Low Level Design(LLD)

Jet Engine Remaining Useful Life(RUL) Prediction

Revision Number: 1.0
Last date of revision: 27/03/2023
Written By: Abhijeet Srivastav

Document Version Control

Change Record:

Date Issued	Version	Description	Author
27/03/2023	1.0	Document Initialized	Abhijeet Srivastav
02/03/2023	1.0	Architecture Defined	Abhijeet Srivastav
03/03/2023	1.0	Architecture Describe	Abhijeet Srivastav
10/03/2023	1.0	Unit Test Case	Abhijeet Srivastav

Reviews:

Date Issued	Version	Description	Author
02/04/2023	1.0	Document Content	Abhijeet Srivastav

Approval Status:

Date Issued	Version	Reviewed By	Approved By

Contents

1. Introduction

1.1. Why this High Level Design Document ?

The purpose of this Low Level Design Document (LLD) is to give an internal logical design of the actual program code for the Remaining Useful Life (RUL) prediction system. LLD describes the class diagrams with the methods, components and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

The LLD will:

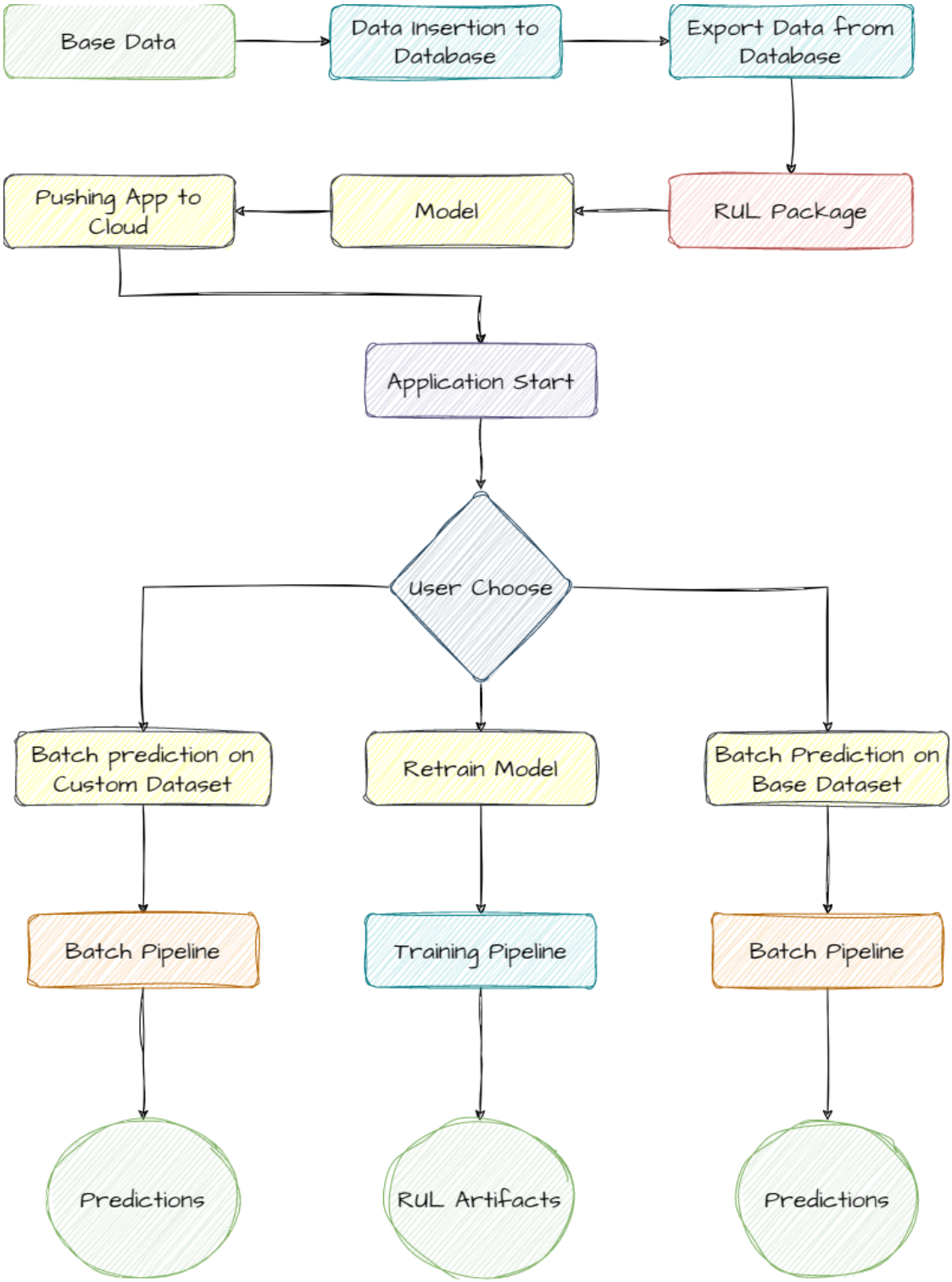
- Describe functionality of every component of the RUL prediction system
- Describe the relation between each component
- Describe the utility classes and supporting function
- Describe the model structure for database
- Describes the experimentation approach
- Describe the user interface being implemented

1.2. Scope

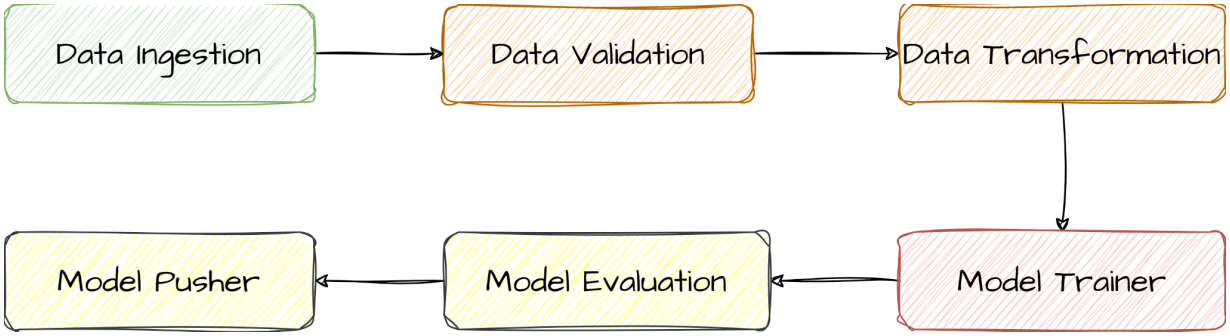
The LLD is a component level design process that follows a step-by-step refinement process. This process documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

2. Architecture

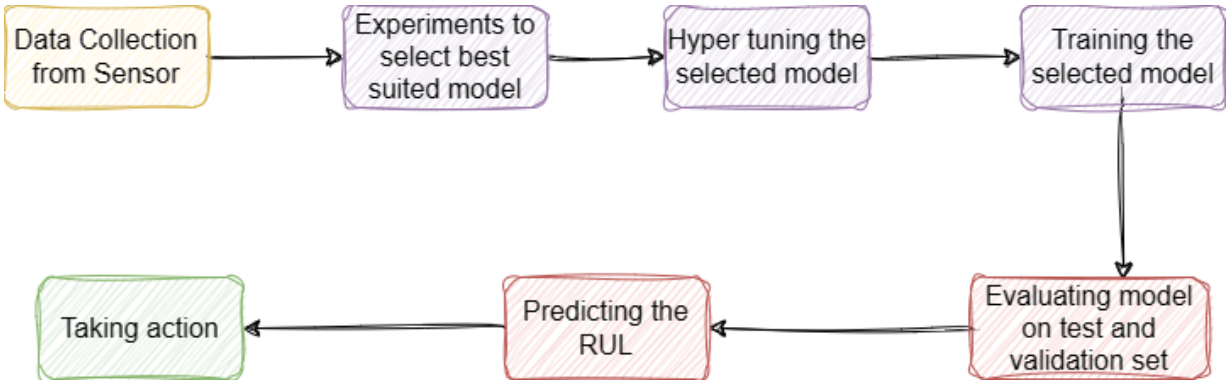
2.1. Architecture Flow



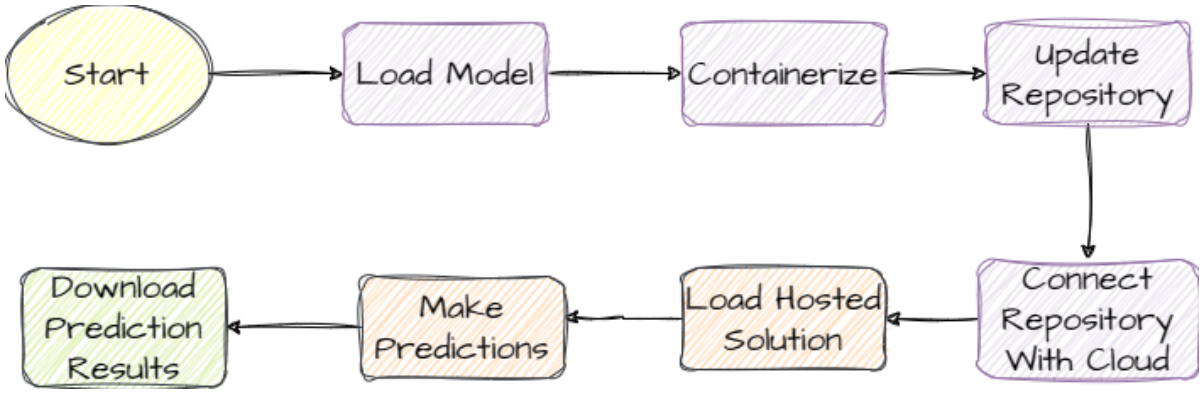
2.2. RUL Package Flow



2.3. Experiment Flow



2.4.Deployment Flow



3. Architecture Description

3.1. Data Description

The base data set is the well-known NASA asset degradation modeling public data set. Run-to-Failure simulated data from turbofan jet engines is included. Utilizing C-MAPSS, an engine degradation simulation was performed. Four distinct sets were simulated under various arrangements of operational circumstances and fault types. records data from multiple sensor channels to track the evolution of faults. The Prognostics CoE at NASA Ames supplied the given dataset.

This dataset contains 20631 rows and 26 columns. Each row is a snapshot of data taken during a single operational cycle, each column is a different variable. The columns correspond to:

- unit number
- time, in cycles
- operational setting 1
- operational setting 2
- operational setting 3
- sensor measurement 1
- sensor measurement 2
- ……
- sensor measurement 26

3.2. Data Insertion into Database

- **Database Creation:** Create MongoDB Cluster with specified name, IP filter, Region, Users and Accessibility.
- **Database Connection:** Connect database to RUL projects *data_dump.py* script using client and credentials.
- **Create Database:** Create MongoDB database table using client with specified name.
- **Insertion:** Insert base data to database.

3.3. Export Data from Database

Data stored in MongoDB database is exported as a **CSV** to be used for **experimentation** or **RUL package**.

3.4. RUL Package

RUL package contains various components, configurations and artifacts of the machine learning pipeline.

3.5. Model

Model with best **r2_score** and its associated transformer pipeline is saved as objects in **saved_models** and **artifact** directory which are produced by the **RUL package,** which are later utilized by the **app** to make predictions or retrain the model.

3.6. Pushing App to Cloud

RUL package and **flask app** are containerized and pushed to the **Github repository main branch.** Later deployed to the cloud using the same repository.

3.7. App Start & User Choices

Deployed App is initiated for users. User has three possible options to choose from.

3.8. Batch prediction on Custom Dataset

This option leads to activation of the **batch prediction pipeline** on a **custom dataset**. Therefore the user is prompted to provide the data on which they want prediction.

A quick data shape check is provided onto the given dataset to verify whether its fit for the trained model to predict or not by comparing it to the **base dataset**.

Generated predictions are displayed and can be downloaded as a **csv** file.

3.9. Batch prediction on Base Dataset

This option leads to activation of the **batch prediction pipeline** on the **base dataset** on which model was trained.

Generated predictions are displayed and can be downloaded as a **csv** file.

3.10. Retraining

This option leads to activation of the **training pipeline** on the **base dataset** on which the model will retrain. The pipeline will activate the **rul package** and export updated data from the database to retrain the model. If the newly trained model is found better then the deployed one then redeployment of the latest trained model occurs else latest trained model is saved as artifact and no redeployment happens.

4. RUL Package Description

4.1. Data Ingestion

Connection with MongoDb database is established using the client. Then collection is fetched as dataframe and missing values are dealt**.** Theirafter this dataframe is exported to **feature store** as **rul.csv** as artifact, also after splitting it into train and test set they are exported as **train.csv** and **test.csv** as artifact.

4.2. Data Validation

Drift report is generated as an artifact based on several validations like threshold, shape and required columns are performed on the ingested dataset rul.csv from the previous component with base dataset.

4.3. Data Transformation

A **transformer pipeline object** is created with a **Simple Imputer** & **Min Max Scaler** at its core. **RUL feature** is appended to validated ingested train and test dataset. As well as several irrelevant features such as settings, constant sensor, unit number and time cycles are dropped from the train and test dataset. Theirafter using transformer pipeline object transformation is performed on both dataset and they are exported as **train and test numpy array** as artifact

4.4. Model Trainer

A **random forest regressor model** object is created with following configuration which we found to best suited during experimentation:

- *criterion= “poisson”*
- *max_features= “sqrt”*
- *ccp_alpha= 0.0*
- *n_estimators= 100*

Then we load the **train test array** from data transformation artifacts and perform **feature segregation, fitting, prediction and r_2 score calculation**. If this r_2 score is less than expected score we reject the model as it would be **underfitting** and if this r_2 score is more overfitting threshold then also we would reject it due to **overfitting**. If not, the both case model would be accepted.

4.5. Model Evaluation

This component checks whether there is already any saved model or if the pipeline is being run for the first time. If it is being run for the first time, the trained model along with its transformed object would be saved as an **artifact** and also to **saved_model**s directory to be deployed.
If there's an already saved model then the **r2 score** of both models is compared and if the latest trained one is found better then its saved as an artifact else rejected.

4.6. Model Pusher

Latest trained model is pushed to the saved model directory for deployment if it is found better than the **latest saved model** by the **model evaluation component**.

4.7. Artifact Entity

It defines all the **artifacts** which would be generated by each individual component of the **rul package.**

4.8. Configuration Entity

It defines all the **configuration** or simply **input** required by each individual component of the **rul package** to perform their respective task.

5. Experiment Description

Experimentation is performed on the **base dataset:**

- To check the required steps to perform data preprocessing
- Evaluate various model pipelines containing combinations of the various imputeres, scalers to verify our hypothesis and select the best suited one
- Hypertune the best suited model to select the bests parameters
- Validate the best tuned model on validation set

6. Unit Test Cases

Test Case Description	Prerequisite	Expected Result
Verify whether the application URL is accessible to the user	1. Application URL should be defined	Application URL should be accessible to the user
Verify whether the Application loads completely for the user when the URL is accessed the URL is accessed	1. Application URL is accessible 2. Application is deployed	The Application should load completely for the user when the URL is accessed
Verify whether retraining feature is working	1. Application URL is accessible 2. Application is deployed	The application should load the result screen with a message of retraining complete and specify whether the newly trained model is accepted or not.
Verify whether Batch prediction feature is working	1. Application URL is accessible 2. Application is deployed	The application should load the result screen displaying the rul features with all the input features
Verify whether Custom Batch prediction feature is working	1. Application URL is accessible 2. Application is deployed 3. Appropriate shape and feature having dataset	The application should load the result screen displaying the rul features with all the input features if dataset has appropriate shape and features