# PrognosAI: AI-Driven Predictive Maintenance

PrognosAI is a prototype predictive maintenance system designed to estimate the Remaining Useful Life (RUL) of industrial machinery using multivariate time-series sensor data. Leveraging deep learning models such as LSTM/GRU with a custom self-attention mechanism, the system enables timely maintenance decisions, minimizes unplanned downtime, and optimizes asset utilization.

## Table of Contents



### Project Objective

Goal: Estimate machinery RUL from NASA CMAPSS dataset.
Impact: Enable proactive maintenance, reduce costs, and improve equipment reliability.

### Architecture

1. Data Ingestion: Loading and cleaning sensor data.
2. Feature Engineering: Rolling window sequence generation and RUL target computation.
3. Model Training: LSTM/GRU network training with self-attention for RUL prediction.
4. Evaluation: RMSE computation and predicted vs actual RUL analysis.
5. Alerting: Defining RUL thresholds and triggering maintenance alerts.
6. Visualization & Dashboard: Interactive charts showing RUL trends and alert zones.

### Setup & Installation

Follow these step-by-step instructions to set up and run the PrognosAI project:

1. **Clone the Repository**
    - Open your terminal or command prompt.
    - Run the command below to clone the project repository:

      ```
      git clone https://github.com/prognosAI-Infosys-Intern-project/prognosAI-Intern-project.git
      ```

    - Change directory to the project folder:

      ```
      cd prognosAI-Intern-project/Project
      ```

2. **Create and Activate Python Virtual Environment**
    - Run the following command to create a virtual environment named `venv`:

      ```
      python -m venv venv
      ```

    - Activate the virtual environment:
        - On Linux/macOS:

          ```
          source venv/bin/activate
          ```

        - On Windows:

          ```
          venv\Scripts\activate
          ```

3. **Install Required Dependencies**
    - Install the required Python packages using:

      ```
      pip install -r requirements.txt
      ```

4. **Run Data Preprocessing**
    - Prepare the datasets and feature sequences by running:

      ```
      python data_preprocessing.py
      ```

5. **Train the Model**
    - Train the predictive maintenance model by executing:

      ```
      python train_model.py
      ```

6. **Start Streamlit Dashboard**
    - Launch the interactive web dashboard for predictions and alerts:

      ```
      streamlit run app.py
      ```

7. **Upload Test Data**
    - Use the Streamlit app interface to upload `sequence` and `metadata` files located in:

      ```
      processed_data/test
      ```

    - View model predictions and alert zones on the dashboard.

## Tech Stack

- `Python` : Core development language
- `Pandas` , `NumPy` : Data processing
- `Matplotlib, Seaborn` : Visualization
- `TensorFlow` / `Keras` : Deep learning models
- `Scikit-learn` : Preprocessing & metrics
- `Streamlit` : Dashboard interface
- `NASA CMAPSS Dataset` : Source data

## Model Design & Architecture

- Input: Multivariate time-series with shape (timesteps, features)
- Layers:
    1. Bidirectional LSTM (128 units) → LayerNormalization → Dropout(0.3)
    2. Bidirectional LSTM (64 units) → LayerNormalization → Dropout(0.3)
    3. Custom Self-Attention layer pooling attended features over time
    4. Dense layer (64 units, activation='relu') → Dropout(0.3)
    5. Output Dense layer (1 unit, linear activation) for regression
- Regularization: L2 regularization (1e-4) applied to LSTM kernels
- Optimizer: Adam optimizer (learning rate 5e-4, clipnorm=1.0)
- Loss function: Mean Squared Error (MSE)
- Metrics: Mean Absolute Error (MAE) for performance tracking

## Model Performance & Evaluation Metrics

- Root Mean Squared Error (RMSE): 3.0430
- Mean Absolute Error (MAE): 2.1536

These metrics indicate the model predicts the Remaining Useful Life with an average absolute deviation of about 2.15 cycles, and an RMSE indicating moderate penalty for larger errors.

## Project Milestones

- Milestone 1: Data preparation & sequence generation
- Milestone 2: Model development & training
- Milestone 3: Evaluation & performance assessment
- Milestone 4: Alert logic implementation
- Milestone 5: Interactive dashboard development

## License