iNeuron

Low Level Design

Predictive Maintenance

Written By	Alok Chauhan
Document Version	0.1
Last Revised Date	21 − Sept -2021

Document Control

Change Record:

Version	Date	Author	Comments
0.1	22 – Sept - 2021	**Alok Chauhan**	Introduction & Architecture defined

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

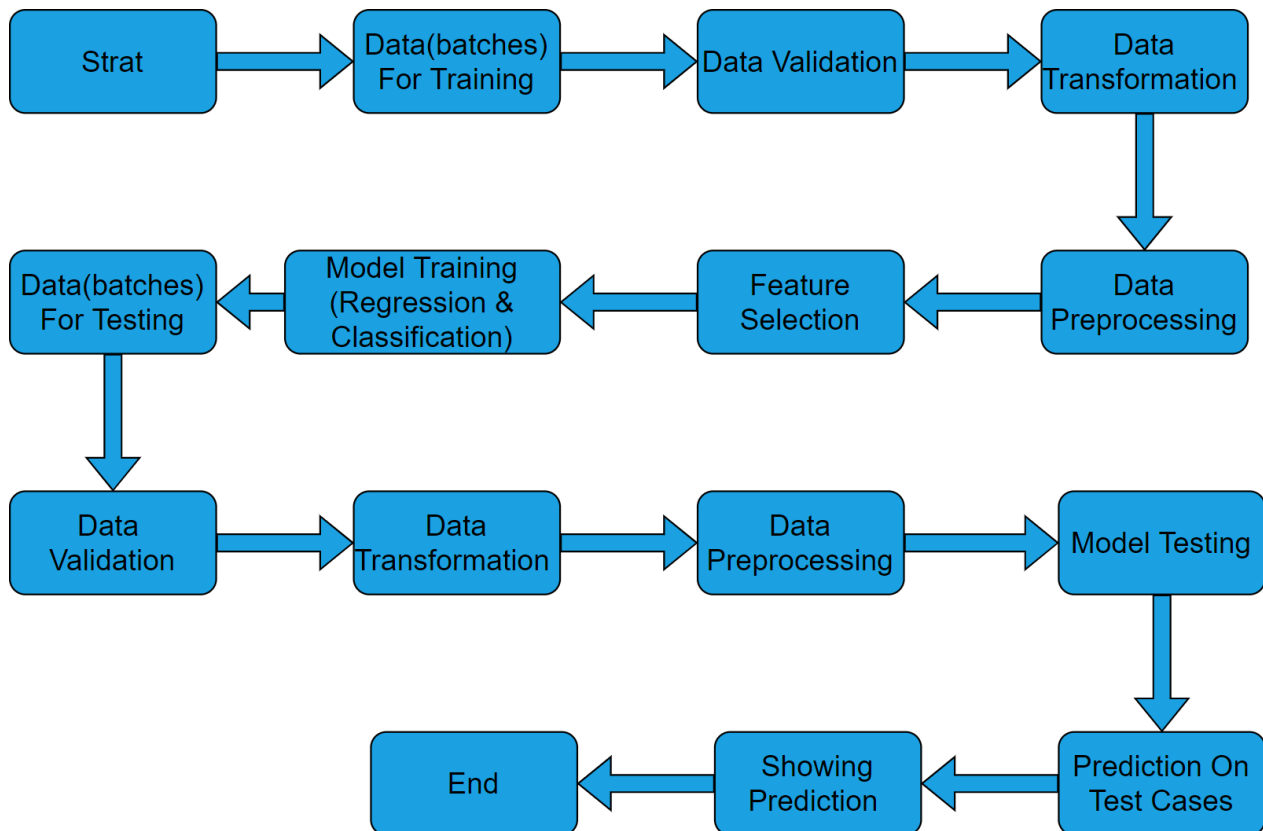# Contents

1. Introduction

1.1. What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation System. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

1.2. Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

2. Architecture

```
┌──────────┐     ┌──────────────┐     ┌──────────────┐     ┌──────────────┐
│  Strat   │ ──▶ │ Data(batches)│ ──▶ │Data Validation│ ──▶ │     Data     │
│          │     │ For Training │     │              │     │Transformation│
└──────────┘     └──────────────┘     └──────────────┘     └──────────────┘
                                                                    │
                                                                    ▼
┌──────────────┐  ┌──────────────┐     ┌──────────────┐     ┌──────────────┐
│ Data(batches)│◀─│Model Training│ ◀── │   Feature    │ ◀── │     Data     │
│ For Testing  │  │ (Regression &│     │   Selection  │     │Preprocessing │
│              │  │Classification)│    │              │     │              │
└──────────────┘  └──────────────┘     └──────────────┘     └──────────────┘
       │
       ▼
┌──────────────┐     ┌──────────────┐     ┌──────────────┐     ┌──────────────┐
│     Data     │ ──▶ │     Data     │ ──▶ │     Data     │ ──▶ │ Model Testing│
│  Validation  │     │Transformation│     │Preprocessing │     │              │
└──────────────┘     └──────────────┘     └──────────────┘     └──────────────┘
                                                                    │
                                                                    ▼
              ┌──────────┐     ┌──────────────┐     ┌──────────────┐
              │   End    │ ◀── │    Showing   │ ◀── │ Prediction On│
              │          │     │   Prediction │     │  Test Cases  │
              └──────────┘     └──────────────┘     └──────────────┘
```

3. Architecture Description

3.1. Data Description

Engine degradation simulation was carried out using C-MAPSS. Four different were sets simulated under different combinations of operational conditions and fault modes. Records several sensor channels to characterize fault evolution. The data set was provided by the Prognostics CoE at NASA Ames

The data are provided as a zip-compressed text file with 26 columns of numbers, separated by spaces. Each row is a snapshot of data taken during a single operational cycle, each column is a different variable.

3.2. Data Transformation/Validation

1. Convert the txt into csv file.
2. Labeling columns.
3. Marge 'Test' and 'RUL_test' data.

3.3. Data Pre-processing

Data Pre-processing steps we could use are Null value handling, Calculating Rul Of train data set, covert the RUL into Binary RUL ,scaling data.

3.4. Feature Selection

Analysis of data and finding the best features. Dropping not useful features. Creating train and test variables for model training.

3.5 Model Training

After the creation of the Train and test data we will. Apply training data on many model and select the best model whose MSE is low and accuracy is high.

After selection of the best model we will perform model training on training data.

3.6 Test Data

Collecting Test Data

3.7 Data Validation

Here Data Validation will be done, given by the user

3.8 Data Transformation
The model created during training will be loaded, data will transformed same as Train data.

3.9 Data Pre-Processing

As based on training data test will be pre processed

3.10   Model Testing

Testing the model and finding the test accuracy of the model which is trained on training data.

3.11   Predicting on test cases

Predicting will made on model which is trained on trained data and validated on test data.

3.12   Deployment

We will be deploying the model to Amazon Azure using Gradio.
This is a workflow diagram for the Recipe Recommendation..

4. Unit Test Cases

Test Case Description	Pre-Requisite	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	1. Application URL is accessible 2. Application is deployed	The Application should load completely for the user when the URL is accessed
Verify whether the User is able to sign up in the application	1. Application is accessible	The User should be able to sign up in the application
Verify whether user is able to successfully login to the application	1. Application is accessible 2. User is signed up to the application	User should be able to successfully login to the application
Verify whether user is able to see input fields on logging in	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should be able to see input fields on logging in
Verify whether user is able to edit all input fields	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should be able to edit all input fields
Verify whether user gets Submit button to submit the inputs	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should get Submit button to submit the inputs
Verify whether user is presented with recommended results on clicking submit	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should be presented with recommended results on clicking submit
Verify whether the recommended results are in accordance to the selections user made	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	The recommended results should be in accordance to the selections user made
Verify whether user has options to filter the recommended results as well	1. Application is accessible 2. User is signed up	User should have options to filter the recommended results as well

	to the application 3. User is logged in to the application	
Verify whether KPIs modify as per the user inputs for the user's health	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	KPIs should modify as per the user inputs for the user's health
Verify whether the KPIs indicate details of the suggested recipe	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	The KPIs should indicate details of the suggested recipe