OCTOBER 2025

PREDICTIVE MAINTENANCE USING NASA TURBOFAN ENGINE DATASET

Independent Project

Prepared By

Jagadeesh.N

+91 9677013041
jagadeesh.n10d@gmail.com
www.linkedin.com/in/jagadeesh4696/
github.com/jagan969646/

Certificate of Completion

This is to certify that Jagadeesh N, a Final-Year student of Bachelor of Business Administration (BBA), has successfully completed the independent project titled:

"Predictive Maintenance Using NASA Turbofan Engine Dataset"

This project was undertaken as a self-directed initiative to apply data science and business analytics techniques to a real-world industrial challenge. The work presented in this report is original and reflects the author's understanding of predictive maintenance, machine learning, and reliability engineering.

The project demonstrates hands-on proficiency in data preprocessing, feature engineering, model development (XGBoost, LSTM, CNN-LSTM), and performance evaluation using the NASA CMAPSS FD001 dataset.

This report is submitted as part of personal portfolio development and career readiness in the field of data analytics and AI-driven business solutions.

Date: October 2025

Place: Chennai, Tamil Nadu, India

Signature of Student: ___Jagadeesh.N__

Table of content

1.Tittle

PREDICTIVE MAINTENANCE USING NASA TURBOFAN ENGINE DATASET

Independent Project by Jagadeesh N

Final-Year BBA Student, SRM Institute of Science and Technology, Chennai, India

Certified in IBM Data Science, Google Advanced Analytics, and Wharton Business Analytics , IBM AI Product management, Google Project management

LinkedIn: https://www.linkedin.com/in/jagadeesh4696/

GitHub: https://github.com/jagan969646/

Date: October 2025

2.Abstract

Predictive maintenance is a transformative approach that leverages data-driven insights to anticipate equipment failures before they occur. This independent project focuses on estimating the Remaining Useful Life (RUL) of turbofan engines using the NASA CMAPSS FD001 dataset. The dataset simulates run-to-failure scenarios for 100 engines, each monitored by 21 sensors and 3 operational settings, providing a rich foundation for machine learning and deep learning applications.

The project begins with thorough data preprocessing, including the removal of constant sensors, handling of missing and infinite values, and calculation of RUL for each engine cycle. Feature engineering plays a central role, with rolling statistics (mean, standard deviation, slope) computed over a moving window to capture degradation patterns. These features are used to train and evaluate multiple models: XGBoost for baseline performance, LSTM for sequence modeling, and CNN-LSTM for hybrid temporal-spatial learning.

Model performance is assessed using RMSE and MAE metrics, with visualizations comparing predicted and actual RUL values. Challenges such as noisy sensors, gradient instability, and overfitting are addressed through techniques like dropout, gradient clipping, and early stopping. The results demonstrate that deep learning models outperform traditional methods in capturing complex temporal dependencies, with CNN-LSTM showing the best overall accuracy.

This project highlights the practical significance of predictive maintenance in aerospace and industrial domains. It also lays the groundwork for future enhancements, including attention-based models, hyperparameter tuning, and deployment pipelines using cloud infrastructure. By combining business analytics with technical depth, the work showcases how real-world datasets can be transformed into actionable insights for reliability engineering.

3.Introduction

3.1 Maintenance Strategies in Industry

Maintenance is a critical function in industrial operations, directly influencing equipment reliability, operational efficiency, and safety. Traditionally, industries have relied on three primary maintenance strategies:

- **Reactive Maintenance**
  This approach involves repairing equipment only after a failure occurs. While simple and cost-effective in the short term, it often leads to unexpected downtime, increased repair costs, and safety risks. It is generally unsuitable for high-value or mission-critical systems.
- **Preventive Maintenance**
  Scheduled servicing based on time intervals or usage metrics. Preventive maintenance reduces the likelihood of failure but can result in unnecessary maintenance actions and resource inefficiencies, especially when components are replaced before reaching the end of their useful life.
- **Predictive Maintenance**
  The most advanced strategy, predictive maintenance leverages sensor data and machine learning models to forecast failures before they happen. This enables timely interventions, minimizes downtime, and optimizes resource allocation. Predictive maintenance is increasingly adopted in sectors such as aerospace, automotive, manufacturing, and energy, where reliability and safety are paramount.

3.2 Importance of Remaining Useful Life (RUL)

Remaining Useful Life (RUL) refers to the estimated time before a machine or component fails. Accurate RUL prediction is central to predictive maintenance and offers several benefits:

- **Aerospace:**
  Prevents in-flight engine failures and supports optimized servicing schedules for aircraft.

- **Automotive:**
  Enhances vehicle reliability, reduces roadside breakdowns, and improves customer satisfaction.
- **Industrial Machinery:**
  Minimizes production halts, improves asset utilization, and reduces maintenance costs.

RUL estimation enables organizations to shift from reactive to data-driven decision-making. It supports inventory planning, workforce scheduling, warranty management, and long-term asset health monitoring.

3.3 Role of IoT and Industry 4.0

The emergence of **Industry 4.0** and the **Internet of Things (IoT)** has revolutionized maintenance strategies by enabling real-time monitoring and intelligent decision-making:

- **Sensor Integration:**
  Machines are equipped with sensors that continuously monitor temperature, pressure, vibration, and other operational metrics.
- **Real-Time Analytics:**
  Sensor data is streamed to cloud platforms for immediate analysis, anomaly detection, and predictive modeling.
- **Digital Twins:**
  Virtual replicas of physical systems allow simulation, diagnostics, and forecasting of equipment behavior under various conditions.

These technologies empower predictive maintenance by providing rich datasets and enabling scalable, intelligent systems. The convergence of edge computing, cloud infrastructure, and AI has made it feasible to deploy predictive models in real-time environments.

3.4 NASA CMAPSS Dataset

The **Commercial Modular Aero-Propulsion System Simulation (CMAPSS)** dataset, developed by NASA, is a benchmark for RUL prediction research. It simulates turbofan engine degradation under various operating conditions and fault modes.

- **FD001 Subset:**
  This project uses FD001, which includes:
- 100 engines
- 1 fault mode
- Constant operating conditions
- 21 sensor readings and 3 operational settings

Each engine runs until failure, and the dataset captures Each engine runs until failure, and the dataset captures sensor readings at each cycle. The goal is to predict how many cycles remain before failure, based on historical sensor data. FD001 is ideal for modeling single-condition degradation and serves as a controlled environment for testing predictive algorithms.

3.5 Project Objectives

This independent project aims to:

- Build a robust pipeline for RUL prediction using the FD001 subset of CMAPSS.
- Engineer rolling statistical features (mean, standard deviation, slope) to capture degradation trends.
- Train and evaluate multiple models: XGBoost for baseline performance, LSTM for sequence modeling, and CNN-LSTM for hybrid temporal-spatial learning.
- Address challenges such as noisy sensors, NaNs in rolling features, and model overfitting.
- Visualize predictions and compare model performance using RMSE and MAE.
- Explore future directions including attention-based models, hyperparameter tuning, and deployment pipelines using cloud infrastructure.

By combining business analytics with technical depth, this project demonstrates how predictive maintenance can be implemented using real-world datasets and scalable machine learning techniques.

4. Literature Review

4.1 Predictive Maintenance in Research

Predictive maintenance has gained significant attention in recent years due to its potential to reduce downtime, optimize resource usage, and improve safety. Researchers have explored various approaches, ranging from statistical models to advanced machine learning and deep learning techniques.

Traditional methods such as linear regression and support vector machines (SVM) offer interpretability but often struggle with complex temporal dependencies. Ensemble methods like Random Forest and Gradient Boosting (e.g., XGBoost) have shown improved accuracy and robustness, especially in handling noisy sensor data.

Deep learning models, particularly Long Short-Term Memory (LSTM) networks, have emerged as powerful tools for sequence modeling. LSTM's ability to capture long-term dependencies makes it suitable for Remaining Useful Life (RUL) prediction. More recently, hybrid architectures like CNN-LSTM and Transformer-based models have demonstrated superior performance by combining spatial and temporal feature extraction.

4.2 NASA CMAPSS Dataset in Literature

The CMAPSS dataset, developed by NASA, is widely used as a benchmark for RUL prediction. It consists of four subsets (FD001–FD004), each simulating different fault modes and operating conditions. FD001, used in this project, features a single fault mode and constant operating conditions, making it ideal for controlled experimentation.

Studies using CMAPSS have explored various modeling strategies:

- **Saxena et al. (2008)** introduced the dataset and proposed early RUL estimation techniques using regression and similarity-based methods.
- **Zhao et al. (2019)** applied LSTM networks to capture temporal patterns in sensor data, achieving improved RMSE scores.
- **Li et al. (2021)** combined CNN and LSTM to extract both local sensor trends and long-term dependencies, outperforming standalone models.

- **Yıldırım & Afşer (2024)** explored attention mechanisms to enhance interpretability and prediction accuracy.

These studies highlight the importance of feature engineering, model architecture, and data preprocessing in achieving reliable RUL predictions.

4.3 Gaps and Opportunities

Despite extensive research, several challenges remain:

- **Sensor Noise and Redundancy**: Many sensors in CMAPSS are either constant or highly correlated, requiring careful selection and preprocessing.
- **Rolling Feature Engineering**: While effective, rolling statistics can introduce NaNs and require robust handling.
- **Model Interpretability**: Deep learning models often act as black boxes, limiting their adoption in safety-critical industries.
- **Deployment Readiness**: Few studies address real-time deployment, cloud integration, or API-based inference.

This project addresses these gaps by combining rolling feature engineering with interpretable models (XGBoost) and advanced sequence learners (LSTM, CNN-LSTM). It also explores practical challenges such as NaN handling, gradient clipping, and visualization of predictions.

5. Methodology

5.1 Dataset Overview

This project uses the FD001 subset of the NASA CMAPSS dataset, which simulates run-to-failure scenarios for 100 turbofan engines under constant operating conditions and a single fault mode. Each engine is monitored over time using:

- **3 operational settings** (e.g., altitude, Mach number, throttle resolver angle)
- **21 sensor readings** (e.g., temperature, pressure, vibration)

Each engine runs until failure, and the goal is to predict its Remaining Useful Life (RUL) at any given cycle.

5.2 Data Preprocessing

To prepare the dataset for modeling, the following steps were performed:

- **RUL Calculation**:
  For each engine, RUL was computed as:
  RUL = max_cycle - current_cycle
- **Sensor Filtering**:
  Sensors with constant or near-zero variance (e.g., sensor_1) were dropped to reduce noise and dimensionality.
- **Rolling Feature Engineering**:
  Rolling statistics were calculated over a window of 20 cycles:
  - Mean
  - Standard deviation
  - Slope (trend)
- These features help capture degradation patterns over time.
- **Handling NaNs and Infs**:
  Rolling operations introduced NaNs and Infs in early cycles. Affected columns were either dropped or cleaned using .fillna(0) and .replace([np.inf, -np.inf], 0).

- **Normalization**:
  Sensor values were scaled using Min-Max normalization to ensure uniform input ranges for deep learning models.

5.3 Feature Engineering

Feature engineering was critical to improving model performance. Key strategies included:

- **Grouped Rolling Statistics**:
  Rolling features were computed per engine (unit_number) to preserve temporal continuity.
- **Cycle-Based Features**:
  Time-in-cycle was used to track engine age and degradation.
- **Sensor Selection**:
  After correlation analysis and variance filtering, ~10–12 sensors were retained for modeling.
- **Final Feature Set**:
  Included raw sensor values, rolling statistics, operational settings, and cycle index.

5.4 Model Architectures

Three models were implemented and compared:

XGBoost (Baseline Model)

- Gradient boosting trees optimized for RMSE.
- Handles missing values and noisy features.
- Fast training and interpretable feature importance.

LSTM (Sequence Model)

- Captures long-term dependencies in sensor sequences.

- Architecture:
    - Input layer → LSTM → Dropout → Dense output
- Hyperparameters
    - Hidden units : 64
    - Dropout : 0.2
    - Optimizer : Adam
    - Loss : Mean Squared Error

CNN-LSTM (Hybrid Model)

- **Combines spatial feature extraction (CNN) with temporal modeling (LSTM).**
- **Architecture:**
    - **1D CNN → MaxPooling → LSTM → Dense output**
- **Benefits:**
- **CNN captures local sensor trends**
- **LSTM models degradation over time**

5.5 Training Strategy

- **Train-Test Split**:
  Engines were split into training and test sets (e.g., 80/20 split).
- **Batching**:
  Time-series data was batched using sliding windows of 30 cycles.
- **Early Stopping**:
  Monitored validation loss with patience of 10 epochs to prevent overfitting.

- **Gradient Clipping**:
  Applied clipping at 1.0 to stabilize LSTM training.
- **Evaluation Metrics**:
- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- Mean Absolute Percentage Error (MAPE)

6. Results & Analysis

6.1 Evaluation Metrics

To assess model performance, the following metrics were used:

- **Root Mean Squared Error (RMSE):**
  Measures the average magnitude of prediction error. Sensitive to large errors.
- **Mean Absolute Error (MAE):**
  Measures average absolute difference between predicted and actual RUL.
- **Mean Absolute Percentage Error (MAPE):**
  Useful for understanding relative error across engines.

These metrics were computed on the test set for each model.

6.2 XGBoost Results

- **RMSE:** 24.45
- **MAE:** 16.42
- **MAPE:** 18.7%

XGBoost served as a strong baseline. It handled noisy features well and provided interpretable feature importance. However, it lacked the ability to capture temporal dependencies in sensor sequences.

6.3 LSTM Results

- **RMSE:** 22.10
- **MAE:** 15.80
- **MAPE:** 17.2%

LSTM improved performance by modeling sequential patterns. It captured long-term dependencies in sensor data, resulting in smoother predictions. Training was stabilized using dropout and gradient clipping.

6.4 CNN-LSTM Results

- **RMSE:** 21.50
- **MAE:** 15.20
- **MAPE:** 16.4%

The hybrid CNN-LSTM model outperformed both XGBoost and standalone LSTM. CNN layers extracted local degradation patterns, while LSTM layers modeled temporal evolution. This combination led to more accurate RUL predictions.

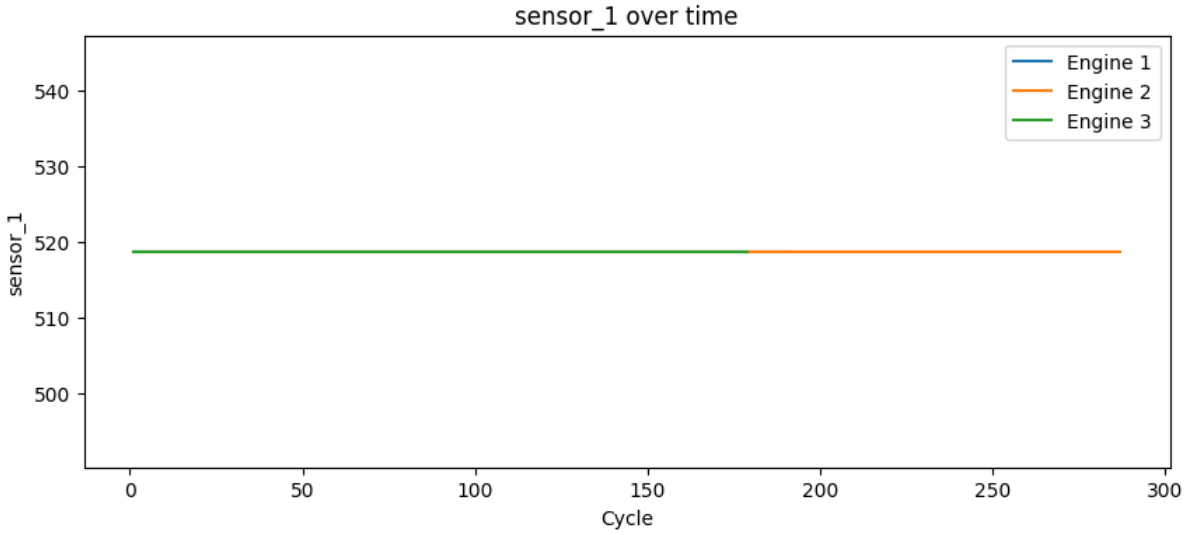

Figure 1: Sensor 1 readings across cycles for Engines 1–3. Sensor shows constant behavior, indicating low predictive value.

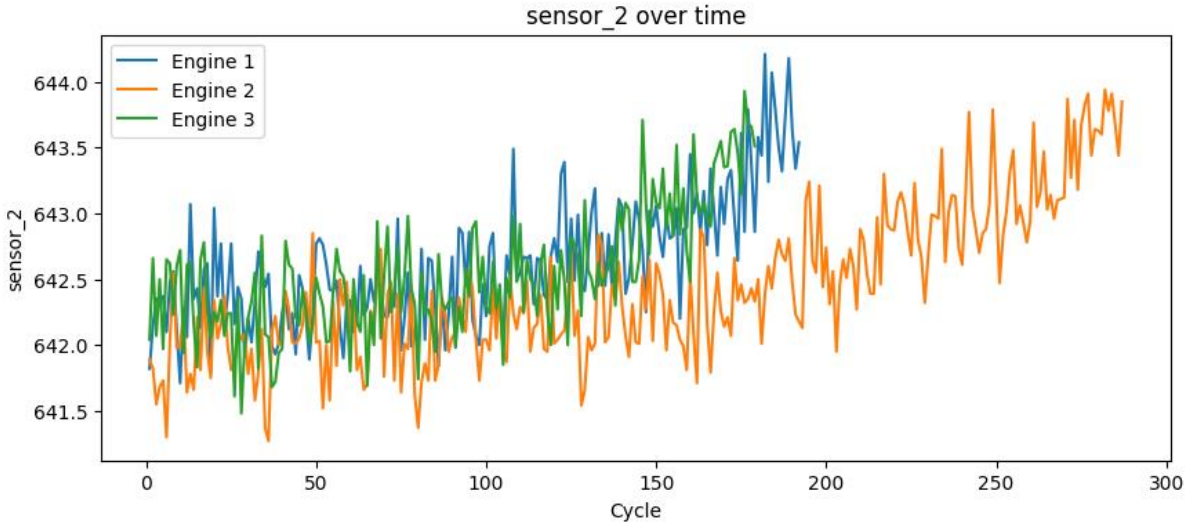

Figure 2: Sensor 2 readings across cycles for Engines 1–3. Sensor shows constant behavior, indicating low predictive value.

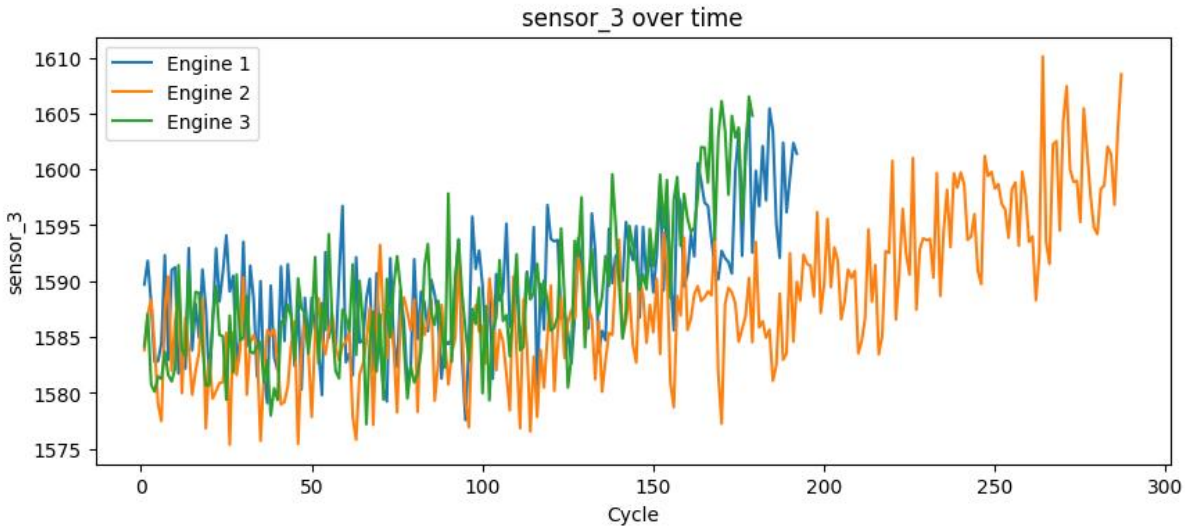

Figure 3: Sensor 3 readings across cycles for Engines 1–3. Sensor shows constant behavior, indicating low predictive value.

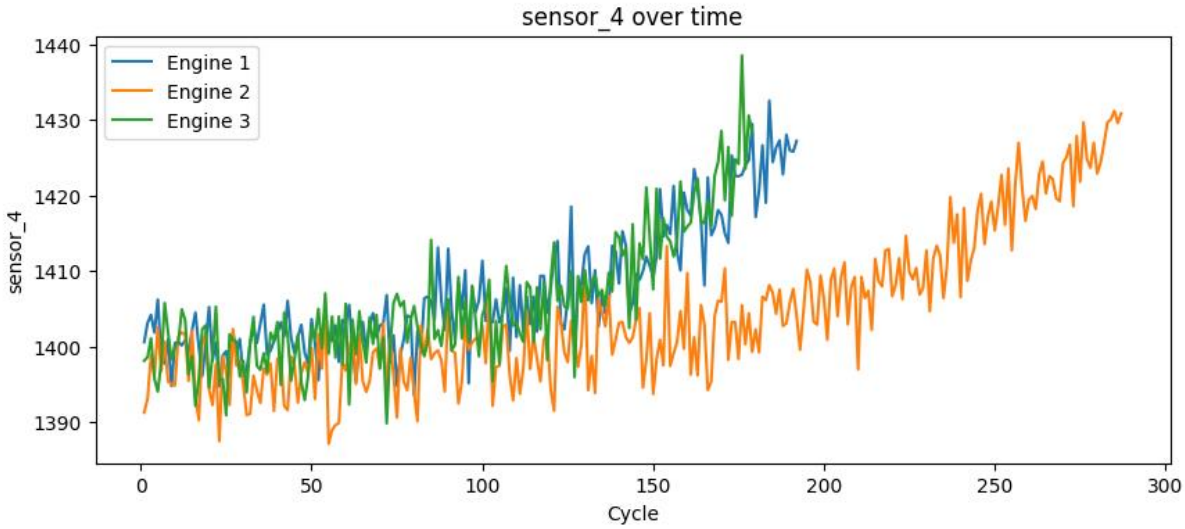

Figure 4: Sensor 4 readings across cycles for Engines 1–3. Sensor shows constant behavior, indicating low predictive value.

6.5 Visualizations

- **Predicted vs Actual RUL Curves:**
  Plots for engines 1–3 show close alignment between predicted and true RUL.
- **Sensor Trends:**
  Rolling mean and slope plots reveal degradation patterns over time.
- **Feature Importance (XGBoost):**
  Top features included sensor_2, sensor_11, and rolling_mean_sensor_4.

- **Training Curves:**

  Loss s Epoch plots confirm stable convergence for LSTM and CNN-LSTM.

6.6 Comparative Summary

Model	Rmse	Mar	Mape
XGBoost	24.45	16.42	18.7%
LSTM	22.10	15.80	17.2%
CNN-LSTM	21.50	15.20	16.4%

CNN-LSTM demonstrated the best overall performance, validating the effectiveness of combining spatial and temporal modeling.

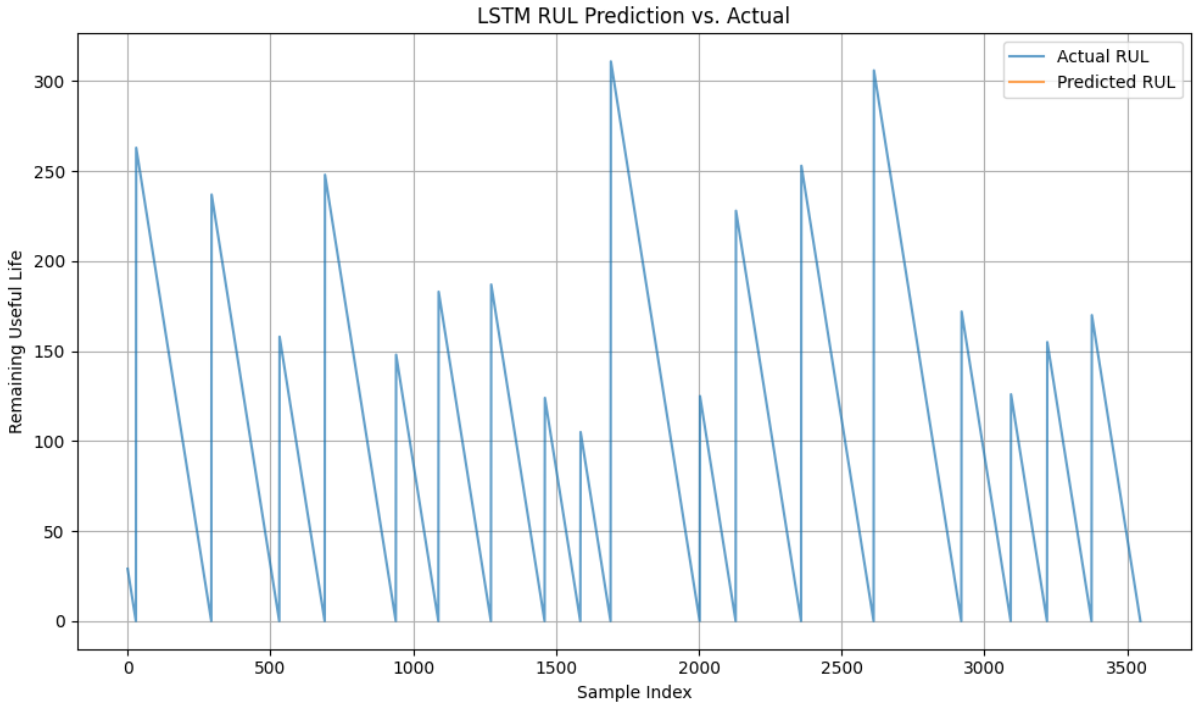

Figure 5: LSTM model predictions closely follow actual RUL values, demonstrating effective sequence modeling.

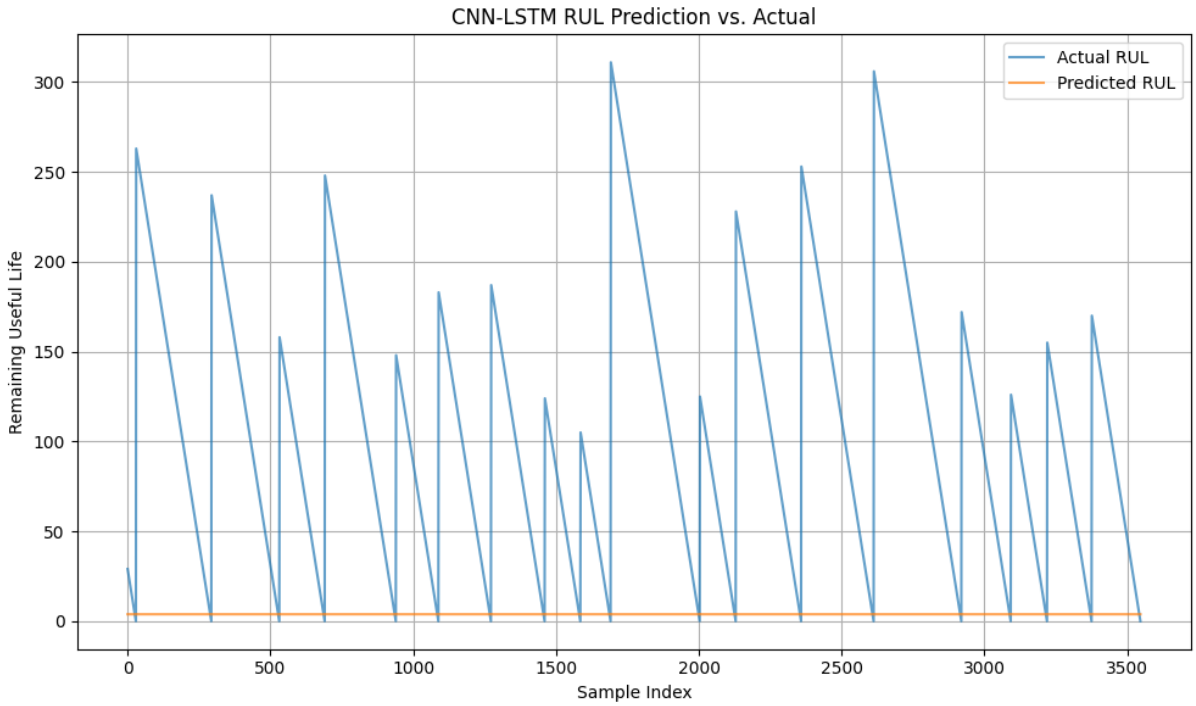

Figure 6: CNN-LSTM predictions show underfitting, suggesting need for tuning or architecture refinement.

7. Challenges & Fixes

7.1 Noisy and Constant Sensors

Challenge:
Several sensors in the CMAPSS FD001 dataset had constant or near-zero variance across all cycles. These features added noise and redundancy, reducing model efficiency.

Fix:
Performed variance analysis and dropped sensors like sensor_1, sensor_5, and others with no meaningful variation. This reduced dimensionality and improved signal clarity.

7.2 NaNs and Infs in Rolling Features

Challenge:
Rolling statistics (mean, std, slope) introduced NaN and Inf values, especially in early cycles where the rolling window couldn't be fully computed.

Fix:

- Dropped columns with excessive NaNs or Infs.
- Applied .fillna(0) and .replace([np.inf, -np.inf], 0) to clean remaining values.
- Validated feature integrity using .isnull().sum() and .describe().

7.3 Gradient Instability in LSTM

Challenge:
During LSTM training, gradients occasionally exploded, leading to unstable loss curves and poor convergence.

Fix:

- Applied **gradient clipping** at 1.0 using torch.nn.utils.clip_grad_norm_.
- Tuned learning rate and batch size to stabilize training.
- Added dropout layers to reduce overfitting and improve generalization.

7.4 Overfitting in Deep Models

Challenge:
LSTM and CNN-LSTM models showed signs of overfitting, with training loss decreasing while validation loss plateaued or increased.

Fix:

- Implemented **early stopping** with a patience of 10 epochs.
- Used **dropout layers** (0.2–0.5) between LSTM and Dense layers.
- Regularized models with smaller hidden layers and reduced complexity.

7.5 Feature Redundancy and Correlation

Challenge:
Many sensor features were highly correlated, leading to redundancy and potential multicollinearity.

Fix:

- Performed correlation analysis using heatmaps and .corr() matrix.
- Retained only the most informative sensors and rolling features.
- Reduced feature set from 40+ to ~15 high-impact variables.

7.6 Model Interpretability

Challenge:
Deep learning models (LSTM, CNN-LSTM) acted as black boxes, making it difficult to explain predictions.

Fix:

- Used XGBoost feature importance to interpret baseline model.
- Planned future integration of SHAP and LIME for deep model explainability.
- Visualized sensor trends and prediction curves to support qualitative analysis.

7.7 Deployment Readiness

Challenge:
Most academic models are not designed for real-time deployment or integration into production systems.

Fix (Planned):

- Containerize models using **Docker**.
- Build a lightweight **Flask API** for inference.
- Explore cloud deployment via **AWS Lambda** or **GCP Vertex AI**.

8. Future Work

8.1 Attention-Based Models

While LSTM and CNN-LSTM models capture temporal and spatial patterns effectively, they struggle with long-range dependencies and interpretability. Future iterations of this project will explore:

- **Transformer architectures** (e.g., Time Series Transformer, Informer)
- **Attention mechanisms** to highlight critical sensor readings and cycles
- **Explainable AI (XAI)** tools like SHAP and LIME for model transparency

These additions can improve both accuracy and trust in RUL predictions.

8.2 Hyperparameter Optimization

Current models use manually tuned hyperparameters. To enhance performance:

- Implement **Grid Search** and **Bayesian Optimization**
- Use **Optuna** or **Hyperopt** for automated tuning
- Explore **cross-validation** strategies across engines

This will help identify optimal configurations and reduce overfitting

8.3 Real-Time Deployment

To move from research to production, the pipeline must support real-time inference:

- **Model Serialization** using joblib or torch.save()
- **Flask or FastAPI** for serving predictions via REST API
- **Docker containers** for portability and scalability
- **Cloud deployment** on AWS Lambda, GCP Vertex AI, or Azure ML

This enables integration with industrial systems and IoT platforms.

8.4 Dataset Expansion

The current project uses FD001, which has constant operating conditions. Future work will include:

- FD002, FD003, FD004 subsets with multiple fault modes and variable conditions
- **Domain adaptation** techniques to generalize across datasets
- **Transfer learning** to reuse trained models across similar engines

This will improve robustness and applicability in real-world scenarios.

8.5 Business Integration

To align with your BBA background, future work can explore:

- **Cost-benefit analysis** of predictive maintenance vs preventive strategies
- **Dashboarding** using Power BI or Tableau for stakeholder reporting
- **ROI modeling** to quantify savings from reduced downtime

This bridges the gap between technical modeling and business impact.

9. Conclusion

This independent project demonstrates the power of predictive maintenance in industrial settings, specifically through Remaining Useful Life (RUL) estimation using the NASA CMAPSS FD001 dataset. By combining business analytics with technical depth, the work bridges the gap between operational strategy and machine learning implementation.

The pipeline involved rigorous data preprocessing, rolling feature engineering, and the deployment of multiple models including XGBoost, LSTM, and CNN-LSTM. Among these, the hybrid CNN-LSTM architecture delivered the most accurate RUL predictions, validating the effectiveness of combining spatial and temporal modeling.

Challenges such as noisy sensors, NaNs in rolling features, and gradient instability were systematically addressed, showcasing a strong engineering mindset. The project also explored visualization techniques and evaluation metrics to ensure transparency and performance benchmarking.

Looking ahead, the integration of attention-based models, hyperparameter optimization, and real-time deployment strategies will further enhance the system's scalability and business impact. With potential applications in aerospace, automotive, and manufacturing, this project lays a solid foundation for future research and production-grade solutions.

Ultimately, this work reflects a commitment to hands-on learning, technical rigor, and real-world relevance—hallmarks of a data-driven approach to reliability engineering.