensor Behavior and Anomaly Detection**

* **Sensor\_2** exhibited early signal drift patterns in multiple engine units, with anomaly points appearing up to **30 cycles before final failure**.
* **Sensor\_3** and **Sensor\_7** also showed significant deviations during late stages of engine degradation.
* By applying a z-score threshold of ±2.5, the system effectively flagged high-risk periods across multiple units.
* Anomaly concentration increased in the final 15% of each unit's operational life, validating the detection logic.

**5.2 Mean Time To Failure (MTTF)**

* The average MTTF across engine units was approximately **197 cycles**.
* Units that triggered a high number of anomaly flags tended to have shorter operational spans.

A graph with blue lines

AI-generated content may be incorrect.

**5.3 Correlation Insights**

* Sensor correlation analysis revealed that certain signals (e.g., sensor\_2, sensor\_4, sensor\_10) had strong positive correlations, which could be leveraged in multivariate analysis.
* *A screenshot of a computer generated image

  AI-generated content may be incorrect.*The heatmap also helped to eliminate redundant sensors from the model pipeline.

**5.4 Anomaly Distribution**

* Over **1,000 anomaly points** were detected across all units.
* A graph of orange lines

  AI-generated content may be incorrect.Units 5, 17, and 42 had the **highest concentration of anomaly flags**, suggesting a more aggressive degradation trend.

**5.5 Machine Learning Model Results**

* The **XGBoost classifier** achieved the following metrics on the test set:
  + **Accuracy:** 89.4%
  + **Precision:** 84.2%
  + **Recall:** 78.9%
  + **F1-score:** 81.5%
* The **confusion matrix** confirmed strong performance with relatively few false positives.

A graph with numbers and a blue square

AI-generated content may be incorrect.

***6. Business Impact & Real-World Application***

Predictive maintenance is a critical operational strategy in industries where unplanned downtime can lead to significant financial losses and service disruptions. The insights and techniques developed in this project have several real-world applications across logistics, manufacturing, and system operations.

**6.1 Operational Benefits**

| **Area** | **Impact** |
| --- | --- |
| **Warehouse Systems** | Detect malfunctioning conveyors, sorters, or robotic arms in advance to avoid delays in fulfillment and shipping. |
| **Fleet Maintenance** | Monitor vehicles or delivery assets using embedded sensors to schedule repairs before breakdowns occur. |
| **Automated Quality Control** | Use anomaly signals to flag equipment drift in real time, improving product consistency. |
| **IT Infrastructure Monitoring** | Adapt signal-drift logic for CPU temperature, network traffic, or memory load monitoring. |

**6.2 Quantifiable Advantages**

* **⚙️ Reduced Downtime:**  
  Early failure detection enables timely repairs, minimizing revenue loss from halted operations.
* **💸 Lower Maintenance Cost:**  
  Data-driven repair timing reduces unnecessary routine checks and avoids catastrophic breakdowns.
* **📈 Improved Efficiency:**  
  Operations run with higher stability when risk zones are proactively managed.
* **📊 Transparent Monitoring:**  
  Visual dashboards and automated alerts improve communication between engineers, operators, and decision-makers.

***7. Limitations & Future Work***

While the project successfully implemented anomaly detection and predictive modeling using simulated sensor data, several limitations remain. These serve as opportunities for future development and real-world deployment.

**7.1 Limitations**

* **Synthetic Dataset:**  
  The CMAPSS dataset is simulated and lacks real-time noise, hardware constraints, and multi-modal sensor integration (e.g., sound, vibration waveform).
* **Binary Simplification:**  
  The current model only flags whether a time step is anomalous or not. In reality, degradation exists on a spectrum and may need multi-class or continuous estimation.
* **Static Thresholding:**  
  The anomaly detection logic uses a fixed z-score threshold (e.g., ±2.5), which may not adapt well across different operating environments or sensor types.
* **Limited Features:**  
  Only a subset of sensor readings was used. Operational settings and contextual metadata (e.g., maintenance history) were not incorporated.
* **No Real-Time System:**  
  The project works offline. There is no deployed dashboard or API pipeline for real-time monitoring or feedback.

**7.2 Future Work**

| **Direction** | **Description** |
| --- | --- |
| **Remaining Useful Life (RUL) Modeling** | Implement regression models to predict the number of cycles left before failure. This would enhance decision-making in repair scheduling. |
| **Multivariate Anomaly Detection** | Use models like Isolation Forest or Autoencoders for deeper unsupervised analysis of hidden patterns. |
| **Streaming Dashboard** | Build a real-time dashboard using Streamlit or Dash to visualize anomalies as they occur. |
| **Explainable AI (XAI)** | Apply SHAP values to explain why the model predicts failure at certain points, increasing trust and interpretability. |
| **Edge Deployment** | Test lightweight versions of the anomaly logic for deployment in embedded systems or warehouse IoT environments. |

This section ensures that the project is not just a finished assignment, but a scalable technical prototype with potential to evolve into a production-grade solution.

***8. Conclusion***

This project demonstrates a practical and technically sound approach to predictive maintenance through anomaly detection in multivariate sensor data. By leveraging statistical techniques, data visualization, and machine learning modeling, it successfully identifies early warning signs of engine degradation and failure.

Key achievements include:

* Constructing a full data pipeline from raw .txt to structured .csv
* Applying rolling and z-score methods for anomaly detection
* Visualizing sensor drift and performance trends across engine units
* Training an XGBoost classifier to predict failure risk with high accuracy
* Connecting the results to real-world applications in logistics and warehouse systems

From both a learning and application standpoint, this project enhances the author’s skills in Python programming, data wrangling, machine learning, and operational analytics. It also demonstrates readiness for internship or junior data roles in companies like Lazada, where system optimization and data-driven decision-making are core components of their technology operations.

The entire source code, visual assets, and structured documentation are publicly available on GitHub for further reference and improvement.

***9. References***

1. **CMAPSS Jet Engine Dataset – NASA (via Kaggle):**  
   https://www.kaggle.com/datasets/palbha/cmapss-jet-engine-simulated-data
2. **XGBoost Documentation:**  
   <https://xgboost.readthedocs.io>
3. **Pandas – Data Manipulation Library:**  
   https://pandas.pydata.org/
4. **Matplotlib & Seaborn – Data Visualization Libraries:**  
   <https://matplotlib.org/>  
   https://seaborn.pydata.org/
5. **Google Colab – Cloud Notebook Environment:**  
   https://colab.research.google.com/
6. **SHAP Explainability (optional for future work):**  
   <https://shap.readthedocs.io/>