Software Requirement

Specification

Project ID & Name	Engine Maintenance Predictive Analysis		
Project Manager	Jasiya Fairiz Raisa		
Customer Request Reference			
Prepared by	Khair Ahammed	**Date**	13/5/2024
Reviewed by		**Date**	

Revision History

Ver. No	Date	Prepared By	Approved By	List of changes from the Previous
0.1				
0.2				1.

Table of Contents

1. Requirement Scope Summary

The engine maintenance predictive model application aims to provide a reliable and efficient solution for predicting engine failures based on input data related to various engine parameters. The scope of this project encompasses the following key components and functionalities:

Prediction API:

- Provides a RESTful POST API endpoint for accepting input data.
- Validates input data integrity and completeness.
- Utilizes a trained predictive model to generate predictions regarding the likelihood of engine failure within a specified timeframe.
- Returns prediction results in a structured format (e.g., JSON) for consumption by client applications.

Model Training and Update:

- Supports periodic retraining of the predictive model to incorporate new data and enhance prediction accuracy.
- Automates the retraining process, including data preprocessing, feature selection, model training, and evaluation.

Performance:

- Ensures that the API responds to prediction requests within an acceptable timeframe (< 1 second) to maintain responsiveness.
- Designed to scale efficiently to accommodate a potentially large volume of concurrent prediction requests.

Reliability:

- Demonstrates high accuracy and reliability in predicting engine failures by utilizing robust machine learning algorithms.
- Equipped to handle input data anomalies and errors gracefully to prevent system

failures or inaccurate predictions.

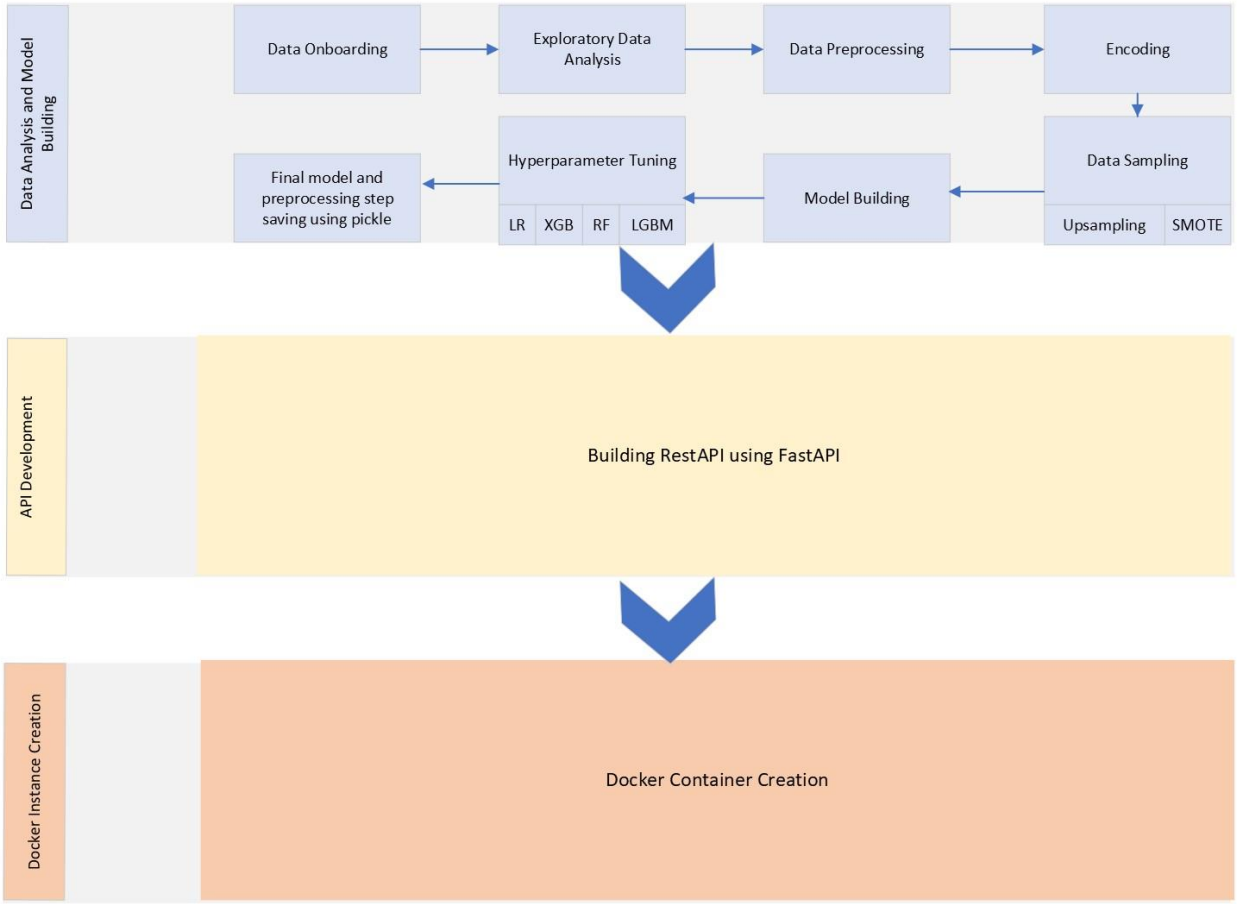
System Architecture:

- Comprises components such as the Prediction API, Model Service, and Data Pipeline to facilitate efficient prediction generation.
- Utilizes Docker containers for easy deployment and scalability, with potential deployment management using container orchestration tools like Kubernetes.

This summary outlines the primary objectives and functionalities of the engine maintenance predictive model application, aiming to deliver a reliable, scalable, and secure solution for predicting engine failures and enabling proactive maintenance strategies.

2. System Architecture

The system arhitecture provides a high-level view of the whole system, how it is designed and what steps are followed to accomplish the goal. The following figure 1 shows the system architecture:



It's a 3 layers architecture that starts with data analysis and modelling then building the RESTful API and ends with Docker Instance Creation to ease the deployment process.

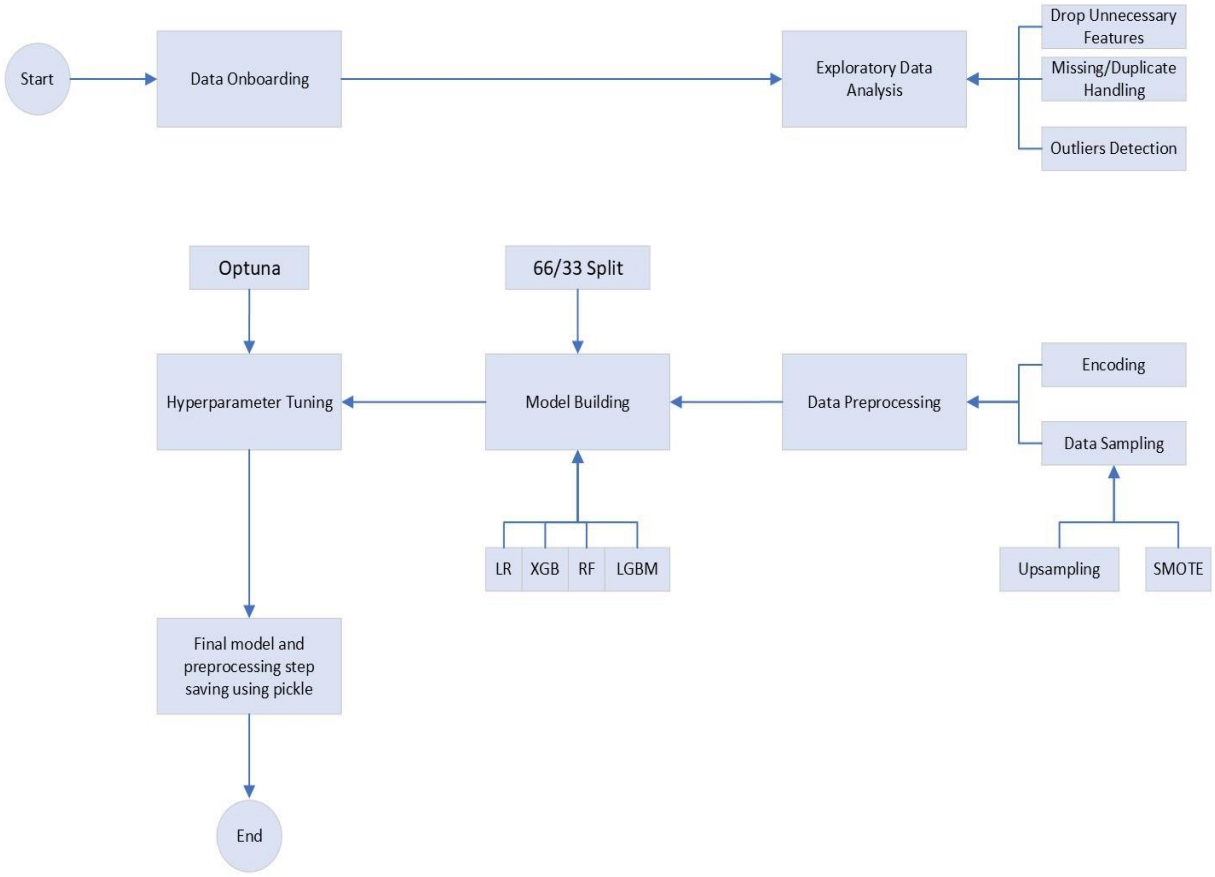
3. Functional Requirements Identification

Table 1: Functional Requirements Definition

Module Name	SRS ID	UI & Reports name
Maintenance Prediction	1.1	Model Training and Validation
	1.2	Prediction API

SRS -3.1 Model Training and Validation

3.1.1. Process Flow – Model Training and Validation



3.1.2. Model Training and Validation

SRS ID	SRS 1.1
SRS Title	Model Training and Validation
Proposed Requirement	1. Building a robust generalized model that solves the imbalance issues efficiently
Precondition	1. Must have the dataset.
Requirements	1. The data should be processed perfectly with proper code comments to understand the intuition behind the process and logic building. 2. Show proper visuals to help understand the data and to take action properly. 3. Applying a variety of algorithms to find the best one for predicting engine maintenance failure. 4. Hyperparameter tune the best model or all the models if possible, to find the best outcome from the models. 5. Finally train the final best model

	6. Save the model
Post-Condition	
Requirement Collection Method	Requirements gathering session.
Priority	High

SRS -3.2 Prediction API

3.2.2. Prediction API Detail

Description:

This API endpoint allows users to predict maintenance failure based on the engine maintenance features.

Endpoint: http://127.0.0.1:8000/predict_failure

Request Parameters:

- **Name**: product_type
  - **Type**: Character
  - **Description**: The type of the product
  - **Constraints**: Required, single character only
- **Name**: air_temperature_K
  - **Type**: float
  - **Description**: The air temperature of the environment in kelvin scale
  - **Constraints**: Required, valid float format.
- **Name**: process_temperature_K
  - **Type**: float
  - **Description**: The temperture of the process in kelvin scale
  - **Constraints**: Required
- **Name**: rotational_speed_rpm
  - **Type**: float
  - **Description**: The rotational speed of the engine in rpm

    - **Constraints**: Required
- **Name**: torque_nm
    - **Type**: float
    - **Description**: The temperture of the process in nanometer
    - **Constraints**: Required
- **Name**: total_wear_mins
    - **Type**: float
    - **Description**: Total Wear in Minutes
    - **Constraints**: Required

Request Body:

```
{
    "air_temperature": 298.5,
    "process_temperature": 309.4,
    "rotational_speed": 1360,
    "torque": 60.9,
    "tool_wear": 187,
    "type": "H"
}
```

Response:

- **Status Code**: 201 Created
- **Content Type**: application/json

Response Body:

{

“prediction": “No Failure”

}

Error Responses:

- **Status Code**: 400 Bad Request
  Content Type: application/json

- **Status Code**: 409 Conflict
    - **Content Type**: application/json
- **Status Code**: 500 Internal Server Error
  Content Type: application/json

Security:

- **Authentication**: Not required.
- **Authorization**: Not applicable.