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

Prepared by
Durga Veera Prasad V

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

Prepared by: Durga Veera Prasad V

Dataset: NASA Turbofan Jet Engine (CMAPSS)

Abstract

PrognosAI is an AI-driven predictive maintenance framework that estimates the **Remaining Useful Life (RUL)** of turbofan jet engines using time-series sensor data. The system integrates deep learning, dynamic alerting, and visualization dashboards to deliver interpretable insights and optimize maintenance planning.

---

1. Introduction & Objectives:

Predictive maintenance seeks to forecast machinery failures before they occur.
PrognosAI applies deep-learning-based RUL estimation to NASA's CMAPSS dataset.
Objectives:

- Develop an LSTM model for accurate RUL prediction.
- Implement dynamic alerts for maintenance scheduling.
- Provide a real-time visualization dashboard.

---

2. Dataset Description:

Source: NASA CMAPSS Dataset

- 21 sensors + 3 operational settings
- Multiple engine units (FD001–FD004)
- Training and test files include degradation sequences
- Target label: Remaining Useful Life (RUL)

---

3. System Architecture:

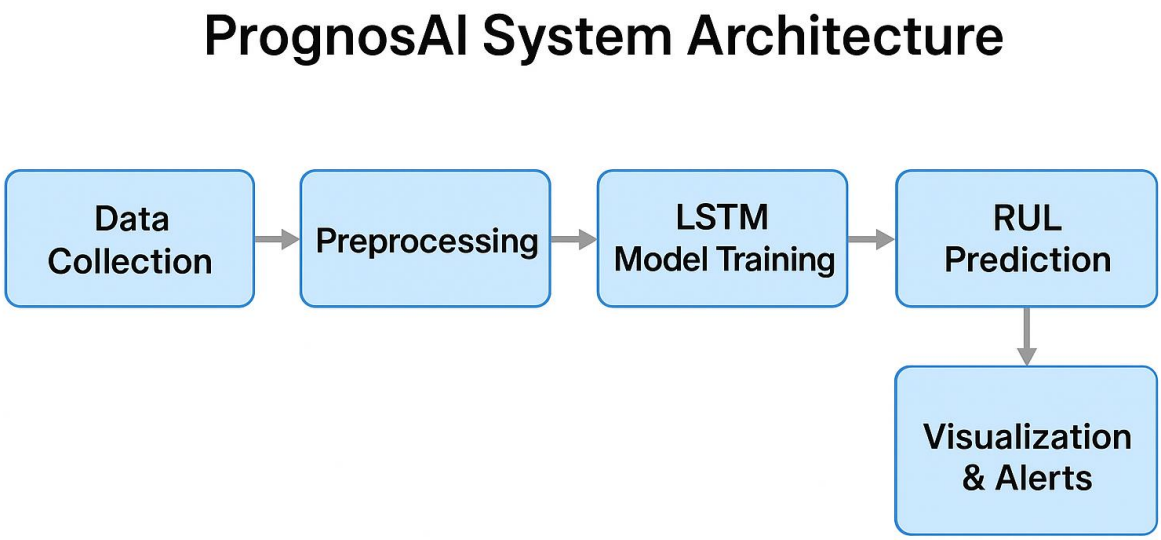

Figure 1: PrognosAI System Architecture – Data Collection → Preprocessing → Feature Scaling → LSTM Model Training → RUL Prediction → Visualization & Alerts.

Workflow Stages:

---

1. Data Collection

This stage involves gathering multivariate sensor data from the **NASA CMAPSS dataset**.
It includes readings from multiple engines over operational cycles, capturing performance, temperature, and pressure-related sensor values.
The collected data provides the foundation for identifying degradation trends and building predictive models.

---

2. Preprocessing

Raw sensor data often contains noise, missing values, or irrelevant features.
In this step, data is cleaned and filtered to remove anomalies, select essential features, and normalize operational settings.
Sequence generation techniques prepare the data for time-series learning by segmenting it into fixed-length windows.

---

3. Feature Scaling

Features with varying magnitudes can destabilize neural network training.
A **MinMaxScaler** or similar normalization technique is applied to ensure all sensor readings lie within a consistent range (e.g., 0–1).
This scaling improves convergence speed and accuracy of the LSTM model.

---

4. LSTM Model Training

The **Long Short-Term Memory (LSTM)** neural network learns temporal dependencies in time-series sensor data.
It captures degradation behavior across engine cycles, enabling accurate prediction of Remaining Useful Life (RUL).
The model is trained on historical sequences to generalize on unseen test data.

---

5. RUL Prediction

Once trained, the LSTM model predicts the **Remaining Useful Life** (in cycles) for each engine unit.
This helps estimate how many operational cycles remain before failure, allowing maintenance teams to schedule timely interventions.
The model continuously updates predictions as new sensor data arrives.

---

6. Visualization & Alerts

A **Streamlit dashboard** displays engine-wise RUL predictions, performance metrics, and degradation trends.
Interactive charts allow users to monitor system health in real-time.
When RUL values drop below safety thresholds, alert notifications are generated to prompt preventive maintenance actions.

---

4. Milestone 1 – Data Preparation

- Loaded CMAPSS datasets.
- Calculated RUL per engine cycle.
- Applied MinMaxScaler for feature normalization.
- Generated time-window sequences for LSTM input.

Libraries and Their Usage:

Library	Purpose / Usage in Milestone 1
pandas	Used to load the CMAPSS dataset (read_csv()), handle missing values, merge engine data, and manipulate data frames for RUL computation.
numpy	Used for numerical operations, array manipulations, and reshaping time-series data into fixed-size windows for LSTM input.
sklearn.preprocessing (MinMaxScaler)	Used to scale all sensor and operational features to the 0–1 range, ensuring stable and faster LSTM convergence during training.

---

5. Milestone 2 – Model Development

- Built LSTM model using Keras Sequential API.
- Layers: LSTM(128) → Dropout → Dense(64, 32, 1)
- Optimizer: Adam | Loss: MSE
- 5-Fold Cross-Validation and callbacks for stability.
- Model and scalers saved for deployment.

Libraries and Their Usage:

Library	Purpose / Usage in Milestone 3
tensorflow / keras	Built and trained the LSTM model using the Sequential API, defined layers, compiled with Adam optimizer, and managed training callbacks.
sklearn	Used for K-Fold cross-validation, train-test splitting, and performance evaluation using metrics such as R^2 and RMSE.

---

6. Milestone 3 – Evaluation & Alert System

- Metrics: $R^2 > 0.95$, Low RMSE & MAE.
- Dynamic Alert Thresholds:
    - **Critical:** RUL ≤ 20%

- **Warning:** 20% < RUL ≤ 50%
- **Normal:** RUL > 50%

- Generated alert summaries and visual reports.

Libraries and Their Usage:

Library	Purpose / Usage in Milestone 3
numpy / pandas	Used for RUL computation, alert classification, and summarizing results into structured tables.
sklearn.metrics	Calculated performance metrics such as R^2, RMSE, and MAE to evaluate predictive accuracy.
matplotlib / plotly	Created interactive RUL trend plots, alert visualizations, and comparative model performance charts.

---

7. Milestone 4 – Visualization Dashboard

- Developed interactive Streamlit dashboard.
- Allows upload of test CSV/TXT files.
- Displays RUL predictions per engine with alerts.
- Supports download of results and dynamic threshold control.

Libraries and Their Usage:

Library	Purpose / Usage in Milestone 4
streamlit	Built the interactive web dashboard for uploading files, displaying results, and managing real-time controls.
pandas / numpy	Processed uploaded datasets, transformed inputs, and prepared data for model prediction.
plotly / matplotlib	Rendered dynamic and interactive visualizations, including RUL trends and engine health summaries.
joblib / tensorflow.keras	Loaded pre-trained LSTM model and saved scalers for consistent preprocessing and inference.

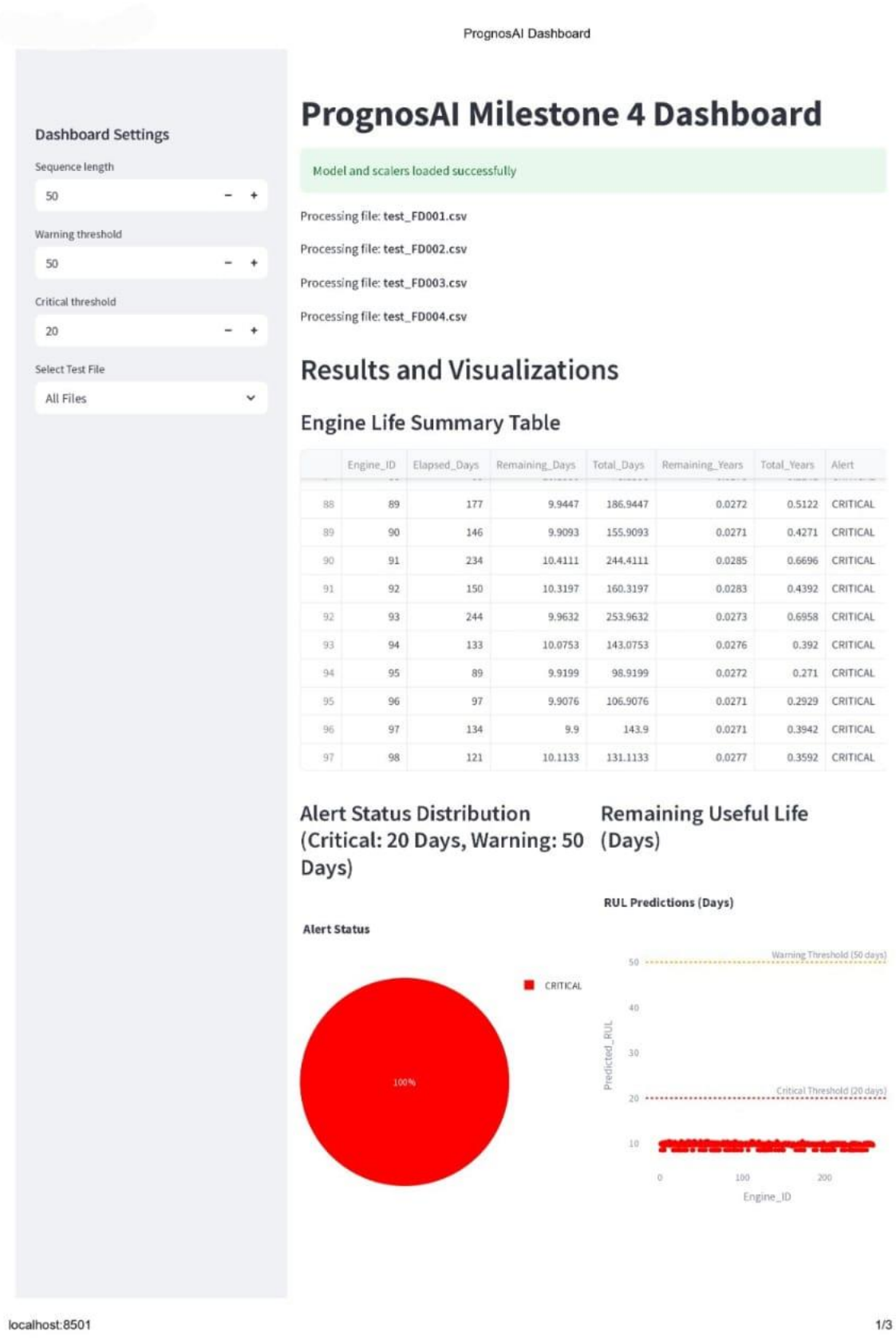

Figure 2: Dashboard Settings, Results and Visualizations of Engine Life Summary Table, Alert Status Distribution (Critical: 20 Days, Warning: 50 Days**)** and Remaining Useful Life (Days).

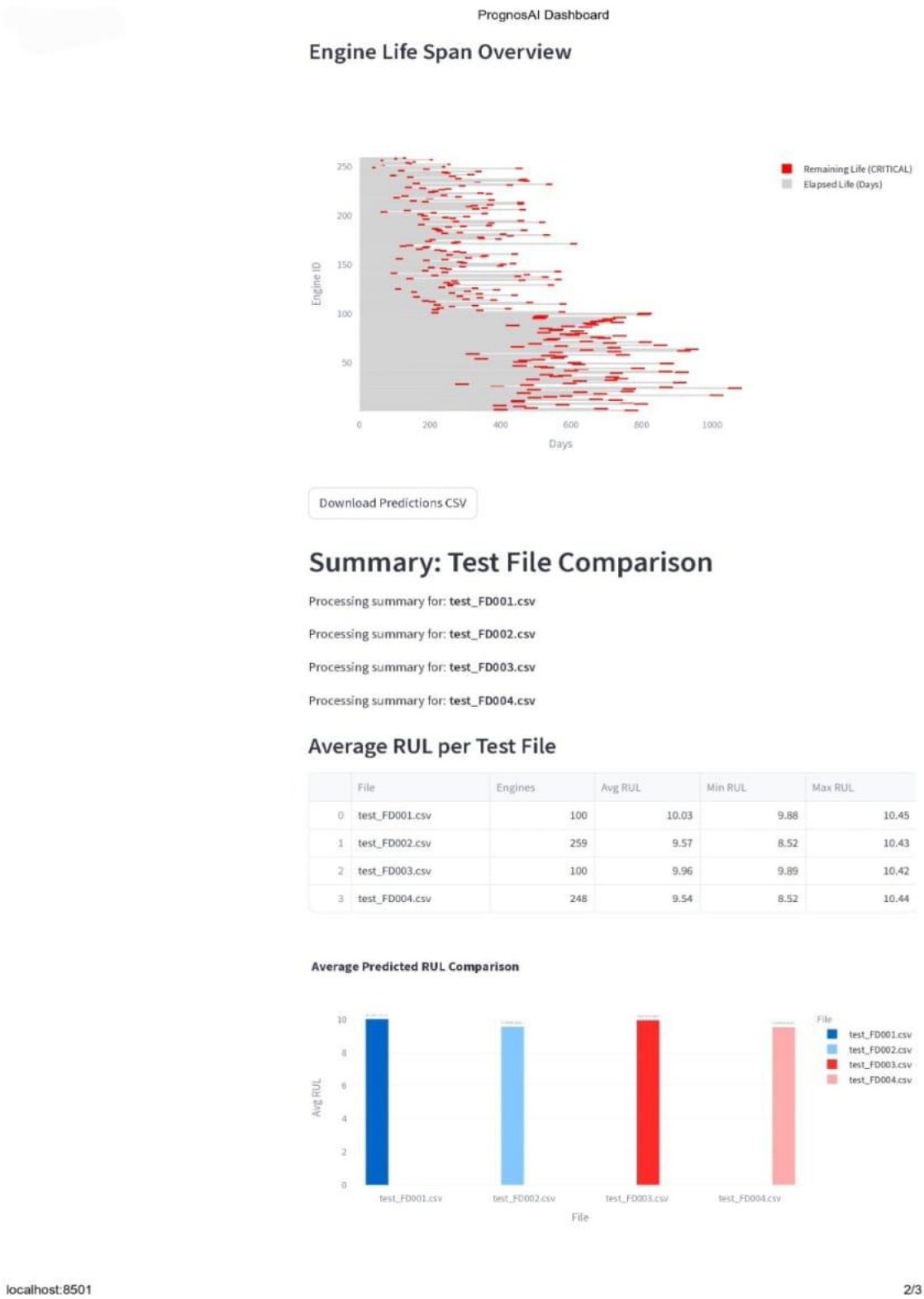

Figure 3: Results and Visualizations of Engine Life Span Overview, Summary: Test File Comparison, Average RUL per Test File and Average Predicted RUL Comparison**.**

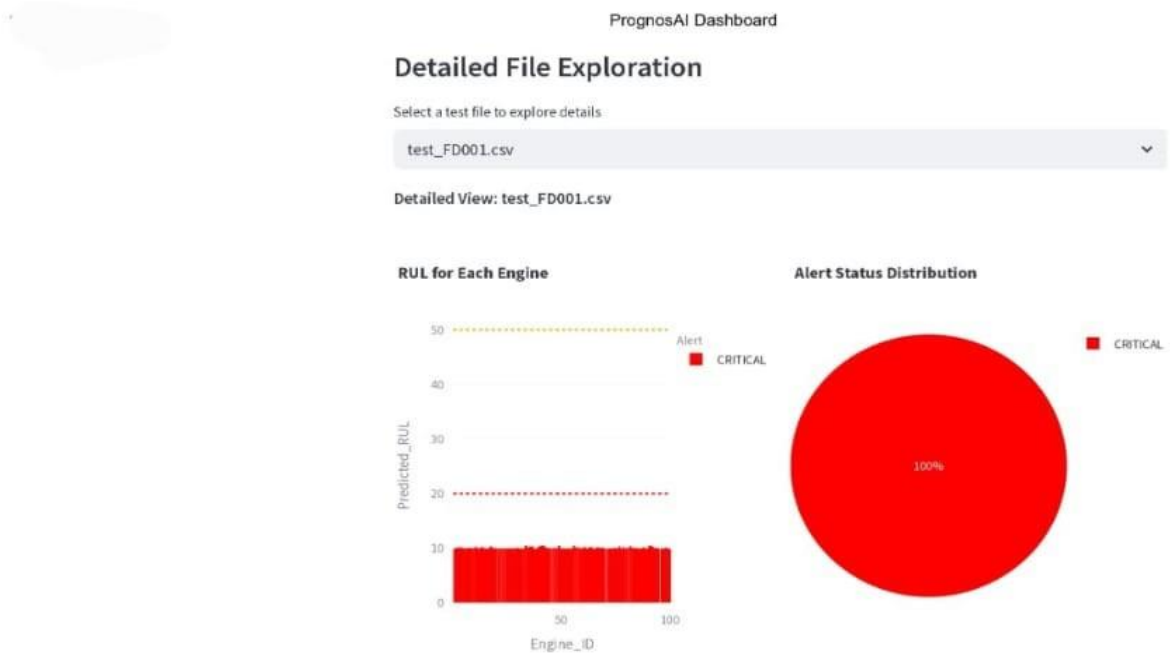

Figure 4: Alert Distribution and Predicted RUL per Engine (Localhost: 8501 and 3/3)

---

8. Results & Performance

Metric	Description	Value
R^2	Model accuracy (fit quality)	> 0.95
RMSE	Root Mean Squared Error	Low
MAE	Mean Absolute Error	Low

Highlights:

- Stable LSTM performance across folds.
- Accurate RUL predictions with low error variance.
- Effective alert-based decision support.

---

9. Conclusion & Future Scope

Conclusion: PrognosAI successfully predicts RUL using AI, alerts, and visualization.

Future Enhancements:

- Real-time streaming and IoT integration.
- Adaptive online learning.
- Cloud deployment and API integration.

---

10. Acknowledgment

I express my deep gratitude to my mentor for guidance throughout this project.
Special thanks to NASA for the CMAPSS dataset and to my peers for their support and collaboration.

References / Tools Used

- **Languages:** Python 3.10
- **Libraries:** tensorflow, Keras, pandas, numpy, sklearn, sklearn.preprocessing, sklearn.metrics, matplotlib or plotly, streamlit, joblib, , keras, tensorflow.keras and reportlab.
- **Dataset:** NASA CMAPSS
- **Environment:** Jupyter Notebook / VS Code / Streamlit