Milestone 5 - Visualization & Dashboard Development

Project: PrognosAI — Remaining Useful Life (RUL) Prediction

Date: November 10, 2025

Author: Nithin G J

• Objective

Develop an interactive visualization dashboard to monitor Remaining Useful Life (RUL) trends, predictions, and alerts in real time.
The goal is to provide clear insights into equipment health through intuitive dashboards using Streamlit, enabling proactive maintenance decisions.

• Summary of Implementation

A comprehensive Streamlit-based Prognostics Dashboard was created, offering seamless model integration, alert logic, and visualization.
Key components include model and scaler loading, preprocessing, real-time predictions, and interactive Plotly visualizations (line, scatter, and summary tables).
Performance metrics (RMSE, MAE, R2) are displayed using dynamic KPI cards.

• Key Files & Paths

- Model: models_m2/optimized_fd1_milestone4.h5
- Scaler: models_m2/scaler_fd1_milestone4.save
- Processed test data: processed/fd1_test_ws30.npz
- App Script: app_streamlit_prognosai.py

• Dashboard UX & Visual Components

The dashboard features a sleek layout with distinct sections:
- Header banner with project context
- KPI cards summarizing key performance indicators
- Dual-line RUL trend chart (Actual vs Predicted)
- Color-coded alert scatter plots
- Recent alert table with conditional highlighting for rapid triage

• Alert Thresholds & Logic

Alert levels are determined using predefined thresholds:
- Normal: RUL > 70
- Warning: 30 < RUL <= 70
- Critical: RUL <= 30

Recommended improvements include adaptive thresholding, historical data calibration, and confidence-based alert filtering.

• Deployment & Instructions

1. Install required libraries: pip install streamlit plotly joblib tensorflow scikit-learn pandas numpy
2. Verify model, scaler, and test data paths.
3. Run the app: streamlit run app_streamlit_prognosai.py
4. Access via http://localhost:8501 to explore the dashboard.

• Performance & Usability

- Cache data and models using st.cache_resource for efficiency.
- Precompute predictions when possible for smoother UX.
- Use Docker for deployment and nginx for scalable serving.

• Extensions & Next Steps

- Add engine-level drilldown and time-series playback.
- Integrate alert notifications via email/SMS.
- Enable user authentication for role-based access.
- Provide live API endpoint for real-time inference.