UOW MALAYSIA KDU PENANG UNIVERSITY COLLEGE

BACHELOR OF COMPUTER SCIENCE (HONS)

PROJECT TITLE – [Predictive Maintenance Dashboard]

LIM KHAI SIAN (0201576)

SEPTEMBER 2021

UOW MALAYSIA KDU PENANG UNIVERSITY COLLEGE

BACHELOR OF COMPUTER SCIENCE (HONS)

PROJECT TITLE – [Predictive Maintenance Dashboard]

Student name: Lim Khai Sian
Student ID: 0201576
Supervisor Name: Associate Prof Dr J. Joshua Thomas

A dissertation submitted in partial fulfilment of the regulations governing the award of the degree of Bachelor of Computer Science (HONS)

at University of Lincoln and UOW Malaysia KDU Penang University College

Application Project

SEPTEMBER 2021

DECLARATIONS

I declare the following:

(1) that the material contained in this dissertation is the end result of my own work and that due acknowledgement has been given in the bibliography and references to **ALL** sources be they printed, electronic or personal.

(2) the Word Count of this Dissertation is 14470

(3) that unless this dissertation has been confirmed as confidential, I agree to an entire electronic copy or sections of the dissertation to being placed on the OpenLearning Portal, if deemed appropriate, to allow future students the opportunity to see examples of past dissertations. I understand that if displayed on eLearning Portal it would be made available for no longer than five years and that students would be able to print off copies or download.

(4) I agree to my dissertation being submitted to a plagiarism detection service, where it will be stored in a database and compared against work submitted from this or any other School or from other institutions using the service.

In the event of the service detecting a high degree of similarity between content within the service this will be reported back to my supervisor and second marker, who may decide to undertake further investigation that may ultimately lead to disciplinary actions, should instances of plagiarism be detected.

SIGNED:

Lim Khai Sian

Date: 06/12/2021

ACKNOWLEDGEMENT

Throughout completing this project, I had received a lot of help and encouragement.

Foremost, I would like to express my sincere gratitude to my supervisor, Dr. J. Joshua Thomas who guided me throughout this project from the very beginning until the completion. I thank him for the useful guidance, supports and recommendation provided during the development of this project. He also provided constructive comments on the report to further enhance the quality of the report.

Next, I would like to thank my second marker, Dr. Ng Fong Chiu for reviewing my project proposal and providing some constructive advice to improve the proposal. Besides, I would also like to Ms. Nursakirah for the report writing support provided. She had provided various guidance on report writing and answered my questions.

Furthermore, I would like to thank all the faculty members of the Department of Computing. Without their support and guidance, this project would not progress smoothly especially during a hard time of the Covid 19 pandemic.

Finally, my sincere thanks also go to my parent. Their encouragement during the hard time of this project is much appreciated. I am very much thankful to my family for their understanding and continuing support to complete this project.

ABSTRACT

Maintenance is a very important activity in any industries especially the secondary industry which is involved in construction and manufacturing. It can impact the operation time and efficiency of the equipment. Therefore, equipment defects must be identified and fixed to avoid production downtime. However, traditional maintenance strategies are inefficient and costly. With the advancement in digital technologies, the storage of historical data from the embedded sensor has become easy. With the availability of a large amount of data, Machine Learning (ML) approaches have emerged as a promising tool in maintenance management. Therefore, the aim of this project is to develop a Predictive Maintenance (PdM) strategy with a machine learning approach.

This project adopted the Data-driven approach of the PdM which process and interpret the historical data to train a predictive model. This is a machine learning project which aims to develop a predictive model to predict the Remaining Useful Life (RUL) of the equipment. It is a regression problem to approximate a mapping function from input data to continuous output. There are two products developed in this project which are the predictive model and dashboard. The predictive model is built using Gradient Boosting Regression (GBR) algorithm with Principle Component Analysis (PCA) as the feature extraction method. The dashboard developed is used to display the condition of the equipment and the predicted RUL. There are also additional functions such as lock screen and speech recognition. The aim of the project is to develop a PdM system to replace the traditional maintenance strategies thus improving the productivity of equipment and reducing maintenance costs.

In this project, various well-known machine learning algorithms are implemented and compared their result against the main algorithm. The main algorithm is an ensemble method where based learners are created independently and combine their results via an averaging process. The predictive model trained is then implemented in the dashboard to predict the RUL of different turbofan engines. The sensor measurements of the engine are displayed on the dashboard with the predicted RUL to provide the complete condition of the engine.

TABLE OF CONTENTS

LIST OF TABLE

LIST OF FIGURE

LIST OF CODE SNIPPET

CHAPTER 1
INTRODUCTION

1.1 Introduction

The global competition and the demand for quick adaptation to the rapid-changing market demand are driving industrial production today. In order to fulfil such requests, the industries are undergoing “The Fourth Industrial Revolution” also named Industry 4.0. This is closely related to the integration of physical systems and digital technologies. The integration between these sectors has allows for the collection of a huge amount of data from various production equipment (Carvalho et al., 2019). Besides, industries are also adopting new approaches to further improve the efficiency of production. Effective and flexible maintenance management has become increasingly important serving as the primary means to achieve this goal (Mathew, Luo and Pang, 2017). Thanks to Industry 4.0, data analytics and machine learning techniques can be applied to collected data to support strategic decision making in maintenance management.

Data visualization is an essential segment that supports the presentation of information after processing and interpretation (Noonpakdee, Khunkornsiri, Phothichai and Danaisawat, 2018). Dashboards have become a common method to display vital data at a glance to support better communication and decision making. It is the most efficient method to trace various data. The speed of information availability and reliability are critical in effective decision-making. As such, almost every company require the assistance of an Information and technology system (ICT) to acquire data rapidly and accurately (Subrahmanyam, Ketha, Balakrishna and Kumar, 2018).

1.2 The aims and objectives of the project

The aims and objectives had been refined due to the Non-Disclosure Agreement (NDA) issue which will be discussed later in Chapter 7.

There are two aims in this project which are to investigate on suitable Machine Learning model to integrate into the dashboard for Data Analytics and to develop dashboard and implements Machine Learning methodologies for Data Analytics

Based on the research conducted on various machine learning use cases. Predictive Maintenance (PdM) has been selected as the main topic. Among different methods in PdM, the Data-driven approach is selected in this project. Different maintenance strategies and PdM approaches will be discussed in Chapter 2. The second aim was achieved through the development of a predictive machine learning model and dashboard.

Below are the objectives to be achieved in this project.

1. To write a Projection Initiation Document (PID)
2. To write a proposal on improving dashboard through Machine Learning techniques
3. To construct a Gantt Chart for time management
4. To install and explore the required software to improve the dashboard
5. To learn a suitable programming language such as C# for improving the system
6. To investigate Machine Learning Use Cases in Manufacturing (Predictive Maintenance, Digital Twin and quality optimization)
7. To acquire dataset for the Use Case (sensor reading of Equipment)

8. To research on various Machine Learning models for the Use Case (Regression, KNN and Random Forest)
9. To train the selected Machine Learning Model to predict the Remaining Useful Life of Equipment
10. To evaluate the accuracy of trained Machine Learning Model
11. To perform Hyperparameter optimization to generate a better result
12. To display the result of the Machine Learning Model on the dashboard
13. To design the User Interface of Dashboard
14. To implements Speech Recognition function in the Dashboard
15. To complete the final document with all the stated chapters

Objectives 1-8 has been completed in CPP3012 IP2. The remaining objectives (Objectives 9-15) were completed in this subject CCP3026 IP2.

1.3 Product Overview

There are two products to be developed in this project. The first product is a predictive model. It is a machine learning model used to predict the Remaining Useful Life (RUL) of equipment. Since the dataset used in this project is the sensor measurements of the turbofan engine thus, the model is used to predict the RUL of the turbofan engine. The dataset is preprocessed using the Principle Component Analysis (PCA) to extract useful information and Gradient Boosting Regression (GBR) algorithm to develop the model.

The second product is a dashboard to display the result of the predictive model. The dashboard can execute a python script to predict the RUL of the user-selected engine. There are three main functionalities in this dashboard. The first functionality is a lock screen which is used to hide the engine data and prevent others from peeking at the information. The second functionality is to display the sensor values and RUL of the engines. Users can select the engine to be displayed by changing the drop-down menu. When the engine is changed, the system will retrieve data regarding the engine and predict the RUL. The data is refreshed every 5 minutes to ensure that the latest data is always displayed. The third function is to allow users to change between different predictive models. Users can select different predictive models by changing in the drop-down menu. The performance of the model is also displayed on the dashboard. Users can change between functions by clicking the buttons in the side menu

There is additional functionality to improve the usability of the dashboard which is speech recognition. Users can control the dashboard through different commands. For example, the command "Lock Screen" is used to hide the engine data.

1.4 Discussion on problem Statement

Nowadays, different machines are used for production, these machines and their associated components may deteriorate and lead to failure. In order to prevent production downtime due to machine failure, maintenance management has become very crucial. The two major maintenance strategies adopted by industries are Run-To-Failure (R2F) and Preventive (PvM) maintenance (Susto et al., 2015). R2F maintenance takes place when equipment failure occurs while PvM maintenance takes place periodically according to a schedule. These approaches are inefficient as they may result in production line failure or costly maintenance. In order to improve production efficiency and reduce production costs, Predictive Maintenance (PdM) is introduced.

Among different methods in PdM, the Data-driven approach is selected in this project. Many Data-driven approaches such as Neural Networks, Random Forest and Support Vector Regression have been introduced. Therefore, this project utilizes a different machine learning algorithm named Gradient Boosting Regression to build the predictive model.

1.5 Motivation of project

Nowadays, machine learning is receiving increasing attention and it has been applied in different areas to solve problems. The first motivation for this project is to learn more about different machine learning algorithms. In this computer science program, I have learned various machine learning algorithms such as linear regression and decision trees. After learning how these algorithms can solve problems, I have the interest to learn more about different algorithms and how they work. This project allows me to explore more on machine learning and understand different algorithms. The second motivation of this project is to learn how machine learning algorithms can be used to solve real-world problems. Although I had learned different algorithms, I do not have experience with how these algorithms can be used to solve real-world problems. This project allows me to investigate different machine learning use cases. Since PdM is one of the most popular use cases, I had decided to investigate how different machine learning algorithms can be used in this use case.

1.6 Areas of improvement from CCP3012 IP1

Based on the Project Proposal Review Form returned by second marker Dr Ng Fong Chiu, various changes have been made to improve the project proposal. The first change is the project objectives. Based on the second marker’s feedback, various research areas are specified in the objectives. Machine learning use cases are a wide research area thus predictive maintenance, digital twins and quality optimization are specified in the project objectives to narrow down the research scope. Besides, the development of speech recognition is also stated as one of the project objectives because it is a functionality of the dashboard.

The key concepts section of the project proposal is also modified based on the second marker's comments. The data analytics concept is explained in this section because predictive analytics which is part of data analytics is tightly coupled with machine learning. Machine learning techniques are included in predictive analytics to interpret data efficiently. Furthermore, the concept of speech recognition is also added as it is a function of the dashboard which allows users to control the dashboard.

After receiving some feedback from the second marker, the proposed work section is also modified to consider more aspects of the proposal. The users of the product are discussed to identify the potential users who can make use of the product. In this project, the potential users are targeted as the production team and engineers who are concerned with the condition of the equipment. Furthermore, each function of the dashboard is also discussed in more detail to provide more information about the dashboard.

1.7 Summary of tools used and approach taken

In this project, Rapid Application Development (RAD) is adopted to develop the product. Each function of the dashboard will have an isolated RAD process. Each process can progress simultaneously to ensure the proposed work can be completed within the time frame. Each stage of the RAD process will be reviewed and refined before moving to the next stage. Finally, the satisfied prototype of each function will be combined to produce the final dashboard.

Python is the main tool used to develop the predictive model. It is used in Jupyter Notebook which allows users to view the intermediate result. Python is used due to its extensive support for machine learning modules. There are several Python libraries required during the development process. The first library is pandas. It is used to hold the data as a data frame thus allowing data analysis and manipulation. The second library is scikit-learn. It is a python machine learning library. It provides various methods which allow the development to be more efficient. All the functions such as standardization, feature extraction and machine learning algorithms are imported from scikit-learn library. The third library is NumPy which is used to perform some mathematical operations. The fourth library is matplotlib. It is used to present the performance of the model visually. It allows the information to be more interpretable. **Table 1** shows all the Python libraries used.

Python Library	Version
numpy	1.21.2
matplotlib	3.3.4
pandas	1.2.4
seaborn	0.11.1
sklearn (scikit-learn)	0.24.1
pickleshare	0.7.5

Table 1: Python libraries

Visual Studio is the tool used to develop the dashboard. It is used because it supports a wide range of extensions and provides various useful features. Features such as code suggestion and syntax checking had made the development process easier and efficient. The drag and drop function had made the user interface design more intuitive. Besides, it allows all the files to be organized systematically.

1.8 Summary

Maintenance management has become increasingly important. Predictive Maintenance is the most efficient method among the maintenance strategies. Therefore, this project developed a predictive model to predict the Remaining Useful Life of equipment and a dashboard to display the condition of the equipment and the predicted outcome.

CHAPTER 2
ANALYSIS/LITERATURE REVIEW

2.1 Overview

This chapter discusses the main concept of predictive maintenance. Besides, common Machine Learning algorithms used in predictive maintenance and the related papers are discussed. It also includes two main techniques used in this project which are Gradient Boosting Regression and Principle Component Analysis.

2.2 Predictive Maintenance

Maintenance management can be classified into three categories:

- Run-to-Failure (R2F)
- Preventive Maintenance (PvM)
- Predictive Maintenance (PdM)

R2F maintenance only takes place when an equipment failure occurs (Carvalho et al., 2019). This is the simplest maintenance approach, but it is also the least efficient approach. The cost of downtime after failure is higher than the cost of maintenance conducted in advance.

PvM maintenance takes place periodically according to a planned schedule with time or process iterations (Susto et al., 2015). Although this approach can prevent failure, unnecessary corrective actions are often performed causing waste of resources and increased operating costs.

PdM maintenance takes place when corrective actions are needed. It is based on continuous monitoring of equipment, allowing maintenance to be performed timely (Carvalho et al., 2019). Besides, a predictive tool based on historical data is used to enable the early detection of failure.

Since a good maintenance approach should minimize equipment failure and operating cost, while maximizing the life of equipment, Predictive Maintenance is selected in this project.

Predictive maintenance can be further distinguished into three different categories:

- Model-based approach
- Data-driven approach
- Hybrid approach

Model-based approach models the entire equipment describing the degradation process (Khelif et al., 2017). Although this approach can achieve higher accuracy, it requires complex mechanistic knowledge and a psychical understanding of the equipment (Li, Ding and Sun, 2018). Besides, the developed model is difficult to generalize to other equipment.

Data-driven approach processes and analyze historical data to train a predictive model. It relies on statistical or machine learning methods to reveal the behaviour of the equipment (Khelif et al., 2017). The model is then used with real-time data to estimate the state of health of the equipment.

The hybrid approach combines Model-based and Data-driven approaches to promote the advantages and avoid the disadvantages of both approaches. The model is derived from the physical understanding of the equipment and the parameters are learned and updated using the data-driven approach (Khelif et al., 2017).

This project focus on a machine learning-based data-driven approach as the model-based and hybrid approach requires expert knowledge of the equipment.

2.3 Machine Learning

Machine learning has emerged as a prominent method for constructing an effective predictive model in various applications due to the ability to interpret high dimensional and multivariate data and extract the hidden correlations within data. However, the performance of machine learning is depending on the suitable choice of ML algorithms. This section discusses the main machine learning algorithms used in the PdM field.

2.3.1 Artificial Neural Network

Artificial Neural Network (ANN) is one of the most common ML algorithms due to its excellent performance in many industrial applications. It is a computational network inspired by biological neurons. An ANN is composed of nodes or neurons in a layered structure that resembles a human brain (IBM, 2020). These neurons are connected and each connection has an associated weight. The weights are adjusted automatically during the training process until the network is able to produce the desired result.

In [18], a Multilayer Perceptron (MLP) is proposed to predict the RUL of aero-engines. The authors proposed a moving time window to consider the values in prior cycles. In this case, the input to the network is a time window, thus the dimension of the input increases as a multiple of the window size. (Lim, Goh and Tan, 2016)

In order to achieve the same effect as the time-window based method, [5] proposed a Recurrent Neural Network (RNN) based on an encoder-decoder framework to predict the Health Index (HI) of the rolling bearing. The RUL of the component is then retrieved from a linear regression model as the HI is constructed closely related to the RUL. However, the result cannot be compared as both studies use different datasets. (Chen, Peng, Zhu and Li, 2020)

2.3.2 Random Forest

Random Forest (RF) is developed by Leo Breiman in 2001 [1]. RF is a supervised machine learning algorithm that can be used for classification and regression problems. The name Random Forest came from the idea that this method creates a group (forest) of randomized Decision Trees (Carvalho et al., 2019). It is an ensemble method which is a machine learning technique that incorporates various base models to produce an improved model. Although RF is a group of decision trees, unlike conventional decision trees, RF creates trees randomly thus, it can avoid overfitting in most cases (Breiman, 2001).

In [3], RF is used to create a predictive model to predict the failure of wind turbines. This work proposes an improvement to the artificial immune network algorithm adopted in [16] in terms of speed, scalability and accuracy. The author is able to achieve 5% - 9% of accuracy improvement compared to the original solution. (Canizo et al., 2017) (Kusiak and Zhang, 2011)

In [29], RF is used to develop a predictive maintenance system named HDPass to detect the Hard Disk Drive (HDD) failure in data centres. The authors train the model on historical data and utilize data collected from end-users to perform real-time predictions. (Su and Huang, 2018)

Other than the papers mentioned above, many authors also selected the RF model to compare against their proposed methods.

2.3.3 Support Vector Machine

Support Vector Machine (SVM) is another machine learning algorithm widely used in PdM. It is a supervised algorithm. Although SVM is mostly used in classification problems, it can also be useful in regression problems. The goal of SVM is to find a hyperplane in N-dimensional space that can best distinguish data points into different classes (Gandhi, 2018). The N-dimension is depending on the number of features.

In [24], a Support Vector Classifier is used. Vibration signal is used as input to the SVM model for fault identification in a gearbox. In this work, the data is acquired from 4 gearboxes with different motor speeds and loads, and 2 gear fault conditions. The data is fed to SVM after feature extraction. In this approach, the author is able to achieve accuracy greater than 90% in 4 different datasets. (Praveenkumar, Saimurugan, Krishnakumar and Ramachandran, 2014)

On the other hand, Support Vector Regression (SVR) is used in [21]. Unlike conventional SVR, a modified regression kernel that can be used for SVR is proposed for prognostic problems. In this work, the authors modify the kernel to account for different cycles in the prognostic domain and integrate the time dependencies. (Mathew, Luo and Pang, 2017)

The papers above prove that SVM can be used for classification and regression problems in PdM.

2.4 Gradient Boosting Regression

The works in the Random Forest section had proven that ensemble methods can be used in the data-driven approach of PdM. In this project, another ensemble method named Gradient Boosting Regression (GBR) is used to train the predictive model.

Unlike conventional machine learning algorithms such as Linear Regression and Support Vector Machine, approaches such as Bootstrap Aggregation (Bagging) and Random Forest are built on the idea of creating base learners independently and combining their results via certain deterministic averaging processes (Singh, 2018). However, boosting method is based on different strategies to combine the weak learners.

The concept of boosting arose from the idea of whether a weak learner can be improved. It is articulated by Michael Kearns as the "Hypothesis Boosting Problem" (Kearns, 1988). Hypothesis Boosting was the concept of filtering observations, keeping those that can be addressed by weak learners and focusing on building new weak learners that can address the remaining observations (Brownlee, 2020).

Gradient Boosting algorithm can be best described by first introducing the Adaptive Boosting also known as AdaBoost. The weak learners in AdaBoost are decision stumps which are decision trees with one split. AdaBoost works by increasing the weight of difficult observations and lowering the weight of easier observations (Singh, 2018). New weak learners are then added sequentially to solve more difficult patterns. The final result is voted by the majority prediction of weak learners weighted by their associated accuracy.

Gradient Boosting is a generalization of AdaBoost which is an optimization problem in which the goal is to minimize the loss function by adding new weak learners via a gradient descent like process (Brownlee, 2020). This generalization allows differentiable loss functions to be used thus expanding the applicability of the algorithm to regression and multi-class classification problems.

There are 3 important elements in Gradient Boosting:

1. Loss Function
2. Weak Learners
3. Additive Model

Loss Function

Loss Function is a measurement of how well the model's coefficients are at fitting the underlying data (Singh, 2018). Different loss functions can be used depending on the problem being solved as long as the functions are differentiable.

Weak Learner

In Gradient Boosting, decision trees are used as weak learners. More specifically regression trees are used as they can produce real values for splits and the outputs can be added, allowing future models output to be added and “correct” the residuals (Brownlee, 2020).

Additive Model

Trees are added one at a time, and existing trees remain unchanged. A gradient descent procedure is used to minimize the loss function when adding trees. Unlike conventional Gradient Descent which is used to update a set of parameters such as weights in a neural network, it is used to modify the parameterized trees and add them into the model to reduce the loss function (Brownlee, 2020). This type of Gradient Descent is known as Functional Gradient Descent.

2.4.1 Algorithm

The first step of this algorithm is to create a base model to predict the data. The predicted value of the base model is the average of the dependent variables (Gupta, 2021). The idea of the average of the dependent variables come from **formula 1**.

$$F_0(x) = \underset{\gamma}{\arg min} \sum_{i=1}^{n} L(y_i, \gamma)$$

Formula 1: Initialise base model

Here, L is the loss function and γ is the predicted value. The goal of this formula is to find a γ value where the loss function is minimum.

In this project, the dependent variable is a continuous value, thus the loss function is:

$$L = \frac{1}{n}\sum_{i=0}^{n}(y_i - \gamma_i)^2$$

Formula 2: Loss function

Here, y_i is the dependent variable and γ_i is the predicted value. In order to find the minimum value of γ such that the loss function is minimum, the derivative of the loss function is required as shown in formula 3.

$$\frac{dL}{d\gamma} = \frac{2}{2}(\sum_{i=0}^{n}(y_i - \gamma_i))(-1)$$

Formula 3: Derivative of the loss function

The next step is to calculate the pseudo residuals. The pseudo residual is equal to the $\text{dependent variable} - \text{predicted value}$. This can be proven by formula 4.

$$\gamma_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}, for\ i = 1, \dots, n$$

Formula 4: Pseudo residual

Here, $F(x_i)$ is the previous model trained and the m is the number of decision trees made. The derivative of the loss function is calculated:

$$\frac{dL}{d\gamma} = -(y_i - \gamma_i) = -(\text{dependent variable} - \text{predicted value})$$

Formula 5: Derivative of the loss function

After the residual is calculated, a decision tree is created by using the residual and the independent variables (Gupta, 2021).

The third step is to apply **formula 6**.

$$\gamma_m = \underset{\gamma}{\arg min}\sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

Formula 6: Step 3 formula

Here, $h_m(x_i)$ is the decision tree made based on the residual and the m is the number of decision trees made. For example, $m = 1$ refers to the first decision tree.

Formula 6 is similar to **formula 1** but here the previous prediction is taken instead of the average of the dependent variable because there is no previous prediction in step 1 (Gupta, 2021).

The last step is to update the predictive model by using formula 7.

$$F_m(x) = F_{m-1}(X) + \alpha(h_m(x))$$

Formula 7: Update predictive model

Here, the $F_{m-1}(x)$ is the prediction of the previous model, α is the learning rate and $h_m(x)$ is the recent decision tree made based on the residuals.

2.5 Principle Component Analysis (PCA)

In order to speed up the training process of the predictive model, a feature extraction method named Principle Component Analysis (PCA) is used. It is a dimensionality reduction method that aims to reduce the dimension of a large dataset while retaining most of the information in the large set (Jaadi, 2021). Naturally, reducing the number of features in the data set will impact the accuracy. However, the trick to dimensionality reduction is to trade minimum accuracy for efficiency. Since the main focus of the project is not PCA, the complex mathematical calculation is not being discussed.

PCA can be broken into 5 different steps which are:

1. Standardization
2. Covariance matrix
3. Compute the eigenvectors and eigenvalues of the covariance matrix to identify principle component
4. Feature vector
5. Recast data along principle components axes

The first step is to standardize the range of feature values so that each feature contributes equally to the analysis. PCA is sensitive to the variance of the feature values (Jaadi, 2021). If there is a large difference between features' values, those features with larger values tend to dominate those with smaller values which can lead to biased results.

The second step is to compute the covariance matrix. The aim of this step is to identify any relationship between features. In certain cases, features can be highly correlated in such a way that they hold duplicated information. Thus covariance matrix is computed to identify such relationship.

The third step is to compute the eigenvectors and eigenvalues to identify the principle components. They are the linear algebra concepts that need to be computed from the covariance matrix to identify the principle components of the data (Jaadi, 2021). It is best to introduce principle component before getting into eigenvectors and eigenvalue.

Principle components are new features that are created as linear combinations of the original features (Jaadi, 2021). These new features are uncorrelated and the majority of the information in the original features are compressed into the first component. By doing so, it can reduce the dimensionality of the data without sacrificing much information. However, principle components are less interpretable and do not contain any actual meaning.

Eigenvectors and eigenvalues are in pairs thus; each eigenvector has an eigenvalue. The number of pairs is determined by the number of dimensions of the data set. The direction of the axes which contains the most information, which we named principle components are the eigenvectors of the covariance matrix (Jaadi, 2021). Eigenvalues are simply coefficients associated with eigenvectors which represent the amount of variance held in each eigenvector.

The fourth step is to form a feature vector. After computing the eigenvectors and arranging them in descending order based on the corresponding eigenvalue, principle components can be identified in order of significance. The selected components will form a matrix which is also called a feature vector.

The last step is to use the feature vector to reorient the original data to the ones represented by the principle components. This is done by multiplying the transpose of original data with the transpose of the feature vector.

2.6 Speech Recognition

Speech Recognition is an additional function that allows users to control the dashboard through commands. Speech Recognition is different from Voice Recognition. It is responsible for converting verbal speech into text form whereas voice recognition is used to identify an individual's voice.

Speech Recognition is considered one of the most challenging areas of computer science. Speech input, feature extraction, feature vectors, decoder and word output are the components of speech recogniser (IBM, 2020). The decoder uses the acoustic models, pronunciation dictionary and language models to identify the proper output of the speech (IBM, 2020).

2.7 Summary

In this chapter, various commonly used machine learning algorithms and related works have been discussed. However, this project uses another algorithm which is Gradient Boosting Regression. This chapter also discussed the main concept of Gradient Boosting Regression and the Principle Component Analysis feature extraction method.

CHAPTER 3
SYNTHESIS/REQUIREMENTS SPECIFICATION

3.1 Overview

System Requirements Specification (SRS) is a document that describes the features and behaviours of a system. This section will discuss the main functionalities required to satisfy the goal of the project and additional requirements that complete the entire system. Besides, this section also includes various requirements for the system to work as intended and justification on the algorithms used to build the machine learning model.

3.2 System Requirements Specification

3.2.1 Functional Requirements

Functional requirements define the features that the system must provide (Martin, 2021). These features allow the system to work as intended and fulfil user requirements. A functional requirement can be a calculation, user interaction process or any behaviours that the system must perform.

The first functional requirement in this system is to display the latest values from various sensors measuring the condition of the turbofan engine. These sensor values include 3 operation settings and 21 anonymous sensors. This feature allows users to observe the current condition of each engine. Since there is more than 1 engine being observed, the dashboard also allows users to switch between engines through a drop-down menu to observe their condition.

The second functional requirement is to predict the RUL of each engine. The main purpose of this project is to predict the RUL of different turbofan engines using a machine learning algorithm. The RUL is predicted in python by using the trained model. The result is then displayed on the dashboard together with the current sensor values of a specific engine. The predicted RUL is changed when users change the engine in the drop-down menu.

The third functional requirement is to update the dashboard to display the latest sensor values. Since the sensors will update the engine's condition at a regular time interval, the latest condition needs to be displayed on the dashboard. Therefore, the dashboard will read the latest values from the database at a regular time interval. The predicted RUL is changed when the latest record is updated.

3.2.2 Non-functional Requirements

While Functional Requirements define the features that the system must have, Non-functional Requirements define the features that the system should have. These features do not impact the basic functions of the system. They describe the capabilities and constraints that enhance the system (Martin, 2021). These requirements can be security, usability and reliability.

There are three non-functional requirements in this system. The first requirement is scalability. Currently, the dashboard only includes 100 engines as the testing dataset only have 100 engines. However, the number of engines can continue to increase and they can be selected from the drop-down menu. This allows the dashboard to scale when the organization have more turbofan engines. Besides, users are allowed to change the predictive model used. This function allows comparison between different models and better models developed in the future can be used to replace easily.

The second non-functional requirement is security. Since these data can be sensitive and may not be appropriate for other people to access, the dashboard provides users with a function to hide the interface when it is not in use. This helps to prevent others from peeking at the data or taking photos of the data.

The third non-functional requirement is usability. The dashboard includes a speech recognition function that allows users to control the dashboard. With this function, users can control the dashboard directly without a mouse. Besides, the dashboard is intuitive where there are only buttons, drop-down menus and a toggle button. The buttons allow users to switch between lock screen, engine detail and model details interface while a drop-down menu allows users to change between different engines and another is used to change between different models. The toggle button is used to enable or disable the speech recognition function.

3.2.3 Software Requirements

There are several software requirements that the computer must possess in order for the system to work normally. The first software requirement is to have a Python executable file. Although the dashboard is developed in Visual Studio using C# language, the prediction process is executed in python. This constraint is inevitable as the model is built in a python environment and cannot be converted to C#. In order to use the model in the C# application, the python executable file needs to be passed to the ProcessStartInfo class to run the python script. The python version used during this project is Python 3.8.8.

The second requirement is also related to python where the computer needs to have several python libraries. These libraries are mandatory because python needs them to load the model and perform prediction. The first library is pandas. It is used to convert the data retrieved dashboard into Data Frame and for data manipulation. The second library is sys in which is used to receive arguments from the C# process. The last library is pickleshare. It is required to load the model into the environment. **Table 2** shows the version of each library used in this project.

Library	Version
pandas	1.2.4
pickleshare	0.7.5

Table 2: Python libraries

3.3 Decision of algorithms and Justification

Feature selection and extraction are important stages in building a machine learning model. After evaluating different methods, PCA is selected to reduce the dimensionality of a large set of features. This method is selected as it is able to keep the least amount of features while remaining the most information (Jaadi, 2021).

During the data preprocessing stage, manual feature selection is also conducted to select the most relevant features by analysing the data visualization graphs. However, there is still space for improvement.

Other than feature selection, model selection is also an important stage in building a machine learning model. After testing and evaluating several models to predict the RUL of the turbofan engine, the GBR algorithm is selected to build the predictive model. GBR is selected instead of other models such as Linear Regression or Random Forest Regression is because it does not have many applications in PdM. Other algorithms discussed in Chapter 2 are very common algorithms in PdM. GBR is selected because it is also an ensemble method like Random Forest which is able to introduce new weak learners to improve the performance of existing trees.

Although Random Forest Regression is also an ensemble method that combines all the outputs from each individual tree, there are several differences among them which make GBR a better option in this project. One of the differences is the way the decision tree is built. GBR builds trees one at a time and is added to the set to minimize the loss of the model (Ravanshad, 2018). However, RF build trees independently using a random sample (Ravanshad, 2018). Another difference between GBR and RFG is the way the outputs are added. GBR combines the outputs from all trees along the training process while RF combines the output at the end of the training process (Glen, 2019). More details of the training process will be discussed in chapter 5.

3.4 Summary

This chapter discussed the functional and non-functional requirements of the software. The software requirements are also stated to ensure that the program can be run successfully. Furthermore, the decision of the algorithm and technique used in the Machine Learning Life Cycle is also justified in this chapter.

CHAPTER 4
DESIGN SPECIFICATION

4.1 Analysis Model

4.1.1 Use Case Diagram

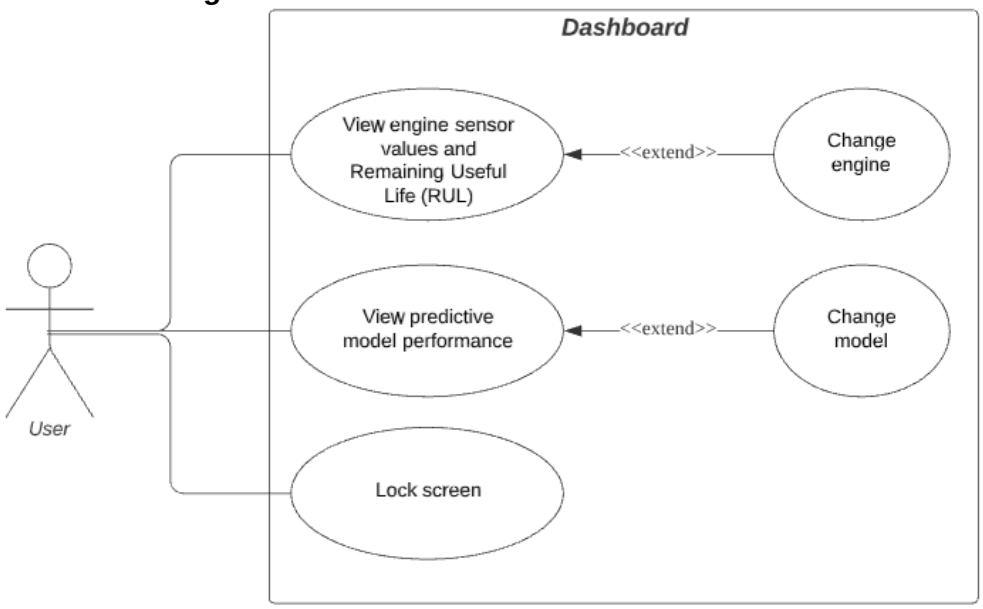


Figure 1: Use Case Diagram

Figure 1 shows the use case diagram of the dashboard. There are 3 use cases in which users can interact with the dashboard. In this dashboard, users can "View engine sensor values and Remaining Useful Life (RUL)" to comprehend the condition of the engine. "View engine sensor values" and "View Remaining Useful Life (RUL)" are combined into one because both information is displayed on the same interface. Since the system contains 100 engines, users can further extend the use case to "Change engine" to view the condition of different engines.

Besides, users can also "View predictive model performance" to understand the performance of the model used to predict the RUL. There are several models trained during this project, thus users can extend this use case to "Change model" used by the dashboard and view its performance. Furthermore, users can "lock screen" to prevent others from accessing the information.

The table below shows the use case description of the use case discussed above.

Use case name:	View engine sensor values and Remaining Useful Life (RUL)
Scenario:	1. User click the Engine side menu button 2. The dashboard displays the sensor values and RUL
Triggering event:	User click the Engine side menu button
Actors:	Users
Precondition:	The database must have related data
Post condition:	The sensor values and RUL are displayed
Exception condition:	-

Use case name:	View predictive model performance
Scenario:	1. User click the Model side menu button 2. The dashboard displays the performance of the model in use
Triggering event:	User click the Model side menu button
Actors:	Users
Precondition:	There is a model available
Post condition:	The performance matrix of the model is displayed
Exception condition:	-

Use case name:	Lock screen
Scenario:	1. User click the Lock side menu button 2. The dashboard displays a lock image
Triggering event:	User click the Lock side menu button
Actors:	Users
Precondition:	-
Post condition:	The lock image is displayed
Exception condition:	-

4.1.2 Entity Relationship Diagram

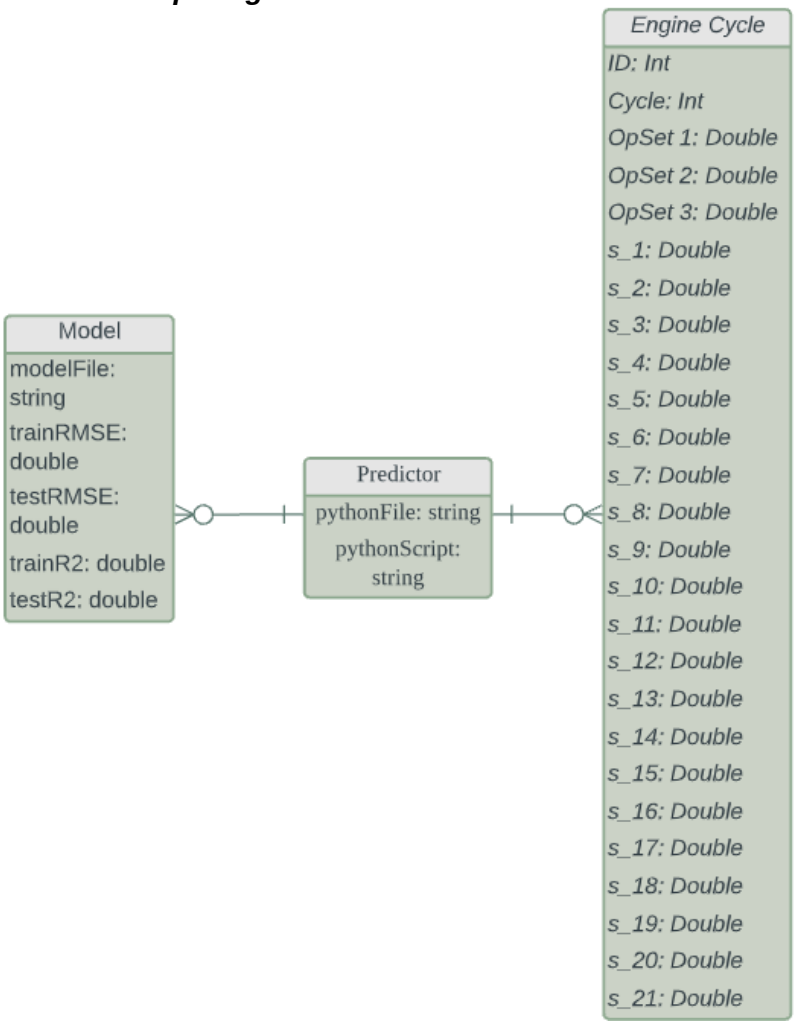


Figure 2: Entity Relationship Diagram

Figure 2 shows the Entity Relationship Diagram of the dashboard. There are three entities in this dashboard which are model, predictor and engine cycle. The model entity represents the predictive model in the dashboard which holds the file path and various performance matrices of the model. The predictor entity represents the predictor used by the dashboard to perform prediction. It holds the file path of the python script and the python executable that executes the script. The engine cycle represents the last cycle of a specific engine that is used by the predictor. This entity includes the ID of the engine, the current cycle, 3 different operation settings and 21 sensor measurements. The relationship can be described as one or more models can be used by the predictor to predict the RUL based on the different engine cycles.

4.1.3 Sequence Diagram

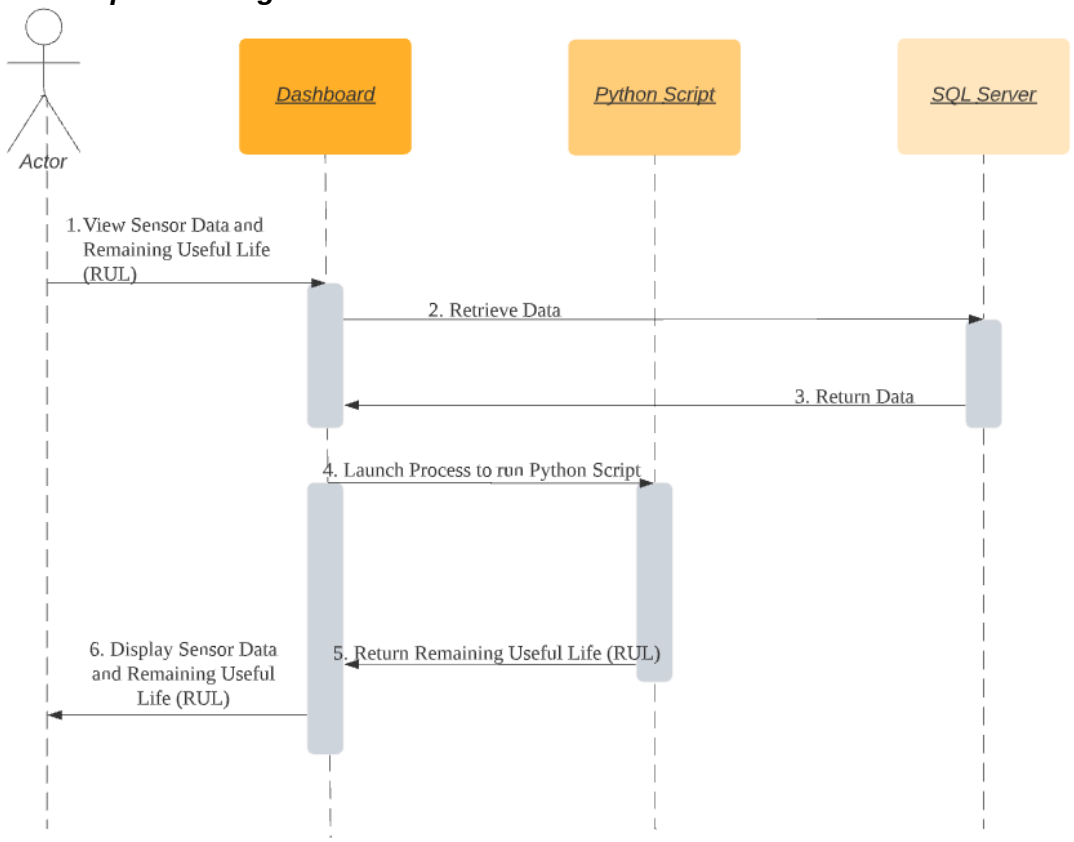


Figure 3: Sequence Diagram

Figure 3 shows the Sequence Diagram of the dashboard. When the users want to view the engine condition, the dashboard sends a request to retrieve the data from the database. Next, the dashboard launches a process to execute the python script and pass the data to the script. The data is used as input to the model after some manipulation. The result is then returned to the dashboard. When the dashboard receives the result, it displays both the sensor values and the RUL to the users.

4.2 Design Specification

4.2.1 System Architecture

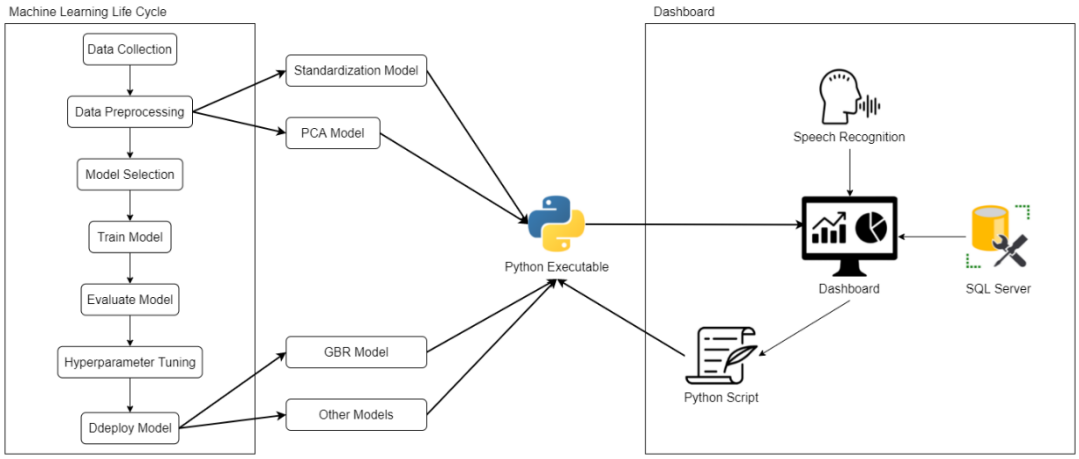


Figure 4: System Architecture

Figure 4 shows the system architecture of the entire software. The predictive model is trained using the Machine Learning Life Cycle. Throughout this process, the standardization, PCA and various predictive models are saved to be used later in the dashboard. The dashboard interacts with the SQL server to retrieve the latest cycle of a specific engine. In order to perform predictions, the dashboard creates a process to execute the python script using the python executable. The data and model file path are then passed to the script as arguments for prediction. The prediction result returned by the script is displayed on the dashboard.

4.2.2 User Interface Design

Figure 5: Side Menu

Figure 5 shows the side menu of the dashboard. This side menu is used to change between different user interfaces. The Lock button represents the lock screen interface. The Engine button represents the engine details interface and the Model button represents the model interface. The button is highlighted with colour when the associated interface is displayed on the dashboard. The logo of the dashboard also helps to reset to the model interface. Besides, the toggle button below is used to enable or disable the speech recognition function.

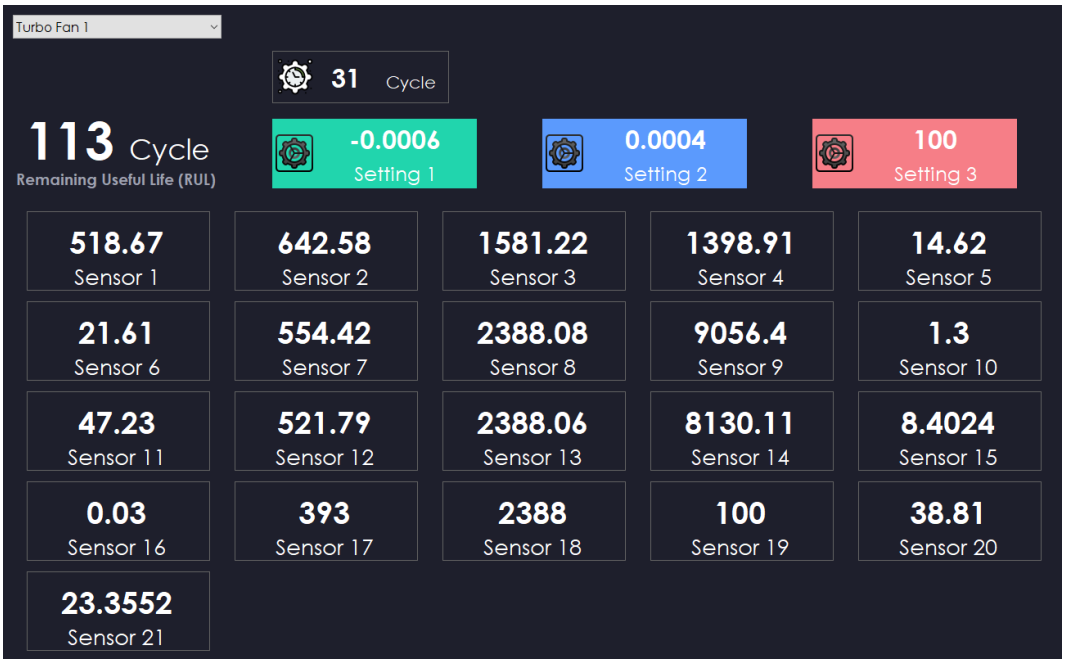


Figure 6: Turbofan engine interface

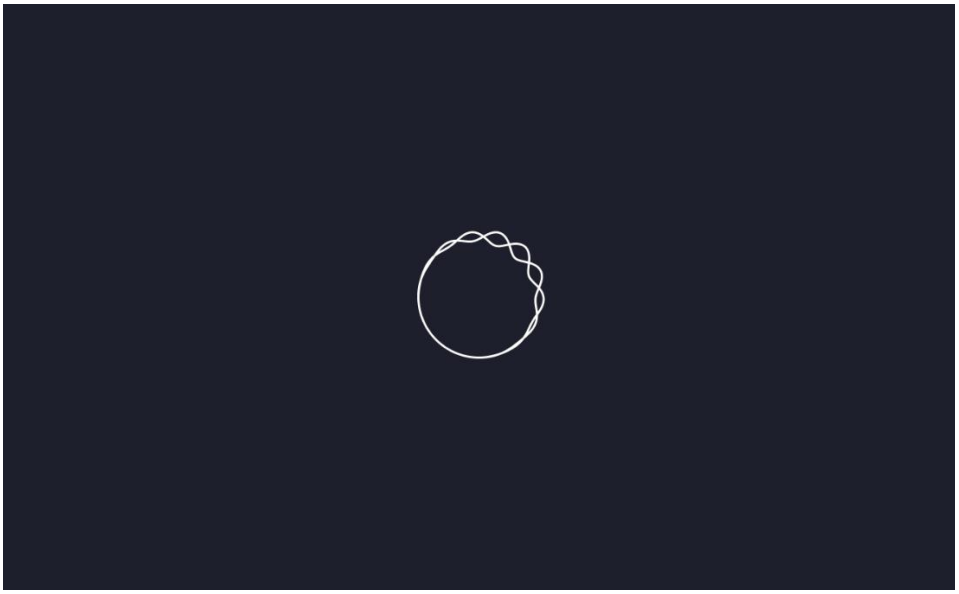


Figure 7: Loading screen

Figure 6 shows the main interface of the dashboard. This interface is shown when users click the Engine button in the side menu. In this interface, various information regarding an engine is displayed such as RUL, current cycle, 3 anonymous operation settings and 21 anonymous sensor values. The settings and sensors are anonymous as the dataset does not provide any information related to those features. This anonymity had proven that no prior knowledge regarding the equipment is required in order to build the data-driven model. In this interface, users can select different engines by changing the drop-down menu on the top left corner. **Figure 7** shows the loading screen of the dashboard. This interface is shown when the dashboard is retrieving data from the SQL server and performing predictions in Python. This process occurs when the dashboard refreshes the interface to retrieve the latest data or when the engine ID is changed by the users. This interface is shown until the dashboard completes its update operations.

Figure 8: Lock interface

Figure 8 shows the lock screen interface. This interface is shown when the users click the Lock button in the side menu. It is used to hide the engine interface when the dashboard is not in use to prevent others from peeking at the information.

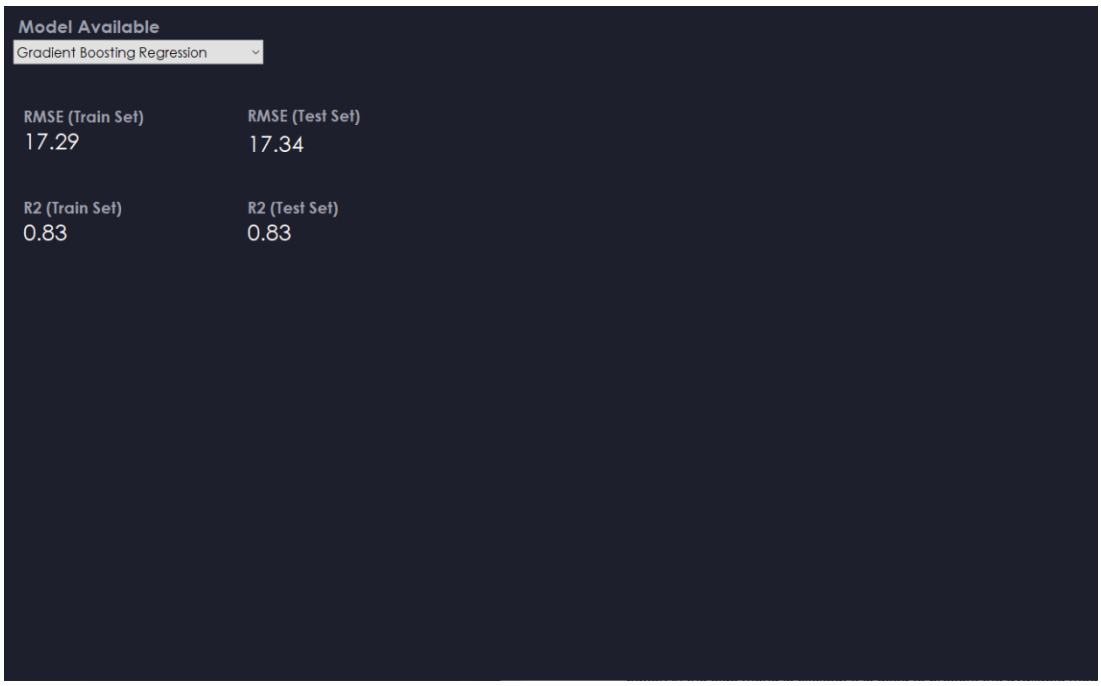


Figure 9: Model interface

Figure 9 show the model interface. This interface is displayed when the Model button in the side menu is clicked. It is used to present the available models and their associated performance. The performance metrics include Root Mean Squared Error (RMSE) and R Squared (R2) on both train and test datasets. Users can change the model by using the drop-down menu on the top left corner. Upon changing, the performance of that specific model is displayed. Besides, the model used to predict the RUL of the engine also depends on users' selection in this interface. When the model is changed, the engine interface updates the RUL of the engine based on the new model.

4.3 Summary

This chapter discussed the overall structure and design of the software by explaining various Unified Modeling Language (UML) diagrams. Besides, the architecture and user interface design of the software are also discussed in this chapter.

CHAPTER 5
IMPLEMENTATION

5.1 Overview

This chapter discusses the Exploratory Data Analysis (EDA) on NASA's turbofan engine degradation simulation dataset (C-MAPSS) to summarise their main characteristics. The EDA includes various data visualisation graphs to understand the underlying pattern of the dataset. Besides, this chapter explains the process of building the predictive model using various ML algorithms and compare their performance. This process includes data pre-processing, model selection, training model, evaluating model, hyperparameter tuning and making predictions. Finally, this chapter discusses the development of a dashboard that utilise the predictive model.

5.2 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a process to perform analysis on the dataset to discover any patterns and make an assumption based on the summary statistics and graphical representations.

Although the C-MAPSS dataset was published over a decade ago, it is still one of the most popular datasets used for PdM. C-MAPSS is a tool that simulates a realistic commercial turbofan engine. This software is written in the MATLAB and Simulink environment and it features various editable input parameters where users can enter custom values for operational profile, closed-loop controllers, and environmental conditions (Saxena, Goebel, Simon and Eklund, 2008).

The turbofan engine dataset consists of four different datasets of increasing complexity as shown in **table 3**. The engines in all datasets operate normally in the beginning and develop a fault over time. For the training set, the engines are recorded until a failure occurs, while the test set stops recording some time prior to the failure. The objective is to predict the RUL of the turbofan engines in the test set. In other words, a predictive model is required to predict the remaining number of operation cycles after the latest cycle of the engine. Due to the time constraint of the project, only the FD001 dataset is used to develop the predictive model.

Dataset	Train size (no. of engines)	Test size (no. of engines)	Operating Conditions	Fault Modes
FD001	100	100	1	HPC Degradation
FD002	260	259	6	HPC Degradation
FD003	100	100	1	HPC Degradation Fan Degradation
FD004	248	249	6	HPC Degradation Fan Degradation

Table 3: Turbofan Engine Datasets

Table 4 shows the data structure of the dataset. Each row contains the engine ID, current cycle, 3 operational settings and 21 sensor values. The operational settings and sensor values are anonymous meaning that no domain knowledge can be used. Thus, the results are solely based on applying the correct techniques.

ID	Cycle	Setting1	Setting2	Setting3	Sensor1	… …	Sensor21
Int	Int	Double	Double	Double	Double	Double	Double

Table 4: The data structure of datasets

	ID		Cycle
count	20631.000000	count	100.000000
mean	51.506568	mean	206.310000
std	29.227633	std	46.342749
min	1.000000	min	128.000000
25%	26.000000	25%	177.000000
50%	52.000000	50%	199.000000
75%	77.000000	75%	229.250000
max	100.000000	max	362.000000

Figure 10: Descriptive statistics of engine ID and cycle

Based on the descriptive statistics of engine ID shown in **figure 10**, there is a total of 20631 rows in the dataset and engine ID from 1 to 100. When observing the descriptive statistics of the engine cycle, it shows that the earliest engine failure occurs after 128 cycles whereas the longest operation cycle is 362 cycles. The average operation cycle is between 199 and 206 cycles. However, the standard deviation of 46 cycles is quite large.

	OpSet1	OpSet2	OpSet3
count	20631.000000	20631.000000	20631.0
mean	-0.000009	0.000002	100.0
std	0.002187	0.000293	0.0
min	-0.008700	-0.000600	100.0
25%	-0.001500	-0.000200	100.0
50%	0.000000	0.000000	100.0
75%	0.001500	0.000300	100.0
max	0.008700	0.000600	100.0

Figure 11: Descriptive statistics of engine settings

Figure 11 shows the descriptive statistics of the operational settings. The standard deviation of setting 3 shows that there is only one value among all the records. Although the values of setting 1 and 2 are not completely stable, the fluctuations are very small. Based on the information gathered, the description of the FD001 dataset is correct where there is only one operating condition.

	count	mean	std	min	25%	50%	75%	max
SensorMeasure1	20631.0	518.670000	6.537152e-11	518.6700	518.6700	518.6700	518.6700	518.6700
SensorMeasure2	20631.0	642.680934	5.000533e-01	641.2100	642.3250	642.6400	643.0000	644.5300
SensorMeasure3	20631.0	1590.523119	6.131150e+00	1571.0400	1586.2600	1590.1000	1594.3800	1616.9100
SensorMeasure4	20631.0	1408.933782	9.000605e+00	1382.2500	1402.3600	1408.0400	1414.5550	1441.4900
SensorMeasure5	20631.0	14.620000	3.394700e-12	14.6200	14.6200	14.6200	14.6200	14.6200
SensorMeasure6	20631.0	21.609803	1.388985e-03	21.6000	21.6100	21.6100	21.6100	21.6100
SensorMeasure7	20631.0	553.367711	8.850923e-01	549.8500	552.8100	553.4400	554.0100	556.0600
SensorMeasure8	20631.0	2388.096652	7.098548e-02	2387.9000	2388.0500	2388.0900	2388.1400	2388.5600
SensorMeasure9	20631.0	9065.242941	2.208288e+01	9021.7300	9053.1000	9060.6600	9069.4200	9244.5900
SensorMeasure10	20631.0	1.300000	4.660829e-13	1.3000	1.3000	1.3000	1.3000	1.3000
SensorMeasure11	20631.0	47.541168	2.670874e-01	46.8500	47.3500	47.5100	47.7000	48.5300
SensorMeasure12	20631.0	521.413470	7.375534e-01	518.6900	520.9600	521.4800	521.9500	523.3800
SensorMeasure13	20631.0	2388.096152	7.191892e-02	2387.8800	2388.0400	2388.0900	2388.1400	2388.5600
SensorMeasure14	20631.0	8143.752722	1.907618e+01	8099.9400	8133.2450	8140.5400	8148.3100	8293.7200
SensorMeasure15	20631.0	8.442146	3.750504e-02	8.3249	8.4149	8.4389	8.4656	8.5848
SensorMeasure16	20631.0	0.030000	1.556432e-14	0.0300	0.0300	0.0300	0.0300	0.0300
SensorMeasure17	20631.0	393.210654	1.548763e+00	388.0000	392.0000	393.0000	394.0000	400.0000
SensorMeasure18	20631.0	2388.000000	0.000000e+00	2388.0000	2388.0000	2388.0000	2388.0000	2388.0000
SensorMeasure19	20631.0	100.000000	0.000000e+00	100.0000	100.0000	100.0000	100.0000	100.0000
SensorMeasure20	20631.0	38.816271	1.807464e-01	38.1400	38.7000	38.8300	38.9500	39.4300
SensorMeasure21	20631.0	23.289705	1.082509e-01	22.8942	23.2218	23.2979	23.3668	23.6184

Figure 12: Descriptive statistics of engine sensors

Figure 12 shows the descriptive statistics of the sensor values. The standard deviation of sensors 18 and 19 is 0 meaning that these values do not fluctuate thus they can be excluded as they do not contain any useful information. Sensors 1, 5, 10, and 16 have little fluctuation thus further inspection is required. Based on the figure above, sensors 9 and 14 have the highest fluctuation.

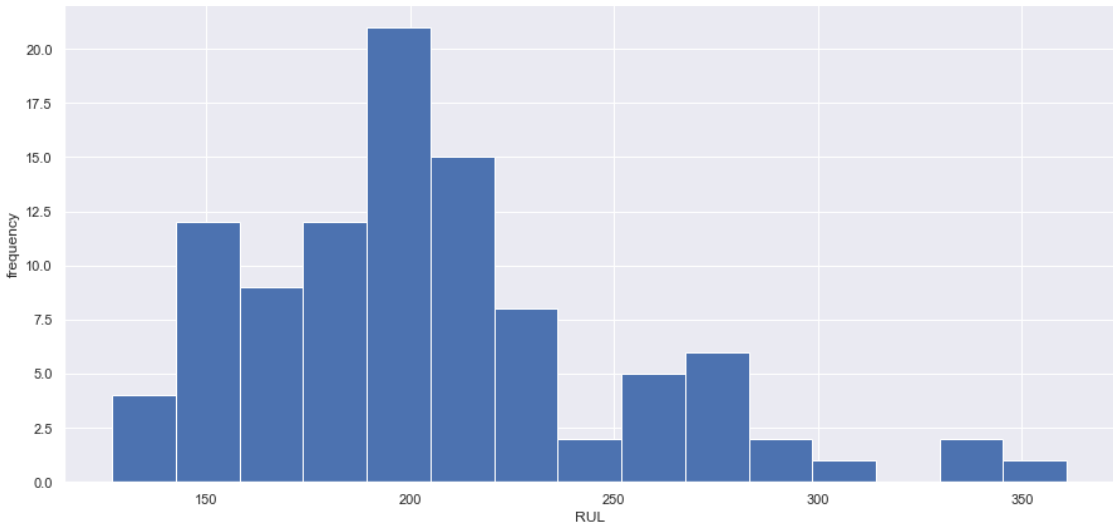


Figure 13: RUL histogram

Figure 13 shows the histogram of RUL. The calculation of RUL is discussed later in this chapter. This histogram has reconfirmed the descriptive statistics of the engine cycles above where most engine fails around 200 cycles. The distribution of this histogram is right-skewed where only a few engines can survive more than 300 cycles. Due to a large number of features, the histogram of each feature is not presented in this report. However, they can be accessed in the python script provided to understand their data distributions.

```
def plot_sensor(sensor_name):
    plt.figure(figsize=(13,5))
    for i in raw_data['ID'].unique():
        if (i % 10 == 0):  # only plot every 10th unit_nr
            plt.plot('RUL', sensor_name, data=raw_data[raw_data['ID']==i])
    plt.xlim(250, 0)  # reverse the x-axis so RUL counts down to zero
    plt.xticks(np.arange(0, 275, 25))
    plt.ylabel(sensor_name)
    plt.xlabel('Remaining Use fulLife')
    plt.show()
```

Code Snippet 1: Plot signal for each sensor

In order to better understand the trend of each sensor, the signal of each sensor is plotted. Since there are 100 engines, it is not feasible to plot every engine for every sensor otherwise the graphs would become unreadable. Therefore, only engines with ID divisible by 10 and the remainder of 0 are plotted. Due to a large number of sensors, graphs that shows a similar trend are not displayed. **Code snippet 1** shows the code used to plot the signal graphs.

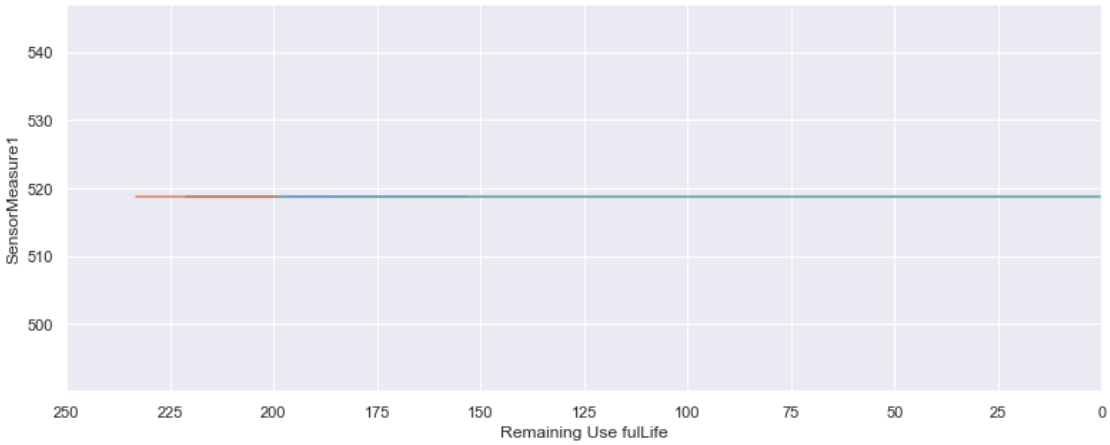


Figure 14: Graph of sensor 1

Figure 14 shows the graph of sensor 1 with declining RUL as the X-axis. The graphs of sensors 1, 5, 10, 16, 18, and 19 look similar. The flat line indicates that there are no fluctuations in the values thus these sensors do not hold any useful information.

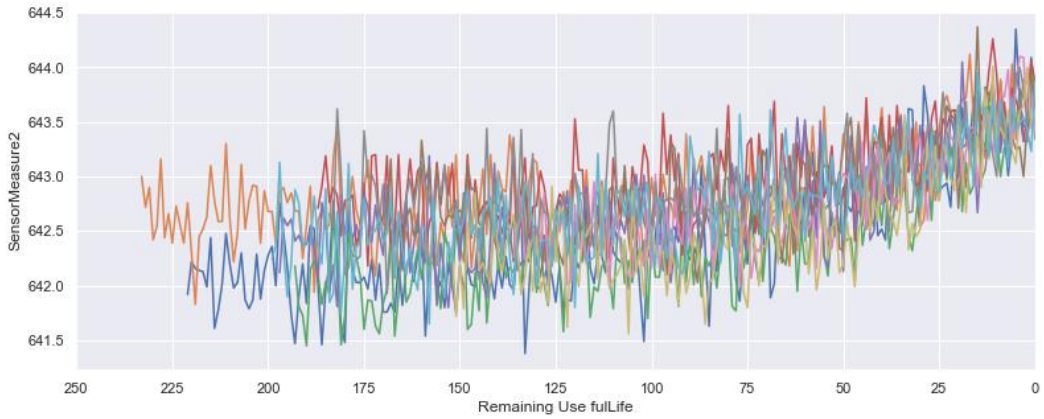


Figure 15: Graph of sensor 2

Figure 15 shows the graph of sensor 2. The value shows an increasing trend as the RUL decreases. A similar trend can be found in sensors 3, 4, 8, 11, 13, 15, and 17.

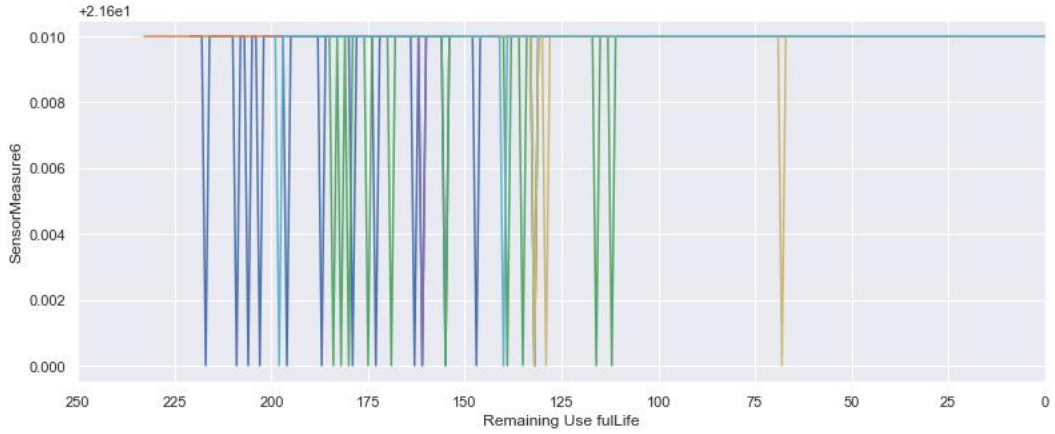


Figure 16: Graph of sensor 6

Figure 16 shows the graph of sensor 6. The value decreases to 0 at a certain point. However, this change does not have any clear relation with the decreasing RUL.

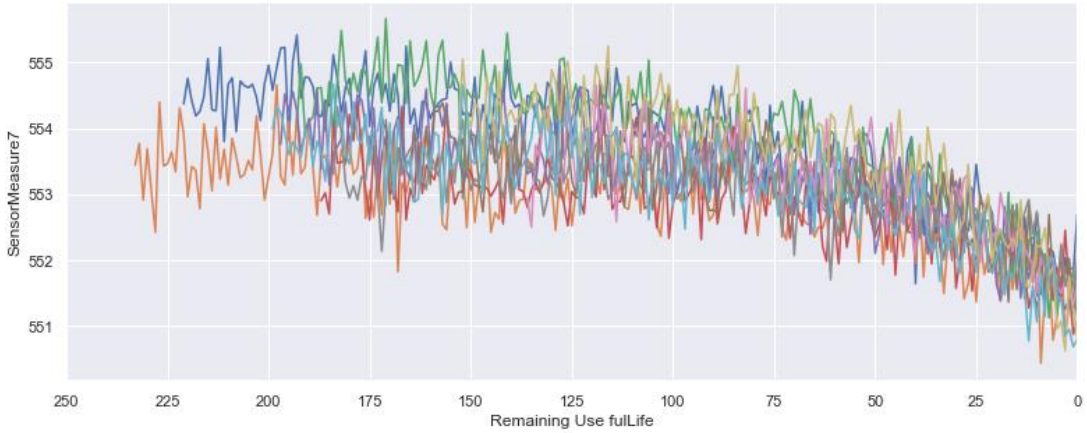


Figure 17: Graph of sensor 7

Figure 17 shows the graph of sensor 7. The value shows a decreasing trend as the RUL decreases. The decreasing trend can also be found in sensors 12, 20, and 21.

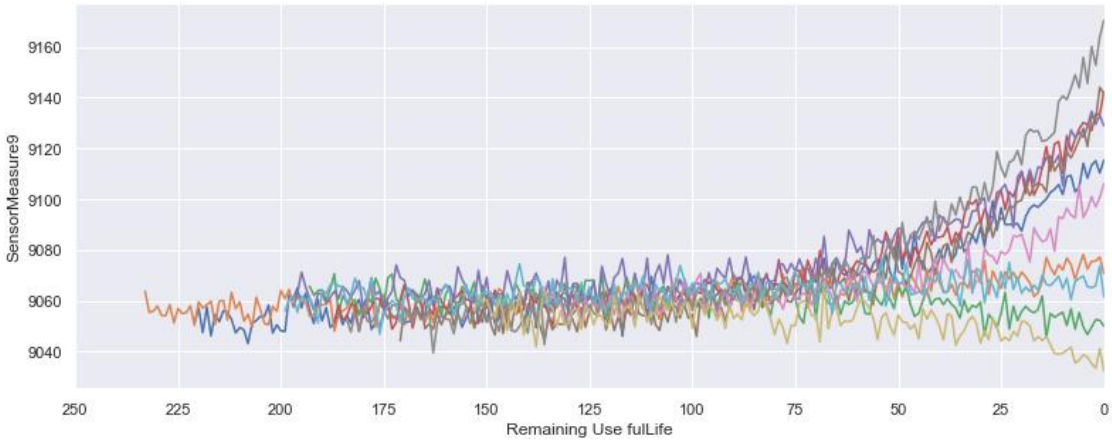


Figure 18: Graph of sensor 9

Figure 18 shows the graph of sensor 9. In this graph, the values of certain engines show an increasing trend while others show a decreasing trend. A similar pattern can be found in sensor 14.

Based on the EDA conducted, sensors 1, 5, 6, 10, 16, 18 and 19 are determined as sensors that hold no useful information related to RUL.

5.3 Predictive Model

After performing EDA on the dataset, the predictive model is built. Other than GBR, various algorithms are used to create the model and compare their performance. Since dataset FD001 operates under 1 operating condition, only the sensor measurements are considered during the training process.

5.3.1 Compute RUL

Before training the model, data pre-processing is performed. Since the dataset does not contain any incorrect data, missing values or irrelevant data, the data cleaning process is skipped. The original dataset does not contain the RUL of the engines thus it needs to be calculated. The RUL serves as the target variable for the supervised machine learning model.

ID	Cycle	RUL	ID	Cycle	RUL
1	1	191	1	1	142
1	2	190	1	2	141
1	3	189	1	3	140
1	4	188	1	4	139
1	5	187	1	5	138

Figure 19: Train (left) and test (right) set RUL

Figure 19 shows the calculated RUL in the train and test set. Only the last cycle of each engine in the test set is considered during the training process as they are the records that have true RUL. The code used to calculate the RUL can be accessed in the python script provided. Both sets are then saved as CSV files to be used later in the training process.

5.3.2 Feature Selection and Feature Extraction

In order to confirm the assumption made above where sensor 1, 5, 6, 10, 16, 18 and 19 does not contain any useful information, Pearson's correlation coefficient method is performed. It is a filter-based feature selection method.

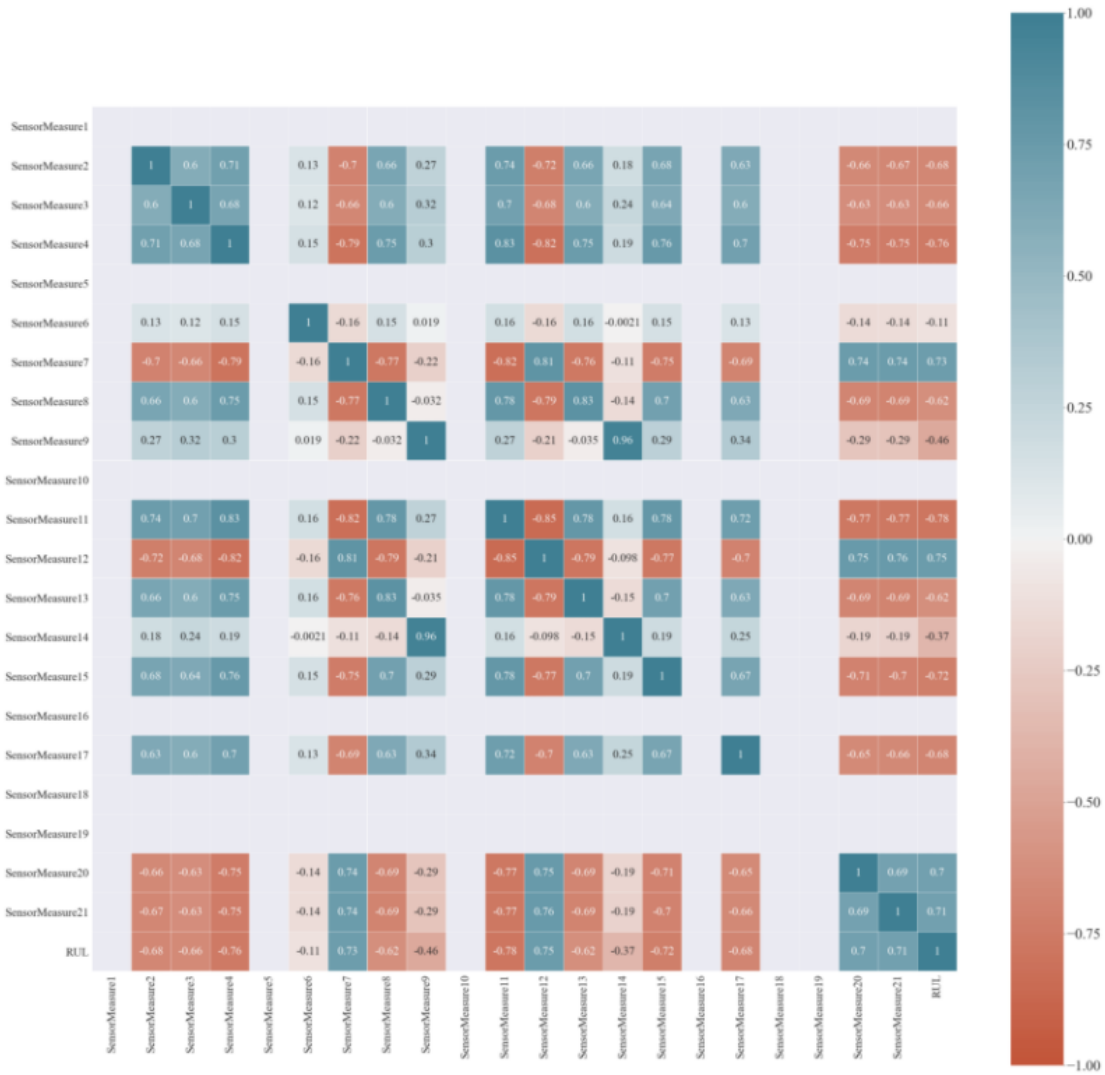


Figure 20: Heatmap (Pearson)

Figure 20 shows the heatmap of the feature selection. The positive value (blue) indicates there is a positive relationship between the two features while the negative value (red) indicates the negative relationship. Sensor 1, 5, 10, 16, 18, and 19 have values 0 related to RUL indicating there is no relationship between them. Although sensor 6 is not 0, it has a value very close to 0 thus it can be assumed that the relationship is not significant. Based on the heatmap, the assumption made in EDA is confirmed.

Since the dataset contains 21 sensor measurements, feature extraction is required to reduce the dimension of the dataset thus speeding up the training process. The feature extraction method used in this project is PCA.

```python
#Principle Component Analysis
def pca(X_train, X_test, y_train, y_test):

    #for reproducible result
    np.random.seed(2)

    X_train, X_test = standard_scaler(X_train, X_test) #standardize data

    #rename columns after standardization
    sensor_names = ['SensorMeasure{}'.format(i) for i in range(1,22)]
    X_train = pd.DataFrame(X_train, columns = sensor_names).reset_index(drop = True)
    X_test = pd.DataFrame(X_test, columns = sensor_names).reset_index(drop = True)

    #--------------------------------Fitting Model----------------------------
    # Make an instance of the Model
    pca = PCA(0.95, random_state = 1) #95% of the variance (information) is retained

    x1 = pca.fit_transform(X_train) #PCA based on train data
    x2 = pca.transform(X_test) #transform test data

    #Graph to indicate information contains in each component
    fir = plt.figure(figsize=(8,5))
    sing_vals = np.arange(len(pca.components_)) + 1
    plt.plot(sing_vals, pca.explained_variance_ratio_, 'ro-', linewidth=2)
    plt.title('Scree Plot', fontsize = 20)
    plt.xlabel('Principal Component', fontsize = 20)
    plt.ylabel('Eigenvalue', fontsize = 20)
    plt.xticks(fontsize=10)
    plt.yticks(fontsize=10)
    #------------------------------------------------------------------------

    #convert to data frame
    X_train = pd.DataFrame(data = x1)
    X_test = pd.DataFrame(data = x2)
    #------------------------------------------------------------------------

    return X_train, X_test, y_train, y_test
```

Code Snippet 2: PCA method

Code snippet 2 shows the method used to perform PCA. First, the datasets are standardized. Features other than sensor measurements are then removed from the datasets. Next, the train set is used to fit the PCA model. The 0.95 in the PCA model defines only 95% of the information are retained. After the model is trained, the train and test set is transformed using the model. In order to understand the outcome, a graph is created.

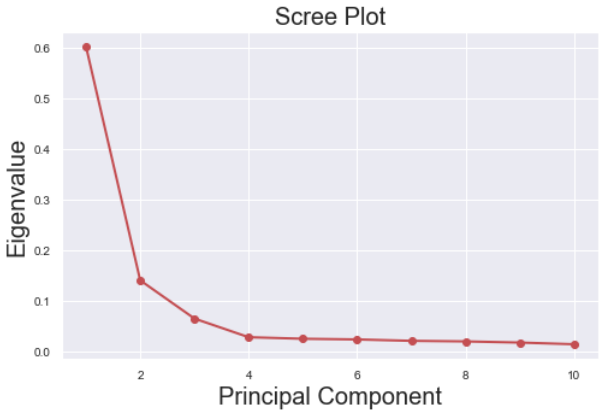


Figure 21: Graph of PCA

Figure 21 shows the graph of the PCA components. The graph shows that only 10 components are required to hold 95% of the information. The eigenvalues can be assumed as the amount of information the associated components hold. For example, the first PCA component holds more than half of the retained information. By using PCA feature extraction, the dimension of the dataset is reduced thus fewer computing resources are needed to train the predictive model.

5.3.3 Baseline Model

A baseline model provides a minimum performance that an algorithm should achieve. There are two baseline models created in this project which are the average model and linear regression model. The average model is derived from the idea of the Zero Rule algorithm for classification problems where the majority category is taken as the prediction. Standardization and feature extraction processes are not performed to remain the model as simplest as possible.

```
def zero_rule_algorithm(X_train, X_test, y_train, y_test):

    prediction = sum(y_train) / len(y_train) #get average

    #average as prediction
    y_pred_train = [prediction for i in range(len(X_train))]
    y_pred_test = [prediction for i in range(len(X_test))]

    #--------------------------------RMSE & R2--------------------------------
    evaluate(y_train, y_pred_train, 'Train')
    evaluate(y_test, y_pred_test, 'Test')
    #-------------------------------------------------------------------------
```

Code Snippet 3: Average model method

Code snippet 3 shows the method used to perform the average model. This model takes the average RUL of the train set as the prediction for the test set. The RMSE of the train and test set are 68.88 and 52.62 respectively. The R^2 values are 0.0 and -0.60 respectively. The results are expectable because the residual increase as the RUL increases beyond the average. The performance matrices indicate that the model is very inaccurate.

```
def linear_regression(X_train, X_test, y_train, y_test):

    #for reproducible result
    np.random.seed(2)

    #-------------------------------Train Model-------------------------------
    LR = LinearRegression()

    LR.fit(X_train, y_train) #train model
    #-------------------------------------------------------------------------

    #-------------------------------Predict X---------------------------------
    y_pred_train = LR.predict(X_train) #predict on train data
    y_pred_test = LR.predict(X_test) #predict on test data
    #-------------------------------------------------------------------------

    #-------------------------------Accuracy----------------------------------
    # Use score method to get accuracy of model
    accuracy_score = LR.score(X_test, y_test)
    print('Accuracy of Linear Regression on test set: {:.2f}'.format(accuracy_score))
    #-------------------------------------------------------------------------

    #-------------------------------RMSE & R2---------------------------------
    evaluate(y_train, y_pred_train, 'Train')
    evaluate(y_test, y_pred_test, 'Test')
    #-------------------------------------------------------------------------
```

Code Snippet 4: Linear regression model method

Code snippet 4 shows the method used to perform linear regression. The random.seed is used to ensure that the result is reproducible. The linear model is fitted using the train set. The trained model is then used to predict the train and test set to evaluate the performance. The RMSE of the train and test set are 44.66 and 31.95 respectively. The R^2 values are 0.58 and 0.41 respectively. The result of the linear model is slightly better than the average model. Thus it is used as the baseline performance for other models.

5.3.4 Improved Linear Regression

The first model trained is the improved version of linear regression where standardization and feature extraction are considered.

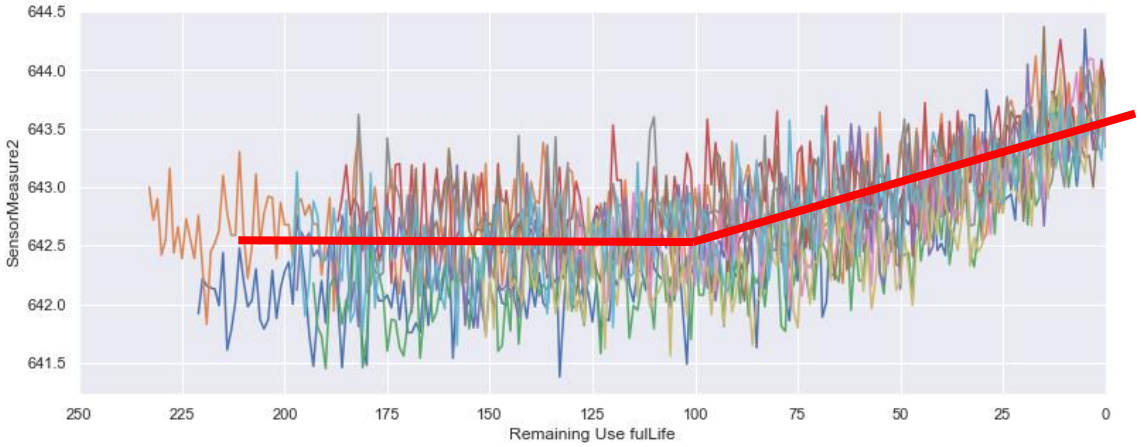


Figure 22: Signal plot of sensor 2

```
train_RUL = train_RUL.clip(upper = 125)
```

Code Snippet 5: Set upper bound for RUL

Originally, an assumption is made where RUL reduces linearly over time. However, most of the sensors remain constant in the beginning as shown in **figure 22**. This condition can be explained as engines only degrade over time. In order to reflect this logic, **code snippet 5** is used to fix the RUL in the beginning and only start to reduce linearly after a certain point. 125 is selected as it is the value that leads to the best model performance.

```
X_train, X_test, y_train, y_test = pca(train, test, train_y, test_y)
linear_regression(X_train, X_test, y_train, y_test)
```

Code Snippet 6: Call methods to train a linear regression model

Code snippet 6 shows the methods used to perform improved linear regression. The linear regression algorithm remains the same as the previous model. However, standardization and PCA feature extraction are performed before training the predictive model.

The RMSE of the train and test set reduced to 21.51 and 21.90 respectively while the R^2 increase to 0.73 and 0.72 respectively. This indicates that the standardisation and PCA are helping in training the model.

5.3.5 Polynomial Regression

Polynomial regression is considered the special case of multiple linear regression. It models the relationship between the independent and dependent variables as the nth degree polynomial.

```
def polynomial_regression(X_train, X_test, y_train, y_test):

    #for reproducible result
    np.random.seed(2)

    #------------------------------Train Model-------------------------------
    degree=4 #degree ogf polynomial

    polyreg_scaled=make_pipeline(PolynomialFeatures(degree),LinearRegression()) #pipeline
    polyreg_scaled.fit(X_train,y_train.values.ravel()) #train model
    #------------------------------------------------------------------------

    #------------------------------Predict X---------------------------------
    y_pred_train = polyreg_scaled.predict(X_train) #predict on train data
    y_pred_test = polyreg_scaled.predict(X_test) #predict on test data
    #------------------------------------------------------------------------

    #-----------------------------------Accuracy-----------------------------
    # Use score method to get accuracy of model
    accuracy_score = polyreg_scaled.score(X_test, y_test)
    print('Accuracy of Polynomial Regression on test set: {:.2f}'.format(accuracy_score))
    #------------------------------------------------------------------------

    #-----------------------------------RMSE & R2----------------------------
    evaluate(y_train, y_pred_train, 'Train')
    evaluate(y_test, y_pred_test, 'Test')
    #------------------------------------------------------------------------
```

Code Snippet 7: Polynomial regression model method

Code snippet 7 above shows the code used to train the polynomial regression model. The model is built using linear regression but the input to the model is a feature matric that consists of polynomial combinations of the features with a specified degree. The *PolynomialFeatures()* is the method used to create the polynomial feature matrix. The degree of polynomial selected is 4 which produce the best result before overfitting. The RMSE of the train and test set are 18.06 and 18.42 respectively while the R2 are 0.81 and 0.80 respectively.

5.3.6 Support Vector Regression

Support Vector Regression (SVR) uses the same principle as the Support Vector Machine. The idea of SVR is to find the best fit line which is the hyperplane that has the maximum number of points.

```
def support_vector_regression(X_train, X_test, y_train, y_test):

    #for reproducible result
    np.random.seed(2)

    #------------------------------Train Model-------------------------------
    SVRM = SVR()

    c = [int(x) for x in np.linspace(1, 15, num = 15)] #different value for C
    epsilon = [int(x) for x in np.linspace(1, 20, num = 20)] #different value for epsilon

    #Create the dictionary
    param_grid = {'C': c,
                  'epsilon': epsilon}

    #K-fold cross validation
    cv = KFold(n_splits=10, shuffle = True, random_state = 1)

    #randomized search
    rand_search = RandomizedSearchCV(SVRM, param_distributions = param_grid,
                                     cv = cv, n_jobs=-1, random_state=1)

    rand_search.fit(X_train, y_train.values.ravel()) #train model

    #SVRM.fit(X_train, y_train.values.ravel()) #train model on default settings
    #------------------------------------------------------------------------

    #------------------------------Predict X---------------------------------
    y_pred_train = rand_search.predict(X_train) #predict on train data
    y_pred_test = rand_search.predict(X_test) #predict on test data

    #for default model
    #y_pred_train = SVRM.predict(X_train)
    #y_pred_test = SVRM.predict(X_test)
    #------------------------------------------------------------------------

    #---------------------------------Accuracy-------------------------------
    # Use score method to get accuracy of model
    accuracy_score = rand_search.score(X_test, y_test)
    #accuracy_score = SVRM.score(X_test, y_test) #for default model
    print('Accuracy of Support Vector Regression on test set: {:.2f}'.format(accuracy_score))
    #------------------------------------------------------------------------

    #---------------------------------RMSE & R2------------------------------
    evaluate(y_train, y_pred_train, 'Train')
    evaluate(y_test, y_pred_test, 'Test')
    #------------------------------------------------------------------------

    #---------------------------------Best Param-----------------------------
    print(rand_search.best_params_) #parameters that provide best result
    #------------------------------------------------------------------------
```

Code Snippet 8: Support vector regression model method

Code snippet 8 shows the code used to train the SVR model. Initially, the model is trained using the default parameters with Radial Basis Function (RBF) kernel. The default model generates RMSE of 19.53 and R^2 of 0.78 for the test set. The result is worse than the Polynomial Regression.

In order to improve the result, hyperparameter tuning is performed. There are two main parameters to be tuned in this model which are C (regularization parameter) and epsilon (margin of tolerance where no penalty is given to errors) (Sharp, 2020). The hyperparameter tuning is performed using randomized search and K-fold cross-validation. Randomized search randomly forms a set of parameters from the available values and the K-fold cross-validation splits the dataset into the specified number of groups. The values of C are from 1 to 15 and the epsilon values are from 1 to 20. The dataset is split into 10 different groups during the process.

After performing hyperparameter tuning, the best value for C and epsilon are both 10. The RMSE of the train and test set are 18.13 and 18.38 respectively while the R2 are 0.81 and 0.80 respectively.

5.3.7 Random Forest

Since Random Forest has various similarities with GBR, it is also trained to compare the result against GBR.

```
def random_forest_reggressor(X_train, X_test, y_train, y_test):

    #for reproducible result
    np.random.seed(2)

    #------------------------------Train Model-------------------------------
    RFR = RandomForestRegressor(random_state = 1)

    #The number of trees in the forest (100)
    n_estimators = [int(x) for x in np.linspace(80, 150, num = 20)]
    #The maximum depth of the tree (None)
    max_depth = [int(x) for x in np.linspace(2, 15, num = 10)]
    max_depth.append(None)
    #The minimum number of samples required to split an internal node (2)
    min_samples_split = [2, 5, 10]
    #The minimum number of samples required to be at a leaf node (1)
    min_samples_leaf = [1, 2, 4]

    # Create the dictionary
    param_grid = {'n_estimators': n_estimators,
                  'min_samples_split': min_samples_split,
                  'min_samples_leaf': min_samples_leaf,
                  'max_depth': max_depth}

    #K-fold cross validation
    cv = KFold(n_splits=10, shuffle = True, random_state = 1)

    #randomized search
    rand_search = RandomizedSearchCV(RFR, param_distributions = param_grid,
                                     cv = cv, n_jobs=-1, random_state=1)

    rand_search.fit(X_train, y_train.values.ravel()) #train model

    #RFR.fit(X_train, y_train.values.ravel()) #train model on default settings
    #------------------------------------------------------------------------

    #------------------------------Predict X---------------------------------
    y_pred_train = rand_search.predict(X_train) #predict on train data
    y_pred_test = rand_search.predict(X_test) #predict on test data

    #for default model
    #y_pred_train = RFR.predict(X_train)
    #y_pred_test = RFR.predict(X_test)
    #------------------------------------------------------------------------

    #----------------------------------Accuracy------------------------------
    # Use score method to get accuracy of model
    accuracy_score = rand_search.score(X_test, y_test)
    #accuracy_score = RFR.score(X_test, y_test) #for default model
    print('Accuracy of Random Forest on test set: {:.2f}'.format(accuracy_score))
    #------------------------------------------------------------------------

    #----------------------------------RMSE & R2-----------------------------
    evaluate(y_train, y_pred_train, 'Train')
    evaluate(y_test, y_pred_test, 'Test')
    #------------------------------------------------------------------------

    #----------------------------------Best Param----------------------------
    print(rand_search.best_params_) #parameters that provide best result
    #------------------------------------------------------------------------
```

Code Snippet 9: Random Forest Regression model method

Code snippet 9 shows the method used to train the Random Forest Regression model. The training process is similar to SVR where the default model is trained and hyperparameter tuning is then performed. The default model generates RMSE of 6.85 and 17.72, and R^2 of 0.97 and 0.82 for the train and test set respectively. The model is overfitting as the RMSE for the train set is nearly half of the RMSE for the test set. In order to solve this problem, hyperparameter tuning is performed. There are 4 main parameters to be considered in RF (Pedregosa et al., 2011).

1. *n_estimators*: The number of trees in the forest.
2. *max_depth*: The maximum depth of the tree
3. *min_samples_split*: The minimum number of samples required to split an internal node
4. min_samples_leaf: The minimum number of samples required to be at a leaf node

The best parameters selected by Randomised Search are shown in **table 5**. The RMSE of the train and test set are 16.51 and 17.68 respectively while the R2 are 0.84 and 0.82 respectively.

Parameter	Value
n_estimators	83
max_depth	9
min_samples_split	10
min_samples_leaf	4

Table 5: Random Forest parameters

5.3.8 Gradient Boosting regression

Gradient Boosting Regression is the main algorithm used in this project. The training process is similar to the previous algorithm but a more tedious hyperparameter tuning is performed after Randomised Search.

```
def gradient_boosting_regressor(X_train, X_test, y_train, y_test):

    #for reproducible result
    np.random.seed(2)

    #-----------------------------Train Model-------------------------------

    GBR = GradientBoostingRegressor(learning_rate = 0.1,
                                    n_estimators = 100,
                                    subsample = 0.6,
                                    min_samples_split = 1400,
                                    max_depth = 5,
                                    min_samples_leaf = 1,
                                    random_state=1)

    #shrinks the contribution of each tree (0.1)
    learning_rate = [0.001, 0.01, 0.1, 0.2, 0.3]
    #The number of boosting stages to perform (100)
    n_estimators = list(range(40, 200, 20))
    #The fraction of samples to be used for fitting the individual base learners
    subsample = [0.2, 0.4, 0.6, 0.8, 1.0]

    #minimum number of samples required to split an internal node (2)
    min_samples_split = list(range(1000,10000,400))
    #Maximum depth of the individual regression estimators (3)
    max_depth = list(range(1,10,2))
    #max_depth.append(None)

    #minimum number of samples required to be at a leaf node (1)
    min_samples_leaf = list([1])

    param_grid = {'learning_rate': learning_rate,
                  'n_estimators': n_estimators,
                  'subsample': subsample,
                  'min_samples_split': min_samples_split,
                  'max_depth': max_depth,
                  'min_samples_leaf': min_samples_leaf}

    #K-fold cross validation
    cv = KFold(n_splits=10, shuffle = True, random_state = 1)

    #grid search
    #grid_search = GridSearchCV(GBR, param_grid = param_grid,
                               #cv = cv, n_jobs=-1)

    #grid_search.fit(X_train, y_train.values.ravel()) #train model using grid search

    GBR.fit(X_train, y_train.values.ravel())
    #-----------------------------------------------------------------------
```

Code Snippet 10: Gradient Boosting Regression model method

```
#------------------------------Predict X_test-----------------------------
#y_pred_train = grid_search.predict(X_train)
#y_pred_test = grid_search.predict(X_test)

y_pred_train = GBR.predict(X_train) #predict on train data
y_pred_test = GBR.predict(X_test) #predict on tets data
#-------------------------------------------------------------------------

#---------------------------------Accuracy--------------------------------
# Use score method to get accuracy of model
#accuracy_score = grid_search.score(X_test, y_test) #for grid search
accuracy_score = GBR.score(X_test, y_test)
print('Accuracy of Gradient Boosting Regression on test set: {:.2f}'.format(accuracy_score))
#-------------------------------------------------------------------------

#---------------------------------RMSE & R2-------------------------------
evaluate(y_train, y_pred_train, 'Train')
evaluate(y_test, y_pred_test, 'Test')
#-------------------------------------------------------------------------

#---------------------------------Best Param------------------------------
#print(grid_search.best_params_) #best parameters of randomised search
#print(grid_search.best_score_) #best score of grid search
#print('\n')
#-------------------------------------------------------------------------
```

Code Snippet 11: Gradient Boosting Regression model method

Code snippet 10 and 11 shows the method used to train the GBR model. By using the default parameters, the RMSE of the train and test set are 17.488 and 17.36 respectively and R^2 are 0.824 and 0.825 respectively. Next, Randomised Search is performed to tune the parameters. There are 6 parameters to be considered in this model (Pedregosa et al., 2011).

1. *learning_rate*: The step size at each iteration when moving towards the minimum of the loss function
2. *n_estimators*: The number of boosting stages
3. *subsample*: The fraction of samples to be used for fitting the individual base learners
4. *min_samples_split*: The minimum number of samples required to split an internal node
5. *max_depth*: Maximum depth of the individual regression estimators
6. *min_samples_leaf*: The minimum number of samples required to be at a leaf node

The best parameters selected by Randomised Search are shown in **table 6**. The trained model produces RMSE of 17.760 and 17.371, and R^2 of 0.818 and 0.825 for train and test set respectively.

Parameter	Value
learning_rate	0.3
n_estimators	120
subsample	0.8
min_samples_split	9800
max_depth	3
min_samples_leaf	17

Table 6: Randomised Search parameters' value

After the Randomised Search, Grid Search is performed to further tune the parameters. Unlike Randomised Search, it is performed in batches. **Table 7** shows the parameters tuned in each batch.

Batch	Parameters
1	learning_rate, n_estimators, subsample
2	min_samples_split, max_depth
3	min_samples_leaf

Table 7: Parameters in each batch

The parameters selected by Grid Search are shown in **table 8**. The trained model produces RMSE of 17.287 and 17.344, and R^2 of 0.828 and 0.826 for the train and test set. The result improved very slightly from the Randomised Search.

Parameter	Value
learning_rate	0.1
n_estimators	100
subsample	0.6
min_samples_split	1400
max_depth	5
min_samples_leaf	1

Table 8: Grid Search parameters' value

5.4 Dashboard

5.4.1 Engine interface

Engine interface is the main form in this dashboard. It is used to display the condition and RUL of the engines. Users can change the engine displayed through the drop-down menu.

```
/// <summary>
/// when drop down menu index changed
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
2 references
public async void MachineList_SelectedIndexChanged(object sender, EventArgs e)
{
    //show loading screen
    TopPanel.Visible = false;
    BottomPanel.Visible = false;
    loadingPanel.Visible = true;

    await showMachine(); //change engine data

    //unload loading screen
    TopPanel.Visible = true;
    BottomPanel.Visible = true;
    loadingPanel.Visible = false;
}
```

Code Snippet 12: Display loading screen and run task

Code snippet 12 shows the action performed when users change the engine ID in the drop-down menu. First, the engine interface is replaced by a loading screen to indicate the system is processing the new information. Next, the *showMachine()* task is performed.

```
/// <summary>
/// task to get engine data
/// </summary>
/// <returns></returns>
2 references
private async Task showMachine()
{
    machineID = MachineList.SelectedIndex + 1; //selected engine ID
    modelFile = ModelForm.currentModelFile; //selected model

    await Task.Run(() =>
    {
        Cycle machineCycle = getCycle(machineID); //get last cycle of engine

        int RUL = getRUL(machineID, modelFile, machineCycle); //predict RUL of engine

        this.Invoke(new MethodInvoker(delegate ()
        {
            //display value
            show value
        }));

    });
}
```

Code Snippet 13: Task to retrieve information of engine

```
/// <summary>
/// get last cycle of engine
/// </summary>
/// <param name="ID"></param>
/// <returns></returns>
1 reference
private Cycle getCycle(int ID)
{
    Cycle machineCycle = new Cycle(); //initialise cycle object

    //connect to database
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        conn.Open();

        //SQL query
        var sql = @"SELECT a.*
                    FROM [FinalYearProject].[dbo].[Turbofan_Engine] a
                    INNER JOIN (
                        SELECT [ID], MAX([Cycle]) cycle
                        FROM [FinalYearProject].[dbo].[Turbofan_Engine]
                        WHERE [ID] = @ID
                        GROUP BY [ID]
                    ) b ON a.ID = b.ID AND a.Cycle = b.cycle";

        SqlCommand cmd = new SqlCommand();
        cmd.CommandText = sql;
        cmd.CommandType = System.Data.CommandType.Text;
        cmd.Connection = conn;
        cmd.Parameters.AddWithValue("@ID", ID);
        SqlDataReader dr = cmd.ExecuteReader();

        //get data
        while (dr.Read())
        {
            machineCycle.ID = dr.GetInt32(dr.GetOrdinal("ID"));
            machineCycle.cycle = dr.GetInt32(dr.GetOrdinal("Cycle"));
            machineCycle.OpSet1 = (double)dr.GetDecimal(dr.GetOrdinal("OpSet1"));
            machineCycle.OpSet2 = (double)dr.GetDecimal(dr.GetOrdinal("OpSet2"));
            machineCycle.OpSet3 = (double)dr.GetDecimal(dr.GetOrdinal("OpSet3"));

            for (int i = 1; i <= 21; i++)
            {
                machineCycle.GetType().GetProperty("s_" + i).SetValue(machineCycle, (double)dr.GetDecimal(dr.GetOrdinal("SensorMeasure"+i)), null);
            }
        }
        conn.Close();
    }
    return machineCycle;
}
```

Code Snippet 14: Get the latest cycle of the engine from the database

```
/// <summary>
/// predict RUL of engine
/// </summary>
/// <param name="ID"></param>
/// <param name="modelFile"></param>
/// <param name="machineCycle"></param>
/// <returns></returns>
1 reference
private int getRUL(int ID, string modelFile, Cycle machineCycle)
{
    Predictor b = new Predictor(pythonFile, scriptFile); //create predictor

    double output = b.prediction(ID, modelFile, machineCycle); //perform prediction
    int result = Convert.ToInt32(output);

    return result;
}
```

Code Snippet 15: Get the RUL of the engine

Code snippet 13 to 15 shows the sequence that the information is retrieved. In order to process the information, the engine ID and model file path is retrieved. **Code snippet 14** shows the code used to connect the SQL server and retrieves the latest cycles of the selected engine. The RUL prediction is then performed to get the predicted RUL as shown in **code snippet 15**. Finally, this information is updated on the engine interface.

5.4.2 Model interface

The model interface allows users to view the available models and their performance. Users can also change the model used in the prediction through the drop-down menu.

```
/// <summary>
/// when drop down menu index changed
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 reference
private void ModelList_SelectedIndexChanged(object sender, EventArgs e)
{
    Model currentModel = (Model)modelList[ModelList.SelectedIndex];
    currentModelFile = currentModel.modelFile; //get model fiel path

    if(EngineForm.machineList != null)
    {
        EngineForm.form2.MachineList_SelectedIndexChanged(EngineForm.machineList, new EventArgs());
    }

    //display performance matrix
    TrainRMSEMeasure.Text = currentModel.trainRMSE.ToString();
    TestRMSEMeasure.Text = currentModel.testRMSE.ToString();
    TrainR2Measure.Text = currentModel.trainR2.ToString();
    TestR2Measure.Text = currentModel.testR2.ToString();
}
```

Code Snippet 16: Update the model file path and display performance matrices

Code snippet 16 shows the code performed when the index of the drop-down menu changed (*ModelList_SelectedIndexChanged*). First, the file path of the selected model is retrieved. Next, the index changed event (*MachineList_SelectedIndexChanged*) of the drop-down menu in the engine form is activated to update the engine information using the selected model. Finally, the performance matrix of the model is displayed on the dashboard.

5.4.3 Speech Recognition

The speech recognition function is created in the parent form which contains the engine and model forms. The code snippet above shows the initialization of the speech recognition when the form is loaded. The number of engine commands is based on the number of engines specified in the configuration file. There are 3 fixed commands which are used to switch between different child forms.

```
/// <summary>
/// when command detected by recogniser
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 reference
private void Default_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
    if (e.Result.Confidence >= 0.75) //if confidence more than 75%
    {
        string speech = e.Result.Text; //convert to string
        string[] tokens = speech.Split(' '); //number of words detected

        if (speech == "Lock Screen")
        {
            LockButton_Click(LockButton, new EventArgs());
        }
        else if (speech == "Show Engine")
        {
            EngineButton_Click(EngineButton, new EventArgs());
        }
        else if (speech == "Show Model")
        {
            ModelButton_Click(ModelButton, new EventArgs());
        }
        else if (tokens.Length >= 3) //if mroe than 3 words deteced
        {
            if (tokens[0] == "Show" && (tokens[1] == "Engine" || tokens[1] == "Number"))
            {
                string result = String.Join(" ", speech.Split(' ').Skip(2)); //catch words after first 2 words

                try
                {
                    int machineNumb = Convert.ToInt32(WordsToNumbers.ConvertToNumbers(result)); //convert to engine ID
                    EngineForm.machineList.SelectedIndex = machineNumb - 1; //change engine ID in engine form

                    EngineButton_Click(EngineButton, new EventArgs());
                }
                catch (Exception ex)
                {
                }
            }
        }
    }
}
```

Code Snippet 17: Perform actions based on command identified

Code snippet 17 shows the action performed when a command is identified. When the fixed command is identified, the corresponding button event is activated to display the associated form. When the identified command has more than 3 words, the first and second word is checked to match the expected word. Next, the remaining words after the second word are extracted to convert to an integer using the static method. The integer represents the engine ID which is used to change the engine displayed in the engine form.

5.4.4 Toggle Button

The speech recognition function is controlled through the toggle button in the main form. Since C# does not support toggle button control, it is created manually by inheriting checkbox control.

```
/// <summary>
/// draw button
/// </summary>
/// <param name="pevent"></param>
0 references
protected override void OnPaint(PaintEventArgs pevent)
{
    int toggleSize = this.Height - 5;
    pevent.Graphics.SmoothingMode = SmoothingMode.AntiAlias;
    pevent.Graphics.Clear(this.Parent.BackColor);

    if (this.Checked) //ON
    {
        //Draw control surface
        pevent.Graphics.FillPath(new SolidBrush(OnBackColor), GetFigurePath());
        //Draw toggle
        pevent.Graphics.FillEllipse(new SolidBrush(OnToggleColor),
            new Rectangle(this.Width - this.Height + 1, 2, toggleSize, toggleSize));
    }
    else //OFF
    {
        //Draw control surface
        pevent.Graphics.FillPath(new SolidBrush(OffBackColor), GetFigurePath());
        //Draw toggle
        pevent.Graphics.FillEllipse(new SolidBrush(OffToggleColor),
            new Rectangle(2, 2, toggleSize, toggleSize));
    }
}
```

Code Snippet 18: Toggle button

The *OnPaint()* event shown in **code snippet 18** is used to draw the toggle button. When the button is checked, the background of the toggle button is drawn with a specific colour using the *FillPath()* and *GetFigurePath()* functions, and the circle is drawn on the right side of the background using the *FillEllipse()* function. On the other hand, when the button is unchecked, the background is drawn with different colours and the circle is drawn on the left side.

5.4.5 WordsToNumbers

The command identified by speech recognition is converted to words. In order to identify the engine ID, the command needs to be converted to integer as discussed above. Thus, a new class with a static method is created to convert words to an integer using a predefined dictionary.

```
public static long ConvertToNumbers(string numberString)
{
    Debug.WriteLine($"Original: {numberString}");

    var numbers = Regex.Matches(numberString, @"\w+").Cast<Match>()
            .Select(m => m.Value.ToLowerInvariant())
            .Where(v => numberTable.ContainsKey(v))
            .Select(v => numberTable[v]);

    foreach(var n in numbers)
    {
        Debug.WriteLine($"n: {n}");
    }

    long acc = 0, total = 0L;
    foreach (var n in numbers)
    {
        if (n >= 1000)
        {
            total += acc * n;
            acc = 0;
        }
        else if (n >= 100)
        {
            acc *= n;
        }
        else acc += n;
    }
    return (total + acc) * (numberString.StartsWith("minus",
            StringComparison.InvariantCultureIgnoreCase) ? -1 : 1);
}
```

Code Snippet 19: Convert words to a number

Code snippet 19 shows the static method used to convert the string to an integer. The string parameter is first separated into digits using the Regex class. A series of calculations are then performed to get the numeric form of the input string. By using this method, the text representation of numbers can be converted to numeric form. For example, “one hundred” can be converted to 100.

5.5 Summary

This chapter discussed each stage in the Machine Learning Life Cycle. The Exploratory Data Analysis of the FD001 dataset is discussed to identify the characteristics of the dataset. Besides, the feature extraction process is also explained in this chapter to understand how the dimension of the dataset can be reduced thus improving the efficiency of the training process. Furthermore, the development of a predictive model using 5 different algorithms are illustrated. Lastly, the development of the dashboard and the method to implement the predictive model in the dashboard are also discussed.

CHAPTER 6
TESTING

6.1 Overview

The testing phase is a critical process in Software Development Lifecycle (SDLC). It helps to assess the product to ensure that it is bug-free and fulfils the user requirements. The functional and non-functional requirements are tested in this phase to identify any bugs or errors. There are 3 different test plans conducted in this phase.

6.2 Testing Strategy

Both white box testing and black box testing is performed in this process.

1. White box testing: It is a technique in which the internal structure and coding of the software are tested to verify the flow of input-output (Hamilton, 2021). In this testing, the code of the program is visible to the testers. Unit testing is performed which is based on each block of code.
2. Black box testing: It is a technique in which the functionalities of the program are tested (Hamilton, 2021). In this testing, the testers do not have knowledge of the internal structure and coding of the software. Functional and Non-functional testing is performed to verify the system requirements.

6.3 Unit Testing

6.3.1 Verify side menu buttons

Test Scenario	**Test Case ID**	**Test Case Scenario**	**Expected Result**	**Result**
Verify correct form is displayed when the button clicked	TC1	Users click the lock button	Lock screen form is displayed	Pass
	TC2	Users click the engine button	Engine form is displayed	Pass
	TC3	Users click the model button	The model form is displayed	Pass

6.3.2 Verify words to numbers function

Test Scenario	**Test Case ID**	**Test Case Scenario**	**Attribute and value**	**Expected Result**	**Result**
Verify correct integer is converted from a string	TC1	Speech recognition identifies a number	result: one hundred	The function returns 100 in integer	Pass

6.3.3 Verify loading screen function

Test Scenario	Test Case ID	Test Case Scenario	Expected Result	Result
Verify loading screen is displayed when engine form performs operations	TC1	The program is first started	The loading screen is displayed until the dashboard finish operations	Pass
	TC2	Users change engine ID in the drop-down menu	The loading screen is displayed until the dashboard finish operations	Pass
	TC3	Dashboard update the data every 5 minutes	The loading screen is displayed until the dashboard finish operations	Pass

6.3.4 Verify automated update operation

Test Scenario	Test Case ID	Test Case Scenario	Expected Result	Result
Verify engine Data is updated every 5 minutes	TC1	Users view the engine form until the 5th minute	The loading screen pop up until the update is completed	Pass
	TC2	Users viewing other forms	The update process performed in the background	Pass

6.3.5 Verify changing predictive model

Test Scenario	Test Case ID	Test Case Scenario	Expected Result	Result
Verify changing predictive model	TC1	Users change the model in the model form	The engine form is updated and the performance matrix of the model is displayed on the model form	Pass

6.3.6 Verify connection to SQL server

Test Scenario	Test Case ID	Test Case Scenario	Expected Result	Result
Verify connection to SQL server	TC1	When *getCycle()* function is called	A correct record is retrieved	Pass

6.4 Functional testing

6.4.1 Verify engine data

Test Scenario	Test Case ID	Test Case Scenario	Expected Result	Result
Verify information of engine	TC1	Users click the engine button	The correct information of a specific engine is displayed	Pass
	TC2	Users change the engine ID in the drop-down menu	The information of the corresponding engine is displayed	Pass

6.4.2 Verify engine's predicted RUL

Test Scenario	Test Case ID	Test Case Scenario	Expected Result	Result
Verify predicted RUL of engine	TC1	Users click the engine button	The predicted RUL of the specific engine is displayed	Pass
	TC2	Users change the engine ID in the drop-down menu	The predicted RUL of the corresponding engine is displayed	Pass

6.4.3 Verify update engine data operation

Test Scenario	Test Case ID	Test Case Scenario	Expected Result	Result
Verify engine data is updated every 5 minutes	TC1	Users viewing the engine form	The loading screen is displayed and the update operation is performed	Pass
	TC2	Users viewing other forms	The update operation is performed in the background	Pass

6.5 Non-functional testing

6.5.1 Verify lock screen function

Test Scenario	Test Case ID	Test Case Scenario	Expected Result	Result
Verify lock screen is displayed	TC1	Users click the lock button	Lock screen form is displayed	Pass

6.5.2 verify speech recognition function

Test Scenario	Test Case ID	Test Case Scenario	Expected Result	Result
Verify speech recognition function	TC1	Users speak “lock screen”	Lock screen form is displayed	Pass
	TC2	Users speak “show engine”	Engine form is displayed	Pass
	TC3	Users speak “show model”	The model form is displayed	Pass
	TC4	Users speak “show engine one hundred”	Engine 100 is displayed on the engine form	Pass

6.5.2 Verify models’ performance matrices

Test Scenario	Test Case ID	Test Case Scenario	Expected Result	Result
Verify performance matrices of model	TC1	Users click the model button	The performance matrices of the currently used model are displayed	Pass
	TC2	Users change the model in the drop-down menu	The performance matrices of the corresponding model are displayed	Pass

6.6 Summary

This chapter discussed the strategy used to test the software. Besides, the test cases are stated to explain how the functionalities of the software are tested and the corresponding result.

CHAPTER 7
EVALUATION

7.0 Introduction

This chapter discusses the evaluation of the product and project process. It evaluates the product in terms of its strengths and weaknesses. Besides, the successes and failures during the development process are covered in this chapter. This chapter also discusses the changes made in the project proposal. Finally, it evaluates the project process.

7.1 Evaluation of product

There are two products developed in this project. The first product is the predictive model that predicts the RUL of turbofan engines based on the current conditions. Another product is a dashboard that displays the conditions of the engine and the predicted value from the predictive model.

The strength of the product is it provides the critical information to support predictive maintenance (PdM). The current maintenance strategy adopted by industries is Run-To-Failure (R2F) and preventive maintenance (PvM). Both strategies are inefficient and lead to costly maintenance. This product estimates the RUL of the engine and support flexible maintenance. Besides, the strength of the dashboard is the performance matrices of various predictive models are visible to users. For example, users can view the RMSE and R2 matrices to understand the performance of the GBR model on predicting the RUL of engines. Furthermore, the dashboard is simple and intuitive. The dashboard does not require any complex operation to perform its main functionalities.

Although the dashboard has various strengths, there are also some weaknesses. One of the weaknesses of the product is it is not the best model that provides the highest accuracy. Due to the time constraint, the techniques used in this project is relatively simple as compared to time series based feature extraction and neural network. Based on the research conducted, Truncated Back Propagation Through Time (BPTT) and Extended Kalman Filter can be applied together with Recurrent Neural Network (RNN) to achieve high accuracy. However, due to the complexity of these algorithms and the time constraint, simpler algorithms are used. Furthermore, there is a weakness in one of the functionalities of the dashboard which is speech recognition. The accuracy of the recognition is highly based on the accent of the speakers. Although various alternatives provide better accuracy, they are not used because they recognize words in an audio file. For example, users have to start recording the command as an audio file and send it to the recognizer. This method is inefficient as the users have to start and stop recording every time. In order to improve the accuracy of the existing recognizer, recognition confidence is added. For example, if the confidence is below 75%, the dashboard will not perform the command. However, more improvement can be made to further improve the accuracy.

The main feature of the product is the prediction of RUL of turbofan engine because it is the main objective of the project. It uses Principle Component Analysis (PCA) and Gradient Boosting Regression (RBG) algorithms to build the predictive model to predictive the RUL of the engine based on its current conditions. Displaying the sensor values of the engine is also another important feature. This feature displays the sensor measurement of the corresponding engine to the users. The sensor values allow users to better understand the condition of the engine. By combining the RUL and sensor values, users are able to gain insight to decide whether to perform maintenance. Furthermore, displaying the model performance is considered as an additional feature. This feature allows users to better understand the performance of the model and make a comparison between models.

There were many issues faced during the development process. One of the issues was finding a suitable algorithm that provides acceptable accuracy. Due to the complexity of various algorithms, it is almost impossible to train such models within the time constraint. In order to search for suitable algorithms, some of the famous algorithms such as Linear Regression, Support Vector Machine and Random Forest are trained. However, there is still space for improvement. After conducting a lot of research on machine learning, a GBR algorithm was found. It is an algorithm that has many achievements in many competitions especially in Kaggle, one of the famous machine learning communities. Besides, another difficulty faced during the dashboard development was integrating the trained model into C#. Unlike common development, this project requires the integration between model trained in python and dashboard developed in C#. The trained model cannot be used directly in C#. One of the proposed solutions was to convert the model to file type that C# supports. However, such conversion may lead to performance loss. After researching on this topic, another solution was proposed where a python script that predict the RUL was executed directly from C#. After solving this problem, another issue was faced which was extended from the proposed solution. Although the prediction can be performed, the time required to retrieve data from SQL server to python was long. In order to solve this problem, the data is sent directly from C# to python instead of python retrieving data directly from the SQL server.

```
for (int i = 1; i <= 21; i++)
{
    machineCycle.GetType().GetProperty("s_" + i).SetValue(machineCycle, (double)dr.GetDecimal(dr.GetOrdinal("SensorMeasure"+i)), null);
}
```

Code Snippet 20: Loop to set cycle properties

```
for (int i = 1; i <= 21; i++)
{
    double value = (double)lastCycle.GetType().GetProperty("s_" + i).GetValue(lastCycle, null);
    sensorsArray[i - 1] = value;
}
```

Code Snippet 21: Loop to get cycle properties

There are various implementation techniques that are worth pointing out. The first technique is the looping of variables to set values. Since there are 21 sensor values, normally it requires 21 lines of code to set the value for each variable. However, this method is not efficient. Since the sensor variables are distinguished using numbers, various type class functions are used to loop through 21 variables and set values as shown in **code snippet 20**. This method only requires 2 lines of code which is more efficient. A similar method is also used when passing the engine record to the python script as shown in **code snippet 21**.

```
var psi = new ProcessStartInfo();
psi.FileName = pythonfile;

var script = pythonScript;

double[] sensorsArray = new double[21];

//loop object variables to add to array
for (int i = 1; i <= 21; i++)
{
    double value = (double)lastCycle.GetType().GetProperty("s_" + i).GetValue(lastCycle, null);
    sensorsArray[i - 1] = value;
}

string sensors = string.Join(", ", sensorsArray);

psi.Arguments = $"\"{script}\" \"{modelFile}\" \"{sensors}\""; //arguments to python script

psi.UseShellExecute = false;
psi.CreateNoWindow = true;
psi.RedirectStandardOutput = true;
psi.RedirectStandardError = true;

var error = "";
var output = "";

using (var process = Process.Start(psi))
{
    error = process.StandardError.ReadToEnd();
    output = process.StandardOutput.ReadToEnd();
}
```

Code Snippet 22: C# process

Another implementation technique worth mentioning is the method to execute the python script in C#. Since the predictive model cannot be supported in C#, the execution of python script to use the model is inevitable. In order to do so, a process is created with the python executable in C# as shown in **code snippet 22**. The script and other inputs are passed as arguments of the process to python. With this method, the model trained in python can be directly used in C# without any modification.

There are five different algorithms tested on data set FD001 as discussed in chapter 5. **Table 9** shows their comparison result on the data set in terms of RMSE and R^2 values. It is observed that GBR achieved the lowest RMSE value. Among the five algorithms, the baseline model achieved the highest RMSE value which is almost double the RMSE of GBR. This indicates that the GBR algorithm is able to produce results closer to the actual RUL. Besides, GBR also achieved the highest R2 value while the baseline model has the lowest R2 value. The GBR is twice as better as the baseline model. This indicates that the model trained using the GBR algorithm is able to explain the dependent variable of the data.

Algorithm	RMSE	R^2
Baseline model	31.95	0.41
Linear Regression	21.90	0.72
Polynomial Regression	18.42	0.80
Support Vector Regression	18.38	0.80
Random Forest Regression	17.68	0.82
Gradient Boosting Regression	17.34	0.83

Table 9: Performance comparison among algorithms

Table 10 shows the GBR results as compared to works done by other authors. It is observed that the proposed method is able to achieve slightly better performance than the work proposed in [7]. The methods proposed in [32] and [18] are able to achieve a lower RMSE value. The proposed method is able to achieve result close to Neural Network technique which is more difficult and complicated to train. Since the difference of RMSE is only within 3 cycles, the result achieved by the GBR algorithm is considered acceptable.

Author	Method	RMSE
[7]	Convolution Neural Network	18.4480
[32]	MODBNE	15.04
[18]	Time Window Neural Network	15.1593
Proposed method	Gradient Boosting Regression	17.34

Table 10: Performance comparison with other authors

Due to the Non-Disclosure Agreement (NDA) signed during the internship, there are various changes made in the project proposal. The first change is the nature of the project. Originally, the project was served as an improvement to the existing dashboard of the company by improving the user interface and speech recognition and implements machine learning. After the internship ends, the program is no longer accessible. Therefore, in this project, a new dashboard, user interface, speech recognition and machine learning use case are created. Due to this reason, the project title, aims and objectives need to be refined. However, there are no major changes in the project but all products have to be developed from scratch.

7.2 Evaluation of project process

Although the Gantt Chart created in Project Initialization Document (PID) was not followed after the Year 3 Semester 1 was completed due to the NDA issue discussed above, the progress of the project went rather smoothly thanks to the experience gained from the internship. Unlike the Gantt Chart, the implementation phase of the project started after the internship. During the progress, the priorities of the features are adjusted such that the machine learning part is prioritized as it is the most time consuming which took around 6-7 weeks.

Once the machine learning section is done, the user interface and speech recognition sections started simultaneously to ensure the project can be completed within the time frame. Thanks to the experience gained during the internship, these sections only takes around 2 weeks to complete. The last few weeks of the time frame was used to prepare the documentation of the project and improve the program.

There are many lessons learned during the process of the project. One of the lessons learned is project planning. Planning is critical as it helps to respond to uncertainty and make adjustments rapidly. The project planning was done systematically in this project to ensure all aims and objectives are achieved. Due to the good project planning, the project is able to complete on time even uncertainty such as NDA issues does not impact the project progress significantly.

Besides, more machine learning techniques and knowledge are learned during this project. In order to train a model that helps in PdM, lots of research are conducted on machine learning algorithms. During this process, many knowledge and technique are learned to gain hands-on experience in training machine learning models.

Over and above, the aims and the objectives of the project have been achieved. All the features proposed in the project proposal are developed. A dashboard with a predictive model was developed to predict the RUL of the turbofan engine thus helping to perform predictive maintenance.

7.3 Summary

This chapter evaluates the overall product developed in this project. It discussed the strengths and weaknesses of the predictive model and the dashboard. Besides, it also discussed the difficulties faced during the development process and the lessons learned in this project. Finally, it evaluates the performance of the predictive models and compares them against the works done by other authors.

CHAPTER 8
CONCLUSION & RECOMMENDATION

8.1 Conclusion

To sum up, the predictive maintenance dashboard is software that displays the sensor measurements of an engine and its RUL predicted by the trained predictive model. It combines data visualization and data analytics to ease the process of decision making. It aims to provide critical information to support flexible maintenance strategies such as predictive maintenance. It is believed that as the "The Fourth Industrial Revolution" continue to progress, predictive maintenance will become increasingly important due to the rapidly changing market demand.

A simple process of the dashboard is the users are able to view the sensor measurements and the RUL of various engines through the engine interface. From this interface, users can understand the current conditions of the engines and decide whether to perform maintenance based on the information provided. Users can also decide to lock the screen to prevent others from accessing the information. Furthermore, in certain use cases, users may want to view the performance of the available models through the model interface. By changing the model in the interface, the selected model is used to predict the RUL of the engines. This allows users to make a comparison between different models.

Throughout this project, all the aims and objectives proposed in Project Proposal are achieved with little refinement due to the unexpected NDA issue. The objectives accomplished includes investigating various machine learning use cases, researching various machine learning algorithms for the use case, developing the functional and non-functional requirements of the system and preparing the final report for the project.

Although all aims and objectives are achieved, there are several improvements that could be made to further improve the system. One of the possible improvements is on the machine learning algorithm. Due to the time constraint, more complex algorithms such as Long-Short Term Memory (LSTM) and Convolution Neural Network (CNN) are not tested. The optimum outcome of the project is to train a model that is able to predict the RUL with an accuracy of more than 95%. However, it requires a lot of time to investigate different complex algorithms especially when the dataset contains time series data. For now, the algorithm used is the Gradient Boosting Regression. This algorithm is able to provide a great result but a better result is possible with a more complex algorithm that considers the time cycle of the data.

Other than the predictive model, there are also some improvements that can be made in the dashboard. One of the improvements is the speech recognition function. In this project, the speech recognition function is not mature enough to identify the command accurately every time. Further improvement can be made to enhance this function such that different accents can be identified correctly. In the future, different languages can even be considered to accommodate the needs of different users.

Furthermore, additional functions can be added to the dashboard in the future. One of the possible features is to allow users to upload different models. With this feature, the dashboard can continue to improve as the developers trained better models.

In conclusion, the emergence of machine learning and data analytics can change the way maintenance is performed. The main objective of predictive maintenance is to minimize production downtime, maximize equipment lifespan, optimize asset productivity and increase revenue. The aim of this project is to create a predictive model and dashboard to support predictive maintenance. It is hoped that the product developed in this project is able to provide value to businesses that wish to implement flexible maintenance strategies to improve production efficiency.

REFERENCES

1. Breiman, L., 2001. Random Forest. Machine Learning, [online] 45, pp.5-32. Available at: <https://link.springer.com/article/10.1023/a:1010933404324#citeas> [Accessed 2 November 2021]. (Breiman, 2001)

2. Brownlee, J., 2020. A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/> [Accessed 20 September 2021]. (Brownlee, 2020)

3. Canizo, M., Onieva, E., Conde, A., Charramendieta, S. and Trujillo, S., 2017. Real-time predictive maintenance for wind turbines using Big Data frameworks. 2017 IEEE International Conference on Prognostics and Health Management (ICPHM), [online] pp.70-77. Available at: <https://ieeexplore.ieee.org/document/7998308> [Accessed 1 October 2021]. (Canizo et al., 2017)

4. Carvalho, T., Soares, F., Vita, R., Francisco, R., Basto, J. and Alcalá, S., 2019. A systematic literature review of machine learning methods applied to predictive maintenance. Computers & Industrial Engineering, [online] 137, p.106024. Available at: <https://www.sciencedirect.com/science/article/pii/S0360835219304838> [Accessed 30 October 2021]. (Carvalho et al., 2019)

5. Chen, Y., Peng, G., Zhu, Z. and Li, S., 2020. A novel deep learning method based on attention mechanism for bearing remaining useful life prediction. Applied Soft Computing, [online] 86. Available at: <https://www.sciencedirect.com/science/article/pii/S1568494619307008> [Accessed 8 November 2021]. (Chen, Peng, Zhu and Li, 2020)

6. Gandhi, R., 2018. Support Vector Machine — Introduction to Machine Learning Algorithms. [online] towards data science. Available at: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47> [Accessed 13 September 2021]. (Gandhi, 2018)

7. Giduthuri, S., Zhao, P. and Li, X., 2016. Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life. Database Systems for Advanced Applications, [online] pp.214-228. Available at:<https://www.researchgate.net/publication/314933361_Deep_Convolutional_Neural_Network_Based_Regression_Approach_for_Estimation_of_Remaining_Useful_Life> [Accessed 2 November 2021]. (Giduthuri, Zhao and Li, 2016)

8. Glen, S., 2019. Decision Tree vs Random Forest vs Gradient Boosting Machines: Explained Simply. [online] Datasciencecentral. Available at: <https://www.datasciencecentral.com/profiles/blogs/decision-tree-vs-random-forest-vs-boosted-trees-explained> [Accessed 25 September 2021]. (Glen, 2019)

9. Gupta, S., 2021. Gradient Boosting Algorithm: A Complete Guide for Beginners. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/> [Accessed 20 October 2021]. (Gupta, 2021)

10. Hamilton, T., 2021. What is BLACK Box Testing? Techniques, Example & Types. [online] Guru99. Available at: <https://www.guru99.com/black-box-testing.html> [Accessed 15 November 2021]. (Hamilton, 2021)

11. Hamilton, T., 2021. What is WHITE Box Testing? Techniques, Example & Types. [online] Guru99. Available at: <https://www.guru99.com/white-box-testing.html> [Accessed 15 November 2021]. (Hamilton, 2021)

12. IBM. 2020. Neural Networks. [online] Available at: <https://www.ibm.com/my-en/cloud/learn/neural-networks> [Accessed 5 November 2021]. (IBM, 2020)

13. IBM. 2020. Speech Recognition. [online] Available at: <https://www.ibm.com/my-en/cloud/learn/speech-recognition> [Accessed 25 November 2021]. (Speech Recognition, 2020)

14. Jaadi, Z., 2021. A Step-by-Step Explanation of Principal Component Analysis (PCA). [online] Built In. Available at: <https://builtin.com/data-science/step-step-explanation-principal-component-analysis> [Accessed 24 September 2021]. (Jaadi, 2021)

15. Kearns, M., 1988. Thoughts on Hypothesis Boosting. Machine Learning class project, pp.1-9. (Kearns, 1988)

16. Khelif, R., Chebel-Morello, B., Malinowski, S., Laajili, E., Fnaiech, F. and Zerhouni, N., 2017. Direct Remaining Useful Life Estimation Based on Support Vector Regression. IEEE Transactions on Industrial Electronics, [online] 64(3), pp.2276-2285. Available at: <https://ieeexplore.ieee.org/document/7726039> [Accessed 10 October 2021]. (Khelif et al., 2017)

17. Kusiak, A. and Zhang, Z., 2011. Optimization of power and its variability with an artificial immune network algorithm. 2011 IEEE/PES Power Systems Conference and Exposition, [online] pp.1-8. Available at: <https://ieeexplore.ieee.org/document/5772600> [Accessed 25 October 2021]. (Kusiak and Zhang, 2011)

18. Li, X., Ding, Q. and Sun, J., 2018. Remaining useful life estimation in prognostics using deep convolution neural networks. Reliability Engineering & System Safety, [online] 172, pp.1-11. Available at: <https://www.sciencedirect.com/science/article/pii/S0951832017307779> [Accessed 23 September 2021]. (Li, Ding and Sun, 2018)

19. Lim, P., Goh, C. and Tan, K., 2016. A time window neural network based framework for Remaining Useful Life estimation. 2016 International Joint Conference on Neural Networks (IJCNN), [online] pp.1746-1753. Available at: <https://ieeexplore.ieee.org/document/7727410> [Accessed 13 September 2021]. (Lim, Goh and Tan, 2016)

20. Martin, M., 2021. What is a Functional Requirement in Software Engineering? Specification, Types, Examples. [online] Guru99. Available at: <https://www.guru99.com/functional-requirement-specification-example.html> [Accessed 2 November 2021]. (Martin, 2021)

21. Martin, M., 2021. What is Non-Functional Requirement in Software Engineering? Types and Examples. [online] Guru99. Available at: <https://www.guru99.com/non-functional-requirement-type-example.html> [Accessed 2 November 2021]. (Martin, 2021)

22. Mathew, J., Luo, M. and Pang, C., 2017. Regression kernel for prognostics with support vector machines. 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), [online] pp.1-5. Available at: <https://ieeexplore.ieee.org/document/8247740> [Accessed 13 September 2021]. (Mathew, Luo and Pang, 2017)

23. Noonpakdee, W., Khunkornsiri, T., Phothichai, A. and Danaisawat, K., 2018. A framework for analyzing and developing dashboard templates for small and medium enterprises. 2018 5th International Conference on Industrial Engineering and Applications (ICIEA), [online] pp.479-483. Available at: <https://ieeexplore.ieee.org/document/8387148> [Accessed 4 March 2021].

24. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, É., 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, [online] 12, pp.2825-2830. Available at: <https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html> [Accessed 21 September 2021]. (Pedregosa et al., 2011)

25. Praveenkumar, T., Saimurugan, M., Krishnakumar, P. and Ramachandran, K., 2014. Fault Diagnosis of Automobile Gearbox Based on Machine Learning Techniques. Procedia Engineering, [online] 97, pp.2092-2098. Available at: <https://www.sciencedirect.com/science/article/pii/S187770581403522X> [Accessed 17 September 2021]. (Praveenkumar, Saimurugan, Krishnakumar and Ramachandran, 2014)

26. Ravanshad, A., 2018. Gradient Boosting vs Random Forest. [online] Medium. Available at: <https://medium.com/@aravanshad/gradient-boosting-versus-random-forest-cfa3fa8f0d80> [Accessed 25 September 2021]. (Ravanshad, 2018)

27. Saxena, A., Goebel, K., Simon, D. and Eklund, N., 2008. Damage propagation modeling for aircraft engine run-to-failure simulation. 2008 International Conference on Prognostics and Health Management, [online] pp.1-9. Available at: <https://ieeexplore.ieee.org/document/4711414> [Accessed 16 April 2021]. (Saxena, Goebel, Simon and Eklund, 2008)

28. Sharp, T., 2020. An Introduction to Support Vector Regression (SVR). [online] towards data science. Available at: <https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2> [Accessed 2 October 2021]. (Sharp, 2020)

29. Singh, H., 2018. Understanding Gradient Boosting Machines. [online] towards data science. Available at: <https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab> [Accessed 30 September 2021]. (Singh, 2018)

30. Su, C. and Huang, S., 2018. Real-time big data analytics for hard disk drive predictive maintenance. Computers & Electrical Engineering, [online] 71, pp.93-101. Available at: <https://www.sciencedirect.com/science/article/pii/S0045790617328409> [Accessed 20 September 2021]. (Su and Huang, 2018)

31. Subrahmanyam, K., Ketha, T., Balakrishna, S. and Kumar, T., 2018. Development of Research & Development Dashboard For an University. International Journal of Engineering & Technology, [online] 7(2.32), pp.60-63. Available at: <https://www.researchgate.net/publication/325881968_Development_of_Research_Development_Dashboard_For_an_University> [Accessed 4 March 2021]. (Subrahmanyam, Ketha, Balakrishna and Kumar, 2018)

32. Susto, G., Schirru, A., Pampuri, S., McLoone, S. and Beghi, A., 2015. Machine Learning for Predictive Maintenance: A Multiple Classifier Approach. IEEE Transactions on Industrial Informatics, [online] 11(3), pp.812-820. Available at: <https://ieeexplore.ieee.org/document/6879441> [Accessed 9 November 2021]. (Susto et al., 2015)

33. Zhang, C., Lim, P., Qin, A. and Tan, K., 2017. Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics. IEEE Transactions on Neural Networks and Learning Systems, [online] 28(10), pp.2306-2318. Available at: <https://ieeexplore.ieee.org/document/7508982> [Accessed 12 October 2021]. (Zhang, Lim, Qin and Tan, 2017)

APPENDICES

Appendix A: Project Proposal

UOW MALAYSIA KDU PENANG UNIVERSITY COLLEGE

BACHELOR OF COMPUTER SCIENCE (HONS)

PROJECT TITLE – Implementing Machine Learning into the Dashboard for data analytics

Lim Khai Sian (0201576)

CCP3012 PROJECT PROPOSAL

Application Project

SEPTEMBER 2020

TABLE OF CONTENT

INTRODUCTION

The global competition and the demand for quick adaptation to the rapid-changing market demand are driving industrial production today. Only the revolutionary improvements in current manufacturing technology can fulfil such requests. Industry 4.0 is an approach that merges the business and manufacturing procedures and also stakeholders in an organization's value chain. The implementation of generic concepts of Cyber-Physical Systems (CPS) and the Internet of Things (IoT) to the production systems can be the solution to the technical aspect of these demands (Rojko, 2017). As a result, the Industry 4.0 'execution system' is built on top of the connections of CPS building blocks (Rojko, 2017). These blocks are embedded systems that collect and transfer real-time data among themselves with the objective of determining, detecting, monitoring and enhancing the production process. Besides, for seamless integration of manufacturing and business processes, a large scale software support that based on decentralised and adapted versions of Manufacturing Execution Systems (MES) and Enterprise Resource Planning (ERP) is required (Rojko, 2017). The management of a vast amount of data gathered from the activities, systems and products is the third critical aspect. The data collected is normally saved in cloud storage. An extensive analysis is required for the data to transform from 'raw' data to meaningful information. The information is then transformed into a physical process that supports an adaptive and self-optimising production process.

Business Intelligence (BI) has become a critical consideration for many organizations as they begin to transform into Industry 4.0. BI systems have been extensively implemented in various business. With the analytic capability of BI systems, the key business process can be enhanced significantly. BI allows organizations to leverage data within the enterprise to accomplish business objectives, maximise revenue, minimise expenses or even all the above.

Visualisation is an essential segment of the BI that support the presentation of data after processing and interpretation (Noonpakdee, Khunkornsiri, Phothichai and Danaisawat, 2018). Dashboards have become a common method to display vital data at a glance to support better communication. Digital dashboards are evolving rapidly and are widely applied in many areas of industry.

Dashboards are flexible to fulfil any requirements of an organization or department to display any critical data. Besides, it is the most efficient method to trace various data. The speed of information availability and reliability are critical in effective decision-making. In the event of inaccurate data arise, it may have a disastrous consequence on the decision-making process. As such, almost every company require the assistance of an Information and technology system (ICT) to acquire data rapidly and accurately (Subrahmanyam, Ketha, Balakrishna and Kumar, 2018). In fact, information is the outcome of processed data that are accessible for a certain purpose. The vast amount of data gathered from the activities, systems and products can be a burden for companies to arrange and interpret to be meaningful information (Subrahmanyam, Ketha, Balakrishna and Kumar, 2018). The Data Warehouse is the solution such that the huge amount of data does not hinder the existing business operations. The application dashboard allows the organization to access information in visualised form. Such information representation can help the decision-maker to interpret rapidly and efficiently.

In order to perform extensive analysis on the data collected, Machine Learning can also be integrated to further expand the effectiveness of the dashboard. It can be used to analyse data retrieved thus providing more valuable information to the organization. For example, the Machine Learning model can identify the factors that contribute to the changes in the data. It can help organizations to discover the underlying pattern that is usually missed out especially when facing a huge amount of data. The suitability of a Machine Learning model plays an important role in affecting the value of provided information. Inappropriate models not only produce meaningless data but also impact the overall data visualization. On the other hand, an appropriate model can help organizations to explore more valuable information from the visualised data.

This proposal is organised as follow. Section 2 presents the background of the project. Section 3 describes the key concepts of the project. In Section 4, the methodology is proposed. Section 5 presents the aim of the project. The Objectives of the project is discussed in Section 6. Section 7 presents the required skills for the project.

BACKGROUND OF THE PROJECT

Dashboards can be applied in various industry and department including sales, supply social media, manufacturing, etc. In this project, the Dashboard is used as a manufacturing dashboard to monitor the essential production KPI. It allows manufacturers to monitor and improve production efficiency. It is also a useful analytics tool for managing all manufacturing costs effectively. In general, a manufacturing dashboard is a real-time visualisation of the manufacturing process. It shows the critical KPI or metrics that reflect the performance of the process in graphical or chart form (Mitchell, n.d.).

The manufacturing dashboard visualises data and insight from sensors and systems in a graphical manner that is easy to understand. Thus, companies can understand their current and historical performance rapidly via a dashboard.

One of the core technology driving Industrial Revolution 4.0 is Artificial Intelligence (AI). Data has become a very valuable resource and it is now easier and more affordable to store than ever. Thanks to Artificial Intelligence specifically Machine Learning, more manufacturers are making use of data to boost their manufacturing performance.

Manufacturing Artificial Intelligence with Machine Learning methods is now being majorly invested by industry giants such as Bosch, Microsoft and NVIDIA to enhance any aspect of the manufacturing process (Chuprina, 2020).

Maintenance of equipment is one of the major expenses in the manufacturing sector as it is an essential part of any asset-based production activity. It is also preferable to identify problems before they exist rather than repairing failure as they occur or scheduling equipment checks. Machine Learning algorithms fine-tune the predictive Maintenance system to evaluate failure trend and forecast potential issues. The data used for predictive maintenance is time-series data which consist of a timestamp, a collection of sensor readings taken at the same time as timestamps, and system identifiers. The objective of predictive maintenance is to predict at the time of ‘t’ whether the equipment will malfunction in near future by utilising the data up to the time ‘t’. Thus, it is able to help organizations to save an enormous amount of money. Machine Learning can be utilised to produce reliable prediction about the condition of the assets and machines. The Remaining Useful Life (RUL) of equipment is extended greatly. Technicians will always know ahead whether there is something to be fixed or replaces and even know which approached to use exactly.

There are several machine learning models which can be used for predictive maintenance. There are several machine learning development approaches depending on the objective of prediction.

The following information regarding the approaches for predictive maintenance is from Maksymenko, 2021.

Regression Model: It can be used for estimating the Remaining Useful Life (RUL). This approach is able to estimate how many days left before a system failure by using historical and static data.

Classification Model: It is used to forecast system failure within a predetermined time frame. To predict when the system will malfunction, a model can be developed to predict failure within a certain number of days.

Anomaly detection models to flag devices. This method can be used to forecast failure by detecting variations between normal system behaviour and failure events.

The main focus of the project is to predict the Remaining Useful Life of equipment based on the data collected by sensors installed in the equipment. The sensors observe and collect the data regarding the equipment's operation. The regression Model is used to predict the remaining time left before the next failure. This can result in significant cost reduction, high stability, and increase the availability of the systems. Predictive Maintenance can minimise or even avoid system downtimes and improve the periodic maintenance activities thus optimising all three aspects of the Overall Equipment Effectiveness which are Availability Performance and Quality.

The predicted RUL of the equipment is then displayed on the dashboard to act as an indicator to the organization when the equipment will fail and require actions such as corrective actions, repair or even replacement of the system.

One of the main benefits of Machine Learning-based predictive maintenance is accuracy and effectiveness. It is possible to enhance the performance before a malfunction event occurs by identifying irregularities in the production appliances and evaluate their nature and frequency.

By doing predictive maintenance, it is able to improve the Overall Equipment Performance (OEE). Overall Equipment Performance is a gold standard for assessing the manufacturing efficiency of an asset. In general, it determines the amount of time spent on manufacturing that is genuinely efficient.

DEFINITION OF KEY CONCEPTS

Dashboard

A dashboard is an information management tool that displays data in an interactive and visual way. It is the most effective approach to trace various data sources as it offers a central location to monitor and interpret the performance (Klipfolio, n.d.). A dashboard usually links with a database to retrieve the required data. It is customisable to allow users to opt for required data and the method to visualise the data. The purpose of a dashboard relies on the role it performs within an organization.

A dashboard visualises data to allow users to understand the analytics that is critical to the organization. Furthermore, it also allows non-technical users to comprehend the analytics process by compiling data and visualise the trends and occurrences (Klipfolio, n.d.). It offers an analytical view of performance indicator which can be an effective support for further dialogue.

Machine Learning

Machine Learning is a subject of Artificial Intelligence which concentrate on offering system the capability to learn from a large amount of data and improve the accuracy over time without being explicitly programmed (IBM, 2020). There are 3 types of Machine Learning which are Supervised Machine Learning, Unsupervised Machine Learning and Semi-Supervised Machine Learning.

In data science, an algorithm is a set of statistical processing steps. Machine Learning algorithms are used to find the patterns and behaviours in a huge amount of data to aid decision-making and prediction based on new data. A good algorithm can offer higher decision and prediction accuracy as more data is interpreted.

There are several steps to build a Machine Learning model. The first step is to select and prepare an appropriate dataset. A training dataset is used to represent the data that the Machine Learning model will ingest to solve the problem. The dataset should be pre-processed to ensure that there is no bias that will impact the training. Normally it is divided into 2 subsets which is a training set (used to train the model) and an evaluation set (used to test and tune the model).

The next step is to select a suitable algorithm to execute on the training dataset. The types of algorithm chosen depend on the types of the dataset which is either labelled or unlabelled. The type of problem to be solved should be considered during the selection of an algorithm.

The algorithm is then trained to generate a model. Running variables through the algorithm, comparing the outcome to the predicted output and tuning the weights and biased are all part of the algorithm training process (IBM, 2020). This process is performed repeatedly until the algorithm produces a more accurate result.

The last step is to improve the model. New data is processed in the created model to enhance accuracy and performance over time.

Data Analytics

Data Analytics is the process of analysing to discover trends and draw conclusions regarding the underlying information contained (Stedman, 2020). Data Analytic techniques can be used to uncover trends in the raw data thus extracting useful insight from it. Increasingly data analytic techniques are used with the aid of specialised systems and software that incorporate machine learning algorithms, automation and other capabilities.

Data Analytics is a broad concept that encompasses a wide range of applications ranging from basic Business Intelligence (BI), reporting and online analytical processing (OLAP) and advanced analytics (Stedman, 2020). Therefore, it is similar to the nature of business analytics. However, Business Intelligence is more focused on business applications whereas data analytics is more general.

Data analytics can be classified into four different categories which are descriptive, diagnostic, predictive and prescriptive analytics. Each category is used for a different purpose. They are also the most popular data analytics applications in business.

The following information regarding the category of data analytics is from Master's in Data Science, 2020.

Descriptive analytics answer questions about what happened. These methods interpret huge datasets for stakeholders to understand the result. This process involves the gathering of related data, data processing, data interpretation and data visualisation. It also offers crucial information about the past.

Diagnostic analytics explain the reason for the result. These methods are used in conjunction with more basic descriptive analytics. They take the results of descriptive analytics and discover the root of the problem. The performance indicators are then interpreted to identify the reason for such performance.

Predictive analytics helps to predict the future based on existing data. These approaches make use of historical data to classify the patterns and determine whether they will happen again. Predictive analytical methods use a range of statistical and machine learning techniques to provide useful insight into what could happen in the future.

Prescriptive analytics provides suggestions for decisions to be made. Data-driven decisions can be made by making use of useful insight from predictive analytics. Machine learning algorithms are used in prescriptive analytics techniques to identify trends in large datasets. The probability of different various result can be estimated by evaluating the pest decision and events.

Predictive Maintenance

Predictive maintenance is an approach to detect irregularities in the operation and potential faults in equipment and processes by using data analysis tools and techniques, thus the system can be repaired before they result in failure.

Theoretically, predictive maintenance helps to keep maintenance frequency as low as possible to reduce unplanned reactive maintenance while avoiding the costs by doing excessive preventive maintenance.

Predictive maintenance is made up of many primary components with technology and software being part of it. Different systems can be linked, work together, share, and interpret data thanks to the Internet of Things, Artificial Intelligence and integrated systems. These tools make use of Predictive maintenance sensors, industrial controls and business systems such as EAM software and ERP software to collect data. They further decode it and use it to determine any aspect that needs to be aware of.

Speech Recognition

Speech Recognition is also called Automatic Speech Recognition (ASR) which enable a computer or program to process human voice into readable text (IBM, 2020). Speech Recognition is always confused with Voice Recognition. However, Speech Recognition perform conversion of spoken words to text format, while, Voice Recognition attempt to recognise the voice of a particle user. Speech Recognition research encompasses many topics including artificial intelligence, linguistics, digital signal processing and cognitive science.

PROPOSED METHODOLOGY

The methodology used in this project is Rapid Application Development. It is a type of agile software development methodology that emphasises rapid prototyping and iteration. It prioritises the use of software and user feedback than extensive planning and requirements documentation in the Waterfall model. There are 4 steps in Rapid Application Development which includes Define Project Requirements, Build Prototypes, Receive User Feedback and Finalise Project.

There are several benefits of implementing Rapid Application Development. It allows developers to make modification quickly by enhancing the flexibility and additivity of the software development process. Besides, rapid iteration can shorten the development time and accelerates this implementation. Furthermore, it also improves customer satisfaction as a result of continuous collaboration and coordination between stakeholders.

In this project, each feature discussed in the proposed work will have an isolated RAD process. The various RAD process can progress simultaneously to ensure the proposed work can be completed within the time frame. Each stage of the RAD process will be reviewed and refined before moving to the next stage. More effort will be put into the RAD process of implementing a Machine Learning model in the dashboard for data analytics as it is the main purpose of the project.

User Interface

For User Interface Improvement, the requirements such as programming language and development software will be defined during the "Define Project Requirements" stage. Furthermore, the aesthetics requirements of the user interface such as colour, orientation and theme will be identified in this stage. The finalise requirements are then reviewed.

After the requirements of the User Interface is finalise, sketching that fulfil the requirements is created for the User Interface. Before the development process, the sketching is reviewed by the supervisor to provide some feedback. After modification on the sketching is completed, the actual User Interface is created using the required programming language.

In the final process, the User Interface is finalised with the supervisor. The aesthetics, usability and maintainability of the User Interface must be considered before finalising the result.

Speech Recognition

For Speech Recognition, the requirements of speech recognition such as accuracy, commands, recognition time and budget are documented during the initial stage of the process.

During the prototyping stage, a speech recognition API that satisfies the requirements will be found from the API marketplace named RapidAPI. Other API marketplaces will be considered if there is no suitable API in the RapidAPI. Before an API is selected, it is important to consider the supported language and programming language. Once an appropriate API is found, it will be explored to understand the available function and how can it be integrated into the software. Next, it is integrated to provide a speech recognition function for the dashboard. Before finalising this feature, tests will be conducted for the speech recognition function to identify the accuracy and recognition speed. The speed recognition function and the result is then reviewed by the supervisor.

Lastly, the speech recognition function in the dashboard is finalised with the supervisor.

Machine Learning

During the initial process, the problem to be solved for predictive maintenance is identified. The available problems include estimating the Remaining Useful Life (RUL) and flagging irregular behaviour of the asset. Besides, the machine learning model to build to solve the problem is determined. Furthermore, a suitable dataset for building the model must be obtained before moving to the next stage.

The dataset used in this project is obtained from NASA data repository which is the Turbofan Engine Degradation Dataset (Benchmark Data). The Turbofan engine utilised by NASA space exploration agency is a modern gas turbine engine (Saxena and Goebel, 2008). The dataset was developed by NASA to forecast the Turbofan engine failures over time. The dataset can be obtained from NASA Ames Prognostics Data Repository [1].

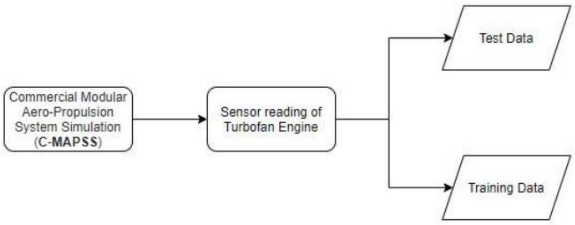


Figure 2: Data Pipeline of Turbofan Engine Degradation Dataset

The dataset is generated by using the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) which is a turbofan simulation model. The dataset is further divided into training and test sets.

The following information regarding the Turbofan Engine Degradation Dataset (Benchmark Data) used in this project is from Saxena and Goebel, 2008

The run-to-failure sensor measurements of various degrading turbofan engine are included in the dataset. Despite the fact that all turbofan engines are of the same type, each engine starts with different degrees of initial wear and manufacturing variation which is unknown to the user.

Each engine has three different optional settings which can be utilised to alter the performance. Each engine has 21 sensors that collect data about the condition of the engine at run time. As mentioned in the readme.txt, the data collected is contaminated with sensor noise.

At the start of the time series, the operations of the engines are under normal condition. As the engine operates, it begins to develop fault. The time series comes to an end before the failure.

The Machine Learning used in this project is Supervised Machine Learning. There are two sets of data available in the selected dataset which are Training data and Testing data. Both data include the actual RUL that the Regression model will predict.

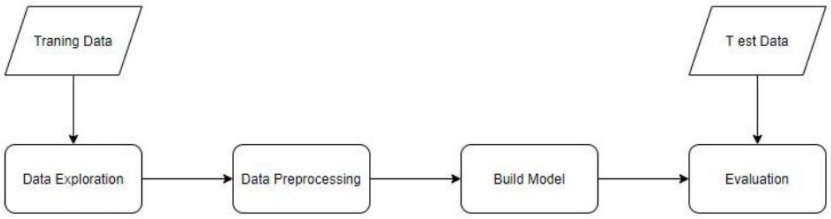


Figure 3: Flow Chart for Building Machine Learning Model

The Figure above shows how both data can be used for building the Machine Learning Model.

In the prototyping stage, the dataset is pre-processed before being used to build the model. The pre-processing steps include: (Mayo, 2018)

- Cleaning the dataset by removing the duplicate record, handle missing value, normalization, data type conversion and etc.
- Visualise the data to identify any relationship between the columns or perform any exploratory analysis.
- Separate the dataset into the Training dataset and Test dataset

After the dataset is cleaned, it is used to train a machine learning model. During the training process, a random value is initialised for the weight of the model which will multiply or impact the relationship between the inputs and outputs (Yufeng, 2017). The weight value will be adjusted during the process to produces a more ideal result.

Once the training process is completed, the test dataset which contains the inputs that the model does not train with before is feed into it. This allows us to understand whether the model is able to produce a similar result as in the training process. Parameter tuning can be performed if the model does not produce a good result in the evaluation stage (Yufeng, 2017).

Finally, the machine learning model is integrated with the dashboard to solve real-world problems.

During the process of building the machine learning model, the intermediate result of each stage is reviewed by the supervisor to get some useful feedback.

After the machine learning model is integrated into the dashboard, it is reviewed again by the supervisor before finalising it.

13

PROPOSED WORK

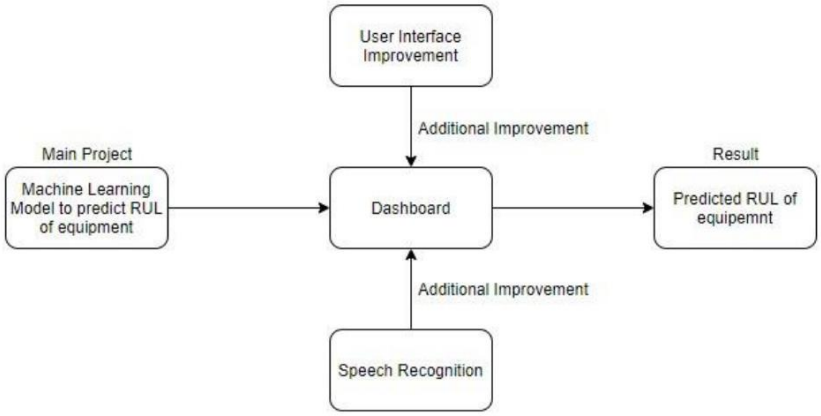


Figure 1: Proposed Work

As shown in the figure above, the main focus of the project is to develop a Machine Learning model for predicting the RUL of equipment. The Result of the predicted RUL is then displayed on the dashboard. An additional improvement of the user interface is also included in this project to enhance the user experience of using the dashboard. Besides, the accuracy of the existing speech recognition in the dashboard is also improved during this project.

This machine learning model can be made use by different users such as the production department or engineer responsible for the production machine or anyone who concerned with the condition of the machine. This system allows users to be informed of the current condition of the equipment thus able to perform necessary action before the machines break down. Users can control the system via User Interface or Speech Recognition.

User Interface Enhancement

User Interface plays an important role in helping users to understand the function of the dashboard. It should be designed in such a way that all the main features are displayed intuitively to the users. A good user interface not only focuses on aesthetics but also maximising the effectiveness and accessibility of the dashboard. A user interface is an end where the user interacts with the dashboard. A good user interface can present a better impression of the dashboard but also allowing users to use it without additional training.

In order to improve the user interface of the dashboard, there are several principles that should be considered such as Nielsen 10 Usability Heuristics and Shneiderman's Eight Golden Rules of Interface Design. Before, modifying the user interface of the dashboard, these principles should be referred to assess the usability and efficiency of the user interface.

The paragraphs below are examples of how the user interface of a dashboard can be assessed based on the Nielsen 10 Usability Heuristics principles.

According to the "visibility of system status" in Nielsen 10 Usability Heuristics, the dashboard should provide suitable feedback upon the user's action to inform the user about the current state of the system. For example, the dashboard can display a progression bar when it is retrieving data from databases or performing machine learning tasks.

Furthermore, the dashboard should display text and icon that the users can understand. This concept refers to the principle of “match between system and the real world”. The sentences and icons displayed in the dashboard should be replaced if users have to search explicitly.

Improve Speech recognition

Speech recognition is integrated into the Dashboard to ease the process of human-computer interaction. Speech recognition can be used to perform control tasks that usually performed through mouse and keyboard. It can increase the efficiency of using the dashboard by providing a more intuitive method of interaction. With speed recognition, one can retrieve or display desired data through speech. However, speech recognition with low accuracy can have the opposite effect on effectiveness. Speech recognition that unable to recognize the speech or recognize incorrectly can be annoying to the users. Thus it is important to ensure a certain accuracy for speech recognition.

Currently, the dashboard is using the System. Speech for speech recognition. System. Speech uses a desktop version of API. However, the accuracy of the API is not sufficient. Oftentimes, it is unable to detect the command from the user or identifying incorrect command.

In order to improve the accuracy of speech recognition, System. Speech API should be replaced with other API with high accuracy. An API marketplace named "RapidAPI" will be used to search for any suitable API for speech recognition in C#.

The accuracy of speech recognition should be assessed to ensure that it can fulfil the objective of using commands to operate the dashboard.

Develop Machine Learning Model for Predictive Maintenance

There are several steps involved to build a Machine Learning Model.

1. Collect Data
2. Data pre-processing
3. Select and Train Machine Learning Model
4. Evaluation
5. Parameter Tuning
6. Prediction

The dataset used in this project is obtained from NASA data repository which is the Turbofan Engine Degradation Dataset (Benchmark Data). In the initial stage, the dataset is explored to understand the information represented by the rows and columns.

Next, the dataset should be pre-processed as it will impact the result of the Machine Learning model directly. In order to produce an unbiased result, the dataset must remove any imbalanced data. Besides, the dataset is divided into two groups: one for training the model and another for evaluation.

After the dataset is processed, an appropriate model will be selected depending on the objective to be achieved. The type of data to be processed also play an important role in determining the model to be trained. There are several algorithms available such as classification, prediction, linear regression, clustering and etc.

The trained model will be assessed by using the evaluation dataset during the evaluation stage. The accuracy of the model is one of the criteria to determine that the model is successful.

Next, the parameter tuning stage will take place to tune the model parameter to generate a better result. The number of time to iterate the training data can be tested to improve the performance. Another essential parameter is the learning rate. It is a value that multiplies the gradient to continuously bring it closer to the global or local minimum to reduce the cost of the function (Rodriguez, 2020).

Finally, the model is used to solve some real-world problem. This is the stage where the value of the machine learning model is realised.

AIMS OF THE PROJECT

1. The aim of the project is to investigate on suitable Machine Learning model to integrate into the dashboard for Data Analytics.
2. The aim of the project is to improve the dashboard through Machine Learning methodologies for Data Analytics.

PROJECT OBJECTIVES

1. To write a Projection Initiation Document (PID)
2. To write a proposal on improving dashboard through Machine Learning techniques
3. To construct a Gantt Chart for time management
4. To install and explore the required software to improve the dashboard
5. To learn a suitable programming language such as C# for improving the system
6. To investigate Machine Learning Use Cases in Manufacturing (Predictive Maintenance, Digital Twin and quality optimization)
7. To acquire dataset for the Use Case (sensor reading of Equipment)
8. To research on various Machine Learning models for the Use Case (Regression, KNN and Random Forest)
9. To train the selected Machine Learning Model to predict the Remaining Useful Life of Equipment
10. To evaluate the accuracy of trained Machine Learning Model
11. To perform Hyperparameter optimization to generate a better result
12. To display the result of the Machine Learning Model on the dashboard
13. To improve the User Interface of Dashboard for enhanced User Experience
14. To improve the accuracy of Speech Recognition existing in the Dashboard
15. To complete the final document with all the stated chapters

SKILLS

In the Foundation of Human-Computer Interaction (CHI3031) subject, I had learned to create a User Interface that is easy to learn, effective to use and provide an enjoyable user experience. From this subject, I had learned to create a system that satisfies the objectives of Utility and Usability. Utility refers to the ability of the system to perform the intended tasks while Usability refers to the ease of learning and ease of use. Furthermore, I had learned the importance of having a good User Interface and the consequences of a bad User Interface. In order to create a good User Interface, there are 2 sets of principles that should be considered which are Nielsen's 10 usability Heuristics and Shneifermann's 8 golden rules of UI design.

In the subject of Object-Oriented Programming (CPR3023), I had learned about the concept of objects and classes. Besides, I had learned to develop an object-oriented program by applying the principles of inheritance, polymorphism and abstraction.

In the Software engineering (CSE3033) subject, I had learned about various phases in the Software Development Life Cycle (SDLC). Besides, I had learned multiple software development methodologies. This allows me to develop a system with the most efficient method by choosing the appropriate methodology based on the system and customer requirements. Furthermore, I had learned about different types of code smells. This helps me to improve the quality of the code, enhance the code comprehensibility and maintainability. Besides, I have also learned various methods to test the software.

In the Introduction to Artificial Intelligence (CAI3014) subject, I had learned about the basic concept of Artificial Intelligence. Besides, I had also learned to apply basic principles of AI in the solution that require problem-solving, inference, knowledge representation and learning.

In the subject of Principles of Programming language (CPR3033), I had learned about coherent syntax and memory management for various programming language. Besides, I had learned to evaluate the flows of execution of instructions in various run time environment. Furthermore, I had also learned about the concept of exception handling and apply it in various run time environments.

In the subject of Data Science Tools and Techniques (CDS3034), I had learned about how to pre-process data to remove any unwanted data from the dataset and converting to a suitable data type. This allows me to clean the dataset used to train the Machine Learning Model. An Unbiased dataset can help to improve the accuracy of the Machine Learning Model.

SOURCES OF INFORMATION/REFERENCES/ BIBLIOGRAPHY

1. Chuprina, R., 2020. *AI and Machine Learning in Manufacturing: The Complete Guide*. [online] SDP Group. Available at: <https://spd.group/machine-learning/ai-and-ml-in-manufacturing-industry/> [Accessed 10 March 2021].

2. IBM. 2020. *What is Machine Learning?*. [online] Available at: <https://www.ibm.com/cloud/learn/machine-learning> [Accessed 17 March 2021].

3. IBM. 2020. *What is Speech Recognition?*. [online] Available at: <https://www.ibm.com/cloud/learn/speech-recognition> [Accessed 16 March 2021].

4. Klipfolio. n.d. *What is a data dashboard?*. [online] Available at: <https://www.klipfolio.com/resources/articles/what-is-data-dashboard> [Accessed 15 March 2021].

5. Maksymenko, S., 2021. *AI in Manufacturing Use Cases and Trends in 2021*. [online] MobiDev. Available at: <https://mobidev.biz/blog/ai-machine-learning-in-manufacturing> [Accessed 12 March 2021].

6. Master's in Data Science. 2020. *What is Data Analytics?*. [online] Available at: <https://www.mastersindatascience.org/learning/what-is-data-analytics/> [Accessed 20 March 2021].

7. Mayo, M., 2018. *Frameworks for Approaching the Machine Learning Process*. [online] KDnuggets. Available at: <https://www.kdnuggets.com/2018/05/general-approaches-machine-learning-process.html> [Accessed 22 March 2021].

8. Mitchell, T., n.d. *The 4 Dashboards Every Manufacturer Needs to Optimize Production*. [online] SMART FACTORY. Available at: <https://elisasmartfactory.com/manufacturing-dashboards-that-help-optimize-production/> [Accessed 10 March 2021].

9. Noonpakdee, W., Khunkornsiri, T., Phothichai, A. and Danaisawat, K., 2018. A framework for analyzing and developing dashboard templates for small and medium enterprises. *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*, [online] pp.479-483. Available at: <https://ieeexplore.ieee.org/document/8387148> [Accessed 4 March 2021].

10. Rodriguez, R., 2020. *The 7 Key Steps To Build Your Machine Learning Model*. [online] Analytics India Magazine. Available at: <https://analyticsindiamag.com/the-7-key-steps-to-build-your-machine-learning-model/> [Accessed 24 March 2021].

11. Rojko, A., 2017. Industry 4.0 Concept: Background and Overview. *International Journal of Interactive Mobile Technologies (iJIM)*, [online] 11(5), pp.77-90. Available at: <https://www.researchgate.net/publication/318667421_Industry_40_Concept_Background_and_Overview> [Accessed 4 March 2021].

12. Saxena, A. and Goebel, K., 2008. *Turbofan Engine Degradation Simulation Data Set*, NASA Ames Prognostics Data Repository Available at: <http://ti.arc.nasa.gov/project/prognostic-data-repository> [Accessed 23 March 2021]

13. Stedman, C., 2020. *Data Analytics (DA)*. [online] SearchDataManagement. Available at: <https://searchdatamanagement.techtarget.com/definition/data-analytics> [Accessed 18 March 2021].

14. Subrahmanyam, K., Ketha, T., Balakrishna, S. and Kumar, T., 2018. Development of Research & Development Dashboard For an University. *International Journal of Engineering & Technology*, [online] 7(2.32), pp.60-63. Available at: <https://www.researchgate.net/publication/325881968_Development_of_Research_Development_Dashboard_For_an_University> [Accessed 4 March 2021].

15. Yufeng, G., 2017. *The 7 Steps of Machine Learning*. [online] towards data science. Available at: <https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e> [Accessed 21 March 2021].

RESOURCES - STATEMENT OF HARDWARE/SOFTWARE REQUIRED

Visual Studio: It is an Integrated Development Environment (IDE) that allows user to edit, debug and build code. In this project, it is used to implement a machine learning model into the dashboard as the dashboard is developed in C# using Visual Studio.

ML.NET: It is an open-source machine learning framework for .NET developer platform. It is used to build, train, and evaluate a machine learning model.

RapidAPI: It is the leading API marketplace. In this project. It is used to search for suitable Speech Recognition API

REPORT STRUCTURE

Authorship declaration

Acknowledgements

Abstract

TOC

Chapter 1 – Introduction

Chapter 2 – Analysis / Literature Review

Chapter 3 – Synthesis / Requirements specification

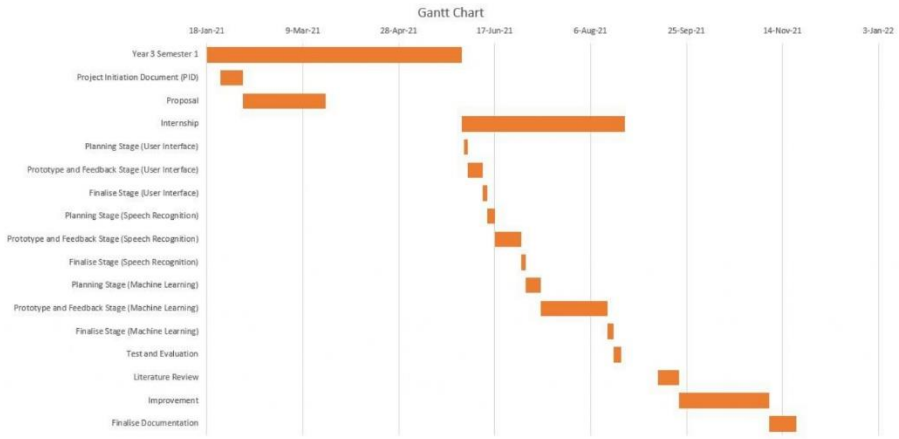
Chapter 4 – Testing

Chapter 5 – Evaluation and conclusions

References and Bibliography

Appendices

GANTT CHART



22

Appendix B: Proposal Review Form

UOW MALAYSIA KDU PENANG UNIVERSITY COLLEGE

PART OF THE UNIVERSITY OF WOLLONGONG AUSTRALIA GLOBAL NETWORK

CCP3012-3026 INDIVIDUAL PROJECT

PROJECT PROPOSAL REVIEW

Marking Scheme (100 Marks, Weighted @ 20%)

Student Name: Lim Khai Sian	**Student ID:** 0201576
Supervisor: Dr. Joshua Thomas	**2nd Marker:** Ng Fong Chiu
Date of Project Proposal Review: 01/4/2021	

Review Outcomes:

The topic is appropriate to the student's programme — Yes / No

The project contains sufficient practical work using computing skills relevant to the programme. — Yes / No

The proposal (select one)…

- is accepted without changes
- needs the changes listed overleaf. ✓
- cannot be made satisfactory and a new topic is required.

Project Type (select one):

Application Project ✓ | Investigative Project

Ethics Form (select one):

- Form B has been reviewed. ✓
- Form B is required but has not been provided.
- Form A is required and has already been submitted to the Project supervisor
- Form A is required and has not yet been submitted.
- Other (explain)

Risk Assessment (select one):

- Reviewed ✓
- Required but not provided
- Not Required

Signatures: **Student** 

Supervisor 

Second Marker Ng Fong Chiu

Page **1** of **2**

Changes To Proposed Aims/Objectives
Objectives: - to add in research area, predictive analytic system and speech recognition.
Changes To Project Plan
Nil.
Resource Issues
Nil.
Other Comments
Key concept - Add in predictive analytic and speech recognition. Proposed work - need to include who is your users, what is the functionality of ssyte, what are the features available in the system and how do you test the system. Citation - Wrong format.

Appendix C: Ethical Form

UOW MALAYSIA KDU PENANG UNIVERSITY COLLEGE
PART OF THE UNIVERSITY OF WOLLONGONG AUSTRALIA GLOBAL NETWORK

UoW Malaysia KDU Penang University College Research Ethics Registration and Approval Form

Ethical category:	
Red	☐
Blue	☐
Green	☑

Section One: Registration ***[To be completed by student]***

Title of research project/dissertation	Implementing Machine Learning into Dashboard for Data Analytics
Researcher's name	Lim Khai Sian
Programme of study	Bachelor of Computer Science (Hons)
Academic Year	2021
Module code	CCP3012
Supervisor's name	Dr. Joshua Thomas
Second marker's name	Ng Fong Chiu
Start Date of Project	18-January-2021

Brief outline of research topic:

The objective of this project is to implement Machine Learning model in the Dashboard to perform Predictive Maintenance to enhance the Overall Equipment Effectiveness. It also improves the User Interface and Speech Recognition as additional tasks.

Short description of proposed research methods including identification of participants:

Effective from: 29 May 2017 Revision: 00

UOW MALAYSIA KDU PENANG UNIVERSITY COLLEGE

PART OF THE UNIVERSITY OF WOLLONGONG AUSTRALIA GLOBAL NETWORK

UoW Malaysia KDU Penang University College Research Ethics Registration and Approval Form

Ethical considerations in the research project	YES	NO
1. Does your research involve an external organisation or partner?	☐	☑
2. Does your research involve human participants?	☐	☑
3. If yes to Q.2, will you inform the participants about the research?	☐	☑
4. Will you obtain their consent using the standard consent form?	☐	☑
5. Is any deception involved?	☐	☑
6. Do any participants constitute a 'vulnerable group'?	☐	☑
7. Will the research involve the following information?		
Commercially sensitive	☐	☑
Personally sensitive	☐	☑
Politically sensitive	☐	☑
Legally sensitive	☐	☑
8. Is the research likely to have any significant environmental impacts?	☐	☑
9. Are there likely to be any risks for the participants in your research?	☐	☑
10. Are there likely to be any risks for you in conducting the research?	☐	☑
11. If yes [to 5, 6, 7, 8, 9 or 10 above] have you identified steps to address the issues and mitigate any risks to participants, yourself or the environment?	☐	☐

Statement to explain how any issues identified above will be addressed and what steps will be taken to mitigate such risks or adverse impacts

Ethical category of research project *[Please tick as appropriate]*

Red	☐	vulnerable participants; human tissue; sensitive data; risks to participants & researchers etc.
Blue	☐	human participants requiring informed consent; commercially sensitive information etc.
Green	☑	no participants involved; secondary data only; no sensitive data

UOW MALAYSIA KDU PENANG UNIVERSITY COLLEGE

PART OF THE UNIVERSITY OF WOLLONGONG AUSTRALIA GLOBAL NETWORK

UoW Malaysia KDU Penang University College Research Ethics Registration and Approval Form

I have read the University College Ethics Policy and Procedures and confirm that the answers I have given above are correct. Where issues arise under items 5, 6, 7, 8, 9 or 10 [above] I have described in writing how I intend to approach these issues in the research.

Researcher's signature: _____

Date: 16/4/2021

Section 1 Ethics Registration to be submitted to Principal Supervisor or Module Tutor and allocated to a reviewer as follows:
Green risk - may be approved by Supervisor
Blue risk - to be submitted for approval by one independent reviewer (second marker)
Red risk - to be submitted for approval by two independent members of Research Committee

Section Two: Approval

Supervisor/Module Tutor's name confirming ethical risk status	Dr. Joshua Thomas

Ethical approval *[Please tick as appropriate]*

Green - Ethical approval is given without conditions (supervisor may approve).	☑
Blue – (to be approved by one independent reviewer) Ethical approval is given with the following conditions:	
* Information to be provided to all participants.	☐
* Participant consent to be obtained in accordance with the University College procedures.	☐
* Data to be stored and destroyed securely in accordance with University College guidelines.	☐
* Adherence to Data Protection Act.	☐
* Anonymity to be offered to participants.	☐
* Commercial confidentiality to be provided to organisations(s).	☐
* Software vulnerabilities, exploits, hazardous software etc. not to be published without prior 'responsible disclosure' to the affected supplier.	☐
* Other (please state):	☐
Red - Project is referred to Research Committee for approval by two independent reviewers.	☐

Name & role of reviewer 1: Dr. J. Joshua Thomas

Signature: _____

Date: 16/4/2021

Name & role of reviewer 2: Ng Fong Chiu

Signature: Ng Fong Chiu

Date: 15.4.2021

Outcome of Review:
Approved, proceed with the research work.

Effective from: 29 May 2017 Revision: 00

Appendix D: Turnitin Result

IP2

ORIGINALITY REPORT

10%	7%	5%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	towardsdatascience.com Internet Source	1%
2	www.journaltocs.ac.uk Internet Source	1%
3	Submitted to University of Technology, Sydney Student Paper	<1%
4	www.ncbi.nlm.nih.gov Internet Source	<1%
5	trepo.tuni.fi Internet Source	<1%
6	doctorpenguin.com Internet Source	<1%
7	turcomat.org Internet Source	<1%
8	www.mdpi.com Internet Source	<1%
9	www.ibm.com Internet Source	<1%