**Milestone 3**

**Model Evaluation, Performance Assessment,**

**Risk Thresholding & Alert System**

**Project Name:** PrognosAI: AI-Driven Predictive Maintenance System Using Time-Series Sensor Data

**Dataset:** NASA Turbofan Jet Engine Data Set

**Prepared by:** Durga Veera Prasad V

# **Table of Contents**

1. 1. Objective
2. 2. Modules Used & Their Purpose
3. 3. Workflow Overview
4. Step 1: Configuration
5. Step 2: Reproducibility
6. Step 3: Data Generation
7. Step 4: LSTM Model Builder
8. Step 5: 5-Fold Cross-Validation
9. Step 6: Final Model Training on Full Dataset
10. Step 7: Dynamic Alerts
11. Step 8: Plots
12. Step 9: PDF Report
13. 4. Key Results
14. 5. Saved Files
15. 6. Conclusion

# **1. Objective**

- Train an LSTM model for RUL prediction with corrected cross-validation scaling.  
- Compute evaluation metrics on the original RUL scale.  
- Introduce dynamic alert thresholds (20% critical, 50% warning).  
- Generate a PDF report summarizing CV results, predictions, and alert diagnostics.

# **2. Modules Used & Their Purpose**

|  |  |
| --- | --- |
| **Module** | **Purpose / Use** |
| **numpy** | Numerical operations, array manipulation for synthetic data and predictions. |
| **pandas** | Organize cross-validation results and save CSV reports. |
| **os** | File path management for saving models, plots, and PDFs. |
| **joblib** | Save/load fitted MinMaxScaler objects for reproducible scaling. |
| **matplotlib.pyplot** | Plotting training loss, predicted RUL, residuals, and R² across folds. |
| **matplotlib.backends.backend\_pdf.PdfPages** | Generate a multipage PDF report without extra dependencies. |
| **tensorflow / keras.models.Sequential** | Build sequential LSTM models for time-series regression. |
| **tensorflow.keras.layers.LSTM** | Learn temporal dependencies in RUL sensor data sequences. |
| **tensorflow.keras.layers.Dense** | Fully connected layers for mapping LSTM outputs to RUL prediction. |
| **tensorflow.keras.layers.Dropout** | Regularization to reduce overfitting. |
| **tensorflow.keras.layers.Input** | Explicit input layer to avoid Keras input warnings. |
| **tensorflow.keras.callbacks.EarlyStopping** | Stop training when validation loss stagnates. |
| **tensorflow.keras.callbacks.ReduceLROnPlateau** | Reduce learning rate when model stops improving. |
| **tensorflow.keras.optimizers.Adam** | Optimizer for faster convergence. |
| **sklearn.preprocessing.MinMaxScaler** | Scale features and targets to 0–1 range; fit on training fold only to avoid leakage. |
| **sklearn.model\_selection.KFold** | 5-fold cross-validation for robust evaluation. |
| **sklearn.metrics.r2\_score** | Compute R² (coefficient of determination) for train/test splits. |
| **sklearn.metrics.mean\_squared\_error** | Compute RMSE for test evaluation. |
| **sklearn.metrics.mean\_absolute\_error** | Compute MAE for test evaluation. |

# **3. Workflow Overview**

## **Step 1:** Configuration

Define number of samples, timesteps, features, CV splits, epochs, and batch sizes for both CV and final training.

## Step 2: Reproducibility

Set numpy and tensorflow seeds for deterministic results.

## **Step 3:** Data Generation

Synthetic CMAPSS-style dataset with linear decay + noise to simulate engine RUL signals. X = sensor sequences, y\_raw = final RUL target.

## **Step 4:** LSTM Model Builder

Sequential model: LSTM(128) -> Dropout(0.1) -> Dense(64) -> Dense(32) -> Dense(1). Adam optimizer with MSE loss.

## **Step 5: 5-Fold Cross-Validation**

For each fold:  
1. Split raw data into train/test.  
2. Fit MinMaxScaler only on training fold.  
3. Scale train and test sets.  
4. Train model with EarlyStopping + ReduceLROnPlateau.  
5. Predict and inverse-transform to original RUL scale.  
6. Compute Train/Test R², RMSE, MAE.  
7. Save model, scalers, and fold loss plot.  
Placeholder for fold loss plot: Insert loss\_foldX.png images here for each fold.

## **Step 6: Final Model Training on Full Dataset**

Fit scalers on full dataset, train LSTM for FINAL\_EPOCHS, save final model and scalers.  
Placeholder for final training loss plot.

## **Step 7: Dynamic Alerts**

Predicted RUL thresholds: Critical (bottom 20%), Warning (bottom 50%). Identify critical, warning, and normal samples. Save alert indices and predictions as .npy files.  
Placeholder for predicted RUL with alerts: predicted\_rul\_with\_alerts\_full.png.

## **Step 8: Plots**

R² across folds: r2\_across\_folds.png  
Residual histogram: residuals\_hist.png

## **Step 9: PDF Report**

Multi-page report using PdfPages: 1. Summary of CV results, metrics, and alerts. 2. R² plot across folds. 3. Predicted RUL with alerts. 4. Residual distribution.  
Placeholder for PDF: Milestone3\_Report\_Corrected.pdf

# **4. Key Results**

- Cross-validation R² consistently high; average Train/Test R² reported.  
- RMSE and MAE calculated on original RUL scale.  
- Alerts dynamically categorize critical and warning samples.  
- Complete PDF report generated.

# **5. Saved Files**

* optimized\_lstm\_final.keras → final trained model
* final\_scalers.pkl → feature & target scalers
* crossval\_results\_corrected.csv → CV metrics
* predicted\_rul\_with\_alerts\_full.png → RUL prediction with alerts
* r2\_across\_folds.png → fold R² visualization
* residuals\_hist.png → residual distribution
* Milestone3\_Report\_Corrected.pdf → full report

# **6. Conclusion**

- Corrected scaler leakage in CV.  
- Metrics computed on original scale for realistic evaluation.  
- Introduced dynamic alerts for predictive maintenance monitoring.  
- Fully automated PDF report enables quick results sharing.