High Level Design (HLD)
Predictive Maintenance

Contents

Abstract

We believe predictive maintenance with big data analytics can be a tremendous source of new value for asset owners and maintenance service providers. To deepen our understanding and sharpen our insights, we have jointly carried out a market survey on predictive maintenance. This involved surveying 280 companies from Belgium, Germany and the Netherlands about their current use of, and future plans for, predictive maintenance, and conducting interviews with leading companies in the field. This report presents the results of this research and our approach to successfully implementing predictive maintenance with big data. Our findings should be of interest to those responsible for the maintenance and asset management of fleets, factories and infrastructure, who are looking for new ways to increase the reliability of their assets.

1 Introduction

1.1 Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

- Present all of the design aspects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like:
    - Security
    - Reliability
    - Maintainability
    - Portability
    - Reusability
    - Application compatibility
    - Resource utilization
    - Serviceability

1.2 Scope

The HLD 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 General Description

2.1 Product Perspective

The application of big data analytics in maintenance represents predictive maintenance, as model which will help us to detect the anomalies in the society and take the necessary action.

2.2 Problem statement

In industry, prognostics and health management are key topics for anticipating asset state and avoiding downtime and breakdowns. Run-to-Failure simulation data from turbofan jet engines is included.
The C-MAPSS software was used to simulate engine degradation. Four separate sets of operational conditions and fault modes were simulated in four different ways. To characterize fault progression, record numerous sensor channels. The Prognostics CoE at NASA Ames provided the data set.
The main goal is to predict the remaining useful life (RUL) of each engine. RUL is equivalent of number of flights remained for the engine after the last data point in the test dataset.

2.3 PROPOSED SOLUTION

Predictive Maintenance (PdM) techniques are used to determine when in-service equipment need maintenance to prevent costly operational interruptions resulting from equipment failures.

While preventive maintenance is characterized by routine or scheduled checks of equipment, the PdM system relies on the collection and analysis of real-time data on the conditions of the equipment. The PdM system conducts non-interference monitoring in the background, collects operational data via the use of sensors, and transmits the collected information through cloud or designated servers for the appropriate team to monitor the equipment conditions. Control center staff can use the data to build failure models and program the PdM system to recognize these failure models as a part of the machine learning process. The PdM will develop the ability to make predictions for future maintenance.

2.4 FURTHER IMPROVEMENTS

PdM can be added with more use cases like any device or machines. It implementation will reduce the accidents

2.5 Data Requirements

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. The columns correspond to:

1) unit number

2) time, in cycles

3) operational setting 1

4) operational setting 2

5) operational setting 3

6) sensor measurement 1

7) sensor measurement 2

…

26) sensor measurement 26

Data Set Organization
Data Set: FD001

Train trjectories: 100

Test trajectories: 100

Conditions: ONE (Sea Level)

Fault Modes: ONE (HPC Degradation)

Data Set: FD002

Train trjectories: 260

Test trajectories: 259

Conditions: SIX

Fault Modes: ONE (HPC Degradation)

Data Set: FD003

Train trjectories: 100

Test trajectories: 100

Conditions: ONE (Sea Level)

Fault Modes: TWO (HPC Degradation, Fan Degradation)

Data Set: FD004

Train trjectories: 248

Test trajectories: 249

Conditions: SIX

Fault Modes: TWO (HPC Degradation, Fan Degradation)

2.6 Tools used

Python programming language and frameworks such as NumPy, Pandas, Scikit-learn, TensorFlow, Keras and Roboflow are used to build the whole model.

- Spyder is used as IDE.
- For visualization of the plots, Matplotlib, Seaborn and Plotly are used.
- Azure is used for deployment of the model.
- Front end and backend development is done using Gradio
- Sklearn is used for models
- GitHub is used as version control system.

2.7 Hardware Requirements

- Software Requirements:

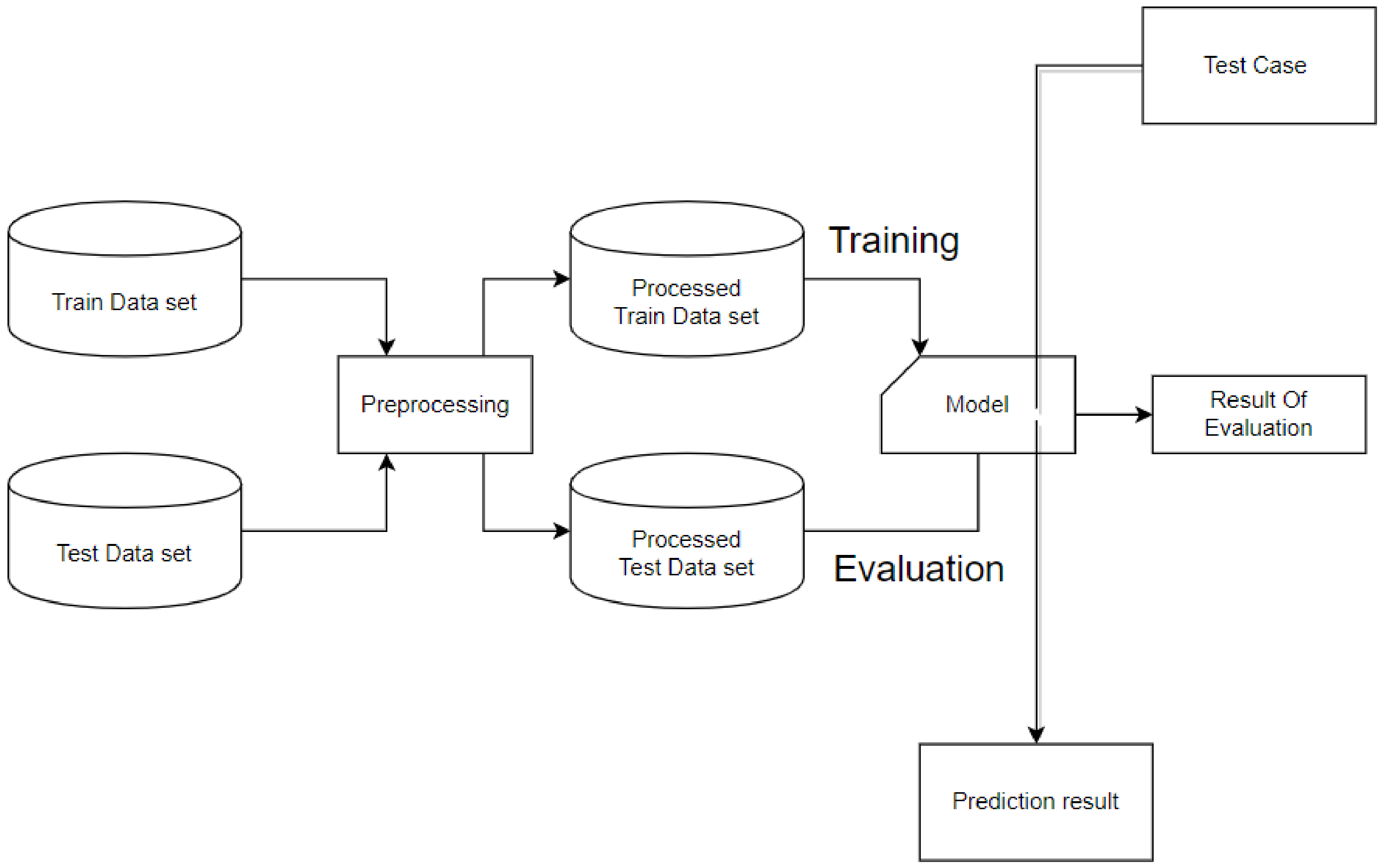
  OS: Linux or Windows 7 or higher version

- Language: Python
  - Hardware Requirement:
- Ram : 2GB or more
  - Processor: Any Intel Processor
- Hard Disk : 6GB and more
  - Platform : Jupyter Lab

3 Design Details

3.1 Process Flow

For identifying the different types of anomalies, we will use a deep learning base model. Below is the process flow diagram is as shown below.



3.2 Event log

The system should log every event so that the user will know what process is running internally.

Initial Step-By-Step Description:

1. The System identifies at what step logging required
2. The System should be able to log each and every system flow.
3. Developer can choose logging method. You can choose database logging/ File logging as well.
4. System should not hang even after using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.

3.3 Error Handling

Should errors be encountered, an explanation will be displayed as to what went wrong? An error will be defined as anything that falls outside the normal and intended usage.

3.4 Performance

3.4.1 PdM's Model performance close to accurate.
3.4.2 Accuracy is very much on very good range
3.4.3 Latency is very low (1~3 s)

3.5 Reusability

The code written and the components used should have the ability to be reused with no problems.

3.6 Application Compatibility

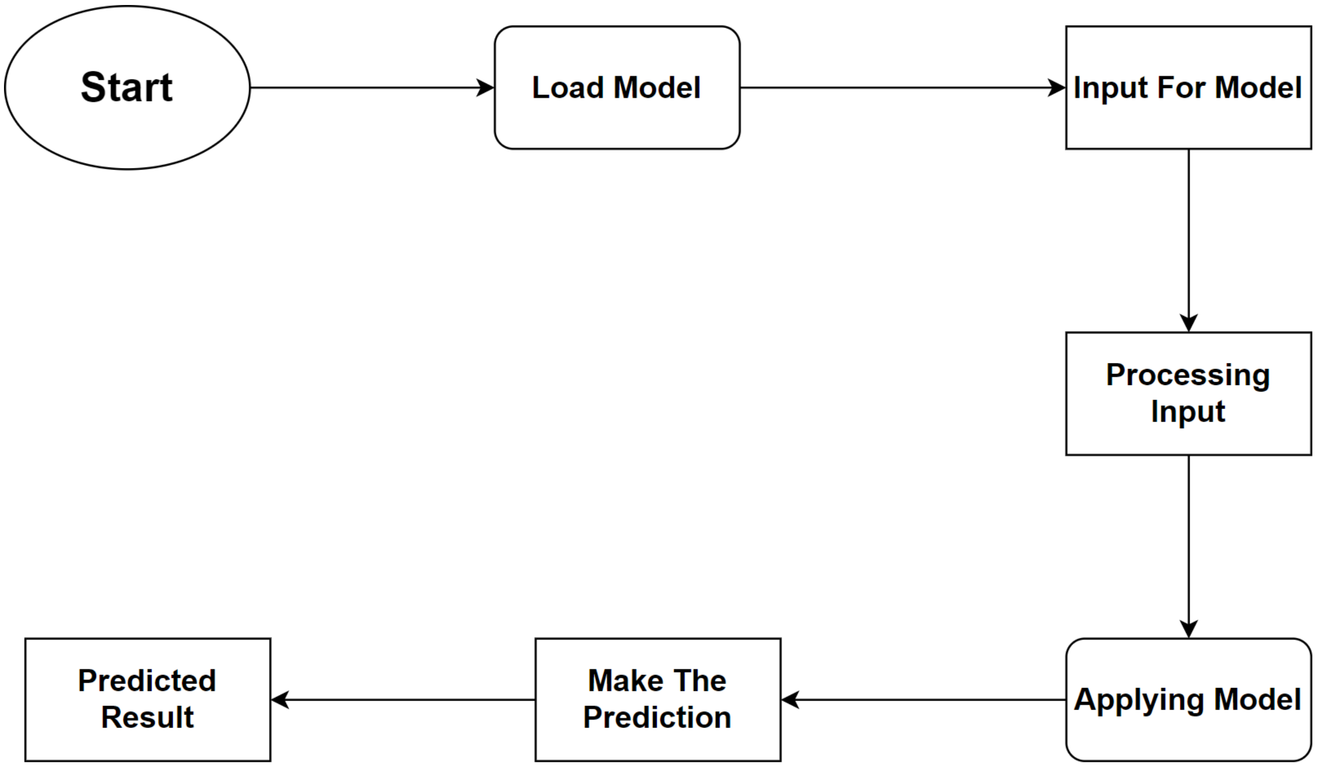
The different components for this project will be using Python as an interface between them. Each component will have its own task to perform, and it is the job of the Python to ensure proper transfer of information.

3.7 Resource Utilization

When any task is performed, it will likely use all the processing power available until that function is finished.

3.8 Deployment

Deployment Process



4 Conclusion

The Designed UGV (Unmanned Ground Vehicle) will detect an anomaly in the locality based on various anomalies data used to train our algorithm, so we can identify the imbalance in the society in early stages and can take necessary action to stop them immediately, so we can have a pleasant environment in that area or location.

5 References

1. https://www.kaggle.com/behrad3d/nasa-cmaps
2. https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/