The background features a dark blue gradient with faint, light blue concentric circles and a scale. The scale is a semi-circular arc on the left side, with numerical labels ranging from 150 to 260 in increments of 10. Several dashed and solid circular lines with arrows indicate a clockwise direction of movement or flow.

PROGNOSAI: AI-DRIVEN PREDICTIVE MAINTENANCE SYSTEM USING TIME-SERIES SENSOR DATA

PREPARED BY: DURGA VEERA PRASAD V

DATASET: NASA TURBOFAN JET ENGINE (CMAPSS)

TABLE OF CONTENTS

- 1. Abstract
- 2. Introduction & Objectives
- 3. Dataset Description
- 4. System Architecture
- 5. Data Preparation
- 6. Model Development
- 7. Evaluation & Alert System
- 8. Visualization Dashboard
- 9. Results & Performance
- 10. Conclusion & Future Scope
- 11. Issues Faced
- 12. References & Tools Used

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 provide interpretable insights and optimize maintenance planning.

INTRODUCTION AND OBJECTIVES

- Predictive maintenance aims to forecast equipment failures before they occur.
- Objectives:
 - - Develop an LSTM model for accurate RUL prediction
 - - Implement dynamic alerts for maintenance scheduling
 - - Provide a real-time visualization dashboard

DATASET DESCRIPTION

- Source: NASA CMAPSS Dataset
- Features: 21 sensors + 3 operational settings
- Units: Multiple engine units (FD001–FD004)
- Target: Remaining Useful Life (RUL)

SYSTEM ARCHITECTURE

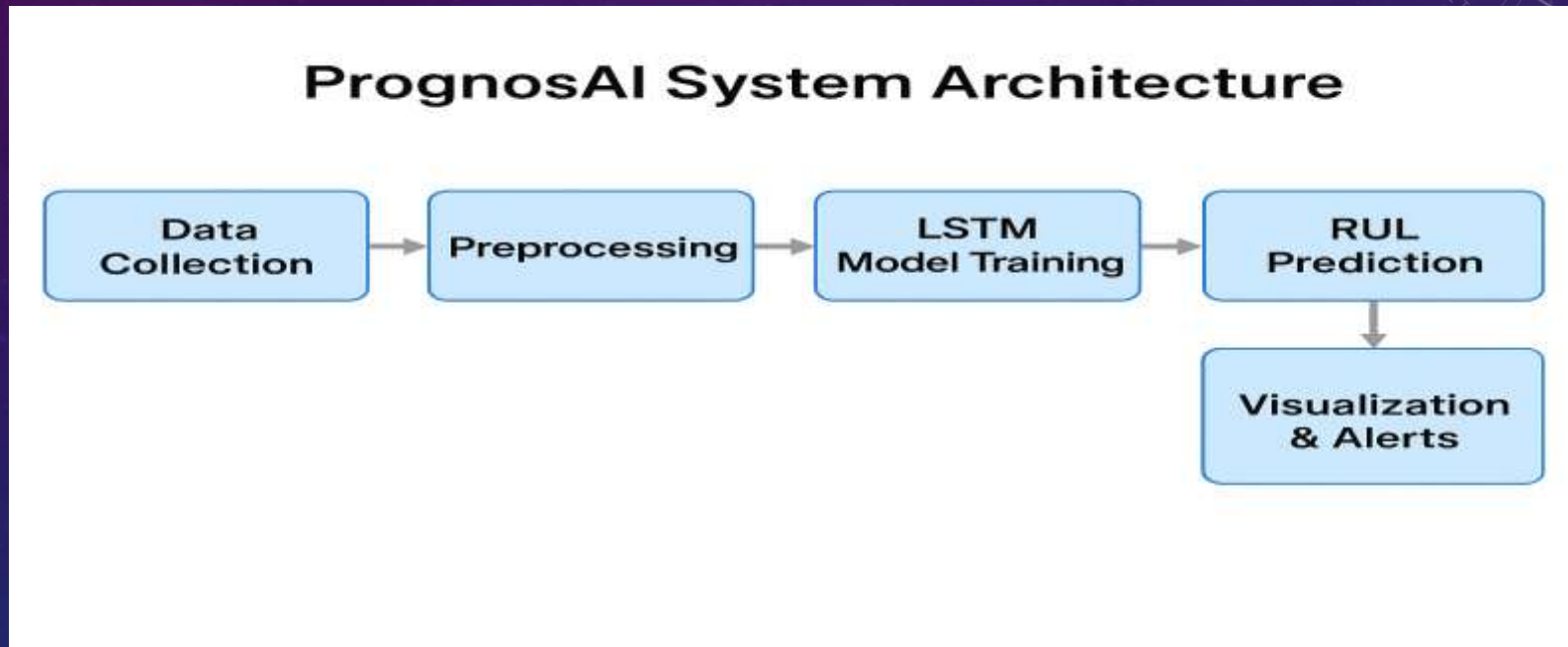


Figure1: PrognosAI System Architecture

DATA PREPARATION

- • Loaded CMAPSS datasets
- • Calculated RUL per engine cycle
- • Applied MinMaxScaler for normalization
- • Generated time-window sequences for LSTM input
- Libraries: pandas, numpy, sklearn.preprocessing

MODEL DEVELOPMENT

- • Built LSTM model using Keras Sequential API
- • Layers: LSTM(128) → Dropout → Dense(64, 32, 1)
- • Optimizer: Adam | Loss: MSE
- • Used 5-Fold Cross-Validation
- • Saved model and scalers for deployment
- Libraries: tensorflow, keras, sklearn

EVALUATION AND ALERT SYSTEM

- • Achieved $R^2 > 0.95$ with low RMSE and MAE
- • Dynamic alert thresholds:
 - - Critical: $RUL \leq 20\%$
 - - Warning: $20\% < RUL \leq 50\%$
 - - Normal: $RUL > 50\%$
- • Generated alert summaries and visual reports
- Libraries: numpy, pandas, sklearn.metrics, matplotlib, plotly

VISUALIZATION DASHBOARD

- • Developed interactive Streamlit dashboard
- • Allowed upload of test CSV/TXT files
- • Displayed engine-wise RUL predictions and alerts
- • Supported result download and threshold control
- Libraries: streamlit, pandas, numpy, plotly, joblib, tensorflow.keras
- Figures 2,3 and 4: Dashboard Screens, RUL Trends, Alert Distribution

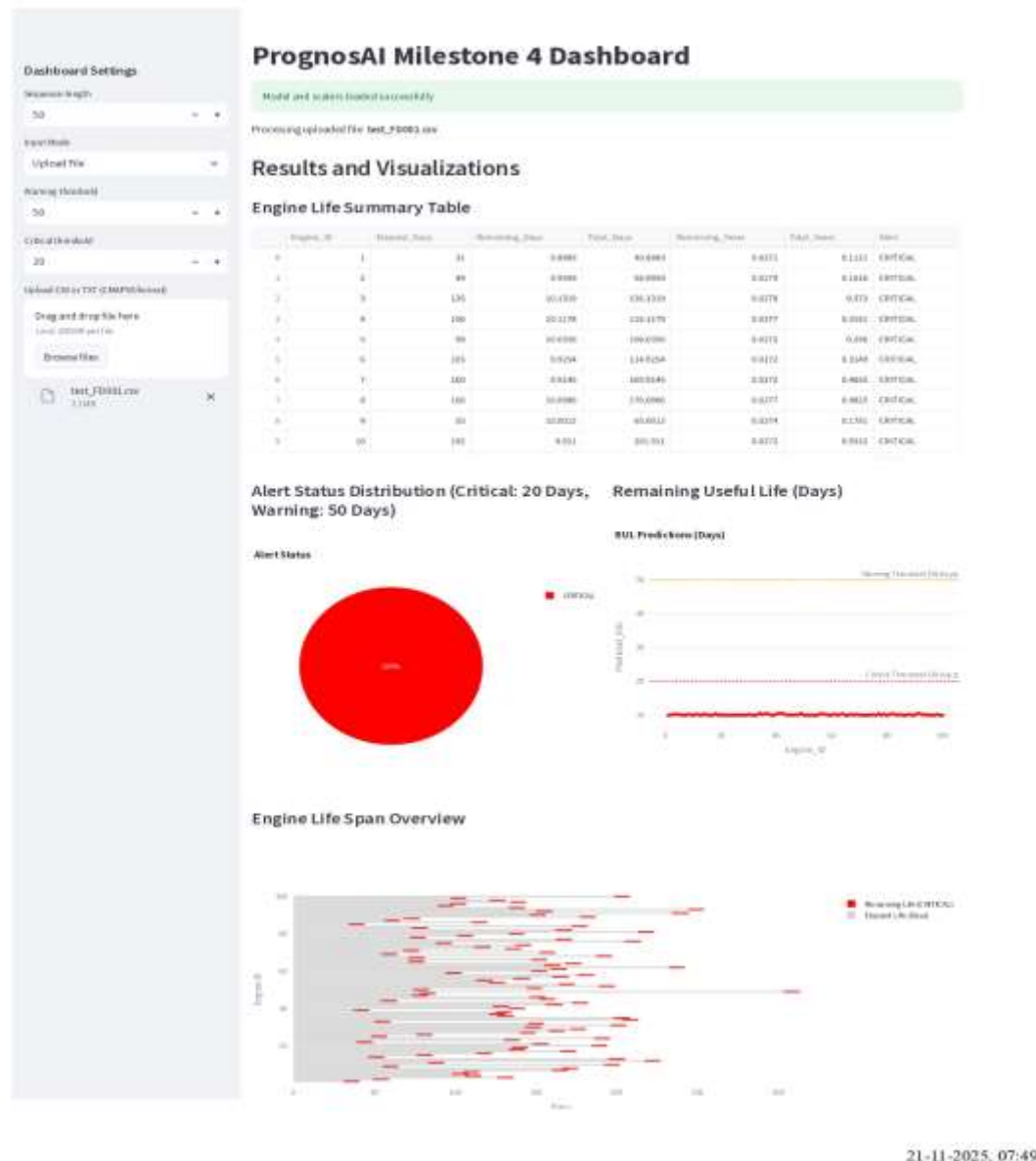


Figure2: PrognosAI Alert Distribution, Remaining Useful Life and Engine Life Span

[Download Predictions CSV](#)

Summary: Test File Comparison

Processing summary for: test_FD001.csv

Processing summary for: test_FD002.csv

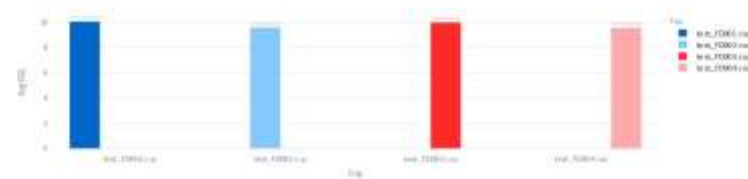
Processing summary for: test_FD003.csv

Processing summary for: test_FD004.csv

Average RUL per Test File

File	Engines	avg RUL	Min RUL	Max RUL	Mean RUL
1 test_FD001.csv	100	10.00	0.00	9.99	50.47
1 test_FD002.csv	200	8.07	0.00	9.99	50.89
1 test_FD003.csv	100	9.99	0.00	9.99	50.90
1 test_FD004.csv	200	9.99	0.00	9.99	50.44

Average Predicted RUL Comparison



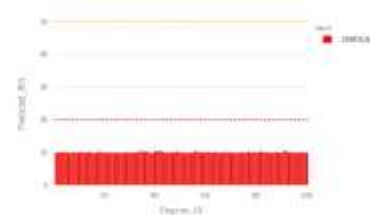
Detailed File Exploration

Select a test file to explore details

test_FD001.csv

Detailed View: test_FD001.csv

RUL for Each Engine



Alert Status Distribution



Figure3: PrognosAI Average RUL per Test File and Detailed File Exploration

RESULTS AND PERFORMANCE

- Metrics:
 - - R^2 : >0.95
 - - RMSE: Low
 - - MAE: Low
- Highlights:
 - - Stable LSTM performance
 - - Accurate RUL predictions with low error
 - - Effective alert-based decision support

CONCLUSION AND FUTURE SCOPE

- Conclusion:
- PrognosAI effectively predicts Remaining Useful Life (RUL) using AI-based modeling and visualization.
- Future Enhancements:
 - - Real-time IoT data integration
 - - Adaptive online learning
 - - Cloud deployment and API services

ISSUES FACED

- Feature-name mismatches between training and test sets caused pipeline failures. Thanks to peers and collaborators for support.
- Scaling and sequence generation failed when sensor columns had NaNs or fewer than the required 50 time-steps.
- Model inference and plotting lagged significantly when combining multiple large CMAPSS datasets.

REFERENCES AND TOOLS USED

- Languages: Python 3.10
- Libraries: tensorflow, keras, pandas, numpy, sklearn, matplotlib, plotly, streamlit, joblib, reportlab
- Dataset: NASA CMAPSS
- Environment: Jupyter Notebook / VS Code / Streamlit

The background is a gradient of dark blue to purple, speckled with small white dots resembling stars. In the top right corner, there is a large, faint circular graphic with concentric rings and numerical markings from 0 to 210. In the bottom right, there is a smaller circular graphic with concentric rings and an arrow. In the bottom left, there is another circular graphic with concentric rings and an arrow. The text "Thank You" is centered in a white, serif font.

Thank You