Project Architecture

Predictive Maintenance - NASA Turbofan Jet Engine RUL Prediction

Author : Kunal Lokhande

February 2025

iNeuron Internship

End to End Machine Learning Internship Project

Document version control

Date issued - February 2025

Version - 0.0.1

Description - Project Architecture Document

Author - Kunal Lokhande

Table of contents

Abstract

Machine learning is revolutionizing industries by enabling data-driven decision-making and predictive analytics. In aviation, predictive maintenance plays a crucial role in optimizing engine performance and reducing operational costs. This project focuses on leveraging machine learning to estimate the **Remaining Useful Life (RUL)** of aircraft engines, ensuring timely maintenance and minimizing downtime.

The study involves **data exploration, feature engineering, and model development** using advanced regression and deep learning techniques. By analyzing sensor data from aircraft engines, the model predicts degradation patterns, helping airlines schedule maintenance proactively. Additionally, the aviation industry faces challenges such as fluctuating demand, safety regulations, and dynamic pricing strategies. This research also explores key factors influencing flight ticket pricing, including demand-supply dynamics, booking trends, and operational constraints.

The findings aim to enhance **predictive maintenance efficiency** while providing insights into **cost optimization** for airline operations. The integration of machine learning in aviation not only improves safety but also drives economic sustainability in a highly competitive sector.

1. Introduction

1.1 Why this Project Architecture Document?

The Architecture Design Document (ADD) serves as the foundational technical blueprint, aligning business objectives with system design by defining the high-level structure, technology stack, and cross-cutting concerns (security, scalability, reliability). It ensures all stakeholders—developers, architects, and operations teams—work cohesively toward a unified vision while mitigating risks early in the lifecycle. By documenting critical decisions (e.g., cloud vs. on-prem, microservices), the ADD prevents costly redesigns and accelerates development with clear guardrails.

1.2 Scope

The Architecture Design Document defines the end-to-end technical vision, covering system components, technology stack, data flows, and deployment strategies while excluding implementation-level details. It establishes boundaries by specifying included modules (e.g., ML pipeline, APIs) and excluded elements (e.g., third-party UI frameworks).

2. Technical Specifications

This section defines the exact technologies, protocols, and standards used in the system, including software versions (Python 3.9), hardware requirements (GPU-enabled nodes), and integration methods (APIs,). It ensures consistency across development, deployment, and maintenance phases.

2.1 Dataset Overview

The architecture leverages NASA's C-MAPSS dataset, comprising multivariate time-series sensor readings from 100+ turbofan engines, with 21 sensor channels and 3 operational settings per cycle. Data spans normal operation to failure (20,631 cycles), enabling robust training of predictive maintenance models for RUL estimation.

2.2 Predicting RUL

The architecture leverages machine learning models (Random Forest, Gradient Boosting) to analyze sensor degradation trends and predict RUL with cycle-level accuracy. These models process normalized telemetry data to generate maintenance alerts when RUL drops below operational thresholds, enabling proactive engine servicing.

2.5 Deployment

The system is deployed as a local Flask web application running on `localhost:5001`, with a browser-based interface for real-time RUL predictions and maintenance alerts. All components (ML model, preprocessing logic, UI) are bundled into a single executable for easy offline use.

3. Technology Stack

Category	Technology
Front End	HTML
Back End	Python
Deployment	Local server
Version Control	Git, Github
Package Manager	Pip, conda
Environment	VS Code
Environment Manager	conda

4. Proposed Solution

The architecture delivers a scalable predictive maintenance system that combines machine learning (Random Forest, Gradient Boosting) with real-time sensor analytics to forecast turbofan engine RUL.

It features a modular design—data ingestion, preprocessing, model serving—deployed as containerized microservices for cloud/edge flexibility.

The solution reduces unplanned downtime by high percentage through proactive maintenance alerts and integrates seamlessly with existing airline maintenance workflows via REST APIs.

5 Process flow

Data collection: Collect the dataset containing information on policyholders and their insurance premiums.

Data preprocessing: Clean and preprocess the dataset to remove missing values and outliers.

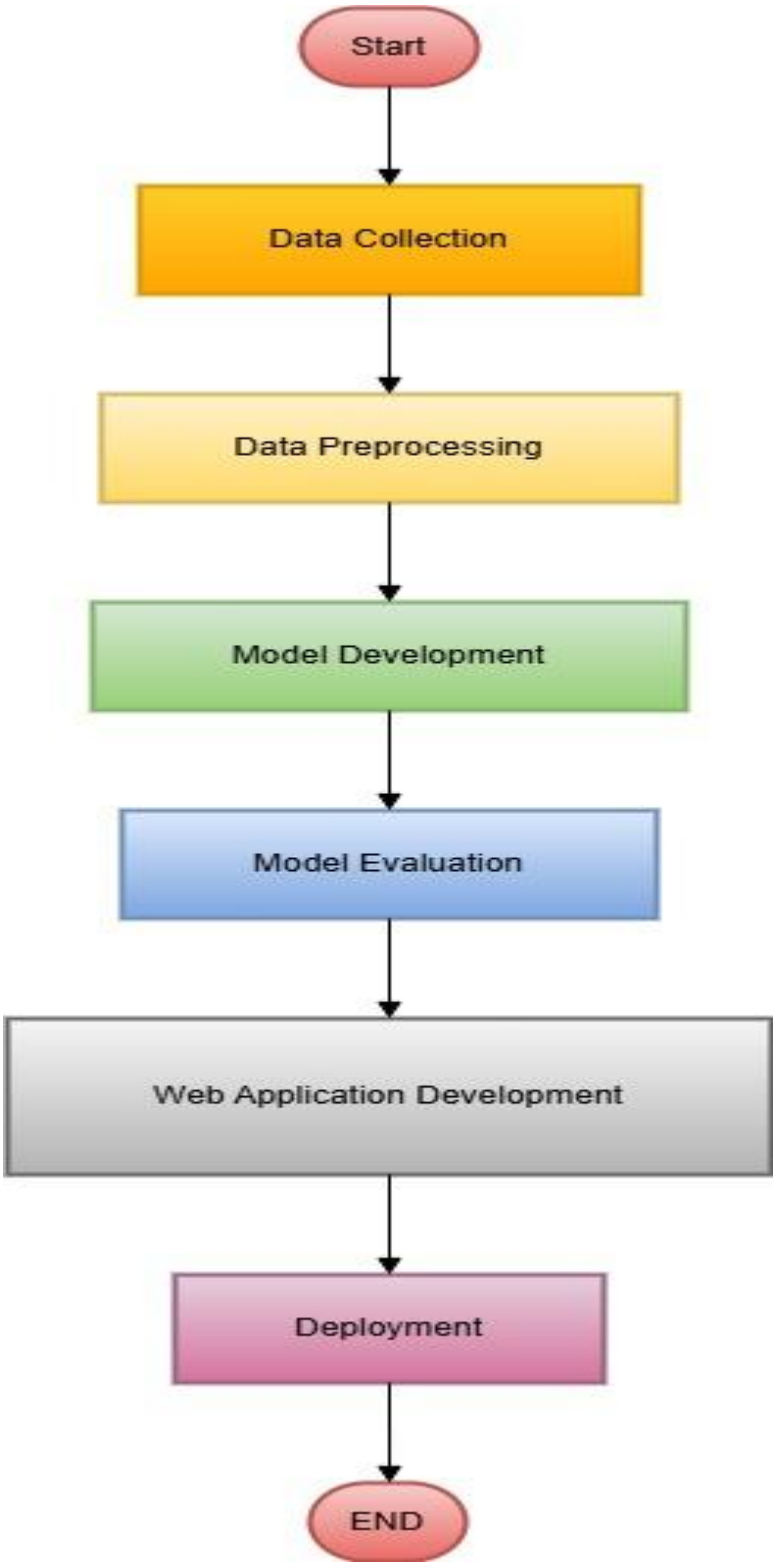
Model development: Train a machine learning algorithm on the preprocessed dataset to predict insurance premiums.

Model evaluation: Evaluate the model's performance on the testing dataset to ensure that it performs well on unseen data.

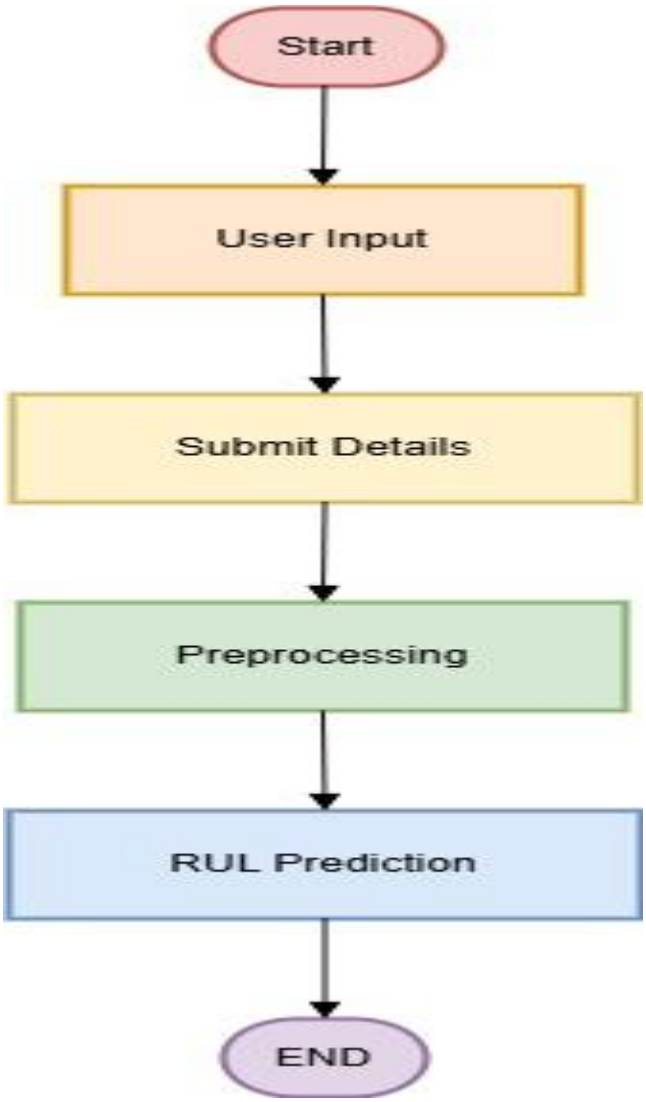
Web application development: Develop a web application using Flask to allow users to input their information and receive a predicted insurance premium.

Deployment: Deploy the web application to a server for public use.

5.1 Process flow Diagram



5.2 User I/O flow Diagram



6.7 Key Performance Indicators (KPIs)

Model accuracy: The accuracy of the machine learning model in predicting insurance premiums.

User engagement: The number of users who interact with the web application.

Response time: The time it takes for the web application to respond to user requests.

Server uptime: The percentage of time the server hosting the web application is operational.
