Low Level Design (LLD)

Predictive Maintenance

Revision Number: 1.0
Last Data of Revision: 25/07/2022

Jinendra Sontakke

Document Version Control

Date Issued	Version	Description	Author
25/07/2022	V1.0	HLD – V1.0	Jinendra Sontakke

Contents

Abstract

A jet engine is the most important part of the plane, and it helps to move the airplane forward with a great force that is produced by a tremendous thrust Maintenance of equipment is a critical activity for any business involving machines. And causes the plane to fly very fast. During operation, degradation occurs in each of the components. If the degradation level is any component exceeds a threshold the engine is said to have failed. Therefore, the jet engines are inspected before every take-off and maintenance of equipment is critical activity for any business involving machines. Predictive Maintenance is the method of scheduling maintenance based on the prediction of the failure time of any equipment. The prediction can be done by analyzing the data measurements from the equipment. Machine learning is technology by which outcomes can be predicted based on a model prepared by training it on past input data and its output behavior. The model developed can be used to predict machine failure before it happens.

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.

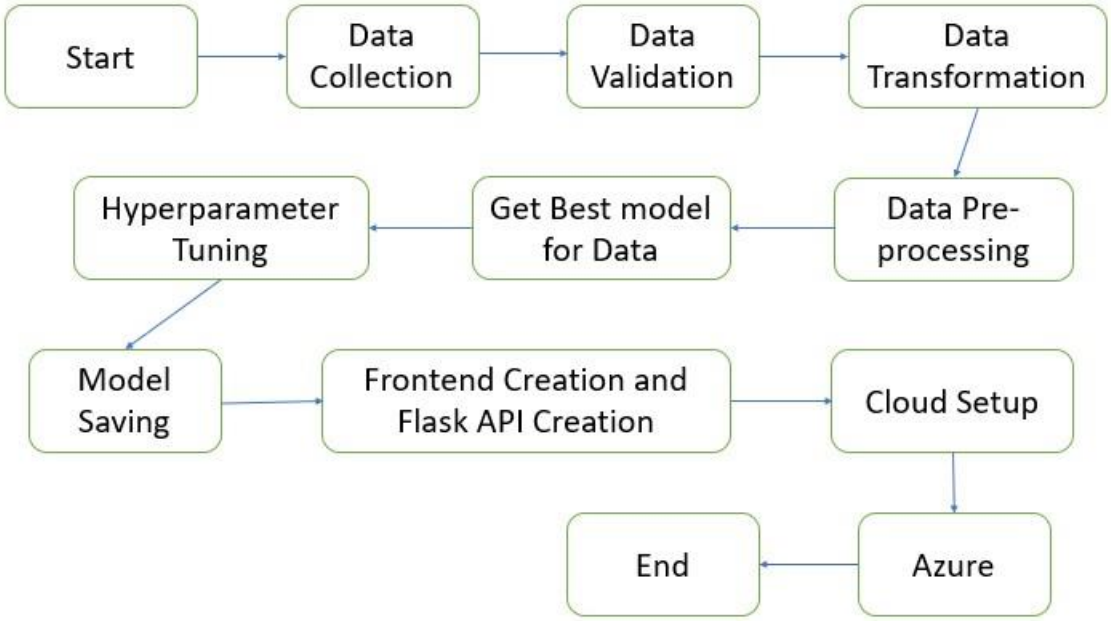
1.2 Scope

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

1.3 Constraints

Internet connection is a constraint for the application. Since the application fetched the data from the database, it is crucial that there is an Internet connection for the application to function. Since the model can make multiple requests at same time, it may be forced to queue incoming requests and therefore increase the time it takes to provide the response.

2. Architecture



3. Architecture Description

3.1 Data Description

We have 1.6 lakh Dataset row columnar data includes Engine ID, Cycle, Operational Setting1, Operational Setting2, Operational Setting3, Sensor Measure 1, Sensor Measure 2, Sensor Measure 3, Sensor Measure 4, Sensor Measure 5, Sensor Measure 6, Sensor Measure 7, Sensor Measure 8, Sensor Measure 9, Sensor Measure 10, Sensor Measure 11, Sensor Measure 12, Sensor Measure 13, Sensor Measure 14, Sensor Measure 15, Sensor Measure 16, Sensor Measure 17, Sensor Measure 18, Sensor Measure 19, Sensor Measure 20, Sensor Measure 21, Sensor Measure 22, Sensor Measure 23. This is given in the text format we converted it into comma separated value format (.csv) . these data is given by company team which contains both the test data and train data.

3.2 Data Cleaning / Data Transformation

In this cleaning process, we have cleaned up all the data because data which contain null value because of this bad format the machine will not recognized it. So dara cleaning is after that data transformation process come where we will convert our original dataset which is in text format to CSV format.

3.3 Exploratory Data Analysis

In EDA we have seen various insights from the data so we have selected which column is most important and dropped some of the columns by observing them spearman rank correlation and plotting their heatmap.

3.4 Event Log

The system should log every event so that the user will know what process is running Internally. Logging is implemented using python's standard logging library. Initial step-by-step description:-

- The system should be able to log each and every system flow.
- System must be able to handle logging at greater scale because it helps debugging the issue and hence it is mandatory to do.

3.5 Data Pre-Processing

Data Pre-processing steps we could use are Null value handling, Categorical to Numerical Transformation of columns , Splitting Data into Dependent and Independent Features , Robust Scaling , Remove those columns which are does not participate in model building Processes , Imbalanced data set handling, Handling columns with standard deviation zero or below a threshold, etc.

3.6 Model Creation / Model Building

After cleaning the data and completing the feature Engineering/ data Per processing. we have done splitted data in the train data and test data using method build in pre-processing file and implemented various Classification Algorithm like RandomForestClassifier and XgBoostClassifier also calculated their accuracies on test data and train data.

Algorithm	Accuracy Score
Random Forest	89.9%
K-Nearest Neighbour	83.8%
Logistic Regression	76.3%
Naïve Bayes classifier	72.1%

3.7 Hyperparameter Tuning

In hyperparameter tuning we have implemented randomized search cv or grid search cv and from that we also implemented cross validation techniques for that. From that we have choose best parameters according to hyperparameter tunning and best score from their accuracies.

3.8 Model Dump

After comparing all accuracies I have choose hyper parameterized random forest classifier as our best model by their result so I have dumped this model in a pickle file format.

3.9 Data from User

Here we will collect user's requirement to predict whether a engine is good or not their Engine Cycle, Sensor Measure 2, Sensor Measure 3, Sensor Measure 4, Sensor Measure 7, Sensor Measure 8, Sensor Measure 11, Sensor Measure 12, Sensor Measure 13, Sensor Measure 15, Sensor Measure 17, Sensor Measure 20 and Sensor Measure 21.

3.10 Data Validation

Here Data Validation will be done, given by the user

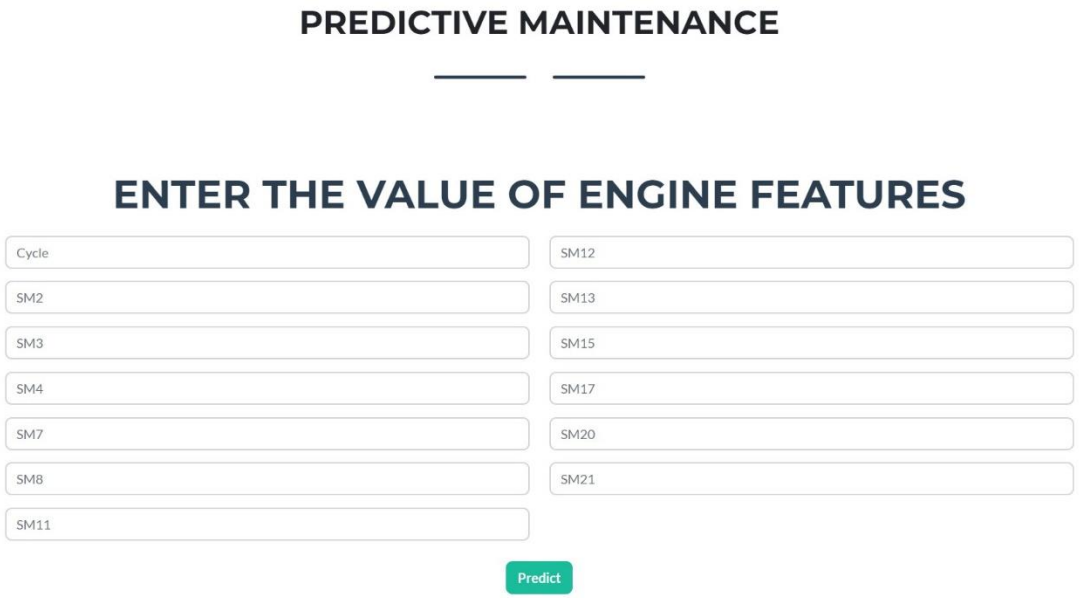
3.11 Model Call for specific input

Based on the User input will be throwing to the backend in the variable format then it converted into pandas data frame then we are loading our pickle file in the backend and predicting whether engine condition is good or not as an output and sending to our html page.

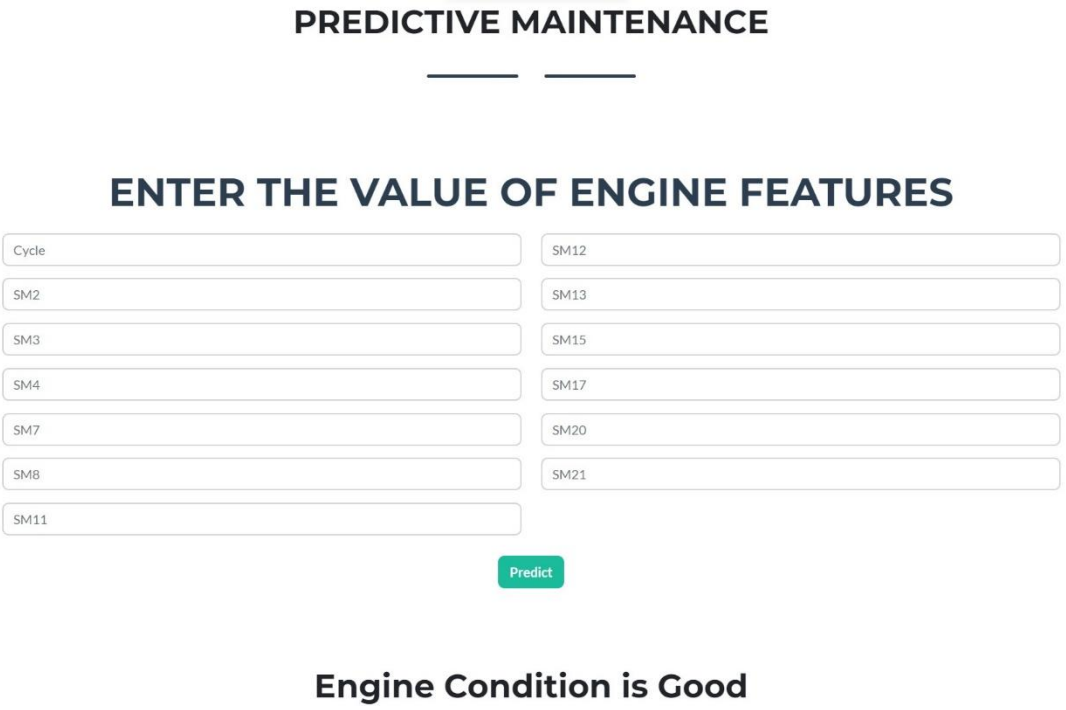
3.12 User Interface

In Frontend creation we have made a user interactive page where user can enter their input values to our application. In these frontend page we have made a form which has beautiful styling with CSS and bootstrap. These HTML user input data is transferred in variable format to backend. Made these html fully in a decoupled format

Input Page:



Output Page:



3.13 Deployment

We will be deploying the model with the help of Azure cloud platforms.

4 Technology Stack

Front End	**HTML, CSS, Bootstrap**
Back End	Flask, Pandas, Numpy, Scikit-learn etc.
Deployment	Azure

5 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 Response time of URL from backend model.	1. Application is accessible	The latency and accessibility of application is very faster we got in Azure service
Verify whether user is giving standard input.	1. Handled test cases at backends.	User should be able to see successfully valid results.
Verify whether user is able to see input fields on logging in	1. Application is accessible 2. 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 logged in to the application	User should be able to edit all input fields
Verify whether user gets Predict Back-Order button to submit the inputs	1. Application is accessible 2. 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 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 logged in to the application and database	The recommended results should be in accordance to the selections user made
Verify whether the	1. Application is	The KPIs should

KPIs indicate details Of the correct inputs	accessible 2. User is logged in to the application	indicate details of the suggested inputs to users.