# 🔋 Battery Health Prediction using RUL Models and Time Series Forecasting

**DevifyX Assignment | Submitted by: ONKAR JAMMA**

**Introduction**

This project focuses on building an intelligent system for predicting the Remaining Useful Life (RUL) of batteries using machine learning techniques on time series data. The core objective is to estimate how many operational cycles a battery can sustain before failure. Predictive maintenance like this is vital in industries such as aerospace, automotive, and renewable energy. The project involves data preprocessing, exploratory data analysis (EDA), machine learning model training (XGBoost), performance evaluation, and optional deployment via FastAPI and visualization using Streamlit. Automation and hyperparameter tuning are implemented through Prefect and Optuna, respectively.

---

**Dataset Overview**

The data used in this project is taken from NASA's CMAPSS FD001 dataset, which consists of multivariate time series data representing the degradation behavior of engines. Each row in the dataset corresponds to one time step of an individual engine (battery) unit and includes an ID, the current cycle number, three operating conditions (op_set1, op_set2, op_set3), and 21 sensor readings. The dataset ends when the battery fails, and hence, the goal is to predict how many more cycles remain from any given point in its lifetime. The RUL is not explicitly provided and must be calculated based on the maximum cycle count for each unit.

---

**Project Structure**

The project is structured in a modular way. The data/ folder contains the raw and preprocessed data files, while the models/ folder stores the trained machine learning model. The notebooks/ directory is used for performing EDA and model training in Jupyter notebooks. API and dashboard files are placed in separate folders for backend and frontend applications. Other files include the requirements.txt for dependencies, .gitignore for version control exclusions, and README.md for repository documentation.

---

**Data Preprocessing**

The preprocessing stage involves reading the raw FD001 dataset and assigning meaningful column names. Each battery unit's maximum cycle is determined to calculate the Remaining Useful Life (RUL) for every time step. The RUL is calculated as the difference between the maximum cycle and the current cycle. Next, we performed Min-Max normalization to scale all sensor readings and operational settings between 0 and 1. This makes the model training process more stable and faster. Low-variance sensors that do not contribute useful information were removed.

---

**Feature Engineering**

Effective feature selection was crucial for improving model performance. From the 21 sensors, only the most informative ones were selected based on their correlation with RUL and visible degradation patterns. These included sensor2, sensor3, sensor5, sensor7, sensor11, sensor12, and sensor15, along with the three operational settings. We also considered extracting statistical features like rolling averages, standard deviations, and cycle-based derivatives, although the basic model was kept simple for interpretability and time efficiency.

---

**Exploratory Data Analysis (EDA)**

EDA was performed to understand degradation patterns across sensors. Line plots of sensor values over cycles revealed that certain sensors, like sensor2 and sensor3, displayed a clear decline as the battery approached the end of its life. Correlation heatmaps were used to identify relationships among sensors and their relevance to the target RUL. Distribution plots helped identify skewed sensor data. These visualizations guided feature selection and model assumptions.

---

**Modeling and Training**

We used XGBoost, a powerful gradient boosting regression algorithm, to model the RUL based on the selected features. XGBoost was chosen for its speed, performance, and ability to handle structured data. The dataset was split into training and test sets in an 80:20 ratio. The model was trained with 100 estimators, a max depth of 5, and a learning rate of 0.1. It showed good generalization performance without overfitting. The model was then saved using joblib for future inference.

---

**Model Evaluation**

The model was evaluated using three primary regression metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared ($R^2$) Score. The XGBoost model achieved an MAE of approximately 29.63 cycles, an RMSE of around 34.7 cycles, and an $R^2$ score of 0.83 on the test set. These results indicate the model was able to capture the underlying degradation trends effectively and can reliably predict the remaining life of batteries.

---

**API Deployment (FastAPI)**

To make the model accessible for real-time predictions, a REST API was developed using FastAPI. The API has an endpoint /predict that accepts a JSON input with all required sensor and operational values. It returns the predicted RUL as a response. The API was tested using Swagger UI and Postman, ensuring smooth and correct predictions. The FastAPI app enables backend integration for industrial or cloud-based deployment scenarios.

---

**Interactive Dashboard (Streamlit)**

A user-friendly dashboard was built using Streamlit. It allows users to simulate battery health by adjusting sliders for sensor values and operational conditions. Upon clicking the “Predict RUL” button, the system displays the predicted number of cycles left. The dashboard includes a clean sidebar layout, real-time input visualization, and animated success messages for better user interaction. This tool makes it easier for non-technical users to interact with the prediction system.

---

**Automation and Optimization (Bonus Features)**

The project includes optional automation with Prefect, allowing the entire preprocessing, training, and deployment pipeline to be run as a flow. Additionally, Optuna was used for hyperparameter optimization, enabling efficient exploration of the best model configurations. These tools contribute to the reproducibility and scalability of the system in a production environment.

---

**Technologies Used**

The project was implemented in Python 3.10. The major libraries used include Pandas, NumPy, Scikit-learn, XGBoost, Matplotlib, and Seaborn for data science tasks. FastAPI and Uvicorn were used for backend API development. Streamlit was used for dashboard creation. Prefect and Optuna were optionally integrated for automation and hyperparameter tuning, respectively. The entire codebase was version-controlled using Git and organized for GitHub deployment.

---

**Conclusion**

This project successfully demonstrates a real-world machine learning pipeline for battery health prediction. It includes essential components such as data preprocessing, feature engineering, modeling, evaluation, and deployment. The XGBoost model performed well and was efficiently deployed via REST API and dashboard. This approach can be scaled and adapted for predictive maintenance in other domains, such as electric vehicles, drones, and industrial machinery. Future work may involve integrating deep learning architectures, real-time data ingestion, and cloud deployment.